Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation

Xin Xia[1], Hongzhi Yin[1*], Junliang Yu[1], Qinyong Wang[1], Lizhen Cui[2], Xiangliang Zhang[3]
[1]The University of Queensland, [2]Shandong University,
[3]King Abdullah University of Science and Technology
xin.xia1@uqconnect.edu.au, {h.yin1, jl.yu, qinyong.wang}@uq.edu.au, clz@sdu.edu.cn, xiangliang.zhang@kaust.edu.sa

Abstract

Session-based recommendation (SBR) focuses on next-item prediction at a certain time point. As user profiles are generally not available in this scenario, capturing the user intent lying in the item transitions plays a pivotal role. Recent graph neural networks (GNNs) based SBR methods regard the item transitions as pairwise relations, which neglect the complex high-order information among items. Hypergraph provides a natural way to capture beyond-pairwise relations, while its potential for SBR has remained unexplored. In this paper, we fill this gap by modeling session-based data as a hypergraph and then propose a dual channel hypergraph convolutional network – DHCN to improve SBR. Moreover, to enhance hypergraph modeling, we innovatively integrate self-supervised learning into the training of our network by maximizing mutual information between the session representations learned via the two channels in DHCN, serving as an auxiliary task to improve the recommendation task. Extensive experiments on three benchmark datasets demonstrate the superiority of our model over the SOTA methods, and the ablation study validates the effectiveness and rationale of hypergraph modeling and self-supervised task. The implementation of our model is available via https://github.com/xiaxin1998/DHCN.

Introduction

Session-based recommendation (SBR) is an emerging recommendation paradigm, where long-term user profiles are usually not available (Wang, Cao, and Wang 2019; Guo et al. 2019). Generally, a session is a transaction with multiple purchased items in one shopping event, and SBR focuses on next-item prediction by using the real-time user behaviors. Most of the research efforts in this area regard the sessions as ordered sequences, among which recurrent neural networks (RNNs) based (Hidasi et al. 2015; Jannach and Ludewig 2017; Hidasi and Karatzoglou 2018) and graph neural networks (GNNs) (Wu et al. 2020) based approaches have shown great performance.

In RNNs-based approaches, modeling session-based data as unidirectional sequences is deemed as the key to success, since the data is usually generated in a short period of time and is likely to be temporally dependent. However, this assumption may also trap these RNNs-based models because it is ill-considered. Actually, unlike linguistic sequences which are generated in a strictly-ordered way, among user behaviors, there may be no such strict chronological order. For example, on Spotify[1], a user can choose to shuffle an album or play it in order, which generates two different listening records. However, both of these two play modes serialize the same set of songs. In other words, reversing the order of two items in this case would not lead to a distortion of user preference. Instead, strictly and solely modeling the relative orders of items would probably make the recommendation models prone to overfitting.

Recently, the effectiveness of graph neural networks (GNNs) (Wu et al. 2020; Yu et al. 2020; Yin et al. 2019) has been reported in many areas including SBR. Unlike the RNNs-based recommendation method, the GNNs-based approaches (Wu et al. 2019b; Xu et al. 2019; Qiu et al. 2020b) model session-based data as directed subgraphs and item transitions as pairwise relations, which slightly relaxes the assumption of temporal dependence between consecutive items. However, existing models only show trivial improvements compared with RNNs-based methods. The potential reason is that they neglect the complex item correlations in session-based data. In real scenarios, an item transition is often triggered by the joint effect of previous item clicks, and many-to-many and high-order relations exist among items. Obviously, simple graphs are incapable of depicting such set-like relations.

To overcome these issues, we propose a novel SBR approach upon hypergraph to model the high-order relations among items within sessions. Conceptually, a hypergraph (Bretto 2013) is composed of a vertex set and a hyperedge set, where a hyperedge can connect any numbers of vertices, which can be used to encode high-order data correlations. We also assume that items in a session are temporally correlated but not strictly sequentially dependent. The characteristics of hyperedge perfectly fit our assumption as hyperedge is set-like, which emphasizes coherence of the involved elements rather than relative orders. Therefore, it provides us with a flexibility and capability to capture complex interactions in sessions. 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

*Corresponding author

[1]https://www.spotify.com/

via shared items, constitute the hypergraph that contains the item-level high-order correlations. Then a line graph is built based on the hypergraph by modeling each hyperedge as a node and focuses on the connectivity of hyperedges, which depicts the session-level relations. After that, a **D**ual channel **H**ypergraph **C**onvolutional **N**etwork (DHCN) is developed to capture the complex item correlations and cross-session information with its two channels from the two graphs, respectively. Figure 1 illustrates the hypergraph construction and the pipeline of the proposed method.

By stacking multiple layers in the two channels, we can borrow the strengths of hypergraph convolution to generate high-quality recommendation results. However, since each hyperedge only contains a limited number of items, the inherent data sparsity issue might limit the benefits brought by hypergraph modeling. To address this problem, we innovatively integrate self-supervised learning (Hjelm et al. 2018) into our model to enhance hypergraph modeling. Intuitively, the two channels in our network can be seen as two different views that describe the intra- and inter- information of sessions, while each of them knows little information of the other. By maximizing the mutual information between the session representations learned via the two channels through self-supervised learning, the two channels can acquire new information from each other to improve their own performance in item/session feature extraction. We then unify the recommendation task and the self-supervised task under a *primary&auxiliary* learning framework. By jointly optimizing the two tasks, the performance of the recommendation task achieves decent gains.

Overall, the main contributions of this work are summarized as follows:

- We propose a novel dual channel hypergraph convolutional network for SBR, which can capture the beyond-pairwise relations among items and the cross-session information 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 our proposed model has overwhelming superiority over the state-of-the-art baselines and achieves statistically significant improvements on benchmark datasets.

