Multilevel Feature Interaction Learning for Session-Based Recommendation via Graph Neural Networks

Ming He(✉), Tianshuo Han, and Tianyu Ding

Beijing University of Technology, Beijing, China
heming@bjut.edu.cn, {hants,dingtianyu}@emails.bjut.edu.cn

Abstract. Predicting users' actions based on anonymous sessions is a challenging problem due to the uncertainty of user behavior and limited information. Recent advances in graph neural networks (GNN) have led to a promising approach for addressing this problem. However, existing methods have three major issues. First, they are incapable of modeling the transitions between inconsecutive items. Second, they are infeasible for learning the cross-feature interactions when learning the item relationships. Third, very few models can adapt to the improvement of embedding quality to help improve recommendation performance. Therefore, to address these issues, we propose a novel model named ***M****ultilevel* ***F****eature* ***I****nteractions* ***L****earning* (MFIL) that effectively learns item and session representation using GNN. By leveraging item side information, e.g., brands and categories, MFIL can model transitions between inconsecutive items in the session graph (session-level). We further design hierarchical structures to learn the feature interactions, which is effective to estimate the importance weights of different neighboring items in the global graph (global-level). In addition, an effective learning strategy is employed to enhance MFIL's capability, and it performs better than the classic regularization methods. Extensive experiments conducted on real-world datasets demonstrate that MFIL, significantly outperforms existing state-of-the-art graph-based methods.

Keywords: Graph neural networks · Recommender systems · Session-based recommendation

1 Introduction

The recommender systems play a crucial role in helping users target their interests. Conventional recommendation approaches usually rely on clicks feedback and may perform poorly in real-world scenarios. Consequently, session-based recommendation has attracted considerable interest recently, which predicts the next interacting item based on users' behaviors. Recently, the graph-based approaches [13,30] use graph neural networks (GNN) to capture the higher-order interaction and get the item representations. However, these methods can not effectively address the following issues in a session-based recommendation.

T. Di Noia et al. (Eds.): ICWE 2022, LNCS 13362, pp. 31–46, 2022.
https://doi.org/10.1007/978-3-031-09917-5_3

The first issue is the insufficient modeling of item transitions in the session. Previous works are based only on the pairwise item-transitions [22,23,31], and do not fully model the transitions between inconsecutive items. The idea of these works is to calculate how consecutive items communicate with each other. Some recent attempts have employed multilayer structures [9,17], but they only use the direct connections between consecutive items. Hence, there is a need for more effective exploration to learn the transitions of inconsecutive items.

Second, current recommendation methods ignore feature interactions, and this may affect recommendation performance in two aspects: 1) these methods may not be expressive enough to capture complex patterns of feature interactions. For example, the element-wise product [11,14,16] and nonlinear transformation [20,24] operations lack the ability of learning feature interactions [5]; 2) these models do not make full use of co-occurrence information [15,26]. Thus, exploring more effective structures and using the co-occurrence information when learning the feature interactions will be helpful.

The third is that they fail to adapt themselves to the embedding quality improvement, which may limit the further optimization of weight parameters and decrease model-learning capability. In the literature, some methods [8] have proposed employing regularization techniques (e.g., *learning rate decay and dropout*) to enhance performance. However, these strategies prevent the further improvement of model performance when embedding quality is improved to a certain extent. Hence, a more effective learning strategy to enhance the model's learning capability rather than merely prevent overfitting is required.

The main contributions of this work are summarized as follows:

- We leverage side information (e.g., *brands or categories*, as we mentioned in the Abstract) to learn the transitions of items within a session graph. An adjacent matrix is constructed, and it is capable of propagating information between relevant items, even if they are inconsecutive.
- We propose a hierarchical graph model to learn feature interactions and use the co-occurrence information, which can estimate the importance weights of different items in the global graph.
- A more effective learning strategy is employed, which can adapt the model to the improvement of embedding by adjusting internal structures, and performs better than the classical regularization methods, e.g., *learning rate decay.*
- Extensive experiments show that MFIL achieves significant improvements over state-of-the-art graph-based baseline models.

2 Related Work

2.1 Graph Neural Networks (GNN)

The most critical aspect of the aforementioned methods is the network structures because it directly decides the recommendation performance [31]. So, we mainly introduce the GNN techniques here. Among various GNN architectures [21], gated GNN (GGNN), graph attention network (GAT), and graph convolutional

network (GCN) are widely used. GGNN-based methods [8,9,25] adopt long short-term memory (LSTM) in the update step. GAT-based methods [10,16] update the vector of node by a weighted summation of its neighboring items. GCN-based methods [6,28] generally aggregate feature information using convolutional neural networks. The other methods [7,15,31] usually adopt mean/pooling for aggregation and concatenation/nonlinear transformation for update. *Different from these works, we employ hierarchical structures for information propagation, which effectively estimate the importance weights between items.*

