World Wide Web
https://doi.org/10.1007/s11280-021-00930-2

Exploiting intra- and inter-session dependencies for session-based recommendations

Nan Wang[1] · Shoujin Wang[1] · Yan Wang[1] · Quan Z. Sheng[1] · Mehmet A. Orgun[1]

Received: 2 January 2021 / Revised: 2 June 2021 / Accepted: 12 July 2021 /
Published online: 06 August 2021

Abstract

Session-based recommender systems (SBRSs) aim at predicting the next item via learning the dynamic and short-term preferences of users. Most of the existing SBRSs usually make predictions based on the intra-session dependencies embedded in session information only, ignoring more complex inter-session dependencies and other available side information (e.g., item attributes, users), which in turn greatly limits the improvement of the recommendation accuracy. In order to effectively extract both intra- and inter-session dependencies from not only the session information but also the side information, to further improve the accuracy of next-item recommendations, we propose a novel *hypergraph learning (HL)* framework. The HL framework mainly contains three modules, *i.e.*, a hypergraph construction module, a hypergraph learning module, and a next-item prediction module. The hypergraph construction module constructs a hypergraph to connect the users, items and item attributes together in a unified way. Then, the hypergraph learning module learns the informative latent representation for each item by extracting both intra- and inter-session dependencies embedded in the constructed hypergraph. Also, a latent representation for each user is learned. After that, the learned latent representations are fed into the next-item prediction module for next-item recommendations. We conduct extensive experiments on two real-world datasets. The experimental results show that our HL framework outperforms the state-of-the-art approaches.

Keywords Next-item recommendation · Session-based recommendations · Hypergraph learning

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2020*
Guest Editors: Hua Wang, Zhisheng Huang, and Wouter Beek

✉ Yan Wang
yan.wang@mq.edu.au

Extended author information available on the last page of the article.

1 Introduction

In the current information era, a huge amount of information leads to the critical issue of information overload. Recommender systems (RSs) have been proposed to solve this challenge through information filtering, and thus can effectively provide suggestions for users' choices on a large number of items [7, 35]. Various approaches have been developed for RSs during the past few years, including collaborative filtering-based approaches, content-based approaches, and hybrid approaches [17, 23]. These approaches mainly focus on learning the long-term static preferences of users for recommendations [25–27], which have two major drawbacks: (1) long-term and static preferences of users are sometimes unavailable, and (2) the short-term and dynamic preference of users are usually ignored [28].

Session-based recommender systems (SBRSs) have emerged in recent years, which can alleviate these two drawbacks. SBRSs learn a user's short-term and dynamic preferences based on the items interacted by the user within sessions. In general, there are four basic types of SBRSs, including (1) Markov chain based SBRSs, (2) factorization based SBRSs, (3) recurrent neural network (RNN) based SBRSs, and (4) graph neural network (GNN) based SBRSs. Markov chain based SBRSs [10, 16] and factorization based SBRSs [2, 12] recommend the next item based on the previous one, which can capture the *first-order dependencies* (i.e., the dependencies between any two adjacent items within a session). RNN-based SBRSs [5, 6, 13] can further capture the *high-order dependencies* (i.e., the cascaded dependencies between multiple items within a session) across multiple items by modelling each session as a sequence for next-item recommendations. To further capture the complex transitions over items within each session, GNN-based SBRSs [20, 21, 31] have been proposed. GNN-based SBRSs firstly map each session to a subgraph separately, and then takes the subgraph as the input of a GNN to further capture the dependencies over nodes (i.e., items) in the subgraph for next-item recommendations.

However, most of the existing GNN-based SBRSs have two shortcomings: (1) they often utilize the limited session information only for next-item recommendations, while ignoring other important side information including item attributes and user IDs, which can greatly complement the limited session information; and (2) they usually model the intra-session dependencies (i.e., the dependencies between items within one session) and ignoring the complex inter-session dependencies (i.e., the dependencies between items across multiple sessions). Both shortcomings greatly reduce the performance of next-item recommendations. Accordingly, to address these shortcomings and further improve the performance of SBRSs, there are two challenges still need to be addressed:

- **CH1**: How to effectively make full use of the available side information including item attribute information and user information for improving the accuracy of next-item recommendations?
- **CH2**: How to effectively extract both intra- and inter-session dependencies to further improve next-item recommendations?

To address the aforementioned two challenges, in this paper, we propose a novel hypergraph learning framework. Specifically, to address **CH1**, we introduce item attribute information and user information simultaneously to enrich the information for next-item recommendations. To effectively incorporate such side information, we construct a *hypergraph* to connect the items, item attribute values and users together by integrating the session-based *item-item graph*, the *item-item attribute value graph* and *user-session graph* into a unified hypergraph. In such a case, items are connected not only by sessions, but also by their

shared attribute values and the joint users (see Figure 1). A *hypergraph* is a generalization of graphs in which the edges are arbitrary non-empty subsets of the vertex set. To address **CH2**, we propose a hypergraph learning framework to learn both the intra-session dependencies (i.e., the dependencies between items within the same session) and inter-session dependencies (i.e., the dependencies between items from different sessions) with a focus on learning inter-session dependencies. For intra-session dependencies, we take each subgraph built on each session into gated graph neural networks to learn the dependencies between items within sessions. For inter-session dependencies, we devise three types of aggregation operations to learn the inter-session dependencies to absorb information from other sessions to benefit the next-item recommendation in the current session. Each aggregation is based on one type of hyperedges in the hypergraph.

The first aggregation operation, called *user aggregation*, is utilized to learn the dependencies between items that are purchased by the same users. Here the hyperedges connecting each user and all the sessions from this user serve as the bridges to connecting items from different sessions from the same user. The second aggregation operation, called *item aggregation*, is utilized to learn the dependencies between items from different sessions. The hyperedge connecting a given item and all of its co-occurred items (i.e., those items that concurred with the given item in the same sessions) forms the bridge to connect items from different sessions. The third aggregation operation, called *attribute aggregation*, is utilized to learn the dependencies between items which have shared attribute values. Here the hyperedge connecting each item and all of its attribute values serve as the bridge to connect different items sharing attribute values.

In addition, we devise *session aggregation* to learn a given user's preference towards items. Specifically, session aggregation aggregates all the historical sessions of a user to represent her/his preference. Session aggregation is based on hyperedges connecting each users and all of her/his historical sessions.

The main contributions of our framework are summarized as follows:

- We propose a hypergraph to encode not only the session information, but also the user-session information and the item-attribute information. In this way, all the items and users in a given data set are comprehensively connected together in a unified way. This greatly enriches the information for SBRSs.
- We propose a hypergraph learning framework to effectively learn not only the intra-session dependencies, but also the widely existed inter-session dependencies for more accurate next-item recommendations. In addition, each user's preference towards items

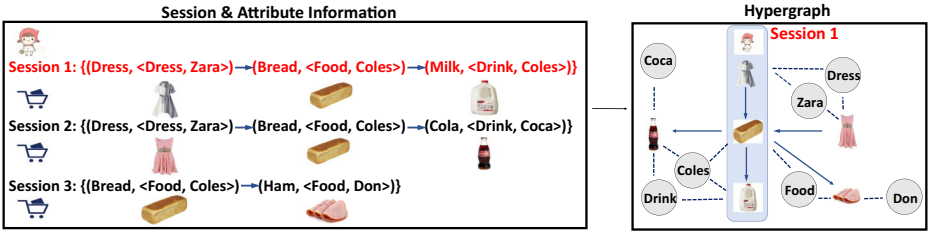

Figure 1 An example of a Hypergraph constructed on user transaction information and item attribute information. The hyperedges connecting user and session sets (i.e., the sessions purchased by one user) indicate the user-session relations, the arrowed edges connecting two items indicate the sequential relations between items from a given session, and the dotted edges connecting item and item attribute values mean the item-attribute relations

has also been well captured by the devised session aggregation to further improve the recommendation accuracy.
– The extensive experiments have been conducted on two real-world datasets, and the experimental results show that our HL framework outperforms the state-of-the-art methods.

