Dynamic Graph Neural Networks for Sequential Recommendation

Mengqi Zhang, Shu Wu, *Senior Member, IEEE*, Xueli Yu, Qiang Liu, *Member, IEEE*, and Liang Wang, *Fellow, IEEE*

Abstract—Modeling user preference from his historical sequences is one of the core problems of sequential recommendation. Existing methods in this field are widely distributed from conventional methods to deep learning methods. However, most of them only model users' interests within their own sequences and ignore the dynamic collaborative signals among different user sequences, making it insufficient to explore users' preferences. We take inspiration from dynamic graph neural networks to cope with this challenge, modeling the user sequence and dynamic collaborative signals into one framework. We propose a new method named *Dynamic Graph Neural Network for Sequential Recommendation* (DGSR), which connects different user sequences through a dynamic graph structure, exploring the interactive behavior of users and items with time and order information. Furthermore, we design a Dynamic Graph Recommendation Network to extract user's preferences from the dynamic graph. Consequently, the next-item prediction task in sequential recommendation is converted into a link prediction between the user node and the item node in a dynamic graph. Extensive experiments on four public benchmarks show that DGSR outperforms several state-of-the-art methods. Further studies demonstrate the rationality and effectiveness of modeling user sequences through a dynamic graph.

Index Terms—Sequential recommendation, dynamic collaborative signals, dynamic graph neural networks

1 Introduction

With dramatic growth of the amount of information on the Internet, recommender systems have been applied to help users alleviate the problem of information overload in online services, such as e-commerce, search engines, and social media. Recently, several collaborative filtering methods have been proposed, which focus on static user-item interactions [1], [2], [3] but ignore the rich historical sequential information of users. However, user preferences change dynamically over time, varying with the historical interacted items. Therefore, sequential recommendation has attracted lots of attention, which seeks to utilize the sequential information from each user's interaction history to make accurate predictions.

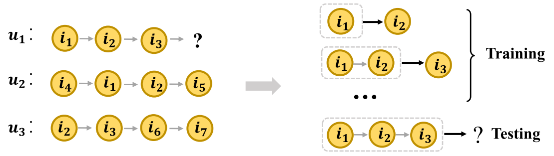
A series of methods have been proposed in the field of sequential recommendation. For example, the Markov-chain model [4] makes recommendation based on k previous interactions. Several RNN-based models [5], [6], [7] utilize Long Short-term Memory (LSTM) [8] or Gated Recurrent Unit (GRU) [9] networks to capture sequential dependencies in user sequences. Furthermore, Convolutional Neural Networks (CNN) and Attention Networks are also effective in modeling user sequences. For example, Caser [10] employs convolutional filters to incorporate the order of user interaction. SASRec [11] and STAMP [12] apply the attention mechanism to model the relationship between items to capture user intent. Recently, Graph Neural Networks (GNNs) [13], [14] have gained increasing attention. Inspired by the success of GNN in a wide variety of tasks, some GNN-based sequence models [15], [16], [17] are proposed, which use improved GNN to investigate the complex item transition relationships in each sequence.

Although these methods have achieved compelling results, we argue that these methods lack explicit modeling of the *dynamic collaborative signals* among different user sequences, which is mainly manifested in two aspects:

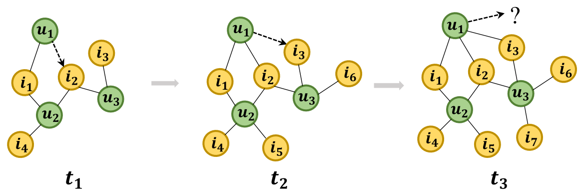
1) These models do not explicitly leverage the collaborative information among different user sequences, in other words, most of them focus on encoding each user's own sequence, while ignoring the high-order connectivity between different user sequences, as can be seen in Fig. 1a, in which the encoding of user sequence during training and testing are all within a single sequence. However, as shown in Fig. 1b, at time t_3, u_1 interacts with i_1, i_2 and i_3 directly, and also has high-order connections with u_2 and u_3 as well as their interactive items. Obviously, the interaction information of u_2 and u_3 can assist in the prediction of u_1's sequence. This information is overlooked by most of the existing models.
2) These models ignore the dynamic influence of the high-order collaboration information at different

- *Mengqi Zhang, Shu Wu, Qiang Liu, and Liang Wang are with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 101408, China, and also with Center for Research on Intelligent Perception and Computing (CRIPAC), Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. E-mail: mengqi.zhang@cripac.ia.ac.cn, {shu.wu, qiang.liu, wangliang}@nlpr.ia.ac.cn.*
- *Xueli Yu is with the Beijing Institute for General Artificial Intelligence (BIGAI), Beijing 100081, China. E-mail: yuxueli@bigai.ai.*

Manuscript received 21 June 2021; revised 20 December 2021; accepted 29 January 2022. Date of publication 0 0000; date of current version 0 0000.
This work was supported by the National Natural Science Foundation of China under Grants 62141608 and U19B2038.
(Corresponding author: Shu Wu.)
Recommended for acceptance by T. Weninger.
Digital Object Identifier no. 10.1109/TKDE.2022.3151618



(a) The left figure presents differnt sequences of u_1, u_2 and u_3. Our goal is to predict the next interaction of u_1. The figure on the right illustrates the training and testing paradigm of most sequential models.



(b) Interaction of user-item graph composed of u_1, u_2 and u_3 at different times. Each edge represents the interaction between user and item, and has time attribute. The node u_1 is the target user to predict. The solid line indicates the interaction that has occurred at the current time. The dotted arrow represents the next interactions of u_1. The timestamp sequence of u_1 is (t_1, t_2, t_3).

Fig. 1. Illustration of user-item sequential interaction. Figure (a) illustrates the interaction sequences of u_1, u_2 and u_3. Figure (b) is the user-item interaction graph composed of u_1, u_2 and u_3 at different times, which can be seen as a refined representation of Figure (a).

times. From Fig. 1b, we can see that the graph formed by u_1's sequence and its high-order associated users and items vary with t_1, t_2 and t_3 time. In this case, the change of u_1's interest is influenced not only by the change of first-order interaction items i_1, i_2 and i_3, but also by the varied high-order connected users and items. Similarly, the semantic information of items may also shift with the change of first- and higher-order relevance.

Consequently, the above two aspects result in the difficulty of accurately capturing user preference in sequential recommendation. To deal with these challenges, two important problems need to be solved:

1) *How to dynamically represent user-item interactions with a graph.* The order of interaction between users and items is vital for sequential recommendation. Most existing methods [3] represent user-item interactions as a static bipartite graph and fail to record the interaction order of the user-item pair. So, we need to consider incorporating sequence information or interaction order into the graph flexibly and efficiently.
2) *How to explicitly encode the dynamic collaborative signal for each user sequence.* For each user sequence, its dynamic associated items and users form a graph structure, which includes more time/order information than conventional static graph. It is not trivial to encode the preference of each user from this dynamic graph.

To this end, inspired by dynamic graph representation learning [18], we propose a novel method named *Dynamic Graph Neural Network for Sequential Recommendation* (DGSR), which explores interactive behaviors between users and items through a dynamic graph. The framework of DGSR is as follows: *firstly*, we convert all user sequences into a dynamic graph annotated with time and order information on edges (Section 4.1). Consequently, the user sequences having common items are associated with each other via *user* → *item* and *item* → *user* connections. *Second*, we devise a sub-graph sampling strategy (Section 4.2) to dynamically extract sub-graphs containing user's sequence and associated sequences. *Third*, to encode user's preference from the sub-graph, we design a Dynamic Graph Recommendation Network (DGRN) (Section 4.3), in which a dynamic attention module is constructed to capture the long-term preference of users and long-term character of items, and a recurrent neural module or attention module is further utilized to learn short-term preference and character of users and items, respectively. By stacking multiple DGRN layers, the rich dynamic high-order connectivity information of each user and each item node can be better utilized. *Finally*, our model converts the next-item prediction task into a link prediction task for user nodes (Section 4.4). Extensive experiments conducted on four public benchmark datasets verify the effectiveness of our DGSR method.

To summarize, our work makes the following main contributions:

- We highlight the critical importance of explicitly modeling dynamic collaborative signals among user sequences in the sequential recommendation scenario.
- We propose DGSR, a new sequential recommendation framework based on dynamic graph neural networks.
- We conduct empirical studies on four real-world datasets. Extensive experiments demonstrate the effectiveness of DGSR.

2 Related Work

2.1 Sequential Recommendation

Sequential recommendation is to predict the next item based on users' historical interaction sequences. Compared with the static recommender system, it usually generates user's representation based on its sequential interactions for prediction. Pioneering works, such as Markov chain-based methods [4], [19] predict the next item based on k-order interaction. Translation-based approaches [20] model third-order interactions with a TransRec component.

With the development of deep learning, many related works have been proposed for sequential recommendation task. GRU4Rec [5] is the first one to use Recurrent Neural Network (RNN) to session-based recommendation task. Due to the excellent performance of RNN, it has been widely used for sequential recommendation task [6], [7], [21]. What's more, Convolution Neural Network (CNN) is also used in sequential recommendation to investigate the different patterns. The CNN-based model Caser [10] applies convolution filters to incorporate different order of users' interactions. Furthermore, the attention network is also a powerful tool applied in the sequential recommendation. NARM [22] employs the attention mechanism on RNN to capture users' main purposes. STAMP [12] uses a novel attention memory network to efficiently capture both the users' general interests and current interests. SASRec [11] applies self-attention mechanisms to sequential recommendation problems to explicitly model the relationship between items. More recently, based on SASRec, TiSASRec [23] is proposed to model the absolute positions of items as well as the time intervals between them in a sequence.

