Self-Adaptive Graph Neural Networks for Personalized Sequential Recommendation*

Yansen Zhang[1], Chenhao Hu[1], Genan Dai[1], Weiyang Kong[1], and Yubao Liu[1,2](✉)

[1] Sun Yat-Sen University
{zhangys7, huchh8, daign, kongwy3}@mail2.sysu.edu.cn
[2] Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
liuyubao@mail.sysu.edu.cn

Abstract. Sequential recommendation systems have attracted much attention for the practical applications, and various methods have been proposed. Existing methods based on graph neural networks (GNNs) mostly capture the sequential dependencies on an item graph by the historical interactions. However, due to the pre-defined item graph, there are some unsuitable edges connecting the items that may be weakly relevant or even irrelevant to each other, which will limit the ability of hidden representation in GNNs and reduce the recommendation performance. To address this limitation, we design a new method called Self-Adaptive Graph Neural Networks (SA-GNN). In particular, we employ a self-adaptive adjacency matrix to improve the flexibility of learning by adjusting the weights of the edges in the item graph, so as to weaken the effect of unsuitable connections. Empirical studies on three real-world datasets demonstrate the effectiveness of our proposed method.

Keywords: Sequential recommendation · Graph neural networks · Self-adaptive adjacency matrix.

1 Introduction

Sequential recommendation predicts the item(s) that a user may interact with in the future given the user's historical interactions. It has attracted much attention, since the sequential manner in user historical interactions can be used for future behavior prediction and recommendation in many real-world applications. Numerous methods based on user's past interactions have been proposed for sequential recommendation, such as the Markov Chain based methods [3, 15], the recurrent neural networks (RNNs) [5, 6, 17, 18] and the convolutional neural networks (CNNs) [19, 28, 29], etc. Due to the ability of aggregating neighborhood information and learning local structure [16], GNNs are a good match to model the user's personalized intents over time. Recently, a surge of works have employed GNNs for sequential recommendation and obtained promising results [12, 23–26]. In this paper, we also pay close attention to the GNN-based methods.

* Supported by the National Nature Science Foundation of China (61572537, U1501252)

Specifically, existing methods need to construct an item graph for all sub-sequences generated in a pre-defined manner based on the given user-item interactions. Due to the uncertainty of user behaviors or external factors [22], some items in the sequence may be weakly relevant or even irrelevant. These items are connected by some unsuitable edges, which are assigned the fixed weights in the item graph. These weights will not change during the learning process of GNN. In this way, the items will aggregate inappropriate information from neighbors and result in bad effects on the recommendation performance. For example, given a sub-sequence of a user, s = (MacBook, iPhone, Bread, iPad, Apple Pencil). The connections among the 'Bread' and the other items will negatively affect the learning of items representations and cannot effectively capture the true interests of the user. To address this issue, in our model, we design a self-adaptive adjacency matrix to weaken the effect of unsuitable connections by adjusting the weights of the edges in the item graph. The self-adaptive method can learn the weights among item 'Bread' and others based on user-item interactions. In particular, it can learn the different weights on any pair of connections in the sub-sequence, while existing methods will not. By doing so, the true relation among items can be learned in our method, and more accurate item embedding representations can be modeled for next prediction tasks.

In general, our contributions are listed below. First, we propose a GNN-based method named SA-GNN for personalized sequential recommendation. In SA-GNN, we automatically learn the weights among the items in the sequences by a self-adaptive adjacency matrix. In addition, we try to capture both local interest and global interest of user to enhance the prediction performance. For instance, the user's local interest hidden in the given sub-sequence s is the electronic product. Accordingly, the global interest of the user is often hidden in long-range sequence. Second, detailed experiments conducted on three representative datasets from real-world demonstrate the effectiveness of our method compared with the state-of-art approachs.

