



Exploring latent connections in graph neural networks for session-based recommendation

Fei Cai¹ · Zhiqiang Pan¹ · Chengyu Song¹ · Xin Zhang¹

Received: 25 August 2021 / Accepted: 20 June 2022 / Published online: 18 July 2022
© The Author(s), under exclusive licence to Springer Nature B.V. 2022

Abstract

Session-based recommendation, without the access to a user’s historical user-item interactions, is a challenging task, where the available information in an ongoing session is very limited. Previous work on session-based recommendation has considered sequences of items that users have interacted with sequentially. Such item sequences may not fully capture the complex transition relationship between items that go beyond the inspection order. This issue is partially addressed by the graph neural network (GNN) based models. However, GNNs can only propagate information from adjacent items while neglecting items without a direct connection, which makes the latent connections unavailable in propagation of GNNs. Importantly, GNN-based approaches often face a serious overfitting problem. Thus, we propose Star Graph Neural Networks with Highway Networks (SGNN-HN) for session-based recommendation. The proposed SGNN-HN model applies a star graph neural network (SGNN) to model the complex transition relationship between items in an ongoing session. To avoid overfitting, we employ the highway networks (HN) to adaptively select embeddings from item representations before and after multi-layer SGNNs. Finally, we aggregate the item embeddings generated by SGNN in an ongoing session to represent a user’s final preference for item prediction. Experiments are conducted on two public benchmark datasets, i.e., *Yoochoose* and *Diginetica*. The results show that SGNN-HN can outperform the state-of-the-art models in terms of Recall and MRR for session-based recommendation.

Keywords Session-based recommendation · Graph neural networks · Highway networks

This article is an extension of our conference paper titled “Star Graph Neural Networks for Session-based Recommendation” (Pan et al., 2020), which has appeared in CIKM’20. In this extension, we (1) investigate the effectiveness of the highway networks by replacing it with other aggregation methods in SGNN-HN; (2) test the sensitivity of SGNN-HN to the scale of training set as well as the hyper parameters; (3) analyze the time complexity theoretically and compare the computational cost empirically of SGNN-HN and the competitive baselines; (4) include more related works and provide a detailed analysis of the approach.

✉ Zhiqiang Pan
panzhiqiang@nudt.edu.cn

Extended author information available on the last page of the article

1 Introduction

Recommender systems can help people obtain personalized information (Adomavicius & Tuzhilin, 2005; Guo et al., 2017), which have wide applications in web search, e-commerce, etc. Many existing recommendation approaches apply users' long-term historical interactions to capture their preference for recommending future items, e.g., collaborative filtering (Zhang et al., 2020; He et al., 2017), Factorizing Personalized Markov Chains (FPMC) (Rendle et al., 2010), and deep learning based methods (Yu et al., 2016; Song et al., 2019; Wang et al., 2019).

However, for some cases such as newly registered or anonymous users, the user's long-term historical interactions are unavailable, making it challenging to capture user's preference in an accurate manner (Hidasi et al., 2016). Thus, session-based recommendation is proposed to generate recommendations merely based on the given ongoing session, where the session is typically defined as a user's sequential interactions with items within a period of time, e.g., 24 hours. Comparing to other recommendation tasks, the most significant characteristic of session-based recommendation is that the information contained in the current session is limited. Thus, how to accurately detect user's purpose from a limited number of items in the ongoing session is the key to successful session-based recommendation.

As items are organized into the session according to the chronological order, most existing methods for session-based recommendation focus on modeling the sequential information of items based on Recurrent Neural Networks (RNNs) (Hidasi et al., 2016) for capturing user's instant preference. Besides, to treat the interactions with items differently, attention mechanism is utilized alone (Liu et al., 2018; Pan et al., 2020) or applied together with other methods like RNNs to distinguish different items (Li et al., 2017; Wu et al., 2019). For example, Pan et al. (2020) propose an importance extraction module to individually estimate the item importance, and Li et al. (2017) apply an attention mechanism to capture a user's main purpose based on RNNs. However, RNNs plus an attention mechanism cannot fully take the transition relationship of items into consideration as the transition pattern is more complicated than a simple time order (Qiu et al., 2019). For instance, in an example session in Fig. 1, a user wants to buy a phone but issues clicks on a *Macbook Pro* and an *AirPods* out of curiosity. Such complex sequential interactions—with unrelated items—will mislead an RNN and attention mechanism, failing to accurately identify the user's main purpose.

To more accurately model the transition pattern of items, GNNs have been utilized to model an ongoing session (Wu et al., 2019; Qiu et al., 2019). For instance, Wu et al. (2019) first propose to transform the session into a session graph and then utilize Gated Graph Neural Networks (GGNN) to model the transition relationship between items. Furthermore, Qiu et al. (2019) propose to utilize a weighted graph attention networks (WGAT) to dynamically compute the information flow between items. However, GNN-based methods mostly propagate information between adjacent items, thus neglecting items without direct connections, making the long-distance information cannot be propagated. For instance, as shown in Fig. 1, when updating item embeddings for *iPhone XS*, only the information from recent items like *AirPods* can be propagated by GNNs, ignoring the long-instance item *iPhone X*, though *iPhone X* and *iPhone XS* are highly similar. This is because that: (1) Current GNNs can merely propagate information from adjacent items according to the adjacent matrices at each GNN layer; (2) Although multi-layer GNNs have subsequently been employed to propagate information between items

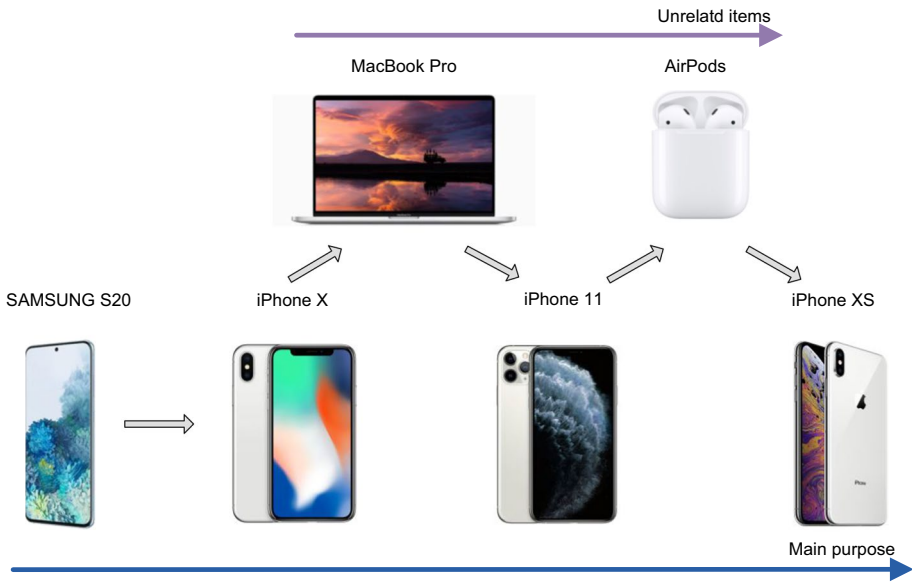


Fig. 1 An example session, where the gray arrows indicate the sequential order of interacted items, and the items *MacBook Pro* and *AirPods* are regarded as “unrelated items” since they are uncorrelated to the user’s main purpose

without direct connections, they easily lead to overfitting (Wu et al., 2019; Qiu et al., 2019). Besides, this serious overfitting problem of GNNs on session-based recommendation also restricts the number of deep layers of GNNs, limiting the information propagating for generating accurate item embeddings.

In summary, the main challenges in GNNs for session-based recommendation are:

1. *How to design graph neural networks that can simultaneously propagate the information from adjacent items and long-distance items without direct connections?*
2. *How to solve the overfitting problem in GNNs for session-based recommendation to enable deep layers of GNNs for generating an accurate item representation?*

To address the above issues, we propose Star Graph Neural Networks with Highway Networks (SGNN-HN) for session-based recommendation. We first use a star graph neural network (SGNN) to model the complex transition pattern in an ongoing session, which can solve the long-range information propagation problem by adding a star node to take the non-adjacent items into consideration on the basis of gated graph neural networks. Then, to circumvent the overfitting problem of GNNs, we apply a highway networks (HN) to dynamically select the item embeddings before and after multi-layer SGNNs, which can help to explore complex transition relationship between items. After that, we attentively aggregate item embeddings generated by the SGNN in an ongoing session so as to represent user’s preference for making item recommendation.

We conduct experiments on two publicly available benchmark datasets, i.e., *Yoochoose* and *Diginetica*. Our experimental results show that the proposed SGNN-HN model can outperform state-of-the-art baselines in terms of Recall and MRR. We also

conduct detailed analysis to evaluate the effectiveness of each component in SGNN-HN and investigate the scalability and sensitivity of our proposal.

In summary, our contributions in this paper are as follows:

1. To the best of our knowledge, we are the first to consider long-distance relations between items in a session for information propagation in graph neural networks for session-based recommendation.
2. We propose a star graph neural network (SGNN) to model the complex transition relationship between items in an ongoing session and apply a highway networks (HN) to deal with the overfitting problem existing in GNNs.
3. We compare the performance of SGNN-HN against the state-of-the-art baselines on two public benchmark datasets and the experimental results show the superiority of SGNN-HN over the state-of-the-art models in terms of Recall and MRR.

2 Related work

We review the related work about general recommendation models in Sect. 2.1, the attention based recommendation models in Sect. 2.2, the sequential recommendation models in Sect. 2.3, and the graph neural networks based models in Sect. 2.4.

2.1 General recommendation models

General recommenders produce recommendations by relying on the long-term historical user-item interactions, most methods are based on collaborative filtering (CF), which can be mainly divided into KNN based methods and latent factor based methods. KNN based methods generate recommendations based on the similarity between users or items, which is defined as *user-based* KNN methods and *item-based* KNN methods, respectively. For example, Jin et al. (2004) propose to automatically obtain the weights for different items based on the scores rated by the training users. Moreover, Sarwar et al. (2001) propose an item-based technique to first discover the relationships from the user-item matrix, where the relationships are then utilized for generating the recommendation list. Differently, latent factor based methods consider the interactions with items based on the user-item rating matrix, where common methods include Matrix Factorization (MF), Singular Value Decomposition (SVD) and SVD++. For example, Koren (2008) proposes SVD++ to simultaneously exploit the explicit and implicit feedback in user behaviors. In recent years, deep learning has been widely investigated in CF. For instance, He et al. (2017) propose Neural Collaborative Filtering (NCF), which utilizes a multi-layer perceptron to express and generalize matrix factorization in a non-linear way. Furthermore, Chen et al. (2019) propose a Joint Neural Collaborative Filtering (J-NCF) to further learn deep features of users and items, and design a hybrid loss function to combine both point-wise and pair-wise loss for optimizing by utilizing both implicit and explicit feedback.

However, both KNN and latent factor based methods generally neglect the sequential information, making it hard to capture user's instant interest. Moreover, user's latent factor cannot be generated in session-based recommendation since the long-term interactions are unavailable, making the latent factor based methods fail to work.

2.2 Attention based recommendation models

Considering items in user's interaction history have different importance, which means that they contribute differently to detecting user intent, attention mechanism is introduced for user preference modeling. Moreover, attention mechanism can be used alone (Liu et al., 2018; Pan et al., 2020; Kang & McAuley, 2018) or combined with other methods like RNNs as a component (Li et al., 2017; Wu et al., 2019). For instance, Liu et al. (2018) apply an attention mechanism to obtain a user's general preference and recent interest by relying on the long-term and short-term memories in the current session, respectively. Moreover, in order to better distinguish the importance of items and avoid potential bias brought by unrelated items, Pan et al. (2020) propose to measure item importance using an importance extraction module, and consider the global preference and recent interest to make item prediction. For scenarios where user's long-term historical interactions are available, attention mechanism is utilized for distinguishing the contribution of each item or session to detect user intent in the current session. For instance, Ying et al. (2018) propose a two-layer hierarchical attention network, where the first attention layer is applied to the historical interactions to obtain user's general preference, which is combined with the recent interacted items by the second attention layer to generate the final preference. Moreover, to capture the dynamic interactions between user's long- and short-term behaviors, Chen et al. (2019) design a dynamic co-attention networks for session-based recommendation (DCN-SR) to generate co-dependent representations of user's general and recent interests. Furthermore, different from DCN-SR that focuses on the distinction of historical items, Sun et al. (2019) propose to utilize a two-layer nonlocal architecture to identify relevant historical sessions to the current session for producing the long-term user preference.

However, attention-based methods merely focus on the relative importance of items or sessions, without considering the complex transition relationship between items in an ongoing session, making it hard to model the complex user behavior pattern for generating recommendations.