In the last few years, graph neural networks (GNNs) have achieved state-of-the-art performance in processing graph structure data. There are also some studies [15], [16], [17], [24], [25] applying GNNs to sequential recommendation. SR-GNN [15] firstly utilizes the Gated GNNs to capture the complex item transition relationship in session scenario. Based on this work, A-PGNN [16] combining personalized GNN and attention mechanism is proposed for session-aware scenarios. MA-GNN [25] employs a memory augmented graph neural network to capture both the long- and short-term user interests.

Although these GNN-based models have shown promising direction for sequential recommendation, they only focus on modeling user preferences on intra-sequence and ignore the item relationship across sequences. To this end, some models [26], [27], [28], [29], [30], [31] that utilize the cross sequence information are proposed. For example, HyperRec [26] adopts hypergraph to model the high-order correlations connections between items within or across sequences. CSRM [27] considers neighborhood sessions by calculating that of similarity between with current session. DGRec [28] explicitly associate different user sequences through social relationships. SERec [31] adopts a heterogeneous graph neural network to learn user and item representations that extract the knowledge from social networks, but not all data have social relationship attributes. Furthermore, to effectively learn user and item embeddings, THIGE [29] utilizes the temporal heterogeneous graph for next-item recommendation. GCE-GNN [30] leverages a new global graph to learn the global-level item embedding from all sessions, but does not accurately consider sequential information in the global graph. In this paper, our model processing sequential recommendation task is distinct from the above-mentioned methods. The detailed comparative analysis with these models will be elaborated in Section 4.5.

2.2 Dynamic Graph Neural Networks

Nowadays, graph neural networks have been employed to address different problems, such as node classification [13], [14], graph embedding [32], [33], graph classification [34], recommendation [15], [16], [17], [24], [25], [35] and so on.

However, in many applications, the graph data change over time, such as academic network, social network, and recommender system. As a result, a surge of works considers modeling dynamic graphs. DANE [36] leverages matrix perturbation theory to capture the changes of adjacency and attribute matrix in an online manner. DynamicTriad [37] imposes the triadic closure process to preserve both structural information and evolution patterns of dynamic network. DynGEM [38] uses a dynamically expanding deep Auto-Encoder to capture highly nonlinear first-order and second-order proximities of the graph nodes. CTDNE [39] designs a time-dependent random walk sampling method for learning dynamic network embeddings from continuous-time dynamic networks. HTNE [40] integrates the Hawkes process into network embeddings to capture the influence of historical neighbors on the current neighbors for temporal network embedding. Dyrep [41] utilizes a deep temporal point process model to encode structural-temporal information over graph into low dimensional representations. JODIE [42] utilizes two types of RNN to model the evolution of different node representations. MTNE [43] not only integrates the Hawkess process to stimulate the triad evolution process, but also combines attention to distinguish the importance of different motifs. To inductively infer embeddings for both new and observed nodes as the graph evolves, Xu *et al.* [44] propose the temporal graph attention mechanism based on the classical Bochner's theorem. There are also some works [45] crop the dynamic graph into a sequence of graph snapshots.

Although some of the above-mentioned dynamic methods are tested on e-commerce data sets, they are not adapt to sequential recommendation scenarios. As far as we know, there is no study to illustrate the sequential recommendation problem from the perspective of dynamic graphs.

3 Preliminaries

In this section, we describe problems about sequential recommendation and dynamic graph.

3.1 Sequential Recommendation

In the setting of sequential recommendation, let $\mathcal{U}$ and $\mathcal{I}$ represent the set of users and items, respectively. For each user $u \in \mathcal{U}$, its action sequence is denoted as $S^u = (i_1, i_2, \ldots, i_k)$, where $i \in \mathcal{I}$. $T^u = (t_1, t_2, \ldots, t_k)$ is the corresponding timestamp sequence of S^u. The set of all S^u is denoted as $\mathcal{S}$. The object of sequential recommendation is to predict the next item of S^u employing sequence information before time t_k and t_k. In general, sequential recommendation task limits the maximum length of S^u to n. When k is greater than n, taking the most recent n items $(i_{k-n}, i_{k-n+1}, \ldots, i_k)$ to make predictions.

Each user and item can be converted into low-dimensional embedding vector $\mathbf{e}_u$, $\mathbf{e}_i \in \mathbb{R}^d$, respectively, where $u \in \mathcal{U}$ and $i \in \mathcal{I}$, d is the dimension of embedding space. We use the $\mathbf{E}_U \in \mathbb{R}^{|\mathcal{U}| \times d}$ and $\mathbf{E}_I \in \mathbb{R}^{|\mathcal{I}| \times d}$ to represent the user embedding and item embedding matrix, respectively.

3.2 Dynamic Graph

A dynamic network can be defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the node set and $e \in \mathcal{E}$ represents the interaction between v_i and v_j at time $t \in \mathcal{T}$, so edge e_{ij} between v_i and v_j is generally represented by triplet (v_i, v_j, t). In some cases, t can also indicate the order of interactions between two nodes. By recording the time or order of each edge, a dynamic graph can capture the evolution of the relationship between nodes. Dynamic graph embedding aims to learn a mapping function $f : \mathcal{V} \rightarrow \mathbb{R}^d$, where d is the number of embedding dimensions.

4 Methodology

We now present the proposed DGSR model, the framework of which is illustrated in Fig. 2. There are four components in the architecture: 1) *Dynamic Graph Construction* is to convert all sequences of users to a dynamic graph; 2) *Sub-graph*

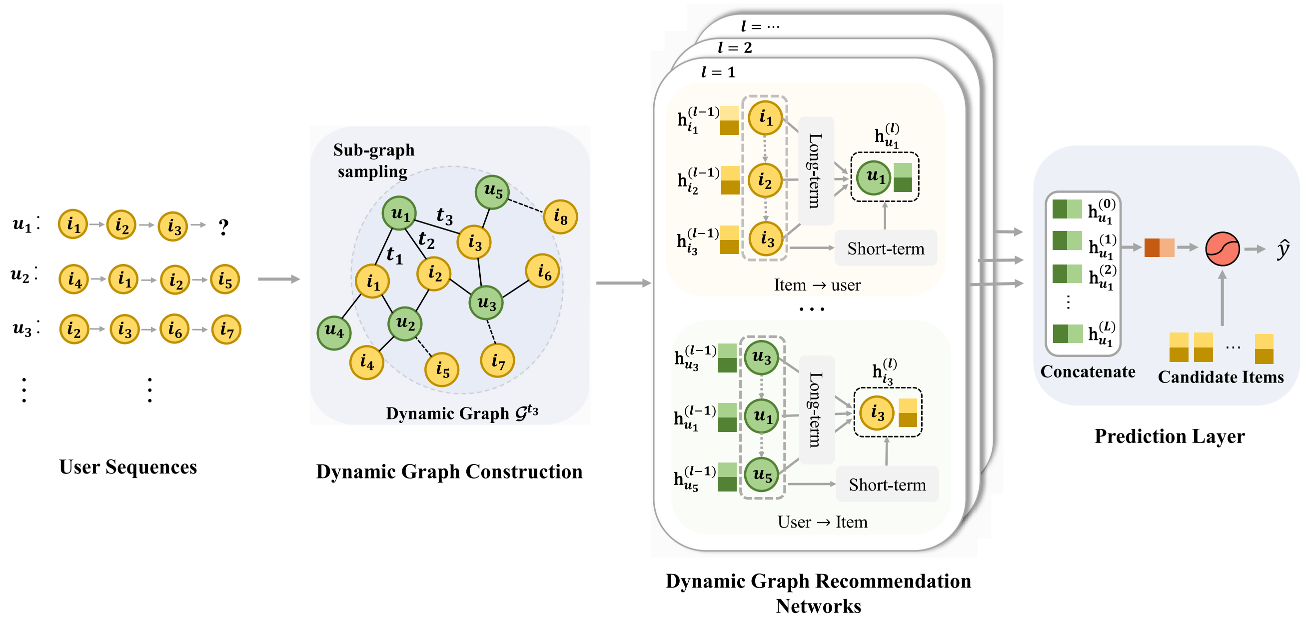


Fig. 2. Overview of DGSR framework. Take predicting the next interaction of u_1's sequence (i_1, i_2, i_3) as an example. The corresponding timestamp sequence is (t_1, t_2, t_3). We first convert u_1' sequence and its related sequences into dynamic graph $\mathcal{G}^{t_3}$, each edge represents the interaction between user and item, and has time attribute. The edges represented by the dotted line are interactions that occurred after t_3, which is not included in $\mathcal{G}^{t_3}$ (Section 4.1). Then we sample a m-order sub-graph $\mathcal{G}^m_{u_1}(t_3)$ from $\mathcal{G}^{t_3}$ (Section 4.2). Following this, the well-designed Dynamic Graph Recommendation Network propagates and aggregates the information among different user sequences (Section 4.3). Finally, we concatenate user node embedding of each layer for final predication (Section 4.4).