Related Work

Session-based Recommendation

The initial exploration of SBR mainly focuses on sequence modeling, where Markov decision process is the preferred technique at this phase. (Shani, Heckerman, and Brafman 2005; Rendle, Freudenthaler, and Schmidt-Thieme 2010; Zimdars, Chickering, and Meek 2013) are the representative works of this line of research. The boom of deep learning provides alternatives to exploit sequential data. Deep learning models such as recurrent neural networks (Hochreiter and Schmidhuber 1997; Cho et al. 2014) and convolutional neural networks (Tuan and Phuong 2017) have subsequently been applied to SBR and achieved great success. (Hidasi et al. 2015; Tan, Xu, and Liu 2016; Li et al. 2017; Liu et al. 2018) are the classical RNNs-based models which borrow the strengths of RNNs to model session-based data.

Graph Neural Networks (GNNs) (Wu et al. 2020; Zhou et al. 2018) recently have drawn increasing attention and their applications in SBR also have shown promising results (Wang et al. 2020b,c; Yuan et al. 2019; Chen and Wong 2020). Unlike RNNs-based approaches working on sequential data, GNNs-based methods learn item transitions over session-induced graphs. SR-GNN (Wu et al. 2019b) is the pioneering work which uses a gated graph neural network to model sessions as graph-structured data. GC-SAN (Xu et al. 2019) employs self-attention mechanism to capture item dependencies via graph information aggregation. FGNN (Qiu et al. 2019) constructs a session graph to learn item transition pattern and rethinks the sequence order of items in SBR. GCE-GNN (Wang et al. 2020c) conduct graph convolution on both the single session graph and the global session graph to learn session-level and global-level embeddings. Although these studies demonstrate that GNN-based models outperform other approaches including RNNs-based ones, they all fail to capture the complex and higher-order item correlations.

Hypergraph Learning

Hypergraph provides a natural way to complex high-order relations. With the boom of deep learning, hypergraph neural network also have received much attention. HGNN (Feng et al. 2019) and HyperGCN (Yadati et al. 2019) are the first to apply graph convolution to hypergraph. (Jiang et al. 2019) proposed a dynamic hypergraph neural network and (Bandyopadhyay, Das, and Murty 2020) developed the line hypergraph convolutional networks.

There are also a few studies combining hypergraph learning with recommender systems (Bu et al. 2010; Li and Li 2013). The most relevant work to ours is HyperRec (Wang et al. 2020a), which uses hypergraph to model the short-term user preference for next-item recommendation. However, it does not exploit inter-hyperedge information and is not designed for session-based scenarios. Besides, the high complexity of this model makes it impossible to be deployed in real scenarios. Currently, there is no research bridging hypergraph neural networks and SBR, and we are the first to fill this gap.

Self-supervised Learning

Self-supervised learning (Hjelm et al. 2018) is an emerging machine learning paradigm which aims to learn the data representation from the raw data. It was firstly used in visual representation learning (Bachman, Hjelm, and Buchwalter 2019). The latest advances in this area extend self-supervised learning to graph representation learning (Velickovic et al. 2019). The dominant paradigm based on contrastive learning (Hassani and Khasahmadi 2020; Qiu et al. 2020a) suggests that contrasting congruent and incongruent views of graphs with mutual information maximization can help encode rich graph/node representations.

As self-supervised learning is still in its infancy, there are only several studies combining it with recommender sys-

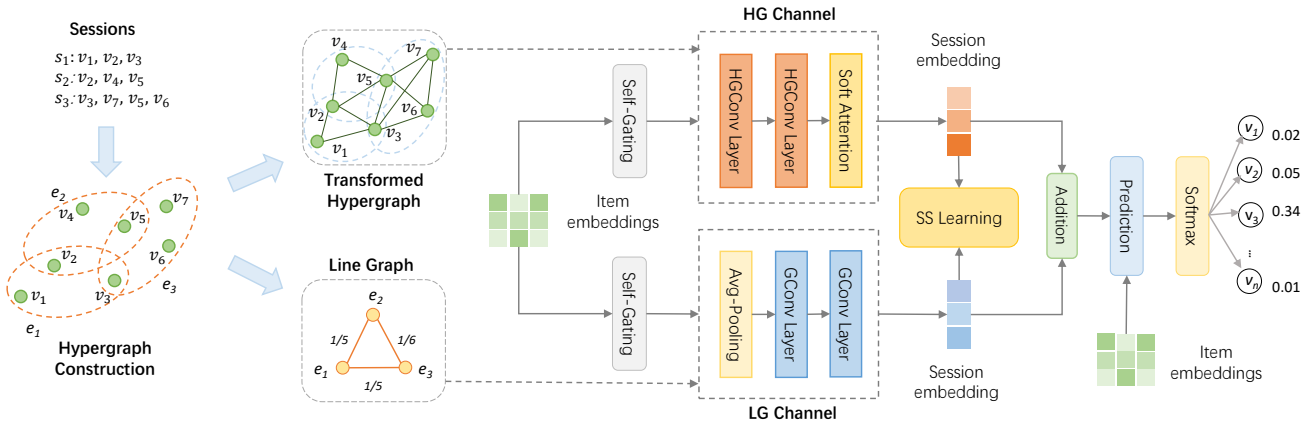

Figure 1: The construction of hypergraph and the pipeline of the proposed DHCN model.

tems (Zhou et al. 2020; Ma et al. 2020; Xin et al. 2020). The most relevant work to ours is S^3-Rec (Zhou et al. 2020) for sequential recommendation, which uses feature mask to create self-supervision signals. But it is not applicable to SBR since the session data is very sparse and masking features cannot generate strong self-supervision signals. Currently, the potentials of self-supervised learning for hypergraph representation learning and SBR have not been investigated. We are the first to integrate self-supervised learning into the scenarios of SBR and hypergraph modeling.

