Modelling Local and Global Dependencies for Next-item Recommendations

Nan Wang, Shoujin Wang, Yan Wang, Quan Z. Sheng, and Mehmet Orgun

Department of Computing, Macquarie University, Australia
nan.wang12@students.mq.edu.au,{shoujin.wang, yan.wang, michael.sheng, mehmet.orgun}@mq.edu.au

Abstract. Session-based recommender systems (SBRSs) aim at predicting the next item by modelling the complex dependencies within and across sessions. Most of the existing SBRSs make recommendations only based on *local dependencies* (i.e., the dependencies between items within a session), while ignoring *global dependencies* (i.e., the dependencies across multiple sessions), leading to information loss and thus reducing the recommendation accuracy. Moreover, they are usually not able to recommend cold-start items effectively due to their limited session information. To alleviate these shortcomings of SBRSs, we propose a novel *heterogeneous mixed graph learning (HMGL)* framework to effectively learn both local and global dependencies for next-item recommendations. The HMGL framework mainly contains an *heterogeneous mixed graph (HMG) construction* module and an *HMG learning* module. The HMG construction module map both the session information and the item attribute information into a unified graph to connect items within and across sessions. The HMG learning module learns a unified representation for each item by simultaneously modelling the local and global dependencies over the HMG. The learned representation is then used for next-item recommendations. Results of extensive experiments on real-world datasets show the superiority of HMGL framework over the start-of-the-art methods in terms of recommendation accuracy.

Keywords: session-based recommendations, heterogeneous mixed graph learning, next-item recommendation, graph neural network.

1 Introduction

Recommender systems (RSs) have been playing an ever-increasingly important role in our daily lives to help users effectively and efficiently find items, services or contents that may be of their interests from a large amount of choices. Conventional RSs, including content-based RSs and collaborative filtering RSs, usually make recommendations based on users' long-term and static preference while ignoring their short-term and dynamic preference, which usually leads to the duplicated recommendations of similar items that can not match users' changing preferences well [20]. To this end, *session-based recommender systems (SBRSs)* were proposed to suggest the next item for a user given the purchased items in the current session [17]. Here a *session* refers to a shopping basket consisting of multiple purchased items in one transaction event.

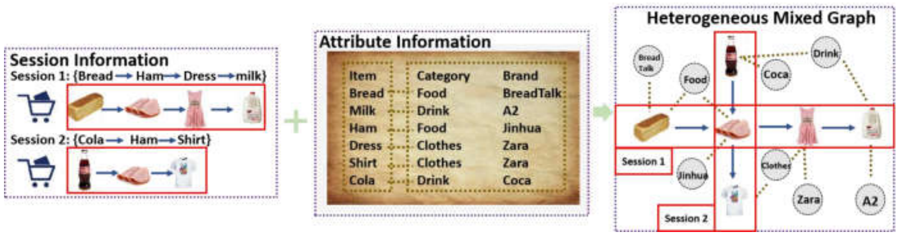
In general, an SBRS recommends the next item by modelling the complex dependencies within and across sessions. *Markov chain based SBRSs* and *recurrent neural networks (RNN) based SBRSs* are two typical types of SBRSs. In particular, a *Markov chain based SBRS* [16] predicts the next item by modelling the transitions between any two adjacent items in a session. Therefore, it only captures the *first-order dependencies* (i.e., the dependency between any two adjacent items within a session) while ignoring the *high-order dependencies* (i.e., the cascaded dependencies across multiple items within a session) within a session and hence the recommendation accuracy may suffer. To model high-order dependencies within sessions, *RNN-based SBRSs* have been proposed. Particularly, Forsati et al. [12] took gated recurrent units (GRU) as the basic cells of an RNN to model the sequential dependencies among items with rigid order assumption within each session to predict the next item. However, these models only capture the single-way transitions from the starting item to the last one within a session, while neglecting complex transitions among distant items. To capture complex transitions among items within sessions, graph neural networks (GNN) have been employed in SBRSs to predict the next item and they have achieved superior performance [15]. However, three critical gaps still remain unresolved in existing GNN-based SBRSs:

Gap1: Existing GNN-based SBRSs model each session separately in a subgraph and thus can only capture the *local dependencies* (i.e., the dependencies between items within a session), failing to explicitly capture the *global dependencies* (i.e., the dependencies across multiple sessions). This leads to information loss and thus is harmful to the subsequent recommendations.

Gap2: Existing GNN-based SBRSs usually fail to recommend cold-start items (i.e., newly appearing items with few or even no session information like transaction records) as they are based on the session information only. Recommending cold-start items is necessary in real-world cases since new items are always coming successively for sale.

Gap3: Existing GNN-based SBRSs may easily be locally optimized to fit those minor frequent items and sessions, and thus cause the overfitting problem, which reduces the recommendation accuracy.

This study addresses the above three gaps by proposing an *heterogeneous mixed graph learning (HMGL) framework* to learn the complex local and global dependencies for next-item recommendations. To be specific, to address **Gap1**, we construct an *heterogeneous mixed graph (HMG)* to connect the items from all sessions by integrating the *item-item graph* built on session information (e.g., which items are purchased together in one session), and the *item-attribute graph* built on item attribute information (e.g., the brand and category of items) into a unified graph. In such a case, items from different sessions are easily connected by taking their shared attribute values as bridges. Here, a *mixed graph* means that there are both *directed edges* between sequential items within sessions and *indirected edges* between items and their attribute values in the graph (see Fig.