2.3 Sequential recommendation models

Considering that items are organized into the session according to their chronological order, many sequential recommendation models are proposed to model the sequential signal for item prediction. Markov chains have been widely investigated to model the sequential behaviors. For instance, Rendle et al. (2010) propose Factorizing Personalized Markov Chains (FPMC) to combine Markov chains with matrix factorization for modeling user's sequential interactions and capturing their long-term preference, respectively. Considering the utility of Recurrent neural networks (RNNs) like GRU for dealing with sequential data, RNNs have also been widely applied in session-based recommendation. For example, Hidasi et al. (2016) first propose GRU4REC, which utilizes GRU to model the sequential signal in session-based recommendation and adopts a session-parallel minibatch for model training. Then, Hidasi and Karatzoglou (2018) point that GRU4REC faces gradients vanishing problem in training and design novel loss functions for optimizing the model parameters. In addition, considering information in the current session is limited, many neighbor-enhanced methods are proposed based on the sequential recommenders to incorporate similar sessions as neighbors for helping model the ongoing session. For instance, Jannach and Ludewig (2017) introduce neighbor sessions with a K -nearest neighbor method

(KNN) and combine them with GRU4REC in a weighted hybrid approach. Different from traditional KNN, Wang et al. (2019) propose to incorporate neighbor sessions as auxiliary information via memory networks (Weston et al., 2015; Sukhbaatar et al., 2015).

However, Markov Chain based models can only capture information from adjacent transactions, making it hard to adequately capture interest migration. Moreover, sequential recommendation methods process the item sequences strictly according to the time order, which may introduce bias since user's interaction patterns tend to be more complex than the simple sequential signal as pointed in Qiu et al. (2019).

2.4 Graph Neural Networks based models

Recently, because of their ability to model complex relationships between objects, graph neural networks (GNNs) (Kipf & Welling, 2017; Velickovic et al., 2018; Li et al., 2016) have attracted much interest in the context of session-based recommendation. For instance, Wu et al. (2019) propose SR-GNN to apply gated graph neural networks (GGNN) for modeling the transition relationship among items in the ongoing session so as to generate accurate item embeddings and aggregate the items as the session representation using an attention mechanism. Xu et al. (2019) propose a graph contextual self-attention model based on graph neural network (GC-SAN), that extends SR-GNN with adopting self-attention mechanism to obtain contextualized non-local representations for accurate recommendation. Furthermore, Qiu et al. (2019) propose a Full Graph Neural Networks (FGNN), which utilizes a weighted graph attention network (WGAT) based approach for generating item representation, which is then aggregated by a Readout function (Vinyals et al., 2016) as user preference for item prediction. Unlike the above methods, the global session graph to propagate information from similar sessions for generating item embeddings is also constructed. For instance, Qiu et al. (2020) improve FGNN by constructing a Broadly Connected Session (BCS) graph to connect different sessions and design a Mask-Readout function to generate the session representation. Moreover, Wang et al. (2020) propose Global Context Enhanced Graph Neural Networks (GCE-GNN) to exploit item transitions from session-level and global-level, which learns item embeddings from the session graph transformed from the current session and the global graph constructed from all sessions, respectively.

Although graph neural networks have achieved considerable performance in session-based recommendation, there remains problems in the foregoing GNN based methods: (1) only the information from adjacent items can be propagated at each GNN layer, neglecting the long-distance items; (2) the overfitting problem is serious, limiting the layer number of deep GNNs.

3 Approach

In this section, we present our proposed SGNN-HN in detail. We first give the definition of the session-based recommendation task. Then, in Fig. 2, we plot the workflow of the SGNN-HN model. Specifically, we first transform the item sequence into a session star graph for each session by introducing the star node in Sect. 3.1. Then in Sect. 3.2, we feed in the nodes in the star graph into multi-layer SGNNs to learn their embeddings with utilizing the highway networks to combine the node vectors before and after SGNNs, and then restore the item sequence from the star graph. Finally, in Sect. 3.3, we integrate user's

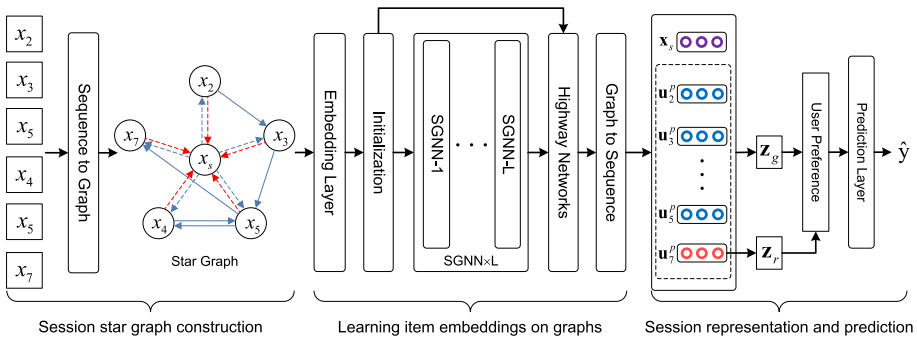


Fig. 2 The workflow of SGNN-HN

general and recent preferences to generate the session representation, which is utilized to make item recommendations by predicting the scores on all candidates in the item set.

Given the current session, session-based recommendation aims to predict the item for user to click at the next timestamp. This task can be formulated as follows. Assuming the item set is denoted as $V = \{v_1, v_2, \dots, v_{|V|}\}$, which is constituted of all unique items in the dataset and $|V|$ indicates the length of V . For each user, we are given user’s current session as $S = \{v_1, v_2, \dots, v_t, \dots, v_n\}$ which contains n chronological items, where $v_t \in V$ is the item that user clicked at the t -th timestamp, the goal is to predict the next item v_{n+1} to click. More specifically, we calculate the prediction score of each item in V as $\hat{y} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{|V|}\}$, where $\hat{y}_i \in \hat{y}$ denotes the probability of clicking item i . Finally, the items ranking at the top- K positions in prediction constitute the recommendation list for the user.

We list the main notations used in this paper in Table 1.

3.1 Session star graph construction

Given the current session $S = \{v_1, v_2, \dots, v_t, \dots, v_n\}$, a star graph is constructed for representing the complex transition relationship among the items in S . In the star graph, besides propagating information from the adjacent items, the non-adjacent items are also taken into consideration via the introduced additional star node, which leads to a more complete connection of all items in the ongoing session than the existing graph neural networks such as GCN (Kipf & Welling, 2017) and GAT (Velickovic et al., 2018). Specifically, each session S is transformed into a star graph denoted as $G_s = \{V_s, E_s\}$, where $V_s = \{x_1, x_2, \dots, x_m, x_s\}$ indicates all nodes in the star graph. Among the nodes, $\{x_1, x_2, \dots, x_m\}$ denotes all unique items in the current session, which we regard as the *satellite nodes*, and x_s is the additionally introduced *star node*. Note that $m \leq n$ since in the session items may be clicked repeatedly (Ren et al., 2019; Wu et al., 2019). Moreover, E_s indicates the edge set in G_s that can be divided into two types, i.e., satellite connections and star connections, which are utilized to propagate information from the corresponding satellite nodes and star node for updating the item embeddings, respectively.

Satellite connections The satellite connections are utilized to represent the transition relationship between adjacent items in each session. Here, we adopt the gated graph neural networks (GGNN) (Li et al., 2016) as an example in our star graph neural network (SGNN). Actually, other GNNs such as GCN (Kipf & Welling, 2017) and

Table 1 Main notations used in this paper

Notation	Description
U	A set of sessions in the dataset
V	The item set containing all candidate items
$S = \{v_1, v_2, \dots, v_n\}$	A session consisting of n sequential items
K	The number of items recommended to the user
$G_s = \{\mathcal{V}_s, \mathcal{E}_s\}$	The constructed graph for session S with its nodes and edges
$\{x_1, x_2, \dots, x_m\}$	The satellite nodes in \mathcal{V}_s
x_s	The incorporated star node in \mathcal{V}_s
$\mathbf{h}^l, \mathbf{x}_s^l$	The item embeddings of the satellite nodes and star node at layer l
$\mathbf{A}^l, \mathbf{A}^o$	The incoming and outgoing matrices in the GGNN
$\mathbf{W}^l, \mathbf{W}^o$	The learnable parameters for the incoming and outgoing edges
$\mathbf{a}_i^l, \hat{\mathbf{h}}_i^l$	The input and output of node i in GRU of GGNN at layer l
$\mathbf{W}_{q1}, \mathbf{W}_{k1}, \mathbf{W}_{q2}, \mathbf{W}_{k2}$	The learnable parameters of the self-attention mechanism in SGNN
α_i^l	The similarity of satellite node i to the star node
β	The importance of satellite nodes for updating the star node
\mathbf{W}_g	The trainable parameters in highway networks
\mathbf{g}	The gating score of each candidate items in SGNN
\mathbf{h}^f	The hybrid item representation outputted by the highway networks
\mathbf{p}	The learnable position embeddings
$\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$	The trainable parameters in the attention mechanism for aggregation
γ_i	The weight of item i for session aggregating
$\mathbf{z}_l, \mathbf{z}_r$	The global and recent preferences in the session
\mathbf{W}_4	The trainable parameters for combining global and recent interests
$\mathbf{z}_h, \bar{\mathbf{z}}_h$	The representations of user preference before and after normalization
$\mathbf{v}_i, \bar{\mathbf{v}}_i$	The representations of candidate item x_i before and after normalization
τ	A scale coefficient applied inside the softmax layer
$\tilde{\mathbf{y}}_i, \hat{\mathbf{y}}$	The prediction scores before and after scaled softmax

GAT (Velickovic et al., 2018) can also be adopted in SGNN to replace GGNN in our approach. As shown in Fig. 2, the satellite connections in the star graphs are indicated by the blue solid lines, each edge $(x_i, x_j) \in \mathcal{E}_s$ means that the user clicks item x_j after clicking x_i . Specifically, the satellite connections in each session can be represented by the connection matrices, including an incoming matrix and an outgoing matrix. For example, considering a session $S = \{x_2, x_3, x_5, x_4, x_5, x_7\}$, the incoming and outgoing matrices of GGNN in SGNN can be constructed as in Fig. 3.

Star connections Borrowing the merits from Guo et al. (2019), we propagate long-distance information from items without direct connections through the introduced star node. Specifically, as shown in Fig. 2, we add a bidirectional edge between the star node x_s and each satellite node x_i in the session star graph. On the one hand, the directed edges from the star node to the satellite nodes, i.e., the blue dotted lines $(x_s, x_i) \in \mathcal{E}_s$ are utilized to update the representation of the satellite nodes. Through the star node, long-distance information from unconnected items can be propagated across two hops by regarding the star node as an intermediate node. On the other hand, we use the directed

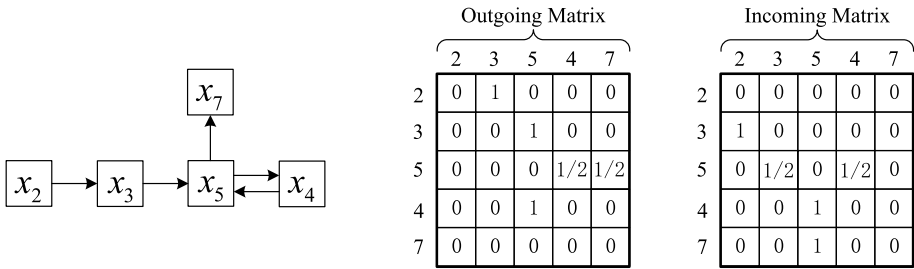


Fig. 3 An example of adjacent matrices in GGNN

edges from the satellite nodes to the star node, i.e., the red dotted lines $(x_i, x_s) \in \mathcal{E}_s$ to update the star node representation by combining all the satellite nodes.

3.2 Learning item embeddings on graphs

Next, we illustrate how SGNN propagates information in the session star graph, including mainly three stages, i.e., the node initialization, the node update which consists of *satellite nodes update* and *star node update*, and the usage of the highway networks in multi-layer SGNNs.

3.2.1 Initialization

Before propagating information in the star graph, we initialize the representation of the satellite nodes and the star node. Specifically, given a star graph with a node set $\mathcal{V}_s = \{x_1, x_2, \dots, x_m, x_s\}$, we first generate an embedding vector $\mathbf{x}_i \in \mathbb{R}^d$ for each satellite node x_i in $\{x_1, x_2, \dots, x_m\}$ through an embedding layer, where d is the dimension of \mathbf{x}_i . Thus the satellite nodes can be represented as:

$$\mathbf{h}^0 = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}, \tag{1}$$

As for the star node x_s , we initialize a d -dimensional representation $\mathbf{x}_s^0 \in \mathbb{R}^d$ by utilizing an average pooling on the embeddings of the satellite nodes as:

$$\mathbf{x}_s^0 = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i, \tag{2}$$

3.2.2 Update

After obtaining the initialization of the nodes in the graph, we update the representation of the satellite nodes and the star node in each SGNN layer as follows:

Satellite nodes update When updating the representation of each satellite node, the neighbors comes from two sources, i.e., the adjacent nodes and the star node, which helps propagate information from connected and unconnected items in the session, respectively. Compared to existing GNNs applied in session-based recommendation such as GGNN (Wu et al., 2019)

and GAT (Qiu et al., 2019), the introduced star node in our SGNN can make long-distance information from items without connections available in propagating in a two-hop way.

