Star Graph Neural Networks for Session-based Recommendation

Pan, Z.; Cai, F.; Chen, W.; Chen, H.; de Rijke, M.

DOI
10.1145/3340531.3412014

Publication date
2020

Document Version
Author accepted manuscript

Published in
CIKM '20

Star Graph Neural Networks for Session-based Recommendation

Zhiqiang Pan[1] Fei Cai[1]* Wanyu Chen[1] Honghui Chen[1] Maarten de Rijke[2,3]
[1]Science and Technology on Information Systems Engineering Laboratory,
National University of Defense Technology, Changsha, China
[2]University of Amsterdam, Amsterdam, The Netherlands
[3]Ahold Delhaize, Zaandam, The Netherlands
{panzhiqiang, caifei, wanyuchen, chenhonghui}@nudt.edu.cn, derijke@uva.nl

ABSTRACT

Session-based recommendation is a challenging task. Without access to a user's historical user-item interactions, the information available in an ongoing session may be 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 complex transition relationship between items that go beyond inspection order. Thus graph neural network (GNN) based models have been proposed to capture the transition relationship between items. However, GNNs typically propagate information from adjacent items only, thus neglecting information from items without direct connections. Importantly, GNN-based approaches often face serious overfitting problems.

We propose Star Graph Neural Networks with Highway Networks (SGNN-HN) for session-based recommendation. The proposed SGNN-HN applies a star graph neural network (SGNN) to model the complex transition relationship between items in an ongoing session. To avoid overfitting, we employ highway networks (HN) to adaptively select embeddings from item representations. Finally, we aggregate the item embeddings generated by the SGNN in an ongoing session to represent a user's final preference for item prediction. Experiments on two public benchmark datasets show that SGNN-HN can outperform state-of-the-art models in terms of P@20 and MRR@20 for session-based recommendation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Session-based recommendation, graph neural networks, highway networks.

ACM Reference Format:

Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen and Maarten de Rijke. 2020. Star Graph Neural Networks for Session-based Recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3412014

*Corresponding author.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412014

1 INTRODUCTION

Recommender systems can help people obtain personalized information [2], 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 [4, 8, 9], Factorizing Personalized Markov Chains (FPMC) [20], and deep learning based methods [23, 32, 37]. For cases where a user's long-term historical interactions are unavailable, e.g., a new user, it is challenging to capture her preferences in an accurate manner [10]. The session-based recommendations task is to generate recommendations based only on the ongoing session.

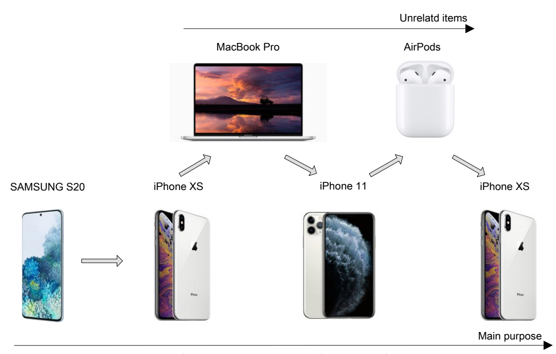


Figure 1: An example session, where the gray arrows indicate the sequential order of items, and the items *MacBook Pro* and *AirPods* are regarded as "unrelated items" since they are uncorrelated to the user's main purpose.

Existing methods for session-based recommendation mostly focus on modeling the sequential information of items based on Recurrent Neural Networks (RNNs) [10]. Moreover, Li et al. [14] apply an attention mechanism to capture a user's main purpose. However, as stated in [19], 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. Consider, for instance, the example session in Fig. 1, where a user wants to buy a phone but issues clicks on an *Macbook Pro* and *AirPods* out of curiosity. Such complex sequential interactions – with unrelated items – will mislead an RNN and attention mechanism so that it will fail to accurately identify the user's main purpose. To more accurately model the transition pattern of items, graph neural networks (GNNs) have been utilized to model an ongoing session [19, 34]. However, GNN-based methods mostly propagate information between adjacent items, thus neglecting information from items without direct connections. Multi-layer GNNs have subsequently been employed to propagate information between items without direct connections; however, they easily lead to overfitting as multiple authors have pointed out [19, 34].

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 SGNN, which can help to explore complex transition relationship between items. Finally, 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 P@20 and MRR@20.

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 results show the superiority of SGNN-HN over the state-of-the-art models in terms of P@20 and MRR@20.

2 RELATED WORK

We review related work from four angles: general recommendation models, sequential recommendation models, attention-based models, and GNN-based models.

2.1 General recommendation models

Collaborative Filtering (CF) has been widely applied in recommender systems to capture user's general preferences according to their historical interactions with items [9]. Many CF methods consider the user-item interaction matrix based on Matrix Factorization (MF) [39]. In recent years, neural network based methods have been applied to CF. For instance, He et al. [8] propose Neural Collaborative Filtering (NCF), which utilizes a multi-layer perceptron to express and generalize matrix factorization in a non-linear way. Chen et al. [4] propose a Joint Neural Collaborative Filtering (J-NCF) to learn deep features of users and items by fully exploring the user-item interactions. Some neighborhood-based CF models have been proposed to concentrate on the similarity among users or items. For instance, Item-KNN [22] has been utilized in session-based recommendation by focusing on the similarity among items in terms of co-occurrences in other sessions [5].

Importantly, even though collaborative filtering-based methods can capture a user's general preferences, it is unable to detect changes in user interest; as a consequence, they cannot generate recommendations that instantly adapt to a user's recent needs.

2.2 Sequential recommendation models

To capture the sequential signal in interactions, Markov Chains (MCs) have been widely applied. For instance, Rendle et al. [20] propose Factorizing Personalized Markov Chains (FPMC) to capture both user's sequential behaviors and long-term interest. Wang et al. [31] propose a Hierarchical Representation Model (HRM) to improve the performance of FPMC by a hierarchical architecture to non-linearly combine the user vector and the sequential signal.