2 Related Work

The sequential recommendation refers to recommend which items the user will visit in the future. It usually converts a user's interactions into a sequence as input. Naturally the most conventional popular Markov Chain based methods [3, 15] capture the sequential patterns by item-item transition matrices. Recently, some deep neural network-based methods of learning sequential dynamics have been proposed due to the success of sequence learning in the field of natural language processing. On one hand, some session-based recommendation methods [5, 6, 17, 18] adopted RNNs to learn the sequential patterns. The general thought of these models is to represent user's interest into a vector with different recurrent architectures and loss functions given the historical records. Apart from RNNs, some CNN-based models [19, 27–29] are also proposed for recommendation. For instance, pioneering work [19] proposes to apply the CNNs on item embedding sequence, and capture the local contexts by the convolutional operations. On the other hand, attention mechanisms [8, 10, 17] is utilized in sequential recommendation and achieves promising performance. They adopt self-attention methods to consider interacted items adaptively. In addition, memory networks [1, 12] are utilized to memorize the important items in predicting the user's future behaviors. GNN can not only process graph structure data, but also have superiority in capturing richer information in sequence data [25]. For example, in session-based recommendation, [24] first uses GNN to capture complex relationships among items in a session sequence. GC-SAN [26] uses GNN with self-attention mechanism to process session data. MA-GNN [12] applies GNN and external memories to model the user interests. However, different from these studies, we learn the weights in the item graph by a self-adaptive adjacency matrix to achieve better learning.

3 Problem Statement

The task of sequential recommendation uses the historical user-item interaction sequences as input and predicts the next-item that the user will interact with. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ denotes a set of users, and $\mathcal{I} = \{i_1, i_2, \ldots, i_{|\mathcal{I}|}\}$ denotes a set of items, where $|\mathcal{U}|$ and $|\mathcal{I}|$ are the number of users and items, respectively. The user-item interaction sequences can be represented by a sequence $\mathcal{S}^u = (i_1^u, i_2^u, \ldots, i_{|\mathcal{S}^u|}^u)$ in chronological order, where $i_*^u \in \mathcal{I}$ represents a clicked item of the user u from $\mathcal{U}$ within the sequence $\mathcal{S}^u$. We define the sequential recommendation task by the above notations as follows. Given the historical sub-sequence $\mathcal{S}_{1:t}^u (t < |\mathcal{S}^u|)$ of M users, the goal is to recommend top K items from all of $|\mathcal{I}|$ items ($K < |\mathcal{I}|$) for every user and assess whether the items in $\mathcal{S}_{t+1:|\mathcal{S}^u|}^u$ appear in the recommended list.

4 Proposed Methodology

In this section, we present the personalized sequential recommendation model SA-GNN proposed. It consists of three components, i.e., embedding layer, local interest modeling layer and prediction layer, demonstrated in Fig. 1. We first introduce the input data and how to generate the training data with future contexts. Next, we illustrate the self-adaptive adjacency matrix and attention module to get the more accurate items representations and select representative items that can reflect user's preference, respectively. Then we could generate the user's local interest. Moreover, we model unified interests representation of the user by combining the global interest with the local interest to capture the personalized user's interests. Lastly, we present the prediction layer and how to train the SA-GNN.

4.1 Embedding Layer

To obtain local interest of the user explicitly, we generate fine-grained subsequences by splitting the historical sequence and adopting a way of sliding

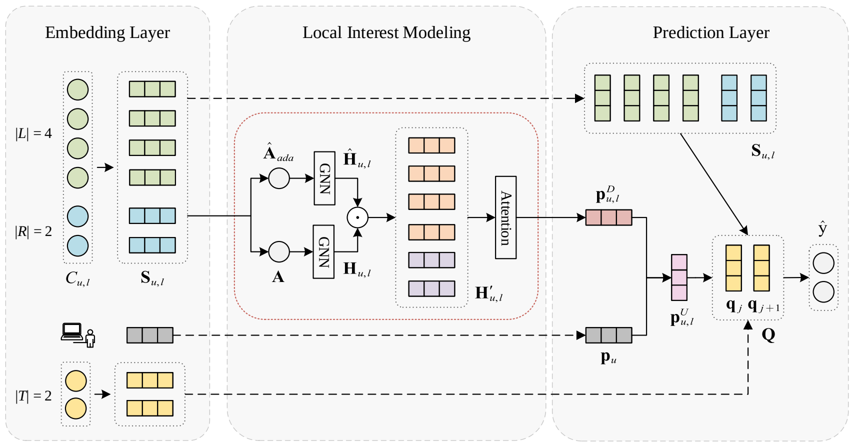

Fig. 1. The architecture of SA-GNN. In the embedding layer, the top four items of the $C_{u,l}$ represent the past contexts, and the two items below represent the future contexts. The $\odot$ represents the element-wise product.