First, we consider the information from adjacent items. For each satellite node x_i in the star graph at layer l , we obtain the propagated information through the incoming matrix and the outgoing matrix in GGNN, which can be formulated as:

$$\mathbf{a}_i^l = \text{Concat}(\mathbf{A}_i^I([\mathbf{x}_1^{l-1}, \mathbf{x}_2^{l-1}, \dots, \mathbf{x}_m^{l-1}]^\top \mathbf{W}^I + \mathbf{b}^I), \mathbf{A}_i^O([\mathbf{x}_1^{l-1}, \mathbf{x}_2^{l-1}, \dots, \mathbf{x}_m^{l-1}]^\top \mathbf{W}^O + \mathbf{b}^O)), \tag{3}$$

where $[\mathbf{x}_1^{l-1}, \mathbf{x}_2^{l-1}, \dots, \mathbf{x}_m^{l-1}]$ are the embedding vectors of the satellite nodes in layer $l - 1$, $\mathbf{A}_i^I, \mathbf{A}_i^O \in \mathbb{R}^{1 \times m}$ are the i -th row of the incoming and the outgoing matrices, which determine the incoming and outgoing importance weights of different neighbors of node x_i , respectively. $\mathbf{W}^I, \mathbf{W}^O \in \mathbb{R}^{d \times d}$ and $\mathbf{b}^I, \mathbf{b}^O \in \mathbb{R}^d$ denote the trainable parameters. Thus, we can obtain $\mathbf{a}_i^l \in \mathbb{R}^{1 \times 2d}$ to represent the information propagated to satellite node x_i from the adjacent items at layer $l - 1$. After that, we feed \mathbf{a}_i^l and the previous state \mathbf{h}_i^{l-1} of node x_i into a GRU of GGNN as in Wu et al. (2019); Xu et al. (2019), to determine how much information in \mathbf{a}_i^l and \mathbf{h}_i^{l-1} is preserved, respectively:

$$\hat{\mathbf{h}}_i^l = \text{GRU}(\mathbf{a}_i^l, \mathbf{h}_i^{l-1}), \tag{4}$$

where $\hat{\mathbf{h}}_i^l$ denotes the node vector of x_i with propagated information from the adjacent items.

Next, we consider the long-distance information from unconnected items through the star node. In SGNN, the star node can provide an overall representation of all the satellite nodes in the graph. For each satellite node x_i , we utilize a gating layer to selectively combine the information from the adjacent nodes $\hat{\mathbf{h}}_i^l$ and the star node \mathbf{x}_s^{l-1} as the final representation of x_i :

$$\mathbf{h}_i^l = (1 - \alpha_i^l)\hat{\mathbf{h}}_i^l + \alpha_i^l\mathbf{x}_s^{l-1}, \tag{5}$$

where \mathbf{x}_s^{l-1} is the representation of the star node in layer $l - 1$, and the weight α_i^l for each satellite node is calculated using the self-attention mechanism as follows:

$$\alpha_i^l = \frac{(\mathbf{W}_{q1}\hat{\mathbf{h}}_i^l)^\top \mathbf{W}_{k1}\mathbf{x}_s^{l-1}}{\sqrt{d}}, \tag{6}$$

where $\mathbf{W}_{q1}, \mathbf{W}_{k1} \in \mathbb{R}^{d \times d}$ denote the learnable parameters, and \sqrt{d} is used to scale the generated attention scores.

Star node update In each SGNN layer, we generate the new star node vector based on the updated representation of the satellite nodes. Specifically, we obtain the feature vector of the star node x_s by distinguishingly combing the satellite nodes \mathbf{h}^l as follows:

$$\mathbf{x}_s^l = \beta \mathbf{h}^l, \tag{7}$$

where $\beta \in \mathbb{R}^m$ denote the weights of different satellite nodes, which are calculated based on the representation of x_s in layer $l - 1$ (i.e., \mathbf{x}_s^{l-1}) using a self-attention mechanism:

$$\begin{cases} \beta = \text{softmax}(\frac{\mathbf{K}^\top \mathbf{q}}{\sqrt{d}}), \\ \mathbf{K} = \mathbf{W}_{k2}\mathbf{h}^l, \mathbf{q} = \mathbf{W}_{q2}\mathbf{x}_s^{l-1}, \end{cases} \tag{8}$$

where $\mathbf{q} \in \mathbb{R}^d$ and $\mathbf{K} \in \mathbb{R}^{d \times m}$ are the *query* and *key* for attention calculation, and $\mathbf{W}_{q_2}, \mathbf{W}_{k_2} \in \mathbb{R}^{d \times d}$ are the learnable parameters.

In this way, we can generate an accurate representation of the star node, so as to better transmit the long-distance information between every two non-adjacent items in the session.

Algorithm 1 Procedure of Multi-layer SGNNs

Input: $S = \{v_1, v_2, \dots, v_n\}$: The sequential items in an ongoing session;

Output: $\mathbf{u} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n\}, \mathbf{x}_s$: Representation of sequential items in the session and star node;

- 1: $G_s = \{\mathcal{V}_s, \mathcal{E}_s\}, \mathbf{A}^I, \mathbf{A}^O \leftarrow SeqtoGraph(S)$;
 - 2: $\mathbf{h}^0 = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ based on (1);
 - 3: $\mathbf{x}_s^0 = average(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)$ based on (2);
 - 4: **for** l in range(L) **do**
 - 5: $\hat{\mathbf{h}}^l = GGNN(\mathbf{h}^{l-1}, \mathbf{A}^I, \mathbf{A}^O)$ based on (3) and (4);
 - 6: $\alpha^l = SelfAtt(\hat{\mathbf{h}}^l, \mathbf{x}_s^{l-1})$ based on (6);
 - 7: $\mathbf{h}^l = Gating(\hat{\mathbf{h}}^l, \mathbf{x}_s^{l-1}, \alpha^l)$ based on (5);
 - 8: $\mathbf{x}_s^l = SelfAtt(\mathbf{x}_s^{l-1}, \mathbf{h}^l)$ based on (7) and (8);
 - 9: **end for**
 - 10: $\mathbf{h}^f = HighwayNetworks(\mathbf{h}^0, \mathbf{h}^L)$ based on (10) and (11);
 - 11: $\mathbf{u} = GraphtoSeq(\mathbf{h}^f)$;
 - 12: **return** \mathbf{u} and \mathbf{x}_s .
-

3.2.3 Highway networks

Through iteratively updating the satellite nodes and the star node, we can obtain accurate feature vectors of the nodes. Furthermore, in order to exploit the high-order connectivity between items, we can stack multi-layer SGNNs, where the l -layer of SGNN can be formulated as:

$$\mathbf{h}^l, \mathbf{x}_s^l = SGNN(\mathbf{h}^{l-1}, \mathbf{x}_s^{l-1}, \mathbf{A}^I, \mathbf{A}^O). \tag{9}$$

However, multi-layer GNNs may introduce noise to the updating of the nodes, which leads to a serious overfitting problem in session-based recommendation as stated in Wu et al. (2019); Qiu et al. (2019). To alleviate overfitting, we utilize the highway networks (Srivastava et al., 2015) to selectively combine the representation of the satellite nodes before and after multi-layer SGNNs. The embeddings of the satellited nodes before SGNNs are initialized in Eq. 1 as \mathbf{h}^0 . As L -layer SGNNs are stacked, the feature vectors of the satellite nodes after multi SGNN layers are denoted as \mathbf{h}^L . Then, we can formulate the highway networks as:

$$\mathbf{h}^f = \mathbf{g} \odot \mathbf{h}^0 + (1 - \mathbf{g}) \odot \mathbf{h}^L, \tag{10}$$

where the gating weight $\mathbf{g} \in \mathbb{R}^{d \times m}$ is determined by both \mathbf{h}^0 and \mathbf{h}^L :

$$\mathbf{g} = \sigma(\mathbf{W}_g[\mathbf{h}^0; \mathbf{h}^L]), \tag{11}$$

where $[\cdot]$ stands for the concatenation, $\mathbf{W}_g \in \mathbb{R}^{d \times 2d}$ is the trainable matrix that transforms the concatenated vector from \mathbb{R}^{2d} to \mathbb{R}^d , and σ denotes the *sigmoid* function.

Through the highway networks, the satellite nodes can be finally denoted as \mathbf{h}^f , and the star node is \mathbf{x}_s^L (here we denote as \mathbf{x}_s for brevity). After generating the node representations based on multi-layer SGNNs, we restore the item sequence from the corresponding satellite nodes in the star graph, which is a reverse operation of the session star graph construction in Sect. 3.1. Then we can get the representation of the item sequence as $\mathbf{u} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n\}$, where $\mathbf{u}_i \in \mathbb{R}^d$ is the feature vector of the item that user clicks at the i -th timestamp in the session, and n denotes the session length.

In Algorithm 1, we detail the procedure of SGNN. First, we construct the star graph by adding a star node and define the adjacent matrices in line 1. Then we initialize the representation of the satellite nodes and star node in line 2 and 3. Next, we update the representations through the L -layer SGNNs by lines from 4 to 9. In detail, at each SGNN layer, we first propagate information from adjacent items in line 5, then selectively combine it with the star node in line 6 and 7 to update the satellite nodes, and finally generate the new star node in line 8. After that, we utilize the highway networks to combine the representation of the satellite nodes before and after multi-layer SGNNs in line 10. Finally, we recover the item sequence from the session star graph in line 11 and return the representations of the sequential items and the star node.

3.3 Session representation and prediction

After obtaining the accurate embeddings of the chronological items in the ongoing session, we aggregate the items to generate the session representation as the user preference for item recommendation. In order to incorporate the sequential information into SGNN-HN, we add learnable position embeddings $\mathbf{p} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n]$ to the item representations $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n]$ as:

$$\mathbf{u}^p = \mathbf{u} + \mathbf{p}, \tag{12}$$

where $\mathbf{p}_i \in \mathbb{R}^d$ is the embedding of the absolute position i corresponding to item x_i in the session.

Then we integrate both user’s global preference and recent interest within the session as the session representation. Since previous work has proved that the last item in a session can reflect user’s recent interest (Liu et al., 2018; Wu et al., 2019), we directly regard the last item embeddings as user’s recent interest:

$$\mathbf{z}_r = \mathbf{u}_n^p, \tag{13}$$

As for the global preference, considering that items in a session have different degrees of priority, we obtain the importance of each item using an attention mechanism. Specifically, for item x_i the importance is determined by both the star node vector \mathbf{x}_s and the recent interest \mathbf{z}_r , as follows:

$$\gamma_i = \mathbf{W}_0^\top \sigma(\mathbf{W}_1 \mathbf{u}_i^p + \mathbf{W}_2 \mathbf{x}_s + \mathbf{W}_3 \mathbf{z}_r + \mathbf{b}), \tag{14}$$

where $\mathbf{W}_0 \in \mathbb{R}^d$, $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are learnable parameters in the soft attention. Then we combine the embeddings of items according to their importance scores to obtain the global preference $\mathbf{z}_g \in \mathbb{R}^d$ as follows:

$$\mathbf{z}_g = \sum_{i=1}^n \gamma_i \mathbf{u}_i^p, \tag{15}$$

After generating user’s global preference \mathbf{z}_g and current interest \mathbf{z}_r , we integrate them together as the final session representation as follows:

$$\mathbf{z}_h = \mathbf{W}_4[\mathbf{z}_g; \mathbf{z}_r], \tag{16}$$

where $[\cdot]$ denotes concatenation and $\mathbf{W}_4 \in \mathbb{R}^{d \times 2d}$ is the trainable parameter.

Then we make item recommendations by utilizing the session representation to calculate the prediction scores on all items in the candidate item set V , which indicates the probabilities of clicking each item in the next timestamp. To address the long-tail problem in recommendations as pointed in Gupta et al. (2019); Abdollahpouri et al. (2017), we apply the layer normalization on the representation of the current session \mathbf{z}_h and each candidate item \mathbf{v}_i , respectively:

$$\begin{aligned} \tilde{\mathbf{z}}_h &= \text{LayerNorm}(\mathbf{z}_h), \\ \tilde{\mathbf{v}}_i &= \text{LayerNorm}(\mathbf{v}_i), \end{aligned} \tag{17}$$

Then for a given candidate item x_i in the item set V , we calculate its prediction score by multiplying its item embeddings with the session representation as follows:

$$\tilde{\mathbf{y}}_i = \tilde{\mathbf{z}}_h^T \tilde{\mathbf{v}}_i, \tag{18}$$

Finally, we adopt a softmax layer to normalize the scores. It is worth noting that after normalization there is a convergence problem as the softmax loss will be trapped at a very high value on the training set (Wang et al., 2017; Gupta et al., 2019). To solve this problem, we apply a scale coefficient τ in the softmax to generate the final prediction scores:

$$\hat{\mathbf{y}} = \frac{\exp(\tau \tilde{\mathbf{y}}_i)}{\sum_i \exp(\tau \tilde{\mathbf{y}}_i)}, \quad \forall i = 1, 2, \dots, |V|, \tag{19}$$

where $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \dots, \hat{\mathbf{y}}_{|V|})$. Finally, the items with the highest scores in $\hat{\mathbf{y}}$ will constitute the recommendation list for the user.

To train SGNN-HN, we adopt the cross-entropy loss as the objective function:

$$L(\hat{\mathbf{y}}) = - \sum_{i=1}^{|V|} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i), \tag{20}$$

where \mathbf{y} denotes the one-hot encoding vector of the ground truth. Finally, the Back-Propagation Through Time (BPTT) algorithm (Rumelhart et al., 1986) is utilized to train our proposed SGNN-HN.