Deep learning methods like RNNs have also been widely applied in recommender systems. For instance, Hidasi et al. [10] propose GRU4REC to apply GRUs in session-based recommendation; they utilize a session-parallel minibatch for training. Li et al. [14] propose a Neural Attentive Recommendation Machine (NARM) to extend GRU4REC by using an attention mechanism to capture user's main purpose. Furthermore, as information contained in an ongoing session may be very limited, neighbor information is introduced to help model an ongoing session [11, 17, 30]. For instance, Jannach and Ludewig [11] introduce neighbor sessions with a K-nearest neighbor method (KNN). Unlike traditional KNN, Wang et al. [30] propose a Collaborative Session-based Recommendation Machine (CSRM) to incorporate neighbor sessions as auxiliary information via memory networks [25, 33].

However, Markov Chain-based models can only capture information from adjacent transactions, making it hard to adequately capture interest migration. Moreover, as to RNN-based methods, interaction patterns tend to be more complex than the simple sequential signal in an ongoing session [19].

2.3 Attention based models

As items in a session have different degrees of importance, many recommendation methods apply an attention mechanism to distinguish the item importance. For instance, Liu et al. [16] apply an attention mechanism to obtain a user's general preference and recent interest 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. [18] propose to measure item importance using an importance extraction module, and consider the global preference and recent interest to make item prediction. Furthermore, for the cases that a user's historical interactions are available, Ying et al. [36] propose a two-layer hierarchical attention network that takes both user's long-term and short-term preferences into consideration. In addition, to account for the influence of items in a user's long-term and short-term interactions, Chen et al. [3] propose to utilize a co-attention mechanism.

However, attention-based methods merely focus on the relative importance of items in a session, without considering the complex transition relationship between items in an ongoing session.

2.4 Graph neural networks based models

Recently, because of their ability to model complex relationships between objects, graph neural networks (GNNs) [12, 15, 28] have

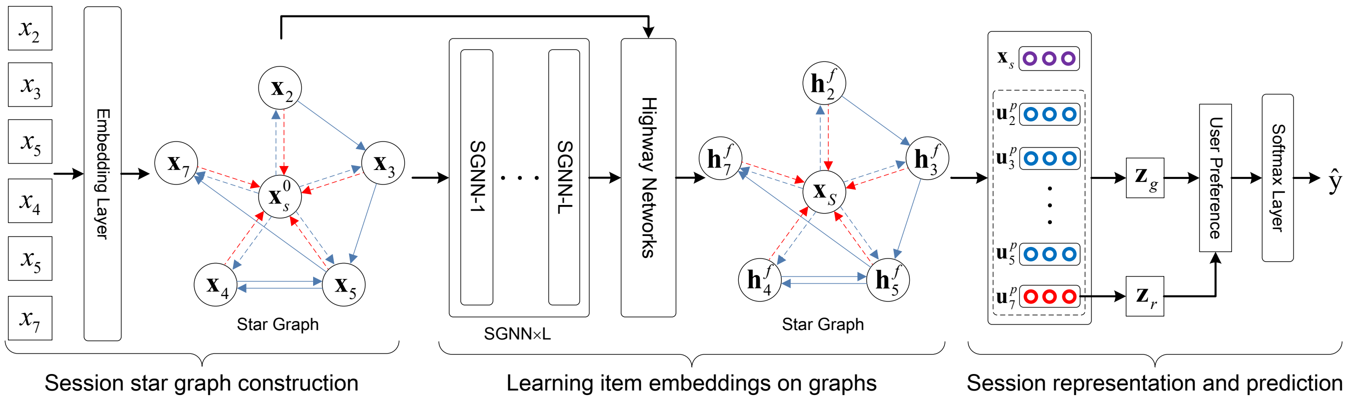


Figure 2: The workflow of SGNN-HN.

attracted much interest in the context of recommender systems. For instance, Song et al. [23] propose to introduce graph attention networks into social recommendation for modeling both dynamic user interests and context-dependent social influences. Wang et al. [32] introduce graph neural networks into collaborative filtering for making recommendations. In addition, GNNs have been utilized to model the complex transition relationship between items in session-based recommendation. For instance, Wu et al. [34] 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 session representations. Xu et al. [35] extend SR-GNN with a self-attention mechanism to obtain contextualized non-local representations for producing recommendations. Furthermore, Qiu et al. [19] propose a weighted graph attention network (WGAT) based approach for generating item representation, which is then aggregated by a Readout function as user preference for item prediction.

The GNN-based methods listed above propagate information between adjacent items, failing to take unconnected items into consideration. Moreover, GNN-based methods for session-based recommendation tend to overfit easily, thereby limiting the ability to investigate complex transition relationship.

3 APPROACH

In this section, we first formulate the task of session-based recommendation. Then we detail the SGNN-HN model, which consists of three main components, i.e., session star graph construction (see Section 3.1), learning item embeddings on star graphs (see Section 3.2), and generating session representations and predictions (see Section 3.3).

The workflow of SGNN-HN is plotted in Fig. 2. First, a d-dimension embedding $\mathbf{x}_i \in \mathbb{R}^d$ is generated for each unique item x_i in the session through an embedding layer, and each session is constructed as a session star graph. Then the nodes (item embeddings) in the graph are input into the multi-layer star graph neural networks (SGNN). After that, we combine the item embeddings before and after SGNN using the highway networks. Finally, the session is represented by combining a general preference and a recent interest in the session. After obtaining the session representation, recommendations are generated by calculating the scores on all candidate items.