Sampling is to extract sub-graphs which contain user's sequence and its related sequences; 3) *Dynamic Graph Recommendation Network* (DGRN) contains message propagation mechanism and node update part to encode each user preference from the sub-graph; and 4) *Predication Layer* aggregates the user's refined embeddings learned from DGRN, and predicts which item node will be most likely linked with the user node next. Algorithm 2 provides the pseudo-code of the overall framework.

4.1 Dynamic Graph Construction

In this section, we describe how to convert all user sequences into a dynamic graph. When the user u acts on the item i at time t, an edge e is established between u and i, and e can be represented by the quintuple (u, i, t, o_u^i, o_i^u). t describes the timestamp when the interaction occurred. Besides, distinguished with the definition of the conventional dynamic graph, o_u^i is the order of $u - i$ interaction, that is, the position of item i in all items that the u has interacted with. o_i^u refers to the order of u in all user nodes that have interacted with item i. For example, u_1's sequence and timestamp sequence are (i_1, i_2, i_3) and (t_1, t_2, t_3), respectively. u_2's sequence and timestamp sequence are (i_2, i_3, i_1) and (t_4, t_5, t_6), where $t_1 < t_2 < t_3 < t_4 < t_5 < t_6$. The edges between users and its interaction items can be written as $(u_1, i_1, t_1, 1, 1)$, $(u_1, i_2, t_2, 2, 1)$, $(u_1, i_3, t_3, 3, 1)$, $(u_2, i_2, t_4, 1, 2)$, $(u_2, i_1, t_5, 2, 2)$, and $(u_2, i_3, t_6, 3, 2)$.

Since a large number of user sequences interacted with the same items, for example, as shown in the Fig. 2, u_1 and u_2 have common items i_1 and i_2, and u_1 and u_3 have common item i_3. Consequently, all the quintuples of dataset form a dynamic graph,

$$\mathcal{G} = \{(u, i, t, o_u^i, o_i^u) | u \in \mathcal{U}, i \in \mathcal{V}\},$$

which is a continuous-time graph according to the definitions in [46]. In addition to the interaction time between users and items, $\mathcal{G}$ also records the order information between them. So, our dynamic graph is more suitable for the sequential recommendation task than static graph and conventional dynamic graph. We define our dynamic graph at time t as $\mathcal{G}^t \in \mathcal{G}$, which is a dynamic graph composed of all users' interaction sequences at time t and before t. For a given user sequence $S^u = (i_1, i_2, \ldots, i_k)$, where the corresponding timestamp sequence is $T^u = (t_1, t_2, \ldots, t_k)$, the next item of the predicted sequence S^u is equivalent to predict the item linked to the node u in $\mathcal{G}^{t_k}$.

4.2 Sub-Graph Sampling

As the user sequence S^u extending, the number of neighbor sequences of it is growing. Similarly, the scale of the dynamic graph composed of all users is also gradually expanding. It will increase the computational cost and introduce too much noise into the target sequence S^u. For efficient training and recommendation, we propose a sampling strategy, details of which are shown in Algorithm 1.

Specifically, we first take user node u as the anchor node and select its most recent n first-order neighbors from graph $\mathcal{G}^{t_k}$, that is, the historical items that u has interacted with, written as $\mathcal{N}_u$, where n is the maximum length of user sequence (Line 6 and 7 in Algorithm 1). Next, for each item $i \in \mathcal{N}_u$, we use each of them as an anchor node to sample the set of users who have interacted with them, written as $\mathcal{N}_i$ (Line 13 and 14 in Algorithm 1). To improve sampling efficiency, we record user and item nodes that have been used to be anchor node to avoid repeated sampling (Line 8 and 15 in Algorithm 1), and set the maximum number of samples to n when sampling user node (Line 19 in Algorithm 1). Followed by analogy, we can obtain the multi-hop neighbors of node u, which could form u's m-order sub-graph $\mathcal{G}_u^m(t_k)$ of S^u (m is a hyper-parameter used to control the size of sub-graph).

After sampling, each sub-graph $\mathcal{G}_u^m(t_k)$ contains the nodes of the sequence S^u and its associated sequences. User and item nodes in these sequences are linked to each other through stacking the *user to item* and *item to user* relationships in $\mathcal{G}_u^m(t_k)$.

Algorithm 1. Sub-Graph Sampling Algorithm

Input: Sequence $S^u = (i_1, i_2, \ldots, i_k)$, timestamp sequence $T^u = (t_1, t_2, \ldots, t_k)$, dynamic graph $\mathcal{G}^{t_k}$, and the order of sub-graph m.
Output: The m-order sub-graph $\mathcal{G}_u^m(t_k)$.

// Initialization
$\mathcal{U}_m, \mathcal{U}_{temp} \leftarrow \{u\}, \mathcal{I}_m, \mathcal{I}_{temp} \leftarrow \emptyset$
// Node sampling
for $k \in [1, .., m]$ **do**
  if *k is an odd number* **then**
    for $u \in \mathcal{U}_{temp}$ **do**
      $\mathcal{I}_{temp} \leftarrow \mathcal{I}_{temp} \cup \mathcal{N}_u$
    $\mathcal{I}_{temp} \leftarrow \mathcal{I}_{temp} \setminus \mathcal{I}_m$
    if $\mathcal{I}_{temp} = \emptyset$ **then**
      Break
    $\mathcal{I}_m \leftarrow \mathcal{I}_m \cup \mathcal{I}_{temp}$
  else
    for $i \in \mathcal{I}_{temp}$ **do**
      $\mathcal{U}_{temp} \leftarrow \mathcal{U}_{temp} \cup \mathcal{N}_i$
    $\mathcal{U}_{temp} \leftarrow \mathcal{U}_{temp} \setminus \mathcal{U}_m$
    if $\mathcal{U}_{temp} = \emptyset$ **then**
      Break
    if $|\mathcal{U}_{temp}| > n$ **then**
      $\mathcal{U}_{temp} \leftarrow$ Sampling n nodes from $\mathcal{U}_{temp}$
    $\mathcal{U}_m \leftarrow \mathcal{U}_m \cup \mathcal{U}_{temp}$
// Sub-graph generation
$\mathcal{G}_u^m(t_k) = (\mathcal{U}_m, \mathcal{I}_m), \mathcal{U}_m, \mathcal{I}_m \in \mathcal{G}^{t_k}$

4.3 Dynamic Graph Recommendation Network

In this section, we design a Dynamic Graph Recommendation Network (DGRN) to encode the preference of each user from dynamic contextual information by acting on the sub-graph $\mathcal{G}_u^m(t_k)$. Similar to most GNNs, The DGRN component consists of message propagation and node updating components. For the sake of discussion, we illustrate the message propagation and node updating from $l-1$-th layer to l-th layer of DGRN.

The message propagation mechanism aims to learn the message propagation information from *user to item* and *item to user* in $\mathcal{G}_u^m(t_k)$, respectively. The challenge is how to encode the sequential information of neighbors from user and item perspectives, respectively. The static graph neural networks, such as GCN [13] and GAT [14], are powerful in various graph structure data. However, they fail to capture sequential information of neighbors suitably for each user and item. Some sequence model, such as RNN [5] and Transformer [47] net, are widely used to model user long- and short-term interest, but they can not deal with graph structured data directly. To this end, we combine the graph neural networks and sequential networks to design a dynamic propagation mechanism.

From Item to User. The set of neighbor nodes of the user node u is the items that u has purchased. To update the user node representations in each layer, we need to extract two types of information from the neighbors of each user node, which are *long-term preference* and *short-term preference* respectively. The *long-term preference* [48] of user reflects his or her inherent characteristics and general preference, which can be induced from the user's all historical items. The *short-term preference* of the user reflects his or her latest interest.

From User to Item. The set of neighbor nodes of the item node i is the users who purchased it, in which the users are arranged in chronological order. Similar to the user, the neighbors of item also reflect its two types of character. On the one hand, the *long-term character* can reflect the general characters of the item. For example, the *wealthy* people usually buy *high-end* cosmetics. On the other hand, *short-term character* reflects the newest property of item. For example, many non-sports enthusiasts may also buy jerseys or player posters during the World Cup. This part of consumers' behavior means the positioning of soccer equipment has changed from professionalism to universality in this period. However, most existing sequential recommendation methods fail to explicitly capture the impact of user nodes on the item node. To settle this problem, we also consider the message propagation from user to item.

4.3.1 Message Propagation Mechanism

In this subsection, we discuss the message propagation mechanism of DGRN, which includes the encoding of long-term and short-term information.

Long-Term Information. To capture the long-term information of each node from its neighbors, we reference graph neural networks and recurrent neural networks, which explicitly consider the relationship of nodes with their neighbors and sequence dependence of neighbors, respectively. Furthermore, we also design an order-aware attention mechanism that is more suitable for dynamic sequential recommendation.

