Collaborative Graph Learning for Session-based Recommendation

ZHIQIANG PAN, FEI CAI, WANYU CHEN, CHONGHAO CHEN, and HONGHUI CHEN,
National University of Defense Technology, Changsha, China

Session-based recommendation (SBR), which mainly relies on a user's limited interactions with items to generate recommendations, is a widely investigated task. Existing methods often apply RNNs or GNNs to model user's sequential behavior or transition relationship between items to capture her current preference. For training such models, the supervision signals are merely generated from the sequential interactions inside a session, neglecting the correlations of different sessions, which we argue can provide additional supervisions for learning the item representations. Moreover, previous methods mainly adopt the cross-entropy loss for training, where the user's ground truth preference distribution towards items is regarded as a one-hot vector of the target item, easily making the network over-confident and leading to a serious overfitting problem. Thus, in this article, we propose a **Collaborative Graph Learning (CGL)** approach for session-based recommendation. CGL first applies the Gated Graph Neural Networks (GGNNs) to learn item embeddings and then is trained by considering both the main supervision as well as the self-supervision signals simultaneously. The main supervisions are produced by the sequential order while the self-supervisions are derived from the global graph constructed by all sessions. In addition, to prevent overfitting, we propose a **Target-aware Label Confusion (TLC)** learning method in the main supervised component. Extensive experiments are conducted on three publicly available datasets, i.e., Retailrocket, Diginetica, and Gowalla. The experimental results show that CGL can outperform the state-of-the-art baselines in terms of Recall and MRR.

CCS Concepts: • **Information systems → Recommender systems;**

Additional Key Words and Phrases: Session-based recommendation, collaborative learning, graph neural networks, self-supervised learning, label confusion learning

ACM Reference format:
Zhiqiang Pan, Fei Cai, Wanyu Chen, Chonghao Chen, and Honghui Chen. 2022. Collaborative Graph Learning for Session-based Recommendation. *ACM Trans. Inf. Syst.* 40, 4, Article 72 (April 2022), 26 pages.
https://doi.org/10.1145/3490479

1 INTRODUCTION

Recommender systems (RS) is an effective way to provide users with personalized information for maintaining their demands [1, 56, 72]. Most existing recommenders generate recommendations

This work was partially supported by the National Natural Science Foundation of China under No. 61702526, and the Postgraduate Scientific Research Innovation Project of Hunan Province under No. CX20200055.
Authors' address: Z. Pan, F. Cai (corresponding author), W. Chen, C. Chen, and H. Chen, Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, 410073, China; emails: {panzhiqiang, caifei, wanyuchen, chenchonghao, chenhonghui}@nudt.edu.cn.

1046-8188/2022/04-ART72 $15.00
https://doi.org/10.1145/3490479

by relying on user preference reflected in their long-term interactions with items [17, 18, 29, 61], which is known as user's inherent interest [6, 7]. However, in some realistic scenarios, only limited sequential interactions are available for detecting user intent, where the general recommenders are not applicable [20, 46, 47]. Thus, session-based recommendation (SBR) is proposed to make recommendations merely based on the ongoing session, where a session can be defined as a user's actions during a short period, e.g., 24 hours [20, 40].

Traditional methods typically rely on the similarity between items [45] or sessions [12, 22] to identify similar items to the current session for recommendations or utilize the Markov Chains to capture the sequential signal between adjacent items in the session [43]. Such models fail to capture user's dynamic interest migration in the entire session [28, 57]. Recently, neural models such as Recurrent Neural Networks (RNNs) and Graph Neural Networks (GNNs) are adopted in SBR by considering their ability for modeling the sequential signal [19, 20] or the complex transition relationship between items [64] in the current session. In addition, as limited information can be used in the current session, the global information is incorporated by introducing neighbor sessions with the gating network [37, 57] or by propagating information on the global graph constructed from all sessions [39, 62, 66].

Although such models have achieved a comparable performance, the following limitations are remained. First, the state-of-the-art SBR models mainly adopt the list-wise loss function, i.e., cross-entropy loss for optimizing the parameters by contrasting the prediction score distribution with the user preference distribution on all items [28, 31, 36, 64]. However, user's true preference distribution is typically unknown since it is impossible for the user to interact with all items. Previous methods simply regard the one-hot encoding vector of the target item as the true distribution [28, 64]. In such cases, raising the prediction score of the target item will decrease the scores of other items at the same time [19], which means that the model optimization totally depends on the one-hot label, leading to the overfitting problem. Moreover, it cannot distinguish the negative items since they are equally treated (i.e., 0) in the one-hot encoding vector. Second, the supervision signals in previous SBR methods are generated according to the chronological order, i.e., predicting the item at timestamp t+1 by modeling previous t interacted items. This kind of training strategy can merely utilize the supervision signals in each session, neglecting the correlations of different sessions. Although neighbor sessions have been proved effective for enriching the current session for recommendation through RNNs or GNNs [37, 57, 62, 66], bias is easily introduced from unrelated neighbors.

To deal with such issues, we propose a **Collaborative Graph Learning (CGL)** approach for session-based recommendation, which mainly consists of two components, i.e., the main supervised module and the self-supervised module. In detail, for each session, the GGNNs are adopted to generate the item representations. Then, on the one hand, to prevent overfitting, in the main supervised component, we contrast user's prediction score distribution against the ground truth preference distribution for optimizing the model parameters, which are respectively generated based on user's dynamic preference and the designed Target-aware Label Confusion (TLC). On the other hand, we construct a global graph to build the connections between different sessions, and utilize the self-supervised learning to enhance the item representations by contrasting the long-term preference in each session with its neighbors on the global graph.

In summary, our contributions in this article are listed as follows:

(1) We propose a CGL model, utilizing the self-supervised learning, jointly learned with the main supervised learning, to explore the correlations of different sessions for enhancing the item representations.

(2) We design a TLC learning method to generate an accurate distribution of user preference on items to deal with the overfitting problem in SBR.
(3) Extensive experiments are conducted on three benchmark datasets, i.e., Retailrocket, Diginetica and Gowalla. The experimental results demonstrate the superiority of CGL over the state-of-the-art baselines in terms of Recall and MRR.

2 RELATED WORK

In this section, we review the major work for session-based recommendation in Section 2.1 and the self-supervised recommendation approaches in Section 2.2.

2.1 Session-Based Recommendation

2.1.1 Sequential SBR. As items in the session are organized according to the chronological order, many sequential methods like Markov Chains and RNNs are applied to capture the sequential signal in the current session [20, 43, 48, 58, 59]. For example, Rendle et al. [43] incorporate personalized Markov Chains based on the personalized transition matrices, and Wang et al. [58] further introduce the nonlinear operations to achieve better aggregation operations. As for the RNN-based methods, Hidasi et al. [20] propose GRU4REC where the Gated Recurrent Unit (GRU) is utilized to achieve the sequential behavior modeling in the session, and Hidasi and Karatzoglou [19] further extend GRU4REC by improving the pair-wise loss functions BPR and TOP1. In addition, attention mechanism [53] is widely applied in SBR, which can be utilized individually to distinguish items in the session according to their corresponding importance for detecting the interest migration [24, 31, 38, 51] or applied together with other methods such as RNNs to emphasize user's main intent [28, 60].

Such sequential methods generate user's preference strictly relying on the sequential signal. However, user's behavior pattern tends to be more complicated than the time order [40, 64], which results in the sequential methods not being able to detect the user intent in the current session accurately.

2.1.2 GNN-Based SBR. To take the complex transition relationship between items in the session into consideration, Graph Neural Networks (GNNs) are applied in session-based recommendation [36, 64, 71]. For instance, Wu et al. [64] propose SR-GNN which transforms each session into a session graph and utilizes the GGNNs to model the current session. Then, Qiu et al. [40] design a weighted graph attention network (WGAT) for computing the information flow between items in the session and obtain the user preference by a Readout function [55]. Then, long-term information is taken into consideration in the stage of information propagation [36] and session aggregation [69] to solve the problem of lacking long-range dependencies in GNNs. For example, Xu et al. [69] propose to utilize the self-attention networks to capture the long-distance dependencies between items in the session for aggregating the items as the user preference. Moreover, Pan et al. [36] propose to explore the long-distance information in the information propagation stage by a star graph neural network (SGNN) and prevent overfitting by adopting the highway networks [49].

However, the supervision signals in the above-mentioned GNN-based methods are merely generated according to the sequential signal in the current session, neglecting the correlations between sessions for learning the item representations. In addition, the widely adopted cross-entropy loss in SBR easily leads to a serious overfitting issue [36, 64].

2.1.3 Collaborative SBR. Considering limited information is contained in the current session, many collaborative methods are proposed to incorporate information from other sessions, so as to enhance the current session representation with neighbor sessions based on RNNs [22, 37, 57] or

by exploring the constructed global graph for information propagation in GNNs [39, 62]. Specifically, many neighbor-aware methods are proposed to incorporate similar sessions for helping to detect user's intent in the current session based on RNNs, where similar sessions can be recognized through the k-nearest neighbor (KNN) method [22], the memory networks [57], or relying on the user intent [37]. In addition, the global graph is constructed to represent the pair-wise transition relationship between items in all sessions to introduce the global information from all sessions. For instance, Qiu et al. [39] propose to propagate the information on the global graph by the graph attention networks to update the item embeddings. Moreover, Wang et al. [62] combine the item representations from both the session-level and global-level graphs for making recommendations. Xia et al. [66] design a dual-channel hypergraph convolutional network to exploit both the item-level and session-level relations, and maximize the mutual information between different channels for generating accurate item representations.

Existing collaborative SBR methods follow the mechanism that introduces external information outside the current session for generating user preference. However, the number of introduced neighbor sessions is usually large [22, 37, 57] and propagating information on the global graph [39, 62, 66] may introduce unrelated items as neighbors, inevitably resulting in introduced bias.

2.2 Self-Supervised Recommendation

Self-supervised learning (SSL) aims to learn representations from the unlabeled data, which can be categorized into the generative models [11, 52] and the contrastive models [4, 13]. Considering the effectiveness of SSL in computer vision [21] and natural language processing [26], recently the self-supervised learning is also introduced into recommender systems for learning accurate item representations. For instance, Xin et al. [68] propose self-supervised reinforcement learning for sequential recommendation to distinguish different behaviors. Moreover, Xie et al. [67] propose to apply the contrastive learning into sequential recommendation by maximizing the agreement of the same sequence from different augmented views. Furthermore, Zhou et al. [73] propose S3-Rec which applies the mutual information maximization (MIM) in sequential recommendation by pretraining and fine-tunes the model by the supervised signals. In addition, Ma et al. [34] propose to introduce the intention disentanglement into the self-supervised sequence-to-sequence training strategy, which is jointly combined with the sequence-to-item training.

Previous self-supervised recommendation methods merely learn the self-supervision signals from the sequence-to-item view by comparing the sequence with a single item or multiple successive items (i.e., the subsequence). They all neglect the correlations between different sequences, which can provide additional supervision information, apart from the sequence-to-item supervisions.