The Proposed Method

In this section, we first introduce the notions and definitions used throughout this paper, and then we show how session-based data is modeled as a hypergraph. After that, we present our dual channel hypergraph convolutional network for SBR. Finally, we integrate self-supervised learning into the network to enhance hypergraph modeling.

Notations and Definitions

Let $I = \{i_1, i_2, i_3, ..., i_N\}$ denote the set of items, where N is the number of items. Each session is represented as a set $s = [i_{s,1}, i_{s,2}, i_{s,3}, ..., i_{s,m}]$ and $i_{s,k} \in I (1 \leq k \leq m)$ represents an interacted item of an anonymous user within the session s. We embed each item $i \in I$ into the same space and let $\mathbf{x}_i^{(l)} \in \mathbb{R}^{d^{(l)}}$ denote the vector representation of item i of dimension d^l in the l-th layer of a deep neural network. The representation of the whole item set is denoted as $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times d^{(l)}}$. Each session s is represented by a vector $\mathbf{s}$. The task of SBR is to predict the next item, namely $i_{s,m+1}$, for any given session s.

Definition 1. Hypergraph. Let $G = (V, E)$ denote a hypergraph, where V is a set containing N unique vertices and E is a set containing M hyperedges. Each hyperedge $\epsilon \in E$ contains two or more vertices and is assigned a positive weight $W_{\epsilon\epsilon}$, and all the weights formulate a diagonal matrix $\mathbf{W} \in \mathbb{R}^{M \times M}$. The hypergraph can be represented by an incidence matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$ where $H_{i\epsilon} = 1$ if the hyperedge $\epsilon \in E$ contains a vertex $v_i \in V$, otherwise 0. For each vertex and hyperedge, their degree D_{ii} and $B_{\epsilon\epsilon}$ are respectively defined as $D_{ii} = \sum_{\epsilon=1}^{M} W_{\epsilon\epsilon} H_{i\epsilon}$; $B_{\epsilon\epsilon} = \sum_{i=1}^{N} H_{i\epsilon}$. $\mathbf{D}$ and $\mathbf{B}$ are diagonal matrices.

Definition 2. Line graph of hypergraph. Given the hypergraph $G = (V, E)$, the line graph of the hypergraph $L(G)$ is a graph where each node of $L(G)$ is a hyperedge in G and two nodes of $L(G)$ are connected if their corresponding hyperedges in G share at least one common node (Whitney 1992). Formally, $L(G) = (V_L, E_L)$ where $V_L = \{v_e : v_e \in E\}$, and $E_L = \{(v_{e_p}, v_{e_q}) : e_p, e_q \in E, |e_p \cap e_q| \geq 1\}$. We assign each edge (v_{e_p}, v_{e_q}) a weight $W_{p,q}$, where $W_{p,q} = |e_p \cap e_q| / |e_p \cup e_q|$.

Hypergraph Construction.

To capture the beyond pairwise relations in session-based recommendation, we adopt a hypergraph $G = (V, E)$ to represent each session as a hyperedge. Formally, we denote each hyperedge as $[i_{s,1}, i_{s,2}, i_{s,3}, ..., i_{s,m}] \in E$ and each item $i_{s,m} \in V$. The changes of data structure before and after hypergraph construction are shown in the left part of Figure 1. As illustrated, the original session data is organized as linear sequences where two items $i_{s,m-1}, i_{s,m}$ are connected only if a user interacted with item $i_{s,m-1}$ before item $i_{s,m}$. After transforming the session data into a hypergraph, any two items clicked in a session are connected. It should be noted that we transform the session sequences into an undirected graph, which is in line with our intuition that items in a session are temporally related instead of sequentially dependent. By doing so, we manage to concretize the many-to-many high-order relations. Besides, we further induce the line graph of the hypergraph according to Definition 2. Each session is modeled as a node and and different sessions are connected via shared items. Compared with the hypergraph which depicts the item-level high-order relations, the line graph describes the session-level relations that are also termed cross-session information.

Dual Channel Hypergraph Convolutional Network

After the hypergraph construction, we develop a **D**ual channel **H**ypergraph **C**onvolutional **N**etwork, DHCN, to capture both the item-level high-order relations and the session-level relations. Each channel in the network is in charge of extracting useful information from one (hyper)graph to improve SBR via (hyper)graph convolution.

Hypergraph Channel and Convolution. The hypergraph channel encodes the hypergraph. As there are two channels, directly feeding the full base item embeddings $\mathbf{X}^{(0)} \in \mathbb{R}^{N\times d}$ to two channels is unwise. To control the magnitude of embedding flowing to each channel, we employ a pre-filter with *self-gating units* (SGUs), which is defined as:

$$\mathbf{X}_c^{(0)} = f_{\text{gate}}^c(\mathbf{X}^{(0)}) = \mathbf{X}^{(0)} \odot \sigma(\mathbf{X}^{(0)}\mathbf{W}_g^c + \mathbf{b}_g^c), \quad (1)$$

where $\mathbf{W}_g^c \in \mathbb{R}^{d\times d}, \mathbf{b}_g^c \in \mathbb{R}^d$ are gating parameters to be learned, $c \in \{h, l\}$ represents the channel, $\odot$ denotes the element-wise product and σ is the sigmoid function. The self-gating mechanism modulates the base item embeddings at a feature-wise granularity through dimension re-weighting, then we obtain the hypergraph channel-specific item embeddings $\mathbf{X}_h^{(0)}$.

