Contrastive Learning for Sequential Recommendation

Xu Xie[1] Fei Sun[2] Zhaoyang Liu[2] Shiwen Wu[1] Jinyang Gao[2]
Jiandong Zhang[2] Bolin Ding[2] Bin Cui[1♯†*]
[1]*School of Computer Science & Key Laboratory of High Confidence Software Technologies (MOE), Peking University*
♯*Center for Data Science, Peking University & National Engineering Laboratory for Big Data Analysis and Applications*
†*Institute of Computational Social Science, Peking University(Qingdao), China*
[2]*Alibaba Group, China*
[1]{*xu.xie, wushw.18, bin.cui*}*@pku.edu.cn*
[2]{*ofey.sf, jingmu.lzy, jinyang.gjy, bolin.ding*}*@alibaba-inc.com*, *chensong.zjd@taobao.com*

2022 IEEE 38th International Conference on Data Engineering (ICDE) | 978-1-6654-0883-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDE53745.2022.00099

***Abstract*—Sequential recommendation methods play a crucial role in modern recommender systems because of their ability to capture a user's dynamic interest from her/his historical interactions. Despite their success, we argue that these approaches usually rely on the sequential prediction task to optimize the huge amounts of parameters. They usually suffer from the data sparsity problem, which makes it difficult for them to learn high-quality user representations. To tackle that, inspired by recent advances of contrastive learning techniques in the computer vision, we propose a novel multi-task framework called Contrastive Learning for Sequential Recommendation (CL4SRec). CL4SRec not only takes advantage of the traditional next item prediction task but also utilizes the contrastive learning framework to derive self-supervision signals from the original user behavior sequences. Therefore, it can extract more meaningful user patterns and further encode the user representations effectively. In addition, we propose three data augmentation approaches to construct self-supervision signals. Extensive experiments on four public datasets demonstrate that CL4SRec achieves state-of-the-art performance over existing baselines by inferring better user representations.**

***Index Terms*—Contrastive Learning, Deep Learning, Recommender Systems**

I. Introduction

Recommender systems [1, 2, 3, 4, 5, 6, 7] have been widely employed in online platforms, e.g., Amazon and Alibaba, to satisfy the users' requirements. On these platforms, users' interests hidden in their behaviors are intrinsically dynamic and evolving over time, which makes it difficult for platforms to make appropriate recommendations. To cope with this problem, various methods have been proposed to make *sequential recommendations* by capturing users' dynamic interests from their historical interactions [8, 9, 10, 11, 12].

For the sequential recommendation task, the essential problem is how to infer a high-quality representation for each user through their historical interactions. With these user representations, we can easily recommend suitable items for each user. Thus, the mainline of research work seeks to derive better user representations by using more powerful sequential models. Recently, with the advances of deep learning techniques [13, 14, 15], a lot of works employ deep neural networks to handle this problem and obtain significant performance improvements [8, 10, 11, 16]. These sequential models, such as Recurrent Neural Network (RNN) [17] and self-attention mechanism [18], can learn more effective representations of users by capturing more complicated sequential patterns. Some previous works also adopt Graph Neural Networks (GNN) [16, 19] to explore more complex item transition patterns in user sequences. Although these methods have achieved promising results, they usually only utilize the item prediction task to optimize these huge amounts of parameters, which is easy to suffer from data sparsity problem [20, 21]. When the training data is limited, these methods may fail to infer appropriate user representations.

Recently, self-supervised learning techniques have made a great breakthrough for the representation learning in domains like computer vision (CV) and natural language processing (NLP) [22, 23, 24, 25]. They attempt to extract intrinsic data correlation directly from unlabeled data. Inspired by the successes of self-supervised learning, we aim to use self-supervised learning techniques to optimize the user representation model for improving sequential recommender systems. To achieve this goal, one straight-forward way could be to directly adopt a powerful sequential model like GPT [23] on a larger user behavior corpus. However, this way is not suitable for the recommender system for two reasons. (i) Recommender systems usually do not have a larger corpus for pre-training. Meanwhile, different from the NLP area, different tasks in the recommender systems usually do not share the same knowledge, thus restricting the application of pre-training. (ii) The objective function of such predictive self-supervised learning is almost the same as the goal of sequential recommendation, which is often modeled as a sequential prediction task. Therefore, if we apply the same item prediction objective in both the pre-training and finetuning procedure like CV and NLP, we could not achieve better performance.

Due to the issues mentioned before, the application of self-supervised learning in recommendation systems is less well studied. The closest line of recent works seeks to enhance feature representations via self-supervision signals derived from the intrinsic structure of the raw feature data, e.g., item

*Corresponding author.

2375-026X/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDE53745.2022.00099

attributes [20, 21]. These previous works usually focus on improving the representations at the item level. However, how to acquire accurate representations in the user behavior sequence level is not well studied.

Different from the previous study focusing on enhancing item representations through the self-supervised task on item features, we aim to learn better sequence representations via self-supervised signals on the user behavior sequence level, even with only identifier (ID) information. Specifically, we propose a novel model-free framework called **C**ontrastive **L**earning for **S**equential **Rec**ommendation (CL4SRec). Our framework combines the traditional sequential prediction objective with the contrastive learning objective. With the contrastive learning loss, we encode the user representation by maximizing the agreement between differently augmented views of the same user interaction sequence in the latent space. In this way, CL4SRec can infer accurate user representations and then easily select appealing items for each user individually. Furthermore, we propose three data augmentation methods (crop/mask/reorder) to project user interaction sequences to different views. We conduct extensive experiments on four real-world public recommendation datasets. Comprehensive experimental results verify that CL4SRec achieves state-of-the-art performance compared with competitive methods.

Our primary contributions can be summarized as follows:

- We propose a novel model-free framework called Contrastive Learning for Sequential Recommendation (CL4SRec). It alleviates the data sparsity problem aroused by complex deep learning models in the sequential recommendation without introducing any auxiliary learnable parameter. Meanwhile, it is also a model-free framework which can be applied on various existing sequential models.
- We propose three different data augmentation approaches, including cropping, masking, and reordering, to construct different views of user sequences.
- Extensive experiments on four public datasets demonstrate the effectiveness of our CL4SRec model. Compared with all competitive baselines, the improvement brought by the contrastive learning framework is 10.71% to 24.30% according to the metric on average.

The rest of the paper is organized as follows. We first list the relevant approaches in section II. In section III, we represent the details of our CL4SRec architecture. Then in secion IV, we provide the evaluation methodology, the datasets characteristics, and the experimental results on four public datasets to support our claims. Finally, we provide some concluding remarks in section V.

II. Related Work

In this section, we will review previous works which are highly related to ours. We will mainly focus on two aspects, including sequential recommendation models and self-supervised learning techniques.

A. Sequential Recommendation

Early works on sequential recommendation usually model the sequential patterns with the Markov Chain (MC) assumption. Rendle et al. [9] combine the first-order MC and matrix factorization and achieve promising results in capturing the subsequent actions. To consider more preceding items and extract more complicated patterns, the high-order MC is also explored in the following work [26]. With the recent advance of deep learning, Recurrent Neural Network (RNN) [17] and its variants, such as Long Short-Term Memory (LSTM) [27] and Gated Recurrent Unit (GRU) [8, 28], are applied to user behavior sequences. For example, Hidasi et al. [8] adopt GRU modules to the session-based recommendation task with the ranking loss function. Later, Hidasi and Karatzoglou [29] improve GRU4Rec with new loss functions and an improved sampling strategy. There are also other following variants modifying GRU4Rec and achieving promising results by introducing attention mechanism [30, 31, 32], hierarchical structure [33], and user-based gated network [28].

