

Exploiting High-order Relations Information for Learning User Intent with Data-driven

Abstract—Session-based recommendation (SBR) plays a vital role in assisting people in finding their desired information. However, prior arts focus only on modeling intra-session characteristics but pay less attention to inter-session relationships of items, which may result in irrelevant recommendations. Meanwhile, graph neural networks based methods regard the item transitions as pairwise relations and neglect the complex high-order information among items, which significantly limits the improvement of the recommendation. To overcome these challenges, we propose a novel yet powerful framework in this paper for SBR, which we refer to as a dual-view collaboration graph neural network (DC-GNN). The key insight is to exploit item transitions over sessions in a more subtle manner by modeling session-based data as a hypergraph. Specifically, DC-GNN learns two views of item embeddings from the graph level and hypergraph level, respectively: (i) Graph view, which is to learn the session-level item embedding by modeling pairwise item transitions within the session; and (ii) Hypergraph view, which is to learn the global item embedding by modeling beyond pairwise item transitions across all sessions. These two types of graph modeling with data-driven can provide complementary information for each other while keeping independent and exhibiting divergence to some degree. Extensive experiments on multiple real-world datasets demonstrate the superiority of the proposed model over the state-of-art methods, and the results validate the effectiveness of graph data modeling and self-supervised task.

Index Terms—High-order Relations, Graph Neural Network, Hypergraph

I. INTRODUCTION

With the ever-growing volume of online information, massive products, content, and services (which are uniformly described as items) are emerging every day [1]. Recommender systems (RS) play a significant role in providing personalized recommendations to alleviate the problem of information overload. Most recommender systems generally provide personalized recommendations based on definitive user information. However, users' identity information is often unavailable in many real-world scenarios, e.g., unregistered users or the ones who are reluctant to log in for privacy concerns [2]. Thereby, the traditional RS is less capable of predicting it. In such a situation, there is an urgently practical need to provide accurate recommendations with limited behavior information [3].

Session-based recommendation (SBR) has emerged in time, which encapsulates a range of latest consecutive user-item interactions as sessions for predicting the next items from the sequential behavior consumed, leads to better performance [4]. Most early studies on SBR deployed for e-commerce are based on relatively simple methods that do not use a user profile, e.g., item similarity [5], or chain-based [6]. While effective, the former often takes only the last click

or selection of the user into account, ignoring past clicks' information. The latter infers all possible sequences of user choices over all items, which may suffer from intractable computation problems where the number of items is large [7]. Afterward, recurrent neural networks (RNNs) [8] exhibited an overwhelming advantage in modeling sequential data since the data is usually temporally dependent. However, these RNNs-based models ignore the coherence of items. There may be no such strict chronological order among user behaviors. For example, as shown in **Figure 1**, if a user intends to buy ingredients for a cake, whether to purchase flour first or eggs later would not affect user preference. Instead, strictly and solely modeling the relative orders of items and ignoring the coherence of items would probably make the recommendation models prone to overfitting [9]. Recently, graph neural networks (GNNs) have effectively represented both item consistency and sequential dependency. Unfortunately, they suffer from performance degradation when dealing with complex and long sessions, where it is difficult to understand user intent [10].

A critical issue is how user intent is revealed in such session-based recommendation approaches. The individual items can reveal user intent, but only provide limited evidence. To illustrate, consider the example in **Figure 1**. The same strawberry can be viewed differently, i.e., as an option for fruits in Session 1, or as part of the ingredients for fruit cake in Session 2. However, if we independently consider strawberry, it might be viewed as exactly the same item across sessions. In a sense, the meaning of an item (and what it reveals about user intent) could be inferred from contextual sessions, each of which contains a set of consecutive items showing up together within a session.

Hence, we suggest modeling session-wise item representations that can robustly capture user intent with the only limited evidence available in short sessions. However, there are two key challenges in eliciting the user intent signal among items: (1) **How do build a relationship beyond pairwise connections in a session?** Conventional GNNs are designed to model the pairwise connections between items. Obviously, simple graphs cannot depict such set-like relations since we need to consider connecting various numbers of items ranging from two to many. For instance, for Session 1 in **Figure 1**, the pairwise linkage between strawberry and apple is not enough to reveal that the user's intention is to buy a variety of fruits. But such evidence could be inferred by analyzing the triadic relations among strawberries, apples, and pitaya as defined. To this end, we adopt the hypergraph

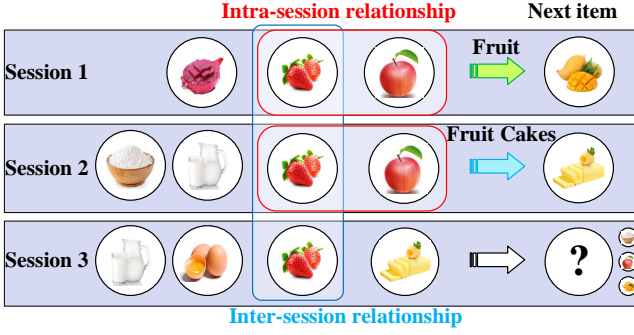


Fig. 1. An example of hidden user intents is revealed by groups of items. Three sessions are given, “Fruit” and “Fruit Cake” are user intents revealed by grouping different consecutive items in the session. Items on the right are possible items the user will click next under each intent.

[11], [12] structure to model the correlations amongst items within each session. Each node denotes an item in a session hypergraph, and a hyperedge connects the collection of items that show up together within a session. This hypergraph structure supports capturing correlations among items defined by sessions, which could be arbitrary order depending on the usage scenario; Meanwhile, The characteristics of hyperedge perfectly fit our assumption as hyperedge is set-like, which emphasizes coherence of the involved items rather than relative orders. (2) **How to associate among sessions?** Figure 1 describes intra-session and inter-session relationships; some items (e.g., apples and strawberries) do not occur within a single session but share their neighboring items for multiple sessions, implying potential item correlations. Technically, we first model each session as a hyperedge in which all the items are connected with each other, and different hyperedges, which are connected via shared items, constitute the hypergraph that contains the item-level high-order correlations. We can borrow the strengths of hypergraph convolution to generate high-quality recommendation results.

To address the above challenges, we present a dual-view collaboration graph neural network (**DC-GNN**) to exploit item transitions over sessions in a more subtle manner by modeling session-based data as a hypergraph for better inferring the user preference of the current session. Intuitively, we propose to learn two levels of item embeddings in our network, which can be seen as two different views that describe the intra- and inter- information of sessions, while each of them knows little information about the other. To be concrete: (i) Graph level, which is to learn the session-level item embedding by modeling pairwise item transitions within the current session; and (ii) Hypergraph level, which is to learn the global-level item embedding by modeling beyond pairwise item transitions across all sessions. Furthermore, we introduce hypergraph and innovatively integrate self-supervised learning [13] into our model to enhance hypergraph modeling. By maximizing the mutual information between the session representations learned via the two levels of the items embedding through self-

supervised learning, the model can force the embeddings of views generated from the same sessions instance to be closer to each other, while those from different instances apart.