3 PRELIMINARIES

3.1 Problem Statement

We formally illustrate the task of session-based recommendation (SBR) aiming to predict user's action at the next timestamp in this section. Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ denote all items, where $|V|$ is the number of items in V. Assuming that all sessions are denoted as $U = \{S_1, S_2, \ldots, S_\tau \ldots, S_{|U|}\}$, where $|U|$ is the number of sessions. $S_\tau = \{v_1, v_2, \ldots, v_t, \ldots, v_n\}$ indicates the τ-th session consisting of n sequential items, and v_t denotes the item that the user interacted with at the t-th timestamp in session S_τ. Given a session S_τ, we output a probability distribution on all candidate items which indicates the probability of each item to be recommended to the user. Finally, items with the top N prediction scores will constitute the recommendation list. The main notations used in this article are listed in Table 1.

Table 1. Main Notations used in This Article

Notation	Description
U	a set of all sessions in the training set
V	a set of all candidate items
$S_\tau = \{v_1, v_2, \ldots, v_n\}$	the τ-th session in U which contains n chronological items
N	the number of items constituting the recommendation list
$\mathcal{G}_l = \{\mathcal{V}_l, \mathcal{E}_l\}$	the constructed local graph for session S_τ
$\mathbf{X}$	the learnt representation of nodes in $\mathcal{V}_l$ outputted by GGNNs
$\mathbf{A}^{in}, \mathbf{A}^{out}$	the incoming and outgoing matrices in GGNN
$\mathbf{W}^I, \mathbf{W}^O$	the trainable parameters for information propagation in GGNN
$\mathbf{x}_i^k$	the representation of node x_i in the k-th layer GGNN
$\mathbf{a}_i^k$	the propagated neighbor information to node x_i in GGNNs at layer k
$\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$	the representation of items in session S_τ outputted by the GGNNs
$\mathbf{z}^d, \mathbf{z}_l^d, \mathbf{z}_r^d$	the hybrid, long-term and recent interests in main supervised learning
$\mathbf{y}_i^p$	the prediction score on item v_i in V
α	a trade-off parameter in the target-aware label confusion
$\mathbf{y}^c, \mathbf{y}^{soft}, \mathbf{y}^{hard}$	the combined, soft and hard label distribution
$\mathcal{G}_g = \{\mathcal{V}_g, \mathcal{E}_g\}$	the constructed global graph based on all sessions U
M	the number of neighbor sessions incorporated for contrasting
$\mathbf{z}^l$	the long-term preference in self-supervised learning
$S_i \in N_{S_\tau}, \tilde{S}_i \in U \backslash N_{S_\tau}$	the sampled connected and unconnected session of S_τ in $\mathcal{G}_g$
L, L_{main}, L_{ssl}	the final, main-supervised and self-supervised loss
λ	a trade-off parameter for balancing L_{main} and L_{ssl}

3.2 Item Representation Learning

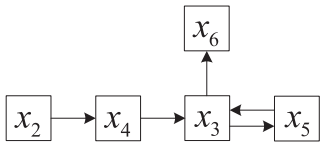
Given a session $S_\tau = \{v_1, v_2, \ldots, v_n\}$, the graph learning methods typically generate the accurate representation of items in the session by two stages, i.e., constructing the session graph and propagating information on the graph for learning item representations:

$$\begin{aligned} \mathcal{G}_l &= \text{Seq2Graph}(S_\tau), \\ \mathbf{X} &= \text{GNN}(\mathcal{G}_l), \end{aligned} \tag{1}$$

where $\mathcal{G}_l$ is the constructed local session graph, $\mathbf{X}$ is a matrix of the learnt representation of items in session S_τ, Seq2Graph and GNN denote the graph construction and information propagation operation, respectively.

Graph construction. We construct a directed session graph of S_τ to represent the pair-wise transitions between its contained items, which can be denoted as $\mathcal{G}_l = \{\mathcal{V}_l, \mathcal{E}_l\}$, where $\mathcal{V}_l$ and $\mathcal{E}_l$ indicate the nodes and edges in the graph, respectively. Here $\mathcal{V}_l = \{x_1, x_2, \ldots, x_m\}$ contains the unique items[1] in S_τ, and each edge $e_{ij} \in \mathcal{E}_l$ indicates that the user clicked node x_j after clicking x_i. We formulate a matrix $\mathbf{A}$ to indicate the weighted adjacent matrices which consists of an incoming matrix $\mathbf{A}^{in}$ and an outgoing matrix $\mathbf{A}^{out}$ to represent the transition relations between items, which are utilized to determine the weights of different neighbors in the information propagation stage. For example, given a session as $S_\tau = \{x_2, x_4, x_3, x_5, x_3, x_6\}$, the incoming and outgoing matrices can be constructed as in Figure 1.

[1]Note that $m \leq n$ since items may be repeatedly clicked in the sequence [36, 42].



Incoming Matrix

	2	4	3	5	6
2	0	0	0	0	0
4	1	0	0	0	0
3	0	1/2	0	1/2	0
5	0	0	1	0	0
6	0	0	1	0	0

Outgoing Matrix

	2	4	3	5	6
2	0	1	0	0	0
4	0	0	1	0	0
3	0	0	0	1/2	1/2
5	0	0	1	0	0
6	0	0	0	0	0

Fig. 1. An example of the adjacent matrices in GGNN.

Information Propagation. After constructing the graph, we conduct the information propagation on the graph to learn the accurate item representations. Here we adopt GGNN [30] as an example [36, 64, 69], other GNNs such as GAT [54] and GCN [25] can also be adopted to replace GGNN in our framework. First, the item vectors inputted into GGNNs are initialized by the embeddings of the items generated via an embedding layer, i.e., $\mathbf{X}^0 = \{\mathbf{x}_1^0, \mathbf{x}_2^0, \ldots, \mathbf{x}_m^0\}$, where $\mathbf{x}_i^0 \in \mathbb{R}^d$ is the representation of node x_i and d is the embedding dimension. Then, at the k-layer GGNN, we first obtain the information from the neighbors of node x_i as follows:

$$\begin{aligned}\mathbf{a}_i^k = Concat\Big(&\mathbf{A}_i^{in}\left[\mathbf{x}_1^{k-1}, \mathbf{x}_2^{k-1}, \ldots, \mathbf{x}_m^{k-1}\right]\mathbf{W}^I + \mathbf{b}^I,\\ &\mathbf{A}_i^{out}\left[\mathbf{x}_1^{k-1}, \mathbf{x}_2^{k-1}, \ldots, \mathbf{x}_m^{k-1}\right]\mathbf{W}^O + \mathbf{b}^O\Big),\end{aligned} \tag{2}$$

where $\mathbf{x}_i^{k-1}$ denotes the item embeddings of node x_i at layer k-1, $\mathbf{A}_i^{in}$ and $\mathbf{A}_i^{out}$ are respectively, the i-th row of $\mathbf{A}^{in}$ and $\mathbf{A}^{out}$, i.e., the weights for controlling how much information comes from each neighbor of x_i for updating the node vector of x_i. $\mathbf{W}^I, \mathbf{W}^O \in \mathbb{R}^{d\times d}, \mathbf{b}^I, \mathbf{b}^O \in \mathbb{R}^d$ are learnable parameters.

After that, a GRU is utilized to combine the representation of x_i at layer k-1 and the propagated neighbor information at layer k as:

$$\mathbf{x}_i^k = \text{GRU}\left(\mathbf{a}_i^k, \mathbf{x}_i^{k-1}\right). \tag{3}$$

Moreover, multi-layer GGNNs can be stacked to combine the representation of the nodes in different layers, thus to take the high-order connections between the items in session S_τ into consideration. After stacking K layers of GGNNs, the item representations can be represented as $\mathbf{X}^K$, denoted as $\mathbf{X}$ for brevity, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$. Then, we recover the item sequence in the session from the graph as $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, where n is the sequence length.

4 APPROACH

The framework of CGL is plotted in Figure 2, which consists of two main modules, i.e., the main supervised and the self-supervised components. In the supervised component, we train the model based on the supervision signals generated according to the sequential order, where the target-aware label confusion is designed to generate accurate label distribution. Then, as for the self-supervised module, we derive the supervision signals from the correlations between different sessions based on the constructed global graph for enriching the item representations.

4.1 Main Supervised Learning

4.1.1 Dynamic Preference Generator. For session $S_\tau = \{v_1, v_2, \ldots, v_n\}$, we generate user's dynamic preference in the main supervised component. Specifically, the accurate representation of items is learned by Equation (1). We then obtain user's dynamic interest $\mathbf{z}^d$ by aggregating the

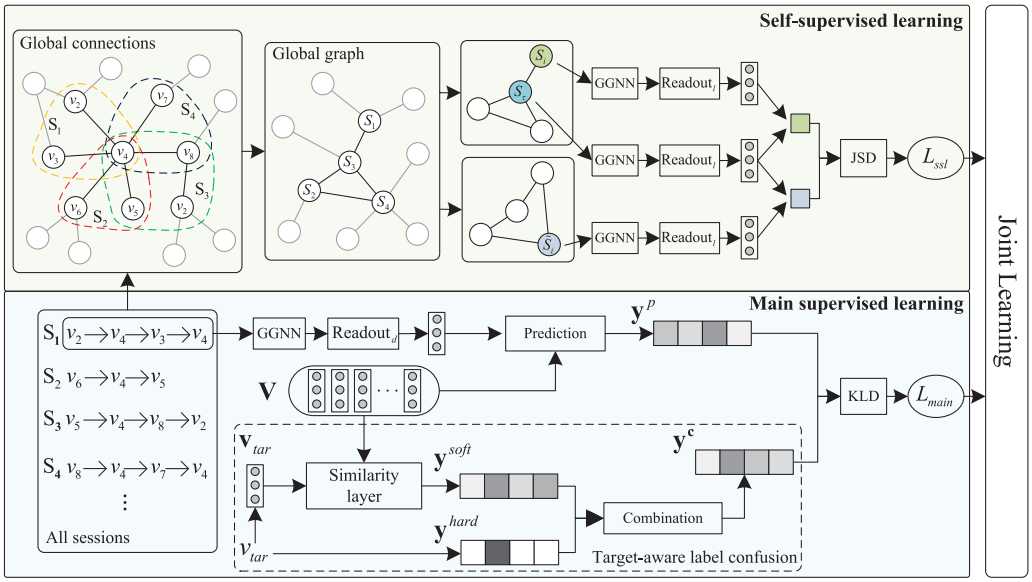


Fig. 2. The framework of CGL.

items in the session using the dynamic Readout function Readout_d as:

$$\mathbf{z}^d = \text{Readout}_d(\mathbf{v}_{1:n}) = \mathbf{W}_1\left[\mathbf{z}_l^d; \mathbf{z}_r^d\right], \tag{4}$$

where user's long-term preference $\mathbf{z}_l^d$ and recent interest $\mathbf{z}_r^d$ are simultaneously considered by concatenation, $\mathbf{W}_1 \in \mathbb{R}^{d\times 2d}$ contains trainable parameters. As the last item in the session can reflect user's current interest to capture the dynamic interest migration [31, 36, 64], here we directly adopt the embeddings of the last item as the recent preference, i.e., $\mathbf{z}_r^d = \mathbf{v}_n$.