2.2 GNN for Session-Based Recommendation

The use of GNN techniques in session-based recommender systems is attracting increasing attention recently. The core idea of these works is to capture transition patterns in sessions. For example, SR-GNN [22] considers the transitions between items, and combines long-term preferences with current interests. FGNN [10] captures item transitions and employs the readout function to learn the session embedding. DGTN [31] and GCE-GNN [16] model item transitions in not only the current session and but also the neighboring sessions. TAGNN [25] designs a target-aware attention module to learn interest representations with different target items. MA-GNN [7] integrates the static, dynamic, and long-term user preferences with item co-occurrence patterns. *Compared to previous works that construct the session graph using edges between two consecutive items, we propose building a graph with side information and capture transition patterns between inconsecutive items.*

2.3 Regularization Techniques

Regularization techniques have been investigated extensively because of their capability to avoid overfitting; a widely used approach is l_p regularization [10]. Moreover, item sharing, dropout, layer normalization [4], gradient clipping, max norm regularization [19], and the learning rate decay [16] are effective in practice. Many new regularization techniques have been developed recently. Stochastic shared embedding [18,19], stochastically replaces embeddings with another embedding with some pre-defined probability. A graph-based regularization approach that serves as a counterpart of the l_2 regularization has been proposed in [29]. In addition, a regularization-based approach that optimizes for robustness on rule-based (a set of expert-defined categorical rules) input perturbations has been proposed in [1]. *In this paper, we apply a more effective learning strategy to enhance MFIL's capability instead of simply using classic regularization techniques, such as learning rate decay and dropout.*

3 Methodology

3.1 Problem Definition

Let $V = \{v_1, v_2, ..., v_m\}$ be the item set, where m denotes the number of items. A session is defined as an interaction list $S = \{v_1^s, v_2^s, ..., v_l^s\}$ in chronological

order, where v_i^s denotes the i-th item within session S, and l denotes the length of S, and we use $I = \{i_1^s, i_2^s, ..., i_l^s\}$ to represent the corresponding side information. The co-occurrence information of each item can be denoted as $N = \{N_\varepsilon(v_1^s), N_\varepsilon(v_2^s), ..., N_\varepsilon(v_l^s)\}$, where $N_\varepsilon(v_i^s)$ represents the co-occurrence items of v_i^s in neighbor sessions. The details of $N_\varepsilon(\cdot)$ will be described in Subsect. 3.4. Formally, the task is to predict the top-N items that the user is likely to interact with at $(l+1)$-th step.

3.2 Model Overview

We illustrate MFIL in Fig. 1, which contains three main components: a *session-level* item representation learning module, a *global-level* item representation learning module, and a *session representation* learning module. The *session-level* module learns item embeddings with the *side information* I in the session graph, and the *global-level* module incorporates information from neighboring items in the global graph based on the *co-occurrence information* N. The *session representation* learning module generates a representation of the session by aggregating the representations from the two *learning modules*.

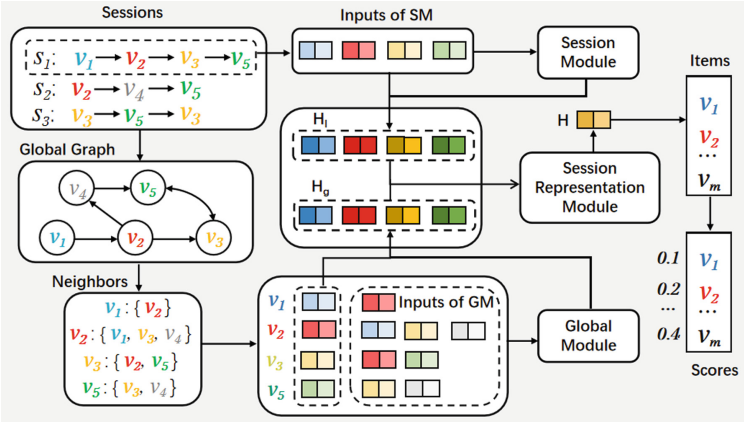


Fig. 1. Overall framework of MFIL. At first, the *global graph* and *neighbors* of session s_1 are extracted from the given *sessions* simultaneously and then fed into the *session module* and *global module* to learn item embeddings H_l and H_g. Then, they are fed into the *session represent module* to assemble the representation of session s_1, which is denoted as H.

3.3 Session-Level Item Representation Learning Module

In this subsection, we introduce how to obtain the representation $H_l \in \mathbb{R}^{L \times d}$ of items from *session-level*, and different from Subsect. 3.1, we use l to represent session-level. First, we extract the latest L items in s in chronological order, which is abbreviated as $S = \{v_1^s, v_2^s, ..., v_L^s\}$. Let $M \in \mathbb{R}^{m \times d}$ denotes the

learnable item embedding matrix with d as the latent dimensionality. The session S is represented as $E_l = \{m^s_{v_1}, m^s_{v_2}, ..., m^s_{v_L}\} \in \mathbb{R}^{L\times d}$. We also employ the corresponding side information $I = \{i^s_1, i^s_2, ..., i^s_L\} \in \mathbb{R}^L$.