The main contributions of this work are summarized as follows:

- We present a dual-view collaboration graph neural network (**DC-GNN**) by modeling session-based data as a hypergraph for better inferring the user preference of the current session, which can capture the beyond pairwise relations among items through hypergraph modeling.
- We innovatively integrate a self-supervised task into the training of our network to enhance hypergraph modeling and improve the recommendation task.
- Extensive experiments show that the proposed framework has overwhelming superiority over the state-of-the-art baselines and achieves statistically significant improvements on benchmark datasets.

II. RELATED WORK

In this section, we review three main topics of previous research: the session-based recommendation, hypergraph learning and self-supervised learning.

A. Session-based Recommendation

The initial exploration of SBR mainly focuses on sequence modeling, where nearest neighbors [5], [14] and Markov decision process [6] are the preferred technique at this phase. While early methods are easy to implement, they do not capture the sequential transition of items and only consider the user’s last click. The boom of deep learning methods brings significant performance gain on SBR [8], [15], [16]. Such as GRU4Rec [8], NARM [15], STAMP [16]. Though deep learning methods have made promising results, they are almost incapable of capturing the collective dependencies. GNNs adopt graph structure to model item-transition patterns, relaxing the strict order assumption over items by RNNs [1]. To capture the complex transitions over items within the entire session, SR-GNN [17] was perhaps the first to consider GNN for SBR. Other models [7], [9], [18] improved the performance by considering different aspects of GNN, such as SR-GNN [18], GCE-GNN [7]. Although significant progress has been achieved, previous methods have not considered users’ multiple intents in learning session embeddings. They all fail to capture the complex and higher-order item correlations. Our proposed model employs graphs and hypergraphs to explore the higher-order dependency of sessions and items.

B. Hypergraph Learning

Although the graph neural network approaches have achieved successful results in capturing high-order relations in various tasks, [11], [12], these approaches are only appropriate in pairwise connections, which has limitation in expressing complex structures of data [19]. Recently, constructing hypergraphs to learn the data structure has become a popular approach. A hypergraph is a generalization of a simple graph

in which a hyperedge can connect more than two nodes. Hypergraph provides a natural way to model complex structures of data with high-order relations and has been extensively employed to tackle various problems [20]. Hayashi et al. [21] propose a flexible framework for clustering hypergraph-structured data based on recently proposed random walks utilizing edge-dependent vertex weights. Xue et al. [22] develop an unsupervised DualHGCN that transforms the bipartite multiplex network into two sets of homogeneous hypergraphs, along with intra- and inter-message passing strategies to promote information exchange within and across domains. Due to this flexibility of hypergraphs, some studies have tried to combine the hypergraph with the recommender systems to improve their performances recently. Bu et al. [23] introduce hypergraph learning to music recommender systems, which is the earliest attempt. Li et al. [24] propose a novel architecture named hyperbolic hypergraph representation learning method for sequential recommendation (H^2 SeqRec) with the pre-training phase. Different from their works, our proposed model exploits inter-hyperedge information and designs for session-based scenarios.

C. Graph Contrastive Learning

Self-supervised learning's success is figuring out a way to leverage the tremendous amounts of unlabeled data that becomes available to dig out the representation of general data. Existing graph contrastive learning [12], [25] is a class of self-supervised approaches that train an encoder to measure the divergence in latent space by contrasting samples from a distribution that contains depict statistical dependencies of interest and those that do not [26]. The main idea of graph contrastive learning is to treat each sample as a distinct category and learn how to distinguish them [25]. For instance, [27] proposes Subcon by utilizing the strong correlation between central nodes and their sampled subgraphs to capture regional structure information, which is a novel data augmentation strategy. [11] developed contrastive learning with augmentations to address the challenge of data heterogeneity in graphs. The theoretical analysis sheds light on the reasons behind their success [12]. Objectives used in these methods can be seen as designing different graph augmentation strategies to enhance the graph representation. As for our work, we combine graph contrastive learning with multi-view semi-supervised learning for the recommendation. It gives us clues about applying training to the session-based recommendation.

III. PRELIMINARIES

Notations. Given a set of sessions $\mathcal{S} = \{s^{(1)}, s^{(2)}, \dots, s^{(P)}\}$ over a set of items $\mathcal{I} = \{i_1, i_2, \dots, i_N\}$, where N is the number of items. An arbitrary session $s \in \mathcal{S}$ is represented as a sequence $s = [i_{s,1}, i_{s,2}, \dots, i_{s,k}, \dots, i_{s,m}]$ ordered by timestamps with a length m . Here, $i_{s,k} \in \mathcal{I} (1 \leq k \leq m)$ represents an interacted item of an anonymous user within the session s . For simplicity, we embed each item $i \in \mathcal{I}$ into the same space and let $x_i^l \in \mathbb{R}^{d^{(l)}}$ denote the representation of item i of

dimension $d^{(l)}$ in the l -th layer of a deep neural network. By stacking P sessions, let $\mathbf{X} \in \mathbb{R}^{P \times d^{(l)}}$ denote a session-item interaction matrix, where P is the number of sessions. Each session s is represented by a vector \mathbf{s} .

Problem Formulation. Here we give a formulation of session-based recommendations problem: The goal of personalized session-based recommendation is to predict the next item, namely $i_{s,m+1}$, for any given session s . Specifically, given \mathcal{I} and s , the output of recommendation model is trained to generate a ranked list $y = [y_1, y_2, \dots, y_N]$ of top- K candidate items as the recommended next items, where y_i corresponds to the score of item i and the top- K items ($1 \leq K \leq N$) with highest score in y will be selected as the recommendations.

Hypergraph Definition. A hypergraph [9] is defined as $G_h = (V, E, \mathbf{W})$, which includes a vertex set V containing N unique vertices, a hyper-edge set E containing M hyper-edges. Each hyper-edge contains two or more vertices and is assigned with a weight by \mathbf{W} , and all the weights formulate a diagonal matrix $\mathbf{W} \in \mathbb{R}^{M \times M}$. The hypergraph can be denoted by an incidence matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$, with entries defined as:

$$h_{v\epsilon} = \begin{cases} 1 & \text{if } v \in \epsilon \\ 0 & \text{if } v \notin \epsilon \end{cases} \quad (1)$$

For each vertex and hyper-edge, their degree D_{vv} and $B_{\epsilon\epsilon}$ are respectively defined as $D_{vv} = \sum_{\epsilon=1}^M W_{\epsilon\epsilon} h_{v\epsilon}$; $B_{\epsilon\epsilon} = \sum_{v=1}^N h_{v\epsilon}$. \mathbf{D} and \mathbf{B} are diagonal matrices.