Except for recurrent neural network, other deep learning models are also adopted to sequential recommendation tasks and have achieved excellent performance. Tang and Wang [10] propose a Convolutional Sequence Embedding Recommendation Model (Caser). By abstracting the user behaviors as a one-dimensional image, the horizontal and vertical convolutional filters are effective in capturing the sequential patterns in neighbors. Chen et al. [34] and Huang et al. [35] leverage the memory-augmented neural network to store and update useful information explicitly. Recently, the self-attention mechanism [18] has shown promising potential in modeling sequential data. Kang and McAuley [11] utilize Transformer layers to assign weights to previous items adaptively. Sun et al. [12] improve that by adopting a bidirectional Transformer to incorporate information from both directions, since the user's historical interactions may not be a rigid order [36].

Along another line, graph neural network (GNN) [37] has also aroused comprehensive awareness in the deep learning society. Since GNN can capture complicated item transition patterns hidden in user sequences, many works attempt to utilize powerful GNN techniques in sequential recommendation tasks, especially when the user sequence can be constructed as a graph. For example, Wu et al. [16] adopt the gated graph network to aggregate the neighborhood information and make the adjustment for the directed graph. Qiu et al. [19] utilize the attention mechanism as the aggregation function to differentiate the influence of different neighbors instead of mean pooling. To capture both global and local structure, Xu et al. [38] combine self-attention with GNN and achieve promising results for the session-based recommendation.

B. Self-supervised Learning

Recently, various self-supervised learning methods have shown their superior ability on learning useful representations from unlabeled data. Their basic idea is to construct training signals from the raw data with designed pretext tasks. We clas-

sify the existing pretext tasks into two categories, prediction tasks and discrimination tasks.

The prediction tasks have been widely adopted in CV and NLP areas, which usually transform the raw data into another form and then predict the transformation details. In the CV community, various self-supervised learning pretext tasks are proposed, including predicting image rotations [39] and relative patch locations [40], solving jigsaw puzzles [25], and colorization problems [41]. For NLP, most works focus on acquiring universal word representations and effective sentence encoder through self-supervised learning. Language model is naturally a self-supervised objective that learns to predict the next word given the previous sentence [42]. Besides, various pretext tasks have also been proposed recently, e.g., the Cloze task [22] and restoring sentence order [43].

The discrimination tasks adopt a contrastive learning (CL) framework to make a comparison between different samples. Previous works have already achieved promising results in CV area [24, 44, 45]. Even without supervised signals during pretraining, contrastive learning methods can outperform strong baselines on various CV tasks [24]. To make a more difficult comparison, Wu et al. [44] design a shared memory bank to increase the batch size during training, thus enhancing the final performance. Later, He et al. [45] improve that with a dynamic queue and a moving-averaged encoder.

When it comes to recommendation systems, existing works apply self-supervised learning to improve recommendation performance. Yao et al. [20] present a self-supervised learning framework to learn item representations combined with a set of categorical features. Zhou et al. [21]learn the correlations among attributes, items, and sequences via auxiliary self-supervised objectives (S^3-Rec). Wu et al. [46] apply self-supervised learning on user-item graph to improve the accuracy and robustness of GCNs for recommendation (SGL). Unlike these methods, our work focuses on exploring contrastive learning on user behavior sequence to improve user representation learning. Different from the work of Yao et al., our CL4SRec concentrates on how to obtain accurate user representations, which is more difficult since users' interests are usually dynamic. Different from S^3-Rec, our CL4SRec does not need the support of auxiliary item attributes It can enhance the sequential recommendation only with users' interactions. Different from SGL, our CL4SRec does not ignore the sequential information provided in the user interactions.

It is also worth mentioning that though CL has been used in CV, we apply it to recommendation domain with different objectives and techniques. First, the additional self-supervision signals derived from CL can alleviate the data sparse problem aroused by complex deep learning models in recommendations. Since users usually only interact with a few items, the huge amounts of parameters could not be well optimized only by the item prediction signals. Second, there are so many tough challenges to apply CL to sequential recommendation. For example, the data augmentations which are essential for CL could not be borrowed from CV. The augmentations of CV, such as rotation, can be easily designed since the images have good geometric properties. However, it is difficult to design meaningful augmentations for user sequences, since the interval of interactions varies sharply.

III. CL4SRec

In this section, we present our contrastive learning framework for sequential recommendation (CL4SRec), which only utilizes information of users' historical behaviors. We first introduce the notations used in this paper and formulate the sequential recommendation problem in Section III-A. Then, a general contrastive learning framework is introduced in Section III-B. In Section III-C, we propose three augmentation methods to construct the contrastive tasks. In section III-D, we introduce the user representation model used in our approach. Since our CL4SRec is a model-free framework, we take one of the state-of-the-art models—SASRec [11] as an example of our user representation model to introduce the framework. Then, we propose how to train the user representation model via a multi-task learning framework in Section III-E. Finally, we analyze the complexity of CL4SRec framework III-G.

A. Notations and Problem Definition

In this paper, we represent column vectors and matrices by bold italic lower case letters (e.g., $\boldsymbol{u}$, $\boldsymbol{v}$) and bold upper case letters (e.g., $\boldsymbol{R}$), respectively. The j-th row of a matrix $\boldsymbol{R}$ is represented by $\boldsymbol{R}_j^\top$. And we use calligraphic letters to represent sets (e.g., $\mathcal{U}$, $\mathcal{V}$, and $\mathcal{A}$).

Let $\mathcal{U}$ and $\mathcal{V}$ denote a set of users and items, respectively, where $|\mathcal{U}|$ and $|\mathcal{V}|$ denote the number of users and items, repsectively. We represent a user or an item with $u \in \mathcal{U}$ or $v \in \mathcal{V}$. In sequential recommendation tasks, users' behavior sequences are usually in a chronological order. Therefore, we represent the interaction sequence for user u with $s_u = [v_1^{(u)}, \ldots, v_t^{(u)}, \ldots, v_{|s_u|}^{(u)}]$, where $v_t^{(u)}$ denotes the item which user u interacts at the time step t and $|s_u|$ denotes the length of interaction sequence for user u. And $s_{u,t} = [v_1^{(u)}, v_2^{(u)}, \ldots, v_t^{(u)}]$ denotes the subsequence of user u, which only focus on items interacted before $t+1$. Also, let $\mathcal{A}$ denotes a set of augmentations that may be applied in the contrastive learning tasks.

Based on the above notations, we now define the task for the sequential recommendation. It focuses on predicting the most possible item which the user u will interact with at the time step $|s_u|+1$, given her/his historical interaction sequence without any other auxiliary contextual information. It can be formulated as follows:

$$v_u^* = \arg\max_{v_i \in \mathcal{V}} P\Big(v_{|s_u|+1}^{(u)} = v_i \mid s_u\Big). \tag{1}$$

B. Contrastive Learning Framework

Inspired by the SimCLR framework [24] for learning visual representation, we apply contrastive learning algorithms to sequential recommendation to obtain a powerful user representation model. The framework comprises three major components, including **a stochastic data augmentation module**, **a user representation encoder**, and **a contrastive loss function**. The framework is illustrated on the left of Figure 1.