window. Meanwhile, to better exploit the past and future contexts information, we extract every left $|L|$ successive items and right $|R|$ successive items as input and the middle $|T|$ items as the targets to be predicted for each user u, where $C_{u,l} = \{i_l, \ldots, i_{l+|L|-1}, i_{l+|L|+|T|}, \ldots, i_{l+|L|+|R|+|T|-1}\}$ is the l-th subsequence of the user u. And items in $T_{u,l} = \{i_{l+|L|}, \ldots, i_{l+|L|+|T|-1}\}$ represent the corresponding targets. The input is a sequence of $|L| + |R|$ items. We embed every item $i \in \mathcal{I}$ by an item embedding matrix $\mathbf{E} \in \mathbb{R}^{d \times |\mathcal{I}|}$, where d and $|\mathcal{I}|$ are the item embedding dimension size and the number of items, respectively. Then the item sub-sequence $C_{u,l}$ embeddings for user u are represented as follows:

$$\mathbf{S}_{u,l} = [\mathbf{e}_l, \cdots, \mathbf{e}_{l+|L|-1}, \mathbf{e}_{l+|L|+|T|}, \cdots, \mathbf{e}_{l+|L|+|R|+|T|-1}] \tag{1}$$

To distinguish the input item embeddings, we use the $\mathbf{Q} \in \mathbb{R}^{d \times |\mathcal{I}|}$ to represent the output item embeddings. Similar to the item embeddings, for a user u, we also have an user embedding $\mathbf{p}_u \in \mathbb{R}^{d'}$, and d' is the user embedding dimension size.

4.2 Local Interest Modeling

A user's local interest is based on several recently visited items in a short-term period and reflects the user's current intent [12]. Here we will further introduce how to apply the self-adaptive adjacency matrix with GNN and attention mechanism for modeling user's local interest. First, we illustrate that how to get better item representations by the self-adaptive adjacency matrix in detail. Second, with an attention module, we model the user's local interest by considering the different important score of items in the sub-sequence.

Self-Adaptive Graph Neural Network To our knowledge, the existing popular graph-based methods [12, 24, 26] often use the same rule to construct the item graph, which means they capture the relation between items in all generated sub-sequences by constructing a fixed graph. To weaken the negative effects of unsuitable connections in the item graph, we propose a self-adaptive adjacency matrix $\hat{\mathbf{A}}_{ada} \in \mathbb{R}^{(|L|+|R|)\times(|L|+|R|)}$. We use this self-adaptive adjacency matrix $\hat{\mathbf{A}}_{ada}$ to learn the dependencies between items automatically in an end-to-end manner. Unlike the previous methods which explicitly learn the embeddings of items, our proposed $\hat{\mathbf{A}}_{ada}$ can learn the relationship between items implicitly and does not require any prior knowledge. By doing so, we let the model adjust weights and discover true item dependencies. We achieve this by randomly initializing a matrix with learnable parameters. Then we use the *tanh* activation function to restrict the values in the matrix ranging from -1 to 1. The layer-wise propagation rule in matrix form as follows.

$$\hat{\mathbf{H}}_{u,l}^{(r)} = tanh(\mathbf{W}_{h_1}\mathbf{S}_{u,l}^{(r)}\hat{\mathbf{A}}_{ada}) \tag{2}$$

where $\mathbf{W}_{h_1} \in \mathbb{R}^{d\times d}$ controls the weights in the GNN. $\mathbf{S}_{u,l}^{(r)} \in \mathbb{R}^{d\times(|L|+|R|)}$ is the final hidden state of sub-sequence $C_{u,l}$ after propagation step r and $\mathbf{S}_{u,l}^{(0)} = \mathbf{S}_{u,l}$.