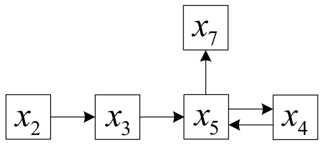
The goal of session-based recommendation is to predict the next item to click based on the ongoing session. We formulate this task as follows. Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ denote all unique items in all sessions, where $|V|$ is the number of all unique items. Given a session as $S = \{v_1, v_2, \ldots, v_t, \ldots, v_n\}$ consisting of n sequential items, where $v_t \in V$ is the t-th item in the session, we aim to predict v_{n+1} to click. Specifically, we output the probability of all items $\hat{\mathbf{y}} = \{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_{|V|}\}$, where $\hat{\mathbf{y}}_i \in \hat{\mathbf{y}}$ indicates the likelihood score of clicking item i. Then, the items with the highest top-K scores in $\hat{\mathbf{y}}$ will be recommended to the user.

3.1 Session star graph construction

For each session $S = \{v_1, v_2, \ldots, v_t, \ldots, v_n\}$, we construct a star graph to represent the transition relationship among the items in the session. We not only consider the adjacent clicked items, but the items without direct connections by adding a star node, which leads to a full connection of all nodes in the session star graph. Specifically, each session is then represented as $G_s = \{\mathcal{V}_s, \mathcal{E}_s\}$, where $\mathcal{V}_s = \{\{x_1, x_2, \ldots, x_m\}, x_s\}$ denotes the node set of the star graph. Here, the first part $\{x_1, x_2, \ldots, x_m\}$ indicates all unique items in the session, which we call *satellite nodes*, and the latter part x_s is the newly added *star node*. Note that $m \leq n$ since there may exist repeated items in the session. $\mathcal{E}_s$ is the edge set in the star graph, which consists of two types of edges, i.e., satellite connections and star connections, used for propagating information from satellite nodes and star node, respectively.

Satellite connections. We utilize satellite connections to represent the adjacency relationship between items in a session. Here, we use the gated graph neural networks (GGNN) [15] as an example. Actually, other graph neural networks like GAT [28] and GCN [12] can also be utilized to replace GGNN in our model. For the satellite connections, the edges $(x_i, x_j) \in \mathcal{E}_s$, i.e., the blue solid lines in the star graphs in Fig. 2, mean that the user clicks item x_j after clicking x_i. In this way, the adjacent relationship between two items in the session can be represented by an incoming matrix and an outgoing matrix. Considering an example session $S = \{x_2, x_3, x_5, x_4, x_5, x_7\}$, we can construct the incoming matrix and the outgoing matrix of GGNN as in Fig. 3.

Star connections. Inspired by [6], we add a star node to propagate information from non-adjacent items, More specifically, we add a



Outgoing Matrix

	2	3	5	4	7
2	0	1	0	0	0
3	0	0	1	0	0
5	0	0	0	1/2	1/2
4	0	0	1	0	0
7	0	0	0	0	0

Incoming Matrix

	2	3	5	4	7
2	0	0	0	0	0
3	1	0	0	0	0
5	0	1/2	0	1/2	0
4	0	0	1	0	0
7	0	0	1	0	0

Figure 3: An example of adjacent matrices in GGNN.

bidirectional edge between the star node and each satellite node in the star graph. On the one hand, we use the directed edges from the star node to the satellite nodes, i.e., the blue dotted lines $(x_s, x_i) \in \mathcal{E}_s$ in the star graphs in Fig. 2, for updating the satellite nodes, where x_i is a satellite node. Via the star node, information from non-adjacent nodes can be propagated in a two-hop way by taking the star node as an intermediate node. On the other hand, the directed edges from the satellite nodes to the star node, i.e., the red dotted lines $(x_i, x_s) \in \mathcal{E}_s$ in the star graphs in Fig. 2, are used to update the star node, which helps generate an accurate representation of the star node by considering all nodes in the star graph.

Unlike the basic graph neural networks, we consider the connection relationship from both the adjacency matrices and the bidirectional connections introduced by the star node, where the adjacency matrices are constructed according to the sequential click order, and a gating network is utilized to dynamically control how much information should be obtained from the adjacent nodes and the star node, respectively.

3.2 Learning item embeddings on graphs

Next, we present how star graph neural networks (SGNN) can propagate information between the nodes of the graph. In the SGNN, there are two main stages, i.e., *satellite node representing* and *star node representing*.

3.2.1 Initialization. Before passing the nodes into the SGNN, we first initialize the representation of satellite nodes and the star node. For the satellite nodes, we directly take the embeddings of unique items in the session as the satellite nodes representation:

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

where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional embedding of satellite node i in the star graph. As for the star node, denoted as $\mathbf{x}_s^0$, we apply an average pooling on the satellite nodes to get the initialization of the star node, i.e.,

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

3.2.2 Update. To learn a node representation according to the graph, we update the satellite nodes and the star node as follows.

Satellite nodes update. For each satellite node, the neighbor nodes for propagating information are from two types of sources, i.e., the adjacent nodes and the star node, which corresponds to the information from nodes with and without direct connections, respectively. Compared to GGNN [15] and GAT [28], the star node in the proposed SGNN can make information from unconnected items available for propagation across two hops.