To make the proposal more clearly, we show the training process of our SGNN-HN model in Algorithm 2. We first input each session into SGNNs and obtain the representations of the items as well as the star node using the multi-layer SGNNs in step 3. Then, we add the position embeddings in step 4. Next, we generate the session representation in step 5 to 8. Specifically, we get the recent interest in step 5, obtain the global preference in step

7 by using the importance scores generated in step 6, and conduct a preference fusion in step 8 to get the final user preference. After that, we generate the prediction scores in step 9. Finally, we adopt the cross-entropy as the loss function and utilize the back propagation to optimize the parameters of our SGNN-HN model in step 11 and 12.

Algorithm 2 Training process of SGNN-HN.

Input: U : all training sessions;
 Epochs: the number of training interactions;
Output: V : embeddings of all items;
 p : position embeddings;
 W^I, W^O : parameters for the incoming and outgoing edges;
 $W_{q1}, W_{k1}, W_{q2}, W_{k2}$: parameters of the self-attention mechanism in SGNN;
 W_g : parameters in highway networks;
 W_0, W_1, W_2, W_3 : parameters in attention mechanism for aggregation;
 W_4 : parameter for preference fusion;

- 1: **for** epoch in range(Epochs) **do**
- 2: **for** session S in U **do**
- 3: $u, x_s \leftarrow$ use Eq. (1)-Eq. (11) with S as input;
- 4: $u^p \leftarrow$ use Eq. (12) with u, p as inputs;
- 5: $z_r \leftarrow$ use Eq. (13) with u^p as inputs;
- 6: $\gamma \leftarrow$ use Eq. (14) with u^p, x_s, z_r as inputs;
- 7: $z_g \leftarrow$ use Eq. (15) with u^p, γ as inputs;
- 8: $z_h \leftarrow$ use Eq. (16) with z_g, z_r as inputs;
- 9: $\hat{y}_i \leftarrow$ use Eq. (17), (18) and (19) with z_h, v_i as inputs;
- 10: **end for**
- 11: $L(\hat{y}) \leftarrow$ use Eq. (20) with y, \hat{y} as inputs;
- 12: use back propagation to optimize the parameters;
- 13: **end for**
- 14: **return** The trainable parameters of SGNN-HN, item embeddings and position embeddings.

4 Experiments

4.1 Research questions

To examine the performance of SGNN-HN, we address seven research questions:

(RQ1) Can the proposed SGNN-HN model beat the competitive baselines for the session-based recommendation task?

(RQ2) What is the contribution of the star graph neural networks (SGNN) to the overall recommendation performance?

- (RQ3) Do the highway networks alleviate the overfitting problem of graph neural networks in session-based recommendation?
- (RQ4) How does the layer number of SGNNs affect the recommendation performance?
- (RQ5) How does SGNN-HN perform on sessions with different lengths compared to the baselines?
- (RQ6) Can SGNN-HN beat the baselines with different amount of training data for optimizing the parameters?
- (RQ7) What is the impact of the hyper parameters in SGNN-HN on the recommendation performance?
- (RQ8) How is the time complexity and computational cost of SGNN-HN compared with the baselines?

4.2 Datasets

We evaluate the performance of SGNN-HN and the baselines on two publicly available benchmark datasets, i.e., *Yoochoose* and *Diginetica*.

- *Yoochoose*¹ is a public dataset released by the RecSys Challenge 2015, which contains click streams from an e-commerce website within a 6 month period.
- *Diginetica*² is obtained from the CIKM Cup 2016. Here we only adopt the transaction data.

For *Yoochoose*, following (Li et al., 2017; Liu et al., 2018; Wu et al., 2019), we filter out sessions of length 1 and items that appear less than 5 times. Then we split the sessions for training and test, respectively, where the sessions of the last day is used for test and the remaining part is regarded as the training set. Furthermore, we remove items that are not included in the training set. As for *Diginetica*, the only difference is that we utilize the sessions of the last week for test. After preprocessing, 7,981,580 sessions with 37,483 items are remained in the *Yoochoose* dataset, and 204,771 sessions with 43,097 items constitute the *Diginetica* dataset.

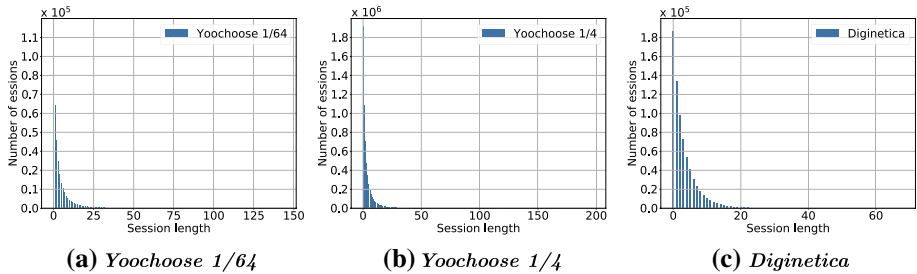
Similar to Li et al. (2017); Liu et al. (2018); Wu et al. (2019), we use a sequence splitting preprocess to augment the training samples. Specifically, for session $S = \{v_1, v_2, \dots, v_n\}$, we generate the sequences and corresponding labels as $([v_1], v_2), ([v_1, v_2], v_3), \dots, ([v_1, \dots, v_{n-1}], v_n)$ for training and test. Moreover, as *Yoochoose* is too large, following Li et al. (2017); Liu et al. (2018); Wu et al. (2019), we only utilize the recent 1/64 and 1/4 fractions of the training sequences, denoted as *Yoochoose 1/64* and *Yoochoose 1/4*, respectively. The statistics of the three datasets, i.e., *Yoochoose 1/64*, *Yoochoose 1/4* and *Diginetica* are provided in Table 2. We also plot the distribution of sessions with different number of items in three datasets in Fig. 4, where we can observe that most sessions are concentrated on the short lengths, especially in the *Yoochoose* datasets.

¹ <http://2015.recsyschallenge.com/challenge.html>

² <http://cikm2016.cs.iupui.edu/cikm-cup>

Table 2 Statistics of the datasets used in our experiments

Statistics	<i>Yoochoose 1/64</i>	<i>Yoochoose 1/4</i>	<i>Diginetica</i>
# clicks	557,248	8,326,407	982,961
# training sessions	369,859	5,917,746	719,470
# test sessions	55,898	55,898	60,858
# items	16,766	29,618	43,097
Average session length	6.16	5.71	5.12

**Fig. 4** Distribution of sessions with various lengths in the training set of three datasets

4.3 Evaluation metrics and baselines

Following previous works (Li et al., 2017; Wu et al., 2019), we adopt Recall@K and MRR@K to evaluate the recommendation performance, where the Recall@K score measures whether the target item is included in the top-K items in the recommendation list, and MRR@K is a normalized hit which takes the position of the target item into consideration.

The baselines adopted for comparison in this paper are the following: 1. Two traditional methods, i.e., S-POP (Adomavicius & Tuzhilin, 2005) and FPMC (Rendle et al., 2010); 2. Three RNN-based methods, i.e., GRU4REC (Hidasi et al., 2016), NARM (Li et al., 2017) and CSRMM (Wang et al., 2019); 3. Two purely attention-based methods, i.e., STAMP (Liu et al., 2018) and SR-IEM (Pan et al., 2020); and 4. Two GNN-based methods, i.e., SR-GNN (Wu et al., 2019) and NISER+ (Gupta et al., 2019). We list all the models to be discussed in Table 3.

4.4 Experimental setup

We implement SGNN-HN with six layers of SGNNs to obtain the item embeddings. The hyper parameters are selected on the validation set which is randomly selected from the training set with a proportion of 10%. Following (Wu et al., 2019; Gupta et al., 2019; Pan et al., 2020), the batch size is set to 100 and the dimension of item embeddings is 256. We adopt the Adam optimizer with an initial learning rate $1e^{-3}$ and a decay factor 0.1 for every 3 epochs. Moreover, L2 regularization is set to $1e^{-5}$ to avoid overfitting, and the scale coefficient τ is set to 12 on three datasets. We set the maximum session length to 10 as in Kang and McAuley (2018); Pan et al. (2020), which means that

Table 3 An overview of models discussed in the paper

Model	Description	Source
S-POP	A traditional method ranking the items according to their popularity.	(Adomavicius & Tuzhilin, 2005).
FPMC	A Markov Chains based hybrid method for sequential recommendation.	(Rendle et al., 2010).
GRU4REC	An RNN-based method that applies GRUs to model the sequential information.	(Hidasi et al., 2016).
NARM	An RNN-based method which augments GRUs with an attention mechanism for capturing user's main purpose.	(Li et al., 2017).
CSRM	A neighbor-enhanced model that extends NARM by introducing neighbor sessions as auxiliary information.	(Wang et al., 2019).
STAMP	A purely attention-based method which utilizes an attention mechanism to assign different importance scores to the items.	(Liu et al., 2018).
SR-IEM	A purely attention-based method which obtains the item importance based on a modified self-attention mechanism.	(Pan et al., 2020).
SR-GNN	A GNN-based model that utilizes the gated graph neural networks to generate accurate item representations.	(Wu et al., 2019).
PEN4Rec	A GNN-based model which solves the user preference drifting problem by designing a second-stage retrieval process.	(Hu et al., 2021)
NISER+	A GNN-based method which improves SR-GNN by solving the long-tail and overfitting problems.	(Gupta et al., 2019).
GGNN-HN	A variant of SGNN-HN which replaces SGNN with single GGNN for information propagation.	This paper
SGNN-SR	A variant of SGNN-HN which removes the highway networks and regards the output of the last SGNN layer as item representation.	This paper
SGNN-HN	A SGNN-HN model which utilizes SGNN for propagating long-distance information and adopts highway networks to prevent overfitting.	This paper

Table 4 Model performance in terms of Recall@20 and MRR@20

Method	<i>Yoochoose 1/64</i>		<i>Yoochoose 1/4</i>		<i>Diginetica</i>	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
S-POP	30.44	18.35	27.08	17.75	21.06	13.68
FPMC	45.62	15.01	–	–	31.55	8.92
GRU4REC	60.64	22.89	59.53	22.60	29.45	8.33
NARM	68.32	28.63	69.73	29.23	49.70	16.17
CSRM	69.85	29.71	70.63	29.48	51.69	16.92
STAMP	68.74	29.67	70.44	30.00	45.64	14.32
SR-IEM	71.15	<u>31.71</u>	71.67	31.82	52.35	17.64
SR-GNN	70.57	30.94	71.36	<u>31.89</u>	50.73	17.59
PEN4Rec	<u>71.53</u>	<u>31.71</u>	–	–	52.50	18.56
NISER+	71.27	31.61	<u>71.80</u>	31.80	<u>53.39</u>	<u>18.72</u>
SGNN-HN	72.06[▲]	32.61[▲]	72.85[▲]	32.55[▲]	55.67[▲]	19.45[▲]

The results of the best performing baseline and the best performer in each column are underlined and bold-faced, respectively. [▲] denotes a significant improvement of SGNN-HN over the best baseline using a paired *t*-test ($p < 0.01$)

for too long sessions we only adopt the most recent 10 items for training. All parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1.

5 Results and discussion

5.1 Overall performance

5.1.1 General comparison

We present the results of the proposed session-based recommendation model SGNN-HN as well as the baselines in terms of Recall@20 and MRR@20 in Table 4. For the baselines we can observe that the neural methods generally outperform the traditional baselines, i.e., S-POP and FPMC. The neural methods can be split into three categories:

RNN-based neural baselines As for the RNN-based methods, we can see that NARM generally performs better than GRU4REC, which validates the effectiveness of emphasizing the user’s main purpose. Moreover, comparing CSRM to NARM, by incorporating neighbor sessions as auxiliary information for representing the current session, CSRM outperforms NARM for all cases on three datasets, which means that neighbor sessions with similar intent to the current session can help boost the recommendation performance.

Attention-based neural baselines As for the purely attention-based methods, STAMP and SR-IEM, we see that SR-IEM generally achieves a better performance than STAMP, where STAMP employs a combination of the mixture of all items and the last item as “query” in the attention mechanism while SR-IEM extracts the importance of each item individually by comparing each item to other items. Thus, SR-IEM can avoid the bias introduced by the unrelated items and make accurate recommendations.

GNN-based neural baselines Considering the GNN-based methods, i.e., SR-GNN and NISER+, we can observe that the best performer NISER+ can generally outperform the RNN-based and attention-based methods for most cases on the three datasets, which verifies the effectiveness of graph neural networks on modeling the transition relationship of items in the session. In addition, NISER+ can outperform SR-GNN for most cases on the three datasets except for *Yoochoose 1/4*, where NISER+ loses against SR-GNN in terms of MRR@20. This may be due to the fact that the long-tail problem and the overfitting problem become more prevalent in scenarios with relatively few training data. In addition, by designing a second-stage retrieval process to strengthen the effect of relevant sessions for modeling the user preference evolving trajectory, PEN4Rec achieves the best performance in terms of both metrics on the *Yoochoose 1/64* dataset.

In later experiments, we take the best performer in each category of the neural methods for comparison, i.e., CSRМ, SR-IEM and NISER+.

Next, we move to the proposed model SGNN-HN. From Table 4, we observe that SGNN-HN can achieve the best performance in terms of Recall@20 and MRR@20 for all cases on the three datasets. The improvement of the SGNN-HN model against the baselines mainly comes from two aspects. One is the proposed star graph neural network (SGNN). By introducing a star node as the transition node of every two items in the session, SGNN can help propagate information not only from adjacent items but also long-distance items without direct connections. Thus, each node can obtain abundant information from their neighbor nodes. The other one is that by using the highway networks to solve the overfitting problem, our SGNN-HN model can stack more layers of SGNNs, leading to a better representation of items.