To better model the accurate item embeddings, we also consider the existing GNN methods. The item graph construction process as follows. For every item in interacted sequences, we extract two subsequent items and build edges between them. According to the process of generating sub-sequences mentioned before, we will not add edges for items in the middle of the sequence unless the length of target $|T|$ is less than the number of subsequent items to be added edges. We apply this for every user, and for all users with the item pairs extracted, we count the total number of edges. Then we perform row regularization on the adjacency matrix. We use $\mathbf{A} \in \mathbb{R}^{(|L|+|R|)\times(|L|+|R|)}$ to represent the adjacency matrix. Similarly, we use the $\mathbf{A}$ to aggregate the neighboring information as follows:

$$\mathbf{H}_{u,l}^{(r)} = tanh(\mathbf{W}_{h_2}\mathbf{S}_{u,l}^{(r)}\mathbf{A}) \tag{3}$$

where $\mathbf{W}_{h_2} \in \mathbb{R}^{d\times d}$ is the weights in the GNN. Then we incorporate these two embeddings by the element-wise product.

$$\mathbf{H}_{u,l}^{\prime(r)} = \hat{\mathbf{H}}_{u,l}^{(r)} \odot \mathbf{H}_{u,l}^{(r)} \tag{4}$$

After the propagation of R layers, we can get the final latent vector $\mathbf{H}_{u,l}^{\prime(\mathrm{R})}$ of each sub-sequence $C_{u,l}$. For simplicity, we mark $\mathbf{H}'_{u,l}$ rather than $\mathbf{H}_{u,l}^{\prime(\mathrm{R})}$. The final latent state of each sub-sequence not only mines its own characteristics, but also gathers its R-order neighbors' information.

Attention Module As we described before, to infer the user's local preference better, we capture the different important scores for each item embeddings generated by GNN. Inspired by [12, 13, 20], we assign each item embedding with an attention weight vector by an importance score matrix.

$$\mathbf{S}'_{u,l} = softmax(\mathbf{W}_{a_2}(tanh(\mathbf{W}_{a_1}\mathbf{H}'_{u,l} + \mathbf{b}_{a_1}) + \mathbf{b}_{a_2})) \tag{5}$$

where $\mathbf{S}'_{u,l} \in \mathbb{R}^{d_a \times (|L|+|R|)}$ is the matrix of attention weight, $\mathbf{W}_{a_1} \in \mathbb{R}^{d \times d}$, $\mathbf{W}_{a_2} \in \mathbb{R}^{d_a \times d}$, $\mathbf{b}_{a_1} \in \mathbb{R}^{d}$, and $\mathbf{b}_{a_2} \in \mathbb{R}^{d_a}$ are the parameters. The d_a represents that we want to assign the important scores from the embeddings with d_a aspects in $\mathbf{H}'_{u,l}$.

By multiplying sub-sequence embeddings with the attention weight matrix, we can represent the matrix form of a sub-sequence:

$$\mathbf{Z}_{u,l} = \mathbf{S}'_{u,l}\mathbf{H}'^{\mathrm{T}}_{u,l} \tag{6}$$

where $\mathbf{Z}_{u,l} \in \mathbb{R}^{d_a \times d}$ is the matrix representation of the sub-sequence. Then we have a *Avg* (average) function to aggregate the sub-sequence matrix representation into a vector representation. So we can obtain the user's local interest representation as follows:

$$\mathbf{p}^{D}_{u,l} = Avg(tanh(\mathbf{Z}_{u,l})) \tag{7}$$

4.3 Unified Interest Modeling

The user's global interest reflects the intrinsic characteristics of a user and can be represented as the embedding $\mathbf{p}_u$. So we concatenate the local and global interest of the user, and then adopt the linear transformation operation to obtain the unified personalized representation of the user:

$$\mathbf{p}^{U}_{u,l} = [\mathbf{p}^{D}_{u,l}; \mathbf{p}_u] \cdot \mathbf{W}_u \tag{8}$$

where $[\cdot;\cdot] \in \mathbb{R}^{d+d'}$ denotes vertical concatenation, and matrix $\mathbf{W}_u \in \mathbb{R}^{(d+d')\times d}$ is the weight of linear transformation.

4.4 Prediction and Training

As shown in [7, 11, 12], it is important for the sequential recommendation to consider the effect of item co-occurrence. To capture co-occurrence patterns of the item, following [11], we also adopt the inner product to model the item relations between the input item embeddings and the output item embeddings:

$$\sum_{\mathbf{e}_k \in \mathbf{S}_{u,l}} \mathbf{e}_k{}^{\mathrm{T}} \cdot \mathbf{q}_j \tag{9}$$

where $\mathbf{q}_j$ is the j-th column of the embedding matrix $\mathbf{Q}$.