Item	Category	Brand
Bread	Food	BreadTalk
Milk	Drink	A2
Ham	Food	Jinhua
Dress	Clothes	Zara
Shirt	Clothes	Zara
Cola	Drink	Coca

Fig. 1. An example of HMG constructed on both session information and item attribute information. The arrowed lines indicate the sequential relations between items from each session, while the dotted lines mean the item-attribute relations.

1). Then, we propose an *HMG learning* module to learn a unified and informative representation for each item over HMG by modelling both the local and global dependencies in preparation for the subsequent next-item recommendations. To address **Gap2**, we incorporate item attributes into the HMG to effectively enhance the connections between clod-start and warm-start items through the shared attribute values. Therefore, it is much easier to identify and recommend those cold-start items that may be preferred by users. To address **Gap3**, we introduce a regularization term in the loss function by taking the overall graph structure as an additional constraint, which can prevent the local optimizations and over-fittings. The main contributions are summarized as follows:

- We construct a heterogeneous mixed graph (HMG) to encode both the session information and the item attribute information into one unified graph to connect all the items together. This enriches both local and global dependencies among items, especially for those clod-start ones.
- We propose an end-to-end HMG learning model to learn an informative representation for each item over the HMG by modelling both local and global dependencies to better prepare for the next-item recommendations.
- A new loss function with an additional regularization term is proposed for global optimization to improve the recommendation accuracy.

Extensive experiments have been conducted on two real-world transaction datasets to evaluate our proposed framework HMGL. The results show the superiority of HMGL over the state-of-the-art methods.

2 Related Work

In this section, we briefly review the representative SBRSs, which can be classified into: (1) conventional SBRSs, including Markov chain based SBRSs and factorization machine based SBRSs, and (2) deep learning based SBRSs, including RNN-based SBRSs and GNN-based SBRSs.

Conventional session-based recommender systems. Markov chain models and factorization machines are two conventional approaches for SBRSs. Given the prior items in a session, Markov chain based SBRSs adopt Markov chain models to model the transitions over these items in a sequence, for the prediction of the next item. However, due to the widely employed Markov property, Markov chain based SBRSs usually can only capture the first-order dependencies between two adjacent items while ignoring the high-order dependencies across multiple items. Factorization machines are adopted to factorize the observed transition from the current item to the next one into the latent representations of items, which are then used to estimate the unobserved transitions between items for predicting the next item [7]. As a combination of Markov chain models and factorization machines, Factorizing Personalized Markov Chain (FPMC) was built for SBRSs to take the advantages of both the models for better recommendations [16]. However, similar to Markov chain based SBRSs, factorization machine based SBRSs capture first-order dependencies only, in addition, they easily suffer from the data sparsity issue.

Deep learning based session-based recommender systems. With the capability of handling sequential data, recurrent neural network (RNN) is an intuitive choice to capture the sequential dependencies in SBRSs. The first RNN-based SBRSs is GRU4Rec [1], which employes gated recurrent unit (GRU) to capture the long-term dependencies within sessions. Later, Balazs, et al. [2] improved GRU4Rec by introducing a ranking loss function. To emphasize those items that are more relevant to the next-item, a self-attention model was combined with RNN for next-item recommendation. However, due to the employed rigid order assumption that any adjacent items within a session are highly sequentially dependent, RNN-based SBRSs may generate false dependencies [20].

In recent years, benefiting from the power of capturing the complex transitions among items, graph neural network (GNN) [22] has been employed to built more powerful SBRSs. The first GNN-based SBRS is SR-GNN [15], which first maps each session into a sub directed graph and then applies GNN on the graph to capture dependencies between items within each session. Later, Xu, et al. [3] proposed a graph contextualized self-attention model (GC-SAN) to incorporate an attention mechanism into GNN to learn the long-range dependencies within sessions for next-item recommendations. Qiu et al. [13] proposed a weighted attention graph layer and a readout function to learn embeddings of items while relaxing the order assumption over items for the next item recommendation. Yu et al. [5] proposed a novel target attentive graph neural network (TAGNN) to adaptively activate a user's different interest w.r.t. varied next-item for more accurate next-item recommendations. However, most of the existing GNN-based SBRSs process each session separately, i.e., the GNN is employed on each subgraph successively to learn item representations for recommendations. In this way, they fail to explicitly capture the global dependencies based on multi-session and attribute information, which leads to information loss and thus is harmful to the subsequent recommendations. Moreover, these methods poor to recommend cold-start items due to their quite limited session information.

Only a few models have been proposed to capture the inter-session dependencies, like hierarchical RNN [10] and hierarchical attention [6] based SBRSs. However, they only capture the single-way sequential dependencies between sessions from the same user, totally different from the global dependencies in this paper. In practice, the dependencies across sessions are much more complex than the aforementioned sequential dependencies, since different sessions can be also widely connected via the shared items or shared item attribute values. This essentially motivates us to develop HMGL to learn the complex global dependencies across sessions to further improve the recommendation performance.