Moreover, we obtain the long-term preference according to their corresponding importance scores, which are generated by the soft attention as:

$$\begin{aligned} \mathbf{z}_l^d &= \sum_{i=1}^{n} \beta_i^d \mathbf{v}_i, \\ \beta_i^d &= \text{Softmax}\left(\epsilon_i^d\right), \\ \epsilon_i^d &= \mathbf{W}_2\sigma\left(\mathbf{W}_3\mathbf{v}_i + \mathbf{W}_4\mathbf{z}_r^d + \mathbf{b}^d\right), \end{aligned} \tag{5}$$

where $\mathbf{W}_2 \in \mathbb{R}^{1\times d}$, $\mathbf{W}_3, \mathbf{W}_4 \in \mathbb{R}^{d\times d}$, $\mathbf{b}^d \in \mathbb{R}^d$ are learnable parameters in the soft attention mechanism, and σ denotes the sigmoid function.

4.1.2 Target-aware Label Confusion. After generating the user preference, we produce the prediction scores $\mathbf{y}^p \in \mathbb{R}^{|V|}$ on all candidate items in V by conducting a dot product on the user preference $\mathbf{z}^d$ and the embeddings of each item $\mathbf{v}_i$ as:

$$\mathbf{y}_i^p = \text{Softmax}\left(\mathbf{z}^{d\top}\mathbf{v}_i\right), \quad \forall i = 1, 2, \ldots, |V|, \tag{6}$$

where the Softmax function normalizes the prediction scores.

After obtaining the prediction distribution, we aim to minimize the distance between the prediction distribution and the user preference distribution. However, a user's true preference distribution is unknown as the user is unable to interact with all items. Previous work adopts the

one-hot vector of the target item [28, 36, 64] as the label distribution, which easily leads to the over-confident problem in the training phase, making the model overfitting. Moreover, the one-hot vector of the target item cannot distinguish the negative items since they are all assigned a value 0 in the label distribution. Here, to overcome these limitations, inspired by [15] we propose a target-aware label confusion method to generate the soft labels, which are then combined with the one-hot encoding vector for optimization to prevent overfitting. Intuitively, based on the hypothesis that items similar to the target item v_{tar} are close to user preference, we first obtain the soft labels generated via a similarity layer, where the similarity scores between all candidates in V and the target item can be obtained as:

$$\mathbf{y}_i^{soft} = \text{Softmax}\left(\mathbf{v}_{tar}^{\top}\mathbf{v}_i\right), \quad \forall i = 1, 2, \ldots, |V|, \tag{7}$$

After that, we combine $\mathbf{y}^{soft} \in \mathbb{R}^{|V|}$ with the original hard label, i.e., the one-hot encoding label vector $\mathbf{y}^{hard} \in \mathbb{R}^{|V|}$ where only the coordinate corresponding to the target item is 1 and the others are 0. This can be formalized as:

$$\mathbf{y}_i^{c} = \text{Softmax}\left(\mathbf{y}_i^{soft} + \alpha\mathbf{y}_i^{hard}\right), \quad \forall i = 1, 2, \ldots, |V|, \tag{8}$$

where $\mathbf{y}^c \in \mathbb{R}^{|V|}$ is the final label distribution and α is a trade-off parameter for controlling the weight between $\mathbf{y}^{soft}$ and $\mathbf{y}^{hard}$.

Finally, we adopt the Kullback-Leibler Divergence (KLD) [27] as the loss function for optimization in the main supervised component with the loss defined as:

$$L_{main} = \text{KLD}(\mathbf{y}^c, \mathbf{y}^p) = \sum_{i=1}^{|V|} \mathbf{y}_i^c \log\left(\frac{\mathbf{y}_i^c}{\mathbf{y}_i^p}\right). \tag{9}$$

Compared to the cross-entropy loss with the Softmax in previous work [28, 36, 64] that regards the one-hot encoding vector as the label distribution, our designed target-aware label confusion method is able to generate different soft labels for the negative items, so as to distinguish them and thus effectively prevent overfitting.

4.2 Self-Supervised Learning

As existing methods for SBR mainly neglect the connections between different sessions, which can provide additional supervision signals for training to enhance the item representations, we derive the self-supervision signals from the global graph of all sessions for training by utilizing the self-supervised learning, so as to help promote the recommendation accuracy on the basis of the main supervised recommender.

4.2.1 Global Graph Construction. To take the correlations between sessions into consideration, we propose to construct a global graph by relying on the global connections, where the relationship between sessions can be considered. Let $\mathcal{G}_g = \{\mathcal{V}_g, \mathcal{E}_g\}$ denotes the constructed global graph, where each node in $\mathcal{V}_g$ denotes a session in U, and the edges are determined by two stages, i.e., the similarity measurement and the max sampling.

Given a session $S_\tau \in U$ represented by a binary-valued vector, we first calculate its cosine similarity with other sessions in U to measure their relevance. Then, to avoid introducing noise by the loosely connected sessions, we apply the max sampling according to the calculated similarity. Specifically, we use the M most similar sessions as the final neighbors of session S_τ to build the global graph where each edge $e_{ij} \in \mathcal{E}_g$ means that session S_i and S_j are similar on the whole.

4.2.2 Long-Term Preference Generator. In the self-supervised component, unlike the user preference modeling in the main supervised component where we generate the user's dynamic preference, we produce the long-term preference in each session for contrasting, since the similarity defined between two sessions are overall, without specific correlations to the recent interest. In detail, given a session $S_\tau = \{v_1, v_2, \ldots, v_n\}$, we first learn the item representations by Equation (1), and then obtain the user's long-term preference by a long-term Readout function Readout_l based on a soft-attention mechanism as follows:

$$\begin{aligned} \mathbf{z}^l &= \sum_{i=1}^{n} \beta_i^l \mathbf{v}_i, \\ \beta_i^l &= \text{Softmax}\left(\epsilon_i^l\right), \\ \epsilon_i^l &= \mathbf{W}_5 \sigma\left(\mathbf{W}_6 \mathbf{v}_i + \mathbf{b}^l\right), \end{aligned} \tag{10}$$

where $\mathbf{W}_5 \in \mathbb{R}^{1\times d}$, $\mathbf{W}_6 \in \mathbb{R}^{d\times d}$ are trainable parameters in the soft-attention, and $\mathbf{b}^l \in \mathbb{R}^d$ is the bias vector.

Although the Readout functions used for session aggregation in the main supervised and self-supervised learning components are different, their corresponding preference generators share the same item embeddings and parameters in the GGNNs for item representation learning. Through the collaborative learning, the two user-preference generators can benefit each other by making accurate recommendations.

4.2.3 Self-Supervision. Based on the constructed global graph, we derive the supervision signals from the correlations between sessions via the self-supervised learning. Specifically, given a session S_τ where the neighbors of S_τ in the global graph is N_{S_τ}, session S_τ is more similar to its neighbor $S_i \in N_{S_\tau}$ than an unconnected session $\tilde{S}_i \in U \backslash N_{S_\tau}$, where "\" indicates the set subtraction operation. Based on this intuition, we adopt the contrastive loss, Jensen-Shannon Divergence (JSD) [21], to maximize the correlations of S_τ with its neighbors N_{S_τ} while minimizing that of S_τ with the unconnected sessions $U \backslash N_{S_\tau}$. Then, the self-supervised loss can be defined as:

$$L_{ssl} = \sum_{i=1}^{M} -\log \sigma(f(S_\tau, S_i)) - \log(1 - \sigma(f(S_\tau, \tilde{S}_i))), \tag{11}$$

where $S_i \in N_{S_\tau}$ is a neighbor session and $\tilde{S}_i \in U \backslash N_{S_\tau}$ is a negative session of S_τ, M is the number of neighbors incorporated for contrasting. For any two sessions S_p and S_q, the function $f(\cdot, \cdot)$ is defined as:

$$f(S_p, S_q) = {\mathbf{z}_p^l}^\top \mathbf{z}_q^l, \tag{12}$$

where $\mathbf{z}_p^l$ and $\mathbf{z}_q^l$ are the respective long-term user preference of S_p and S_q, generated by the long-term user preference generator.

4.3 Multi-Task Learning

After obtaining the main supervised loss by Equation (9) and the self-supervised loss by Equation (11), we combine them as the final loss for the joint training as:

$$L = L_{main} + \lambda L_{ssl}, \tag{13}$$

where λ is a trade-off parameter. Finally, we optimize the final loss L using the Back-Propagation Through Time (BPTT) algorithm [44].

We detail the learning algorithm of CGL in Algorithm 1. Given all sessions U in the training set, we first construct the global graph $\mathcal{G}_g$ in Step 1. Then, for each session S_τ, on the one hand, we

ALGORITHM 1: Training process of CGL.

Input: Session set $U = \{S_1, S_2, \ldots, S_{|U|}\}$;
Epochs: the number of training iterations;
α, λ: the trade-off parameters;
Output: $\mathbf{V}$: the embeddings of items in V;
ψ: the trainable parameters in CGL;

1: $\mathcal{G}_g = \{\mathcal{V}_g, \mathcal{E}_g\} \leftarrow$ GlobalGraphConstruct(U);
2: **for** epoch in range(Epochs) **do**
3: **for** S_τ in U **do**
4: $\mathbf{z}^d$ = DynamicPreference(S_τ) based on (1) and (4);
5: $\mathbf{y}^p$ = Prediction($\mathbf{z}^d, \mathbf{V}$) based on (6);
6: $\mathbf{y}^{soft}$ = SimilarityLayer($\mathbf{y}_{tar}, \mathbf{V}$) based on (7);
7: $\mathbf{y}^c$ = Combine($\mathbf{y}^{soft}, \mathbf{y}^{hard}, \alpha$) based on (8);
8: L_{main} = KLD($\mathbf{y}^c, \mathbf{y}^p$) based on (9);
9: $S_i, \tilde{S}_i$ = Sample($\mathcal{G}_g, S_\tau$);
10: $\mathbf{z}^l_\tau, \mathbf{z}^l_i, \tilde{\mathbf{z}}^l_i$ = Long-termPreference($S_\tau, S_i, \tilde{S}_i$) based on (1) and (10);
11: L_{ssl} = JSD($\mathbf{z}^l_\tau, \mathbf{z}^l_i, \tilde{\mathbf{z}}^l_i$) based on (11);
12: Optimize joint learning loss: $L = L_{main} + \lambda L_{ssl}$;
13: **end for**
14: use back-propagation to optimize the parameters;
15: **end for**
16: **return** $\mathbf{V}$ and ψ.

first obtain the dynamic user preference and make predictions in Steps 4 and 5, then we generate the soft labels in Step 6, which are combined with the hard label as the final label distribution in Step 7. We utilize the KL divergence as the supervised loss in Step 8. On the other hand, we sample the neighbor and negative sample of S_τ from the global graph in Step 9. After that, the long-term interests are obtained in Step 10, and the JS divergence is adopted as the self-supervised loss in Step 11. Finally, we obtain the joint learning loss in Step 12 and use the back-propagation to optimize the model parameters.