We combine the aforementioned factors together to infer user preference. The prediction value of user u on item j of the l-th sub-sequence is:

$$\hat{y}_{u,j} = \mathbf{p}^{U\mathrm{T}}_{u,l} \cdot \mathbf{q}_j + \sum_{\mathbf{e}_k \in \mathbf{S}_{u,l}} \mathbf{e}^{\mathrm{T}}_k \cdot \mathbf{q}_j \tag{10}$$

We adopt the Bayesian Personalized Ranking objective [14] via gradient descent to optimize the proposed model. That is, given the positive (observed) and negative (non-observed) items, we optimize the pairwise ranking:

$$\begin{aligned}\arg\min_{\mathbf{P},\mathbf{Q},\mathbf{E},\mathbf{\Theta}} \sum_{(u,S_u,j_+,j_-)\in\mathcal{D}} & -\log\sigma(\hat{y}_{u,j_+}-\hat{y}_{u,j_-})+ \\ & \lambda(||\mathbf{P}||^2+||\mathbf{Q}||^2+||\mathbf{E}||^2+||\mathbf{\Theta}||^2) \quad (11)\end{aligned}$$

where $(u, S_u, j_+, j_-) \in \mathcal{D}$ denotes the generated set of pairwise preference order, S_u represents one of the $|L| + |R|$ successive items of target user u, j_+ and j_- represent the positive items in $T_{u,l}$ and randomly sample negative items, respectively. σ is the sigmoid function, λ is the regularization term. $\mathbf{\Theta}$ denotes other learnable parameters in the GNN, $\mathbf{p}_*$, $\mathbf{q}_*$ and $\mathbf{e}_*$ are column vectors of $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{E}$, respectively.

5 Experiments and Analysis

5.1 Experiment Setup

Datasets We use three real-world datasets with various sparsities in different domains to assess the proposed model SA-GNN: *Amazon-CDs* [4], *Goodreads-Children* and *Goodreads-Comics* [8]. We also follow the general rules in [11, 12, 17, 19] to process datasets. For all datasets, we convert all the presence of a review or numeric ratings into implicit feedback of 1 and keep those with ratings out of five as positive feedback. To guarantee the quality of the dataset, following [11, 12], we remove the items with less than five ratings and the users with less than ten ratings. The statistics of the processed datasets are summarized in Table 1.

Table 1. The statistics of datasets.

Dataset	#Users	#Items	#Interactions	Avg.length	Density
CDs	17,052	35,118	472,265	27.696	0.079%
Children	48,296	32,871	2,784,423	57.653	0.175%
Comics	34,445	33,121	2,411,314	70.005	0.211%

Evaluation metrics We use both Recall@K and NDCG@K to evaluate all the methods. For each user, Recall@K (R@K) indicates the proportion of correctly recommended items that emerge in the top K recommended items. NDCG@K (N@K) is the normalized discounted cumulative gain at K, which considers the location of correctly recommended items. Here, we employ Top-K (K = 5, 10) for recommendation. For all the models, we perform five times and report the average results.

Baselines We compare our model with the following representative baselines to verify the effectiveness: The non graph-based methods include: the pairwise learning based on matrix factorization **BPRMF** [14], a GRU-based method to capture sequential dependencies **GRU4Rec** [6], a CNN-based method **Caser** [19] from both horizontal and vertical aspects to model high-order Markov Chains, bidirectional self-attention mechanism **BERT4Rec** [17], an encoder-decoder framework **GRec** [28] and a hierarchial gating network method **HGN** [11]. The graph-based methods: GNN with self-attention mechanism **GC-SAN** [26] and GNN with memory network **MA-GNN** [12].

Parameter settings To ensure the comparison is as fair as possible, we fix the length of sub-sequence to 8, the length of input to 6 and the length of targets to 2 in all methods. The latent dimension of all baseline models is set to 50 in all experiments. For GRU4Rec, we set the batch size to 50 and the learning rate to 0.001. Adopting Top1 loss can obtain better results. For Caser, we set the number of vertical filters and horizontal filters to 4 and 16, respectively. For BERT4Rec, we set the number of transformer layers and attention heads to 4 and 2, respectively. For HGN, we use the default hyperparameters following the settings in the author-provided code[1]. For GRec, we set the dilated levels to 1 and 4, and set the kernel size to 3. For GC-SAN, we set the number of self-attention blocks k and the weight factor ω to 4 and 0.6, respectively. For MA-GNN, we implement it with PyTorch and set both the number of attention dimensions and memory units to 20.