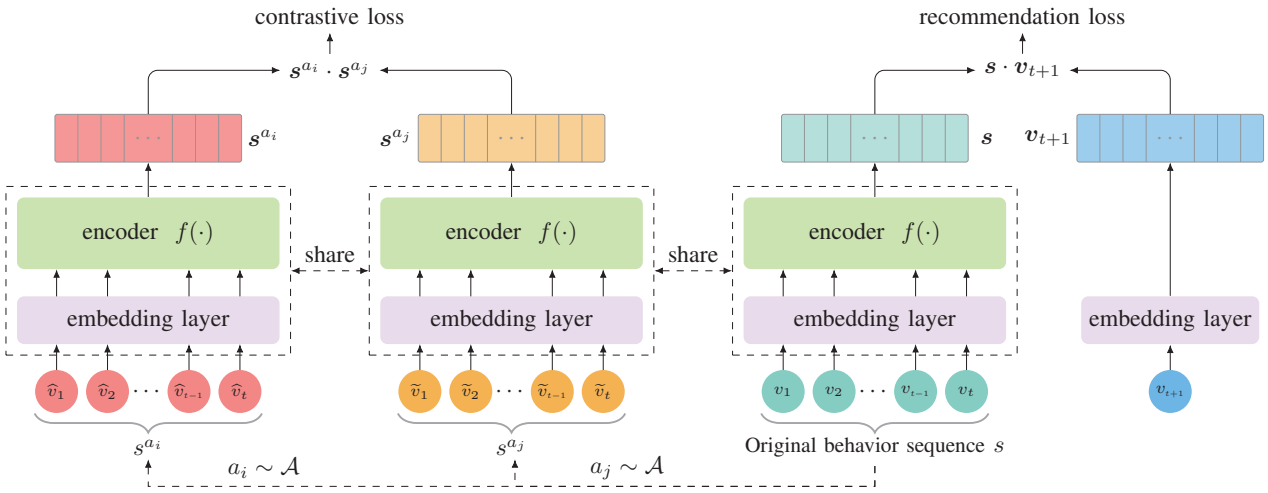


Figure 1: A simple framework for CL4SRec. Two data augmentation methods, a_i and a_j, are sampled from the same augmentation set $\mathcal{A}$. They are applied to each user's sequence and then we can obtain two correlated views of each sequence. A shared embedding layer and the user representation model $f(\cdot)$ transform the original and augmented sequences to the latent space where the contrastive loss and recommendation loss are applied.

1) Data Augmentation Module: A stochastic data augmentation module is employed to transform each data sample randomly into two correlated instances. If the two transformed instances are from the same sample, they are treated as the positive pair. If they are transformed from different samples, they are treated as the negative pair. In the sequential recommendation task, we apply two randomly sampled augmentation methods ($a_i \in \mathcal{A}$ and $a_j \in \mathcal{A}$) to each user's historical behaviors sequence s_u, and obtain two views of the sequence, denoting $s_u^{a_i}$ and $s_u^{a_j}$.

2) User Representation Encoder: We utilize a neural network as a user representation encoder to extract information from the augmented sequences. With this encoder, we can obtain meaningful user representations from their augmented sequences, which is $\boldsymbol{s}_u^a = f(s_u^a)$. Since our CL4SRec has no constraints on the choice of user representation model, in this work, we adopt SASRec [11] to encode the user representation, which has shown promising results in the recent work [11]. It is worth mentioning that SIMCLR also applies an auxiliary non-linear projection module after $f(\cdot)$. However, we find that removing this auxiliary projection leads to a significant improvement in our experiments. In this paper, we discard this additional projection component.

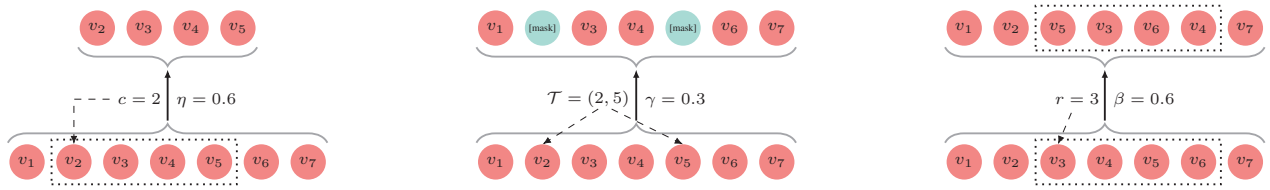
3) Contrastive Loss Function: Finally, a contrastive loss function is applied to distinguish whether the two representations are derived from the same user historical sequence. To achieve this target, the contrastive loss learns to minimize the difference between differently augmented views of the same user historical sequence and maximize the difference between the augmented sequences derived from different users. Considering a mini-batch of N users $u_1, u_2, \ldots, u_N$, we apply two random augmentation operators to each user's sequence and obtain $2N$ augmented sequences $[s_{u_1}^{a_i}, s_{u_1}^{a_j}, s_{u_2}^{a_i}, s_{u_2}^{a_j}, \ldots, s_{u_N}^{a_i}, s_{u_N}^{a_j}]$. Similar to Chen et al. [24] and Yao et al. [20], for each user u, we treat $(s_u^{a_i}, s_u^{a_j})$ as the positive pair, and treat other $2(N-1)$ augmented examples within the same minibatch as negative samples S^-. We utilize dot product to measure the similarity between each representation, $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v}$. Then the loss function $\mathcal{L}_{\text{cl}}$ for a positive pair $(s_u^{a_i}, s_u^{a_j})$ can be defined similarly to the widely used softmax cross entropy loss as:

$$\mathcal{L}_{\text{cl}}(s_u^{a_i}, s_u^{a_j}) = -\log \frac{\exp\left(\text{sim}(\boldsymbol{s}_u^{a_i}, \boldsymbol{s}_u^{a_j})\right)}{\exp\left(\text{sim}(\boldsymbol{s}_u^{a_i}, \boldsymbol{s}_u^{a_j})\right) + \sum_{s^- \in S^-} \exp\left(\text{sim}(\boldsymbol{s}_u^{a_i}, \boldsymbol{s}^-)\right)}. \tag{2}$$

C. Data Augmentation Operators

Based on the above contrastive learning framework, we then discuss the design of the transformations in the contrastive learning framework, which can incorporate additional self-supervised signals to enhance the user representation model. As shown in Figure 2, we introduce three basic augmentation approaches that can construct different views of the same sequence but still maintain the main preference hidden in historical behaviors.

1) Item Crop: Random crop is a common data augmentation technique to increase the variety of images in computer vision. It usually creates a random subset of an original image to help the model generalize better. Inspired by the random crop technique in images, we propose the item crop augmentation method for the contrastive learning task in the sequential recommendation. For each user's historical sequence s_u, we randomly select a continuous sub-sequence: $s_u^{\text{crop}} = [v_c, v_{c+1}, \ldots, v_{c+L_c-1}]$ with the sequence length $L_c =$



(a) item crop (b) item mask (c) item reorder

Figure 2: A brief illustration of augmentation operations applied in our CL4SRec model, including item crop, item mask, and item reorder. Item crop operator randomly selects a continuous sub-sequence of the original behaviors. Item mask operator randomly substitutes a special "mask" item for interacted items. Item reorder operator randomly shuffles a sub-sequence of the original sequences.

$\lfloor \eta * |s_u| \rfloor$. Here, we omit the superscript u for each item v for simplicity. This augmentation method can be formulated as:

$$s_u^{\text{crop}} = a_{\text{crop}}(s_u) = [v_c, v_{c+1}, \ldots, v_{c+L_c-1}]. \tag{3}$$

The effect of our item crop augmentation method can be explained in two aspects. Firstly, it provides a local view of the user's historical sequence. It enhances the user representation model by learning a generalized user preference without comprehensive information of users. Secondly, under the contrastive learning framework, if the two cropped sequences have no intersection, it can be regarded as a next sentence prediction task [22]. It pushes the model to predict the change of the user's preference.