The rest of this paper is organized as follows. we provide a comprehensive discussions on related work in Section 2. Section 3 presents the problem statement. We present our proposed hypergraph learning (HL) framework in Section 4. Datasets, evaluation metrics, comparison methods are introduced in Section 5. Finally, we conclude this paper in Section 6.

2 Related work

In this section, we review several related studies on SBRSs and hypergraph learning, where we divided SBRSs into two categories: (1) conventional SBRSs, including Markov chain based SBRSs and factorization based SBRSs, and (2) deep learning based SBRSs, including RNN-based SBRSs and GNN-based SBRSs.

2.1 Conventional SBRSs

Markov chain based SBRSs and factorization based SBRSs are two typical types of conventional SBRSs. Markov chains based SBRSs mainly employ Markov chains to model the users' behaviour. Specifically, they learn the transition probabilities to capture the first-order dependencies between two adjacent items, and then recommend the next item based on this dependencies [2]. However, due to the property of Markov chains, this method only captures the first-order dependencies between two adjacent items while ignoring the high-order dependencies across multiple items. Factorization based SBRSs factorize the item transition matrix into latent representations of users and items, which are subsequently used to predict the next-item in which a given user may be interested [1]. However, because the users' preference is sometimes unavailable, factorization based SBRSs do not always work. Further, Rendle et al. [16] firstly proposed Factorized Personalized Markov Chains (FPMC) model for next item recommendation, which factorizes the user-item matrix to learn both the general taste of a user and transitions over items by Markov chains method. However, as the combination of Markov chains based SBRSs and factorization based SBRSs, FPMC still cannot capture higher-order dependencies.

2.2 Deep learning based SBRSs

To capture the high-order dependencies, Hidasi et al. [6] proposed an RNN-based approach for SBRSs, called GRU4Rec, which employs gated recurrent units (GRUs) as the basic cells of an RNN to model the rigid order dependencies among items in each session. Hidasi et al. [5] further introduced a novel ranking loss function to improve the performance of GRU4Rec. However, RNN-based SBRSs usually rely on the rigid order assumption among items within a session, leading to two drawbacks: (1) the rigid order is sometimes meaningless, since a user's interactions with items in a session may not always follow a rigid order; and (2) the complex transitions between distant items within a session cannot be effectively captured with a single-way RNN.

To capture the complex transitions over items within the entire session, Wu et al. [20] proposed a GNN-based method for session-based recommendation, called SR-GNN. In SR-GNN, first, a directed graph is built for each session, and then GNN are employed on the graph to capture the complex transitions over items within the session. Xu et al. [31] further proposed a graph contextualized self-attention network (GC-SAN), which introduces a self-attention mechanism into GNN-based SBRSs to learn more attentive and informative session representations for more accurate next-item recommendations. Wu et al. [21] further proposed a personalizing graph neural networks with an attention mechanism (A-PGNN), which employes a personalizing graph neural network to capture complex transitions in users' session sequence and employes a dot-product attention mechanism to explicitly model the effect of historical sessions on the current session. Yu et al. [32] proposed a target attentive graph neural network (TAGNN) to activate each user's different interests with respect to varied target items for next item recommendations. Qiu et al. [18] proposed a full graph neural network (FGNN) to learn the inherent order of item transition pattern and proposed a weighted graph attention layer (WGAT) network to learn different weights for different neighbors, which aims at improving the accuracy of next-item recommendations. However, all the GNN-based methods mentioned above mainly capture the local dependencies within one session and neglect the cross-session global dependencies.

A few studies tried to model the global dependencies across multiple sessions for accurate session-based recommendations. Wang et al. [30] proposed a global context enhanced graph neural network (GCE-GNN) to learn the session level latent representations of items within one session, and learn the global level latent representations of items over all sessions. However, there are two shortcomings of GCE-GNN: (1) it learns the latent representations for items by modeling pairwise item-transitions, while ignoring the more complex relationships of item-transitions (e.g., list-wise item-transitions); and (2) it fails to explicitly capture the global dependencies based on item attribute information, leading to information loss. To alleviate these shortcomings of SBRSs, Wang et al. [29] proposed a heterogeneous mixed graph learning (HMGL) model to learn an informative representation for each item over the heterogeneous graph by effectively modelling both local and global dependencies for next-item recommendations. However, this work does not consider the significant role of users in connecting the different but relevant sessions as well as items. Such a mission the user information leads to information loss and subsequently deteriorates recommendation accuracy.

Most of the existing SBRS methods [21, 31, 32] mentioned above usually capture intra-session dependencies and omit to consider the inter-session dependencies. This omission leads to both information loss and low recommendation accuracy. In this paper, we propose an HL framework based on a previous study [29]. In addition to capturing intra-session dependencies, our proposed HL framework can further capture the inter-session dependencies by taking the items occurring in different sessions, users shared by different sessions, and attribute values shared by the items from different sessions as the bridges to connect multiple sessions. Therefore, our proposed framework can capture the more complex dependencies, and as a result get better recommendation results.

2.3 Hypergraph learning

Traditional graph learning methods focus on pairwise relationships between objects [11, 24]. To capture more complex relationships, Zhou et al. [34] generalized the spectral clustering graphs into hypergraphs, based on which they further developed an algorithm for

hypergraph embedding and transductive classification. Huang et al. [8] employed the hypergraph learning in video object segmentation, where they formulated the task of extracting prominent objects into hypergraph cut. To learn the weights of hyperedges, Gao et al. [4] introduced a l_2 regularizer. With the development of deep learning, Feng et al. [3] firstly proposed a hypergraph neural network (HGNN) framework, where they employed hypergraph convolution operation to learn representation of high-order data correlation in a hypergraph structure. Jiang et al. [9] further separated a dynamic hypergraph neural network framework (DHGNN) into dynamic hypergraph construction (DHG) and hypegraph convolution (HGC) operations to capture both the inherent and dynamic hypergraph structures.