IV. THE PROPOSED METHOD

We propose a novel dual-view collaboration graph neural network (**DC-GNN**) for Session-based Recommendation. DC-GNN aims to exploit both graph-view and hypergraph-view beyond pairwise item transitions for modeling the user preference of the current session. **Figure 2** presents the architecture of DC-GNN, which comprises four main components: 1) Graph-view item representation. It aims to explore the user intent interest flow relation between items. 2) Hypergraph-view item representation. We can learn the intra- and inter-session dependencies between items. 3) Contrastive learning. We employ contrastive learning to maximize the mutual information between the item representations learned via the two views to alleviate the sparsity of session data. 4) Prediction. It models the user preference of the current session by aggregating the learned item representations in both graph-view and global-level. It outputs the predicted probability of candidate items for recommendation. We next present the four components in detail.

A. Graph-view item representation

Graph Construction. We first construct a meaningful graph from all sessions. Given a session $s = [i_{s,1}, i_{s,2}, \dots, i_{s,k}, \dots, i_{s,m}]$, we treat each item $i_{s,k}$ as a node and $(i_{s,k}, i_{s,k+1})$ as an edge that represents a user clicks item $i_{s,k}$ after $i_{s,k+1}$ in the session s . Subsequently, the graph view is educed by aligning all sessions. In other words, any two items $((i_{s,k}$ and $i_{s,k+1})$ which are connected

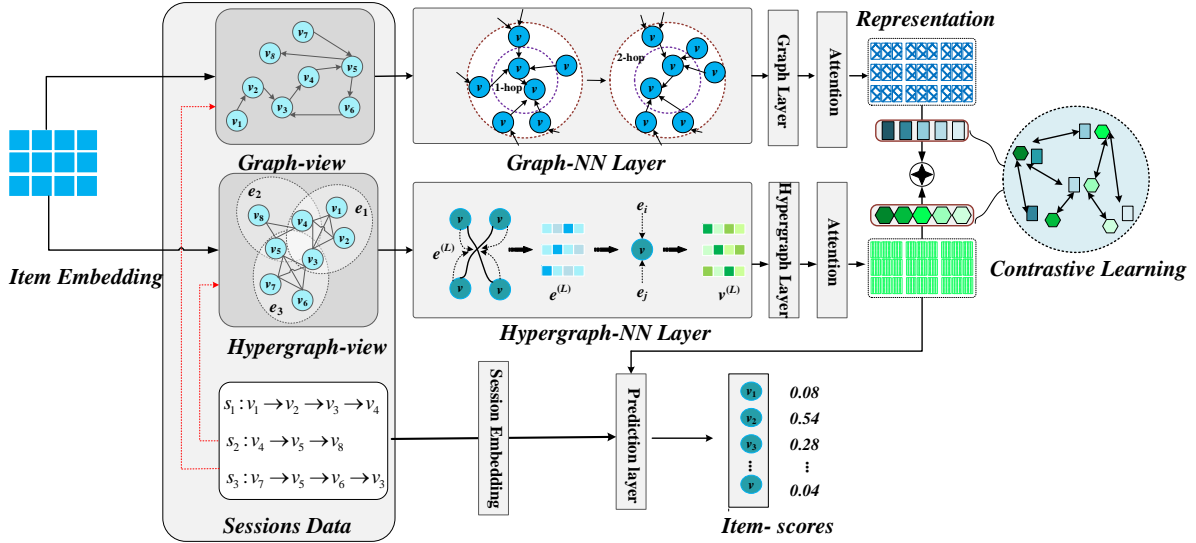


Fig. 2. An overview of the proposed framework. Firstly, DC-GNN learns two levels of item embeddings from graph view and hypergraph view, respectively. (i) graph view, any two items connected in a session also get connected as nodes in the graph view with a weighted edge, counting how many times they are adjacent in different sessions. (ii) hypergraph view transforms each session into a hyperedge, in which any two items clicked in a session are connected to each other. Then we use contrastive learning to recursively leverage the different connectivity information to generate item embeddings from the two views to supervise each other. Finally, we learn the contribution of each item to the next predicted item in the prediction layer. Candidate items will be scored.

in a session also get connected as nodes in the item view with a weighted directed edge, counting how many times they are adjacent in different sessions (shown in the left part of Figure 2). Furthermore, if a node does not contain a self-loop, it will be added with a self-loop with a weight of 1. Based on our observation of our daily life and the datasets, it is common for a user to click two consecutive items a few times within the session.

Graph Neural Network Layer. An item may be involved in multiple sessions, from which we can obtain useful item-transition information to effectively help current predictions. We first convert each item $i \in \mathcal{I}$ into a unified embedding latent space and let $x_i^l \in \mathbb{R}^{d^{(l)}}$ denote the representation of item i of dimension $d^{(l)}$ in the l -th layer of a deep neural network. Then a simplified graph convolution layer for the graph view is defined as:

$$\mathbf{X}_g^{(l+1)} = \sigma(\hat{\mathbf{D}}_g^{-1} \hat{\mathbf{A}}_g \mathbf{X}_g^{(l)} \mathbf{W}_g^{(l)}) \quad (2)$$

where g means in the graph view and $\hat{\mathbf{D}}_{g,j,j} = \sum_{k=1}^m \hat{\mathbf{A}}_{g,j,k}$ is the degree matrix. $\hat{\mathbf{A}}_g = \mathbf{A}_g + \mathbf{I}$, \mathbf{A}_g and \mathbf{I} are the adjacency matrix and the identity matrix, respectively. $\mathbf{W}_g^{(l)}$ denotes the parameter matrix and $\mathbf{X}_g^{(l)}$ represents the l -th layer's item embeddings. $\sigma(\cdot)$ is a nonlinear activation function such as Sigmoid or ReLU. After passing through L graph convolutional layer, we obtained the final item embeddings \mathbf{X}_g for the following process.

B. Hypergraph-view item representation

Hypergraph Construction. A traditional graph can only represent the pairwise relationships between any two nodes.

However, in the real world, there will be much more complex relationships like list-wise relationships than pairwise relationships [28]. To capture the beyond pairwise relations in the session-based recommendation, we adopt a hypergraph $G_h = (V, E, \mathbf{W})$ to denote each session as a hyperedge. For a more rigorous description, we define each item $i_{s,m} \in V$ and each hypergraph as $\epsilon = [i_{s,1}, i_{s,2}, \dots, i_{s,k}, \dots, i_{s,m}] \in E$. We transform the session's data structure into hypergraph construction, as shown in the left part of Figure 2. Compared with traditional graphs that mainly rely on pairwise items as linear sequences, hypergraphs can model higher relations in item interactions. For example, two items $i_{s,k}$ and $i_{s,k+1}$ are connected only if a user interacted with item $i_{s,k}$ before $i_{s,k+1}$. Any two items clicked in a session are connected when we transform the session data into a hyperedge of a hypergraph. It should be noted that items in a session are temporally related instead of sequentially dependent. Besides, if we use a traditional graph, the graph is hard to reveal different item semantics in different sessions directly. Recall back to the previous instance that the strawberries in session 1 and session 2 reveal users' different intentions in Figure 1, while hypergraph is beneficial for dealing with the issue.