The primary challenge of defining a convolution operation over the hypergraph is how the embeddings of items are propagated. Referring to the spectral hypergraph convolution proposed in (Feng et al. 2019), we define our hypergraph convolution as:

$$\mathbf{x}_i^{(l+1)} = \sum_{j=1}^{N}\sum_{\epsilon=1}^{M} H_{i\epsilon}H_{j\epsilon}W_{\epsilon\epsilon}\mathbf{x}_j^{(l)}\mathbf{P}^{(l)}, \quad (2)$$

where $\mathbf{P}^{(l)} \in \mathbb{R}^{d\times d}$ is the learnable parameter matrix between two convolutional layers. Following the suggestions in (Wu et al. 2019a), we do not use nonlinear activation function. For $W_{\epsilon\epsilon}$, we assign each hyperedge the same weight 1. The matrix form of Eq. (1) with row normalization is:

$$\mathbf{X}_h^{(l+1)} = \mathbf{D}^{-1}\mathbf{H}\mathbf{W}\mathbf{B}^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{X}_h^{(l)}\mathbf{P}^{(l)}. \quad (3)$$

The hypergraph convolution can be viewed as a two-stage refinement performing 'node-hyperedge-node' feature transformation upon hypergraph structure. The multiplication operation $\mathbf{H}^\top\mathbf{X}_h^{(l)}$ defines the information aggregation from nodes to hyperedges and then premultiplying $\mathbf{H}$ is viewed to aggregate information from hyperedges to nodes.

After passing $\mathbf{X}_h^{(0)}$ through L hypergraph convolutional layer, we average the items embeddings obtained at each layer to get the final item embeddings $\mathbf{X}_h^* = \frac{1}{L+1}\sum_{l=0}^{L}\mathbf{X}_h^{(l)}$. Session embeddings can be represented by aggregating representation of items in that session. We follow the strategy used in SR-GNN (Wu et al. 2019b) to refine the embedding of session $s = [i_{s,1}, i_{s,2}, i_{s,3}, ..., i_{s,m}]$:

$$\begin{aligned} \alpha_t &= \mathbf{f}^\top\sigma\left(\mathbf{W}_1\mathbf{x}_m^* + \mathbf{W}_2\mathbf{x}_t^* + \mathbf{c}\right), \\ \mathbf{s}_g &= \sum_{t=1}^{m}\alpha_t\mathbf{x}_t^*, \;\; \theta_h = \mathbf{W}_3[\mathbf{x}_m^*; \mathbf{s}_g], \end{aligned} \quad (4)$$

where $\mathbf{x}_m^*$ is the embedding of the last item in session s, and $\mathbf{x}_t^*$ is the embedding of the t-th item in session s. Let $\mathbf{x}_m^*$ denote the current user intent, and the user's general interest embedding $\mathbf{s}_g$ across this session is represented by aggregating item embeddings through a soft-attention mechanism where items have different levels of priorities. $\mathbf{f} \in \mathbb{R}^d$, $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$ are attention parameters used to learn the item weight α_t. The hybrid session embedding θ_h concatenates $\mathbf{x}_m^*$ and $\mathbf{s}_g$ to denote the complete user preference inferred from the hypergraph structure and $\mathbf{W}_3 \in \mathbb{R}^{d\times 2d}$ transforms the hybrid session embedding into $\mathbb{R}^d$ space. Note that, following our motivation in Section I, we abandon the sequence modeling techniques like GRU units and self-attention used in other SBR models. The current intent is the only temporal factor we use, and hence our model is very efficient and lightweight.

Line Graph Channel and Convolution The line graph channel encodes the line graph of the hypergraph. Fig. 1 shows how we transform the hypergraph into a line graph of it. The line graph can be seen as a simple graph which contains the cross-session information and depicts the connectivity of hyperedges. Before the convolution operation, analogously, we pass $\mathbf{X}^{(0)}$ through the SGU to obtain the line graph channel-specific item embeddings $\mathbf{X}_l^{(0)}$. As there are no item involved in the line graph channel, we first initialize the channel-specific session embeddings $\mathbf{\Theta}_l^{(0)}$ by looking up the items belonged to each session and then averaging the corresponding items embeddings in $\mathbf{X}_l^{(0)}$. An incidence matrix for $L(G)$ is defined as $\mathbf{A} \in \mathbb{R}^{M\times M}$ where M is the number of nodes in the line graph and $A_{p,q} = W_{p,q}$ according to Definition 2. Let $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ where $\mathbf{I}$ is an identity matrix. $\hat{\mathbf{D}} \in \mathbb{R}^{M\times M}$ is a diagonal degree matrix where $\hat{\mathbf{D}}_{p,p} = \sum_{q=1}^{m}\hat{\mathbf{A}}_{p,q}$. The line graph convolution is then defined as:

$$\mathbf{\Theta}_l^{(l+1)} = \hat{\mathbf{D}}^{-1}\hat{\mathbf{A}}\mathbf{\Theta}^{(l)}\mathbf{Q}^{(l)}, \quad (5)$$

where $\mathbf{Q}^{(l)} \in \mathbb{R}^{d\times d}$ is the weight matrix. In each convolution, the sessions gather information from their neighbors. By doing so, the learned $\mathbf{\Theta}$ can capture the cross-session information. Likewise, we pass $\mathbf{\Theta}_l^{(0)}$ through L graph convolutional layer, and then average the session embeddings obtained at each layer to get the final session embeddings $\mathbf{\Theta}_l = \frac{1}{L+1}\sum_{l=0}^{L}\mathbf{\Theta}_l^{(l)}$.