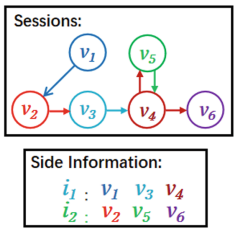
We build the adjacent matrices to model the item transitions leveraging item side information, as we mentioned in the first issue in the introduction. For example, consider a session $S = \{v_1, v_2, v_3, v_4, v_5, v_4, v_6\}$ and the corresponding side information (e.g., *brands or categories*) $I = \{i_1, i_2, i_1, i_1, i_2, i_1, i_2\}$, the session graph, and adjacent matrices are shown in Fig. 2. In the left matrix A^1, for simplicity, we use symbols like *1, 2, 3, and 4* to represent the *self-loop, in-come, out-come, and in-out* relationships between two consecutive items [16], respectively. In the right matrix A^2, we set $A^2_{ij} = 1$, if v_i and v_j belong to the same *brand* or *category*. If we consider only the relationships in matrix A^1, we will be incapable of capturing the transitions between v_1 and $v_3(v_4)$, as well as v_2 and $v_5(v_6)$, v_* represents the item in the sequence. With the help of different matrices, we can learn the transitions of consecutive and inconsecutive items.

Information Propagation: We use the attention mechanism to learn the weights of different items $\alpha \in \mathbb{R}^{L\times L}$ in the session graph. To model the feature interactions, we repeat E_l in the first and second dimensions L times to obtain $E^1_l \in \mathbb{R}^{(L\times L)\times d}$ and $E^2_l \in \mathbb{R}^{L\times(L\times d)}$, respectively, and then model the feature interactions by concatenation operation and multilayer perception (MLP):

$$\begin{aligned} a_1 &= (E^1_l \times E^2_l) \,||\, E^1_l, \\ a_2 &= LeakyReLU(a_1 W_1 + b_1)W_2 + b_2, \\ a_3 &= LeakyReLU(a_2 W_3 + b_3)W_4 + b_4, \end{aligned} \tag{1}$$

where $||$ denotes the vector concatenation operation, $\times$ denotes the element-wise multiplication operation, $W_1 \in \mathbb{R}^{2d\times 2d}$, $W_2, W_4 \in \mathbb{R}^{2d\times d}$, $W_3 \in \mathbb{R}^{d\times 2d}$ and b_* are weight matrices and biases for the two layers, $a_1 \in \mathbb{R}^{L\times L\times 2d}$, a_2 and $a_3 \in \mathbb{R}^{L\times L\times d}$, **only a_3 will be used in subsequent stages**, and it is calculated by a_1 and a_2.

A^1	1	2	3	4	5	6
1	1	2				
2	3	1	2			
3		3	1	2		
4			3	1	4	2
5				4	1	
6				3		1



A^2	1	2	3	4	5	6
1			1	1		
2					1	1
3	1			1		
4	1		1			
5		1				1
6		1			1	

Fig. 2. Session graph of s and its adjacent matrices. In the left matrix (any two directly adjacent items) A^1, for simplicity, we use *1, 2, 3, and 4* to represent the *self-loop, in-come, out-come, and in-out* relationship, and the matrix shows the adjacency of items in the sequence. In the right matrix A^2, $A^2_{ij} = 1$ represents that v_i and v_j belong to the same brand or category.

In the following, we will use the a_3 to get different α_*, and then get the final weight matrix $\alpha \in \mathbb{R}^{L\times L}$. For each relationship (e.g., *in-out or 'adjacent'*, as shown in Fig. 2) in the session graph, MFIL calculates and gets the corresponding matrix α_* similarly:

$$\alpha_* = a_3 W_*^{\alpha}, \tag{2}$$

where $W_*^{\alpha} \in \mathbb{R}^{d\times 1}$ and $\alpha_* \in \mathbb{R}^{L\times L}$. We use $*$ to represent the *self-loop, in-come, out-come, in-out, and 'adjacent'* relationship, respectively, and **please note that each relationship has a corresponding weight matrix W_* and** α_*. We only keep the items in α_* if A_{ij}^1=1 (when $*$ represents self-loop, etc.) or A_{ij}^2=1 (for the 'adjacent' relationship).

We stack different α_* in the order of *self-loop, in-come, out-come, in-out, and 'adjacent'* to obtain the final weight matrix $\alpha \in \mathbb{R}^{L\times L}$, we believe that the 'adjacent' relationship is more important than the others, and the in-out relationship is different from the in-come (or out-come) relationship. The first L represents the length of the session, and the second represents the relationships between a specific item and other L items in the session. We use a toy example to explain the stack operation. For vectors $v_1 = [1, 1, 0, 0]$ and $v_2 = [0, 2, 2, 0]$, if we stack them in the order of v_1 and v_2, we could obtain $v_{stack} = [1, 2, 2, 0]$.

Information Aggregation: the representation of items in the session graph $H_l \in \mathbb{R}^{L\times d}$ is defined with $\beta \in \mathbb{R}^1$ as follows:

$$H_l = \beta(softmax(\alpha)E_l) + E_l. \tag{3}$$

As we mentioned in the introduction, to adapt the model to the embedding quality improvement, we will update β. When the embedding quality is improved to a certain extent, the input E_l already contains explicit semantic information, and we should focus more on E_l for the follow-up tasks, especially in the later stage of training, so we adjust β during training to ensure that the model can be further optimized. It performs better than some classic regularization, and we will discuss it in Subsect. 4.4. We update β with initial value β_0 at fixed *rate* when the training *epoch* is larger than $epoch_l$:

$$\beta = \beta_0 - (epoch - epoch_l) \times rate, \quad epoch \geq epoch_l. \tag{4}$$