Hypergraph Neural Network Layer. After the hypergraph construction, we develop a hypergraph neural network to capture both the item-level high-order relations. The primary challenge of defining a convolution operation over the hypergraph is how the embeddings of items are propagated. Figure 2 illustrates the details of the hypergraph neural networks. Multiple hyperedge structure groups are constructed from the complex correlation of the multi-sessions. We concatenate the hyperedge groups to generate the hypergraph adjacent

matrix \mathbf{H} . The hypergraph adjacent matrix \mathbf{H} and the item embedding are fed into the hypergraph neural network to propagate high-order relations among items. Referring to the spectral hypergraph convolution proposed in [29], we can build a hyperedge convolutional layer in the following formulation:

$$\mathbf{X}_h^{(l+1)} = \sigma(\hat{\mathbf{D}}_h^{-1} \mathbf{H} \mathbf{W} \mathbf{B}^{-1} \mathbf{H}^T \mathbf{X}_h^{(l)} \Theta^{(l)}) \quad (3)$$

where h means in the hypergraph view and $\mathbf{X}_h^{(l)}$ represents the l -th layer's item embeddings. $\sigma(\cdot)$ denotes the nonlinear activation function. $\Theta^{(l)}$ is the learnable filter matrix. Denote that $\hat{\mathbf{D}}_h^{-1}$ play a role of normalization.

Here, we further investigate item embedding in the property of exploiting high-order correlation among data. The hypergraph convolution can be viewed as a two-stage performing 'node-hyperedge-node' feature transformation, which can better refine the features using the hypergraph structure. More specifically, at first, the initial item embedding $\mathbf{X}_h^{(l)}$ is processed by a learnable filter matrix $\Theta^{(l)}$. Then, the item features are gathered according to the hyperedge to form the hyperedge feature, which is implemented by the multiplication of \mathbf{H}^T . Finally, the output node feature is obtained by aggregating their related hyperedge feature, which is achieved by multiplying matrix \mathbf{H} . Thus, the hypergraph layer can efficiently propagate the high-order correlation on the hypergraph by the node-edge-node transform. After passing through the L hypergraph convolutional layer, we obtained the final item embeddings \mathbf{X}_h for the following process.

C. Contrastive learning

As session-based data can be modeled as graphs, there also has been a proliferation of GNNs-based models for SBR with decent improvements. Despite the achievements, however, these approaches are still compromised by the same issue — **data sparsity** [30]. In most cases, these data are too few to induce an accurate user preference, leading to sub-optimal recommendation performance. Contrastive learning (CL) [31], being popular in self-supervised learning, which discovers ground-truth samples from the raw data, is considered to be an antidote to the data sparsity issue. Inspired by the successful practices of self-supervised learning on graphs, we innovatively integrate contrastive learning into the network to enhance hypergraph modeling. We first derive two different views from the session data, i.e., graph view and hypergraph view in the above described, by exploiting sessions' intra- and inter-connectivity patterns. These two views are able to provide complementary information for each other while keeping independent and exhibiting divergence to some degree. The augmented parts differ but inherit essential information from the original data, which can help learn more generalizable representations through a self-supervised task.

To enforce maximizing the consistency between positive pairs $\{\mathbf{X}_g, \mathbf{X}_h\}$ compared with negative pairs, we adopt the noise-contrastive estimation loss [32]. For each mini-batch including n sessions in training, if two-session embeddings both denote the same session in two views, we label this pair as the ground truth $\{\mathbf{X}_g, \mathbf{X}_h\}$, otherwise, we label it as

the negative samples \mathbf{X}_k . then we employ a similarity metric function $\text{sim}(\cdot, \cdot)$ to calculate the similarity of positive pair $\{\mathbf{X}_g, \mathbf{X}_h\}$ and the negative pair $\{\mathbf{X}_h, \mathbf{X}_k\}$. Based on this, the loss function is as follows:

$$\mathcal{L}_{NCE} = -\log \frac{\exp(\text{sim}(\mathbf{X}_g, \mathbf{X}_h)/\tau)}{\exp(\text{sim}(\mathbf{X}_g, \mathbf{X}_h)/\tau) + \exp(\text{sim}(\mathbf{X}_h, \mathbf{X}_k)/\tau)} \quad (4)$$

where τ denotes the temperature parameter. To simplify the calculation, we use dot product as the similarity metric function $\text{sim}(\cdot, \cdot)$. By doing so, they can acquire information from each other to improve their own performance in item feature extraction through the convolution operations. Particularly, those sessions that only include a few items can leverage the cross-session information to refine their embeddings.

D. Prediction

Session embeddings can be represented by the aggregating representation of items in that session. We follow the strategy used in SR-GNN [17] to refine the embedding of session s . Consider information in these embedding may have different levels of priority, we further adopt the soft-attention mechanism to better represent the The embedding of t -th item in session $s = [i_{s,1}, i_{s,2}, \dots, i_{s,k}, \dots, i_{s,m}]$:

$$\alpha_t = \mathbf{q}^T \sigma(\mathbf{W}_1 \mathbf{x}_s^* + \mathbf{W}_2 \mathbf{x}_t^* + \mathbf{c}), \theta_h = \sum_{t=1}^m \alpha_t \mathbf{x}_t \quad (5)$$

$$\mathbf{x}_t^* = \tanh(\mathbf{W}_3 \mathbf{x}_t + \mathbf{b}), \mathbf{x}_s^* = \frac{1}{m} \sum_{t=1}^m \mathbf{x}_m \quad (6)$$

where m is the length of the current session. $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\} \in \mathbb{R}^{d \times d}$, $\{\mathbf{b}, \mathbf{c}\} \in \mathbb{R}^d$ are attention parameters. $\mathbf{q} \in \mathbb{R}^d$ is the attention parameter used to learn the item weight α_t . \mathbf{x}_t^* is the embedding of the t -th item in session s , and \mathbf{x}_s^* is the embedding of session s and here it is represented by averaging the embeddings of items it contains. User's general interest embedding θ_h across this session is represented by aggregating item embeddings through a soft-attention mechanism where items have different levels of priorities.

After obtained the embedding of each session, we compute the score $\hat{\mathbf{z}}_i$ for each candidate item $i \in \mathcal{I}$ by doing inner product between the item embedding \mathbf{X}_h learned from hypergraph channel and θ_h :

$$\hat{\mathbf{z}}_i = \theta_h^T \mathbf{x}_i \quad (7)$$

After that, a softmax function is applied to compute the probabilities of each item being the next one in the session:

$$\hat{\mathbf{y}} = \text{softmax}(\hat{\mathbf{z}}) \quad (8)$$

For each session graph, the loss function is defined as the cross-entropy of the prediction and the ground truth. It can be written:

$$\mathcal{L}_t = - \sum_{i=1}^N \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i) \quad (9)$$

where \mathbf{y} is the one-hot encoding vector of the ground truth.