3 Problem Statement

We formulate the research problem in this section. Let $\mathcal{S} = \{s_1, s_2, ..., s_{|S|}\}$ denote the session set consisting of all the sessions in a dataset, where $|S|$ is the number of sessions in $\mathcal{S}$. $\mathcal{V}^I = \{v_1, v_2, ..., v_{|\mathcal{V}^I|}\}$ denotes the item set consisting of all the unique items from all sessions. $s_i = \{v_{i,1}, v_{i,2}, ..., v_{i,|s_i|}\}$ is a session consisting of sequentially interacted (e.g., purchased by an anonymous user) items. $F = \{f_1, f_2, ..., f_{|F|}\}$ is the set of attributes for all items and each attribute (e.g., category) $f_h = \{a_{h,1}, a_{h,2}, ..., a_{h,|f_h|}\}$ consists of multiple attribute values (e.g., food, beverage), denoted as $a_{h,b} (1 \leq b \leq |f_h|)$. All the attribute values constitute the attribute value set $\mathcal{V}^{\mathcal{A}} = \{a_{h,1}, a_{h,2}, ..., a_{h,|f_h|}\}$ $(h \in \{1, 2, ..., |F|\})$. Usually, there are both categorical and numerical attributes for items. In this paper, we consider categorical attributes only since numerical attributes need to be handled different from the categorical ones. For each item $v_i \in \mathcal{V}^I$, all its attribute values form a set $\mathcal{A}_{v_i} = \{a_1, ..., a_h, ..., a_{|F|}\}$. For a target item v_l from a session s, all the items that occurred prior to v_l in s form the session context of v_l, denoted as $C^s_{v_l} = \{v_1, v_2, ..., v_{l-1}\}$, while all the attribute values of the items in $C^s_{v_l}$ form the corresponding attribute context $C^a_{v_l} = \{\mathcal{A}_{v_1}, \mathcal{A}_{v_2}, ..., \mathcal{A}_{v_{l-1}}\}$.

Given a context $C = [C^s, C^a]$ with $(l-1)$ precedent items associated with their attribute values, the task of our work is to recommend the l^{th} item. Accordingly, our proposed HMGL learns the conditional probability distributions $P(v|C)$ for each candidate item $v (v \in \mathcal{V}^I)$ given the context C. Consequently, once all the model parameters have been learned, the next item to be recommended can be selected from the candidate items by maximizing $P(v|C)$.

4 Heterogeneous Mixed Graph Learning Framework

In this section, we present our proposed heterogeneous mixed graph learning (HMGL) framework. In particularly, the HMGL framework first constructs an HMG by mapping the session information and item attribute information into a unified graph and then jointly learns both local and global dependencies over the HMG for next-item recommendations. Accordingly, as shown in Fig. 2, the HMGL framework consists three components: the HMG construction module,

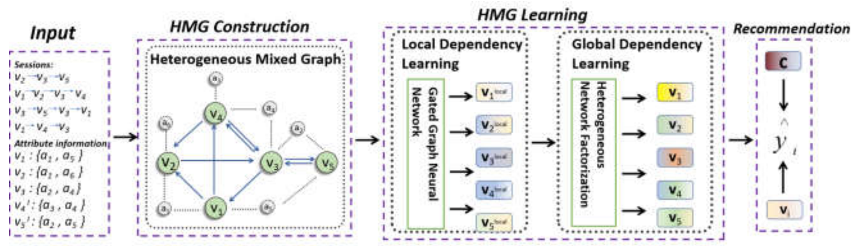


Fig. 2. The workflow of our proposed an HMGL framework. We integrate both session information and attribute information into an HMG. Then, the HMG learning module is devised to learn both local and global dependencies over HMG and export an informative representation for each item. Finally, we predict the conditional probability of each candidate item and recommend the item with maximum probability.

the HMG learning module, the next-item prediction module. Specifically, the HMG learning model includes both the local dependency learning and the global dependency learning modules. In the following subsections, we first introduce some preliminaries and then introduce each component of the HMGL framework.

4.1 Preliminary

We define two types of graphs, i.e., heterogeneous mixed graph and underlying graph, which will be used in the following subsections.

Definition 1 ***Heterogeneous Mixed Graph.*** *A heterogeneous mixed graph (HMG) $\widetilde{\mathcal{G}} = \{\mathcal{V}, \mathcal{E}, \mathcal{D}\}$ is composed of a node set $\mathcal{V}$, an edge set $\mathcal{E}$, and a direction set $\mathcal{D}$. In addition, the numbers of both node types and edge types are larger than one, while there are both directed and undirected edges in the graph.*

Definition 2 ***Underlying Graph.*** *The underlying graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ of a given HMG $\widetilde{\mathcal{G}} = \{\mathcal{V}, \mathcal{E}, \mathcal{D}\}$ is an undirected graph extracted from the HMG by changing the directed edges into undirected ones. Hence, an underlying graph consists of the node set $\mathcal{V}$ and the edge set $\mathcal{E}$.*

4.2 Heterogeneous Mixed Graph Construction

We incorporate the item attribute information into SBRSs to enrich the connections between items for better next-item recommendations. In order to construct a unified HMG ($\widetilde{\mathcal{G}} = \{\mathcal{V}, \mathcal{E}, \mathcal{D}\}$), firstly, we model the items and their attribute values as two types of nodes in a graph, let $\mathcal{V} = \mathcal{V}^{\mathcal{I}} \cup \mathcal{V}^{\mathcal{A}}$ be the node set on the graph, where $\mathcal{V}^{\mathcal{I}}$ and $\mathcal{V}^{\mathcal{A}}$ are the node sets corresponding to items and item attribute values respectively, as defined in Section 3.

Secondly, we model the item-item relations within each session and the item-attribute value relations as two types of edges in HMG, namely, $\mathcal{E} = \mathcal{E}^{\mathcal{I}} \cup \mathcal{E}^{\mathcal{A}}$

constitute the edge set in HMG, where a directed edge $e^{I}_{i,j} \in \mathcal{E}^{\mathcal{I}}$ means item v_j occurs after item v_i in a given session, an undirected edge $e^{A}_{i,l} \in \mathcal{E}^{\mathcal{A}}$ means that item v_i has the attribute value a_l. As a result, an HMG connecting all items based on session information and attribute information is built. In this way, the shared attribute values of different items serve as bridges to connect items indirectly, since there is an edge between the shared attribute values and each of the items sharing these attribute values on the HMG. As a result, such an HMG not only enriches the connections between items, especially those items from different sessions, but also connects cold-start items and warm-start items.