- *Graph Convolution Neural Networks*. GCN [13] is an intuitive approach that aggregates all neighbor node embeddings directly:

$$\mathbf{h}_u^L = \frac{1}{|\mathcal{N}_u|} \sum_{i \in \mathcal{N}_u} \mathbf{W_1}^{(l-1)} \mathbf{h}_i^{(l-1)}, \tag{1}$$

$$\mathbf{h}_i^L = \frac{1}{|\mathcal{N}_i|} \sum_{u \in \mathcal{N}_i} \mathbf{W_2}^{(l-1)} \mathbf{h}_u^{(l-1)}, \tag{2}$$

where $\mathbf{W_1}^{(l-1)}$, $\mathbf{W_2}^{(l-1)} \in \mathbb{R}^{d \times d}$ are encoding matrix parameters of item and user in the $l-1$-th layer, where $|\mathcal{N}_u|$ and $|\mathcal{N}_i|$ are the number of u's and i's neighbor nodes.

- *Recurrent Neural Networks*, such as GRU net, is an effective network to model the sequence dependencies. So, we utilize GRU net to calculate the long-term preference/character for user/item nodes from their neighbors, which is computed as

$$\mathbf{h}_u^L = \mathrm{GRU}_U^{(l)}\left(\mathbf{h}_{i_1}^{(l-1)}, \ldots, \mathbf{h}_{i_{|\mathcal{N}_u|}}^{(l-1)}\right), i \in \mathcal{N}_u, \tag{3}$$

$$\mathbf{h}_i^L = \mathrm{GRU}_I^{(l)}\left(\mathbf{h}_{u_1}^{(l-1)}, \cdots, \mathbf{h}_{u_{|\mathcal{N}_i|}}^{(l-1)}\right), u \in \mathcal{N}_i, \tag{4}$$

where $(\mathbf{h}_{i_1}^{(l-1)}, \ldots, \mathbf{h}_{i_{|\mathcal{N}_u|}}^{(l-1)})$ and $(\mathbf{h}_{u_1}^{(l-1)}, \cdots, \mathbf{h}_{u_{|\mathcal{N}_i|}}^{(l-1)})$ are into GRU in chronological order.

- *Dynamic Graph Attention Mechanism*. In general, the GNN models focus on explicitly capturing the relationship between the central and neighboring nodes while ignoring the sequence information among neighbors. The sequence model is the opposite. To effectively differentiate the influence of different items and make full use of the user-item interaction order information, we combine graph attention mechanism and the encoding of sequence information to define a dynamic attention module (DAT).

  Specifically, for each interaction quintuple (u, i, t, o_u^i, o_i^u), we define r_u^i as the relative-order of item i to the last item in the neighbors of the user node, i.e., $r_u^i = |\mathcal{N}_u| - o_u^i$. For each discrete values r, we assign a unique $\mathbf{p}_r^K \in \mathbb{R}^d$ parameter vector as the relative-order embedding to encode the order information. Then, the attention coefficients between $\mathbf{h}_u^{(l-1)}$ and its neighbor node representation $\mathbf{h}_i^{(l-1)}$ are influenced by $\mathbf{p}_{r_i}^K$. So, we define a relative order-aware attention mechanism to differentiate the importance weight e_{ui} of items to the user, taking $l-1$-th layer node embedding $\mathbf{h}_u^{(l-1)}$ and $\mathbf{h}_i^{(l-1)}$ as the input, formulated as

$$e_{ui} = \frac{1}{\sqrt{d}}\left(\mathbf{W_2}^{(l-1)}\mathbf{h}_u^{(l-1)}\right)^{\mathrm{T}}\left(\mathbf{W_1}^{(l-1)}\mathbf{h}_i^{(l-1)} + \mathbf{p}_{r_u^i}^K\right), \tag{5}$$

  where $\mathbf{h}_u^{(0)}$ and $\mathbf{h}_i^{(0)}$ are the user embedding $\mathbf{e}_u$ and the item embedding $\mathbf{e}_i$, respectively. d is the dimension of the embeddings, the scale factor $\sqrt{d}$ is to avoid exceedingly large dot products and speed up convergence. The weighting scores between user and its neighbors are obtained via the softmax function

$$\alpha_{ui} = \mathrm{softmax}(e_{ui}). \tag{6}$$

  Thus, the *long-term preference* of user can be obtained by aggregating the information from its all neighbors adaptively

$$\mathbf{h}_u^L = \sum_{i \in \mathcal{N}_u} \alpha_{ui}\left(\mathbf{W_1}^{(l-1)}\mathbf{h}_i^{(l-1)} + \mathbf{p}_{r_u^i}^V\right), \tag{7}$$

  where $\mathbf{p}_{r_u^i}^V \in \mathbb{R}^d$ is relative-order embedding to capture the order information in user message aggregation.

  Similarly, the long-term character of item can be calculated by,

$$\mathbf{h}_i^L = \sum_{u \in \mathcal{N}_i} \beta_{iu}\left(\mathbf{W_2}^{(l-1)}\mathbf{h}_u^{(l-1)} + \mathbf{p}_{r_i^u}^V\right), \tag{8}$$

  where

$$\beta_{iu} = \mathrm{softmax}(e_{iu}), \tag{9}$$

$$e_{iu} = \frac{1}{\sqrt{d}}\left(\mathbf{W_1}^{(l-1)}\mathbf{h}_i^{(l-1)}\right)^{\mathrm{T}}\left(\mathbf{W_2}^{(l-1)}\mathbf{h}_u^{(l-1)} + \mathbf{p}_{r_i^u}^K\right), \tag{10}$$

  $r_i^u = |\mathcal{N}_i| - o_i^u$, and $\mathbf{p}_{r_i^u}^V \in \mathbb{R}^d$ is relative-order embedding to capture the order information in item message aggregation.

Short-Term Information. In recommender system, the *short-term information* of the user reflects his or her latest interest, many work [12] utilize the last interaction item embedding as user's short-term embedding, but this ignores the reliance on historical information. To this end, we consider attention mechanism to model the explicit effectiveness between last interaction with historical interactions.

- *Attention Mechanism*. We consider the attention mechanism between the last item/user with each historical item/user

$$\mathbf{h}_u^S = \sum_{i \in \mathcal{N}_u} \hat{\alpha}_{ui}\mathbf{h}_i^{(l-1)}, \tag{11}$$

$$\mathbf{h}_i^S = \sum_{u \in \mathcal{N}_i} \hat{\beta}_{iu}\mathbf{h}_u^{(l-1)}, \tag{12}$$

  where attention coefficient $\hat{\alpha}_{i_k}$ and $\hat{\beta}_{u_k}$ can be calculated by,

$$\hat{\alpha}_{ui} = \mathrm{softmax}\left(\frac{1}{\sqrt{d}}\left(\mathbf{W}_3^{(l-1)}\mathbf{h}_{i_{last}}^{(l-1)}\right)^{\mathrm{T}}\left(\mathbf{W}_2^{(l-1)}\mathbf{h}_i^{(l-1)}\right)\right), \tag{13}$$

$$\hat{\beta}_{iu} = \mathrm{softmax}\left(\frac{1}{\sqrt{d}}\left(\mathbf{W}_4^{(l-1)}\mathbf{h}_{u_{last}}^{(l-1)}\right)^{\mathrm{T}}\left(\mathbf{W}_1^{(l-1)}\mathbf{h}_u^{(l-1)}\right)\right), \tag{14}$$

  where parameters $\mathbf{W}_3$ and $\mathbf{W}_4 \in \mathbb{R}^d$ are to control the weight of last interaction, i_{last} and u_{last} denote the last occurrence of item and user in $\mathcal{N}_u$ and $\mathcal{N}_i$ respectively.

4.3.2 Node Updating

In this stage, we aggregate the long-term embedding, short-term embedding, and the previous layer embedding to update the node's representation of $\mathcal{G}_u^m(t_k)$.

User Node Updating. For user node, the representation updating rule from $l-1$-th layer to l-th layer can be formulated as

$$\mathbf{h}_u^{(l)} = \tanh\left(\mathbf{W_5}^{(l)}\left[\mathbf{h}_u^L \parallel \mathbf{h}_u^S \parallel \mathbf{h}_u^{(l-1)}\right]\right), \tag{15}$$

where $\mathbf{W_5}^{(l)} \in \mathbb{R}^{d \times 3d}$ is a user update matrix to control the information of $\mathbf{h}_u^L$, $\mathbf{h}_u^S$, and $\mathbf{h}_u^{(l-1)}$.

Item Node Updating. Analogously, the item representation updating rule is

$$\mathbf{h}_i^{(l)} = \tanh\left(\mathbf{W_6}^{(l)}\left[\mathbf{h}_i^L \parallel \mathbf{h}_i^S \parallel \mathbf{h}_i^{(l-1)}\right]\right), \tag{16}$$

where $\mathbf{W_6}^{(l)} \in \mathbb{R}^{d \times 3d}$ is an item update matrix to control the information reservation of $\mathbf{h}_i^L$, $\mathbf{h}_i^S$, and $\mathbf{h}_i^{(l-1)}$.

4.4 Recommendation and Optimization

In our model, predicting the next interaction of $S_u = (i_1, i_2, \ldots, i_k)$ is equivalent to predicting the link of user node u of sub-graph $\mathcal{G}_u^m(t_k)$. In this subsection, we design the link prediction function to determine the items that the user may interact with next.