3 Problem statement

Given a user-transaction dataset, $\mathcal{U} = \{u_1, u_2, ..., u_{|u|}\}$ is the user set consisting of all unique users, $|u|$ is the number of users in the dataset. $\mathcal{V}^I = \{v_1, v_2, ..., v_{|\mathcal{V}^I|}\}$ is the item set consisting of all the unique items, $|\mathcal{V}^I|$ is the number of items. Let $\mathcal{D} = \{S_1, S_2, ..., S_{|u|}\}$ be the set of all user-transactions in this dataset. Each user-transaction $S_i = \{s_{i,1}, s_{i,2}, ..., s_{i,|s|}\}$ is the set of sessions that are associated with a user u_i. Each session $s_{i,j} = \{v_{i,j,1}, v_{i,j,2}, ..., v_{i,j,|s_{i,j}|}\}$ consists of all the unique items sequentially interacted by the corresponding user u_i in her/his j^th session. $\mathcal{V}^{\mathcal{A}} = \{a_1, a_2, ..., a_{|\mathcal{V}^A|}\}$ is the set of all item attribute values in the dataset. $\mathcal{A} = \{A_1, A_2, ..., A_{|\mathcal{V}^I|}\}$ is the set of attributes for all items, and $A_h = \{a_{h,1}, a_{h,2}, ..., a_{h,|A_h|}\}$ the set of attribute values associated with item v_h. The task of our work is to recommend the next item based on the purchased items in the current session. Formally, given the session $s_{i,j}$ of user u_i, take v_l ($v_l \in s_{i,j}$) as the target item, $C_{v_l}^{s_i} = \{v_1, v_2, ..., v_{l-1}\}$ is the corresponding session context consisting of all the items that have occurred prior to item v_l in session $s_{i,j}$, while the corresponding item attribute value set $C_{v_l}^{a} = \{\mathcal{A}_{v_1}, \mathcal{A}_{v_2}, ..., \mathcal{A}_{v_{l-1}}\}$ forms the corresponding attribute context. Hence, given a context set $C_{v_l} = [C_{v_l}^{s_i}, C_{v_l}^{a}]$, the task of our work is to recommend the l^{th} item v_l in session $s_{i,j}$.

4 Hypergraph learning framework

In this section, we present our proposed hypergraph learning (HL) framework. In particular, first, the HL framework constructs a hypergraph to connect the items, users and item attribute values by integrating the session-based *item-item graph*, the *item-item attribute value graph*, and *user-session graph* into a unified hypergraph. Then, the hypergraph learning model is built to learn the latent representations of users and items for next-item recommendations from the hypergraph. Accordingly, as shown in Figure 2, the HL framework consists of three components: the *hypergraph construction* module, the *hypergraph learning* module, and the *next-item prediction* module.

4.1 Preliminaries

Graphs can only indicate pairwise relationships. In contrast, hypergraphs preserve the multi-wise relationships, and thus, are a natural choice for the modeling complex relationships. We define a hypergraph as follows:

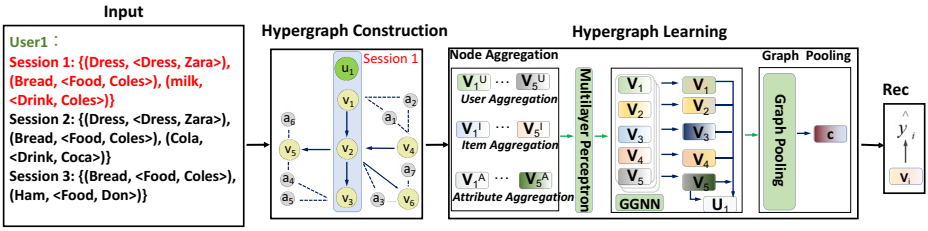

Figure 2 The workflow of our proposed HL framework. First, we integrate session information, user information and item attribute information into a hypergraph. Then, the hypergraph learning module learns the intra- and inter-session dependencies between items. Specifically, three node level aggregations are devised to learn three types of inter-session dependencies, while the node level updating aims to learn the intra-session dependencies and graph pooling learns a unified representation for each session context. Finally, we predict the next item by taking the session context representation as the input

Definition 1 (Hypergraph) A hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ on a finite set of vertices (or nodes) $\mathcal{V} = \{v_i : i \in \|n\|\}$, with a family of hyperedges $\mathcal{E} = \{e_j : j \in \|p\|\}$, where each hyperedge is a non-empty subset of $\mathcal{V}$, such that $\cup_{j \in \|p\|} e_j = \mathcal{V}$.

In a hypergraph, a hyperedge links one or more vertices.

4.2 Hypergraph construction

A traditional graph can only represent the pairwise relationships between any two nodes. However, in the real world, there will be much more complex relationships like list-wise relationships than pairwise relationships. Hypergraphs can preserve these list-wise relationships. For better recommending the next item, we construct the user sessions and item-attribute values into a unified hypergraph. Firstly, we model the users, items and item attribute values as three types of nodes in the graph. Formally, $\mathcal{V} = \mathcal{U} \cup \mathcal{V}^{\mathcal{I}} \cup \mathcal{V}^{\mathcal{A}}$ be the node set in the graph, where $\mathcal{U}$, $\mathcal{V}^{\mathcal{I}}$ and $\mathcal{V}^{\mathcal{A}}$ correspond to the node set of users, items and item attribute values respectively.

Secondly, we model user-session relations, the item-item relations, and the item-attribute value relations as three types of hyperedges in the hypergraph. Formally, $\mathcal{E} = \mathcal{E}^{\mathcal{U}} \cup \mathcal{E}^{\mathcal{I}} \cup \mathcal{E}^{\mathcal{A}}$ constitute the hyperedge set. The hyperedge $e_u \in \mathcal{E}^{\mathcal{U}}$ represents the relation that session $s_u = \{v_{u,1}, v_{u,2}, ..., v_{u,|s_u|}\}$ purchased by user u. In other words, the several items $v_{u,1}, v_{u,2}, ..., v_{u,|s_u|}$ contents of items purchased within one session s_u by user u are linked together by this hyperedge e_u. Hyperedge $e^I_{i,j} \in \mathcal{E}^{\mathcal{I}}$ represents the item v_j occurs after item v_i within one session. The hyperedge $e^A_i \in \mathcal{E}^{\mathcal{A}}$ represents that item v_i has the attribute value set $A_i = \{a_{i,1}, a_{i,2}, ..., a_{i,|v_i|}\}$, and the item v_i and its corresponding attribute values $a_{i,1}, a_{i,2}, ..., a_{i,|v_i|}$ are linked together by this hyperedge.

As a result, the unified hypergraph connect the users, items and item attribute values by integrating user-session graph, session-based item-item graph and item-item attribute graph that are built on the aforementioned three types of hyperedges respectively together. Such a complex hypergraph is powerful to model the complex high-order relations among different types of objects (i.e., users, items and item attribute values).

4.3 Hypergraph learning

HL mainly has three components: (1) node level aggregation, (2) node level updating and (3) graph level learning. In the node level aggregation part, there are three types of nodes in

the hypergraph, including user nodes, item nodes and item attribute nodes, and we mainly learn the latent representations for user nodes and items nodes in the hypergraph. For user nodes, we learn the latent representations of users via session aggregation, which is utilized to aggregate information from a user's historical sessions and the current sessions to learn his/her preference representation. For item nodes, we learn the latent representations based on three different types of hyperedges. Therefore, three aggregation operations are introduced to respectively process the hypergraph. After we aggregate information for user nodes and item nodes, we update the latent representations of items and users by graph gated neural netwoks (GGNN) based on the constructed hypergraph. Finally, we learn a graph level latent representation for each session sub-hypergrap using graph pooling. We discuss each model component in detail below.

4.3.1 Node level aggregation in hypergraph

In this part, we aim to aggregate information from various hyperedges for user nodes $u \in \mathcal{U}$ and item nodes $v \in \mathcal{V}^I$ in hypergraph. Firstly, we map each user node $u \in \mathcal{U}$ and item node $v \in \mathcal{V}^I$ into a unified low-dimension latent space to obtain the initial representation $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, where d denotes the dimension of the representation. Then, for each item node $v \in \mathcal{V}^I$, we employ three aggregation operations to learn three types of latent representations $\mathbf{v}_i^U, \mathbf{v}_i^I, \mathbf{v}_i^A$ by absorbing the information from three types of neighborhood nodes of v respectively in $\mathcal{G}$. Each type of latent representation can be seen as a sub-representation of the item from a particular perspective. Similarly, for each user node $u \in \mathcal{U}$, we employ one aggregation operation to update its latent representation by learning the user preference from her/his historical and current sessions.