4.3 Heterogeneous Mixed Graph Learning

As shown in Fig.2, the HMG learning module contains two parts: *local dependency learning* and *global dependency learning*. First, a gated graph neural network (GGNN) is utilized for the local dependency learning by taking the initialized item representations as the input and output the local item representations which encode the local dependencies within sessions. Then, the local representations are imported into a path-based matrix factorization model to obtain the final item representations while further incorporating the global dependencies between items across sessions. Finally, the learned final representations encoding both local and global dependencies are fed into the prediction layer for next-item prediction. Next, we introduce the two parts one by one in detail.

Local Dependency Learning. In order to learn the local dependencies, we firstly extract a directed subgraph $\mathcal{G}_s = \{\mathcal{V^I}_s, \mathcal{E^I}_s\}$ based on a given session s from the HMG. Then, for the i^{th} node $v_{s,i}$ in $\mathcal{G}_s$, we learn a latent representation $\mathbf{v}_{s,i}$ via gated graph neural networks (GGNN) [8]. GGNN updates the representation of each node through absorbing the information from other nodes in the same subgraph. Subsequently, the local dependencies embedded in each subgraph (session) are encoded into the representation of each node (item).

Firstly, we map each node $v \in \mathcal{V}^I$ into an unified low-dimension latent space to obtain the initial representation $\mathbf{v} \in \mathbb{R}^{1\times d}$, where d denotes the dimension of the representation. Then for each node $v_{s,i} \in \mathcal{G}_s$, we employ GGNN to iteratively update its representation $\mathbf{v}_{s,i}$ by absorbing the information from other nodes in $\mathcal{G}_s$ to learn the local dependencies. In specific, in the t^{th} iteration, we first extract the contextual information $\mathbf{a}^t_{s,i}$ from the neighborhoods of $v_{s,i}$ under the constraint matrix $\mathbf{A}^s$ based on $\mathcal{G}_s$:

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

where $\mathbf{H} \in \mathbb{R}^{d\times d}$ is the weight matrix, $\mathbf{b}$ is the bias vector, $[\mathbf{v}^{t-1}_{s,1}, \mathbf{v}^{t-1}_{s,2}, ..., \mathbf{v}^{t-1}_{s,|s|}]$ is the list of hidden states (i.e., representations) of nodes in $\mathcal{G}_s$ at $(t-1)^{th}$ iteration. The constraint matrix $\mathbf{A}^s = [\mathbf{A}^{s,I}, \mathbf{A}^{s,O}] \in \mathbb{R}^{|s|\times 2|s|}$ is the concatenation of two adjacency matrices $\mathbf{A}^{s,I}$ and $\mathbf{A}^{s,O}$, while $\mathbf{A}^s_{i,:}$ denotes the i^{th} row of $\mathbf{A}^s$ and it corresponds to v_i. $\mathbf{A}^{s,I}$ and $\mathbf{A}^{s,O}$ represent weighted connections of incoming and outgoing edges in $\mathcal{G}_s$ respectively, which are calculated as occurrence times of the corresponding edge divided by the outdegree (the number of tail ends adjacent to a node) of the edge's starting node. In this way, the communications between nodes and different importance scales indicated by their frequencies of edges in the subgraph are captured.

Once the contextual information is extracted, we take $\mathbf{a}_{s,i}^t$ and the hidden state $\mathbf{v}_{s,i}^{t-1}$ of v_i in the $(t-1)^{th}$ iteration as the input to calculate the candidate hidden state $\widetilde{\mathbf{v}}_{s,i}^t$ of $v_{s,i}$ for the t^{th} iteration as presented by Eq (4), where the reset gate vector $\mathbf{r}_{s,i}^t$ and the update gate vector $\mathbf{z}_{s,i}^t$ are calculated using Eqs (2) and (3) respectively:

$$\mathbf{z}_{s,i}^t = \sigma(\mathbf{W}_z a_{s,i}^t + \mathbf{U}_z \mathbf{v}_{s,i}^{t-1}), \tag{2}$$

$$\mathbf{r}_{s,i}^t = \sigma(\mathbf{W}_r a_{s,i}^t + \mathbf{U}_r \mathbf{v}_{s,i}^{t-1}), \tag{3}$$

$$\widetilde{\mathbf{v}}_{s,i}^t = tanh(\mathbf{W}_o a_{s,i}^t + \mathbf{U}_o (r_{s,i}^t \odot \mathbf{v}_{s,i}^t)), \tag{4}$$

where $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_o \in \mathbb{R}^{2d \times d}$ and $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_o \in \mathbb{R}^{d \times d}$ are learnable weight matrices, $\sigma(\cdot)$ is the activation function and is specified as sigmoid function, $\odot$ denotes the element-wise multiplication operation.

Subsequently, the hidden state $\mathbf{v}_{s,i}^t$ of $v_{s,i}$ in the current t^{th} iteration can be determined by the update gate $\mathbf{z}_{s,i}^t$, the hidden state $\mathbf{v}_{s,i}^{t-1}$ of $v_{s,i}$ in the $(t-1)^{th}$ iteration and the candidate hidden state $\widetilde{\mathbf{v}}_{s,i}^t$:

$$\mathbf{v}_{s,i}^t = (1 - \mathbf{z}_{s,i}^t) \odot \mathbf{v}_{s,i}^{t-1} + \mathbf{z}_{s,i}^t \odot \widetilde{\mathbf{v}}_{s,i}^t. \tag{5}$$

In this way, the local dependencies between item $v_{s,i} \in s$ and other items in session s are encoded into the latent representation of $v_{s,i}$. Further, we can learn the representations of other items $v_{s,j} \in s (j \neq i)$ in the same way. After all the sessions in the dataset are processed in the same way, the local representations encoding local dependencies of all items are subsequently learned as $[\mathbf{v}_1^{local}, \mathbf{v}_2^{local}, ..., \mathbf{v}_{|\mathcal{V}^I|}{}^{local}]$.