Moreover, we find that the improvement of SGNN-HN over the best baseline in terms of Recall@20 and MRR@20 on *Yoochoose 1/64* are 1.11% and 2.84%, respectively, and 1.46% and 2.07% on *Yoochoose 1/4*. The relative improvement in terms of MRR@20 is more obvious than that of Recall@20 on both *Yoochoose 1/64* and *Yoochoose 1/4*. In contrast, a higher improvement in terms of Recall@20 than of MRR@20 is observed on *Diginetica*, returning 4.27% and 3.90%, respectively. This may be due to the fact that the number of candidate items are different on *Yoochoose* and *Diginetica*, where the number of candidate items in *Yoochoose 1/64* and *Yoochoose 1/4* are obviously less than that of the *Diginetica* dataset.

Our results indicate that our proposed SGNN-HN model can help put the target item at an earlier position when there are relatively few candidate items and is more effective on hitting the target item in the recommendation list for cases with relatively many candidate items.

5.1.2 Recommendation hits in top positions

To examine the recommendation ability of our proposal for a limited length of recommendation list, we present the result of SGNN-HN as well as the baselines in terms of Recall@K and MRR@K with the recommendation number $K = 5$ and 10. The results are shown in Table 5.

As shown in Table 5, we can observe that our proposed SGNN-HN can achieve the best performance in terms of both Recall@K and MRR@K when $K = 5$ on three datasets. The same phenomenon can also be observed when $K = 10$. This indicates that SGNN-HN can be applied to practical scenarios with a small number of recommended items. Similar to the results when $K = 20$, CSRМ performs worst among four models for all cases on three

Table 5 Model performance in terms of Recall@K and MRR@K where K = 5 and 10

K	Method	<i>Yoochoose 1/64</i>		<i>Yoochoose 1/4</i>		<i>Diginetica</i>	
		Recall@K	MRR@K	Recall@K	MRR@K	Recall@K	MRR@K
5	CSRM	45.75	27.17	45.51	26.84	25.95	14.39
	SR-IEM	<u>48.05</u>	<u>29.26</u>	<u>48.44</u>	29.37	27.09	15.15
	NISER+	48.03	29.23	48.22	<u>29.43</u>	<u>28.38</u>	<u>16.26</u>
		48.54 [▲]	30.13 [▲]	48.75 [▲]	29.99 [▲]	29.85 [▲]	16.92 [▲]
10	CSRM	58.88	28.94	59.27	28.68	37.88	15.97
	SR-IEM	60.76	<u>30.97</u>	61.18	31.08	38.79	16.70
	NISER+	<u>60.84</u>	30.96	<u>61.35</u>	<u>31.20</u>	<u>40.32</u>	<u>17.84</u>
	SGNN-HN	61.54 [▲]	31.88 [▲]	62.08 [▲]	31.79 [▲]	41.83 [▲]	18.51 [▲]

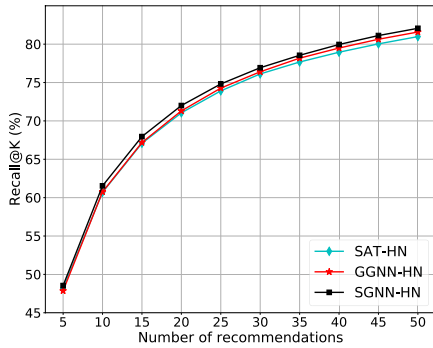
The results of the best performing baseline and the best performer when K = 5 and 10 in each column are underlined and boldfaced, respectively. [▲] denotes a significant improvement of SGNN-HN over the best baseline using a paired *t*-test ($p < 0.01$)

datasets. Moreover, as for SR-IEM and NISER+, when K = 10, NISER+ generally performs better than SR-IEM except the case where NISER+ loses the competition against SR-IEM in terms of MRR on *Yoochoose 1/64*, which is consistent with the findings when K = 20. Differently, when K = 5, SR-IEM outperforms NISER+ in terms of both metrics on *Yoochoose 1/64* and in terms of Recall on *Yoochoose 1/4*. This indicates that by accurately estimating the importance of each item without introducing bias, SR-IEM can recommend the target item at an early position in the recommendation list.

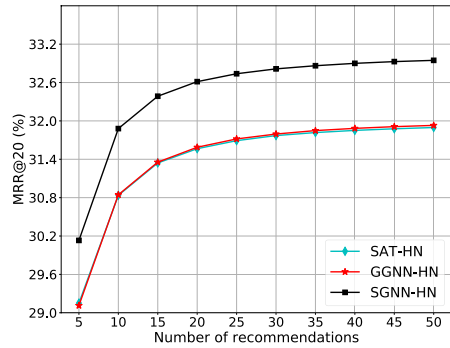
Moreover, the improvement of SGNN-HN against the respective best baselines SR-IEM and NISER+ when K = 5 and 10 are analogous in terms of both Recall@K and MRR@K on *Yoochoose 1/64* and *Yoochoose 1/4*. However, the improvement of SGNN-HN over the best baseline NISER+ on *Diginetica* is more obvious when K = 5 than that when K = 10. Specifically, the improvement in terms of Recall@5 is 5.18% while it is 3.75% in terms of Recall@10; and the improvement in terms of MRR@K are 4.06% and 3.76% when K = 5 and 10, respectively. The difference may be caused by the size of candidate item set of *Yoochoose* and *Diginetica*, which means that, when the number of candidate items is large, SGNN-HN can achieve relatively more obvious improvement with a limited number of recommendations.

5.2 Utility of star graph neural networks

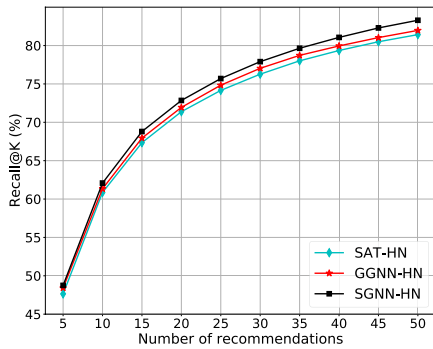
In order to answer RQ2, we compare the star graph neural networks (SGNN) in our proposal with a gated graph neural network (GGNN) (Li et al., 2016) and a self-attention mechanism (SAT) (Vaswani et al., 2017). On the one hand, GGNN can only propagate information from the adjacent nodes while neglecting nodes without direct connections. Correspondingly, we only need to update the satellite nodes at each SGNN layer. In other words, the SGNN will be simplified to GGNN. On the other hand, SAT can be regarded as a full-connection graph that each node can obtain information from all nodes in the graph. In particular, if we remove the GGNN part from the SGNN, set the number of star nodes as the number of items in the session and then replace Eq. (2) by an identity function, i.e., $\mathbf{x}_s = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, where $\mathbf{x}_s \in \mathbb{R}^{d \times n}$, then the SGNN will become a self-attention mechanism. Generally, SGNN can be regarded as a dynamic



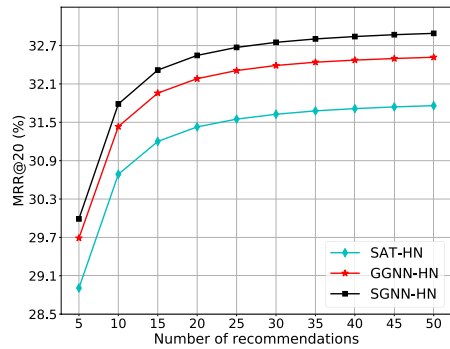
(a) Recall@K on Yoochoose 1/64.



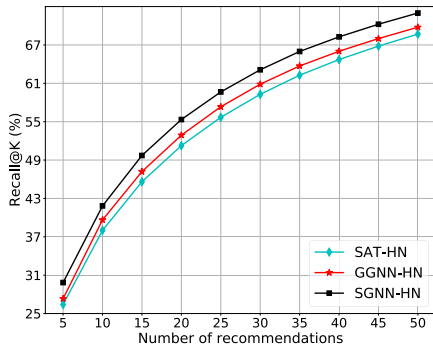
(b) MRR@K on Yoochoose 1/64.



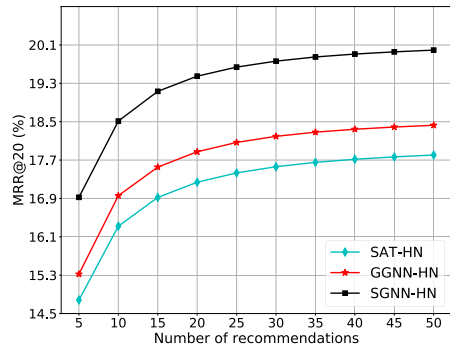
(c) Recall@K on Yoochoose 1/4.



(d) MRR@K on Yoochoose 1/4.



(e) Recall@K on Diginetica.



(f) MRR@K on Diginetica.

Fig. 5 Performance comparison in terms of Recall@20 and MRR@20 to assess the utility of star graph neural networks under different recommendation numbers

combination of GGNN and SAT. To prove the effectiveness of SGNN, we substitute SGNN in our proposal with two alternatives for propagating information between items and evaluate the performance on various recommendation numbers from 5 to 50 in terms of Recall@K and MRR@K on three datasets. The variants are denoted as: (1)

GGNN-HN, which replaces SGNN with a simple GGNN; (2) SAT-HN, which replaces SGNN with GAT. We plot the results in Fig. 5.

As shown in Fig. 5, we can see that SGNN-HN can achieve the best performance for all cases in terms of both Recall@K and MRR@K on three datasets. Moreover, as for the variants, GGNN-HN generally performs better than SAT-HN for all cases on three datasets. We attribute this to the fact that the self-attention mechanism propagates information from all items in a session, which will bring a bias of unrelated items. However, the GNN-based methods, i.e., GGNN-HN and SGNN-HN, can both explore the complex transition relationship between items via GNNs to avoid the bias brought by the unrelated items, thus in general, a better performance is achieved than SAT-HN. Moreover, comparing GGNN-HN to SGNN-HN, we see that GGNN-HN presents a lower performance as GGNN-HN can only propagate information from adjacent items, missing much long-distance information from unconnected items.

5.3 Unity of highway networks

To answer RQ3, we replace the highway networks in SGNN-HN with different aggregation methods, then we compare their performance in terms of Recall@20 and MRR@20 on three datasets. We mainly divide the variants into two categories according to the SGNN layers they combine. Specifically, we mark SGNN combining all layers of SGNNs as *all*, while regard that combining the item embeddings before and after multi-layer SGNNs as *pair*. Here in order to save space, we only list the variants that combine the item embeddings in the way of *all* as an example:

- SGNN-Ave_{all} which combines the item embeddings of all SGNN layers using an average pooling;
- SGNN-SAT_{all} which adopts a self-attention mechanism on the item embeddings of all SGNN layers and then combines them using an average pooling as in Zhang et al. (2018);
- SGNN-GRU_{all} which regards the output of all SGNN layers as a sequence and adopts GRU for combination;
- SGNN-Concat_{all} which combines the item embeddings of all SGNN layers by concatenation;
- SGNN-Gating_{all} which combines the output of all SGNN layers using a gating network (Wang et al., 2019);

Correspondingly, by replacing the output of all SGNN layers in the above variants with the item embeddings before and after multi-layer SGNNs, we can get SGNN-Ave_{pair}, SGNN-SAT_{pair}, SGNN-GRU_{pair}, SGNN-Concat_{pair} and SGNN-Gating_{pair}, respectively. Moreover, we also incorporate the variant that removes the highway networks from SGNN-HN, i.e., SGNN-SR, into comparison, where SGNN-SR directly adopts the output of the last SGNN layer as item representation. The results are given in Table 6.

From Table 6, by comparing SGNN-SR with other methods, we can conclude that combining different SGNN layers can indeed improve the recommendation performance. Clearly, SGNN-HN can generally outperform the variants in terms of both Recall@20 and MRR@20 on three datasets, verifying the effectiveness of the highway networks for solving the serious overfitting problem in GNNs for session-based recommendation. As

Table 6 Model performance in terms of Recall@20 and MRR@20 of SGNN-HN with different aggregation methods for combining information of different SGNN layers

Method	<i>Yoochoose 1/64</i>		<i>Yoochoose 1/4</i>		<i>Diginetica</i>	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
SGNN-SR	71.41	30.55	72.01	31.10	52.04	17.48
SGNN-Ave _{pair}	72.12	31.91	72.76	32.25	55.54	19.21
SGNN-Ave _{all}	71.97	31.71	72.64	31.89	55.47	19.13
SGNN-SAT _{pair}	71.81	31.82	72.67	32.03	54.96	18.91
SGNN-SAT _{all}	71.76	32.10	72.71	32.02	55.15	18.98
SGNN-GRU _{pair}	71.71	31.30	72.65	32.13	54.46	18.65
SGNN-GRU _{all}	70.94	30.74	72.63	32.05	53.78	18.29
SGNN-Gating _{pair}	72.03	32.10	72.78	32.35	55.58	19.20
SGNN-Gating _{all}	72.06	31.90	72.78	32.32	55.60	19.16
SGNN-Concat _{pair}	71.78	31.92	72.71	32.05	54.98	18.90
SGNN-Concat _{all}	71.80	31.81	72.65	32.03	55.05	18.93
SGNN-HN	72.06	32.61	72.85	32.55	55.67	19.45

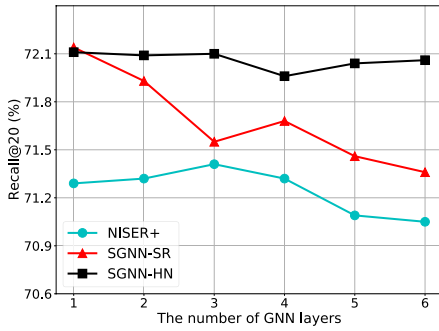
The results of the best performer in each column is boldfaced

for Recall@20, we can see that a simple average pooling can outperform most variants on small datasets *Yoochoose 1/64* and *Diginetica*. This could be due to the fact that there are many parameters in other aggregation method like SAT and GRU, which are hard to be well learnt. Moreover, the most similar method to our highway networks is the gating networks, i.e., *Gating*, where SGNN-Gating_{pair} and SGNN-Gating_{all} also show a competitive performance. The distinction between the highway networks and the gating networks is that the gating in the highway networks is applied to the embedding level while *Gating* works on the item level. Adaptively gating the item representation at the embedding level before and after SGNNs makes the combination flexible, thus SGNN-HN can generate relatively accurate final representation of items.