5 EXPERIMENTS

5.1 Research Questions

We validate the effectiveness of CGL by addressing the following research questions:

(RQ1) Can CGL achieve better performance than the state-of-the-art methods for the session-based recommendation task?
(RQ2) How is the contribution of each component in CGL?
(RQ3) Can other item representation methods work well as the graph neural networks in CGL?
(RQ4) How is the model performance on sessions of different lengths?
(RQ5) How is the impact of the key hyper-parameters on the performance of CGL?

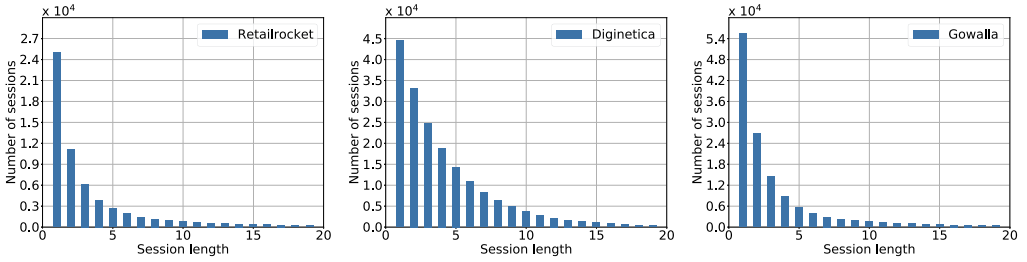
5.2 Datasets and Evaluation Metrics

We evaluate the performance of CGL and the baselines on three publicly available datasets, including two e-commerce datasets, i.e., Retailrocket and Diginetica, and a check-in dataset Gowalla, which are all widely used in session-based recommendation [3, 5, 16, 33, 41, 69]. Retailrocket[2] is

[2]https://www.kaggle.com/retailrocket/ecommerce-dataset.

Table 2. Dataset Statistics

Statistics	Retailrocket	Diginetica	Gowalla
# actions	337,771	1,177,742	696,273
# items	22,144	29,129	56,294
# training sessions	58,804	181,398	130,910
# validation sessions	7,407	21,206	16,754
# test sessions	14,961	37,943	29,234
Average session length	4.16	4.90	3.94
Average action per item	15.23	40.43	12.37



(a) Retailrocket. **(b) Diginetica.** **(c) Gowalla.**

Fig. 3. Distribution of sessions with various lengths in the training set of three datasets.

created from a personalized e-commerce company with user activities in 6 months. Here we only use the click data. Diginetica[3] comes from the CIKM Cup 2016. Following [5, 36, 64], we adopt the transaction data. Gowalla[4] is collected from February 2009 to October 2010 from a location-based social networking website. For Retailrocket, Diginetica, and Gowalla, we adopt the respective clicks, transactions and check-ins in the last month for experiments. Following [5, 41, 62], user's clicks in Retailrocket, transactions in Diginetica, and check-ins in Gowalla in 24 hours are defined as a session. Moreover, items appearing less than three times are removed, and sessions containing less than two interactions are filtered out [41]. We split the sessions according to the chronological order for training, validation, and test by the proportions of 70%, 10%, and 20%, respectively. The statistics of the three datasets after preprocessing are provided in Table 2. Moreover, we plot the distribution of sessions with different lengths in three datasets in Figure 3, where we can observe that the majority of sessions in all three datasets merely contain a few items.

Following [28, 36, 64], Recall@N and MRR@N are adopted as the metrics for evaluating the recommendation performance.

5.3 Model Summary

To examine the performance of our proposal, three groups of 14 state-of-the-art baselines are compared against CGL. (1) Four traditional methods, i.e., Item-KNN [10], FPMC [43], S-KNN [2], and VS-KNN [32]; (2) Four RNN or attention-based methods, i.e., GRU4REC [20], NARM [28], STAMP [31] and CSRM [57] as well as a simple neural method EMDE [9] relying on the fully connected layer; and (3) Five GNN based methods, i.e., SR-GNN [64], TAGNN [71], GCE-GNN [62], SGNN-HN [36] and S^2-DHCN [66].

[3] http://cikm2016.cs.iupui.edu/cikm-cup.
[4] https://snap.stanford.edu/data/loc-Gowalla.html.

- **Item-KNN** recommends candidates similar to the last item in the session by relying on the cosine similarity between the binary vector of every two items.
- **FPMC** is a hybrid method for item recommendation where the Markov Chains are used to model the sequential signal between items.
- **S-KNN** compares the entire current session with the past sessions in the training data to find the neighbor sessions for determining the recommendations.
- **VS-KNN** extends S-KNN by emphasizing the recent items in each session using a linear decay function when computing the similarity between two sessions.
- **GRU4REC** applies the Gated Recurrent Unit (GRU) for the sequential behavior modeling.
- **NARM** emphasizes user's main intent by enhancing GRU4REC with an attention mechanism.
- **STAMP** is an attention-based recommender which utilizes the attention mechanism and memory networks [50, 63] for dynamic user preference modeling.
- **CSRM** proposes to utilize the memory networks [50, 63] for introducing the neighbor sessions, which are used to enhance the current session modeling for helping to detect the user intent.
- **EMDE** adopts a feed-forward fully connected layer to detect user's interest by modeling the item representations generated by a Density-dependent Locality-Sensitive Hashing (DLSH) method.
- **SR-GNN** utilizes the gated graph neural networks to model the session and generates the session representation using an attentive aggregation.
- **TAGNN** enhances SR-GNN by introducing a target-aware attention network to help generate the user preference adaptive to different candidate items.
- **GCE-GNN** models the current session using the graph attention networks, enhanced by the global-level item representation learning on the global graph.
- **SGNN-HN** explores the long-distance information in information propagation by the star graph neural networks and prevents the overfitting problem using the highway networks [49].
- S^2**-DHCN** proposes a dual channel hypergraph convolutional network for modeling the pairwise item transitions and utilizes the self-supervised learning to enhance the session representation.

All models including our proposals are listed in Table 3.

5.4 Experimental Setup

Following [62, 64], both the batch size and the embedding dimension are set to 100. The Adam optimizer is adopted to optimize the model, where the learning rate is initialized as 10^{-3} and decays by 0.1 for every 3 epochs. We implement the GGNNs with one layer following [64]. Other hyper-parameters are tuned by a grid search on the validation set, where we search the trade-off parameters α and λ in $\{8, 10, 12, 14, 16\}$ and $\{0.01, 0.05, 0.1, 0.2, 0.5\}$, respectively, and the number of neighbor sessions M is ranged in $\{1, 2, 3, 4\}$. All parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1.

6 RESULTS AND DISCUSSION

6.1 Overall Performance

6.1.1 General Comparison. The results of CGL as well as the baselines on three datasets are presented in Table 4. In general, we find that the neural methods achieve better performance than the traditional methods in most cases. For the traditional methods, FPMC generally outperforms

Table 3. An Overview of Models Discussed in this Article

Model	Description	Source	Collaborative
Item-KNN	An item-item model relying on the cosine similarity to recommend candidates similar to user's last interacted item.	[10]	×
FPMC	A Markov Chains based hybrid method for item recommendation.	[43]	×
S-KNN	A session-based nearest-neighbor method which recommends items based on the neighbor sessions detected by a session-level similarity.	[2]	√
VS-KNN	An extension of S-KNN which emphasizes the recent items in each session when detecting the neighbor sessions.	[32]	√
GRU4REC	An RNN-based method which utilizes GRUs to capture the sequential signal between items in the session.	[20]	×
NARM	An attention based method which emphasizes user's main purpose using an attention mechanism based on GRU4REC.	[28]	×
STAMP	An attention based method which distinguishes the item importance using the attention mechanism and memory networks.	[31]	×
CSRM	A collaborative RNN based model which incorporates the neighbor sessions as auxiliary information for enriching the current session representation.	[57]	√
EMDE	A neural method encoding items by the Density-dependent Locality-Sensitive Hashing and modeling the user interest by a fully connected layer.	[9]	×
SR-GNN	A GNN-based method which applies the gated graph neural networks for generating the item representations.	[64]	×
TAGNN	A GNN-based method enhancing SR-GNN by a target-aware attention to generate adaptive preference for different candidates.	[71]	×
GCE-GNN	A collaborative GNN-based method which enhances the representation of items in the current session by the global graph.	[62]	√
SGNN-HN	A GNN-based method which extends SR-GNN by propagating long-distance information and preventing the overfitting problem.	[36]	×
S^2-DHCN	A collaborative GNN-based method which models the item-level and session-level relations using the hypergraph and line graph, respectively.	[66]	√
CGL	The CGL model which exploits the correlations between different sessions using self-supervised learning and designs a target-aware label confusion method for preventing overfitting.	This article	√
$\text{CGL}_{w/o[SSL]}$	A CGL model which removes the self-supervised learning component.	This article	×
$\text{CGL}_{w/o[TLC]}$	A CGL model which removes the label confusion learning module.	This article	√
CGL_{LS}	A CGL model replacing the label confusion with the label smoothing.	This article	√
CSATL	A CGL model replacing the GGNN with the self-attention mechanism.	This article	√
CRNNL	A CGL model replacing the GGNN with the recurrent neural networks.	This article	√
$\text{CGL}_{w/o[d]}$	A CGL model which merely models user's long-term preference in the main supervised learning.	This article	√

Item-KNN on Retailrocket and Diginetica while the phenomenon is different on Gowalla. This may be due to the fact that the sequential information is more important in the e-commerce setting, while the similarity between items has relatively more influence on the check-in dataset, i.e., Gowalla. Moreover, by detecting the neighbor sessions to determine recommendations, S-KNN and VS-KNN show competitive performance, which both obviously outperform Item-KNN and FPMC.