Global Dependency Learning. In order to learn global dependencies, we firstly extract an underlying graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ from the whole HMG. Then two types of paths over $\mathcal{G}$ are defined to reveal two kinds of global inter-item dependencies. One type of path contain two adjacent item nodes on $\mathcal{G}$ to real their co-occurrence relations in the same sessions, denoted as "$v_i - v_j$"; the other type of path contains two item nodes plus their shared attribute value nodes, to reveal the indirect dependencies between items sharing the same attribute value, denoted as "$v_i - a_k - v_j$", where $v_i, v_j \in \mathcal{V}^{\mathcal{I}}$ and $a_k \in \mathcal{V}^{\mathcal{A}}$. Accordingly, the following two path matrices are built among all the item nodes: N^I with the entry $n_{i,j}^I$ to denote the number of the first type of paths between v_i and v_j, and N^A with the entry $n_{i,j}^A$ to denote the number of the second type of paths between v_i and v_j. Inspired by [24] and [23], the number of paths between two nodes in a graph reflects the strength of dependency between them and can be estimated by the latent factors of them. Therefore, we factorize the path matrices into the latent factors of items to fine-tune the item representations to incorporate the global dependencies by taking the local representations of items as the input to initialize the latent vectors of items.

Hence, the two path matrices N^I and N^A are factorized jointly by estimating the values of entries in them respectively using the latent vectors of items below:

$$\widehat{n}^I_{i,j} = f(\mathbf{p}^I, \mathbf{v}_i, \mathbf{v}_j) = \sum_{q=1}^{d} \mathbf{p}^I_q (\mathbf{v}_{i,q})^T \mathbf{v}_{j,q}, \tag{6}$$

$$\widehat{n}^A_{i,j} = f(\mathbf{p}^A, \mathbf{v}_i, \mathbf{v}_j) = \sum_{q=1}^{d} \mathbf{p}^A_q (\mathbf{v}_{i,q})^T \mathbf{v}_{j,q}, \tag{7}$$

where $\mathbf{p}^I, \mathbf{p}^A \in \mathbb{R}^d$ are the latent vectors representing the types of paths, and d is the dimension of the latent space. $\mathbf{v}_{i,q}$ is the q^{th} bit of the latent vector $\mathbf{v}_i$ of v_i. After the conduction of the factorization, the item representations are updated accordingly to incorporate the global dependencies. As a result, the informative final representation, e.g., $\mathbf{v}_i$, of each item, e.g., $v_i (v_i \in V^I)$, is achieved, which encodes both local and global dependencies for better next-item prediction.

4.4 Prediction and Optimization

Next-item Prediction. Once the final representations $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{|\mathcal{V}^I|}$ of all items are learned by HMGL framework, the prediction can be made by taking them as the input. Specifically, given a context $C = \{v_{c,1}, v_{c,2}, ..., v_{c,|C|}\}$, the representation $\mathbf{C}$ of C is obtained by integrating the representations of all items in it via a fully connected layer:

$$\mathbf{C} = [\mathbf{v}_{C,1}, \mathbf{v}_{C,2}, ..., \mathbf{v}_{C,|C|}]^T \mathbf{W}_{\mathbf{C}} \tag{8}$$

where $\mathbf{W}_C$ is the weight matrix to be learned. Then, we feed the context representation $\mathbf{C}$ together with the latent representation of candidate item $\mathbf{v}_i$ into the output layer for the target item prediction. Specifically, a score that quantifies the relevance between the context and candidate item is computed as the inner product of the context representation $\mathbf{C}$ and the candidate item representation $\mathbf{v}_i$. Finally, we apply a softmax function to predict the conditional probability $\hat{y}_i$ for each candidate $v_i \in \mathcal{V}^I$:

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

Optimization. To learn the parameters of the proposed model, we utilize an end-to-end training scheme, the total objective is to minimize the following loss:

$$Loss^{Total} = Loss^{local} + \gamma(Loss^{Global}). \tag{9}$$

where γ is the coefficient to control the importance of the $Loss^{Global}$. The $Loss^{Global}$ also acts as regularization term for better optimization and prevent the over-fittings or local optimizations.

The local loss function $Loss^{local}$ is defined as the cross-entropy between the prediction $\hat{y}$ and the ground truth y, which can be written as:

$$Loss^{local} = -\sum_{i=1}^{m} y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_i}) \tag{10}$$

where y is the label of each candidate item, its value is 1 when the candidate item is the true target item, and 0 otherwise.

The global loss function $Loss^{local}$ is defined as the root-mean-square error (RMSE), which is an estimator with respect to the true path matrices N^I, N^A and predicted matrix $\hat{N}^I, \hat{N}^A$, which is defined as the square root of the mean square error:

$$\begin{aligned} Loss^{Global} &= Loss^I + Loss^A \\ &= \frac{1}{|\mathcal{V}^I||\mathcal{V}^I|} \sum_{i,j \in \mathcal{V}^I} \sqrt{(\widehat{n}_{i,j}^I - n_{i,j}^I)^2} + \sqrt{(\widehat{n}_{i,j}^A - n_{i,j}^A)^2} \end{aligned} \tag{11}$$

Finally, we use a mini-batch gradient descent to train the proposed HMGL model. Note that in session-based recommendation scenarios, most sessions are of relatively short lengths, SBRSs are easy to suffer from the over-fitting during the optimization. $Loss^{Global}$ is also acted as the penalty term in the total loss function, which can effectively prevent the local optimization and over-fitting.