For SA-GNN, we set the same step R = 2 as MA-GNN and dimension of item $d = 100$, and set user embedding dimension $d' = 50$ for all datasets. The source code of SA-GNN is available online[2]. The value of d_a is selected from {5, 10, 15, 20}. Both λ and the learning rate are set to 0.001. We set the batch size to 4096 and adopt Adam [9] optimizer to train the model. Hyper-parameters are tuned by grid search on the validation set.

5.2 Comparisons of Performance

Table 2 illustrates comparsion among the experiment results. We have the following observations:

Observations about our model First, on all three datasets with all evaluation metrics, the proposed model SA-GNN achieves the best performance, which illustrates the superiority of our model. Second, SA-GNN obtains better results than MA-GNN and HGN. Despite MA-GNN and HGN capture both local and global user intents and consider item co-occurrence patterns, they all neglect the effect of future contexts information in modeling the user's interests. Third, SA-GNN obtains better results than GC-SAN. One possible main reason is that GC-SAN fails to capture the global item dependencies but only models the user's

[1] https://github.com/allenjack/HGN
[2] https://github.com/Forrest-Stone/SA-GNN

Table 2. Comparisons among the performance of all methods according to R@5, R@10, N@5 and N@10. In each row, the best scores are bold and the second best are underlined.

Method		BPRMF	GRU4Rec	Caser	BERT4Rec	GRec	HGN	GC-SAN	MA-GNN	SA-GNN	Improv.
CDs	R@5	1.104	1.137	1.163	2.064	1.684	2.737	2.477	2.502	**3.162**	15.53%
	R@10	2.018	2.054	2.081	3.732	2.473	4.459	4.042	4.202	**4.946**	10.92%
	N@5	1.408	1.425	1.441	2.278	2.012	3.233	2.609	2.853	**3.655**	13.05%
	N@10	1.709	1.749	1.780	2.705	2.428	3.817	3.378	3.504	**4.271**	11.89%
Children	R@5	5.178	5.773	6.689	7.622	7.506	7.825	7.801	7.664	**8.514**	8.81%
	R@10	8.674	9.182	10.706	12.016	11.863	12.218	12.183	12.123	**12.832**	5.03%
	N@5	8.164	8.697	10.993	12.961	12.248	13.001	12.988	12.511	**14.026**	7.88%
	N@10	9.004	9.588	11.633	13.518	13.025	13.538	13.526	13.190	**14.351**	6.01%
Comics	R@5	5.312	6.017	8.954	11.910	11.597	11.859	11.984	10.204	**13.273**	10.76%
	R@10	8.421	9.222	13.276	16.902	16.453	16.894	16.922	15.187	**18.458**	9.08%
	N@5	8.687	9.435	17.504	22.784	22.438	23.464	23.467	19.567	**25.952**	10.59%
	N@10	9.219	10.128	16.874	21.960	21.555	21.951	21.968	18.985	**24.180**	10.07%

local interest in a short-term window. Moreover, GC-SAN does not consider the item co-occurrence patterns. Fourth, SA-GNN gains better results than GRec and BERT4Rec. Although GRec and BERT4Rec adopt the bidirectional model to improve the learning ability, they neglect the item co-occurrence patterns between two closely related items, which can be learned by our item-item inter product. Moreover, they use the RNN or CNN to learn while we use the GNN. Fifth, SA-GNN obtains better results than Caser and GRU4Rec. One possible main reason is that Caser and GRU4Rec not consider the item attention score for various users, but only apply CNN or RNN to model the group-level representation of several successive items. Last, SA-GNN outperforms BPRMF, since BPRMF does not model the sequential patterns of user-item interactions, but only captures the global interest of the user.