TABLE I
STATISTICS OF THE DATASETS USED IN OUR EXPERIMENTS.

Statistics	Tmall	Nowplaying	RetailRocket	Diginetica	Yoochoose1/64	Yoochoose1/4
# Sessions (Training)	351,268	825,304	433,643	719,470	369,859	5,917,745
# Sessions (Testing)	25,898	89,824	15,132	60,858	55,898	55,898
# Items	40,728	60,417	36,968	43,097	16,766	29,618
Avg. Length of Sessions	6.69	7.42	5.43	5.12	6.16	5.71

Finally, we unify the recommendation task and this self-supervised task into a joint learning framework and combine Eq.(5) and Eq.(10) to get the total loss function:

$$\mathcal{L}_{total} = \mathcal{L}_t + \alpha \mathcal{L}_{NCE} \quad (10)$$

where $\alpha \in [0, 1]$ is initialized as 0 and gradually learns to assign more weight, which controls the magnitude of the self-supervised task.

V. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the proposed DC-GNN. The experiments are unfolded by answering the following four key research questions:

- **RQ1:** Does DC-GNN outperform state-of-the-art SBR baselines in real world datasets? (See Section 5.2)
- **RQ2:** How do different components of our model, such as the hypergraph convolution or graph convolution mechanism, affect the performance? (See Section 5.3)
- **RQ3:** Is the proposed model sensitive to hyperparameters? How do different hyper-parameter settings (e.g., number of hypergraph convolutional layers) affect the DC-GNN’s accuracy? (See Section 5.4)
- **RQ4:** How well does DC-GNN deal with different length of the sessions? (See Section 5.5)

A. Experimental Protocol

Datasets. We employ five real-world benchmark datasets, namely, *Tmall*¹, *Nowplaying*², *RetailRocket*³, *Diginetica*⁴, *Yoochoose*⁵, which are often used in session-based recommendation methods. Particularly, *Tmall* dataset comes from IJCAI-15 competition, which contains anonymized user’s shopping logs on Tmall online shopping platform. *Nowplaying* dataset comes from [33], which describes the music listening behavior of users. *RetailRocket* is a dataset on a Kaggle contest published by an e-commerce company, which contains the user’s browsing activity within six months. *Diginetica* dataset contains sessions of product transaction data from an online retailer and was released as part of the 2016 CIKM Cup. *Yoochoose* dataset is obtained from the RecSys Challenge 2015, which contains a stream of user clicks on an e-commerce website within 6 months.

Data Preprocessing. For fair comparison, following [7], [17], [34], we conduct preprocessing step over the five datasets. More specifically, sessions of length 1 and items appearing less than 5 times were filtered across all the five datasets. Latest data (such as, the data of last week) is set to be test set and previous data is used as training set [7]. Furthermore, similar to [9], we generate sequences and corresponding labels by a sequence splitting preprocessing, i.e., $([i_{s,1}], i_{s,2}), ([i_{s,1}, i_{s,2}], i_{s,3}), \dots, ([i_{s,1}, i_{s,2}, \dots, i_{s,m-1}], i_{s,m})$, for every session $s = [i_{s,1}, i_{s,2}, i_{s,3}, \dots, i_{s,m}]$, where the label of each sequence is the last click item in it. However, because the Yoochoose training set is quite large and training on the recent fractions yields better results than training on the entire fractions as per the experiments of [35]. As in [7], [17], [34], we sort all of the training sequences generated from Yoochoose, and retrieve the most recent 1/64 and 1/4 to be the training samples in Yoochoose1/64 and Yoochoose1/4. In addition, the training samples and testing samples in all of the five datasets are exactly the same as in [7], [9], [17], [34]. The statistics of the six datasets are presented in **Table I**, where dataset Yoochoose1 is sampled into Yoochoose1/64 and Yoochoose1/4.

Baseline Methods. The following models, including the state-of-art and closely related works, are used as representative baselines to evaluate the performance of the proposed model. They are **Item-KNN** [5], **FPMC** [6], **GRU4Rec**⁶ [8], **NARM**⁷ [15], **STAMP**⁸ [16], **SR-GNN**⁹ [17], **FGNN**¹⁰ [36], **GCE-GNN** [7], **S²-DHCN**¹¹ [9].

Evaluation Metrics. As recommender systems can only recommend a few items at once, the actual item a user might pick should be amongst the first few items of the list [8]. To keep the same setting as previous baselines, we adopt two widely used ranking based metrics: **P@K** and **MRR@K** by following previous work [7], [9], [36]. Specifically, we mainly choose to use top-10 and top-20 items to evaluate a recommender system.

Hyper-parameters Setup. Following previous methods [7], [9], [36], the dimension of the embedding size is fixed to 100, and the batch size for mini-batch is set to 100 for all models. We also set the L_2 regularization to 10^{-5} and keep

¹<https://tianchi.aliyun.com/dataset/dataDetail?dataId=42>

²<http://dbis-nowplaying.uibk.ac.at/#nowplaying>

³<https://www.kaggle.com/retailrocket/e-commerce-dataset>

⁴<https://competitions.codalab.org/competitions/11161>

⁵<http://2015.recsyschallenge.com/challenge.html>

⁶<https://github.com/hidasib/GRU4Rec>

⁷https://github.com/lijiangsdu/sessionRec_NARM

⁸<https://github.com/uestcnlp/STAMP>

⁹<https://github.com/CRIPAC-DIG/SR-GNN>

¹⁰<https://github.com/RuihongQiu/FGNN>

¹¹<https://github.com/xiaxin1998/DHCN>

TABLE II

COMPARISON OF DIFFERENT MODELS, THEIR RESULTS ARE OBTAINED FROM THE CORRESPONDING ORIGINAL PAPERS. ALL THE RESULTS ARE IN PERCENTAGE (%). THE BEST PERFORMING METHOD IN EACH COLUMN IS **BOLDFACED**, AND THE SECOND-BEST METHOD IS MARKED WITH †.