Model Optimization and Recommendation Generation Given a session s, we compute scores $\hat{\mathbf{z}}$ for all the candidate items $i \in I$ by doing inner product between the base item embedding $\mathbf{X}^0$ and θ_s^h and θ_s^l, respectively. Then we add up the two predicted scores to get the final predictions:

$$\hat{\mathbf{z}}_i = (\theta_s^h + \theta_s^l)^T\mathbf{x}_i. \quad (6)$$

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

$$\hat{\mathbf{y}} = \mathrm{softmax}(\hat{\mathbf{z}}). \quad (7)$$

We formulate the learning objective as a cross entropy loss function, which has been extensively used in recommender systems and defined as:

$$\mathcal{L}_r = -\sum_{i=1}^{N}\mathbf{y}_i\log\left(\hat{\mathbf{y}}_i\right) + (1-\mathbf{y}_i)\log\left(1-\hat{\mathbf{y}}_i\right), \quad (8)$$

where $\mathbf{y}$ is the one-hot encoding vector of the ground truth. For simplicity, we leave out the L_2 regularization terms. By minimizing $\mathcal{L}_r$ with Adam, we can get high-quality session-based recommendations.

Enhancing DHCN with Self-Supervised Learning

The hypergraph modeling empowers our model to achieve significant performance. However, we consider that the sparsity of session data might hinder hypergraph modeling, which would result in a suboptimal recommendation performance. Inspired by the successful practices of self-supervised learning on simple graphs, we innovatively integrate self-supervised learning into the network to enhance hypergraph modeling. Learning with self-supervision signals is seen as the auxiliary task to benefit our recommendation task and it proceeds by following two steps:

(1) **Creating self-supervision signals.** Recall that, in DHCN, we learn two groups of channel-specific session embeddings via the two channels. Since each channel encodes a (hyper)graph that only depicts either of the item-level (intra-session) or the session-level (inter-session) structural information of the session-induced hypergraph, the two groups of embeddings know little about each other but can mutually complement. For each mini-batch including n sessions in the training, there is a bijective mapping between the two groups of session embeddings. Straightforwardly, the two groups can be the ground-truth of each other for self-supervised learning, and this one-to-one mapping is seen as the label augmentation. If two session embeddings both denote the same session in two views, we label this pair as the ground-truth, otherwise we label it as the negative.

(2) **Contrastive learning.** Following (Velickovic et al. 2019; Bachman, Hjelm, and Buchwalter 2019), we regard the two channels in DHCN as two views characterizing different aspects of sessions. We then contrast the two groups of session embeddings learned via the two views. We adopt InfoNCE (Oord, Li, and Vinyals 2018) with a standard binary cross-entropy loss between the samples from the ground-truth (positive) and the corrupted samples (negative) as our learning objective and defined it as:

$$\mathcal{L}_s = -\log\sigma(f_{\mathrm{D}}(\theta_i^h, \theta_i^l)) - \log\sigma(1 - f_{\mathrm{D}}(\tilde{\theta}_i^h, \theta_i^l)), \quad (9)$$

where $\tilde{\theta}_i^h$ (or $\tilde{\theta}_i^l$) is the negative sample obtained by corrupting Θ_h (Θ_l) with row-wise and column-wise shuffling, and $f_{\mathrm{D}}(\cdot) : \mathbb{R}^d \times \mathbb{R}^d \longmapsto \mathbb{R}$ is the discriminator function that takes two vectors as the input and then scores the agreement between them. We simply implement the discriminator as the dot product between two vectors. This learning objective is explained as maximizing the mutual information between the session embeddings learned in different views (Velickovic et al. 2019). By doing so, they can acquire information from each other to improve their own performance in item/session 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.

Finally, we unify the recommendation task and this self-supervised task into a *primary&auxiliary* learning framework, where the former is the primary task and the latter is the auxiliary task. Formally, the joint learning objective is defined as:

$$\mathcal{L} = \mathcal{L}_r + \beta\mathcal{L}_s, \quad (10)$$

where β controls the magnitude of the self-supervised task.

Experiments

Experimental Settings

Datasets. We evaluate our model on two real-world benchmark datasets: *Yoochoose*[2] and *Diginetica*[3]. For both datasets, we follow (Wu et al. 2019b; Li et al. 2017) to remove all sessions containing only one item and also remove items appearing less than five times. To evaluate our model, we split both datasets into training/test sets, following the settings in (Wu et al. 2019b; Li et al. 2017). For *Yoochoose* dataset, the test set consists of sessions of the subsequent days with respect to the training set. For *Diginetica* dataset, the only difference is that we use the sessions of the subsequent weeks for testing. Then, we augment and label the dataset by using a sequence splitting method, which generates multiple labeled sequences with the corresponding labels $([i_{s,1}], i_{s,2}), ([i_{s,1}, i_{s,2}], i_{s,3}), ..., ([i_{s,1}, i_{s,2}, ..., i_{s,m-1}], i_{s,m})$ for every session $s = [i_{s,1}, i_{s,2}, i_{s,3}, ..., i_{s,m}]$. Note that the label of each sequence is the last click item in it. As the training set of *Yoochoose* is quite large, we follow (Wu et al. 2019b; Liu et al. 2018; Li et al. 2017) to only leverage the most recent $1/64$ and $1/4$ of the whole training sequences and form two new training sets for our experiments, and name them as *Yoochoose1/64* and *Yoochoose1/4*. The statistics of the datasets are presented in Table 1.