After acting L-layers DGRN on $\mathcal{G}_u^m(t_k)$, we obtain the multiple embedding $\{\mathbf{h}_u^{(0)}, \mathbf{h}_u^{(1)}, \ldots, \mathbf{h}_u^{(L)}\}$ of node u, which includes user embedding $\mathbf{h}_u^{(l)}$ in each layer. The user embedding in different layers emphasizes the various user preferences [3]. As a result, we concatenate user multiple embeddings to get the final embedding for node u

$$\mathbf{h_u} = \mathbf{h}_u^{(0)} \parallel \mathbf{h}_u^{(1)} \cdots \parallel \mathbf{h}_u^{(L)}. \tag{17}$$

For a given candidate item $i \in \mathcal{I}$, the link function is defined as

$$\mathbf{s}_{ui} = \mathbf{h_u}^{\mathrm{T}} \mathbf{W_P} \mathbf{e}_i, \tag{18}$$

where vector $\mathbf{s}_u = (\mathbf{s}_{u1}, \mathbf{s}_{u2}, \ldots, \mathbf{s}_{u|\mathcal{I}|})$ represents the score vector of u for each candidate item. $\mathbf{W_P} \in \mathbb{R}^{(L+1)d \times d}$ is the trainable transformation matrix.

Algorithm 2. The DGSR Framework (Forward Propagation)

Input: $S^u = (i_1, i_2, \ldots, i_k)$, timestamp sequence $T^u = (t_1, t_2, \ldots, t_k)$, all sequences of users, and DGRN layer number L.
Output: The next item i_{k+1} of S^u.

// Dynamic Graph Construction.
Convert all user sequences into a dynamic graph $\mathcal{G}$
// The sub-graph of generation for S^u.
Run the **Algorithm 1** to generate $\mathcal{G}_u^m(t_k)$ from $\mathcal{G}^{t_k}$
// The initialization of node representation.
$\mathbf{h}_u^{(0)} \leftarrow \mathbf{e}_u, \mathbf{h}_i^{(0)} \leftarrow \mathbf{e}_i, \forall u, i \in \mathcal{G}_u^m(t_k)$
// The update of user and item by DGRN.
for $l \in [1:L]$ **do**
  $\mathbf{h}_u^{(l)}, \mathbf{h}_i^{(l)} \leftarrow \mathrm{DGRN}(\mathbf{h}_u^{(l-1)}, \mathbf{h}_i^{(l-1)}, \mathcal{G}_u^m(t_k))$:
  $\mathbf{h}_u^L, \mathbf{h}_i^L \leftarrow$ Long-term Information Encoding
  $\mathbf{h}_u^S, \mathbf{h}_i^S \leftarrow$ Short-term Information Encoding
  $\mathbf{h}_u^{(l)} \leftarrow \tanh(\mathbf{W_5}^{(l)}[\mathbf{h}_u^L \parallel \mathbf{h}_u^S \parallel \mathbf{h}_u^{(l-1)}])$
  $\mathbf{h}_i^{(l)} \leftarrow \tanh(\mathbf{W_6}^{(l)}[\mathbf{h}_i^L \parallel \mathbf{h}_i^S \parallel \mathbf{h}_i^{(l-1)}])$
// The prediction of next item.
$\mathbf{h_u} = \mathbf{h}_u^{(0)} \parallel \mathbf{h}_u^{(1)}, \ldots, \parallel \mathbf{h}_u^{(L)}$
Next item $\leftarrow \mathrm{argmax}_{i \in \mathcal{V}}(\mathbf{h_u}^{\mathrm{T}} \mathbf{W_P} \mathbf{e}_i)$

To learn model parameters, we optimize the cross-entropy loss. The normalized vector of user u's score for candidate item is

$$\hat{\mathbf{y}}_\mathbf{u} = \mathrm{softmax}(\mathbf{s_u}). \tag{19}$$

The objective function is as follows:

$$Loss = -\sum_{\mathcal{S}} \sum_{i=1}^{|\mathcal{I}|} \mathbf{y}_{ui} \log(\hat{\mathbf{y}}_\mathbf{ui}) + (\mathbf{1} - \mathbf{y_{ui}}) \log(\mathbf{1} - \hat{\mathbf{y}}_\mathbf{ui}) + \lambda \|\boldsymbol{\Theta}\|_\mathbf{2}, \tag{20}$$

where $\mathbf{y}_u$ denotes the one-hot encoding vector of the ground truth items for the next interaction of S^u. Θ denotes all model parameters, $\|\cdot\|_2$ is L_2 norm. λ is to control regularization strength.

4.5 Model Discussions

In this subsection, we first analyze the computational complexity of DGSR, then compare our model with some representative sequential recommendation models.

4.5.1 Computational Complexity Analysis

We assume that the number of all interactions between users and items is $\overline{N}$, the maximum length of each user sequence is n, the maximum sampling number of users in each layer is n, the order of sub-graph is m, and the batch size is B. For the Dynamic Graph Construction component, the space and time complexity of the dynamic graph construction are $O(\overline{N})$. In the Sub-Graph Sampling component, the maximum number of operations for each user node to obtain a m-order sub-graph is

$$\mathcal{D} = \begin{cases} n, & m = 1 \\ 2n + n^2, & m = 2 \\ (k+1)n + (k+1)n^2 + (k-1)n^3, & m = 2k+1 \\ (k+2)n + (k+1)n^2 + kn^3, & m = 2k+2, \end{cases} \tag{21}$$

where k is a natural number and is greater than or equal to 1. So, the space and time complexity of sub-graph sampling are $O(B\mathcal{D})$. The maximum number of edges of each sub-graph is

$$\mathcal{R} = \begin{cases} (k+1)n + kn^2, & m = 2k+1 \\ (k+1)n + (k-1)n^2, & m = 2k, \end{cases} \tag{22}$$

where k is a natural number. Consequently, in each layer of DGRN, the complexity of message propagation in long- and short-term encoding and node updating is $O(\mathcal{R})$. Therefore, the space and time complexity of DGRN are $O(3LB\mathcal{R})$ in each batch, where L is the layer number.

In practice, the maximum value of m is generally set to 3. So, the space and time complexity of sub-graph sampling and DGRN are at most $O(3LBn^2)$. We could also operate the dynamic graph construction and sub-graph sampling before training and testing to improve training and testing speed. Therefore, the computational burden of our model is acceptable.

4.5.2 Compared With Representative Models

Some sequence models encoding user preference only depends on its intra-sequence, and does not explicitly utilize other sequence information, such as TiSASRec [23], SR-GNN [15], and HGN [49], which can be viewed as special cases of our DGSR. Specifically, within the one layer DGRN net, we can replace our current setting with some complex neural network, self-attention net, GGNN net, or Gated net in message propagation mechanism of *item* $\rightarrow$ *user*, and disable the message propagation and node update of *item* $\rightarrow$ *user*. Then, $\mathbf{h}_u^{(1)}$ is treated as u's final preference representation. So, as a new framework, our model can fuse nearly all single-sequence models by modifying the message propagation mechanism part.

There are also some models [26], [27], [29] which are designed to utilize cross sequence information or capture the item relations between different sequences. However, They have many differences and limitations compared with DGSR. For example, CSRM [27] considers neighborhood sequences by directly calculating the similarity between them and target sequence but fails to utilize the fine-grained interaction information of users, including the interaction-order between each item with all users that interact with it. Compared with that, our model measures the similarity between different sequences based on the well-designed message passing mechanism, which could promote the utilization of interactions between users and items. HyperRec [26] adopts hypergraph to associate the high-order correlations connections between items in each period. Nonetheless, hypergraph is a rough way to model user-item interaction, which results in much-refined information being neglected, such as the explicit order information in each cross sequence. The dynamic graph constructed by our DGSR can be more flexible to represent richer interaction information. In social recommendation, DGRec [28] explicitly associates different user sequences through social attribute information, but not all data have social relationship attributes in sequential recommendation scenario. Our model can also explicitly associate different user sequences without relying on other auxiliary information.

5 Experiments

In this section, we perform experiments on four real-world datasets to evaluate the performance of our model. We aim to answer the following questions through experiments.

- *RQ1*: How does DGSR perform compared with state-of-the-art sequential recommendation methods?
- *RQ2*: How effective is the dynamic graph recommendation network component in DGSR?
- *RQ3*: What are the effects of different hyper-parameter settings (DGRN layer number, sub-graph sampling size, maximum sequence length, and the embedding size) on DGSR.

5.1 Datasets

To evaluate the effectiveness of our model, we conduct experiments on four datasets from real-world platforms. These datasets are widely used in evaluating sequential recommendation methods:

- *Amazon*[1]: A series of datasets introduced in [50], comprising large corpora of product reviews crawled from *Amazon.com*. We consider three categories, *Amazon-CDs*, *Amazon-Games*, and *Amazon-Beauty*. This dataset is notable for its high sparsity and variability.
- *MovieLens*[2]: We use the version (MoveLens-1M) [50] that includes 1million user ratings.

All of these datasets contain the timestamps or specific dates of interactions. For all datasets, we treat the presence of a review or rating as implicit feedback and discard users and items with fewer than five related actions [11]. After precessing, the data statistics are shown in Table 1. For each user sequence, we use the most recent item for testing, the second recent item for validation, and the remaining items for the training set. To fully capture the dynamic collaborative signals, we segment each sequence S^u into a series of sequences and labels. For example, for an input $S^u = (i_1, i_2, i_3, i_4)$ and $T^u = (t_1, t_2, t_3, t_4)$, we generate sequences and labels as $[i_1] \rightarrow i_2$, $[i_1, i_2] \rightarrow i_3$ and $[i_1, i_2, i_3] \rightarrow i_4$. Then, the corresponding sub-graph and the node to linked are $(\mathcal{G}_u^m(t_1), i2)$, $(\mathcal{G}_u^m(t_2), i3)$, and $(\mathcal{G}_u^m(t_3), i_4)$. This processing can be taken before training and testing to reduce the training and inference time.