Methods		Tmall				Nowplaying				RetailRocket			
		P@10	MRR@10	P@20	MRR@20	P@10	MRR@10	P@20	MRR@20	P@10	MRR@10	P@20	MRR@20
Traditional	Item-KNN FPMC	6.65	3.11	9.15	3.31	10.96	4.55	15.94	4.91	-	-	-	-
		13.10	7.12	16.06	7.32	5.28	2.68	7.36	2.82	25.99	13.38	32.37	13.82
RNNs	GRU4Rec NARM STAMP	9.47	5.78	10.93	5.89	6.74	4.40	7.92	4.48	38.35	23.27	44.01	23.67
		19.17	10.42	23.30	10.70	13.60	6.62	18.59	6.93	42.07	24.88	50.22	24.59
		22.63	13.12	26.47	13.36	13.22	6.57	17.66	6.88	42.95	24.61	50.96	25.17
GNNs	SR-GNN	23.41	13.45	27.57	13.72	14.17	7.15	18.87	7.47	43.21	26.07	50.32	26.57
	FGNN	20.67	10.07	25.24	10.39	13.89	6.80	18.78	7.15	42.56	26.24	49.86	25.88
	GCE-GNN	28.01†	15.08†	33.42†	15.42†	16.94	8.03†	22.37	8.40†	-	-	-	-
	S^2 -DHCN	26.22	14.60	31.42	15.05	17.35†	7.87	23.50†	8.18	46.15†	26.85†	53.66†	27.30†
	Ours	32.01	18.22	38.54	18.78	17.89	8.23	24.12	8.89	48.85	30.32	56.58	30.05

TABLE III

COMPARISON OF DIFFERENT MODELS, THEIR RESULTS ARE OBTAINED FROM THE CORRESPONDING ORIGINAL PAPERS. ALL THE RESULTS ARE IN PERCENTAGE (%). THE BEST PERFORMING METHOD IN EACH COLUMN IS **BOLDFACED**, AND THE SECOND-BEST METHOD IS MARKED WITH †.

Methods		Diginetica				Yoochoose 1/64				Yoochoose 1/4			
		P@10	MRR@10	P@20	MRR@20	P@10	MRR@10	P@20	MRR@20	P@10	MRR@10	P@20	MRR@20
Traditional	Item-KNN FPMC	25.07	10.77	35.75	11.57	-	-	51.60	21.81	-	-	52.31	21.70
		15.43	6.20	26.53	6.95	-	-	45.62	15.01	-	-	51.86	17.50
RNNs	GRU4Rec NARM STAMP	17.93	7.33	29.45	8.33	50.04	22.64	60.64	22.89	49.68	22.84	59.53	22.60
		35.44	15.13	49.70	16.17	57.50	27.97	68.32	28.63	57.83	28.10	69.73	29.23
		33.98	14.26	45.64	14.32	58.07	28.92	68.74	29.67	59.62	29.24	70.44	30.00
GNNs	SR-GNN	36.86	15.52	50.73	17.59	60.21	30.13	70.57	30.94	61.28†	30.65†	71.36	31.89
	FGNN	37.72	15.95	50.58	16.84	60.97†	30.85	71.12†	31.68	61.25	30.48	71.97†	32.54†
	GCE-GNN	41.16†	18.15†	54.22†	19.04†	59.68	30.95†	70.58	31.12	59.66	30.12	69.28	30.35
	S^2 -DHCN	40.21	17.59	53.66	18.51	59.38	29.17	69.87	29.86	60.54	29.05	69.95	29.85
	Ours	42.28	18.91	55.03	19.67	61.52	31.56	71.85	31.59†	61.72	31.73	72.38	32.61

the hyper-parameters of each model consistent for a fair comparison. In our model, all weighting matrices are initialized by sampling from a normal distribution $N(0, 0.05^2)$, and all biases are set to zeros. All the item embeddings are initialized randomly with the Gaussian Distribution $N(0, 0.1)$, which are then jointly trained with other parameters. We use Adam optimizer with the initial learning rate 0.001, which will decay by 0.1 after every 3 epochs. In addition, the number of layers is different in different datasets. For *Diginetica*, a two-layer setting is the best, while for *Nowplaying*, a four-layer setting achieves the best performance. We train each model for 30 epochs or until the loss no longer decreases after 5 epochs.

B. Overall Comparison (RQ1)

To demonstrate the performance of the proposed model, we compare it with the state-of-the-art item recommendation approaches, illustrated in **Table II** and **Table III**. We have the following observations: **1)** Overall, from the two tables, we can see that our proposed model consistently shows strong performance across all datasets in terms of the two metrics (with $K=10$ and 20), which ascertains our proposed method's effectiveness. Especially on the Tmall dataset, we note that existing baselines have already obtained high enough performance; our method still pushes that boundary forward. **2)** Compared with traditional and RNN methods, our model

has achieved remarkable performance. Those demonstrate that our proposed model converts the sequential item transitions into graph-structured data for capturing the inherent order of item-transition patterns. This strategy performs strong data representation in constructing relations dependency between sessions, leading to better performance. **3)** We also notice that our method achieves more competitive results compared with graph-based baselines. Particularly, it beats other models by a large margin on Tmall, showing the effectiveness of the self-supervised graph and hypergraph co-training when applied to real e-commerce data. **4)** Considering that S^2 -DHCN and our method both have a two-branch architecture, we think that the improvements mainly derive from the different contrastive learning. We employ two graph structure types (i.e., graph and hypergraph) to encode the item embedding for contrastive learning. While S^2 -DHCN uses the two types of hypergraph to conduct sessions embedding for contrastive learning. Compared with another strong baseline GCE-GNN, DC-GNN is competitive in performance and efficiency. DC-GNN outperforms the GCE-GNN by 4% on Tmall and 1-2% on other datasets on average. Besides, our DC-GNN is more lightweight than GCE-GNN because GCE-GNN with a more complex structure makes it suffer from the out-of-memory problem when performing on RetailRocket on our RTX 3080 GPU. At the same time, we use very limited hypergraph

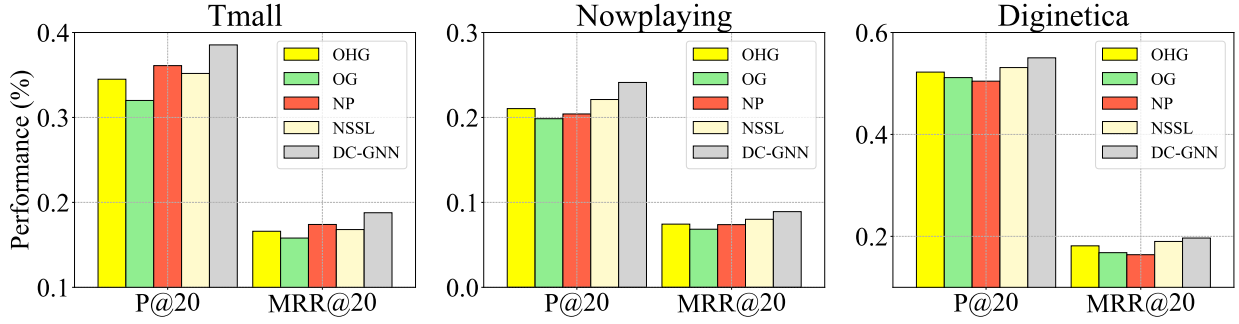


Fig. 3. Ablation Study.

convolution and graph-convolution parameters, showing the efficiency of DC-GNN.

C. Study Ablation (RQ2)

In this section, we conduct experiments on three datasets to investigate the contribution of each component in our model. Specially, we design four variant versions of DC-GNN:

- **OHG**: We only use the hypergraph view to encode item embedding to model session data, removing the graph view and the self-supervised contrastive learning.
- **OG**: We only use the graph view to encode item embedding to model session data, removing the hypergraph view and the self-supervised contrastive learning.
- **NP**: We remove the reversed position embeddings.
- **NSSL**: We remove the self-supervised contrastive learning and replace it with averaging item representations of the two views.