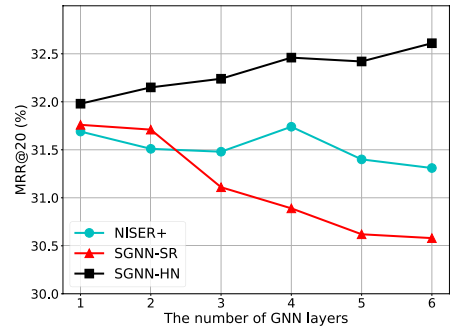
In terms of MRR@20, we can see that different from the phenomenon in terms of Recall@20, most *pair* based models can outperform their corresponding *all* based models for different aggregation methods. This may be due to the fact that the last SGNN layer contains the information from previous SGNN layers, combining all SGNN layers may cause information redundancy, thus resulting in negative influence in ranking the target item in the recommendation list. Moreover, we can observe that the improvement of SGNN-HN above the variants are more obvious in terms of MRR@20 than in terms of Recall@20. For example, on *Diginetica*, the improvement of SGNN-HN against the best variant in terms of Recall@20 and MRR@20 are 0.12% and 1.25%, respectively. This indicates that by combining the item embeddings before and after multi-layer SGNNs, SGNN-HN can make use of both the individual item representation and the neighborhood information from other items, thus push the target item at an early position when making recommendations for users.

5.4 Impact of layer number in SGNNs

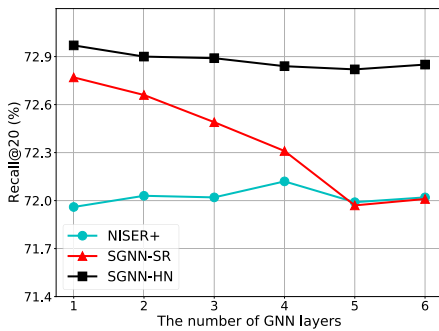
For RQ4, in order to investigate the impact of the number of GNN layers on the proposed SGNN-HN model, we compare SGNN-HN to its variant SGNN-SR, which



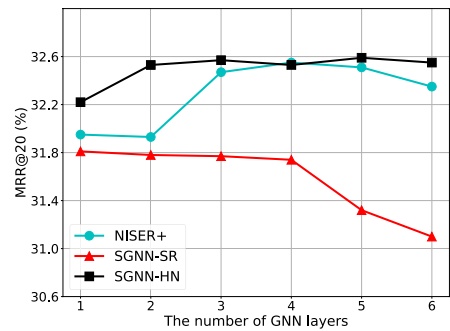
(a) Recall@20 on Yoochoose 1/64.



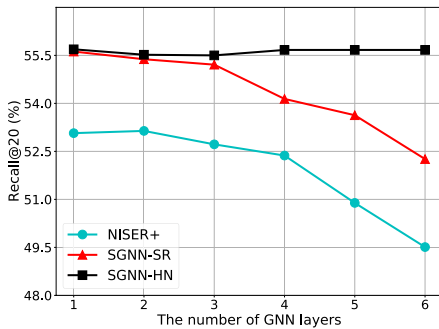
(b) MRR@20 on Yoochoose 1/64.



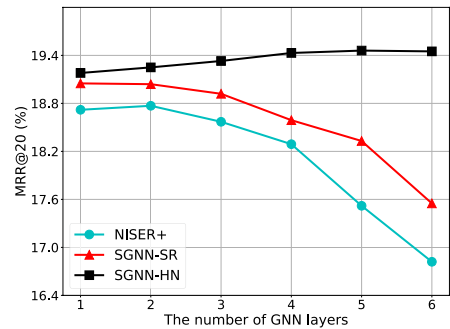
(c) Recall@20 on Yoochoose 1/4.



(d) MRR@20 on Yoochoose 1/4.



(e) Recall@20 on Diginetica.



(f) MRR@20 on Diginetica.

Fig. 6 Model performance in terms of Recall@20 and MRR@20 of different number of GNN layers

removes the highway networks from SGNN-HN. In addition, the comparison involves the best performer in the category of GNN-based methods, i.e., NISER+. Specifically, we increase the number of GNN layers from 1 to 6, to show the performance in terms of Recall@20 and MRR@20 of NISER+, SGNN-SR and SGNN-HN on three datasets. The results are shown in Fig. 6.

As shown in Fig. 6, SGNN-HN achieves the best performance in terms of both Recall@20 and MRR@20 for almost all cases on three datasets. For Recall@20, from Fig. 6a, c and e, we can observe that as the number of GNN layers increases, the performance of both SGNN-SR and NISER+ drops rapidly on the three datasets, especially on *Diginetica*. Graph neural networks for session-based recommendation face a serious overfitting problem. Moreover, SGNN-SR outperforms NISER+ for all cases on the three datasets, which indicates that the proposed SGNN has a better ability than GGNN to represent the transition relationship between items in a session. As to the proposed SGNN-HN model, as the number of layers increases, we can see that the performance in terms of Recall@20 decreases slightly on *Yoochoose 1/64* as well as *Yoochoose 1/4* and remains relatively stable on *Diginetica*. In addition, as the number of layers goes up, the performance of SGNN-HN shows a large gap over SGNN-SR and NISER+. By introducing the highway gating, SGNN-HN can effectively solve the overfitting problem and avoid the rapid decrease in terms of Recall@20 when the number of GNN layers increases.

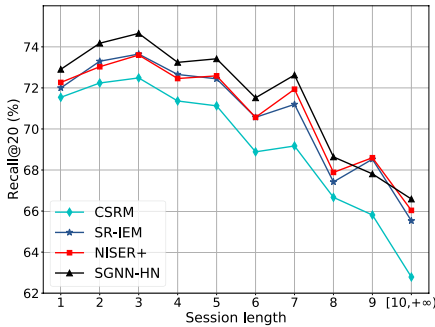
For MRR@20, we can observe that as the number of layers increases, SGNN-SR shows a similar decreasing trend on the three datasets. However, the performance of NISER+ goes down on *Yoochoose 1/64* and *Diginetica* while it increases on *Yoochoose 1/4*. In addition, NISER+ shows a better performance than SGNN-SR with relatively more GNN layers on the *Yoochoose* datasets. Unlike SGNN-SR, we can observe that SGNN-HN achieves the best performance for most cases on the three datasets. Moreover, with the number of layers increasing, the performance of SGNN-HN increases consistently, which may be due to the fact that the highway networks in SGNN-HN can adaptively select the embeddings of item representations before and after the multi-layer SGNNs. Furthermore, comparing SGNN-HN to SGNN-SR, we can see that the improvement brought by the highway networks is more obvious when incorporating more GNN layers. This could be due to the fact that with the highway networks, more SGNN layers can be stacked, thus more information about the transition relationship between items can be obtained.

Moreover, comparing the effect of the highway networks on Recall@20 and MRR@20 in our SGNN-HN model, we can observe that the highway networks can improve the MRR@20 scores as the number of GNN layers increases while maintaining a stable Recall@20 score. This could be due to the fact that SGNN-HN is able to focus on the important items so that it can push the target item at an earlier position by using the highway networks.

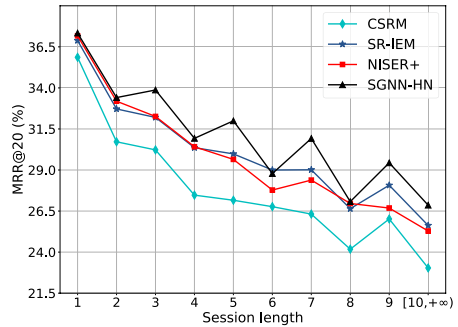
5.5 Impact of the session length

To answer RQ5, we evaluate the performance of SGNN-HN and the state-of-the-art baselines, i.e., CSR, SR-IEM and NISER+ under various session lengths on three datasets. We divide the sessions into groups according to their corresponding length, i.e., 1, 2, ..., 9 and $[10, +\infty)$. Note that for long sessions, we merely take the recent 10 items for training as stated in Sect. 4.4. The result are plotted in Fig. 7.

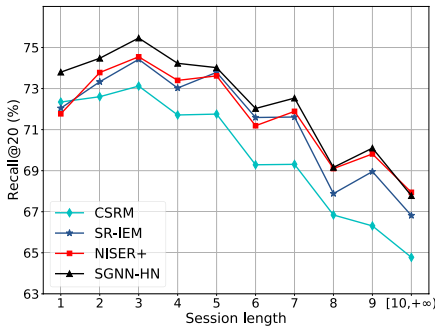
First, from Fig. 7, we can observe that SGNN-HN can generally outperform the baselines in terms of both Recall@20 and MRR@20 in the setting of various session lengths on three datasets, especially on *Diginetica*. Moreover, we can observe that CSR performs worst for most cases on three datasets, especially on sessions with long length, which indicates that the transition relationship in the session is more complicated than a simple sequential signal. As for Recall@20, as shown in Fig. 7a, c and e, we can observe that with the session length increasing, the performance of all models generally first increases and



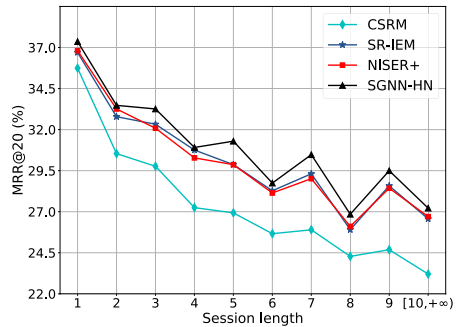
(a) Recall@20 on Yoochoose 1/64.



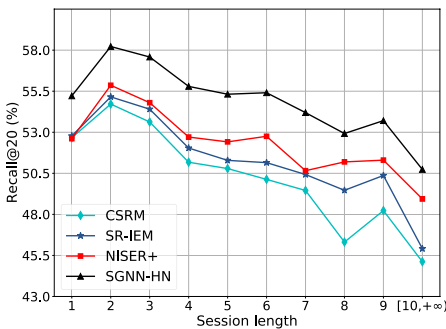
(b) MRR@20 on Yoochoose 1/64.



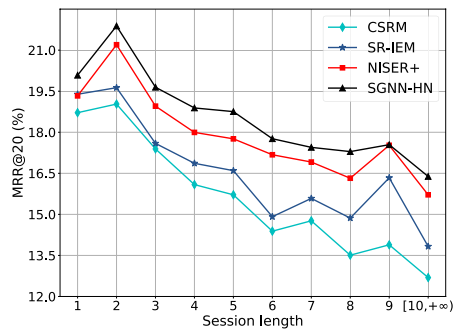
(c) Recall@20 on Yoochoose 1/4.



(d) MRR@20 on Yoochoose 1/4.



(e) Recall@20 on Diginetica.



(f) MRR@20 on Diginetica.

Fig. 7 Model performance in terms of Recall@20 and MRR@20 under various session lengths

then shows a continuing downward trend. This could be explained by the fact that for sessions with short lengths, along with the item number increasing, more information about user purpose can be utilized for intent detection. After achieving the peak performance at a certain session length (e.g., 3 on the *Yoochoose* datasets and 2 on *Diginetica*), bias may be introduced by the unrelated items and thus degrades the recommendation performance. Moreover, comparing NISER+ to SR-IEM, we can observe that their performance are similar on short sessions, especially on the *Yoochoose* datasets, while NISER+ shows

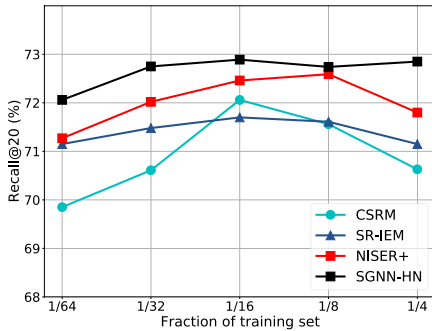
an obvious better performance on sessions with longer lengths. This may be due to that by modeling the complex transition relationship between items, GNNs can well deal with long sessions so as to hit the target item. In addition, comparing SGNN-HN to the baselines, we can see that the gap of the performance between SGNN-HN and the best baseline NISER+ is relatively larger on short sessions than long ones, especially on the *Yoochoose* datasets, which indicates that SGNN-HN is effective on hitting the target item when the user-item interactions are relatively few.