3.4 Global-Level Item Representation Learning Module

In this subsection, we introduce how to incorporate information from the global-level [16] and obtain the representation $H_g \in \mathbb{R}^{L\times d}$ of items. First, we use $E_l = \left\{m_{v_1}^s, m_{v_2}^s, ..., m_{v_L}^s\right\} \in \mathbb{R}^{L\times d}$ to denote items in the current session, and $N = \{N_\varepsilon(v_1^s), N_\varepsilon(v_2^s), ..., N_\varepsilon(v_L^s)\} \in \mathbb{R}^{L\times n\times d}$ to denote the co-occurrence information in the global session, where $N_\varepsilon(v_i^s) \in \mathbb{R}^{n\times d}$ represents the n ε-neighbor items of v_i^s with d as the latent dimensionality [16]. We also employ the co-occurrence information which can be represented as $w = \{w_1^s, w_2^s, ..., w_L^s\}$, where $w_i^s \in \mathbb{R}^1$ represents the co-occurrence times of v_i's n neighbor items in the global graph. For example, consider the sessions $s_1 = \{v_1, v_2, v_3, v_5\}$ and its neighbor sessions (which also contains v_2) $s_2 = \{v_3, v_5, v_2, v_6\}$ and $s_3 = \{v_4, v_2, v_6\}$, as shown in the

center of Fig. 3, for the target item v_2, we can obtain $N_2(v_2) = \{v_3, v_5, v_6, v_1, v_4\}$ and the corresponding $w = \{2, 2, 2, 1, 1\}$, as shown on the left of Fig. 3. The 1-neighbors of v_2 are $\{v_6, v_1, v_3, v_4, v_5\}$ with co-occurrence times $\{2, 1, 1, 1, 1\}$. We also show the global graph of v_2 on the right of Fig. 3.

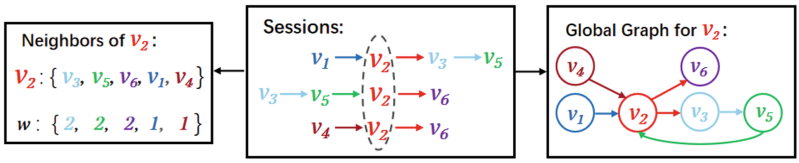


Fig. 3. Sessions that contain v_2. The 2-neighbors of v_2 are $\{v_3, v_5, v_6, v_1, v_4\}$ with co-occurrence times $\{2, 2, 2, 1, 1\}$. The global graph of v_2 is also shown on the right.

Information Propagation: We learn the weights of items in $N_2(v_2)$ for each item v_i in the session. First, we can obtain the representation of current session $s \in \mathbb{R}^d$, which is obtained by computing the average of E_l:

$$s = \frac{1}{|E_l|} \sum_{m_i \in E_l} m_i. \tag{5}$$

To model the feature interactions, we repeat s in the first dimension L times to obtain $s^{'} \in \mathbb{R}^{L \times d}$, and then repeat $s^{'}$ n times in the second dimension to obtain $s^{''} \in \mathbb{R}^{L \times n \times d}$, and model the feature interactions between each item with co-occurrence information w as follows:

$$\begin{aligned} a_4 &= [(s^{''} \times N) \,||\, sigmoid(w)]W_5, \\ a_5 &= mean(a_4 N, -2), \\ a_6 &= \gamma a_5 + E_l, \end{aligned} \tag{6}$$

where $W_5 \in \mathbb{R}^{(d+1)\times 1}$, $a_4 \in \mathbb{R}^{L \times n \times 1}$ models the attention scores of the items in N, $a_4 N \in \mathbb{R}^{L \times n \times d}$, $a_5 \in \mathbb{R}^{L \times d}$ represents the average of N, and $a_6 \in \mathbb{R}^{L \times d}$ is the information propagated from N (a set of neighbours), please note that only a_6 will be used in subsequent stages, and it is calculated by a_4 and a_5. γ is a hyperparameter to be tuned. Notably, for matrix $X \in \mathbb{R}^{a \times b \times c}$, $mean(X, -2) \in \mathbb{R}^{a \times c}$.

As we mentioned in the introduction, in order to adapt the model to the embedding quality improvement, we only reserve the top-k scores in a_4 in the second dimension for the same purpose in the previous section, and we calculate k when the training *epoch* is larger than $epoch_g$ as follows:

$$k = n - epoch//rate, \quad epoch \geq epoch_g, \tag{7}$$

where // is the division operation, for example, $5//2 = 2$ and $7//2 = 3$.