TABLE 1
The Statistics of the Datasets

Datasets	Beauty	Games	CDs	ML-1M
# of Users	52,024	31,013	17,052	6,040
# of Items	57,289	23,715	35,118	3,416
# of Interactions	394,908	287,107	472,265	999,611
Average length	7.6	9.3	27.6	165.5
Density	0.01%	0.04%	0.08%	4.80%

5.2 Experiment Settings

5.2.1 Compared Methods

To demonstrate the effectiveness, we compare our proposed DGSR with the following methods:

- *BPR-MF* [51], a matrix factorization based model that learns pairwise personalized ranking from user implicit feedback.
- *FPMC* [4], a model that combines matrix factorization and first-order Markov Chains to capture users' long-term preferences and item-to-item transitions.
- *GRU4Rec+* [7], an improved RNN-based model that adopts a different loss function and sampling strategy on Top-K recommendation.
- *Caser* [10], a CNN-based model capturing high-order Markov chains by applying convolution operations on the embedding of the L recent items.
- *SASRec* [11], a self-attention-based model to identify relevant items for predicting the next item.
- *SR-GNN* [15], a GNN-based model to capture the complex transition relationships of items for the session-based recommendation.
- *HGN* [49], a sequence model that contains feature gating, instance gating, and instance gating modules to select important features and explicitly capture the item relations.
- *TiSASRec* [23], interval aware self-attention based model, which models both the absolute positions as well as the time intervals between them in a sequence.
- *GCE-GNN* [30], a GNN-based model that leverages global and session-level graphs to capture item transitions overall sessions for better inferring the user preference.
- *SERec* [31], a GNN-based model that utilizes a heterogeneous graph neural network to learning user and item representations with the knowledge from social networks. Due to our dataset lacking social information, we only consider user-to-item and item-to-item relationships in their model.
- *HyperRec* [26], a hypergraphs based model that adopts hypergraph to capture multi-order connections between items for next-item recommendation.

5.2.2 Evaluation Metrics

We adopt two widely-used metrics [11], Hit@K and NDCG@K, to evaluate all methods. Hit@K indicates the proportion of the ground-truth items among the top@K

1. http://jmcauley.ucsd.edu/data/amazon
2. https://grouplens.org/datasets/movielens/1m/

TABLE 2
Performance of DGSR and Compared Methods in Terms of Hit@10 and NDCG@10

Datasets	Metric	BPR-MF	FPMC	GRU4Rec+	Caser	SASRec	SR-GNN	HGN	TiSASRec	GCE-GNN	SERec	HyperRec	DGSR
Beauty	NDCG@10	21.83	28.91	26.42	25.47	32.19	32.33	32.47	30.45	33.46	33.59	23.26	**35.90**
	Hit@10	37.75	43.10	43.98	42.64	48.54	48.62	48.63	46.87	49.12	49.63	34.71	**52.40**
Games	NDCG@10	28.75	46.80	45.64	45.93	53.60	53.25	49.34	50.19	53.68	53.71	48.96	**55.70**
	Hit@10	37.75	68.02	67.15	68.83	73.98	73.49	71.42	71.85	74.14	74.32	71.24	**75.57**
CDs	NDCG@10	36.26	33.55	44.52	45.85	49.23	48.95	49.34	48.97	49.05	48.34	47.16	**51.22**
	Hit@10	56.27	51.22	67.84	68.65	71.32	69.63	71.42	71.00	70.04	69.44	71.02	**72.43**
ML-1M	NDCG@10	32.87	48.66	53.14	53.29	57.34	54.23	52.57	56.57	56.31	57.01	54.61	**57.89**
	Hit@10	57.81	72.77	73.21	76.81	80.36	76.59	76.57	**80.41**	77.4	78.39	78.24	79.74

The best results are boldfaced. The underlined numbers are the second best results.

items, while NDCG@K is position-aware metric, and higher NDCG means target items tend to have more top rank positions. Following [11], [23], for each test sample, we randomly sample 100 negative items, and rank these items with the ground-truth item. We evaluate Hit@K and NDCG@K based on these 101 items. By default, we set K=10.

5.2.3 Parameter Setup

We implement our DGSR model in *DGL* Library[3] [52]. The embedding size is fixed to 50 for all methods. The maximum sequence length n is set to 50. The optimizer is the Adam optimizer [53], the learning rate is set to 0.01. Batch size is 50. λ is $1e-4$. We set the order of sub-graph sampling m to 3. The DAN layer number L is set to 3 for Beauty and CDs, 2 for Games and ML-1M. We run the evaluation four times with different random seeds and report the mean value of each method. For the compared methods, we use the default hyperparameters except for dimensions. All experiments are conducted on a computer server with eight NVIDIA GeForce RX2080Ti (11GB) and four Intel Xeon E5-2660 v4 CPUs.

5.3 Performance Comparison (RQ1)

We first report the performance of all the methods. Table 2 summarizes the performance comparison results. The following observation can be obtained:

- DGSR achieves the best performance on four datasets with most evaluation metrics. In particular, DGSR improves over the strongest baselines w.r.t NDCG@10 by 6.87%, 3.71%, 3.81% in Beauty, Games, and CDs, respectively. Notably, Beauty is the most sparse and short dataset, so many users and items only have a few interactions. In our model, the high-order connectivity of a dynamic graph alleviates this issue. So, there is a significant improvement in Beauty. By stacking the DGRN layers, DGSR can utilize cross-sequences information explicitly to provide more auxiliary information for prediction. While TiSASRec, HGN, SR-GNN, and SASRec only encode each sequence independently as the user's dynamic interest representation. Significantly, HyperRec, SERec, and GCE-GNN utilize much correlated user interaction information, but perform worse than our DGSR, especially on Beauty and Games. We believe that the reason is that these models ignore the interaction order information of correlated user sequences. And Beauty and Games have stronger sequential properties than CDs, resulting in significant improvement in the performance of DGSR over them on Beauty and Games. DGSR does not significantly exceed SASRec and TiSASRec on ML-1M. The reason might be that ML-1M is denser than other datasets, making the user intra sequence able to provide sufficient information for sequential recommendation.
- SASRec, HGN, SR-GNN, and TiSASRec achieve better performance than neural methods GRU4Rec+ and Caser. One possible reason is that they could explicitly capture the item-item relations by employing attention or hierarchical gating mechanism. Caser generally achieves better performance than GRU4Rec+ in most cases. Such improvement might be attributed to the CNN module, which could capture the more complex behavior pattern than the GRU net. Compared with the excellent performance in the session-based recommendation scenario, the performance of SR-GNN is flat in the sequential recommendation. One possible reason is that the lack of repetitiveness of our data makes it challenging for the user sequence to form a graph structure. BPR-MF achieves poor performance on four datasets. Since BPR-MF can only capture users' general interests, it is challenging to model the user's behavior sequence. GRU4Rec+ slightly underperforms FPMC in Beauty and Games while performing better in CDs. The reason might be that FPMC focuses on dynamic transitions of items, so they perform better on sparse datasets [23].

5.4 Study of Dynamic Graph Recommendation Network (RQ2)

5.4.1 Effect of Long- and Short-Term Modules

To investigate DGRN component's superiority in DGSR, we compare DGSR with different variants on four datasets, which set the various modules for encoding long-term and short-term information. We show the variant models and their results in Table 3 and have the following findings:

3. https://www.dgl.ai/

TABLE 3
Performance of Compared With Different Model Variants in Terms of NGCD@10 and Hit@10 ("–" Indicates DGSR Does Not Consider the Setting of This Part)

Variants	Ablation		Beauty		Games		CDs		ML-1M	
	Long-term	Short-term	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10
DGSR-G	GCN	–	33.75	49.94	53.44	73.23	48.66	70.43	56.10	79.02
DGSR-R	RNN	–	35.23	51.44	54.70	74.73	49.57	71.22	57.12	79.45
DGSR-D	DAT	–	35.25	51.36	55.12	74.83	49.66	71.26	57.76	79.48
DGSR-L	–	Last	30.87	46.13	52.43	72.18	46.23	67.38	54.11	75.56
DGSR-A	–	ATT	34.76	51.00	54.30	74.32	48.78	70.09	54.89	76.23
DGSR-GL	GCN	Last	35.24	51.18	54.76	74.58	49.62	70.76	56.45	78.14
DGSR-RL	RNN	Last	35.47	51.68	54.86	74.84	50.26	71.24	57.33	78.98
DGSR-DL	DAT	Last	35.62	51.92	55.53	75.07	50.72	72.06	57.88	79.56
DGSR-GA	GCN	ATT	35.00	51.05	54.97	74.78	50.05	71.46	56.17	79.11
DGSR-RA	RNN	ATT	35.17	51.46	55.02	74.88	51.19	**72.55**	57.43	79.62
DGSR-DA	DAT	ATT	**35.90**	**52.40**	**55.70**	**75.57**	**51.22**	72.43	**57.89**	**79.74**

- DGSR-D outperforms DGSR-R and DGSR-G in Games, CDs and ML-1M datasets. We attribute the improvement to the combination of attention mechanism and relative-order embedding, which could adequately distill the long-term information from neighbors of each node. DGSR-R achieves competitive results in Beauty. The reason might be that the length of sequence is small, GRU net could model their dependencies of sequence like dynamic attention module. The GCN-based variants achieve poor performance on four datasets. It is probably because the GCN module treats all neighbor nodes as equally important, which introduces more noise in message propagation. DGSR-A also performs better than DGSR-L. It verifies that only utilizing the last interaction embedding is insufficient to capture the short-term information.
- All variants with two modules (long-term and short-term) are consistently superior to variants with single module (long-term or short term). It illustrates the necessity of combining long-term and short-term information. Although DGSR-R performs better than DGSR-D in Beauty, DGSR-DA is superior to DGSR-RA and DGSR-RL. One possible reason is that DGSR-DA considers the relationship between central node and neighbor nodes, which is conducive to information propagation in the dynamic graph. In contrast, DGSR-RL and DGSR-RA only focus on the interactions of neighbors and ignore the roles of central node.