The first aggregation operation, denoted as *user aggregation*, is utilized to learn the user-based latent representation $\mathbf{v}_i^U \in \mathbb{R}^d$ of item v_i. The aim of *user aggregation* is to aggregate information for items from the joint users. We mathematically represent this aggregation:

$$\mathbf{v}_i^U = Aggre_{user}\{\mathbf{v}_j, \forall v_j \in N_{user}(v_i)\}, \tag{1}$$

$N_{user}(v_i)$ is the user-based neighborhood set of item node v_i, which consists of item nodes having been interacted by the same user who interacted with item node v_i. We specify the *user aggregation* as follows:

$$Aggre_{user}(\mathbf{v}_i) = \mathbf{A}_i^u[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,|N_{user}(v_i)|}]^T, \tag{2}$$

where $[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,|N_{user}(v_i)|}]$ is the list of latent representations of item nodes $v_{i,1}, v_{i,2}, ..., v_{i,|N_{user}(v_i)|} \in N_{user}(v_i)$. The weight matrix $\mathbf{A}_i^u$ shows the importance of each item node in $N_{user}(v_i)$ for item node v_i, which is calculated as follows:

$$\mathbf{A}_{i,j}^u = \frac{\left\| \mathcal{V}_{i,j}^{e_u} \right\|_2}{\left\| \mathcal{V}_i^{e_u} \right\|_2 + \left\| \mathcal{V}_j^{e_u} \right\|_2} \tag{3}$$

where $\mathcal{V}_{i,j}^{e_u} = \{u|(v_i, v_j, u) \in e_u\}$ consists of the user nodes u that have hyperedges e_u with item node v_i and v_j simultaneously. $\mathcal{V}_i^{e_u} = \{u|(v_i, u) \in e_u\}$ consists of the user nodes u that have hyperedges e_u with item node v_i. $\|\cdot\|_2$ is the L^2 norm.

The second aggregation operation, denoted as *item aggregation*, is utilized to learn the inter-session-based latent representation $\mathbf{v}_i^I \in \mathbb{R}^d$ of item v_i. The aim of *item aggregation* is to aggregate information from other items which occurred in the same session as item v_i.

$$\mathbf{v}_i^I = Aggre_{sess}\{\mathbf{v}_j, \forall v_j \in N_{sess}(v_i)\}, \tag{4}$$

$N_{sess}(v_i)$ is the session-based neighborhood set of item node v_i, which contains all the item nodes in the hypergraph that occur in the same session with item node v_i. The *item aggregation* is specified as:

$$Aggre_{items}(\mathbf{v}_i) = \mathbf{A}_i^s[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,|sess|}]^T, \tag{5}$$

where $[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,|sess|}]$ is the list of latent representations of item nodes $v_{i,1}, v_{i,2}, ..., v_{i,|sess|} \in N_{sess}(v_i)$. The weight matrix $\mathbf{A}_i^s$ shows the importance of each item node in $N_{sess}(v_i)$ for item node v_i, which is calculated as below:

$$\mathbf{A}_i^s = \frac{\left\| \mathcal{V}_{i,j}^{e^I} \right\|_2}{\left\| \mathcal{V}_i^{e^I} \right\|_2 + \left\| \mathcal{V}_j^{e^I} \right\|_2} \tag{6}$$

where $\mathcal{V}_{i,j}^{e^I} = \{sess|(v_i, v_j, sess) \in e^I$ consists of the sessions $sess$ that have hyperedges e^I with item node v_i and v_j simultaneously. $\mathcal{V}_i^{e^I} = \{sess|(v_i, sess) \in e^I\}$ consists of the sessions $sess$ that have hyperedges e^I with item node v_i.

The third aggregation operation, denoted as *attribute aggregation*, is utilized to learn the attribute-based latent representation $\mathbf{v}_i^A \in \mathbb{R}^d$ of item v_i. The aim of *attribute aggregation* is to aggregate information from items, which have shared attribute values with a given item v_i. To mathematically represent this aggregation, we use the following function:

$$\mathbf{v}_i^A = Aggre_{attri}\{\mathbf{v}_j, \forall v_j \in N_{attri}(v_i))\}, \tag{7}$$

where $N_{attri}(v_i)$ is the attribute-based neighborhood set of item node v_i, which consists of item nodes sharing the same attribute values with item node v_i. $Aggre_{attri}$ is the item aggregation function based on attribute information. The aggregation function is specified below:

$$Aggre_{attri}(\mathbf{v}_i) = \mathbf{A}_i^A[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,|N_{attri}(v_i)|}]^T, \tag{8}$$

where $[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,i,|N_{attri}(v_i)|}]$ is the list of latent representations of item nodes $v_{i,1}, v_{i,2}, ..., v_{i,i,|N_{attri}(v_i)|} \in N_{attri}(v_i)$. The weight matrix $\mathbf{A}_i^A$ shows the importance of each item node in $N_{attri}(v_i)$ for item node v_i, which is calculated below:

$$A_i^A = \frac{\left\| \mathcal{V}_{i,j}^{e^A} \right\|_2}{\left\| \mathcal{V}_i^{e^A} \right\|_2 + \left\| \mathcal{V}_j^{e^A} \right\|_2} \tag{9}$$

where $\mathcal{V}_{i,j}^{e^A} = \{a|(v_i, v_j, a) \in e^A$ consists of the attribute values a that have hyperedges e^A with item node v_i and v_j simultaneously. $\mathcal{V}_i^{e^A} = \{a|(v_i, a) \in e^A\}$ consists of the attribute values a that have hyperedges e^A with item node v_i.

4.3.2 Node level updating in hypergraph

Once the three types of latent representations of a given item v_i are learned, they are combined together to form the final latent representation $\mathbf{v_i}$ of item node v_i via a standard MLP. The $\mathbf{v}_i^U$, $\mathbf{v}_i^I$, and $\mathbf{v}_i^A$ are concatenated before feeding into MLP. Formally, the latent representation is defined as

$$\mathbf{v}_i = \sigma(\mathbf{W} \cdot Concat[\mathbf{v}_i^U, \mathbf{v}_i^I, \mathbf{v}_i^A] + b), \tag{10}$$

where the activation function σ is specified as the activation function. Then, for each item node v_i, we employ a gate graph neural network (GGNN) to iteratively update its representation $\mathbf{v}_i$ by aggregating the information from other item nodes within the given session.