Then, the attention-based methods, e.g., NARM and STAMP, as well as EMDE, can all outperform the purely RNN-based method, i.e., GRU4REC. Moreover, by introducing the neighbor

Table 4. Model Performance

Method	Retailrocket		Diginetica		Gowalla	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
Item-KNN	16.12	7.56	17.34	6.89	3.99	1.32
FPMC	17.12	8.56	17.92	7.14	3.85	1.26
S-KNN	39.04	26.42	44.11	17.35	5.27	2.33
VS-KNN	38.11	26.88	43.04	17.38	5.08	2.37
GRU4REC	28.21	16.43	45.49	16.03	5.74	2.24
NARM	41.80	27.38	47.08	16.88	7.40	2.93
STAMP	36.57	23.26	48.77	18.17	7.36	3.05
CSRM	43.13	26.00	47.91	16.21	8.01	3.31
EMDE	41.13	26.06	48.02	17.39	7.15	3.14
SR-GNN	44.96	28.61	49.92	18.10	8.34	3.58
TAGNN	44.43	28.59	50.53	18.24	9.24	3.63
GCE-GNN	36.38	21.50	49.69	17.29	8.33	3.31
SGNN-HN	45.06	29.09	51.28	18.36	9.74	3.90
S^2-DHCN	44.51	27.33	50.90	17.42	9.15	3.05
CGL	**47.86**▲	**29.47**▲	**52.15**▲	**18.67**▲	**12.14**▲	**4.99**▲

The underlined and boldfaced results in each column denote the best baseline and the best performer, respectively. The superscript ▲ indicates the improvement of CGL over the best baseline is significant with p-value < 0.01.

sessions on the basis of NARM, CSRM generally outperforms NARM on three datasets. However, CSRM underperforms NARM in terms of MRR@20 on Retailrocket and Diginetica, which may be due to that the unrelated neighbors are easily incorporated. Besides, the GNN-based methods generally work better than the RNN and attention based models, which verifies the effectiveness of GNNs in modeling the complex item transitions in the session. Furthermore, zooming on the GNN-based methods, GCE-GNN and S^2-DHCN present unsatisfactory performance, especially on the Retailrocket dataset, which is attributed to the fact that they easily introduce unrelated items as neighbors in the information propagation when exploring the cross-session information between items. By propagating long-distance information in GNNs and alleviating the overfitting problem, SGNN-HN performs best among the baselines. Thus, in the following experiments, we take SGNN-HN as well as the collaborative neural baselines including CSRM, GCE-GNN and S^2-DHCN for further comparisons.

For CGL, as shown in Table 4, we can observe that CGL can beat the baselines in terms of Recall@20 and MRR@20 on all three datasets. The improvements of CGL over the best baseline SGNN-HN on Retailrocket are 6.21% and 1.31% in terms of Recall@20 and MRR@20, respectively, and the corresponding improvements are 1.44% and 1.69% on the Diginetica dataset. While on Gowalla, CGL outperforms SGNN-HN in terms of Recall@20 and MRR@20 by 24.64% and 27.95%. The improvements can be explained by two facts: As the correlations between sessions are taken into consideration to derive the self-supervision signals for helping the learning of item representations, CGL can generate more accurate recommendations than baselines; In addition, by adopting the target-aware label confusion strategy, the overfitting problem can be effectively alleviated. It is also observed that the improvements of CGL over the best baseline model in terms of both metrics are more significant on Gowalla than those on Retailrocket and Diginetica. This indicates that CGL shows relatively more effectiveness when applied to the check-in scenario by solving the problems of lacking supervision signals and overfitting.

Table 5. Model Performance in Terms of Recall@N and MRR@N with N = 5 and 10

N	Method	Retailrocket		Diginetica		Gowalla	
		Recall@N	MRR@N	Recall@N	MRR@N	Recall@N	MRR@N
5	CSRM	33.91	25.03	24.77	13.92	4.74	2.99
	GCE-GNN	28.57	20.69	26.43	14.96	5.02	2.99
	SGNN-HN	<u>36.37</u>	<u>28.33</u>	<u>28.11</u>	<u>16.08</u>	<u>5.92</u>	<u>3.52</u>
	S^2-DHCN	35.31	26.38	26.52	15.00	4.72	2.61
	CGL	**38.31**▲	**28.49**▲	**28.55**▲	**16.32**▲	**7.49**▲	**4.52**▲
10	CSRM	38.92	25.71	35.69	15.36	6.29	3.19
	GCE-GNN	32.70	21.24	37.69	16.46	6.51	3.19
	SGNN-HN	<u>40.92</u>	<u>28.88</u>	<u>39.23</u>	<u>17.54</u>	<u>7.79</u>	<u>3.77</u>
	S^2-DHCN	40.01	27.02	38.00	16.52	6.80	2.89
	CGL	**43.20**▲	**29.14**▲	**39.87**▲	**17.82**▲	**9.84**▲	**4.83**▲

The underlined and boldfaced results in each column when N = 5 and 10 denote the best baseline and the best performer, respectively. The superscript ▲ indicates the improvement of CGL over the best baseline is significant with p-value < 0.01.

6.1.2 Recommendation Hits at Top Positions. To prove the superiority of CGL above the baselines at top positions, we set the length of the recommendation list (i.e., N) as 5 and 10. Specifically, we test the performance of CGL as well as the competitive baselines, i.e., CSRM, GCE-GNN, SGNN-HN, and S^2-DHCN in terms of Recall@N and MRR@N. The results are presented in Table 5.

From Table 5, we can observe that CGL can consistently achieve the best performance in terms of Recall@N and MRR@N with N = 5 and 10 on all three datasets. This indicates that our proposal can effectively recommend the target item in recommendation lists with different lengths. Moreover, on the Retailrocket dataset, CGL outperforms the best baseline SGNN-HN by 5.33%, 5.57%, and 6.21% in terms of Recall@N with N = 5, 10, and 20, respectively, while the respective improvements in terms of MRR@N with N = 5, 10, and 20 are 0.56%, 0.90%, and 1.31%. It should be noticed that with the number of recommendations increasing, the improvement rate of CGL above the baselines increases. A similar phenomenon can be observed on the Diginetica dataset. This could be explained by the fact that in the e-commerce scenario, the user intent tends to be diversified [8], which makes it difficult for CGL to rank the target item at the top of the recommendation list. However, on the Gowalla dataset, CGL improves the performance against SGNN-HN by 26.52%, 26.32%, and 24.64% in terms of Recall@N with N = 5, 10, and 20, and 28.41%, 28.12%, and 27.95% in terms of MRR@N with N = 5, 10, and 20, respectively. Different from the phenomenon on Retailrocket and Diginetica, we can see that the improvement of CGL against the best baseline becomes obvious when the length of the recommendation list increases. We attribute this to that in the point-of-interest scenario, the user usually focuses on similar check-ins [65], thus CGL can effectively improve the accuracy of recommending the target item in short lists by accurately modeling the user preference.

6.2 Ablation Study

To answer **RQ2**, we conduct an ablation study on three datasets to validate the utility of our proposed two components in CGL, i.e., the self-supervised component and the target-aware label confusion. Specifically, we compare CGL with its two variants, i.e., $\text{CGL}_{w/o[SSL]}$ and $\text{CGL}_{w/o[TLC]}$, corresponding to the CGL model without the self-supervised component and the target-aware label confusion, respectively. Moreover, to validate the effectiveness of the designed target-aware label confusion method, we replace the target-aware label confusion with the label smoothing [35]

Table 6. Ablation Study

Method	Retailrocket		Diginetica		Gowalla	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
CGL	**47.86**	**29.47**	**52.15**	**18.67**	**12.14**	**4.99**
$\text{CGL}_{w/o[SSL]}$	44.76	28.99	51.22	18.57	10.98	4.54
$\text{CGL}_{w/o[TLC]}$	44.16	27.61	49.64	17.92	8.87	3.81
CGL_{LS}	45.47	28.59	50.71	18.56	10.60	4.43

in our proposal, which is denoted as CGL_{LS}.[5] The results of CGL and its three variants are shown in Table 6. As shown in Table 6, we can see that our proposed CGL outperforms $\text{CGL}_{w/o[SSL]}$ and $\text{CGL}_{w/o[TLC]}$ for all cases on three datasets, indicating that either the self-supervised component or the target-aware label confusion can promote the recommendation performance. Besides, removing the label confusion component will cause a more serious performance drop than removing the self-supervised component for all cases on three datasets. This indicates that overfitting is a common and serious problem in different scenarios that limits the recommendation accuracy.

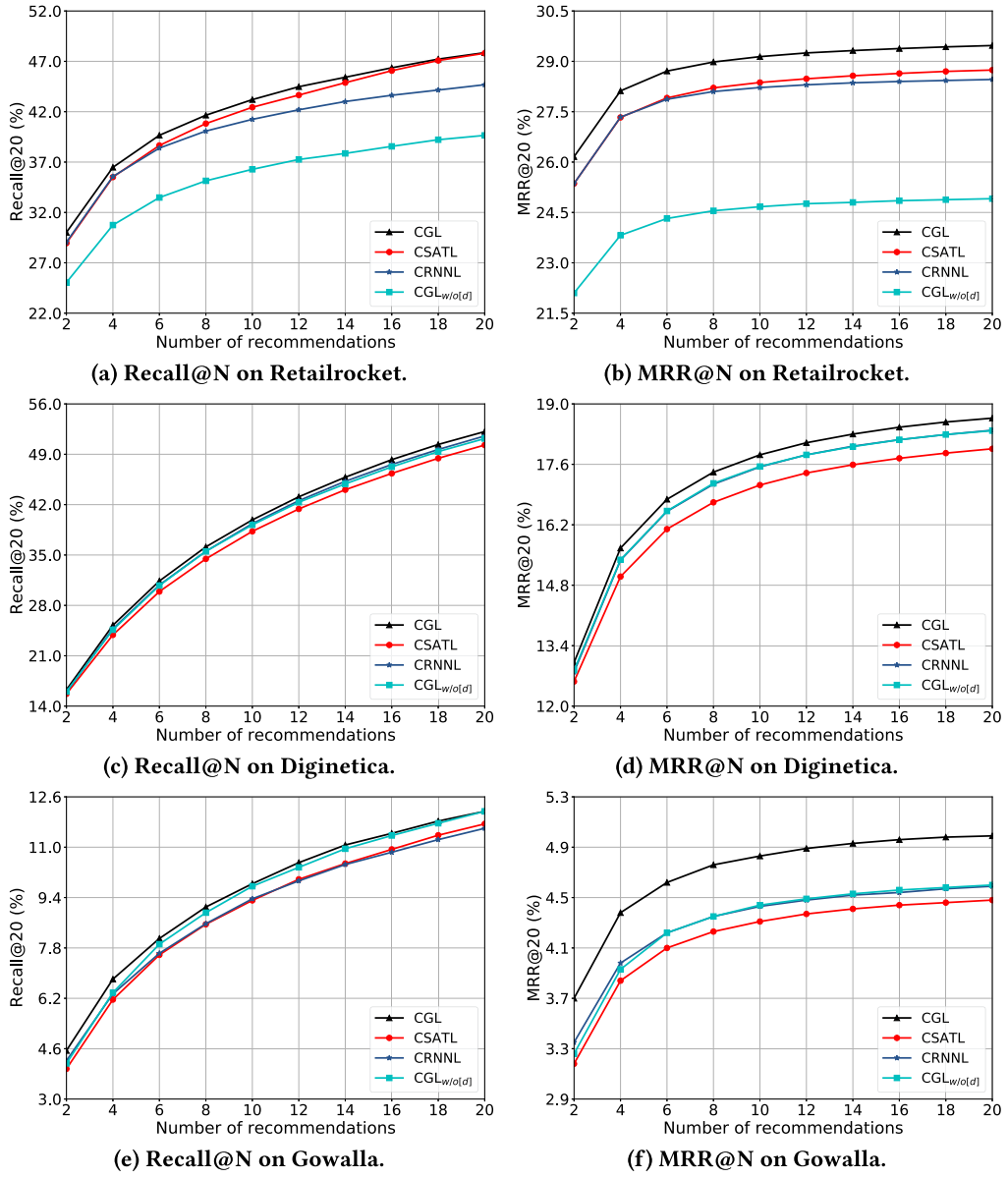
Next, by comparing CGL to $\text{CGL}_{w/o[SSL]}$, it is obvious that after removing the self-supervised component, the performance drops by 6.48% and 1.63% in terms of Recall@20 and MRR@20 on Retailrocket, respectively, and the corresponding decrease rates are 1.82% and 0.54% on Diginetica. On the Gowalla dataset, the corresponding drops are 9.56% and 9.02%, respectively, which are higher than those on Retailrocket and Diginetica. This may be explained by the fact that the average action number per item in Gowalla is smaller than that in Retailrocket and Diginetica as shown in Table 2, which indicates that fewer supervision signals for the model to learn good item representations can be captured in Gowalla than Retailrocket and Diginetica. However, deriving the self-supervision signals can indeed help to alleviate the problem and thus generate accurate item representations as well as accurate recommendations. A similar phenomenon can also be observed when removing the label confusion learning component. This may be because that the number of candidate items to be ranked in the item set of three datasets are different. As shown in Table 2, the number of items in Gowalla is larger than that in Retailrocket and Diginetica, which may easily lead to over-confident because of the one-hot label distribution used in cross-entropy. Thus, the overfitting problem is correspondingly more serious on Gowalla than on Retailrocket and Diginetica. By alleviating the overfitting problem with target-aware label confusion, the performance can be improved more obviously on Gowalla than on Retailrocket and Diginetica.