5 Experiments

In this section, we introduce the datasets, evaluation metrics, comparison methods and parameter settings, and we evaluate the recommendation performance of our proposed HMGL framework by comparing it with the baseline methods.

5.1 Preparation

Data Preparation The following two commonly used real-world transaction datasets are used for experiments:

- Tmall: released by IJCAI-15 competition, which records the pruchased items as well as their attribute information (i.e., category and brand) in each transaction on Tmall.com (Chinese version of Amazon.com).
- Dunnhumby: provides the household transactions of 2,500 households over 2 years (collected by the data science company Dunnhumby). In addition, the category information of each item is provided.

Firstly, a set of sessions is extracted from the original transaction data by putting all the items in one transaction together to form a session. Following a common practice [20], those sessions containing less than three items are removed since at least two items should be used to build an informative context and the addition one as the target item. The item information table contains the attribute values of each item occurred in the sessions. Secondly, the set of sessions is splitted into training set, test set and validation set respectively. We randomly select 70% as the training set, 20% as the test set, and the rest 10% for validation. Finally, to test the performance of our proposed model under different cold-start levels, part of the sessions in training set are removed to form the training sets with various cold-start levels. To be specific, we construct 3 different training sets with a drop rate of 0%, 40%, and 80%, respectively. Taking the one with the drop rate of 80% as an example, for each target item to be predicted in the testing set, 80% of all the sessions containing it in the training set are dropped. The statistic of the datasets are shown in Table 1.

Table 1. Statistics of experimental datasets.

Statistics	**Tmall**	**Dunnhumby**
#Sessions	125,111	173,913
#Items	26,251	24,897
#Item category	763	583
#Item brand	3,641	n.a
Avg. session length	5.21	8.09

Evaluation Metrics We use the following widely used accuracy metrics to evaluate all the comparison approaches.

– Rec@k (Recall) is a metric to measure prediction accuracy. It represents the proportion of the correctly recommended items amongst the recommended top-k items. Here we choose $k \in \{10, 20\}$.
– Mrr@k (Mean Reciprocal Rank) is the average of reciprocal ranks of the true target items over all recommendation instances. Here we choose $k \in \{10, 20\}$.

Comparison Methods To demonstrate the efficacy of our proposed HMGL framework, which extracts both global and local dependencies for next-item recommendations, we implemented two versions of our model: (1) full version of **HMGL** proposed in this work; and (2) $\mathbf{HMGL-L}$, which only utilizes local dependencies. We take the representative methods for performance comparisons, which are built on representative frameworks including matrix factorization, Markov chains, recurrent neural networks (RNN), memory networks, convolutional neural networks (CNN) and graph neural networks (GNN).

– **POP** recommends the top-k frequent items in the training set and in the current session respectively.
– **BPR-MF** is the state-of-the-art method for non-sequential recommendations, which uses a pairwise ranking loss [16].
– **FPMC** is a classic model combining matrix factorization and first-order Markov Chain for next-basket recommendations [11]. Here it is utilized to

factorize the transition matrix between any two items in the session data and thus predict the next-item based on the last item in the session.

- **iGRU4Rec-BPR** is the improved version of typical RNN-based SRBS, namely GRU4Rec, which uses GRU to model the sequences of purchased items in sessions for next-item recommendations. It takes Bayesian Personalized Ranking (BPR) as the loss function [2].
- **STAMP** is a novel short-term memory priority model to capture both the user's long-term preference from previous clicks and the current preference from the last click in a session for the next-item recommendations [9].
- **NextItNet** uses a convolutional generative network to model long range dependencies in sequences of items for next-item recommendations [4].
- **SR-GNN** is an SBRS model using gated graph neural networks to first generate latent representations of items in sessions and then use these representations for next-item recommendations [15].

Parameter Settings We initialize all the baseline models with the parameter settings reported in their papers and then tune them on our datasets for best performance for fair comparison. For our model, all parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. The sizes of item representations and hidden states in HMGL are set to 128. The mini-batch Adam optimizer is utilized to learn the model parameters, where the initial learning rate is set to 0.001, and the coefficient γ is set to 0.1 via cross validation on the specific datasets, the batch size is set to 100. We run 30 epoches to train our HMGL model for best performance.

5.2 Recommendation Accuracy Evaluation

Extensive experiments are conducted to answer the following questions:

- **Q1:** How does our approach perform compared with the state-of-the-art SBRSs in terms of recommendation accuracy in the warm-start situation?
- **Q2:** How does our approach perform compared with the state-of-the-art SBRSs in terms of recommendation accuracy in the cold-start situation?
- **Q3:** How does our full model HMGL which models both global and local dependencies perform compared with its simplified version HMGL-L which models local dependencies only?

Result 1 (for Q1): Comparison with Baselines under Warm-start Situation. We compare the recommendation accuracy of our HMGL model with those of eight representative baselines in the warm-start situation. The results are shown in Table 2 (the drop 0% scenario). The performance of the first two methods PoP and Item-KNN is poor, since they make recommendations only based on the popularity or similarity of items, failing to effectively capture the dependencies between items in sessions. The performance of BPR-MF is a little better, but still poor. The main reason is that both datasets are extremely

Table 2. Results of effectiveness experiments on two datasets.