First, we consider the information from adjacent nodes. For each satellite node x_i in the star graph at layer l, we utilize the incoming and the outgoing matrices to obtain the propagation information as follows:

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

where $\mathbf{A}_i^I, \mathbf{A}_i^O \in \mathbb{R}^{1\times m}$ are the corresponding incoming and outgoing weights for node x_i, i.e., the i-th row in the incoming matrix and the outgoing matrix. $\mathbf{W}^I, \mathbf{W}^O \in \mathbb{R}^{d\times d}$ are the learnable parameters for the incoming edges and the outgoing edges, respectively, while $\mathbf{b}^I, \mathbf{b}^O \in \mathbb{R}^d$ are the bias vectors. Hence, we can obtain $\mathbf{a}_i^l \in \mathbb{R}^{1\times 2d}$ to represent the propagation information for node x_i. After that, we feed $\mathbf{a}_i^l$ and node x_i's previous state $\mathbf{h}_i^{l-1}$ into the gated graph neural networks as follows:

$$\begin{aligned}\mathbf{z}_i^l &= \sigma(\mathbf{W}_z\mathbf{a}_i^l + \mathbf{U}_z\mathbf{h}_i^{l-1}), \\ \mathbf{r}_i^l &= \sigma(\mathbf{W}_r\mathbf{a}_i^l + \mathbf{U}_r\mathbf{h}_i^{l-1}), \\ \tilde{\mathbf{h}}_i^l &= \tanh(\mathbf{W}_h\mathbf{a}_i^l + \mathbf{U}_h(\mathbf{r}_i^l \odot \mathbf{h}_i^{l-1})), \\ \hat{\mathbf{h}}_i^l &= (1 - \mathbf{z}_i^l) \odot \mathbf{h}_i^{l-1} + \mathbf{z}_i^l \odot \tilde{\mathbf{h}}_l,\end{aligned} \tag{4}$$

where $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h \in \mathbb{R}^{d\times 2d}$ and $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_h \in \mathbb{R}^{d\times d}$ are trainable parameters in the network. σ represents the activation function *sigmoid* in above formulas and $\odot$ is the element-wise multiplication. $\mathbf{z}_i^l$ and $\mathbf{r}_i^l$ are the update gate and the reset gate, which controls how much information in the former state $\mathbf{h}_i^{l-1}$ should be preserved and how much information of the previous state should be written in the candidate activation $\tilde{\mathbf{h}}_i^l$, respectively. In this way, information from adjacent items can be propagated in star graph neural networks.

Next, we consider information from the star node. In star graph neural networks, the star node can represent the overall information from all items in a session. For each satellite node, we apply a gating network to decide how much information should be propagated from the star node and the adjacent nodes, respectively. Specifically, we calculate the similarity α_i^l of each satellite node x_i and the star node x_s with a self-attention mechanism as:

$$\alpha_i^l = \frac{(\mathbf{W}_{q1}\hat{\mathbf{h}}_i^l)^\top \mathbf{W}_{k1}\mathbf{x}_s^{l-1}}{\sqrt{d}}, \tag{5}$$

where $\mathbf{W}_{q1}, \mathbf{W}_{k1} \in \mathbb{R}^{d\times d}$ are trainable parameters, $\hat{\mathbf{h}}_i^l$ and $\mathbf{x}_s^{l-1}$ are the corresponding item representation of x_i and x_s, respectively, and $\sqrt{d}$ is used for scaling the coefficient. Finally, we apply a gating network to selectively integrate the information from the adjacent node $\hat{\mathbf{h}}_i^l$ and from the star node $\mathbf{x}_s^{l-1}$ as follows:

$$\mathbf{h}_i^l = (1 - \alpha_i^l)\hat{\mathbf{h}}_i^l + \alpha_i^l\mathbf{x}_s^{l-1}. \tag{6}$$

Star node update. For updating the star node based on the new representation of satellite nodes, we introduce a self-attention mechanism to assign different degrees of importance to the satellite nodes by regarding the star node as *query*. First, the importance of each satellite node is determined by the star node:

$$\beta = \text{softmax}\left(\frac{\mathbf{K}^\top \mathbf{q}}{\sqrt{d}}\right) = \text{softmax}\left(\frac{(\mathbf{W}_{k2}\mathbf{h}^l)^\top \mathbf{W}_{q2}\mathbf{x}_s^{l-1}}{\sqrt{d}}\right), \quad (7)$$

where $\mathbf{q} \in \mathbb{R}^d$ and $\mathbf{K} \in \mathbb{R}^{d\times m}$ are transformed from the star node and the satellite nodes, respectively. $\mathbf{W}_{q2}, \mathbf{W}_{k2} \in \mathbb{R}^{d\times d}$ are the corresponding trainable parameters. After obtaining the degrees of importance, we combine the satellite nodes using a linear combination as a new representation of the star node:

$$\mathbf{x}_s^l = \beta \mathbf{h}^l, \quad (8)$$

where $\beta \in \mathbb{R}^m$ are the weights of all satellite nodes.

3.2.3 Highway networks. By updating the satellite nodes and the star node iteratively, we can obtain accurate item representations. Moreover, in order to accurately represent the transition relationship between items, multi-layer SGNNs can be stacked, where the l-layer of SGNN can be denoted as:

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

However, the multi-layer graph layers can propagate large amounts of information between nodes, which easily brings bias, leading to overfitting problems with the graph neural networks as stated in [19, 34]. To address this problem, we apply the highway networks [24] to selectively obtain information from the item embeddings before and after the multi-layer SGNNs. As we employ an L-layer SGNN, the item embeddings of the satellite nodes before and after the SGNN layers are denoted as $\mathbf{h}^0$ and $\mathbf{h}^L$, respectively. Then, the highway networks can be denoted as:

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

where the gating $\mathbf{g} \in \mathbb{R}^{d\times m}$ is decided by the input and output of multi-layer SGNNs:

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

where $[\cdot]$ is the concatenation operation, $\mathbf{W}_g \in \mathbb{R}^{d\times 2d}$ is a trainable parameter that transforms the concatenated vector from $\mathbb{R}^{2d}$ to $\mathbb{R}^d$, and σ is the *sigmoid* function. After the highway networks, we can obtain the final representation of satellite nodes as $\mathbf{h}^f$, and the corresponding star node as $\mathbf{x}_s^L$ (denoted as $\mathbf{x}_s$ for brevity).