In addition, by comparing CGL_{LS} to $\text{CGL}_{w/o[TLC]}$, label smoothing can also help to improve the performance, since both the label smoothing and the target-aware label confusion can deal with the serious overfitting problem in SBR. Moreover, target-aware label confusion is more effective than the label-smoothing strategy as CGL can significantly outperform CGL_{LS}. This could be explained by the fact that CGL generates the soft labels based on the similarities of the candidate items with the target item, which is more effective than label smoothing where the soft labels are generated uniformly without being well distinguished.

6.3 Analysis on Graph Neural Networks

As for **RQ3**, to prove the effectiveness of GNNs in our method, we make comparisons from two aspects: (1) We replace the GGNNs in our proposal with the self-attention mechanism (SAT) and RNNs, which are denoted as CSATL and CRNNL, respectively, to prove the utility of GNNs for learning item representations; (2) We replace the dynamic Readout function in the main supervised

[5]The smoothing hyper-parameter is searched in {0.1, 0.2, 0.3, 0.4, 0.5}.



(a) Recall@N on Retailrocket.

(b) MRR@N on Retailrocket.

(c) Recall@N on Diginetica.

(d) MRR@N on Diginetica.

(e) Recall@N on Gowalla.

(f) MRR@N on Gowalla.

Fig. 4. Model performance under the number of recommendations ranging from 2 to 20.

component with the long-term Readout function denoted as CGL$_{w/o[d]}$ to validate the necessity of modeling user's dynamic preference in the main supervised component. We plot the performance of CGL and its variants with various recommendation numbers (ranging from 2 to 20) in Figure 4.

As shown in Figure 4, in general, we can observe that, comparing our proposed CGL to CSATL and CRNNL, CGL can achieve the best performance for all cases on three datasets. This indicates that the graph neural networks can accurately model the item transition relationship in the session, and thus generate accurate item representations. In fact, the self-attention mechanism can also be regarded as a fully connected GNN where each item propagates information from all items

in the session [36], which may lead to the overfitting and the over-smoothing problem [5, 36, 70] and harm the performance of CSATL. In addition, for the RNN-based variant CRNNL, we can observe that its performance is unsatisfactory compared with CGL and CSATL, especially on Retailrocket. This could be explained by the fact that the behavior pattern in the session is much more complicated than the sequential order as presented in [36, 40, 62, 64]. In addition, for the user preference modeling in the self-supervised component, it is not appropriate to incorporate the sequential information since the similarity of two sessions is generated globally, not specially related to the interest migration captured by the RNN.

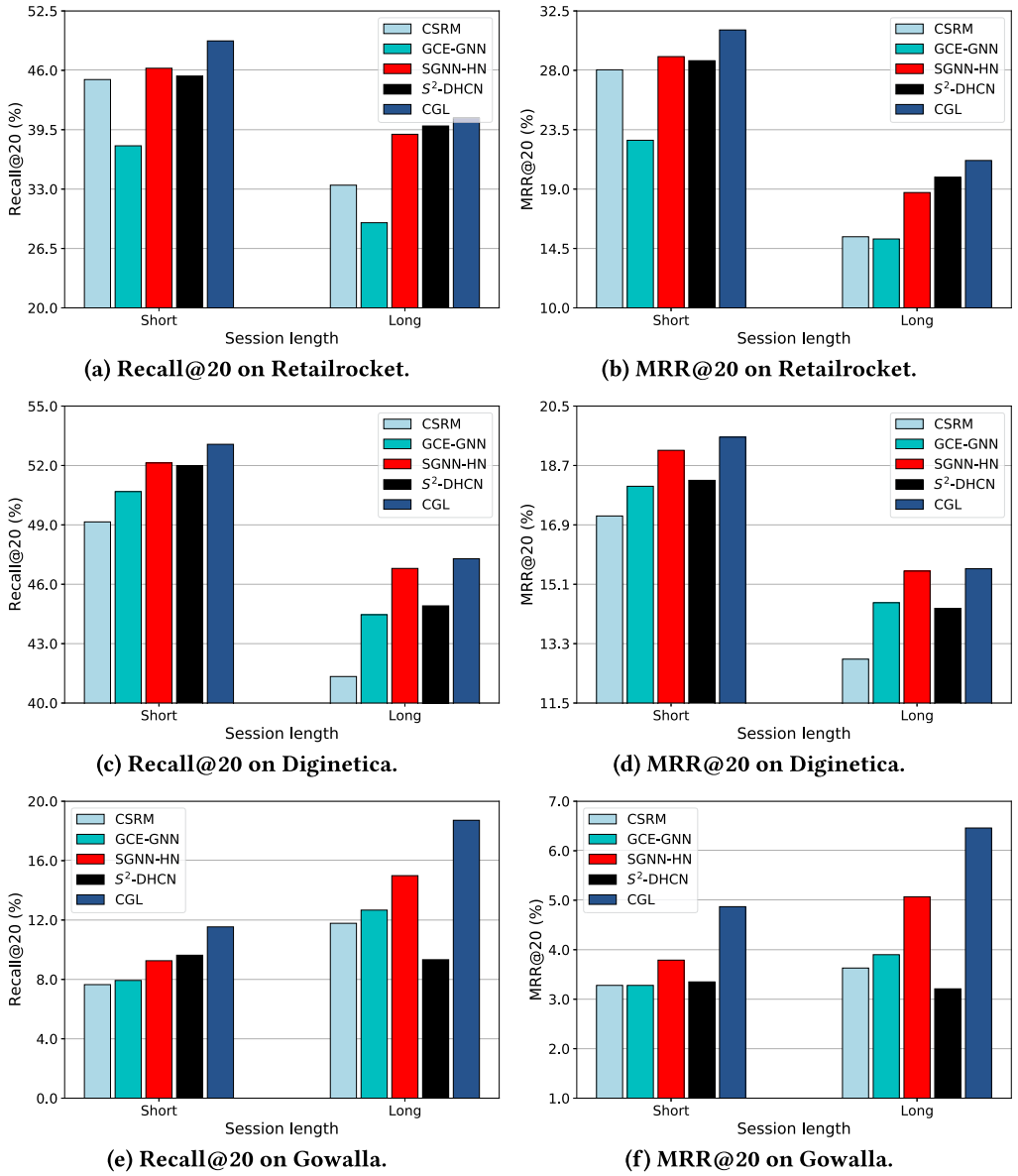
Moreover, by comparing CGL to $\text{CGL}_{w/o[d]}$, on all three datasets, we can see that when replacing the dynamic user preference in the main supervised learning with the long-term interest, the performance will obviously decrease in terms of both metrics, which is consistent with the findings in Section 6.1 and previous work [31, 36, 64]. Moreover, the impacts on Recall@20 and MRR@20 on the e-commerce datasets are similar, which drops by 17.15% and 15.47% on Retailrocket, respectively; and the corresponding decrease rates are 1.90% and 1.58% on Diginetica. However, it is quite different on the Gowalla dataset, respectively returning 0.16% and 7.82%, where a higher improvement is observed in terms of MRR@20. This indicates that in the check-in scenario, capturing user's dynamic interest by emphasizing the recent interaction can help to push the target item at an earlier position more effectively than hitting them in the recommendation list. This can also be validated by the phenomenon that the performance gap between CGL and $\text{CGL}_{w/o[d]}$ becomes large when the number of recommendations goes down. For example, CGL shows 9.93% and 13.80% improvements over $\text{CGL}_{w/o[d]}$ in terms of Recall@2 and MRR@2 on Gowalla, respectively, while the corresponding improvements are 0.82% and 9.01% in terms of Recall@10 and MRR@10.

6.4 Impact of Session Length

As the majority of sessions are short (see Figure 3), and it is meaningful to investigate the impact of the session length on the recommendation accuracy, we compare the performance of CGL with the competitive baselines including CSRM, GCE-GNN, SGNN-HN, and S^2-DHCN. Specifically, we divide the sessions into two groups, i.e., "Short" and "Long", where sessions containing no more than 4 items are deemed as "Short" while the others are regarded as "Long" on Retailrocket and Gowalla, and the threshold is set to 5 on the Diginetica dataset. We adopt such a setting considering that 4, 5, 4 are the respective closest integers to the average session length in Retailrocket, Diginetica, and Gowalla as shown in Table 2. The ratios of sessions in the "Short" and "Long" groups are 83.90% and 16.10% on Retailrocket, respectively, while 84.13% and 15.87% on Diginetica, and 91.58% and 8.42% on Gowalla. The results are presented in Figure 5.

From Figure 5, we can observe that CGL can achieve the best performance for all cases on three datasets, indicating the utility of CGL for modeling the user preference on various session lengths. Moreover, with the session length increasing from "Short" to "Long", the performance of all models decreases on Retailrocket and Diginetica while it increases on the Gowalla dataset. This may be due to that the user intent may be diversified in the e-commerce systems [8]; however, users may focus on similar places in the check-in scenario. Thus, relatively more items in the session on Retailrocket and Diginetica may mislead the models to identify user's current intent, while more check-ins can help to better capture the user interest on Gowalla.