2) Item Mask: The technique of randomly zero-masking input word, which is also called "word dropout", is widely adopted to avoid overfittings in many natural language processing tasks, such as sentence generation [47], sentiment analysis [48], and question answering [49]. Inspired by this word dropout technique, we propose to apply random item mask as one of the augmentation methods for contrastive learning. For each user historical sequence s_u, we randomly mask a proportion γ of items $\mathcal{T}_{s_u} = (t_1, t_2, \ldots, t_{L_\mathtt{m}})$ with the length $L_\mathtt{m} = \lfloor \gamma * |s_u| \rfloor$. Here, t_j is the index of items in s_u which will be masked. If the item in the sequence is masked, it is replaced by a special item [mask]. Therefore, this augmentation method can be formulated as:

$$\begin{aligned} s_u^{\text{mask}} &= a_{\text{mask}}(s_u) = \left[\widehat{v}_1, \widehat{v}_2, \ldots, \widehat{v}_{|s_u|}\right], \\ \widehat{v}_t &= \begin{cases} v_t, & t \notin \mathcal{T}_{s_u} \\ [\text{mask}], & t \in \mathcal{T}_{s_u} \end{cases} \end{aligned} \tag{4}$$

Since a user's intention is relatively stable over a period of time, the user's historical interacted items mostly reflect a similar purpose. For example, if a user intends to purchase a pair of sports shoes, (s)he may click many sports shoes to decide which ones to buy. Thus, with our item mask augmentation, the two different views derived from the same user sequence can still reserve the main intention of the user. In this way, this self-supervised signal can prevent the user representation encoder from co-adapting too much.

3) Item Reorder: Many approaches employ the strict order assumption that most adjacent items in the users' historical sequences are sequentially dependent [8, 9]. However, in the real world, sometimes the order of users' interactions are in a flexible manner [10, 50], due to various unobservable external factors [51]. In such a situation, we can derive a self-supervised augmentation operator to capture the sequential dependencies under the assumption of flexible order. With this operator, we can encourage the user representation model to rely less on the order of interaction sequences, thus enhancing the model to be more robust when encountering new interactions.

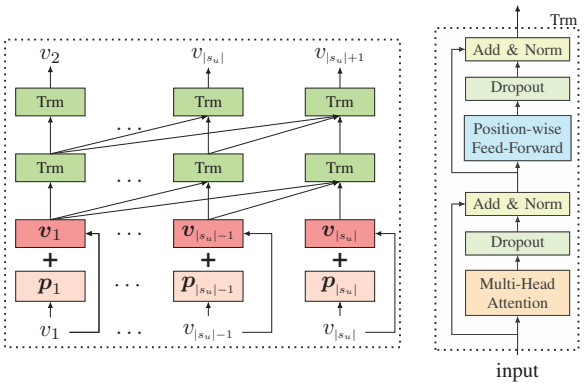
For this purpose, we adopt the item reorder task as another augmentation method for contrastive learning. Inspired by swap operations in the natural language processing [52, 53], we alter the order of items in the user sequence with a proportion of β by randomly shuffling their positions. More concretely, for each user historical sequence s_u, we randomly shuffle a continuous sub-sequence $[v_r, v_{r+1}, \ldots, v_{r+L_\mathtt{r}-1}]$, which starts at r with length $L_\mathtt{r} = \lfloor \beta * |s_u| \rfloor$, to $[\widehat{v}_r, \widehat{v}_{r+1}, \ldots, \widehat{v}_{r+L_\mathtt{r}-1}]$. This augmentation method can be formulated as:

$$s_u^{\text{reorder}} = a_{\text{reorder}}(s_u) = \left[v_1, \ldots, \widehat{v}_r, \ldots, \widehat{v}_{r+L_\mathtt{r}-1}, \ldots, v_{|s_u|}\right]. \tag{5}$$

D. User Representation Model

In this subsection, we describe how we model the user historical sequences by stacking the Transformer encoder. We utilize the architecture of SASRec model in this paper, which applies a unidirectional Transformer encoder and has achieved promising results in the sequential recommendation task [11]. The Transformer encoder consists of three sub-layers: an embedding Layer, a multi-head self-attention module, and a position-wise feed-forward Network, which is briefly illustrated in Figure 3.

1) The Embedding Layer: The Transformer encoder utilizes an item embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ to project high dimensional one-hot item representations to low dimensional dense vectors. In addition, to represent the position information of sequence, it leverages the learnable position embedding $\boldsymbol{P} \in \mathbb{R}^{T \times d}$ to capture this feature of sequences. Note that the number of position vectors T restricts the maximum length of the user's historical sequence. Therefore, when we infer user



(a) SASRec model architecture. (b) Transformer Layer.

Figure 3: A brief architecture of SASRec model and Transformer Encoder Layer. Transformer encoder contains two main components: Multi-Head Self-Attention Layers and Position-wise Feed-Forward Layers.

u representation at time step $t+1$, , we truncate the input sequence $s_{u,t}$ to the last T items if $t > T$:

$$s_{u,t} = [v_{t-T+1}, v_{t-T+2}, \ldots, v_t]\,. \tag{6}$$

Finally, we can obtain the input representations of items in the user sequence by adding the item embedding and position embedding together as:

$$\boldsymbol{h}_i^0 = \boldsymbol{v}_i + \boldsymbol{p}_t, \tag{7}$$

where $\boldsymbol{v}_i \in \boldsymbol{E}$ is the representation of item v_i in the user sequence $s_{u,t}$. Here, we omit the superscript u for convenience.

2) Multi-Head Self-Attention Module: After the embedding layer, the Transformer encoder introduces the self-attention module [18] to capture the dependencies between each item pair in the sequence, which is effective in sequence modeling in many tasks. Moreover, to extract the information from different subspaces at each position, here we adopt the multi-head self-attention instead of a single attention function. It first utilizes different linear projections to project the input representations into h subspaces. Then it applies the self-attention mechanism to each head and derives the output representations by concatenating the intermediates and projecting it once again.

In sequential recommendation, we can only utilize the information before the time step t when we predict the next item v_{t+1}.Therefore, we apply the mask operations to the attention mechanism to discard the connections between $\boldsymbol{Q}_i$ and $\boldsymbol{K}_j$ where $j > i$. These operations can avoid information leakage and are beneficial to the model training.