Other observations First, all the results reported on dataset *CDs* is worse than the results on dataset *Children* and *Comics*. The possible main reason is that the data sparsity can reduce the performance of recommendation and dataset *CDs* is more sparse than others. Second, on all the datasets, GC-SAN, MA-GNN, HGN, and BERT4Rec achieve better results than Caser. The main possible reason is that these methods may capture more personalized user representation because they can learn different importance scores of items in the item sequence. Third, GC-SAN and HGN obtain better results than MA-GNN. This shows that considering the importance of different items is necessary for local interest modeling. Fourth, by comparing GC-SAN, HGN and BERT4Rec. We can see that the BERT4Rec and GC-SAN are worse than HGN on the *CDs* and *Children* datasets, while they are better on the *Comics* dataset. One possible reason may be that HGN with item co-occurrence patterns performs better on the more sparse datasets. On the other hand, GC-SAN outperforms BERT4Rec in all datasets. This shows the superiority of GNNs in sequential recommendation. Fifth, BERT4Rec is much better than GRec. The main reason may be that BERT4Rec distinguishs the different importance of items by the attention mechanism. Sixth, Caser achieves better results than GRU4Rec. One main reason is that Caser can learn user's global interest in prediction layer by explicitly feeding the user embeddings. Last, all the methods achieves better results than BPRMF. This shows that it is not enough to model user's global interest without capturing the user's sequential behaviors.

5.3 Ablation Analysis

To evaluate the contribution of each component for final user intent representations, we conducted experiments of different variants of SA-GNN. (1) **SA-GNN(-U)** is a kind of SA-GNN without the global interest but only with the local interest. (2) **SA-GNN(-U-Att)** is a sort of SA-GNN without the global interest or attention module. (3) **SA-GNN(-U-Att-A)** and (4) **SA-GNN(-U-Att-SA)** denote we model user's local interest by self-adaptive adjacency matrix $\mathbf{A}_{ada}$ and vanilla matrix $\mathbf{A}$, respectively. (5) **SA-GNN(-U-Att-SA+GAT)**: we replace the self-adaptive GNN in SA-GNN(-U-Att) with graph attention network (GAT) [21].

Table 3. The ablation analysis.

Method	*CDs*		*Children*		*Comics*	
	R@10	N@10	R@10	N@10	R@10	N@10
(1) SA-GNN(-U-Att-SA)	4.434	3.848	12.344	13.821	17.892	23.598
(2) SA-GNN(-U-Att-SA+GAT)	4.488	3.902	12.472	13.948	17.923	23.666
(3) SA-GNN(-U-Att-A)	4.635	4.029	12.697	14.189	18.316	24.111
(4) SA-GNN(-U-Att)	4.837	4.210	12.711	14.257	18.332	24.219
(5) SA-GNN(-U)	4.870	4.266	12.728	14.277	18.363	**24.294**
(6) SA-GNN	**4.946**	**4.271**	**12.832**	**14.351**	**18.458**	24.180

According to the results shown in Table 2 and Table 3, we make the following observations. First, comparing (1) with baseline models, we can observe that better results can be achieved by utilizing future contexts only based on GNN in *Children* and *Comics* datasets. Second, we compare (1) - (4). On one hand, compared with (1), (2) and (3) can learn different weights in the item graph, which helps improve performance. Moreover, (3) achieves better performance than (2). One possible reason may be that (2) will be limited by unsuitable connections of the pre-defined matrix, while (3) will not because of the self-adaptive adjacency matrix. On the other hand, (4) achieves significant improvement on all indicators and all datasets. This demonstrates that element-wise product can improve the ability of model representation and boost the performance for recommendation. Third, from (4) and (5), we observe that by considering the different important scores for items in the sequence, the performance can be improved. Last, The performance of (6) is better than (5), which illustrates that the user's global interest makes our model more capable of capturing personalized interests.

6 Conclusion

In this paper, we propose a self-adaptive graph neural network (SA-GNN) for sequential recommendation. SA-GNN applies GNN to model user's local interest, and utilizes a self-adaptive adjacency matrix to weaken the negative effect of unsuitable connections in the item graph. The experiments on three real-world datasets verify the effectiveness of SA-GNN. In the future, we will extend the SA-GNN to capture the user's long-term interest more efficiently.