Table 1: Dataset Statistics

Dataset	Yooch1/64	Yooch1/4	Diginetica
training sessions	369,859	5,917,745	719,470
test sessions	55,898	55,898	60,858
# of items	16,766	29,618	43,097
average lengths	6.16	5.71	5.12

Baseline Methods. We compare DHCN with the following representative methods:

- **Item-KNN**(Sarwar et al. 2001) recommends items similar to the previously clicked item in the session, where the cosine similarity between the vector of sessions is used.
- **FPMC** (Rendle, Freudenthaler, and Schmidt-Thieme 2010) is a sequential method based on Markov Chain.
- **GRU4REC** (Hidasi et al. 2015) utilizes a session-parallel mini-batch training process and adopts ranking-based loss functions to model user sequences.
- **NARM** (Li et al. 2017): is a RNN-based model that models the sequential behavior to generate the recommendations.
- **STAMP** (Liu et al. 2018): employs the self-attention mechanism to enhance session-based recommendation.
- **SR-GNN** (Wu et al. 2019b): applies a gated graph convolutional layer to learn item transitions.

[2]https://2015.recsyschallenge.com/challenge.html

[3]http://cikm2016.cs.iupui.edu/cikm-cup/

Table 2: Performances of all comparison methods on three datasets.

Method	Yoochoose1/64				Yoochoose1/4				Diginetica			
	P@10	M@10	P@20	M@20	P@10	M@10	P@20	M@20	P@10	M@10	P@20	M@20
Item-KNN	48.69	24.94	51.60	21.81	50.50	25.78	52.31	21.70	25.07	10.77	35.75	11.57
FPMC	37.44	20.05	45.62	15.01	-	-	-	-	15.43	6.20	26.53	6.95
GRU4REC	52.43	24.53	60.64	22.89	55.49	26.05	59.53	22.60	17.93	7.33	29.45	8.33
NARM	57.83	27.42	68.32	28.63	57.98	28.51	69.73	29.23	35.44	15.13	49.70	16.17
STAMP	58.07	28.92	68.74	29.67	59.62	29.24	70.44	30.00	33.98	14.26	45.64	14.32
SR-GNN	60.21	30.12	70.57	30.94	61.06	31.08	71.36	31.89	36.86	15.52	50.73	17.59
FGNN	60.97	30.85	71.12	31.68	61.98	32.43	71.97	32.54	37.72	15.95	51.36	18.47
DHCN	**64.02**	**36.75**	**73.43**	**37.34**	**68.72**	**36.96**	**77.61**	**36.72**	**51.45**	**30.05**	**61.62**	**30.58**
S^2-DHCN	**64.33**	**37.32**	**74.10**	**37.89**	**70.01**	**39.50**	**78.96**	**40.13**	**52.52**	**30.42**	**63.20**	**30.94**
Improv. (%)	5.51	20.97	4.19	19.60	12.95	21.80	9.71	23.32	39.23	90.72	23.06	67.51

- **FGNN** (Qiu et al. 2019): formulates the next item recommendation within the session as a graph classification problem.

Evaluation Metrics. Following (Wu et al. 2019b; Liu et al. 2018), we use P@K (Precision) and MRR@K (Mean Reciprocal Rank) to evaluate the recommendation results.

Hyper-parameters Settings. For the general setting, the embedding size is 100, the batch size for mini-batch is 100, and the L_2 regularization is 10^{-5}. For DHCN, a two-layer structure is adopted, and an initial learning rate 0.001 is used. For the baseline models, we refer to their best parameter setups reported in the original papers and directly report their results if available, since we use the same datasets and evaluation settings.

Experimental results

Overall Performance. The experimental results of overall performance are reported in Table 2, and we highlight the best results of each column in boldface. Two variants of DHCN are evaluated, and **S^2-DHCN** (with $\beta = 3$) denotes the self-supervised version. The improvements are calculated by using the difference between the performance of S^2-DHCN and the best baseline to divide the performance of the latter. We do not report the results of FPMC like (Li et al. 2017) since the memory requirement to run FPMC on *Yoochoose1/4* is too large for ordinary deep-learning computing platforms. Analyzing the results in Table 2, we can draw the following conclusions.

- The GNNs-based models: SR-GNN and FGNN outperform RNNs-based models. The improvements can be owed to the great capacity of graph neural networks. However, the improvements are also trivial compared with the improvements brought by DHCN.
- Our proposed DHCN shows overwhelming superiority over all the baselines on all datasets. In particular, the improvements on *Diginetica* are significant. After analyzing the data pattern of *Diginetica*, we find that many items often co-occur in different sessions in the form of frequent itemsets, which is ideal for hypergraph modeling. This could be the reason that leads to such extraordinary results. Compared with SR-GNN and FGNN, our model has two advantages: (1) It uses hypergraph to capture the beyond pairwise relations. By modeling each hyperedge as a clique whose items are fully connected, the connections between distant items can be exploited. (2) The line graph takes the cross-session information into consideration. Besides, it should be noted that the improvements on MRR are more remarkable than those on Precision, which means that DHCN can not only succeed in hitting the ground-truth item, but largely promote its rank in the top-K recommendation lists.
- Although not as considerable as those brought by hypergraph modeling, the improvements brought by self-supervised learning are still decent. In particular, on the two datasets which have shorter average length of sessions, self-supervised learning plays a more important role, which is line with our assumption that the sparsity of session data might hinder the benefits of hypergraph modeling, and maximizing mutual information between the two views in DHCN could address it.