		Tmall				**Dunnhumby**			
Scenario	**Model**	**Rec@10**	**Rec@20**	**Mrr@10**	**Mrr@20**	**Rec@10**	**Rec@20**	**Mrr@10**	**Mrr@20**
drop 0%	POP	0.0170	0.0310	0.0050	0.0060	0.0441	0.0666	0.0215	0.0229
	Item-KNN	0.1280	0.1490	0.0680	0.0720	0.0412	0.0596	0.0186	0.0200
	BPR-MF	0.1760	0.2150	0.0960	0.0968	0.0196	0.0337	0.0074	0.0081
	FPMC	0.2850	0.3070	0.2200	0.2500	0.0445	0.0749	0.0183	0.0223
	iGRU4Rec	0.3170	0.3410	0.2300	0.2360	0.0507	0.0731	0.0205	0.0220
	STAMP	0.3000	0.3090	0.2370	0.2490	0.0734	0.1124	0.0313	0.0354
	NextItNet	0.2490	0.2700	0.1710	0.1720	0.1204	0.1591	0.0604	0.0629
	SR-GNN	0.3680	0.4100	0.2460	0.2490	0.1931	0.2527	0.0951	0.0992
	HMGL-L	0.3681	0.4103	0.2462	0.2492	0.1935	0.2527	0.0952	0.0995
	HMGL	**0.3783**	**0.4185**	**0.2549**	**0.2577**	**0.2006**	**0.2627**	**0.0992**	**0.1035**
	Improvement(%)[1]	2.07	2.54	3.61	3.49	3.88	3.95	4.31	4.33
drop 40%	POP	0.0150	0.0270	0.0050	0.0060	0.0419	0.0641	0.0208	0.0224
	Item-KNN	0.0820	0.1210	0.0350	0.0380	0.0075	0.0108	0.0033	0.0034
	BPR-MF	0.1150	0.1380	0.0780	0.0810	0.0022	0.0339	0.0100	0.0112
	FPMC	0.1176	0.1205	0.0612	0.0661	0.0441	0.0666	0.0215	0.0229
	iGRU4Rec	0.1508	0.1533	0.0731	0.0852	0.0077	0.0128	0.0031	0.0034
	STAMP	0.1698	0.1770	0.0424	0.0555	0.0193	0.0238	0.0059	0.0076
	NextItNet	0.1418	0.1444	0.0752	0.0766	0.0859	0.1092	0.0468	0.0490
	SR-GNN	0.1750	0.2120	0.0890	0.0923	0.1600	0.2140	0.0753	0.0790
	HMGL-L	0.1751	0.2122	0.0893	0.0925	0.1603	0.2141	0.0754	0.0791
	HMGL	**0.1799**	**0.2174**	**0.09232**	**0.0953**	**0.1724**	**0.2284**	**0.0788**	**0.0825**
	Improvement(%)	2.80	2.54	3.73	3.25	7.75	6.72	4.46	4.43
drop 80%	POP	0.0080	0.0215	0.0050	0.0060	0.0291	0.0499	0.0172	0.0187
	Item-KNN	0.0510	0.0720	0.0300	0.0300	0.0052	0.0081	0.0022	0.0024
	BPR-MF	0.0300	0.0340	0.0350	0.0390	0.0215	0.0295	0.0080	0.0087
	FPMC	0.0333	0.0507	0.0328	0.0336	0.0065	0.0072	0.0027	0.0032
	iGRU4Rec	0.0408	0.0425	0.0420	0.0471	0.0075	0.0124	0.0236	0.0338
	STAMP	0.0951	0.1564	0.0689	0.0694	0.0087	0.0666	0.0041	0.0229
	NextItNet	0.0859	0.0938	0.0351	0.0413	0.0119	0.0145	0.0215	0.0313
	SR-GNN	0.1337	0.1652	0.0659	0.0681	0.0652	0.0850	0.0342	0.0356
	HMGL-L	0.1338	0.1653	0.0659	0.0683	0.0653	0.0851	0.0343	0.0357
	HMGL	**0.1395**	**0.1698**	**0.0694**	**0.0702**	**0.0702**	**0.0919**	**0.0359**	**0.0366**
	Improvement(%)	4.33	2.78	5.31	3.08	7.66	8.11	4.97	2.80

[1]Improvement achieved by HMGL over the best-performing compared methods.

sparse and MF models easily suffer from sparse data. FPMC takes the advantages of both Markov Chain and factorization machines, and thus performs a slightly better than BPR-MF. But FPMC is a fist-order Markov Chain model, which can only learn the transitions over adjacent items while ignoring high-order dependencies. Benefiting from the capability of capturing complex relations of deep neural networks, iGRU4Rec, STAMP and NextItNet achieved better performance than the aforementioned models. For example, iGRU4Rec employs RNN built on gated recurrent unit (GRU) to model the sequential dependencies within sessions. STAMP and NextItNet employs attention mechanism and CNN respectively to model intra-session dependencies for next-item recommendations. By capturing the complex transitions among items, the performance of SR-GNN is better. However, it fails to explicitly capture the global dependencies, leading to information loss and thus is harmful to the subsequent recommendations.

By explicitly capturing both the local dependencies and the global dependencies, our HMGL framework achieves the best performance on both datasets.

It outperforms the best-performing method SR-GNN with an average of 3.52%, ranging from 2.07% to 4.33%, in terms of Rec@10, Rec@20, Mrr@10 and Mrr@20.