5.4.2 Effect of Long- and Short-Term Information

We also explore the effects of users' and items' long- and short-term information. Specifically, we modify DGSR by removing the user's long-term preference (w/o user-long), user's short-term preference (w/o user-short), item's long-term character (w/o item-long), and item's short-term character (w/o item-short), respectively. As demonstrated in Fig. 3, the result shows that both long- and short-term information of users and items are beneficial for improving the recommendation performance. In addition, DGSR (w/o user-long) performs worse in most cases, which means the long-term preference of users is more important for recommendation. DGSR (w/o item-long) has the most severe performance degradation than DGSR at Games, CDs, and ML-1M datasets. The reason might be that the intrinsic properties of items in the Games, CD, and ML-1M categories are relatively stable. Moreover, unlike other datasets, the performances of DGSR (w/o item-short) and DGSR (w/o user-short) have a significant decline compared to DGSR at Beauty dataset. One possible reason is that goods in the Beauty category have seasonal or fashion attributes, making users' short-term preferences and items' short-term character susceptible to short-term volatility. Therefore, comprehensive consideration of users' and items' long- and short-term information is crucial in Dynamic Graph Recommendation Network.

5.4.3 Effect of Relative Order-Aware Attention Mechanism

To understand the relative-order embedding, we further visualize the attention values of DGSR and DGSR w/o relative-order embedding, as illustrated in Fig. 4. The results show that the relative-order embedding promotes the DGSR to focus more on interactions occurring in the recent period, which is beneficial for modeling sequence properties.

5.5 The Sensitivity of Hyper-Parameters (RQ3)

To explore the effect of explicit modeling dynamic collaborative information among user sequences on DGSR, we

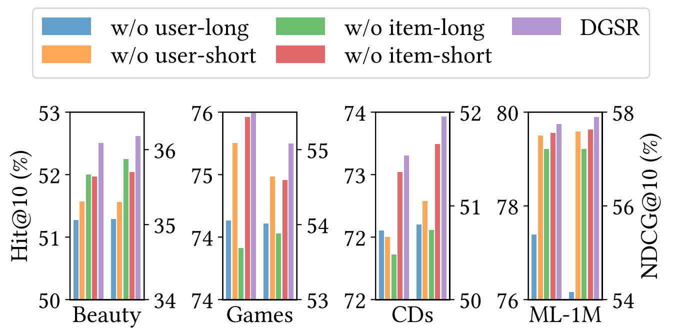


Fig. 3. Effect of users' and items' long- and short-term information (the y-axis on the left is Hit@10 value, and the right is NGCD@10 value).

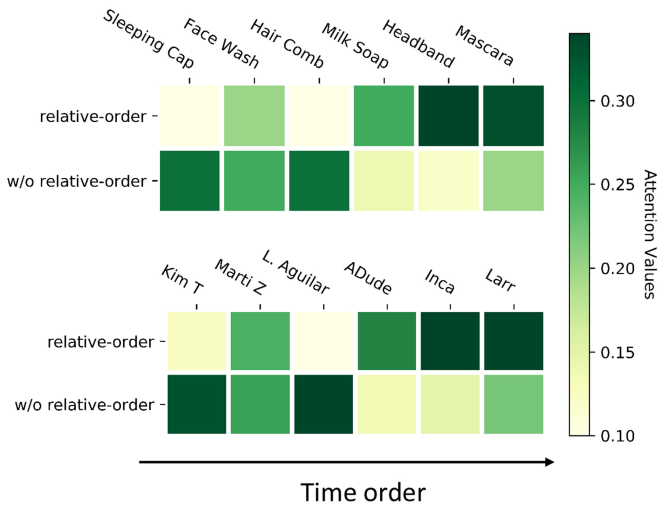


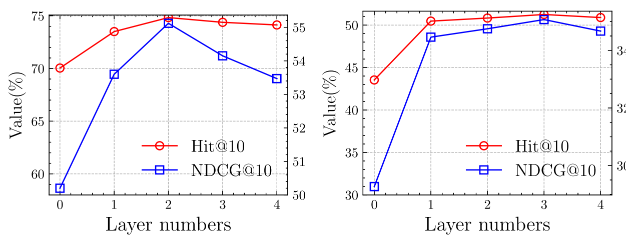
Fig. 4. Attention visualization of DGSR and DGSR w/o relative-order embedding. The first two rows represent the attention values of item sequence that a user interacts with. The last two rows represent the attention values of user sequence who have interacted with an item.

study how three hyperparameters, the DGRN layer number l, the order of sub-graph, and the maximum length of user sequence n , affect the performance of DGSR.

5.5.1 Effect of DGRN Layer Numbers

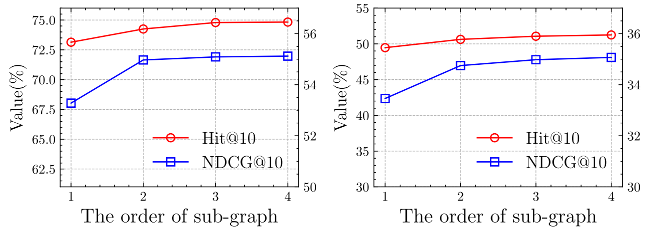
We conduct our method with different DGRN layer number l on Games and Beauty dataset. DGSR-0 represents only use user embedding and last item embedding for recommendation. DGSR-1 represents the DGRN with one layer, indicating to use only intra sequence information for prediction. DGSR-l (l >1) indicates DGSR could utilize l-order user sequence information to make predication. From Fig. 5, we find that :

- Increasing the layer of DGSR is capable of promoting the performance substantially. It demonstrates that exploiting high-order user sequences information explicitly can effectively improve recommendation performance. DGSR-2 and DGSR-3 achieve the best performance on Games and Beauty, respectively. Because Beauty is more sparse than Games, Beauty requires more layers to introduce more contextual information.
- When further stacking propagation layer, we find that the performances of DGSR-3 and DGSR-4 begin to deteriorate. The reason might be that the use of far more propagation layers may lead to over smoothing, which is also consistent to the findings in [54]. DGSR-1 consistently outperforms DGSR-0 in all cases, even outperforms most baselines. We attribute to the power of the message propagation mechanism in dynamic graph recommendation network, which could effectively encode the order information in user sequences to extra users' dynamic preferences accurately.



(a) Games (b) Beauty

Fig. 5. Effect of propagation layer numbers (the y-axis on the left is Hit@10 value, and the right is NGCD@10 value).



(a) Games (b) Beauty

Fig. 6. Effect of the sub-graph sampling size (the y-axis on the left is Hit@10 value, and the right is NGCD@10 value).

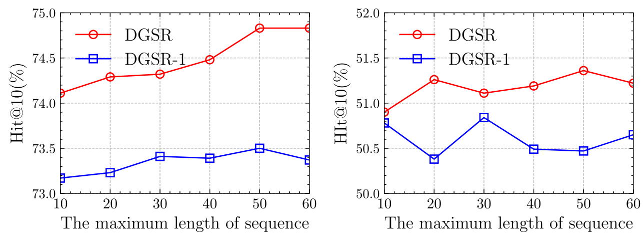
5.5.2 Effect of the Sub-Graph Sampling Size

We conduct our method with different sub-graph sampling size. The order of sub-graph m determines the size of the sampling. In particular, we search the m in the range of $\{1, 2, 3, 4\}$. The results in Fig. 6 show that when m is increased from 1 to 3, the model performance can be effectively improved. The reason is that a larger-sized sub-graph can provide more dynamic contextual information for each user sequence to assist in prediction. With the increase of m, the model performance tends to be stable because of the limitation of the number of DGRN layers. In practice, we cannot blindly increase the value of m because the limitation of computing resources.