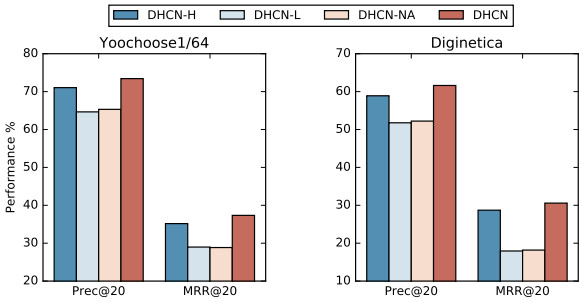

Figure 2: Contribution of each component.

Ablation Study. The overwhelming superiority of DHCN shown in the last section can be seen as the result of the joint effect of hypergraph modeling, and temporal factor exploitation. To investigate the contributions of each module in DHCN, we develop three variants of DHCN: **DHCN-H**, **DHCN-L**, and **DHCN-NA**. DHCN-H means only the hypergraph channel is used, and DHCN-L means only the line

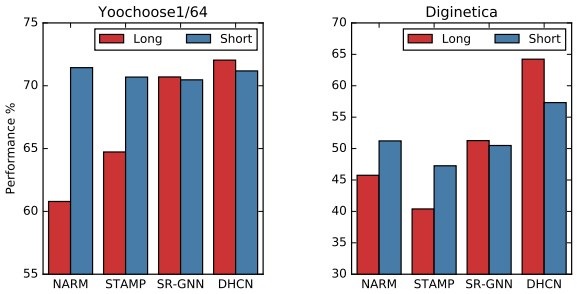

Figure 3: P@20 results with different methods for short and long sessions.

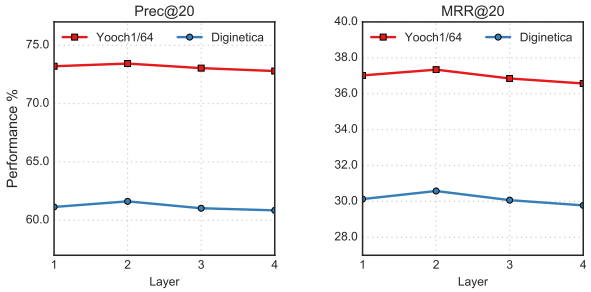

Figure 4: The impacts of the number of layer.

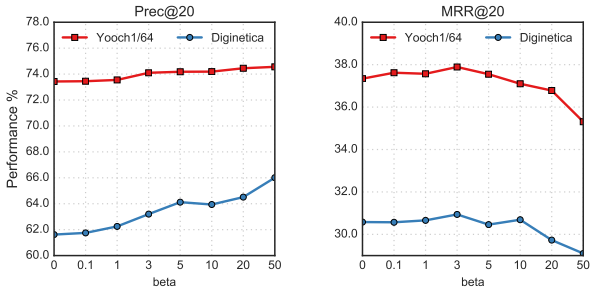

Figure 5: The impact of the magnitude of self-supervised learning.

graph channel is used. DHCN-NA represents the version without the soft attention mechanism. We compare them with the full DHCN on *Yoochoose1/64* and *Diginetica*.

As can be observed in Figure 2, each component contributes to the final performance. The hypergraph channel contributes the most. When only using this channel (attention employed), the network achieves a significant performance which is much higher than the performance of the baselines on both the two datasets. By contrast, only using the line graph channel would lead to a huge performance degradation on both datasets. But on diginetica, DHCN-L is still comparable to the baselines. This can demonstrate the effectiveness of modeling the high-order item correlations and the necessity of capturing cross-session information to complement the item-level information. Besides, removing the soft attention in the hypergraph channel would also result in a big performance drop on both datasets, which is in line with our assumption in Section I that items in a session are temporally related. According to this ablation study, we can draw a conclusion that a successful SBR model should consider both the temporal factors and high-order item correlations.

Impact of Different Session Lengths. In the real world, sessions are usually with various lengths. It is necessary to know how stable DHCN is when dealing with them. We follow (Liu et al. 2018) to split the sessions of *Yoochoose1/64* and *Diginetica* into two groups with different lengths and name them as **Short** and **Long**. Short contains sessions whose lengths are less than or equal to 5, while Long contains sessions whose lengths are larger than 5. We choose the cutting point to be 5 since it is the most commonly seen length of all sessions. Then, we compare the performance of DHCN and NARM, STAMP and SR-GNN in terms of P@20 on Short and Long. Results in Figure 3 show that DHCN stably outperforms all the baseline models on *Yoochoose1/64* and *Diginetica* with different session lengths in most cases. Particularly, the performance on Long is much better than that of the baselines. The results demonstrate the adaptability of DHCN in real-world session-based recommendation.

Impact of Model Depth. To study the impact of the depth of DHCN, we range the numbers of layers of DHCN within $\{1, 2, 3, 4\}$. According to the results presented in Figure 4, DHCN is not very sensitive to the number of layers on both datasets. A 2-layer setting is the best. When the number of layer is larger than 2, the performance slightly drops due to over-smoothing.

Impact of Self-Supervised Learning. We introduce a hyper-parameter β to S^2-DHCN to control the magnitude of self-supervised learning. To investigate the influence of the self-supervised task based on two-view contrastive learning. We report the performance of S^2-DHCN with a set of representative β values $\{0.1, 1, 3, 5, 10, 20, 50\}$. According to the results presented in Figure 5, recommendation task achieves decent gains when optimized with the self-supervised task. Small β can boost both Prec@20 and MRR@20 on the two datasets. However, with the increase of β, MRR@ declines, while Prec@20 still keeps growing. Currently, we have no idea why there is no performance drop on Prec@20, and hope to figure it out in our future work. Overall, it means that it is important to make a trade-off between the hit ratio and item ranks when choosing the value of β.