3) Position-wise Feed-Forward Network: Though multi-head self-attention is beneficial to extract useful information from previous items, it is based on simple linear projections. We endow the model with nonlinearity with a position-wise feed-forward network. It is applied at each position of the above sub-layer's output with shared learnable parameters:

$$\begin{aligned}\mathrm{PFFN}(\boldsymbol{H}^l) &= [\mathrm{FFN}(\boldsymbol{h}_1^l)^\top; \mathrm{FFN}(\boldsymbol{h}_2^l)^\top; \ldots; \mathrm{FFN}(\boldsymbol{h}_{|s_u|}^l)^\top] \\ \mathrm{FFN}(\boldsymbol{h}_i^l) &= \mathrm{RELU}(\boldsymbol{h}_i^l \boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2.\end{aligned} \tag{8}$$

4) User Representations: Based on several Transformer blocks, we obtain the user representation at each time step t, which extracts useful information from the items interacted with before t. Since our task is to predict the item at the time step $|s_u|+1$ for each user u, we set the final representation for user u as her/his preference vector at time $|s_u|+1$:

$$\boldsymbol{s}_u = [\mathtt{Trm}^L(s_u)]_{|s_u|}^\top, \tag{9}$$

where L is the number of stacking Transformer layers. And the $\mathtt{Trm}$ function is the composition of following operations:

$$\begin{aligned}\mathtt{Trm}(\boldsymbol{H}^l) &= \mathrm{LayerNorm}(\boldsymbol{F}^{l-1} + \mathrm{Dropout}(\mathrm{PFFN}(\boldsymbol{F}^{l-1}))) \\ \boldsymbol{F}^{l-1} &= \mathrm{LayerNorm}(\boldsymbol{H}^{l-1} + \mathrm{Dropout}(\mathrm{MH}(\boldsymbol{H}^{l-1}))).\end{aligned} \tag{10}$$

E. Multi-task Training

To leverage the self-supervised signals derived from the unlabeled raw data to enhance the performance of sequential recommendation, we adopt a multi-task strategy where the main sequence prediction task and the additional contrastive learning task are jointly optimized. The total loss is a linear weighted sum as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{main}} + \lambda \mathcal{L}_{\text{cl}}. \tag{11}$$

We adopt the negative log likelihood with binary cross-entropy as the main loss for each user u at each time step $t+1$ as:

$$\mathcal{L}_{\text{main}}(s_{u,t}) = -\log \frac{\exp\left(\boldsymbol{s}_{u,t}^\top \boldsymbol{v}_{t+1}^+\right)}{\exp\left(\boldsymbol{s}_{u,t}^\top \boldsymbol{v}_{t+1}^+\right) + \exp\left(\boldsymbol{s}_{u,t}^\top \boldsymbol{v}_{t+1}^-\right)}, \tag{12}$$

where $\boldsymbol{s}_{u,t}$, $\boldsymbol{v}_{t+1}^+$, and $\boldsymbol{v}_{t+1}^-$ indicate the inferred user representation, the item which user u interacts, and the randomly sampled negative item at the time step $t+1$, respectively.

F. Discussion of Data Sparsity Problem

In this subsection, we conduct a discussion why our proposed CL4SRec can alleviate the data sparsity problem in sequential recommendation tasks. The data sparsity for each dataset can be computed as follows:

$$\text{sparsity} = \frac{\text{No. of total interactions}}{|\mathcal{U}||\mathcal{V}|}. \tag{13}$$

The sparsity value reflects the percentage of interacted items in all items for each user on average. If the dataset is more sparse, it reflects that there are less interactions.

As mentioned above, since previous sequential methods rely on next item prediction loss, $\mathcal{L}_{\text{main}}$, to optimize model parameter, each interaction can only provide one supervision signal for training. The total number of their positive training instances is equal to the number of interactions (except the first

Table I: The comparison of time complexity between SASRec and CL4SRec in training procedure.

Component	SASRec	CL4SRec
Data augmentations	–	$\mathcal{O}(\gamma L_a \lvert\mathcal{U}\rvert)$
Encoding user behaviors	$\mathcal{O}(L_a^2 d \lvert\mathcal{U}\rvert)$	$\mathcal{O}(L_a^2 d \lvert\mathcal{U}\rvert)$
Next item prediction	$\mathcal{O}\big(L_a d \lvert\mathcal{U}\rvert\big)$	$\mathcal{O}\big(L_a d \lvert\mathcal{U}\rvert\big)$
Contrastive loss	–	$\mathcal{O}(Bd \lvert\mathcal{U}\rvert)$

interaction of each user). When the number of interactions of each user is limited, they fail to optimize such a huge amount of learnable parameters well.

To tackle this problem, our proposed CL4SRec framework enriches the signals for optimizing the same learning parameters of base model. For each interaction and its corresponding historical user sequence, we derives self-supervision signals from it as shown in Section III-B. Therefore, the number of training instances can be multiple times than previous works. Therefore, CL4SRec can extremely alleviate the data sparsity problem with the help of $\mathcal{L}_{\text{cl}}$.

Furthermore, it is also worth mentioning that CL4SRec does not introduce any additional learnable parameter. As mentioned in Section III-B, the learnable parameters are only introduced from the user representation component. The number of CL4SRec learnable parameters is the same with these base user representation modules, such as GRU4Rec and SASRec.

G. Complexity Analysis of CL4SRec

In this subsection, we conduct a detailed complexity analysis of CL4SRec. We choose SASRec as the user representation model in this subsection for explanation. It is worth mentioning that other choices of recommendation models can be analyzed similarly. Since our CL4SRec framework does not introduce any auxiliary learnable parameters, the model size of CL4SRec is identical to SASRec. Meanwhile, the procedure of model inference is the same with or without our CL4SRec framework. Therefore, the time complexity does not change during inference.

Then, we analyze the time complexity of training procedure of CL4SRec framework. We select item mask operator for illustration and other augmentations can be analyzed similarly. And we assume the length of each user's sequence is L_a for simplicity. Let γ denotes the mask possibility of item mask operator, B denotes the size of mini-batch, and d denotes the embedding size. The additional operations mainly come from three components: applying data augmentations on original user sequences, deriving user representations from their interactions, and calculating contrastive learning loss.

First, at each epoch, the stochastic item mask augmentation is employed to transform each user sequence randomly into two correlated instances. Therefore, the total complexity is $\mathcal{O}(\gamma L_a |\mathcal{U}|)$.

Second, when it comes to applying sequential models to obtain user representations, the self-attention module induces the most time cost. For simplicity, we only consider the core self-attention operation in our analysis and the number of heads and layers are all set to one. Therefore, the time complexity of this component is $\mathcal{O}(L_a^2 d|\mathcal{U}|)$ for SASRec or CL4SRec , since their time costs are both proportional to the number of user views.

Third, as defined in Equation 2, the main cost of calculating contrastive loss lies in additional inner product operations. Since we treat all other users in the same mini-batch as negative samples, the complexity in each batch is $\mathcal{O}(B^2 d)$. With $\frac{|\mathcal{U}|}{B}$ batches in each epoch, the total complexity is $\mathcal{O}(Bd|\mathcal{U}|)$.

It is worth mentioning that both SASRec and CL4SRec need to compute the next item prediction loss with the time complexity $\mathcal{O}\big(L_a d|\mathcal{U}|\big)$. And we summarize the time complexity in training between SASRec and CL4SRec in Table I. As discussed above, the total time complexity is $\mathcal{O}\big((L_a^2 + L_a)d|\mathcal{U}|\big)$ and $\mathcal{O}\big((L_a^2 + B + L_a)d|\mathcal{U}|\big)$ for SASRec and CL4SRec, respectively[1]. As mentioned above, we observe that the analytical complexity of them are the same in the magnitude. Therefore, our CL4SRec can obtain high-quality user representations with only tolerant additional time cost.

IV. Experiments

In this section, we conduct extensive experiments to answer the following research questions:

RQ1. How does the proposed CL4SRec framework perform compared with the state-of-the-art baselines in the sequential recommendation task?

RQ2. Is CL4SRec really a general framework which can be applied to various sequential models?

RQ3. How do different augmentation methods impact the performance? What is the influence of different augmentation hyper-parameters on CL4SRec performance?

RQ4. How does the weight λ of contrastive learning loss impact on the performance under the multi-task framework?

RQ5. How do different components of CL4SRec benefit its performance, i.e., augmentation methods and contrastive learning loss?

RQ6. Does our CL4SRec really learn a better representation in the user behavior sequence level compared with other state-of-the-art baselines?