Therefore, the intra-session dependencies are learned and encoded into the item representations. We first aggregate information $\mathbf{a}_{s,i}^t$ from the intra-session-based neighborhoods of $v_{s,i}$ via the connection matrix $\mathbf{M}^s$:

$$\mathbf{a}_{s,i}^t = \mathbf{M}_{i,:}^s[\mathbf{v}_{s,1}^{t-1}, \mathbf{v}_{s,2}^{t-1}, ..., \mathbf{v}_{s,|s|}^{t-1}]^T\mathbf{H} + \mathbf{b}, \tag{11}$$

where $\mathbf{H} \in \mathbb{R}^{d\times d}$ is the weight matrix, $\mathbf{b}$ is the bias, $[\mathbf{v}_{s,1}^{t-1}, \mathbf{v}_{s,2}^{t-1}, ..., \mathbf{v}_{s,|s|}^{t-1}]$ is the list of latent representations of item nodes $v_{s,1}, v_{s,2}, ..., v_{s,|s|}$ in the intra-session-based neighborhood set of $v_{s,i}$ at $(t-1)^{th}$ iteration. The connection matrix $\mathbf{M}^s = [\mathbf{M}^{s,I}, \mathbf{M}^{s,O}] \in \mathbb{R}^{|s|\times 2|s|}$ is the concatenation of $\mathbf{M}^{s,I}$ and $\mathbf{M}^{s,O}$, $\mathbf{M}_{i,:}^s$ denotes the i^{th} row of $\mathbf{M}^s$ corresponding to v_i. The matrices $\mathbf{M}^{s,I}$ and $\mathbf{M}^{s,O}$ indicate the connections between two item nodes within session s, which is specified as below:

$$\mathbf{M}_{i,j}^{s,I} = \frac{|e_{i,j}^I|}{|D^I(v_i)|} \qquad \mathbf{M}_{i,j}^{s,O} = \frac{|e_{j,i}^I|}{|D^O(v_i)|} \tag{12}$$

where $e_{i,j}^I$ is the set of heyperedges e^I, which connect from item node v_i to item node v_j, $D^I(v_i)$ and $D^O(v_i)$ represent the indegree and outdegree of item node v_i respectively. In this way, the communications between item nodes within one session indicated by the frequencies of hyperedges are captured. We give an example of a user session $s : v_1 - v_2 - v_3 - v_4 - v_2 - v_1$, specially, in our dataset, we regard the order of the appearance of items in one session as the shopping order in one transaction. The corresponding connection matrix $\mathbf{M}_{s,i}$ in Figure 3.

We input the aggregated information $\mathbf{a}_{s,i}^t$ within one session and the latent representation $\mathbf{v}_{s,i}^{t-1}$ of v_i to update the latent representation $\widetilde{\mathbf{v}}_{s,i}^t$ in the t^{th} iteration by the equations below:

$$\mathbf{z}_{s,i}^t = \sigma(\mathbf{W}_z\mathbf{a}_{s,i}^t + \mathbf{U}_z\mathbf{v}_{s,i}^{t-1}), \tag{13}$$

$$\mathbf{r}_{s,i}^t = \sigma(\mathbf{W}_r\mathbf{a}_{s,i}^t + \mathbf{U}_r\mathbf{v}_{s,i}^{t-1}), \tag{14}$$

$$\widetilde{\mathbf{v}}_{s,i}^t = tanh(\mathbf{W}_o\mathbf{a}_{s,i}^t + \mathbf{U}_o(r_{s,i}^t \odot \mathbf{v}_{s,i}^t)), \tag{15}$$

$$\mathbf{v}_{s,i}^t = (1 - \mathbf{z}_{s,i}^t) \odot \mathbf{v}_{s,i}^{t-1} + \mathbf{z}_{s,i}^t \odot \widetilde{\mathbf{v}}_{s,i}^t, \tag{16}$$

where $\mathbf{r}_{s,i}^t$ represents the reset gate vector and $\mathbf{z}_{s,i}^t$ represents the update gate vector, which are determined by the aggregated intra-session information $\mathbf{a}_{s,i}^t$ in the t^{th} iteration. $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_o \in \mathbb{R}^{2d\times d}$ and $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_o \in \mathbb{R}^{d\times d}$ are learn-able weight matrices, activation

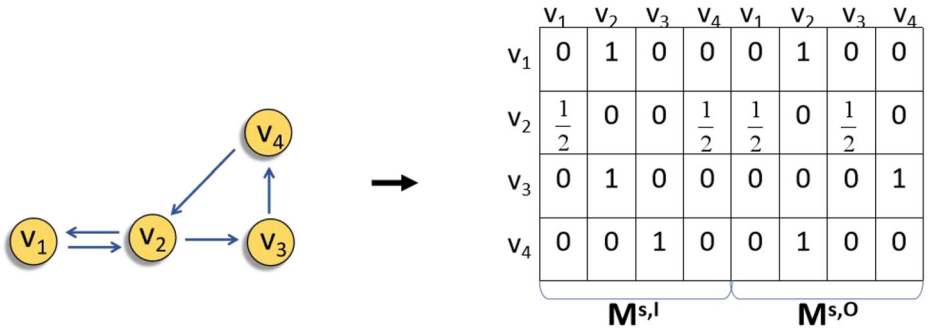

Figure 3 An example of a directed subgraph based on session s and the corresponding connection matrix $\mathbf{M}_i^s$

function $\sigma(\cdot)$ is specified as the sigmoid, the $\odot$ is the element-wise multiplication operation. The latent representation $\mathbf{v}_{s,i}^t$ of $v_{s,i}$ in the t^{th} iteration is decided by the update gate vector $\mathbf{z}_{s,i}^t$, the reset gate vector $\mathbf{r}_{s,i}^t$, the latent representation $\mathbf{v}_{s,i}^{t-1}$ of $v_{s,i}$ in the $(t-1)^{th}$ iteration and aggregated intra-session information $\mathbf{a}_{s,i}^t$.

After we obtain the latent representation of each item node, firstly, we devise *session aggregation* to aggregate the information from historical sessions and current session of user u_i, which is specified as:

$$Aggre^u(\mathbf{u}_i) = \mathbf{A}_i^u[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,|N(u_i)|}]^T, \tag{17}$$

where $N(u_i)$ is the neighborhood set of user node u_i, which consists of all item nodes in both historical and current sessions of user u_i, $[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,|N(u_i)|}]$ is the list of latent representations of item nodes $v_{i,1}, v_{i,2}, ..., v_{i,|N(u_i)|} \in N(u_i)$. The weight matrix $\mathbf{A}_i^A$ shows the importance of each item node in $N(u_i)$ for user node u_i, which is calculated using:

$$A_i^u = \frac{\left\| \mathcal{V}_{i,j}^{e_u} \right\|_2}{\left\| \mathcal{V}_{i,j} \right\|_2} \tag{18}$$

where $\mathcal{V}_{i,j}^{e_u} = \{v_j | (v_i, v_j, u) \in e_u\}$ is the number of item nodes v_j that have connected with item node v_i and user node u with hyperedges e_u. Then, we feed the aggregated information $Aggre^u(\mathbf{u}_i)$ and the initial latent representation $\mathbf{u}_i$ into a MLP, and obtain the final latent representation $\mathbf{u}_i^{Final}$ of user node u_i below:

$$\mathbf{u}_i^{Final} = \sigma(\mathbf{W} \cdot [Aggre^u(\mathbf{u}_i), \mathbf{u}_i]), \tag{19}$$

where the activation function σ is specified as the sigmoid function, and $\mathbf{W}$ is the learnable weight matrix.