Due to the space limitation, we choose Tmall, Nowplaying, and Diginetica datasets as examples and conducted experiments. To compare them under different conditions, we report their performance under two metrics (with $K=20$). We show the results of these four variants in **Figure 3**. From Figure 3, we can observe that each component consistently contributes to three datasets. (1) The self-supervised contrastive learning improves the base model the most, serving as the driving force of performance improvement. When removing the self-supervised contrastive learning, we can observe a remarkable performance drop on both two metrics. (2) Furthermore, OHG can achieve better performance than OG, which shows the effectiveness of the hypergraph structure and the necessity of modeling the contextual relations for item representation learning. (3) Besides, the two views are effective to achieve better performance than the single view in three datasets, which indicates that two-view can cooperate and have mutual complementarity. (4) According to the results of NP, it is shown that learning different item importance across sessions is better than directly averaging representations of contained items for learning session representations in the session-based recommendation. (5) Overall, DC-GNN can outperform all of its variants for $K=20$ values, indicating the effectiveness of its design for the session-based recommendation.

D. Hyper-Parameter Analysis. (RQ3)

In this section, we present the sensitivity analysis of two critical hyper-parameters. The DC-GNN model employs a hyper-parameter α to control the magnitude of self-supervised learning and hypergraph convolutional network's depth N in the session-based recommendation. For brevity, we conducted the experiments on two datasets: Nowplaying and Diginetica. Additionally, the default parameter settings of the analysis are $\alpha=0.05$ and $N=3$. To be strict, when we tested the hyper-parameters specific to DC-GNN, the other parameters were set to the default settings. Finally, to compare them under different conditions, we report their performance under two metrics (with $K=20$).

- **Impact of Self-Supervised Learning.** In our approach, we can control the magnitude of self-supervised learning by changing the weight term α . We respectively tune the factor in a range of $\{0.01, 0.05, 0.1, 0.5, 1\}$, and the results are illustrated in **Figure 4**. According to the results presented in **Figure 4**, recommendation task achieves decent gains when jointly optimized with the self-supervised task. With the rise of α , the performance increases first and then declines. We think it is due to the gradient conflicts between the two tasks. Besides, when $\alpha = 0.1$, we get the best performance.
- **Impact of the number of layers.** We further explore the sensitivity of the hypergraph convolutional network's depth in graph view channels. We stack multiple Hypergraph layers to model the information flowing among items with high-order dependencies. To visualize the high-order information propagation, we show the results in **Figure 3** by ranging the number of layers of the network within $\{1, 2, 3, 4, 5\}$. In each dataset, starting with a single hypergraph layer, we improve the performance by stacking one more layer, indicating the importance of modeling the information via high-order connections. For the Diginetica dataset, stacking more than three layers will worsen the performance since the sessions in this dataset are generally short. Using too many layers will bring in noise for the representation learning process. Meanwhile, our model gets the best performance in the Nowplaying dataset when the number of layers $N=4$. Thus, we conclude that DC-GNN can capture the direct and high-order connections among items in the hypergraphs.

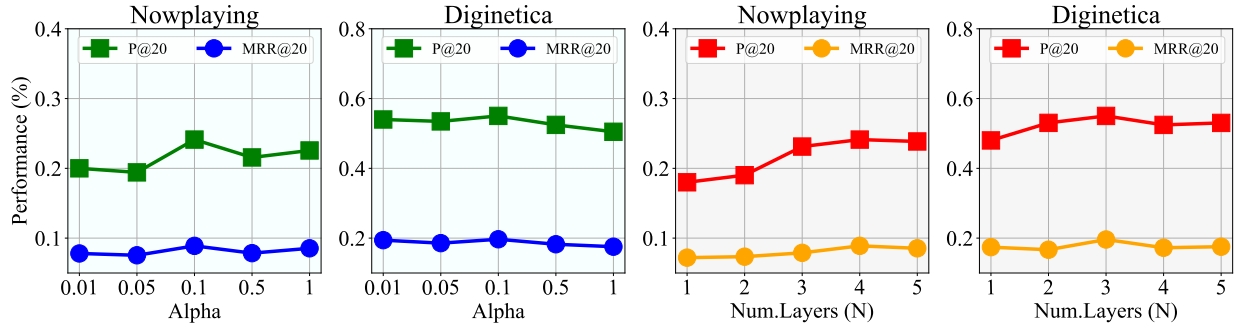


Fig. 4. Hyperparameter Analysis

TABLE IV
THE PERFORMANCE OF DIFFERENT METHODS WITH DIFFERENT SESSION LENGTHS EVALUATED IN TERMS OF P@20. ALL THE RESULTS ARE IN PERCENTAGE (%). THE BEST PERFORMING METHOD IN EACH COLUMN IS **BOLDFACED**.

Methods	Tmall		Diginetica		Yoochoose 1/64	
	Short Session	Long Session	Short Session	Long Session	Short Session	Long Session
SR-GNN	38.62	28.14	50.49	51.27	71.44	60.73
FGNN	39.13	30.57	51.26	52.42	71.21	70.80
GCE-GNN	42.28	34.22	54.40	52.16	70.42	69.68
DHCN	36.47	31.73	53.29	52.43	70.81	68.54
Ours	44.81	35.14	56.21	54.14	72.54	69.77

E. Analysis on Session Lengths. (RQ4)

In real-world situations, sessions of various lengths are common. How to infer user interest based on different sessions lengths is an important indicator to measure the performance of the model, and it is also a critical indicator for production environments. We further analyze the capability of different models to cope with sessions of different lengths. To evaluate this, we follow the works [17], [36], and split sessions of Tmall, Diginetica, and Yoochoose1/64 datasets into two groups, where “Short” indicates that the length of sessions is less than or equal to 5, while each session has more than 5 items in “Long.” For each graph-based method, we report the results evaluated in terms of P@20 in **Table IV**. In the aspect of both Short and Long sessions, DC-GNN achieves almost the best performance compared with other graph embedding generators. It demonstrates the adaptability of our model in real-world session-based recommendations. However, since the percentage of long sessions is low in real-world datasets for a session-based recommendation, the main purpose is to boost the recommendation for sessions with fewer items. Besides, it also demonstrates the performance of our model in the short sessions is better than that in the long sessions.

VI. CONCLUSION

In this paper, we present a dual-view collaboration graph neural network (**DC-GNN**) by modeling session-based data as a hypergraph for better inferring the user preference of the current session which can capture the ubiquitous high-order correlations among items. Moreover, to further enhance the network, we innovatively integrate self-supervised into

the network’s training. Extensive empirical studies demonstrate the superiority of the proposed model over the current SOTA methods, and the results validate the effectiveness of hypergraph modeling. Meanwhile, the research of hypergraph modeling for SBR remains in its infancy, and their application in GCNs has potential development, which is worthy of our further exploration.