Information Aggregation: We aggregate E_l and a_6 with $W_6 \in \mathbb{R}^{2d\times d}$ to obtain the representation $H_g \in \mathbb{R}^{L\times d}$ of items in the current session graph:

$$H_g = dropout([a_6 \,||\, E_l]W_6). \tag{8}$$

3.5 Session Representation Learning and Training

With $W_7 \in \mathbb{R}^{2d\times d}$, we can obtain a representation $H \in \mathbb{R}^{L\times d}$ by aggregating $H_l \in \mathbb{R}^{L\times d}$ and $H_g \in \mathbb{R}^{L\times d}$, and we also add a learnable position embedding matrix $P = [p_1, p_2, ..., p_L] \in \mathbb{R}^{L\times d}$ to suggest the importance of each item:

$$H = tanh([(H_l + H_g) \,||\, P]W_7). \tag{9}$$

Following previous works [8,16], we use a soft-attention mechanism and multi-head attention [10] to learn the corresponding weights $\omega^* \in \mathbb{R}^{L\times 1}$ of each item in H, and obtain the final session representation $E \in \mathbb{R}^{1\times d}$ as follows:

$$\begin{aligned} \omega^k &= sigmoid(HW_8^k + s'W_9^k + b^k)q^k, \\ E &= \frac{1}{K}\sum_{k=1}^{K}(\sum_{i=1}^{L}\omega_i^{k\top}H_i), \end{aligned} \tag{10}$$

where K is the number of heads, $W_8^*, W_9^* \in \mathbb{R}^{d\times d}$, and $q^* \in \mathbb{R}^{d\times 1}$, b^* is the bias and $\top$ denotes the transpose of the vector or the matrix.

We obtain the interaction probability $\widehat{y} \in \mathbb{R}^m$ of each item in M, and adopt binary cross-entropy loss function with one-hot ground truth $y \in \mathbb{R}^m$ as follows:

$$\begin{aligned} \widehat{y} &= softmax(EM^\top), \\ \mathcal{L}(\widehat{y}) &= -\sum_{i=1}^{m} y_i log(\widehat{y}_i) + (1-y_i)log(1-\widehat{y}_i). \end{aligned} \tag{11}$$

4 Experiments

We have conducted experiments to answer the following questions:

- RQ1: How well does MFIL outperform state-of-the-art models?
- RQ2: Do the transitions between inconsecutive items improve the performance?
- RQ3: What is the role of each structure in the proposed model?
- RQ4: Can the proposed learning strategy enhance the performance?

Datasets. We conduct experiments using two million-scale datasets, Diginetica[1] and JDATA[2], for the category information in these two datasets is more diverse, compared with *Yoochoose*, and we abandoned it for the same reason. Please note that JDATA is also a large-scale session-based recommendation dataset.

[1] https://competitions.codalab.org/competitions/11161.
[2] https://jdata.jd.com/html/detail.html?id=8.

The average number of brands in the interaction sequence is 1.95 for Diginetica and 2.54 for JDATA; each training sample is truncated at length 20. We retain the first one million records in JDATA to generate JD-1 m dataset due to computing power limitations. Following previous works [22], we also simulate a sparse dataset JD-100 k through retaining the first 100 k records in the origin JDATA dataset. We took the earlier 90% and subsequent (most recent) 10% user behaviors as the training and test set, respectively. We search the parameters on a validation set which is a random 10% subset of the training set. Statistical details are shown in Table 1.

Table 1. Dataset statistics (after preprocessing)

Dataset	Train sessions	Test sessions	Items	Avg. length
Diginetica	167,506	18,842	33,444	4.36
JD-1m	533,981	60,484	66,976	4.96
JD-100k	43,411	5126	12,346	4.62

Baselines. Due to the space limitation, we ignore the classic models like RNN or KNN, which had been fully explored in [16,31]. We compare our method with classic method SR-GNN and three latest state-of-the-art models:

- **SR-GNN** [22]: It applies a gated graph convolutional layer to obtain item embeddings, followed by a self-attention of the last item to combine long-term and current preferences of sessions to predict users' next actions.
- **FGNN** [10]: It models the item transitions via a weighted attention graph layer, propose a readout function to learn the embedding of the whole session, and incorporates the edge weight of neighboring items.
- **DGTN** [31]: It models item transitions within not only the current session but also the neighbor sessions. They are integrated into a graph, and then the embeddings are fed into the fusion function to obtain the final embedding.
- **GCE-GNN** [16]: Similar to **DGTN**, it exploits item transitions over all sessions in a more subtle manner for inferring the preference of the current session and aggregates the representations with a soft-attention mechanism.

Evaluation Metrics and Parameter Settings. By following previous baselines, we adopt the same widely used metrics P@N and M@N [22]. For our model, we use the Adam optimizer with the initial learning rate of 10^{-3} and the linear schedule decay rate of 0.1 for every 3 epochs. The dimension of the latent vectors is set to 64, and the batch size is set to 128. We set the number of neighbors and the maximum distance of adjacent items ϵ to 12 and 2, respectively. The $epoch_l$ and $epoch_g$ are set to 4. The *rate* used in Eq. (4) is set to 0.01, and the β_0 is set to 0.2, γ is set to 0.2. The dropout ratio is set to 0.2 for Diginetica and 0.5 for JDATA, respectively. For the other baselines, to make a fair comparison, we adjust the hyperparameters (e.g., *learning rate, dropout ratio, and numbers of attention heads*) to obtain a better result.

4.1 Performance Comparisons (RQ1)

In Table 2, the best result in each column is marked in bold, and the second-best one is 'underlined'. 'Impro.(%)' denotes the percentage improvement of MFIL with respect to the best performing value in the baselines.