For the e-commerce datasets Retailrocket and Diginetica, we can see that the GNN-based methods SGNN-HN and S^2-DHCN have a competitive performance. Moreover, on Retailrocket, SGNN-HN performs better than S^2-DHCN in terms of both metrics on the "Short" sessions, while it loses the competition on the "Long" sessions. This indicates that incorporating collaborative information from neighbor sessions is important to capture the intent for users who have relatively more interactions. In addition, we can observe that when comparing the performance on "Short" and



(a) Recall@20 on Retailrocket. **(b) MRR@20 on Retailrocket.**

(c) Recall@20 on Diginetica. **(d) MRR@20 on Diginetica.**

(e) Recall@20 on Gowalla. **(f) MRR@20 on Gowalla.**

Fig. 5. Model performance on short and long sessions.

"Long" sessions, the performance drops of MRR@20 are more obvious than those of Recall@20 for all models on both Retailrocket and Diginetica. This indicates that, for long sessions, it is harder to rank the target item earlier than hitting them in the recommendation list.

On the Gowalla dataset, different from that on the e-commerce datasets, we can observe that the performance of the collaborative method S^2-DHCN is unsatisfactory, especially on "Short" sessions. This may be due to that in the check-in setting, bias is easily introduced when incorporating the collaborative information in S^2-DHCN. However, CGL can still show obviously better performance than the baselines, which could be explained by the fact that the item representations can

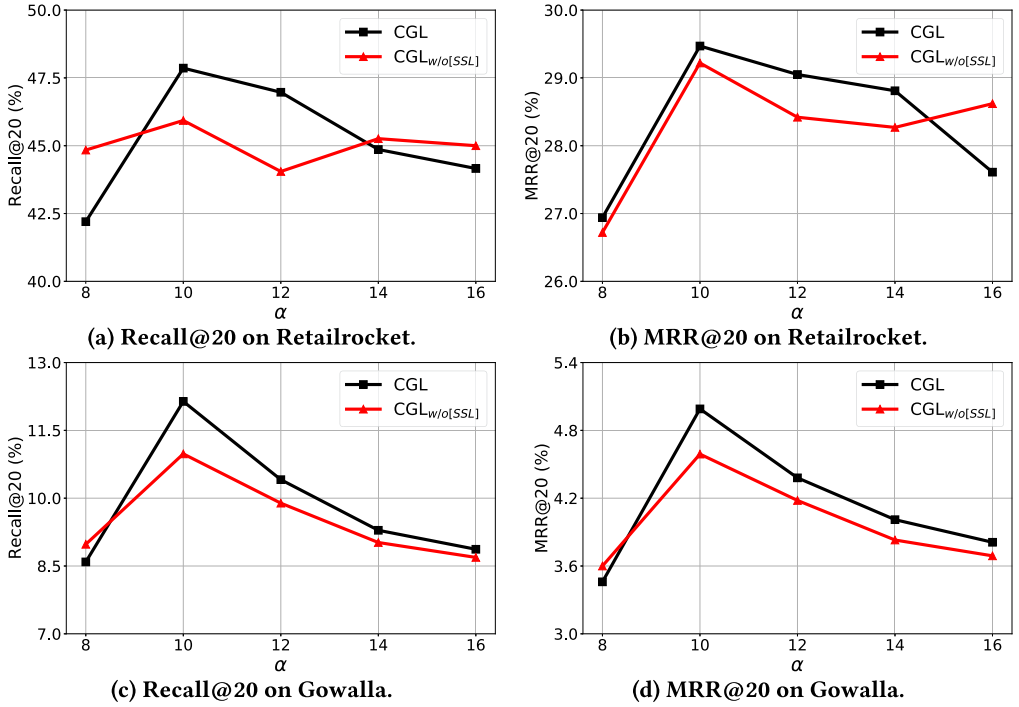


Fig. 6. Model performance with different α in the target-aware label confusion.

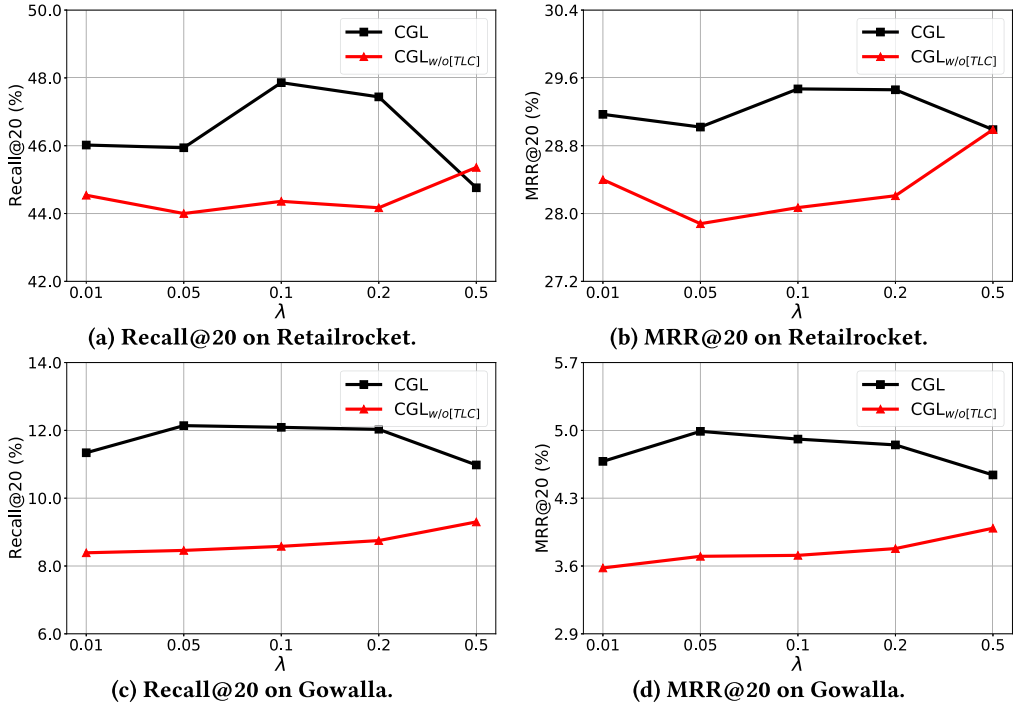
be accurately learned to generate the user preference by introducing the self-supervisions and preventing the overfitting issue.

6.5 Hyper-Parameter Analysis

In this section, we conduct experiments to examine the impact of the key hyper-parameters on the performance of CGL, including the trade-off parameters α and λ introduced in the target-aware label confusion and the multi-task learning, respectively. Moreover, the neighbor number of each session in the self-supervised component is also taken into consideration. Here, the Retailrocket and Gowalla datasets are adopted for experiments, which stand for the e-commerce and check-in recommendation scenarios, respectively.

6.5.1 Impact of the Tradeoff Parameter α. To investigate the impact of the tradeoff parameter α on the model performance, we tune α in $\{8, 10, 12, 14, 16\}$ in the proposal CGL and the variant removing the self-supervised component, i.e., CGL$_{w/o[SSL]}$, to see the impact of α on the recommendation accuracy. We present the results on Retailrocket and Gowalla in Figure 6. As shown in Figure 6, CGL can outperform CGL$_{w/o[SSL]}$ for most cases on two datasets, indicating the utility of the self-supervised learning in scenarios with different degrees of overfitting problem.

On the Retailrocket dataset, we can observe that with α increasing, the performance of both CGL models increases first, achieving a peak performance when α is 10, and then shows a continuous decreasing trend. This may be due to that small α means that the label distribution relies relatively less on the one-hot encoding vector of the target item, thus bias may be introduced to hurt the performance. When α becomes large, the label confusion excessively depends on the hard label, which may lead to overfitting as similar to the problem in the cross-entropy loss stated in Section 1.



(a) Recall@20 on Retailrocket. **(b) MRR@20 on Retailrocket.**

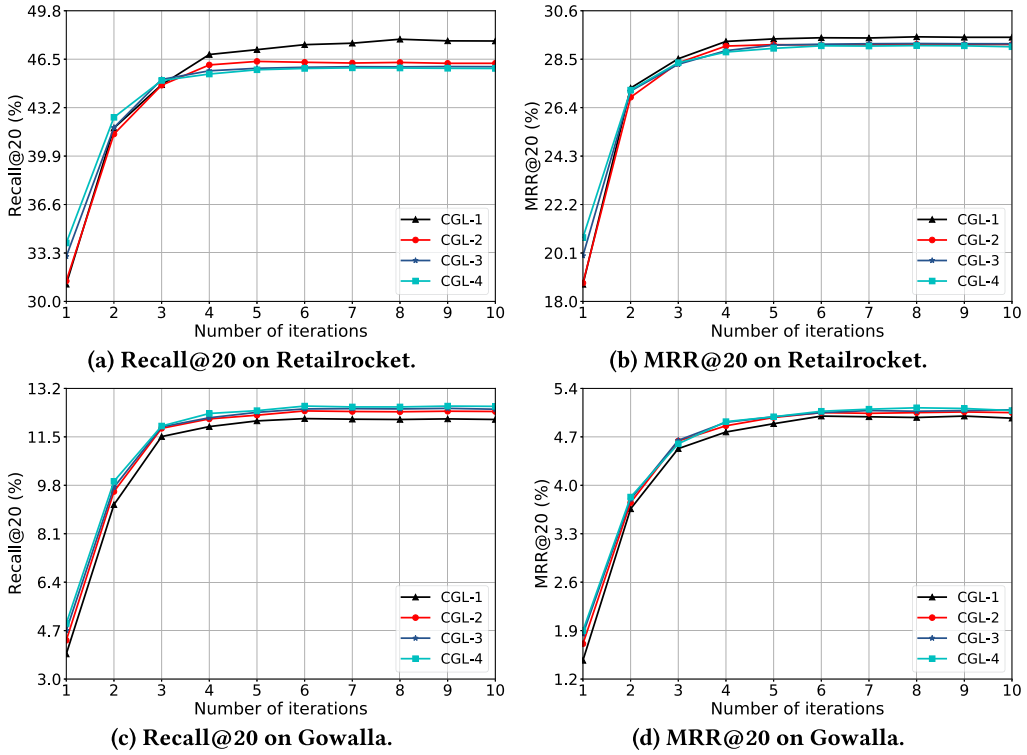
(c) Recall@20 on Gowalla. **(d) MRR@20 on Gowalla.**

Fig. 7. Model performance with different λ for incorporating the self-supervised learning.

In fact, the target-aware label distribution will become the one-hot encoding vector of the target item if α is large enough. A similar phenomenon can be found on the Gowalla dataset. However, for the CGL$_{w/o[SSL]}$ model, the performance in terms of Recall@20 on Retailrocket is stable when α varies. This can be explained by the fact that the incorporated self-supervision signals may aggravate the overfitting problem, thus the target-aware label confusion has a more significant influence on CGL than on CGL$_{w/o[SSL]}$ for hitting the target item in the recommendation list.