We detail the proposed procedure of SGNNs in Algorithm 1. We first initialize the representation of satellite nodes and star node as step 1 and 2. Then we update the representations through L-layer SGNNs using steps 3 to 8, where we update the satellite nodes in steps 4 to 6 and the star node in step 7 at each SGNN layer. After that, we utilize the highway networks to combine the item embeddings before and after the multi-layer SGNNs in step 9. Finally, we return the new representations of the satellite nodes and the star node.

Algorithm 1 The procedure of Star Graph Neural Networks

Input: $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$: Initial embeddings of the unique items in the session

Output: $\mathbf{h}^f$ and $\mathbf{x}_s$: New representation of the satellite nodes and star node

1: $\mathbf{h}^0 = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$
2: $\mathbf{x}_s^0 = average(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$
3: **for** l in range(L) **do**
4: $\quad \hat{\mathbf{h}}^l = GGNN(\mathbf{h}^{l-1}, \mathbf{A}^I, \mathbf{A}^O)$ based on (3) and (4)
5: $\quad \alpha^l = Att(\hat{\mathbf{h}}^l, \mathbf{x}_s^{l-1})$ based on (5)
6: $\quad \mathbf{h}^l = Gating(\hat{\mathbf{h}}^l, \mathbf{x}_s^{l-1}, \alpha^l)$ based on (6)
7: $\quad \mathbf{x}_s^l = SelfAtt(\mathbf{x}_s^{l-1}, \mathbf{h}^l)$ based on (7) and (8)
8: **end for**
9: $\mathbf{h}^f = HighwayNetworks(\mathbf{h}^0, \mathbf{h}^L)$ based on (10) and (11)
10: **return** new satellite nodes $\mathbf{h}^f$ and new star node $\mathbf{x}_s$

3.3 Session representation and prediction

After generating the item representations based on the star graph neural networks, we obtain the sequential item embeddings from the corresponding satellite nodes $\mathbf{h}^f \in \mathbb{R}^{d\times m}$ as $\mathbf{u} \in \mathbb{R}^{d\times n}$. In order to incorporate the sequential information into SGNN-HN, we add learnable position embeddings $\mathbf{p} \in \mathbb{R}^{d\times n}$ to the item representations, i.e., $\mathbf{u}^p = \mathbf{u} + \mathbf{p}$.

Then we consider both the global preference and recent interest within the session to generate a final session representation as user's preference. As previous work has shown that the last item in a session can represent a user's recent interest [16, 34], we directly take the representation of the last item as the user's recent interest, i.e., $\mathbf{z}_r = \mathbf{u}_n^p$. As for the global preference, considering that items in a session have different degrees of importance, we combine the items according to their corresponding priority:

$$\mathbf{z}_g = \sum_{i=1}^{n} \gamma_i \mathbf{u}_i^p. \quad (12)$$

where the priority γ_i is decided by both the star node $\mathbf{x}_s$ and the current interest $\mathbf{z}_r$ simultaneously. Specifically, the weights of items are decided by a soft attention mechanism:

$$\gamma_i = \mathbf{W}_0{}^\top \sigma(\mathbf{W}_1\mathbf{u}_i^p + \mathbf{W}_2\mathbf{x}_s + \mathbf{W}_3\mathbf{z}_r + \mathbf{b}), \quad (13)$$

where $\mathbf{W}_0 \in \mathbb{R}^d$, $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{d\times d}$ are trainable parameters to control the weights, and $\mathbf{b} \in \mathbb{R}^d$ is the bias. Then we combine user's global preference and her current interest as the final session representation by concatenating them:

$$\mathbf{z}_h = \mathbf{W}_4[\mathbf{z}_g; \mathbf{z}_r], \quad (14)$$

where $[\cdot]$ is the concatenation operation and $\mathbf{W}_4 \in \mathbb{R}^{d\times 2d}$ transforms the concatenated representation from $\mathbb{R}^{2d}$ into $\mathbb{R}^d$.

After obtaining user's preference, we use it to make recommendations by calculating the probabilities on all candidate items in V. First, to solve the long-tail problem in recommendation [1, 7], we apply a layer normalization on the session representation $\mathbf{z}_h$ and the embedding of each candidate item $\mathbf{v}_i$, i.e., $\tilde{\mathbf{z}}_h = LayerNorm(\mathbf{z}_h)$ and $\tilde{\mathbf{v}}_i = LayerNorm(\mathbf{v}_i)$, respectively. After normalization, we calculate the scores $\tilde{\mathbf{y}}$ of each item in the item set V by multiplying the session representation with all item embeddings as follows:

$$\tilde{\mathbf{y}}_i = \tilde{\mathbf{z}}_h^{\top} \tilde{\mathbf{v}}_i. \tag{15}$$

Finally, we apply a softmax layer to normalize the preference scores of candidate items. It is worth noting that we will face a convergence problem after normalization as the softmax loss will be trapped at a very high value on the training set [29]. To address this problem, a scale coefficient τ is applied inside the softmax to obtain the final scores:

$$\hat{\mathbf{y}} = \frac{\exp(\tau \tilde{\mathbf{y}}_i)}{\sum_i \exp(\tau \tilde{\mathbf{y}}_i)}, \quad \forall i = 1, 2, \ldots, |V|, \tag{16}$$

where $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_{|V|})$. Finally, the items with the highest scores in $\hat{\mathbf{y}}$ will be recommended to the user.

To train our model, we employ cross-entropy as the optimization objective to learn the parameters as:

$$L(\hat{\mathbf{y}}) = -\sum_{i=1}^{|V|} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i), \tag{17}$$

where $\mathbf{y}_i \in \mathbf{y}$ reflects the appearance of an item in the one-hot encoding vector of the ground truth, i.e., $\mathbf{y}_i = 1$ if the i-th item is the target item of the given session; otherwise, $\mathbf{y}_i = 0$. In addition, we apply the Back-Propagation Through Time (BPTT) algorithm [21] to train the SGNN-HN model.