5.5.3 Effect of the Maximum Sequence Length

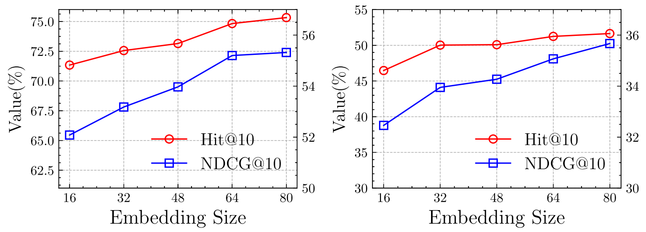
We train and test our method on the Games and Beauty datasets with n from 10 to 60, while keeping other optimal hyperparameters unchanged. Besides, to further investigate the benefit of explicitly utilizing the dynamic collaborative information, we also conduct DGSR-1 with different n. Fig. 7 shows the Hit@10 results. We have the following findings:

- Increasing the n of DGSR from 10 to 50 consistently improves the performance of Games data. DGSR performs better on the beauty when setting n to be 20 and 50. However, blindly increasing the n does not necessarily improve the performance of DGSR



(a) Games (b) Beauty

Fig. 7. Effect of the maximum length of user sequence.



(a) Games (b) Beauty

Fig. 8. Effect of the embedding size (the y-axis on the left is Hit@10 value, and the right is NGCD@10 value).

and DGSR-1. It is likely to bring noise and cause the performance to attenuate.

- Compared with DGSR-1, DGSR performs better than DGSR-1 at each value of n. To be specific, even when n is set to 10, DGSR is still better than the best performance of DGSR-1, which implies that explicitly utilizing high-order contextual information of user sequence can alleviate the issue of insufficient user history information, thus improving the performance of recommendation.

5.5.4 *Effect of the Embedding Size*

We further analyse the impact of different dimensionality of embeddings. Fig. 8 describes the performance of model under the embedding size from 16 to 80. We can observe that the performance of model gradually improves as the dimensionality increases. With the further increase of the dimensionality, the performance tends to be stable. This verifies the stability of our model in different dimensions.

6 Conclusion

This work explores the necessity of explicit modeling dynamic collaborative signals among different user sequences in sequential recommendations. Inspired by dynamic graph neural networks, we propose a novel method, DGSR. In DGSR, all user sequences are converted into a dynamic graph, which contains the chronological order and timestamps of user-item interactions. The key of DGSR is the well-designed Dynamic Graph Recommendation Network, which realizes the explicit encoding of the dynamic collaborative information among different user sequences. The next-item prediction task is finally converted into a node-link prediction of the dynamic graph so that the model can be trained end-to-end. Extensive experiments on four real-world datasets verify the effectiveness and rationality of DGSR.

References

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[3] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2019, pp. 165–174.

[4] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 811–820.

[5] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2015, *arXiv:1511.06939*.

[6] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 130–137.

[7] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proc. 27th ACM Int. Conf. Informat. Knowl. Manage.*, 2018, pp. 843–852.

[8] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 338–342.

[9] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[10] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 565–573.

[11] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 197–206.

[12] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "Stamp: Short-term attention/memory priority model for session-based recommendation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1831–1839.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[14] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[15] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 346–353.

[16] M. Zhang, S. Wu, M. Gao, X. Jiang, K. Xu, and L. Wang, "Personalized graph neural networks with attention mechanism for session-aware recommendation," *IEEE Trans. Knowl. Data Eng.*, early access, Oct. 15 2020, doi: 10.1109/TKDE.2020.3031329.

[17] R. Qiu, J. Li, Z. Huang, and H. Yin, "Rethinking the item order in session-based recommendation with graph neural networks," in *Proc. 28th ACM Int. Conf. Informat. Knowl. Manage.*, 2019, pp. 579–588.

[18] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *Proc. Int. Conf. on Learn. Representations*, 2018.

[19] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *Proc. IEEE 16th Int. Conf. Data Mining*, 2016, pp. 191–200.

[20] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 161–169.

[21] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2016, pp. 729–732.

[22] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proc. ACM Conf. Informat. Knowl. Manage.*, 2017, pp. 1419–1428.

[23] J. Li, Y. Wang, and J. McAuley, "Time interval aware self-attention for sequential recommendation," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 322–330.

[24] C. Xu *et al.*, "Graph contextualized self-attention network for session-based recommendation," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3940–3946.

[25] C. Ma, L. Ma, Y. Zhang, J. Sun, X. Liu, and M. Coates, "Memory augmented graph neural networks for sequential recommendation," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 5045–5052.

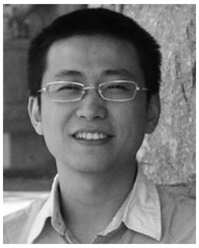
[26] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, "Next-item recommendation with sequential hypergraphs," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2020, pp. 1101–1110.

[27] M. Wang, P. Ren, L. Mei, Z. Chen, J. Ma, and M. de Rijke, "A collaborative session-based recommendation approach with parallel memory modules," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2019, pp. 345–354.

[28] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, 2019, pp. 555–563.

[29] Y. Ji *et al.*, "Temporal heterogeneous interaction graph embedding for next-item recommendation," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2020, pp. 314–329.
[30] Z. Wang, W. Wei, G. Cong, X.-L. Li, X.-L. Mao, and M. Qiu, "Global context enhanced graph neural networks for session-based recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2020, pp. 169–178.
[31] T. Chen and R. C.-W. Wong, "An efficient and effective framework for session-based social recommendation," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 400–408.
[32] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
[33] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
[34] H. Gao and S. Ji, "Graph U-Nets," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 2083–2092.
[35] X. Li, M. Zhang, S. Wu, Z. Liu, L. Wang, and P. S. Yu, "Dynamic graph collaborative filtering," in *Proc. IEEE Int. Conf. Data Mining*, 2020, pp. 322–331.
[36] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proc. ACM Conf. Informat. Knowl. Manage.*, 2017, pp. 387–396.
[37] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 571–578.
[38] P. Goyal, N. Kamra, X. He, and Y. Liu, "DynGEM: Deep embedding method for dynamic graphs," 2018, *arXiv:1805.11273*.
[39] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. Web Conf. Companion World Wide Web Conf.*, 2018, pp. 969–976.
[40] Y. Zuo, J. Guo, G. Liu, X. Hu, H. Lin, and J. Wu, "Embedding temporal network via neighborhood formation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2857–2866.
[41] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "DyRep: Learning representations over dynamic graphs," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.
[42] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1269–1278.
[43] H. Huang, Z. Fang, X. Wang, Y. Miao, and H. Jin, "Motif-preserving temporal network embedding," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2021, pp. 1237–1243.
[44] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," 2020, *arXiv:2002.07962*.
[45] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "DySAT: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 519–527.
[46] S. M. Kazemi *et al.*, "Representation learning for dynamic graphs: A survey," *J. Mach. Learn. Res.*, vol. 21, no. 70, pp. 1–73, 2020.
[47] A. Vaswani *et al.*, "Attention is all you need," 2017, *arXiv:1706.03762*.
[48] Z. Yu, J. Lian, A. Mahmoody, G. Liu, and X. Xie, "Adaptive user modeling with long and short-term preferences for personalized recommendation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 4213–4219.
[49] C. Ma, P. Kang, and X. Liu, "Hierarchical gating networks for sequential recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 825–833.
[50] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2015, pp. 43–52.
[51] S. Rendle, C. Freudenthaler, Z. Gantner, and S. Lars, "BPR: Bayesian personalized ranking from implicit feedback. uai'09," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.
[52] M. Wang *et al.*, "Deep graph library: Towards efficient and scalable deep learning on graphs," *Proc. Workshop Representation Learn. Graphs Manifolds*, 2019.
[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.
[54] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.

Mengqi Zhang is currently working toward the PhD degree in computer application technology with the University of Chinese Academy of Sciences, Beijing, China. His research interests include data mining, graph representation learning, and recommender systems.



Shu Wu (Senior Member, IEEE) is currently an associate professor with the Center for Research on Intelligent Perception and Computing. He has authored or coauthored more than 50 papers in the areas of data mining and information retrieval at international journals and conferences, such as IEEE TKDE, WWW, AAAI, SIGIR, and ICDM.

Xueli Yu is currently a research engineer with the Beijing Institute of General Artificial Intelligence. She has authored or coauthored several papers in such fields at international conferences, such as WWW, ACL, and SIGIR. Her research interests include natural language processing, information retrieval, and data mining.

Qiang Liu (Member, IEEE) received the PhD degree in pattern recognition from the Chinese Academy of Sciences. He is currently an assistant researcher with the Center for Research on Intelligent Perception and Computing, Institute of Automation, Chinese Academy of Sciences. He has authored or coauthored more than 30 papers in top-tier journals and conferences, such as IEEE TKDE, AAAI, IJCAI, NeurIPS, WWW, SIGIR, CIKM, and ICDM. His current research interests include data mining, recommender systems, text mining, knowledge graph, and graph representation learning.

Liang Wang (Fellow, IEEE) received the BEng and MEng degrees from Anhui University in 1997 and 2000, respectively, and the PhD degree from the Institute of Automation, Chinese Academy of Sciences (CASIA) in 2004. He is currently a full professor of the Hundred Talents Program, National Lab of Pattern Recognition, CASIA. He has widely authored or coauthored in highly ranked international journals, such as IEEE TPAMI and IEEE TIP and leading international conferences, such as CVPR, ICCV, and ICDM. His research interests include machine learning, pattern recognition, and computer vision. He is an IAPR Fellow.