Table 2. Recommendation performance. All numbers are in percentage

	Diginetica				JD-1 m				JD-100 k			
	$P@10$	$M@10$	$P@20$	$M@20$	$P@10$	$M@10$	$P@20$	$M@20$	$P@10$	$M@10$	$P@20$	$M@20$
SR-GNN	41.24	14.77	54.64	16.15	28.82	13.00	37.41	14.09	18.83	5.69	21.48	9.07
FGNN	40.58	13.62	53.18	16.03	26.66	10.04	34.24	11.63	22.27	8.98	27.63	11.70
DGTN	46.21	15.89	61.21	16.54	30.28	10.41	41.17	11.72	24.80	10.08	31.16	12.31
GCE-GNN	45.71	16.37	61.84	17.38	32.78	11.35	43.32	12.70	25.63	11.61	32.85	12.01
MFIL	**51.44**	**19.88**	**69.91**	**21.09**	**35.53**	**15.67**	**45.27**	**16.73**	**29.75**	**13.83**	**37.69**	**14.39**
Impro.(%)	11.3	21.4	13.0	21.3	8.4	20.5	4.5	18.7	16.1	19.1	14.7	16.9

MFIL achieved the best performance on all the datasets compared with baselines, which demonstrates its superiority. On average, MFIL improved GCE-GNN by 11.9% (10.73%) in terms of P@10(20), and 19.3% (18.9%) in terms of M@10(20). We observe that out MFIL surpasses the classic method SR and FGNN, which means only modeling the current session are inadequate to obtain a desirable result. Moreover, we notice that though considering the neighbor sessions, DGTN and GCE-GNN are still challenged by MFIL in all cases, which actually justifies our motivations mentioned in the contribution.

We have stated the disadvantages of the four baseline models in the introduction and Subsect. 2.2, and the experiment results can validate our analysis. GCE-GNN exhibited the second-best performances more than others, probably because it did not consider the transitions of inconsecutive items and fail to model the feature interactions, which has been explored in MFIL. The DGTN slightly performed worse than GCE-GNN, except on two metrics, but still significantly outperformed SR-GNN and FGNN in most cases. Different from our model, it did not learn the transitions of inconsecutive items in the target session.

Different from MFIL, FGNN did not consider the items in the neighbor session and did not beat SR-GNN in the first two datasets, but performed better than SR-GNN in the sparse datasets JD-100k. For SR-GNN, it only obtained the second-best performance on JD-1m in terms of M@10(20). Different from these methods, our approach learns the transitions of inconsecutive items, and explores more effective structures with the help of a different learning strategy, leading to better performance.

4.2 Impact of Side Information (RQ2)

Next, we conduct experiments on the first two large datasets, Diginetica and JD-1m, to evaluate the effectiveness of learning the transitions of inconsecutive items by leveraging item side information. We design four contrast models:

- **MFIL** w/o side information: It does not use the right matrix in Fig. 2 and only considers the *self-loop*, *in-come*, *out-come*, and *in-out* relationship.
- **MFIL** summation: It only utilizes the right matrix in Fig. 2 by summation and does not consider the feature interaction in Eq. (1).
- **MFIL** w/o share: The *in-out* and *'adjacent'* relationship do not share the same weight matrix in Eq. (2), so there will be five W_*^α matrices for the *self-loop*, *in-come*, *out-come*, *in-out*, and *'adjacent'* relationship.
- **MFIL** interaction: It uses both the two matrices in Fig. 2, and when modeling the feature interactions in Eq. (1), it combines the item embedding E_l and side information embedding I to obtain a_1 used in Eq. (1).

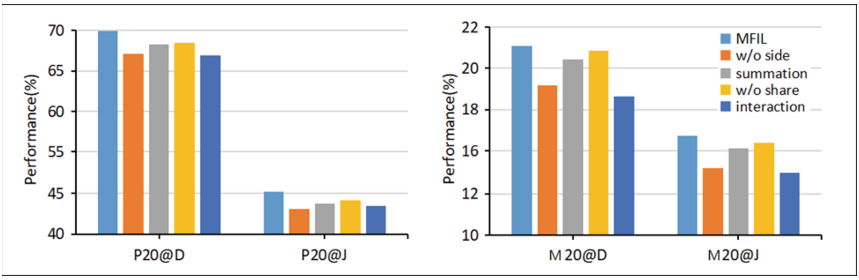


Fig. 4. Recommendation performance of MFIL with different structural designs in RQ2. All the numbers are percentage numbers with % omitted

Figure 4 shows the comparison of performance between different contrast models. The original MFIL achieves better performance. Comparing with MFIL w/o side information and MFIL summation, the original MFIL performs better on two datasets, which demonstrates the importance of side information and the 'adjacent' relationship shown in Fig. 2. The original MFIL also outperforms the last two models, MFIL w/o share and MFIL interaction. Therefore, it is less effective to introduce more parameters to describe the relationship between items, or combine the side information and learn the transitions of inconsecutive items. These results show that the original MFIL that considered about both the two kinds of relationships is more effective in balancing them.

Some possible reasons are as follows: 1) MFIL w/o side information performed worse than MFIL summation in all cases, so the 'adjacent' relationship was important for information propagation; 2) For MFIL interaction and MFIL w/o share, MFIL w/o share performed worse performance than the original MFIL. MFIL interaction's performance was even worse than that of MFIL w/o side information. So, integrating the side information into the interaction may be less efficient.