4 EXPERIMENTS

4.1 Research questions

To examine the performance of SGNN-HN, we address four 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) Can the highway networks alleviate the overfitting problem of graph neural networks?

(RQ4) How does SGNN-HN perform on sessions with different lengths compared to 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 [14, 16, 34], 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 are remained in the *Diginetica* dataset.

Similar to [14, 16, 34], we use a sequence splitting preprocess to augment the training samples. Specifically, for session $S = \{v_1, v_2, \ldots, v_n\}$, we generate the sequences and corresponding labels as $([v_1], v_2), ([v_1, v_2], v_3), \ldots, ([v_1, \ldots, v_{n-1}], v_n)$ for training and test. Moreover, as *Yoochoose* is too large, following [14, 16, 34], 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 1.

Table 1: Statistics of the datasets used in our experiments.

Statistics	*Yoochoose 1/64*	*Yoochoose 1/4*	*Diginetica*
# 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

4.3 Model summary

The models discussed in this paper are the following: (1) Two traditional methods, i.e., S-POP [2] and FPMC [20]; (2) Three RNN-based methods, i.e., GRU4REC [10], NARM [14] and CSRM [30]; (3) Two attention-based methods, i.e., STAMP [16] and SR-IEM [18]; and (4) Two GNN-based methods, i.e., SR-GNN [34] and NISER+ [7].

- **S-POP** recommends the most popular items for the current session.
- **FPMC** is a state-of-the-art hybrid method for sequential recommendation based on Markov Chains. We omit the user representation since it is unavailable in session-based recommendation.
- **GRU4REC** applies GRUs to model the sequential information in session-based recommendation and adopts a session-parallel mini-batch training process.
- **NARM** employs GRUs to model the sequential behavior and utilizes an attention mechanism to capture user's main purpose.
- **CSRM** extends NARM by introducing neighbor sessions as auxiliary information for modeling the current session with a parallel memory module.
- **STAMP** utilizes an attention mechanism to obtain the general preference and resorts to the last item as recent interest in current session to make prediction.
- **SR-IEM** employs a modified self-attention mechanism to estimate the item importance and makes recommendations based on the global preference and current interest.
- **SR-GNN** utilizes the gated graph neural networks to obtain item embeddings and make recommendations using an attention mechanism to generate the session representation.
- **NISER+** introduces L2 normalization to solve the long-tail problem and applies dropout to alleviate the overfitting problem of SR-GNN.

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

[1]http://2015.recsyschallenge.com/challenge.html
[2]http://cikm2016.cs.iupui.edu/cikm-cup

proportion of 10%. Following [7, 18, 34], 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. All parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1.

4.5 Evaluation metrics

Following previous works [14, 34], we adopt **P@K** and **MRR@K** to evaluate the recommendation performance.

P@K (Precision): The P@K score measures whether the target item is included in the top-K recommendations of the list of recommended items.

$$\text{P@K} = \frac{n_{hit}}{N}, \tag{18}$$

where N is the number of test sequences in the dataset and n_{hit} is the number of cases that the target item is in the top-K items of the ranked list.

MRR@K (Mean Reciprocal Rank): The MRR@K score considers the position of the target item in the list of recommended items. It is set to 0 if the target item is not in the top-K of ranking list, and otherwise is calculated as follows:

$$\text{MRR@K} = \frac{1}{N} \sum_{v_{target} \in S_{test}} \frac{1}{Rank(v_{target})}, \tag{19}$$

where v_{target} is the target item and $Rank(v_{target})$ is the position of the target item in the list of recommended items. Compared to P@K, MRR@K is a normalized hit which takes the position of the target item into consideration.

5 RESULTS AND DISCUSSION

5.1 Overall performance

We present the results of the proposed session-based recommendation model SGNN-HN as well as the baselines in Table 2. 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 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.

Table 2: Model performance. The results of the best performing baseline and the best performer 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).

Method	*Yoochoose 1/64*		*Yoochoose 1/4*		*Diginetica*	
	P@20	MRR@20	P@20	MRR@20	P@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
NISER+	<u>71.27</u>	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**▲

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

Next, we move to the proposed model SGNN-HN. From Table 2, we observe that SGNN-HN can achieve the best performance in terms of P@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 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 star graph neural networks, leading to a better representation of items.

Moreover, we find that the improvement of SGNN-HN over the best baseline in terms of P@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 P@20 on both *Yoochoose 1/64* and *Yoochoose1/4*. In contrast, a higher improvement in terms of P@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.

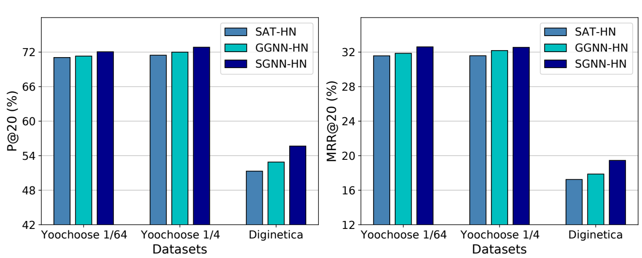
Our results indicate that our proposed SGNN-HN model can help put the target item at an earlier position when there are relatively few candidate items and is more effective on hitting the target item in the recommendation list for cases with relatively many candidate items.

5.2 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) [15] and a self-attention mechanism (SAT) [27]. 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., $\mathbf{x}_s = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, 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 combination of GGNN and SAT.