4.3.3 Graph learning in hypergraph

Once the final representations $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{|\mathcal{V}^I|}$ of all items and $\mathbf{u}_1^{Final}, \mathbf{u}_2^{Final}, ..., \mathbf{u}_{|\mathcal{U}|}^{Final}$ of all users are learned, we can learn the latent representation for each user session. This task is often referred to as graph pooling, i.e., pooling together the latent representations of all the nodes to learn the latent representation of the entire graph [22]. Specifically, given a user u's session context $C = \{v_{c,1}, v_{c,2}, ..., v_{c,|C|}\}$, the representation $\mathbf{C}$ of C is obtained below:

$$\mathbf{C}_0 = \mathbf{u}^{Fianl}, \tag{20}$$

$$\mathbf{C}_t^* = LSTM(\mathbf{C}_{t-1}), \tag{21}$$

Firstly, we set the latent representation of user $\mathbf{u}$ as the initialization vector $\mathbf{C}_0$ and then use LSTM to update it iteratively. $\mathbf{C}_t^* \in \mathbb{R}^d$ represent the query vector, which is used to calculate the attention vector $e_{c,i,t}$ for item node $v_{c,i}$.

$$e_{c,i,t} = f(\mathbf{v}_{c,i}, \mathbf{C}_t^*), \tag{22}$$

$$a_{c,i,t} = \frac{exp(e_{c,i,t})}{\sum_{c,j} exp(e_{c,j,t})} \tag{23}$$

The attention vector $e_{c,i,t}$ of item node $v_{c,i}$ is determined by the query vector $\mathbf{C}_t^*$ and the latent representation of item node $\mathbf{v}_{c,i}$, f is an attention function. Then, we normalize the

attention vector $e_{c,i,t}$ to compute the attention probability score $a_{c,i,t}$.

$$r_t = \sum_{c,i} a_{c,i,t} \mathbf{v}_{c,i}, \tag{24}$$

$$\mathbf{C}_t = Contact[\mathbf{C}_t^*, r_t], \tag{25}$$

Then, in Equation (24), a weighted sum of the latent representations of all item nodes in the session context is computed based on the attention probability score. Finally, we output $\mathbf{C}_t$ as the latent representation for user session context that is used for next-item recommendations.

4.4 Prediction and optimization

After we get the context representation $\mathbf{C}$, we compute the probability $\hat{y}_i$ for target item v_i by input $\mathbf{C}$ and the latent representation $\mathbf{v}_i$ of candidate item v_i. Firstly, we calculate the inner product of the context representation $\mathbf{C}$ and the latent representations $\mathbf{v}_i$ of the candidate item, and then we employ the softmax function to calculate the probability $\hat{y}_i$ for candidate item v_i:

$$\hat{y_i} = softmax((\mathbf{v}_i)^T \mathbf{C}). \tag{26}$$

For each user session, we employ cross-emtropy as the loss function to learn the parameters of our HL framework and predict the next item. The loss function is specified as:

$$Loss = -\sum_{i=1}^{m} y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_i}), \tag{27}$$

for each candidate item v_i, y_i is the label it, When the candidate item is the true target item, the value of y_i is 1, otherwise its 0. Finally, we use the Back-Propagation Through Time (BPTT) algorithm to train the proposed model.

5 Experiments

In this section, we first describe the datasets, evaluation metrics, and comparison methods and parameter settings used in the experiments. Then, we evaluate the recommendation performance of our proposed HL framework by comparing it with other representative methods. Finally, we make detailed analysis of our results.

5.1 Experiment setup

5.1.1 Data preparation

We evaluate the proposed method on two real-world datasets, i.e., Tmall and Dunnhumby:

– Tmall:[1] involves more than 130,000 sessions accumulated on Tmall.com/ Taobao.com and the app Alipay (*i.e.,* the Chinese version of Amazon) within six month. This dataset contains user information, session information, item information and item attribute information.

[1] https://tianchi.aliyun.com/dataset/dataDetail?dataId=53

– Dunnhumby:[2] contains household level transactions over two years from a group of 2,500 households who are frequent shoppers at a retailer. It provides all of each household's purchases, and the category of the items.

For fair comparison, following [25], we firstly filter out items appearing less than 5 times and users appearing less than 3 times in both datasets. Then, we generate a session of a given user by putting all the items purchased in one transaction together. Following, we filter out the user sessions containing less than one item since at least two items should be used to build an information context and one item as the target item. Finally, for each user sessions, we split them into three sets: a training set, a test set and a validation set, specially, we randomly select 70% of each user sessions as the training set, 20% of each user sessions as the test set, and the remaining sessions as validation set. The statistics of two datasets are shown in Table 1.

5.1.2 Evaluation metrics

Following metrics are used to evaluate all the compared methods, which are also widely used in other related works [29, 32]:

– Rec@k (Recall) is widely used as a measure of predictive accuracy. It represents the proportion of correctly recommended items amongst the top-k recommended items. Here we choose $k \in \{10, 20\}$.
– Mrr@k (Mean Reciprocal Rank) is the average of reciprocal ranks of the correctly-recommended items. The reciprocal rank is set to 0 when the rank exceeds 20. The Mrr measure considers the ranking order of recommended items, where a large Mrr value indicates that correctly recommended items are ranked in the top positions of the recommendation list. Here we choose $k \in \{10, 20\}$.
– NDCG@k (Normalized Discounted Cumulative Gain) is a standard ranking metric. In the context of session based recommendation, it is formulated as:

$$NDCG@k = \frac{DCG@k}{IDCG@k}$$

where $DCG@k$ is the discounted cumulative gain and $IDCG@k$ is the ideal discounted cumulative gain. They can be computed as below:

$$DCG@k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)} \qquad IDCG@k = \sum_{i=1}^{k} \frac{1}{log_2(i+1)}$$

where rel_i is the graded relevance of the result at position k, $rel_i \in \{0, 1\}$. Here we choose $k \in \{5, 20\}$.

5.1.3 Comparison methods

In our experiments, we compare the performance of HL with the following ten baselines, which are divided into three classes of SBRSs: (1) conventional SBRSs, including POP, BPR-MF, FPMC, (2) deep learning based SBRSs, including iGRU4Rec-BPR, STAMP, NextItNet, ATEM, SR-GNN, and (3) the state-of-the-art SBRSs, HMGL.

[2] https://www.kaggle.com/frtgnn/dunnhumby-the-complete-journey

Table 1 Statistics of the experimental datasets

Statistics	Tmall	Dunnhumby
#Sessions	134,544	167,327
#Items	24,766	23,987
#Users	2,251	2,067
#Item category	763	583
#Item brand	3,641	n.a.
Avg. session length	4.98	6.56

- **POP** is a simple baseline that always recommends the most popular item in the training dataset.
- **Item-KNN** recommends items similar to the previously purchased items in the session, where similarity is defined as the cosine similarity between the vector of sessions [19].
- **BPR-MF** is a learning method based on stochastic gradient descent with bootstrap sampling, and is optimized for personalized ranking using a Bayesian analysis [15].
- **FPMC** is a Markov chains and factorization based method for next item recommendations, which factorizes the user-item transition matrix between two adjacent items within one session to learn both the general taste of a user and transitions over items [16].
- **iGRU4Rec-BPR** is a RNN-based approach for SBRSs, which employ gated recurrent units (GRUs) to model the rigid order dependencies between items within one session, and introduce a novel ranking loss function to improve the performance [5].
- **STAMP** is an attention-based approach for next item recommendations, which employs attention layer to capture the users' general interest from the previous click and employs a self-attention to capture the current interest from the last click [14].
- **ATEM** is an attention based transaction embedding model for session context embedding to weight each observed item in a transaction without assuming order [25].
- **NextItNet** is a CNN-based approach for next item recommendations, which employs a convolutional generative network to learn the high-level representation from both short- and long-rang item dependencies [33].
- **SR-GNN** is a GNN-based approach for next item recommendations, which employs a gated graph neural networks to capture the complex dependencies within one session, and then generate the latent representations by attention mechanism [20].
- **HMGL** is a GNN-based SBRSs, firstly constructs a heterogeneous graph to integrate session information and item attribute information, and then learns a unified representation for each item by simultaneously modelling the local and global dependencies for next-item recommendations [29].