Result 2 (for Q2): Comparison with Baselines under Cold-start Situations. In order to model the cold-start and warm-start scenario, we construct three different training sets with a drop rate of 0%, 40%, and 80%. It is clear that HMGL achieves higher accuracy than any of the eight baseline methods in these two cold-start scenarios, shown by the scenarios demoted as "drop 40% " and "drop 80% " in Table 2. Specifically, when dropping 40%, the average improvement percentage achieved by HMGL over the best-performing methods is 4.46%, range from 2.8% to 7.75%. When dropping 80%, the average improvement percentage is 4.88%, range from 2.8% to 8.11%. This verifies the effectiveness of of our proposed HMGL framework in handling the cold-start items.

Result 3 (for Q3): Global and Local Dependencies vs. Local Dependencies Only. To demonstrate the efficacy of the modeling of global dependencies, we compare the performance of HMGL with that of HMGL-L. As shown in Table 2, it is clear that HMGL achieves higher accuracy under all scenarios on both datasets than HMGL-L does. This justifies the necessity to explicitly and comprehensively model the global dependencies over the whole dataset for more accurate next-item recommendations.

6 Conclusions

In this paper, we propose an Heterogeneous Mixed Graph Learning (HMGL) framework for session-based recommendations. Firstly, we have constructed a heterogeneous mixed graph based on both session information and attribute information. Then we have designed an HMG learning model to learn a unified representation for each item by modelling both local and global dependencies. The learned representations are further used for next-item recommendations. Extensive experiments on two real-world datasets demonstrated the effectiveness of HMGL. As for future work, we plan to further utilize item attributes by combining them together with item IDs to build more informative item representations for better addressing the cold-start recommendation issue.

7 Acknowledgements

This work was supported by ARC Discovery Project DP180102378.

References

1. Balazs Hidasi, Alexandros Karat-zoglou, et al. Session based recommendations with recurrent neural networks. In ICLR, pages 1–10, 2016.

2. Balazs Hidasi, Alexandros Karat-zoglou, et al. Recurrent neural networks with top-k gains for session-based recommendations. In CIKM, pages 843–852, 2018.
3. Chengfeng Xu, Pengpeng Zhao, et al. Graph contextualized self attention network for session-based recommendation. In IJCAI, pages 3940–3946, 2019.
4. Fajie Yuan, Alexandros Karatzoglou, et al. A simple convolutional generative network for next item recommendation. In WSDM, pages 582–590, 2019.
5. Feng Yu, Yanqiao Zhu, et al. TAGNN: target attentive graph neural networks for session-based recommendation. In SIGIR, pages 1-5, 2020.
6. Haochao Ying, Fuzhen Zhuang, et al. Sequential recommender system based on hierarchical attention networks. In IJCAI, pages 3926–3932, 2018.
7. Greg Linden, Brent Smith and Jeremy York. Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing, 7(1): 76–80, 2003.
8. Yujia Li, Daniel Tarlow, et al. Gated graph sequence neural networks. In ICLR, pages 1-14, 2015.
9. Qiao Liu, Yifu Zeng, et al. Stamp: Short-term attention/memory priority model for session-based recommendation. In KDD, pages 1831–1839, 2018.
10. Massimiliano Ruocco, Ole Steinar Lillestl Skrede, et al. Inter-session modeling for session-based recommendation. In 2nd Workshop on DLRS, pages 24–31, 2017.
11. Steffen Rendle, Christoph Freudenthaler, et al. Factorizing personalized markov chains for next-basket recommendation. In WWW, pages 811–820, 2010.
12. Rana Forsati, Mohammad Reza Meybodi, et al. Web page personalization based on weighted association rules. In ICECT, pages 130–135, 2009.
13. Ruihong Qiu, Jingjing Li, et al. Rethinking the item order in session-based recommendation with graph neural networks. In CIKM, pages 579–588, 2019.
14. Shanshan Feng, Xutao Li, et al. Personalized ranking metric embedding for next new poi recommendation. In IJCAI, pages 2069–2075, 2015.
15. Shu Wu, Yuyuan Tang, et al. Session-based recommendation with graph neural networks. In AAAI, pages 346-353, 2018.
16. Steffen Rendle, Christoph Freudenthaler, et al. Factorizing personalized markov chains for next-basket recommendation. In WWW, pages 811–820, 2010.
17. Shoujin Wang, Liang Hu, et al. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In IJCAI, pages 3771–3777, 2019.
18. Shoujin Wang, Longbing Cao. Inferring implicit rules by learning explicit and hidden item dependency. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50(3): 935-946,2020.
19. Shoujin Wang, Liang Hu, et al. Intention nets: psychology-inspired user choice behavior modeling for next-basket prediction. In AAAI, pages 6259–6266, 2020.
20. Shoujin Wang, Liang Hu, et al. Attention-based transactional context embedding for next-item recommendation. In AAAI, pages 2532–2539, 2018.
21. Wen Wang, Wei Zhang, et al. Beyond clicks: modeling multi-relational item graph for session-based target behavior prediction. In WWW, pages 3926–3932, 2020.
22. Xiaocui Li, Hongzhi Yin, et al. Semi-supervised Clustering with Deep Metric Learning and Graph Embedding. World Wide Web 23(2), 781–798, 2020.
23. Yaqing Wang, Chunyan Feng, et al. User identity linkage across social networks via linked heterogeneous network embedding. World Wide Web 22(6), 2611–2632, 2019.
24. Zekai Wang, Hongzhi Liu, et al. Unified embedding model over heterogeneous information network for personalized recommendation. In IJCAI, pages 3813-3819, 2019.