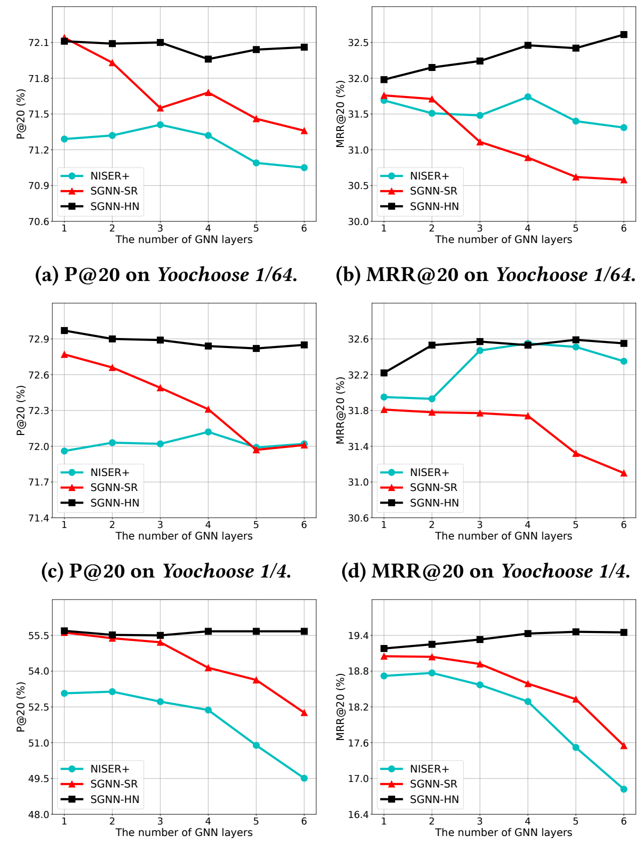
To prove the effectiveness of SGNN, we substitute the SGNN in our proposal with two alternatives for propagating information between items and evaluate the performance in terms of P@20 and MRR@20 on three datasets. The variants are denoted as: (1) GGNN-HN, which replaces the SGNN with a simple GGNN; (2) SAT-HN, which replaces SGNN with GAT. See Figure 4.

In Figure 4, we see that SGNN-HN can achieve the best performance in terms of both P@20 and MRR@20 on three datasets. Moreover, as for the variants, GGNN-HN performs better than GAT-HN for all cases on the 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 a better performance is achieved than GAT-HN. Moreover, comparing GGNN-HN to SGNN-HN, we see that GGNN-HN can only propagate information from adjacent items, missing much information from unconnected items and leading to a worse performance than SGNN-HN.



(a) P@20.　(b) MRR@20.

Figure 4: Performance comparison to assess the utility of star graph neural networks.



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

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

(e) P@20 on *Diginetica*.　(f) MRR@20 on *Diginetica*.

Figure 5: Model performance of different number of GNN layers.

5.3 Utility of highway networks

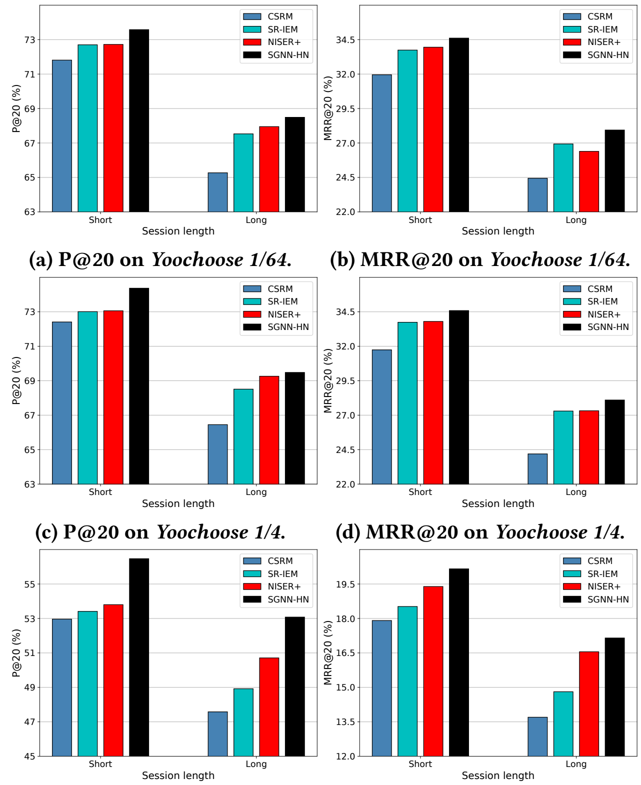
For RQ3, in order to investigate the impact of the number of GNN layers on the proposed SGNN-HN model and to validate the effectiveness of the highway networks, we compare SGNN-HN to its variant SGNN-SR, which removes the highway networks from SGNN-HN. In addition, the comparison involves the best performer 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 P@20 and MRR@20 of NISER+, SGNN-SR and SGNN-HN on three datasets. The results are shown in Figure 5.

As shown in Figure 5, SGNN-HN achieves the best performance in terms of both P@20 and MRR@20 for almost all cases on three datasets. For P@20, from Figure 5a, 5c and 5e, 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. 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 P@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 P@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 tread 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. 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 P@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 P@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.



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

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

(e) P@20 on *Diginetica*. **(f) MRR@20 on *Diginetica*.**

Figure 6: Model performance on short and long sessions.

5.4 Impact of the session length

To answer RQ4, we evaluate the performance of SGNN-HN and the state-of-the-art baselines, i.e., CSRM, SR-IEM and NISER+ on three datasets. Following [16, 34], we separate the sessions according to their length, i.e., the number of clicked items. Specifically, the sessions containing less than or equal to 5 clicked items are regarded as "Short", while others are deemed as "Long". We set the threshold to 5 as in [16, 34] since it is the closest integer to the average length of sessions in three datasets. The ratios of "Short" and "Long" sessions for test for the *Yoochoose 1/64* and *Yoochoose 1/4* datasets are 70.10% and 29.90%, respectively, and 76.40% and 23.60% for *Diginetica*. The performance in terms of P@20 and MRR@20 of SGNN-HN and the baselines are shown in Figure 6.