5.1.4 Parameter settings

For a fair comparison, we initialize all the compared approaches with parameter settings in their papers and optimize them for best performance. For our HL framework, all the parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. The sizes of the latent representations of item nodes and user nodes are set to 128. We use the Adam optimizer with the initial learning rate 0.001. The batch size for

mini-batch optimization is set to 128, and we run 30 epoches to train our HL model for best performance.

5.2 Evaluation of the recommendation accuracy

In this section, we conduct the extensive experiments on two dataset, which aim to answer the following research questions: how does our proposed HL framework perform compared with the state-of-the-art SBRSs?

We compare the performance of our HL model with the nine representative approaches in terms of Rec and Mrr and show the results in Table 2. As for the result of conventional SBRSs, the first two methods PoP and Item-KNN, due to the simple intuition based on the frequency of appearance, the performances is relatively poor. The performance of BPR-MF is better than PoP and Item-KNN, because it factorizes the user-item transition matrix. However, the matrix are extremely sparse and easily suffer from data sparse problem. For instance, in Tmall dataset, each session only contains an average of 4.98 items of over 24,766 ones, leading to the case that each row of the constructed matrix contains less than five items and all other entries are empty. Compared with BPR-MF, FPMC is a Markov Chains and factorization machines based methods, which can takes the advantages of them, thus the performance improved. In a word, both BPR-MF and FPMC predict the target item based on its previous one only while ignoring all other items bought in the same session, which lost some information.

As for the deep learning based SBRSs, including iGRU4Rec, STAMP, ATEM, NextItNet, SR-GNN, they achieved better performance than conventional SBRSs. For example, iGRU4Rec employs a GRUs, which can capture the rigid order dependencies between items within one session. Thus it achieved better performance. STAMP employs attention mechanism to attentively learn the long-term dependencies within sessions and thus performs

Table 2 Rec and Mrr values of all compared methods on two datasets

	Tmall				Dunnhumby			
Model	Rec@10	Rec@20	Mrr@10	Mrr@20	Rec@10	Rec@20	Mrr@10	Mrr@20
POP	0.0165	0.0322	0.0049	0.0057	0.0489	0.0690	0.0245	0.0256
Item-KNN	0.1277	0.14989	0.0693	0.0754	0.0436	0.0611	0.0214	0.0234
BPR-MF	0.1744	0.2156	0.0946	0.103	0.0183	0.0343	0.0080	0.0087
FPMC	0.2889	0.3002	0.2211	0.2543	0.0433	0.0787	0.0168	0.0245
iGRU4Rec	0.3155	0.3445	0.2313	0.2387	0.0500	0.0745	0.0254	0.0298
STAMP	0.3045	0.3150	0.2337	0.2519	0.0715	0.1188	0.0300	0.0354
ATEM	0.3433	0.3938	0.2437	0.2486	0.0867	0.1655	0.0543	0.0749
NextItNet	0.2487	0.2714	0.1725	0.1788	0.1204	0.1631	0.0624	0.0669
SR-GNN	0.3656	0.4033	0.2457	0.2488	0.1925	0.2544	0.0936	0.0992
HMGL	0.3798	0.4200	0.2556	0.2598	0.2006	0.2667	0.0992	0.1035
HL	**0.3867**	**0.4311**	**0.2634**	**0.2696**	**0.2067**	**0.2708**	**0.1033**	**0.1056**
Improvement(%)[3]	1.81	2.64	3.05	3.77	3.04	1.53	4.13	2.03

[3]Improvement achieved by HL over the best-performing compared method.

better. ATEM also employed an attention mechanism to build an attentive context embedding over the session context without any order assumption, which can capture more flexible and complex dependencies within each session. Thus, it achieved better performance than STAMP. NextItNet does not perform well on Tmall dataset but perform well on Dunnhumby dataset, this maybe caused by the different data characteristics, *e.g.,* the average session length. SR-GNN employs a graph neural network, which can capture the complex transitions among items within one session and thus the performance have improved. However, it captures the intra-session dependencies from the limited session information only and fails to explicitly capture the inter-session dependencies by utilizing available side information, leading to information loss.

As for the performance of the state-of-the-art methods HMGL in SBRSs, which can capture both local dependencies (i.e., the dependencies between items within a session) and global dependencies (i.e., the dependencies across multiple sessions). Thus, the performance of HMGL is better than the methods above. However, this work ignores the significant role of users in connecting the different but relevant sessions as well as items. In contrast, our HL framework can explicitly capturing both the intra-session dependencies and the inter-session dependencies via effectively utilizing both session information and side information. we get the best performance of our HL framework on two datasets. As show in Table 2, our proposed HL framework outperforms all the other baseline methods. The improvements of HL over the best-performing method HMGL, range from 1.53% to 3.04% with an average of 2.25% in terms of Rec, and range from 2.03% to 4.13% with an average of 3.24% in terms of Mrr. The NDCG (cf. Figure 4) on two datasets also shows HL leads all the baseline methods.

The simplest method PoP recommends the next item solely based on the occurrence frequency of items, which is problematic in session-based recommendation scenarios. As for Item-KNN, they recommend similar items with previously purchased items in the session are recommended, which lead to the recommendation of the same item repeatedly, thereby reducing the recommendation accuracy. When the conventional SBRSs such as BPR-MF and FPMC are considered, they learn the probabilities to capture first-order dependencies, and can not capture the high-order dependencies. As for the deep-learning based SBRSs, including iGRU4Rec-BPR, STAMP, NextItNet, SR-GNN, HMGL owing to the powerful

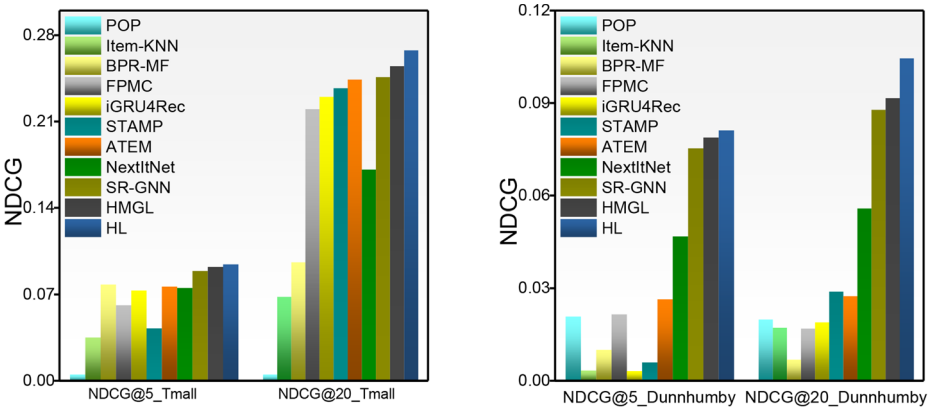

Figure 4 Our proposed HL achieves the highest NDCG values on both datasets

of deep neural networks, these methods outperform the conventional methods. A RNN-based approach iGRU4Rec-BPR employs GRUs to capture the long-term dependencies in each session, while STAMP improves the recommendation accuracy with an attention layer. ATEM achieved better performance due to the non-order assumption. NextItNet is a CNN-based approach for next item recommendations, which employs a convolutional generative network to learn the high-level representation from both short- and long-range item dependencies. Those methods usually model the rigid order within one session, while SR-GNN further capture the item dependencies within one session with graph neural networks. HMGL incorporates the session information with item attribute information to further improve the recommendation accuracy. However, it neglects the significant role of users. Our framework HL incorporates the session information and item attribute information and user information together, and captures the intra and inter-session dependencies to further improve the recommendation accuracy.