4.3 Impact of Structures (RQ3)

Then, we conduct experiments on the first two datasets, to evaluate the effectiveness of the proposed hierarchical structures in two learning modules. Specifically, we design three contrast models:

- **MFIL** w/o session: In the session learning module, it considers two kinds of relationships in Fig. 2 but does not use the MLP or concatenation operations in Eq. (1), which is similar to [16,31]. It only obtains $a_1 \in \mathbb{R}^{L \times L \times d}$:

$$a_1 = (E_l^1 \times E_l^2) \tag{12}$$

- **MFIL** w/o global: In the global learning module, it connects the embedding of items, with $W_5^{'} \in \mathbb{R}^{(2d+1)\times 1}$, the average of neighbor embedding and the co-occurrence information to obtain the a_4 used in Eq. (6) as follows:

$$a_4 = [s^{''} \,||\, N \,||\, sigmoid(w)]W_5^{'} \tag{13}$$

- **MFIL** single head: In the session representation module, we do not use the multi-head attention mechanism in (10).

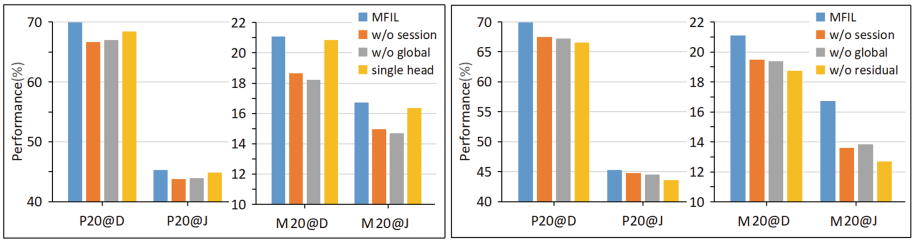


Fig. 5. Recommendation performance of MFIL with different structural designs in RQ3 and RQ4. All the numbers are percentage numbers with % omitted

Figure 5 (a) shows the comparison of performance between different contrast models. The original MFIL achieved better performance and demonstrated the superiority of learning feature interactions with hierarchical structures. The core idea of the two learning modules is to obtain better weight scores of the neighbor items in the current session (or neighbor sessions), and we introduced the feature interactions to model the weight scores more precisely. Compared with the first two methods, MFIL single head usually had better performance for session-based recommendation. This result demonstrated the effectiveness of aggregating different representations from different levels. The first two methods obtained similar performance, indicating that they were equally important.

4.4 Impact of Learning Strategy (RQ4)

We conducted experiments on two large datasets and reported the evaluation score at the end of each epoch to evaluate the effectiveness of the proposed learning strategy:

- **Learning** w/o session: In the session learning module, it ignored embedding quality and did not calculate the β in (3).
- **Learning** w/o global: In the global learning module, it simply reserved all scores in a_4 and ignored the k in Eq. (7).
- **MFIL** w/o residual: It ignored the residual connection operation in Eq. (3) and E_l when computing a_6 in (6).

Figure 5 (b) shows the comparison between different contrast models. Compared with the first two methods, the proposed learning strategies can improve the capability of MFIL. This actually justified our motivation to adopt the model to the embedding quality improvement. We also found that even in the later stage of training (16th epoch or later in JD-1m), the original MFIL could still optimize parameters and achieved better performance, whereas the third model began to overfit in the 10th epoch and stopped in the 13th epoch, as shown in Fig. 6. For MFIL w/o residual, we found that it is reasonable to introduce the proposed learning strategy.

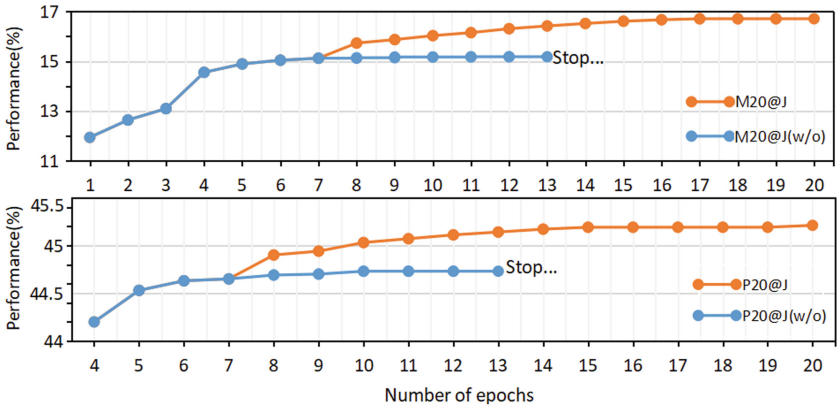


Fig. 6. Recommendation performance of MFIL with different learning strategies. All the numbers are percentage numbers with % omitted

5 Conclusion

In this paper, we propose a novel model, termed as MFIL, which learns the transitions of inconsecutive items and models feature interactions when propagating information. The experimental results show that it is reasonable to build adjacent matrices via item side information, and the hierarchical structure is able

to learn the feature interactions. We also adopt a learning strategy to enhance the model capability, which can enhance the learning capability of out model. Extensive experiments demonstrate the superiority of MFIL.