A. Experiments Settings

1) Datasets: We conduct experiments on four public datasets collected from the real-world platforms. Two of them are obtained from Amazon, one of the largest e-commercial platforms in the world. They have been introduced in [54], which are split by top-level product categories in amazon.

[1]The time cost of augmentations, $\mathcal{O}(\gamma L_a|\mathcal{U}|)$, is omitted since it has a lower complexity order.

Table II: Dataset statistics (after preprocessing).

Dataset	#users	#items	#actions	avg.length	density
Beauty	22,363	12,101	198,502	8.8	0.07%
Sports	25,598	18,357	296,337	8.3	0.05%
Yelp	30,983	29,227	321,087	10.3	0.04%
ML-1M	6,040	3,953	1,000,209	165.6	4.19%

In this work, we follow the settings in [21] and adopt two categories, "Beauty" and "Sports and Outdoors" Another dataset is collected by Yelp [2], which is a famous business recommendation platform for restaurants, bars, beauty salons, etc. We follow the settings in [21] and use the transaction records after January 1st, 2019. The last dataset is MovieLens-1M[3] (**ML-1M**) [55] dataset, which is widely used for evaluating recommendation algorithms.

For dataset preprocessing, we follow the common practice in [11, 21]. We convert all numeric ratings or presence of a review to "1" and others to "0". Then, for each user, we discard duplicated interactions and then sort their historical items by the interacted time step chronologically to obtain the user interacted sequence. It is worth mentioning that to guarantee each user/item with enough interactions, we follow the preprocessing procedure in [9, 21], which only keeps the "5-core" datasets. We discard users and items with fewer than 5 interaction records iteratively. The processed data statistics are summarized in Table II.

2) Evaluation: We adopt the leave-one-out strategy to evaluate the performance of each method, which is widely employed in many related works [11, 21, 56]. For each user, we hold out the last interacted item as the test data and utilize the item just before the last as the validation data. The remaining items are used for training. To speed up the computation of metrics, many previous works use sampled metrics and only rank the relevant items with a smaller set of random items. However, this sample operation may lead to inconsistent with their non-sampled version as illustrated by Krichene and Rendle [57]. Therefore, we evaluate each method on the whole item set without sampling and rank all the items that the user has not interacted with by their similarity scores.

We employ Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Precision (Prec.) to evaluate the performance of each method which are widely used in related works [9, 11]. $\mathbf{HR@}\boldsymbol{k}$ focuses on the presence of the positive items. It measures the proportion of which the target item is recommended among the top-k items. It is worth mentioning that $\mathrm{HR@}k$ does not consider the actual rank of item v. In contrast, $\mathbf{NDCG@}\boldsymbol{k}$ further takes the rank position information into account. It is a position-aware metric which assigns larger weights on higher ranks. $\mathbf{Prec.}$ is a metric to evaluate the accuracy performance. It focuses on whether the recommended top-1 item is exactly the target item. It can be computed as $\mathbf{HR@1}$

In this work, we report HR and NDCG with $k = 5, 10, 20$ for each method to make fair comparison.

3) Baselines: To verify the effectiveness of our method, we compare it with the following representative baselines:

- **Pop**. It is a non-personalized approach which recommends the same items for each user. These items are the most popular items which have the largest number of interactions in the whole item set.
- **BPR-MF** [58]. It is one of the representative non-sequential baselines. It utilizes matrix factorization to model users and items with the pairwise Bayesian Personalized Ranking (BPR) loss.
- **NCF** [56].It employs a neural network architecture to model non-sequential user-item interactions instead of the inner product used by matrix factorization.
- **GRU4Rec$^+$** [8, 29]. It applies GRU modules to model user sequences for session-based recommendation with ranking loss and is improved with a new class of loss functions and sampling strategy.
- **SASRec** [11]. It is one of the state-of-the-art baselines to solve the sequential recommendation task. It models user sequences through self-attention modules to capture users' dynamic interests.
- **GC-SAN** [38]. It combines GNN with self-attention mechanism to capture both local and long-range transitions of neighbor items hidden in each interaction session.
- **$\mathbf{S^3}$-Rec$_{\mathrm{MIP}}$** [21]. It also utilizes the self-supervised learning methods to derive the intrinsic data correlation. However, it mainly focuses on how to fuse the context data and sequence data. In this section, we only compare the mask item prediction (MIP) in S^3-Rec for fairness.

4) Implementation Details: We utilize the public implementation of BPR-MF, NCF, and GRU4Rec provided by Wang et al. [59][4] in PyTorch. For other methods, we implement them by PyTorch as well. For all models with learnable embedding layers, we set the embedding dimension size $d = 64$, following previous works [11, 12, 21]. When computing the next item prediction loss, we follow the common practice and randomly sample a negative item for each interaction [10, 11, 38, 56]. For each baseline, all other hyper-parameters are set following the suggestions from the original settings in their papers and we report each baseline performance under its optimal settings.

When it comes to our CL4SRec method, we initialize all parameters by the truncated normal distribution in the range $[-0.01, 0.01]$. We use Adam optimizer [60] to optimize the parameters with the learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and linear decay of the learning rate, where the batch size is set to 256. We train the model with early stopping techniques according to the performance on the validation set. To investigate

[2]https://www.yelp.com/dataset

[3]https://grouplens.org/datasets/movielens/1m/

[4]https://github.com/THUwangcy/ReChorus

Table III: Performance comparison of different methods on top-N recommendation. Bold scores are the best in method group, while underlined scores are the second best. Improvements over baselines are statistically significant with $p < 0.01$. The reported result of CL4SRec for each dataset is the best result of applying one of the augmentations.

Datasets	Metric	Pop	BPR-MF	NCF	GRU4Rec$^+$	SASRec	GC-SAN	S^3-Rec$_{MIP}$	CL4SRec	Improv.
Beauty	Prec.	0.0008	0.0021	0.0022	0.0038	0.0051	0.0040	0.0045	**0.0062**	21.57%
	HR@5	0.0072	0.0124	0.0134	0.0169	0.0224	0.0165	0.0183	**0.0272**	21.42%
	HR@10	0.0114	0.0263	0.0254	0.0306	0.0435	0.0289	0.0305	**0.0462**	6.21%
	HR@20	0.0195	0.0461	0.0463	0.0518	0.0710	0.0483	0.0503	**0.0753**	6.06%
	NDCG@5	0.0040	0.0069	0.0075	0.0102	0.0128	0.0102	0.0117	**0.0165**	28.90%
	NDCG@10	0.0053	0.0114	0.0116	0.0146	0.0186	0.0142	0.0156	**0.0227**	22.04%
	NDCG@20	0.0073	0.0163	0.0165	0.0199	0.0255	0.0190	0.0205	**0.0300**	17.65%
Sports	Prec.	0.0022	0.0023	0.0025	0.0031	0.0042	0.0031	0.0024	**0.0056**	33.33%
	HR@5	0.0055	0.0097	0.0098	0.0116	0.0151	0.0110	0.0107	**0.0186**	23.18%
	HR@10	0.0090	0.0181	0.0183	0.0189	0.0271	0.0178	0.0195	**0.0312**	15.13%
	HR@20	0.0149	0.0316	0.0296	0.0308	0.0463	0.0303	0.0328	**0.0489**	5.62%
	NDCG@5	0.0040	0.0060	0.0061	0.0073	0.0090	0.0071	0.0065	**0.0121**	34.44%
	NDCG@10	0.0051	0.0087	0.0090	0.0097	0.0119	0.0093	0.0093	**0.0161**	35.29%
	NDCG@20	0.0066	0.0121	0.0116	0.0126	0.0167	0.0124	0.0127	**0.0206**	23.35%
Yelp	Prec.	0.0015	0.0030	0.0032	0.0026	0.0035	0.0025	0.0032	**0.0046**	31.43%
	HR@5	0.0056	0.0127	0.0137	0.0110	0.0142	0.0124	0.0143	**0.0182**	27.27%
	HR@10	0.0095	0.0226	0.0244	0.0208	0.0252	0.0210	0.0245	**0.0331**	31.35%
	HR@20	0.0160	0.0387	0.0419	0.0366	0.0434	0.0364	0.0429	**0.0565**	30.18%
	NDCG@5	0.0036	0.0078	0.0082	0.0074	0.0089	0.0073	0.0085	**0.0113**	26.97%
	NDCG@10	0.0049	0.0110	0.0117	0.0103	0.0124	0.0101	0.0118	**0.0161**	29.84%
	NDCG@20	0.0065	0.0150	0.0160	0.0139	0.0170	0.0139	0.0164	**0.0220**	29.41%
ML-1M	Prec.	0.0029	0.0026	0.0018	0.0167	0.0174	0.0104	0.0175	**0.0194**	10.86%
	HR@5	0.0078	0.0141	0.0123	0.0874	0.0861	0.0757	0.0912	**0.0990**	8.56%
	HR@10	0.0162	0.0301	0.0270	0.1540	0.1555	0.1430	0.1604	**0.1694**	5.61%
	HR@20	0.0402	0.0596	0.0599	0.2526	0.2608	0.2404	0.2679	**0.2705**	0.97%
	NDCG@5	0.0052	0.0081	0.0068	0.0522	0.0516	0.0430	0.0539	**0.0586**	8.72%
	NDCG@10	0.0079	0.0133	0.0116	0.0736	0.0739	0.0647	0.0761	**0.0812**	6.70%
	NDCG@20	0.0139	0.0206	0.0198	0.0983	0.1005	0.0902	0.1031	**0.1066**	3.40%