REFERENCES

- [1] L. Feng, Y. Cai, E. Wei, and J. Li, “for session-based recommendation,” *Neurocomputing*, vol. 472, pp. 113–123, 2022.
- [2] A. Li, Z. Cheng, F. Liu, Z. Gao, W. Guan, and Y. Peng, “Disentangled graph neural networks for session-based recommendation,” *CoRR*, vol. abs/2201.03482, 2022.
- [3] R. Qiu, Z. Huang, T. Chen, and H. Yin, “Exploiting positional information for session-based recommendation,” *ACM Trans. Inf. Syst.*, vol. 40, no. 2, pp. 35:1–35:24, 2022.
- [4] J. Guo, Y. Yang, X. Song, Y. Zhang, Y. Wang, J. Bai, and Y. Zhang, “Learning multi-granularity consecutive user intent unit for session-based recommendation,” in *WSDM ’22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, K. S. Candan, H. Liu, L. Akoglu, X. L. Dong, and J. Tang, Eds. ACM, 2022, pp. 343–352.
- [5] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, Eds. ACM, 2001, pp. 285–295.
- [6] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, “Factorizing personalized markov chains for next-basket recommendation,” in *WWW, M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, Eds.* ACM, 2010, pp. 811–820.
- [7] Z. Wang, W. Wei, G. Cong, X. Li, X. Mao, and M. Qiu, “Global context enhanced graph neural networks for session-based recommendation,” in *SIGIR, J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and Y. Liu, Eds.* ACM, 2020, pp. 169–178.

- [8] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2016.
- [9] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," in *AAAI*. AAAI Press, 2021, pp. 4503–4511.
- [10] M. Choi, J. Kim, J. Lee, H. Shim, and J. Lee, "S-walk: Accurate and scalable session-based recommendation with random walks," in *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, K. S. Candan, H. Liu, L. Akoglu, X. L. Dong, and J. Tang, Eds. ACM, 2022, pp. 150–160.
- [11] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.
- [12] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds. ACM / IW3C2, 2021, pp. 2069–2080.
- [13] G. Chu, X. Wang, C. Shi, and X. Jiang, "Cucu: Graph representation with curriculum contrastive learning," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, Z. Zhou, Ed. ijcai.org, 2021, pp. 2300–2306.
- [14] R. Dias and M. J. Fonseca, "Improving music recommendation in session-based collaborative filtering by using temporal context," in *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2013, Herndon, VA, USA, November 4-6, 2013*. IEEE Computer Society, 2013, pp. 783–788.
- [15] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, E. Lim, M. Winslett, M. Sanderson, A. W. Fu, J. Sun, J. S. Culpepper, E. Lo, J. C. Ho, D. Donato, R. Agrawal, Y. Zheng, C. Castillo, A. Sun, V. S. Tseng, and C. Li, Eds. ACM, 2017, pp. 1419–1428.
- [16] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP: short-term attention/memory priority model for session-based recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Y. Guo and F. Farooq, Eds. ACM, 2018, pp. 1831–1839.
- [17] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 346–353.
- [18] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, "Graph contextualized self-attention network for session-based recommendation," in *IJCAI*, vol. 19, 2019, pp. 3940–3946.
- [19] J. Zhang, M. Gao, J. Yu, L. Guo, J. Li, and H. Yin, "Double-scale self-supervised hypergraph learning for group recommendation," in *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, and H. Tong, Eds. ACM, 2021, pp. 2557–2567.
- [20] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang, "Self-supervised multi-channel hypergraph convolutional network for social recommendation," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds. ACM / IW3C2, 2021, pp. 413–424.
- [21] K. Hayashi, S. G. Aksoy, C. H. Park, and H. Park, "Hypergraph random walks, laplacians, and clustering," in *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, M. d'Aquin, S. Dietze, C. Hauff, E. Curry, and P. Cudré-Mauroux, Eds. ACM, 2020, pp. 495–504.
- [22] H. Xue, L. Yang, V. Rajan, W. Jiang, Y. Wei, and Y. Lin, "Multiplex bipartite network embedding using dual hypergraph convolutional networks," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds. ACM / IW3C2, 2021, pp. 1649–1660.
- [23] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, and X. He, "Music recommendation by unified hypergraph: combining social media information and music content," in *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*, A. D. Bimbo, S. Chang, and A. W. M. Smeulders, Eds. ACM, 2010, pp. 391–400.
- [24] Y. Li, H. Chen, X. Sun, Z. Sun, L. Li, L. Cui, P. S. Yu, and G. Xu, "Hyperbolic hypergraphs for sequential recommendation," in *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, and H. Tong, Eds. ACM, 2021, pp. 988–997.
- [25] G. Fang, J. Song, X. Wang, C. Shen, X. Wang, and M. Song, "Contrastive model inversion for data-free knowledge distillation," *CoRR*, vol. abs/2105.08584, 2021.
- [26] K. Hassani and A. H. K. Ahmadi, "Contrastive multi-view representation learning on graphs," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 4116–4126.
- [27] Y. Jiao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu, "Sub-graph contrast for scalable self-supervised graph representation learning," in *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*, C. Plant, H. Wang, A. Cuzzocrea, C. Zaniolo, and X. Wu, Eds. IEEE, 2020, pp. 222–231.
- [28] N. Wang, S. Wang, Y. Wang, Q. Z. Sheng, and M. A. Orgun, "Exploiting intra- and inter-session dependencies for session-based recommendations," *World Wide Web*, vol. 25, no. 1, pp. 425–443, 2022.
- [29] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 3558–3565.
- [30] X. Xia, H. Yin, J. Yu, Y. Shao, and L. Cui, "Self-supervised graph co-training for session-based recommendation," in *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, and H. Tong, Eds. ACM, 2021, pp. 2180–2190.
- [31] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *CoRR*, vol. abs/2006.08218, 2020.
- [32] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.
- [33] E. Zangerle, M. Pichl, W. Gassler, and G. Specht, "#nowplaying music dataset: Extracting listening behavior from twitter," in *WISMM*, R. Zimmermann and Y. Yu, Eds. ACM, 2014, pp. 21–26.
- [34] J. Wang, K. Ding, Z. Zhu, and J. Caverlee, "Session-based recommendation with hypergraph attention networks," in *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, C. Demeniconi and I. Davidson, Eds. SIAM, 2021, pp. 82–90.
- [35] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, A. Karatzoglou, B. Hidasi, D. Tikk, O. S. Shalom, H. Roitman, B. Shapira, and L. Rokach, Eds. ACM, 2016, pp. 17–22.
- [36] R. Qiu, J. Li, Z. Huang, and H. Yin, "Rethinking the item order in session-based recommendation with graph neural networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, W. Zhu, D. Tao, X. Cheng, P. Cui, E. A. Rundensteiner, D. Carmel, Q. He, and J. X. Yu, Eds. ACM, 2019, pp. 579–588.