From Figure 6, we see that SGNN-HN can achieve the best performance in terms of P@20 and MRR@20 for all cases on the three datasets. Moreover, as the session length increases, the performance of all models in terms of both P@20 and MRR@20 on the three datasets consistently decreases, which may be due to the fact that longer sessions are more likely to contain unrelated items, making it harder to correctly identify the user preference. For P@20, as shown in Figure 6a, 6c and 6e, we see that among the baselines, CSRM performs worst for both "Short" and "Long" sessions on the three datasets, validating that the transition relationship in the session is more complicated than a simple sequential signal. By comparing SR-IEM and NISER+, we can observe that their performance are similar for the "Short" sessions, while NISER+ shows an obvious better performance for the "Long" sessions. This indicates that by modeling the complex transition relationship between items, GNNs can accurately obtain user's preference to hit the target item when user-item interactions are relatively many.

As for MRR@20, NISER+ does not show better performance than SR-IEM on both "Short" and "Long" sessions on *Yoochoose 1/64*, and the same phenomenon is observed on *Yoochoose 1/4*. However, SGNN-HN clearly outperforms SR-IEM for all cases on the three datasets. We attribute the difference in performance between NISER+ and SGNN-HN to: (1) the SGNN makes the information from long-range items available in information propagating; and (2) the highway networks in SGNN-HN allow for the complex transition relationship between items to be investigated more accurately through multi-layer SGNNs, which can boost the ranking of the target item in the list of recommended items.

In addition, in terms of P@20, the improvements of SGNN-HN against the best baseline NISER+ for "Short" and "Long" sessions are 1.18% and 0.79% on *Yoochoose 1/64*, respectively; 4.96% and 4.67% improvements of SGNN-HN over NISER+ on *Diginetica* are observed. This indicates that SGNN-HN is more effective at hitting the target item for relatively short sessions. Moreover, in terms of MRR@20, the improvements of SGNN-HN against the best baselines NISER+ and SR-IEM for "Short" and "Long" sessions are 1.23% and 2.97% on *Yoochoose 1/64*, respectively. Here higher improvement is observed on "Long" sessions. Differently on *Diginetica*, 4.62% and 3.76% improvements of SGNN-HN against NISER+ are observed on "Short" and "Long" sessions, respectively. The difference in terms of

MRR@20 between the two datasets may be due to the fact that the average session length are different; it is bigger in *Yoochoose 1/64* than in *Diginetica*. As there are proportionally more long sessions in *Yoochoose 1/64*, this explains the bigger improvement on "Long" than "Short" sessions for *Yoochoose 1/64*.

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 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 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 the user's general preference, which is then concatenated with her recent interest expressed by her last clicked item in the session for making item recommendation. Experiments conducted on two public benchmark datasets show that SGNN-HN can significantly improve the recommendation performance in terms of P@20 and MRR@20.

As to future work, on the one hand, we would like to incorporate neighbor sessions to enrich the transition relationship in the current session like in [11, 30]. On the other hand, we are interested in applying star graph neural networks to other tasks like conversational recommendation [13, 26] and dialogue systems [38] to investigate its scalability.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China under No. 61702526, the Defense Industrial Technology Development Program under No. JCKY2017204B064, and 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.

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation. In *RecSys'17*. 42–46.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. Knowl. Data Eng.* 17, 6 (2005), 734–749.

[3] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. A Dynamic Co-attention Network for Session-based Recommendation. In *CIKM'19*. 1461–1470.

[4] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. Joint Neural Collaborative Filtering for Recommender Systems. *ACM Trans. Inf. Syst.* 37, 4 (2019), 39:1–39:30.

[5] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam M. Shroff. 2019. Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN. In *SIGIR'19*. 1069–1072.

[6] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-Transformer. In *NAACL'19*. 1315–1325.

[7] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam M. Shroff. 2019. NISER: Normalized Item and Session Representations with Graph Neural Networks. *arXiv preprint arXiv:1909.04276* (2019).

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW'17*. 173–182.

[9] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 1 (2004), 5–53.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR'16*.

[11] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *RecSys'17*. 306–310.

[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR'17*.

[13] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In *WSDM'20*. 304–312.

[14] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM'17*. 1419–1428.

[15] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR'16*.

[16] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD'18*. 1831–1839.

[17] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An Intent-guided Collaborative Machine for Session-based Recommendation. In *SIGIR'20*. 1833–1836.

[18] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking Item Importance in Session-based Recommendation. In *SIGIR'20*. 1837–1840.

[19] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks. In *CIKM'19*. 579–588.

[20] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW'10*. 811–820.

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536.

[22] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW'01*. 285–295.

[23] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. In *WSDM'19*. 555–563.

[24] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway Networks. *arXiv preprint arXiv:1505.00387* (2015).

[25] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *NeurIPS'15*. 2440–2448.

[26] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. In *SIGIR'18*. 235–244.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS'17*. 5998–6008.

[28] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR'18*.

[29] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. 2017. NormFace: L_2 Hypersphere Embedding for Face Verification. In *MM'17*. 1041–1049.

[30] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules. In *SIGIR'19*. 345–354.

[31] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for NextBasket Recommendation. In *SIGIR'15*. 403–412.

[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR'19*. 165–174.

[33] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In *ICLR'15*.

[34] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *AAAI'19*. 346–353.

[35] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI'19*. 3940–3946.

[36] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential Recommender System based on Hierarchical Attention Networks. In *IJCAI'18*. 3926–3932.

[37] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *SIGIR'16*. 729–732.

[38] Hainan Zhang, Yanyan Lan, Liang Pang, Jiafeng Guo, and Xueqi Cheng. 2019. ReCoSa: Detecting the Relevant Contexts with Self-Attention for Multi-turn Dialogue Generation. In *ACL'19*. 3721–3730.

[39] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (2019), 5:1–5:38.