As for MRR@20, the performance in terms of MRR@20 generally decreases on the *Yoochoose* datasets without an upward trend at the beginning like that on *Diginetica*. The could be explained by that the distributions of session length on the *Yoochoose* datasets and *Diginetica* are different. As shown in Fig. 4, the number of sessions decreases more sharply from length 1 to 2 on the *Yoochoose* datasets than that on *Diginetica*, leading to different performance trends. Moreover, NISER+ does not show better performance than SR-IEM for most cases on *Yoochoose 1/64*, and the same phenomenon is observed on *Yoochoose 1/4*. However, SGNN-HN generally shows an obvious improvement than SR-IEM on the three datasets. We attribute the difference of the performance of NISER+ and SGNN-HN to: (1) SGNN makes the information from long-range items available in information propagating; and (2) the highway networks in SGNN-HN paves the way for investigating the complex transition relationship via the multi-layer SGNNs, which helps to hit the target item early in the list of recommended items. In addition, we can observe that the improvement of SGNN-HN against the best baselines SR-IEM and NISER+ is relatively more obvious on long sessions on the *Yoochoose* datasets, while relatively more improvement is achieved on short sessions on *Diginetica*. This difference may be due to the fact that the average session length is different; it is larger on the *Yoochoose* datasets than that on *Diginetica*. As there are proportionally more long sessions in the *Yoochoose* datasets, a larger improvement on sessions of long lengths than of short lengths on the *Yoochoose* datasets is returned.

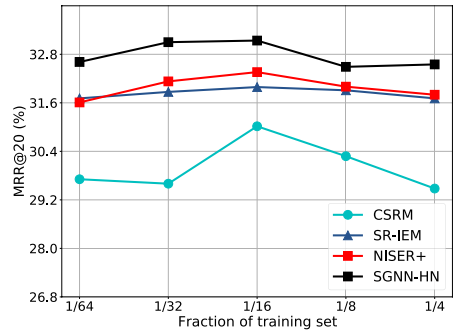
5.6 Sensitivity to the training set scale

To answer RQ6, we preprocess the *Yoochoose* and *Diginetica* datasets by taking different fractions of the training sequences for optimizing the parameters in SGNN-HN and the baselines CSRMM, SR-IEM and NISER+ to investigate the sensitivity of models to different scales of training set. Specifically, we fix the test set as that in Sect. 4.2, where the sessions of the last day and the last week are used for testing in *Yoochoose* and *Diginetica*, respectively. For the *Yoochoose* dataset, we take the recent *1/64*, *1/32*, *1/16*, *1/8* and *1/4* training sequences for learning the parameters. Considering that relatively less training data is contained in the *Diginetica* dataset, the recent *1/16*, *1/8*, *1/4*, *1/2* and *1/1* training sequences are adopted. The performance in terms of Recall@20 and MRR@20 of SGNN-HN and the baselines on different scales of training sets on *Yoochoose* and *Diginetica* are shown in Fig. 8.

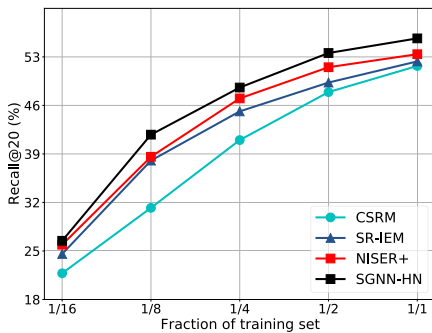
First, we can observe that our proposed SGNN-HN model can beat the baselines for most cases in term of Recall@20 and MRR@20 on both datasets, which indicates the scalability of SGNN-HN that can adapt to scenarios with different amount of training data. On *Yoochoose*, we can observe that the performance of the baselines all increases in the first fractions in terms of both Recall@20 and MRR@20, achieving the peak performance around the *1/16* fraction; and after that they all show a descending trend. We attribute the reason that user's behavior pattern could be influenced by



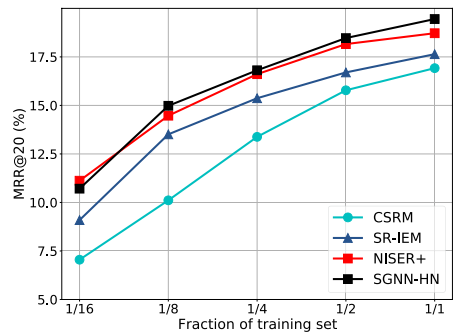
(a) Recall@20 on the Yoochoose dataset.



(b) MRR@20 on the Yoochoose dataset.



(c) Recall@20 on the Diginetica dataset.



(d) MRR@20 on the Diginetica dataset.

Fig. 8 Model performance in terms of Recall@20 and MRR@20 on different scales of training set on Yoochoose and Diginetica

some time-sensitive factors like the season or the fashionable style. For instance on Yoochoose, the shopping behavior data for winter clothes is obviously not suitable for training a recommender used in spring. Moreover, different from the baselines, SGNN-HN shows a stable performance on large fractions, indicating that SGNN-HN can solve the above-mentioned problem to some extent, thus hits the target item accurately using large scales of training sets.

On the Diginetica dataset, which is a smaller dataset than Yoochoose, we can see that the performance of all models generally increases with more training data. Moreover, the gap between NISER+ and SR-IEM is more obvious in terms of MRR@20 than that in terms of Recall@20 on various fractions, indicating that GNNs make sense for pushing the target items. Moreover, we can observe that the performance of SGNN-HN is similar to that of NISER+ on small training fractions. However, the gap of SGNN-HN against NISER+ generally increases with the fraction increasing in terms of both Recall@20 and MRR@20. This may be due to the fact that SGNN-HN contains more learnable parameters in the networks than NISER+, which can be better learnt with large amount of training data.

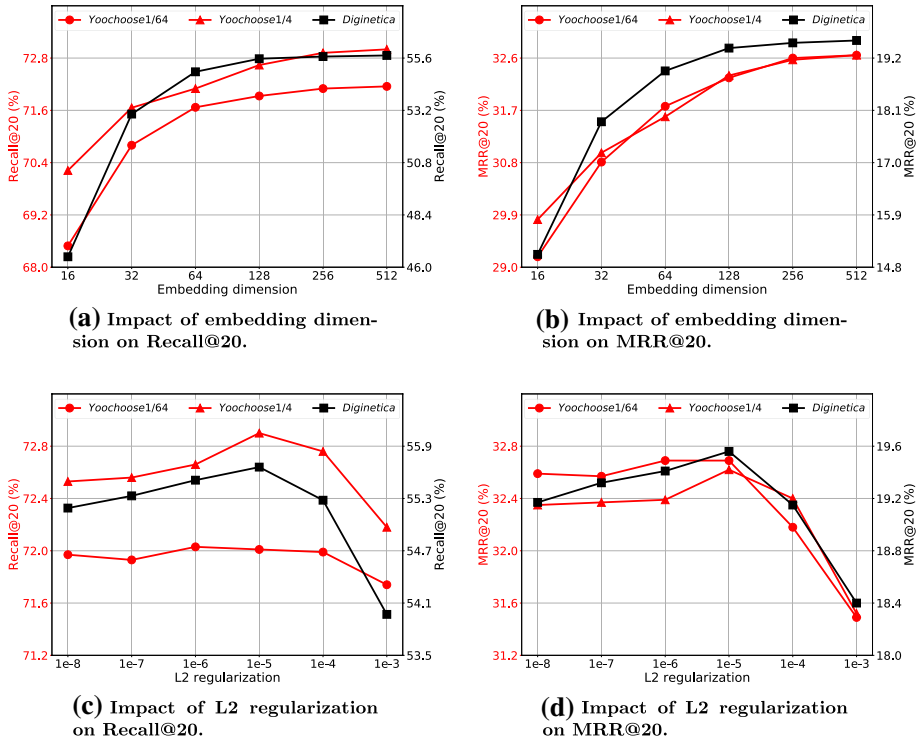


Fig. 9 Model performance in terms of Recall@20 and MRR@20 under different hyper parameters

5.7 Sensitivity to the hyper parameters

To answer RQ7, we conduct experiments to evaluate the scalability of SGNN-HN by studying how the embedding dimension and the L2 regularization affect the recommendation accuracy. Specifically, we set the embedding dimension in {16, 32, 64, 128, 256, 512} and tune the L2 regularization in {1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3}. The performance in terms of Recall@20 and MRR@20 of SGNN-HN on three datasets are presented in Fig. 9.

From Fig. 9a and b, we can observe that with the dimension increasing, the recommendation performance consistently increases, especially from the point of 16 to 32, which is due to that the embeddings of large dimensions have a powerful ability to represent the characters of items. However, we can see that from the point of 256 to 512, the performance improvement in terms of both Recall@20 and MRR@20 are negligible. Considering both the recommendation ability and computational efficiency, we argue the dimension with a moderate number, e.g., 256, is applicable. Moreover, the performance in terms of MRR@20 in *Yoochoose 1/64* and *Yoochoose 1/4* are highly similar, which means that adopting relatively more amount of training sessions not only has no contribution to the ability of ranking the target item when dimension is 256 (as stated in Sect. 5.6), and the same phenomenon also occur under various representation ability of items.

Next, we consider the influence of L2 regularization. Different from the highway networks in SGNN-HN, L2 regularization is used to deal with the issue of overfitting by restricting the trainable parameters in the networks. On *Yoochoose 1/64*, we can see that

Table 7 Computational complexity and efficiency

Method	Time complexity	<i>Yoochoose 1/64</i>		<i>Diginetica</i>		<i>Yoochoose 1/4</i>	
		Training	Test	Training	Test	Training	Test
CSRM	$O(Md^2 + nd^2 + V d)$	4.51	18.24	4.12	18.93	1.24	1.90
SR-IEM	$O(nd^2 + V d)$	0.29	0.33	0.34	0.35	0.11	0.33
NISER+	$O(Lm^2d^2 + nd^2 + V d)$	0.94	0.97	0.91	0.95	0.94	0.96
SGNN-HN	$O(Lm^2d^2 + nd^2 + V d)$	1.00	1.00	1.00	1.00	1.00	1.00

We set the time cost of SGNN-HN in each case to 1 unit, and present the relative time cost of other models

SGNN-HN performs stably in terms of Recall@20 with L2 regularization increasing, and the performance presents a slight drop at the end. Similar phenomenon can be observed in terms of MRR@20, except the case where the performance in terms of MRR@20 shows a sharp decrease from the point of $1e-5$ to $1e-3$. On *Diginetica*, the performance in terms of Recall@20 and MRR@20 first increases, achieving the peak performance at $1e-5$, and then shows a decreasing trend. Similar phenomenon can be found on *Yoochoose 1/4*. This is due to that a proper L2 can help solve the overfitting problem, however, a too large L2 will lead to difficulty in learning the parameters. Hence, $1e-5$ is applicable for L2 regularization.

5.8 Computational complexity

In order to answer RQ8, we theoretically analyze the time complexity of our proposed SGNN-HN and the competitive baselines including CSRM, SR-IEM and NISER+ for modeling a session of length n . We also empirically measure their training time and test time on a single GeForce RTX 3090 GPU. The results are presented in Table 7.

In Table 7, n denotes the session length, d is the embedding dimension, $|V|$ indicates the number of candidate items in the item set V , L means in the layer number of GNNs, m represents the unique item number in the session, and M is the number of incorporated neighbor sessions in CSRM. From the theoretical analysis of time complexity, we can first observe that borrowing the merit from the high efficiency of self-attention mechanism, SR-IEM achieves the lowest time complexity among the compared models. However, the performance of SR-IEM is not satisfactory as shown in Table 4 since it fails to take the complicated item transitions into consideration. Moreover, since $M > Lm^2$, CSRM generally has the highest time complexity because of the large computational cost of RNNs. In addition, we can observe that SGNN-HN theoretically shares the same time complexity as the state-of-the-art GNN-based baseline NISER+, except that SGNN-HN introduces an additional time complexity of nd^2 from the star node and highway networks, which is neglected in the time complexity calculation since $n \ll Lm^2$.