6 Conclusions

In this paper, we have proposed a hypergraph learning (HL) framework for next-item recommendations. We introduced side information including item attribute information and user information simultaneously to enrich the information for next-item recommendations. We have constructed a *hypergraph* to connect the items, item attribute values and users to effectively integrate both the session information and side information together in preparation for leaning informative item representations. Then, we have built a hypergraph leaning model to learn the representation of each item while modeling both the intra- and inter-session dependencies from the constructed hypergraph for more accurate next-item recommendations.

Current works mainly focus on single behaviours (e.g., purchasing), however, in real situations, there are usually multi-behaviour (e.g., clicking, adding to cart and purchasing) in each session. Hence, in order to capture item dependencies among multi-behaviours and further recommend the next item and the corresponding behaviours associated with it, we plan to extend the HL framework to learn more complex dependencies between items embedded in sessions consisting of other types of behaviours, e.g., clicking, adding to cart. Consequently, more informative item representations can be learned for recommending next item and its associated behaviours accurately.

Acknowledgements This work was supported by ARC (Australian Research Council) Discovery Project DP180102378.

References

1. Agarwal, S., Branson, K., Belongie, S.: Higher order learning with graphs. In: ICML, pp. 17–24 (2006)
2. Eirinaki, M., Vazirgiannis, M., Kapogiannis, D.: Web path recommendations based on page ranking and markov models. In: WIDM, pp. 2–9 (2005)
3. Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. In: AAAI, pp. 3558–3565 (2019)
4. Gao, Y., Wang, M., Tao, D., Ji, R., Dai, Q.: 3-d object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. **21**(9), 4290–4303 (2012)
5. Hidasi, B., Karatzoglou, A.: Recurrent neural networks with top-k gains for session-based recommendations. In: CIKM, pp. 843–852 (2018)

6. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session based recommendations with recurrent neural networks. In: ICLR, pp. 1–10 (2016)
7. Hosseini, S., Yin, H., Zhou, X., Sadiq, S., Kangavari, M.R., Cheung, N.-M.: Leveraging multi-aspect time-related influence in location recommendation. World Wide Web J. **22**, 1001–1028 (2019)
8. Huang, Y., Liu, Q., Metaxas, D.: Video object segmentation by hypergraph cut. In: CVPR, pp. 1738–1745 (2009)
9. Jiang, J., Wei, Y., Feng, Y., Cao, J., Gao, Y.: Dynamic hypergraph neural networks. In: IJCAI, pp. 2635–2641 (2019)
10. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Comput. **42**(8), 30–37 (2009)
11. Li, X., Yin, H., Zhou, K., Zhou, X.: Semi-supervised clustering with deep metric learning and graph embedding. World Wide Web J. **23**, 781–798 (2020)
12. Liang, D., Altosaar, J., Charlin, L., Blei, D.M.: Factorization meets the item embedding: regularizing matrix factorization with item co-occurrence. In: RecSys, pp. 59–66 (2006)
13. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: a recurrent model with spatial and temporal contexts. In: AAAI, pp. 194–200 (2016)
14. Liu, Q., Zeng, Y., Mokhosi, R., Zhang, H.: Stamp: short-term attention/memory priority model for session-based recommendation. In: KDD, pp. 1831–1839 (2019)
15. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: bayesian personalized ranking from implicit feedback. In: UAI, pp. 452–461 (2009)
16. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In: WWW, pp. 811–820 (2010)
17. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook. Springer, Berlin (2011)
18. Qiu, R., Li, J., Huang, Z., Yin, H.: Rethinking the item order in session-based recommendation with graph neural networks. In: CIKM, pp. 579–588 (2019)
19. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW, pp. 285–295 (2001)
20. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: AAAI, pp. 346–353 (2018)
21. Wu, S., Zhang, M., Jiang, X., Xu, K., Wang, L.: Personalizing graph neural networks with attention mechanism for session-based recommendation. arXiv:1910.08887 (2019)
22. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: sequence to sequence for sets. In: ICLR (2016)
23. Wang, S., Cao, L., Wang, Y., Sheng, Q.Z., Orgun, M., Lian, D.: A survey on session-based recommender systems. ACM Computing Surveys (CSUR) **54**(7), 1–38 (2021)
24. Wang, Y., Feng, C., Chen, L., Yin, H., Guo, C., Chu, Y.: User identity linkage across social networks via linked heterogeneous network embedding. World Wide Web J. **22**, 2611–2632 (2019)
25. Wang, S., Hu, L., Cao, L., Huang, X., Lian, D., Liu, W.: Attention-based transactional context embedding for next-item recommendation. In: AAAI, pp. 2532–2539 (2018)
26. Wang, S., Hu, L., Wang, Y., He, X., Sheng, Q.Z., Orgun, M., Cao, L., Francesco, R., Yu, P.S.: Graph learning based recommender systems: a review. In: IJCAI, pp. 1–9 (2021)
27. Wang, S., Hu, L., Wang, Y., Sheng, Q.Z., Orgun, M., Cao, L.: Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In: IJCAI, pp. 3771–3777 (2019)
28. Wang, S., Hu, L., Wang, Y., Sheng, Q.Z., Orgun, M., Cao, L.: Intention nets: psychology-inspired user choice behavior modeling for next-basket prediction. In: AAAI, pp. 6259–6266 (2020)
29. Wang, N., Wang, S., Wang, Y., Sheng, Q.Z., Orgun, M.: Modelling local and global dependencies for next-item recommendations. In: WISE, pp. 285–300 (2020)
30. Wang, Z., Wei, W., Cong, G., Li, X.L., Mao, X.L., Qiu, M.: Global context enhanced graph neural networks for session-based recommendation. In: SIGIR, pp. 169–178 (2020)
31. Xu, C., Zhao, P., Liu, Y., Seng, V.S., Xu, J., Zhuang, F., Fang, J., Zhou, X.: Graph contextualized self attention network for session-based recommendation. In: IJCAI, pp. 3940–3946 (2019)
32. Yu, F., Zhu, Y., Liu, Q., Wu, S., Wang, L., Tan, T.: TAGNN: target attentive graph neural networks for session-based recommendation. In: SIGIR, pp. 1–5 (2020)
33. Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M., He, X.: A simple convolutional generative network for next item recommendation. In: WSDM, pp. 582–590 (2019)
34. Zhou, D., Huang, J., Scholkopf, B.: Learning with hypergraphs: clustering, classification, and embedding. In: NIPS, pp. 1601–1608 (2007)
35. Peng, M., Zeng, G., Sun, Z., Huang, J., Wang, Hua., Tian, Gang.: Personalized app recommendation based on app permissions. World Wide Web J. **21**(1), 89–104 (2018)

Affiliations

Nan Wang[1] · Shoujin Wang[1] · Yan Wang[1] · Quan Z. Sheng[1] · Mehmet A. Orgun[1]

Nan Wang
nan.wang12@students.mq.edu.au

Shoujin Wang
shoujin.wang@mq.edu.au

Quan Z. Sheng
michael.sheng@mq.edu.au

Mehmet A. Orgun
mehmet.orgun@mq.edu.au

[1] Department of Computing, Macquarie University, Sydney, Australia