Table IV: Experiments results of whether applying CL4SRec on GRU4Rec$^+$ and SASRec. Best results of three augmentations are reported. HR@10 and NDCG@10 are selected as metrics.

Datasets		GRU4Rec$^+$			SASRec		
		Prec.	HR@10	NDCG@10	Prec.	HR@10	NDCG@10
Beauty	w/o CL4SRec	0.0038	0.0306	0.0146	0.0051	0.0435	0.0186
	w/ CL4SRec	0.0057	0.0392	0.0193	0.0062	0.0462	0.0227
	Improvement	50.00%	28.10%	32.19%	21.57%	6.21%	22.04%
Sports	w/o CL4SRec	0.0031	0.0189	0.0097	0.0042	0.0271	0.0119
	w/ CL4SRec	0.0041	0.0253	0.0129	0.0056	0.0312	0.0161
	Improvement	32.26%	33.86%	32.99%	33.33%	15.13%	35.29%
Yelp2019	w/o CL4SRec	0.0026	0.0208	0.0103	0.0035	0.0252	0.0124
	w/ CL4SRec	0.0040	0.0305	0.0149	0.0046	0.0331	0.0161
	Improvement	53.85%	46.63%	44.66%	31.43%	31.35%	29.84%
ML-1M	w/o CL4SRec	0.0167	0.1540	0.0736	0.0174	0.1555	0.0739
	w/ CL4SRec	0.0170	0.1544	0.0730	0.0194	0.1694	0.0928
	Improvement	1.80%	0.26%	-0.082%	11.49%	5.61%	6.70%

the effect of each two augmentation method combination, we fixed the augmentation methods used in CL4SRec each time. And we test the crop/mask/reorder proportion of items from 0.1 to 0.9. For our user representation model of CL4SRec, we stack 2 self-attention blocks together and set the head number as 2 for each block. For a fair comparison, following the settings in [21], the maximum sequence length of T is set to 50 for all datasets.

B. Overall Performance Comparison (RQ1)

To answer **RQ1**, we compare the performance of our CL4SRec with the above baselines. Table III summarizes the best results of all models on four datasets. Note that the improvement columns are the performance of CL4SRec relative to the second best baselines. Due to the space limitation, the results of CL4SRec reported in Table III are the best results of using one of the three augmentations.

Based on the experiment results, we can observe that:

- The non-personalized method Pop exhibits the worst performance on all datasets since it ignores users' unique

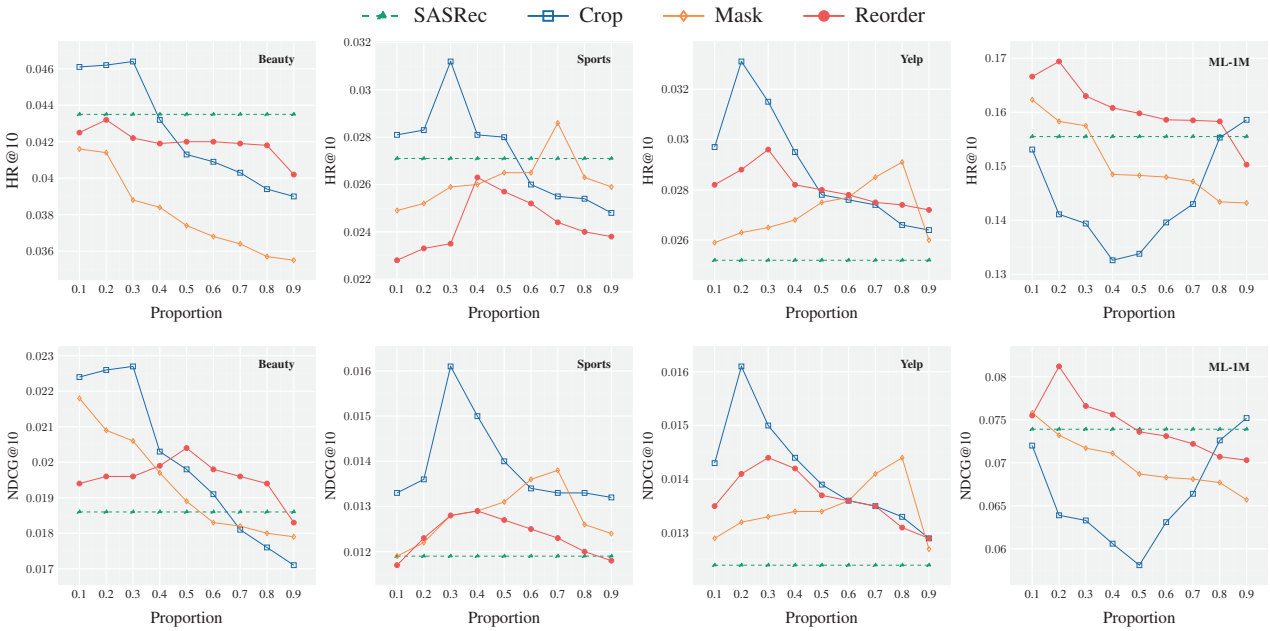


Figure 4: Impact of the different augmentations with different proportion η, γ, and β on HR@10 and NDCG@10. Other hyper-parameters are fixed when tuning the augmentation proportion parameter. And we conduct the same experiments many times and report the average results. The dash line is the performance of the SASRec method.

preferences hidden in their historical interactions. Considering other baseline methods on most datasets, we observe that sequential methods (e.g., GRU4Rec$^+$ and SASRec) outperform non-sequential methods (e.g., BPR-MF and NCF) consistently. Compared with those non-sequential methods, these sequential methods utilize sequential information of users' historical interactions, which contributes to improving performance in recommendation systems. Among all the sequential models, SASRec achieves state-of-the-art performance on all datasets, which indicates that the powerful self-attention mechanism is suitable for capturing sequence patterns.