In addition, to clearly compare the time cost of four models for training and test, we set the time consumption of SGNN-HN to 1 unit in each case and present the relative time cost of CSRM, SR-IEM and NISER+. From Table 7, we can observe that the training and test time of four models generally shows a consistent phenomenon with their corresponding theoretical time complexity. First, CSRM costs the most time on all three datasets, especially for test. Moreover, comparing two GNN-based methods, i.e., NISER+ and SGNN-HN, SGNN-HN slightly increases the time cost due to introducing the star node and the highway networks for propagating latent connections between items and alleviating the

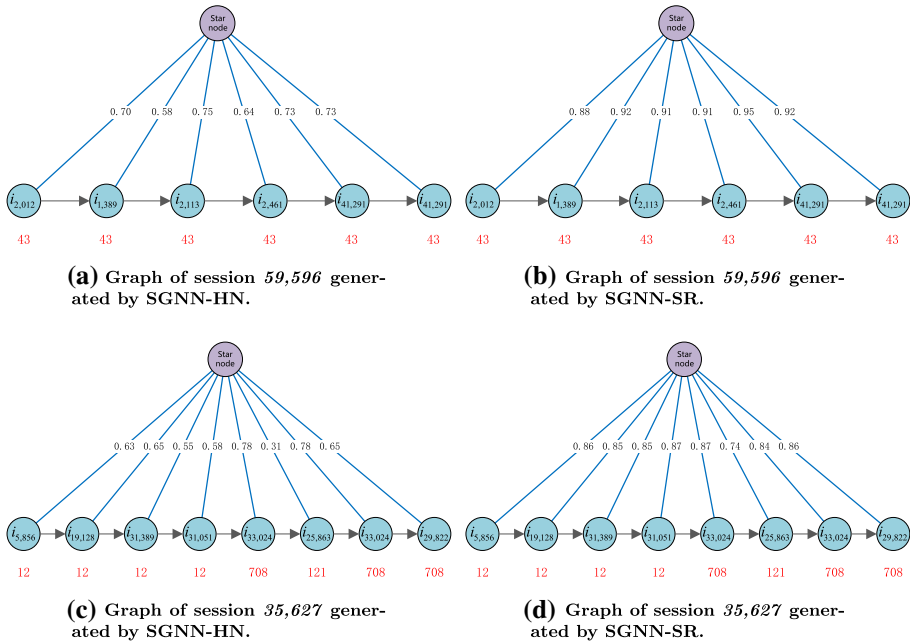


Fig. 10 Illustration of the working mechanism of the proposed SGNN-HN

over-fitting problem, respectively. In addition, we can see that the increasing rates of time consumption are relatively smaller for test than those for training. Specifically, SGNN-HN increases the training time by 6.38, 9.89 and 6.38% above NISER+ on *Yoochoose 1/64*, *Diginetica* and *Yoochoose 1/4*, respectively, while the corresponding increasing rates are 3.09, 5.26 and 4.16% for test. This indicates that after training, our proposal has a comparable time cost with the state-of-the-art baseline during the inference stage when making recommendations for users.

5.9 Case study

To provide a deep insight on the working mechanism of our proposal SGNN-HN, we randomly select two example sessions from *Diginetica*, and show the generated session graph of SGNN-HN as well as SGNN-SR in Fig. 10. The category information in the *Diginetica* dataset marked in red in Fig. 10 can help us capture the relations between items in the session. The purple circle and green circles denotes the star node and the sequential items in the session, respectively. The gray arrows indicates the chronological order between items. Here we omit the satellite connection for brevity. The numbers on the blue lines connecting the star node and each item indicates how much information can be transferred from the star node to the sequential items in the session when updating their embeddings.

From the session graphs in Fig. 10, we can get some interesting findings:

- As in session 35,627, the item 25,863 does not belong to any category of other items, which means that it may be an unrelated item. In this case, we can observe that the corresponding weight is relatively small, i.e., 0.31, while the weights corresponding

to other items are relatively large. This indicates that by introducing the star node, the information from relevant items can be propagated in a two-hop way through the star node, which can effectively avoid the negative impact brought by the unrelated items. Differently, as in session 59,596 where items belong to the same category, the weights between items with the star node are relatively large and similar. This could be explained that, since the sequential items are similar, i.e., in the same category, the learnt star node is also highly similar to items in the session.

- Moreover, comparing the generated graphs of SGNN-HN and SGNN-SR on both sessions, we can observe that the weights of different items are dispersive in SGNN-HN while they are closely similar in SGNN-SR. This could be explained by that multi-layer GNNs face an over-smoothing problem (Xu et al., 2018), making the embeddings of items after SGNNs highly similar and resulting in a difficulty to focus on the important items. However, by utilizing the highway networks, SGNN-HN can effectively solve this problem, which makes it possible to introduce relatively more layers of SGNNs to learn accurate representations of the items.

6 Conclusions and future work

In this paper, we propose a novel approach, i.e., Star Graph Neural Networks with Highway Networks (SGNN-HN), for session-based recommendation. SGNN-HN applies the star graph neural networks (SGNN) to model the complex transition relationship between items in a session to generate accurate item embeddings. Moreover, to deal with the overfitting problem of graph neural networks in session-based recommendation, we utilize the highway networks to dynamically combine information from item embeddings before and after multi-layer SGNNs. Finally, we apply an attention mechanism to combine item embeddings in a session as user's general preference, which is then concatenated with user's recent interest expressed by the last clicked item in the session for making item recommendation. Experiments are conducted on two public benchmark datasets, i.e., *Yoochoose* and *Diginetica*, and the results show that SGNN-HN can significantly improve the recommendation performance in terms of Recall and MRR metrics. Moreover, we provide a detailed analysis on star graph neural networks (SGNN) and the highway networks by replacing the respective module with other alternatives for comparison to verify their respective effectiveness. Furthermore, we investigate the impact of the layer number of SGNN-HN on the recommendation ability, and examine how SGNN-HN performs for cases with different session lengths. We find that deep SGNN layers contribute more to the recommendation accuracy in terms of MRR@20 than Recall@20. In addition, the improvement in terms of Recall@20 is relatively more obvious on short sessions than long ones.

As to future work, we would like to incorporate the neighbor sessions to enrich the transition relationship in the current session. Moreover, we are interested in applying star graph neural networks to other tasks like conversational recommendation (Lei et al., 2020) and dialogue systems (Zhang et al., 2019) to investigate its scalability. In addition, we plan to take the multi behaviors (Gu et al., 2020; Jin et al., 2020) in e-commerce recommender systems into consideration for user purpose modeling.

Acknowledgements This work was partially supported by the Postgraduate Scientific Research Innovation Project of Hunan Province under No. CX20200055. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Declarations


Conflict of interest The authors declare that they have no conflict of interest.

References

- Abdollahpouri, H., Burke, R., & Mobasher, B. (2017). Controlling popularity bias in learning-to-rank recommendation. In *RecSys'17* (pp. 42–46).
- Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749.
- Chen, W., Cai, F., Chen, H., & de Rijke, M. (2019). Joint neural collaborative filtering for recommender systems. *ACM Transactions on Information Systems*, *37*(4), 39–13930.
- Chen, W., Cai, F., Chen, H., & de Rijke, M. (2019). A dynamic co-attention network for session-based recommendation. In *CIKM'19* (pp. 1461–1470).
- Gu, Y., Ding, Z., Wang, S., & Yin, D. (2020). Hierarchical user profiling for e-commerce recommender systems. In *WSDM'20* (pp. 223–231).
- Guo, J., Zhu, X., Lan, Y., & Cheng, X. (2017). Modeling users' search sessions for high utility query recommendation. *Information Retrieval Journal*, *20*(1), 4–24.
- Guo, Q., Qiu, X., Liu, P., Shao, Y., Xue, X., & Zhang, Z. (2019). Star-transformer. In *NAACL'19* (pp. 1315–1325).
- Gupta, P., Garg, D., Malhotra, P., Vig, L., & Shroff, G. M. (2019). NISER: Normalized item and session representations with graph neural networks. arXiv preprint [arXiv:1909.04276](https://arxiv.org/abs/1909.04276).
- He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural collaborative filtering. In *WWW'17* (pp. 173–182).
- Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In *ICLR'16*.
- Hidasi, B., & Karatzoglou, A. (2018). Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM'18* (pp. 843–852).
- Hu, D., Wei, L., Zhou, W., Huai, X., Fang, Z., & Hu, S. (2021). Pen4rec: Preference evolution networks for session-based recommendation. In *KSEM'21*, vol. 12815 (pp. 504–516).
- Jannach, D., & Ludewig, M. (2017). When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys'17* (pp. 306–310).
- Jin, R., Chai, J. Y., & Si, L. (2004). An automatic weighting scheme for collaborative filtering. In *SIGIR'04* (pp. 337–344).
- Jin, B., Gao, C., He, X., Jin, D., & Li, Y. (2020). Multi-behavior recommendation with graph convolutional networks. In *SIGIR'20* (pp. 659–668).
- Kang, W., & McAuley, J. J. (2018). Self-attentive sequential recommendation. In *ICDM'18* (pp. 197–206).
- Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR'17*.
- Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *KDD'08* (pp. 426–434).
- Lei, W., He, X., Miao, Y., Wu, Q., Hong, R., Kan, M., & Chua, T. (2020). Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *WSDM'20* (pp. 304–312).
- Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., & Ma, J. (2017). Neural attentive session-based recommendation. In *CIKM'17* (pp. 1419–1428).
- Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. S. (2016). Gated graph sequence neural networks. In *ICLR'16*.
- Liu, Q., Zeng, Y., Mokhosi, R., & Zhang, H. (2018). STAMP: Short-term attention/memory priority model for session-based recommendation. In *KDD'18* (pp. 1831–1839).
- Pan, Z., Cai, F., Chen, W., Chen, H., & de Rijke, M. (2020). Star graph neural networks for session-based recommendation. In *CIKM'20* (pp. 1195–1204).
- Pan, Z., Cai, F., Ling, Y., & de Rijke, M. (2020). Rethinking item importance in session-based recommendation. In *SIGIR'20* (pp. 1837–1840).
- Qiu, R., Huang, Z., Li, J., & Yin, H. (2020). Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Transactions on Information Systems*, *38*(3), 22–12223.

- Qiu, R., Li, J., Huang, Z., & Yin, H. (2019). Rethinking the item order in session-based recommendation with graph neural networks. In *CIKM'19* (pp. 579–588).
- Ren, P., Chen, Z., Li, J., Ren, Z., Ma, J., & de Rijke, M. (2019). Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *AAAI'19* (pp. 4806–4813).
- Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. In *WWW'10* (pp. 811–820).
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.
- Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW'01* (pp. 285–295).
- Song, W., Xiao, Z., Wang, Y., Charlin, L., Zhang, M., & Tang, J. (2019). Session-based social recommendation via dynamic graph attention networks. In *WSDM'19* (pp. 555–563).
- Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. arXiv preprint [arXiv:1505.00387](https://arxiv.org/abs/1505.00387).
- Sukhbaatar, S., Szlam, A., Weston, J., & Fergus, R. (2015). End-to-end memory networks. In *NeurIPS'15* (pp. 2440–2448).
- Sun, K., Qian, T., Yin, H., Chen, T., Chen, Y., & Chen, L. (2019). What can history tell us?. In *CIKM'19* (pp. 1593–1602).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *NeurIPS'17* (pp. 5998–6008).
- Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *ICLR'18*.
- Vinyals, O., Bengio, S., & Kudlur, M. (2016). Order matters: Sequence to sequence for sets. In *ICLR'16*.
- Wang, M., Ren, P., Mei, L., Chen, Z., Ma, J., & de Rijke, M. (2019). A collaborative session-based recommendation approach with parallel memory modules. In *SIGIR'19* (pp. 345–354).
- Wang, Z., Wei, W., Cong, G., Li, X., Mao, X., & Qiu, M. (2020). Global context enhanced graph neural networks for session-based recommendation. In *SIGIR'20* (pp. 169–178).
- Wang, F., Xiang, X., Cheng, J., & Yuille, A. L. (2017). Normface: L_2 hypersphere embedding for face verification. In *MM'17* (pp. 1041–1049).
- Wang, X., He, X., Wang, M., Feng, F., & Chua, T. (2019). Neural graph collaborative filtering. In *SIGIR'19* (pp. 165–174).
- Weston, J., Chopra, S., & Bordes, A. (2015). Memory networks. In *ICLR'15*.
- Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., & Tan, T. (2019). Session-based recommendation with graph neural networks. In *AAAI'19* (pp. 346–353).
- Xu, C., Zhao, P., Liu, Y., Sheng, V. S., Xu, J., Zhuang, F., Fang, J., & Zhou, X. (2019). Graph contextualized self-attention network for session-based recommendation. In *IJCAI'19* (pp. 3940–3946).
- Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *ICML'18* (pp. 5449–5458).
- Ying, H., Zhuang, F., Zhang, F., Liu, Y., Xu, G., Xie, X., Xiong, H., & Wu, J. (2018). Sequential recommender system based on hierarchical attention networks. In *IJCAI'18* (pp. 3926–3932).
- Yu, F., Liu, Q., Wu, S., Wang, L., & Tan, T. (2016). A dynamic recurrent model for next basket recommendation. In *SIGIR'16* (pp. 729–732).
- Zhang, Z., Zhang, Y., & Ren, Y. (2020). Employing neighborhood reduction for alleviating sparsity and cold start problems in user-based collaborative filtering. *Information Retrieval Journal*, *23*(4), 449–472.
- Zhang, S., Tay, Y., Yao, L., & Sun, A. (2018). Next item recommendation with self-attention. arXiv preprint [arXiv:1808.06414](https://arxiv.org/abs/1808.06414).
- Zhang, H., Lan, Y., Pang, L., Guo, J., & Cheng, X. (2019). Recosa: Detecting the relevant contexts with self-attention for multi-turn dialogue generation. In *ACL'19* (pp. 3721–3730).

Authors and Affiliations

Fei Cai¹ · Zhiqiang Pan¹  · Chengyu Song¹ · Xin Zhang¹

Fei Cai
caifei08@nudt.edu.cn

Chengyu Song
songchengyu@nudt.edu.cn

Xin Zhang
zhangxin16@nudt.edu.cn

¹ Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, Hunan 410073, China