6.5.2 Impact of the Tradeoff Parameter λ. To illustrate the influence of the tradeoff parameter λ on the recommendation accuracy, we tune λ in $\{0.01, 0.05, 0.1, 0.2, 0.5\}$ in our proposal CGL and the variant CGL$_{w/o[TLC]}$ that removes the target-aware label confusion to see how the parameter λ affect the recommendation performance of the proposal with and without the label confusion learning. The results on Retailrocket and Gowalla are presented in Figure 7. As shown in Figure 7, we can see that compared to the variant CGL$_{w/o[TLC]}$, CGL can outperform CGL$_{w/o[TLC]}$ on various λ, which indicates that the target-aware label confusion can improve the performance with various magnitudes of self-supervision signals derived from the correlations between sessions incorporated.

Moreover, we can see that on the Retailrocket dataset, with λ increasing, the performance of CGL in terms of Recall@20 and MRR@20 first increases and then shows a decreasing trend. This is due to that when λ is small, the incorporated self-supervision signals are not enough to enhance the item representations, while introducing too much supervision (i.e., large λ) may lead to overfitting and hurt the performance. Thus, the best performance is achieved with a moderate value of λ at 0.1. However, different from CGL, with λ increasing, the performance of CGL$_{w/o[TLC]}$ in terms of both metrics fluctuates at the beginning and then increases continuously. The different trends



(a) Recall@20 on Retailrocket. (b) MRR@20 on Retailrocket.

(c) Recall@20 on Gowalla. (d) MRR@20 on Gowalla.

Fig. 8. Performance of CGL models with different neighbor numbers incorporated for contrasting in the self-supervised learning component.

of CGL and $\text{CGL}_{w/o[TLC]}$ may be due to that by adopting the target-aware label confusion, the main supervised component in CGL can already learn accurate item representations, and does not require much additional supervision signals from the self-supervised component. Thus, a relatively smaller λ is proper for the collaborative learning. As on the Gowalla dataset, a similar phenomenon can be observed except that CGL performs best when λ is 0.05 in terms of both metrics.

6.5.3 Impact of the Neighbor Number. We then study the influence of the number of neighbors incorporated for contrasting (i.e., M) in the self-supervised component. Specifically, we tune the neighbor number in $\{1, 2, 3, 4\}$, where the corresponding CGL models are denoted as CGL-1, CGL-2, CGL-3, and CGL-4, respectively. We present their corresponding performance across different training iterations on Retailrocket and Gowalla in Figure 8.

On the Retialrocket dataset, from Figures 8(a) and 8(b), we can see that the peak performance of CGL-1 can obviously exceed other models, e.g., CGL-4, indicating that increasing the neighbor number will generally decrease the performance of CGL in terms of both Recall@20 and MRR@20. Interestingly, it should be noticed that at the first iterations, CGLs with large neighbor numbers, e.g., CGL-4, show competitive performance and outperform CGL-1 in terms of both metrics. However, CGL-1 exceeds other models with the training iteration increasing. This could be explained by the fact that introducing many neighbors for contrasting in the self-supervised learning component may lead to the supervision redundancy problem, thus making the model overfitting. On Gowalla, different from the phenomenon on Retailrocket, from Figures 8(c) and 8(d), we can observe that with the neighbor number increasing, the performance of CGL models generally

increases. This indicates that relatively more self-supervisions are required on the Gowalla dataset, which is consistent with the finding in Section 6.2.

7 CONCLUSIONS AND FUTURE WORK

In this article, we propose a novel method, i.e., Collaborative Graph Learning (CGL), for session-based recommendation. CGL utilizes the Gated Graph Neural Network (GGNN) to learn the item representations, and is trained by combining the supervised learning and the self-supervised learning. In detail, in the main supervised component, we calculate user's prediction score distribution, which is contrasted with the label distribution generated by the designed target-aware label confusion for optimizing the parameters. In addition, the self-supervised learning is adopted for deriving the additional supervision signals from the correlations between different sessions for item representation learning, which is combined with the main supervised module via the joint learning. Extensive experiments conducted on three benchmark datasets demonstrate that CGL can achieve the state-of-the-art performance in terms of Recall and MRR for the session-based recommendation task.

As to future work, first, we would like to introduce more supervision signals for enhancing the item representations, such as considering the global-level item correlations. Moreover, we are interested in adopting the pretraining strategy to accurately estimate the similarity between items for generating the target-aware label distribution. In addition, we plan to take the multiple behaviors [14, 23] in the e-commerce platforms into consideration for distinguishing user's different interactions with items, thus helping to detect user's instant intent for recommendations.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* 17, 6 (2005), 734–749.

[2] Geoffray Bonnin and Dietmar Jannach. 2014. Automated generation of music playlists: Survey and experiments. *ACM Comput. Surv.* 47, 2 (2014), 26:1–26:35.

[3] Yi Cao, Weifeng Zhang, Bo Song, and Congfu Xu. 2019. HiCAN: Hierarchical convolutional attention network for sequence modeling. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'19)*. 1723–1732.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML'20)*. 1597–1607.

[5] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling information loss of graph neural networks for session-based recommendation. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'20)*. 1172–1180.

[6] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. A dynamic co-attention network for session-based recommendation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'19)*. 1461–1470.

[7] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. Joint neural collaborative filtering for recommender systems. *ACM Trans. Inf. Syst.* 37, 4 (2019), 39:1–39:30.

[8] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and Maarten de Rijke. 2020. Improving end-to-end sequential recommendations with intent-aware diversification. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'20)*. 175–184.

[9] Jacek Dabrowski, Barbara Rychalska, Michal Daniluk, Dominika Basaj, Konrad Goluchowski, Piotr Babel, and Andrzej Michalowski. 2020. An efficient manifold density estimator for all recommendation systems. *arXiv preprint arXiv:2006.01894* (2020).

[10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube video recommendation system. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'10)*. ACM, 293–296.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'19)*. 4171–4186.

[12] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam M. Shroff. 2019. Sequence and time aware neighborhood for session-based recommendations: STAN. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 1069–1072.

[13] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. 2018. Unsupervised representation learning by predicting image rotations. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.

[14] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, and Dawei Yin. 2020. Hierarchical user profiling for e-commerce recommender systems. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'20)*. ACM, 223–231.

[15] Biyang Guo, Songqiao Han, Xiao Han, Hailiang Huang, and Ting Lu. 2021. Label confusion learning to enhance text classification models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'21)*. 12929–12936.

[16] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'19)*. 1569–1577.

[17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the International World Wide Web Conference (WWW'17)*. 173–182.

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

[19] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'18)*. 843–852.

[20] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.

[21] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *Proceedings of the International Conference on Learning Representations (ICLR'19)*.

[22] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'17)*. 306–310.

[23] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. ACM, 659–668.

[24] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'18)*. 197–206.

[25] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*.

[26] Lingpeng Kong, Cyprien de Masson d'Autume, Lei Yu, Wang Ling, Zihang Dai, and Dani Yogatama. 2020. A mutual information maximization perspective of language representation learning. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*.

[27] Solomon Kullback and Richard A. Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86.

[28] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'17)*. 1419–1428.

[29] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'15)*. 811–820.

[30] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated graph sequence neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.

[31] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'18)*. 1831–1839.

[32] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Model. User Adapt. Interact.* 28, 4–5 (2018), 331–390.

[33] Anjing Luo, Pengpeng Zhao, Yanchi Liu, Fuzhen Zhuang, Deqing Wang, Jiajie Xu, Junhua Fang, and Victor S. Sheng. 2020. Collaborative self-attention network for session-based recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'20)*. 2591–2597.

[34] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled self-supervision in sequential recommenders. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'20)*. 483–491.

[35] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. When does label smoothing help? In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'19)*. 4696–4705.

[36] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten de Rijke. 2020. Star graph neural networks for session-based recommendation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'20)*. 1195–1204.

[37] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An intent-guided collaborative machine for session-based recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 1833–1836.

[38] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking item importance in session-based recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 1837–1840.

[39] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Trans. Inf. Syst.* 38, 3 (2020), 22:1–22:23.

[40] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the item order in session-based recommendation with graph neural networks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'19)*. 579–588.

[41] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. GAG: Global attributed graph neural network for streaming session-based recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 669–678.

[42] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'19)*. 4806–4813.

[43] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the International World Wide Web Conference (WWW'10)*. 811–820.

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

[45] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the International World Wide Web Conference (WWW'01)*. 285–295.

[46] Heng-Shiou Sheu and Sheng Li. 2020. Context-aware graph embedding for session-based news recommendation. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'18)*. 657–662.

[47] Heng-Shiou Sheu, Zhixuan Chu, Daiqing Qi, and Sheng Li. 2021. Knowledge-guided article embedding refinement for session-based news recommendation. *IEEE Trans. Neural Networks Learn. Syst.* (2021).

[48] Bo Song, Yi Cao, Weifeng Zhang, and Congfu Xu. 2019. Session-based recommendation with hierarchical memory networks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'19)*. 2181–2184.

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

[50] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'15)*. 2440–2448.

[51] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'19)*. ACM, 1441–1450.

[52] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. 2016. Conditional image generation with pixelCNN decoders. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'16)*. 4790–4798.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'17)*. 5998–6008.

[54] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.

[55] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.

[56] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'15)*. 1235–1244.

[57] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A collaborative session-based recommendation approach with parallel memory modules. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 345–354.

[58] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15)*. 403–412.

[59] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. 2019. Sequential recommender systems: Challenges, progress and prospects. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'19)*. 6332–6338.

[60] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Longbing Cao. 2019. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'19)*. 3771–3777.

[61] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 165–174.

[62] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 169–178.

[63] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the International Conference on Learning Representations (ICLR'15)*.

[64] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'19)*. 346–353.

[65] Yuxia Wu, Ke Li, Guoshuai Zhao, and Xueming Qian. 2019. Long- and short-term preference learning for next POI recommendation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'19)*. 2301–2304.

[66] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2020. Self-supervised hypergraph convolutional networks for session-based recommendation. *arXiv preprint arXiv:2012.06852* (2020).

[67] Xu Xie, Fei Sun, Zhaoyang Liu, Jinyang Gao, Bolin Ding, and Bin Cui. 2020. Contrastive pre-training for sequential recommendation. *arXiv preprint arXiv:2010.14395* (2020).

[68] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2020. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 931–940.

[69] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph contextualized self-attention network for session-based recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'19)*. 3940–3946.

[70] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the International Conference on Machine Learning (ICML'18)*. PMLR, 5449–5458.

[71] Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2020. TAGNN: Target attentive graph neural networks for session-based recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. ACM, 1921–1924.

[72] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.* 52, 1 (2019), 5:1–5:38.

[73] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'20)*. 1893–1902.

Received May 2021; revised August 2021; accepted September 2021