References

1. Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., Zha, H.: Sequential recommendation with user memory networks. In: WSDM'18. pp. 108–116. ACM, Marina Del Rey, CA, USA (2018)
2. He, R., Kang, W., McAuley, J.J.: Translation-based recommendation. In: RecSys'17. pp. 161–169. ACM, Como, Italy (2017)
3. He, R., McAuley, J.J.: Fusing similarity models with markov chains for sparse sequential recommendation. In: ICDM'16. pp. 191–200. IEEE Computer Society, Barcelona, Spain (2016)
4. He, R., McAuley, J.J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: WWW'16. pp. 507–517. ACM, Montreal, Canada (2016)
5. Hidasi, B., Karatzoglou, A.: Recurrent neural networks with top-k gains for session-based recommendations. In: CIKM'18. pp. 843–852. ACM, Torino, Italy (2018)
6. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: ICLR'16. San Juan, Puerto Rico (2016)
7. Kabbur, S., Ning, X., Karypis, G.: FISM: factored item similarity models for top-n recommender systems. In: SIGKDD'13. pp. 659–667. ACM, Chicago, IL, USA (2013)
8. Kang, W., McAuley, J.J.: Self-attentive sequential recommendation. In: ICDM'18. pp. 197–206. IEEE Computer Society, Singapore (2018)
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR'15. San Diego, CA, USA (2015)
10. Liu, Q., Zeng, Y., Mokhosi, R., Zhang, H.: STAMP: short-term attention/memory priority model for session-based recommendation. In: SIGKDD'14. pp. 1831–1839. ACM, London, UK (2018)
11. Ma, C., Kang, P., Liu, X.: Hierarchical gating networks for sequential recommendation. In: SIGKDD'19. pp. 825–833. ACM, Anchorage, AK, USA (2019)
12. Ma, C., Ma, L., Zhang, Y., Sun, J., Liu, X., Coates, M.: Memory augmented graph neural networks for sequential recommendation. In: AAAI'20. pp. 5045–5052. AAAI Press, New York, NY, USA (2020)
13. Ma, C., Zhang, Y., Wang, Q., Liu, X.: Point-of-interest recommendation: Exploiting self-attentive autoencoders with neighbor-aware influence. In: CIKM'18. pp. 697–706. ACM, Torino, Italy (2018)
14. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: bayesian personalized ranking from implicit feedback. In: UAI'2009. pp. 452–461. AUAI Press, Montreal, QC, Canada (2009)
15. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In: WWW'10. pp. 811–820. ACM, Raleigh, North Carolina, USA (2010)

16. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Networks **20**(1), 61–80 (2009)
17. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In: CIKM'19. pp. 1441–1450. ACM, Beijing, China (2019)
18. Tan, Y.K., Xu, X., Liu, Y.: Improved recurrent neural networks for session-based recommendations. In: DLRS@RecSys'16. pp. 17–22. ACM, Boston, MA, USA (2016)
19. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: WSDM'18. pp. 565–573. ACM, Marina Del Rey, CA, USA (2018)
20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS'17. pp. 5998–6008. Long Beach, CA, USA (2017)
21. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. CoRR **abs/1710.10903** (2017)
22. Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q.Z., Orgun, M.A.: Sequential recommender systems: Challenges, progress and prospects. In: IJCAI'19. pp. 6332–6338. ijcai.org, Macao, China (2019)
23. Wu, S., Zhang, W., Sun, F., Cui, B.: Graph neural networks in recommender systems: A survey. CoRR **abs/2011.02260** (2020)
24. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: AAAI'19. pp. 346–353. AAAI Press, Honolulu, Hawaii, USA (2019)
25. Wu, S., Zhang, M., Jiang, X., Xu, K., Wang, L.: Personalizing graph neural networks with attention mechanism for session-based recommendation. CoRR **abs/1910.08887** (2019)
26. Xu, C., Zhao, P., Liu, Y., Sheng, V.S., Xu, J., Zhuang, F., Fang, J., Zhou, X.: Graph contextualized self-attention network for session-based recommendation. In: IJCAI'19. pp. 3940–3946. ijcai.org, Macao, China (2019)
27. Xu, C., Zhao, P., Liu, Y., Xu, J., Sheng, V.S., Cui, Z., Zhou, X., Xiong, H.: Recurrent convolutional neural network for sequential recommendation. In: WWW'19. pp. 3398–3404. ACM, San Francisco, CA, USA (2019)
28. Yuan, F., He, X., Jiang, H., Guo, G., Xiong, J., Xu, Z., Xiong, Y.: Future data helps training: Modeling future contexts for session-based recommendation. In: WWW'20. pp. 303–313. ACM / IW3C2, Taipei, Taiwan (2020)
29. Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M., He, X.: A simple convolutional generative network for next item recommendation. In: WSDM'19. pp. 582–590. ACM, Melbourne, VIC, Australia (2019)