In the future, we plan to find more methods to fuse different representations [2,12]. We also try to use multiple attributes, such as price or creation time. We will also explore knowledge graphs [8] and transformers [3,24] to improve the performance and efficiency of the proposed model, as well as use the items attribute [27,32] to better propagate information.

References

1. Balashankar, A., Beutel, A., Subramanian, L.: Enhancing neural recommender models through domain-specific concordance. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 1002–1010 (2021)
2. Cen, Y., Zhang, J., Zou, X., Zhou, C., Yang, H., Tang, J.: Controllable multi-interest framework for recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2942–2951 (2020)
3. Cho, S.M., Park, E., Yoo, S.: Meantime: mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In: Fourteenth ACM Conference on Recommender Systems, pp. 515–520 (2020)
4. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE International Conference on Data Mining, ICDM, pp. 197–206. IEEE (2018)
5. Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: xdeepfm: combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1754–1763 (2018)
6. Liu, Y., et al.: Decoupled graph convolution network for inferring substitutable and complementary items. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2621–2628 (2020)
7. Ma, C., Ma, L., Zhang, Y., Sun, J., Liu, X., Coates, M.: Memory augmented graph neural networks for sequential recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 5045–5052 (2020)
8. Meng, W., Yang, D., Xiao, Y.: Incorporating user micro-behaviors and item knowledge into multi-task learning for session-based recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1091–1100 (2020)
9. Pan, Z., Cai, F., Chen, W., Chen, H., de Rijke, M.: Star graph neural networks for session-based recommendation. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 1195–1204 (2020)
10. Qiu, R., Li, J., Huang, Y.: Rethinking the item order in session-based recommendation with graph neural networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 579–588 (2019)
11. Song, W., Xiao, Z., Wang, Y., Charlin, L., Zhang, J.: Session-based social recommendation via dynamic graph attention networks. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 555–563 (2019)

12. Tan, Q., et al.: Sparse-interest network for sequential recommendation. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 598–606 (2021)
13. Tanjim, M.M., Ayyubi, H.A., Cottrell, G.W.: Dynamicrec: a dynamic convolutional network for next item recommendation. In: Proceedings of the 29th ACM International Conference on Information, pp. 2237–2240 (2020)
14. Wang, J., Ding, K., Hong, L., Liu, H.: Next-item recommendation with sequential hypergraphs. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1101–1110 (2020)
15. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 165–174 (2019)
16. 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: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 169–178 (2020)
17. Wu, L., Sun, P., Fu, Y., Hong, R., Wang, M.: A neural influence diffusion model for social recommendation. In: Proceedings of the 42nd international ACM SIGIR conference on research and development, pp. 235–244 (2019)
18. Wu, L., Li, S., Hsieh, C.J., Sharpnack, J.: Stochastic shared embeddings: data-driven regularization of embedding layers (2019). arXiv preprint, arXiv:1905.10630
19. Wu, L., Li, S., Hsieh, C.J., Sharpnack, J.: Sse-pt: sequential recommendation via personalized transformer. In: Fourteenth ACM Conference on Recommender Systems, pp. 328–337 (2020)
20. Wu, Q., et al.: Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In: The World Wide Web Conference, pp. 2091–2102 (2019)
21. Wu, S., Zhang, W., Sun, F., Cui, B.: Graph neural networks in recommender systems: a survey (2020). arXiv preprint, arXiv:2011.02260
22. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 346–353 (2019)
23. Wu, S., Zhang, M., Jiang, X., Xu, K., Wang, L.: Personalized graph neural networks with attention mechanism for session-aware recommendation (2019). arXiv preprint arXiv:1910.08887
24. Xu, C., et al.: Graph contextualized self-attention network for session-based recommendation. In: IJCAI, vol. 19, pp. 3940–3946 (2019)
25. Yu, F., Zhu, Y., Liu, Q., Wu, S., Wang, L., Tan, T.: Tagnn: target attentive graph neural networks for session-based recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1921–1924 (2020)
26. Zhang, J., Shi, X., Zhao, S.: Star-gcn: stacked and reconstructed graph convolutional networks for recommender systems (2019). arXiv preprint, arXiv:1905.13129
27. Zhang, T., Zhao, P., Liu, Y., Sheng, V.S., Xu, J.: Feature-level deeper self-attention network for sequential recommendation. In: IJCAI, pp. 4320–4326 (2019)
28. Zhang, W., Mao, J., Cao, Y., Xu, C.: Multiplex graph neural networks for multi-behavior recommendation. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2313–2316 (2020)
29. Zhang, Y., Sun, F., Yang, X., Xu, C.: Graph-based regularization on embedding layers for recommendation. ACM Trans. Inf. Syst. **39**(1), 1–27 (2020)

30. Zheng, Y., Liu, S., Li, Z., Wu, S.: Cold-start sequential recommendation via meta learner (2020). arXiv preprint, arXiv:2012.05462
31. Zheng, Y., Liu, S., Li, Z., Wu, S.: Dgtn: dual-channel graph transition network for session-based recommendation (2020). arXiv preprint, arXiv:2009.10002
32. Zhou, K., Wang, H., Zhao, W.X., Zhu, Y., Wang, S., Zhang, F., Wang: S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In: Proceedings of the 29th ACM International Conference on Information Knowledge Management, pp. 1893–1902 (2020)