Conclusion

Existing GNNs-based SBR models regard the item transitions as pairwise relations, which cannot capture the ubiquitous high-order correlations among items. In this paper, we propose a dual channel hypergraph convolutional network for SBR to address this problem, Moreover, to further enhance the network, we innovatively integrate self-supervised into the training of the network. Extensive empirical studies demonstrate the overwhelming superiority of our model, and the ablation study validates the effectiveness and rationale of hypergraph convolution and self-supervised learning.

Acknowledgment

This work was supported by ARC Discovery Project (GrantNo.DP190101985, DP170103954).

References

Bachman, P.; Hjelm, R. D.; and Buchwalter, W. 2019. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, 15535–15545.

Bandyopadhyay, S.; Das, K.; and Murty, M. N. 2020. Line Hypergraph Convolution Network: Applying Graph Convolution for Hypergraphs. *arXiv preprint arXiv:2002.03392* .

Bretto, A. 2013. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer* .

Bu, J.; Tan, S.; Chen, C.; Wang, C.; Wu, H.; Zhang, L.; and He, X. 2010. Music recommendation by unified hypergraph: combining social media information and music content. In *Proceedings of the 18th ACM international conference on Multimedia*, 391–400.

Chen, T.; and Wong, R. C.-W. 2020. Handling Information Loss of Graph Neural Networks for Session-based Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1172–1180.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; and Gao, Y. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3558–3565.

Guo, L.; Yin, H.; Wang, Q.; Chen, T.; Zhou, A.; and Quoc Viet Hung, N. 2019. Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1569–1577.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive Multi-View Representation Learning on Graphs. *arXiv preprint arXiv:2006.05582* .

Hidasi, B.; and Karatzoglou, A. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 843–852.

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

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* .

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780.

Jannach, D.; and Ludewig, M. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 306–310.

Jiang, J.; Wei, Y.; Feng, Y.; Cao, J.; and Gao, Y. 2019. Dynamic hypergraph neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2635–2641.

Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; and Ma, J. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1419–1428.

Li, L.; and Li, T. 2013. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *Proceedings of the sixth ACM international conference on Web search and data mining*, 305–314.

Liu, Q.; Zeng, Y.; Mokhosi, R.; and Zhang, H. 2018. 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*, 1831–1839.

Ma, J.; Zhou, C.; Yang, H.; Cui, P.; Wang, X.; and Zhu, W. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 483–491.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* .

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020a. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1150–1160.

Qiu, R.; Huang, Z.; Li, J.; and Yin, H. 2020b. Exploiting Cross-session Information for Session-based Recommendation with Graph Neural Networks. *ACM Transactions on Information Systems (TOIS)* 38(3): 1–23.

Qiu, R.; Li, J.; Huang, Z.; and Yin, H. 2019. 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*, 579–588.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, 811–820.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation al-

gorithms. In *Proceedings of the 10th international conference on World Wide Web*, 285–295.

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6(Sep): 1265–1295.

Tan, Y. K.; Xu, X.; and Liu, Y. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 17–22.

Tuan, T. X.; and Phuong, T. M. 2017. 3D convolutional networks for session-based recommendation with content features. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 138–146.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *ICLR (Poster)*.

Wang, J.; Ding, K.; Hong, L.; Liu, H.; and Caverlee, J. 2020a. Next-item Recommendation with Sequential Hypergraphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1101–1110.

Wang, S.; Cao, L.; and Wang, Y. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* .

Wang, W.; Zhang, W.; Liu, S.; Liu, Q.; Zhang, B.; Lin, L.; and Zha, H. 2020b. Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In *Proceedings of The Web Conference 2020*, 3056–3062.

Wang, Z.; Wei, W.; Cong, G.; Li, X.-L.; Mao, X.-L.; and Qiu, M. 2020c. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 169–178.

Whitney, H. 1992. Congruent graphs and the connectivity of graphs. In *Hassler Whitney Collected Papers*, 61–79. Springer.

Wu, F.; Zhang, T.; Souza Jr, A. H. d.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019a. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153* .

Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; and Tan, T. 2019b. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 346–353.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* .

Xin, X.; Karatzoglou, A.; Arapakis, I.; and Jose, J. M. 2020. Self-Supervised Reinforcement Learning forRecommender Systems. *arXiv preprint arXiv:2006.05779* .

Xu, C.; Zhao, P.; Liu, Y.; Sheng, V. S.; Xu, J.; Zhuang, F.; Fang, J.; and Zhou, X. 2019. Graph contextualized self-attention network for session-based recommendation. In *Proc. 28th Int. Joint Conf. Artif. Intell.(IJCAI)*, 3940–3946.

Yadati, N.; Nimishakavi, M.; Yadav, P.; Nitin, V.; Louis, A.; and Talukdar, P. 2019. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *Advances in Neural Information Processing Systems*, 1509–1520.

Yin, H.; Wang, Q.; Zheng, K.; Li, Z.; Yang, J.; and Zhou, X. 2019. Social influence-based group representation learning for group recommendation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 566–577. IEEE.

Yu, J.; Yin, H.; Li, J.; Gao, M.; Huang, Z.; and Cui, L. 2020. Enhance Social Recommendation with Adversarial Graph Convolutional Networks. *arXiv preprint arXiv:2004.02340* .

Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J. M.; and He, X. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 582–590.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* .

Zhou, K.; Wang, H.; Zhao, W. X.; Zhu, Y.; Wang, S.; Zhang, F.; Wang, Z.; and Wen, J.-R. 2020. Sˆ 3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. *arXiv preprint arXiv:2008.07873* .

Zimdars, A.; Chickering, D. M.; and Meek, C. 2013. Using temporal data for making recommendations. *arXiv preprint arXiv:1301.2320* .