- When it comes to the GC-SAN, it does not achieve obvious improvements and even sometimes exhibits worse compared to the SASRec in our experimental settings. One possible reason is that no rings are existing in each user sequence and the degree of each interaction is less than two in these datasets with the pre-processing precedure. GNN cannot capture auxiliary useful information in this kind of sequences, but increases the burden of optimization. For S^3-Rec$_{\text{MIP}}$, it also sometimes exhibits worse compared to the SASRec. Without auxiliary contextual information, it pre-trains and finetunes the base model on the exact same dataset, which may lead to catastrophic forgetting [61].
- Finally, according to the results, it is obvious that our proposed CL4SRec outperforms all baselines on all datasets in terms of all the evaluation metrics. It achieves improvements on both sparse (e.g., Sports and Yelp) and dense datasets (e.g., ML-1M), especially on the sparse datasets. CL4SRec gains 14.58% on HR@10, 10.71% on HR@20, 23.46% on NDCG@10, 18.45% on NDCG@20, and 24.30% on Prec. on average against the start-of-the-art baselines. These experiments verify the effectiveness of our CL4SRec method in the sequential recommendation task. Different from S^3-Rec$_{\text{MIP}}$ which still focuses on sequence prediction issue, we adopt the contrastive learning framework to introduce other information, which enhances the user representation model to capture more accurate user representations.

C. Applied on Different Sequential Models (RQ2)

To answer RQ2, we analyze whether our CL4SRec is a model-free framework which can enhance the performance of various basic sequential user representation models. In this subsection, we select GRU4Rec$^+$ and SASRec, which are representative sequential models, to verify the flexibility of our proposed CL4SRec framework. It is worth mentioning that many other sequential models can also serve as the user representation models as well. Meanwhile, due to the space limitation, we also only report the best results with one of the three augmentations. HR@10, NDCG@10, and Precision are reported as metrics for these experiments.

Table IV summarizes the best results of all models on four datasets. The improvement columns are the performance of CL4SRec relative to its Basemodel. CL4SRec gains 27.21%, 14.58% on HR@10, 27.26%, 23.46% on NDCG@10 on

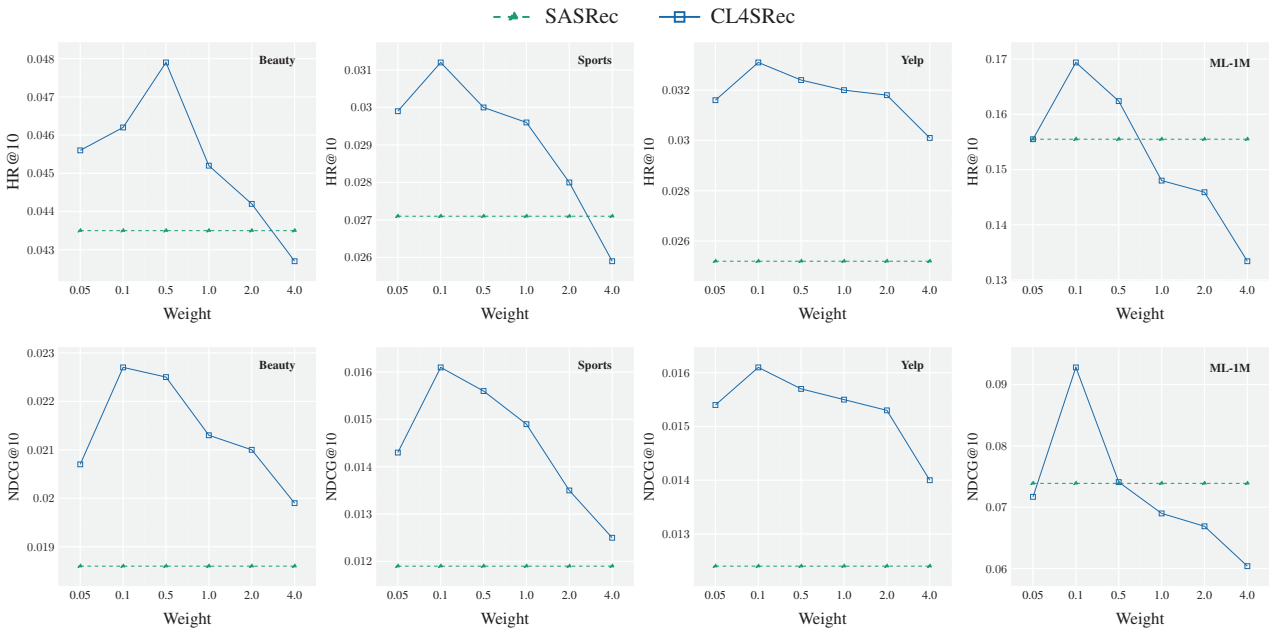


Figure 5: Performance comparison on CL4SRec w.r.t. different λ. The dash line is the performance of SASRec.

average, and 34.48%, 24.45% on Precision against the basic GRU4Rec$^+$ and SASRec, respectively. It demonstrates that applying our CL4SRec framework can achieve significant improvements on various kinds of sequential models. Therefore, our CL4SRec is really a general framework which can incorporated into existing sequential models to enhance encoding user representations.

D. Comparison of Different Augmentation Methods (RQ3)

To answer **RQ3**, we analyze how different augmentation operators and their proportion rates impact the performance. To examine the effect of each augmentation operator, we only utilize one kind of operator in the contrastive learning task with the same proportion parameters each time. Note that the higher rate of γ (mask) and β (reorder) and lower rate of η (crop) in augmentations will make it more difficult to distinguish instances transformed from the same sequence from negative samples. We report HR@10 and NDCG@10 as metrics for the experiments on the Sports and Yelp datasets due to the space limitation.

Figure 4 demonstrates the performance of different augmentation methods with different proportion rates η, γ, and β varing from 0.1 to 0.9. We observe a few trends of different augmentation methods with the variation of proportion rates. First, the performance of CL4SRec equipped with any augmentation method can outperform the SASRec baseline on most datasets for most choices of proportion rates. It indicates the effectiveness of the proposed augmentation methods, since they all introduce auxiliary self-supervised signals hidden in raw data. And we also observe that none of the three augmentation operations can always achieve the best performance compared with other augmentations. For example, Crop operation achieves the best results on the Sports dataset, but the Reorder operation achieves the best results on the ML-1M dataset. This demonstrates that different augmentation methods are appropriate for different datasets since they focus on different aspects of the raw data.

Second, we observe how the proportion rates of different augmentation methods affect the recommendation performance. Considering the item crop and item mask operators, a general pattern is that the performance peaks at a special proportion rate and then deteriorates if we increase or decrease the rate. For example, item mask operator peaks at the proportion rate of 0.8 on the Yelp dataset. It can be explained that when the proportion rate γ equals to 0, item mask operator does not function and when γ equals to 1.0, the whole user sequence only consists of $[mask]$ items, thus hurting the performance. Item crop operator acts similarly to item mask operation, except that item crop operator does not function with $\eta = 1.0$ and the user sequence is empty with $\eta = 0$.

E. Impact of Contrastive Learning Loss (RQ4)

To answer **RQ4**, we investigate how the contrastive learning loss of our proposed CL4SRec interacts with the sequential prediction loss. Specifically, we explore how different λ impacts the recommendation performance. We select the best augmentation method with best proportion rate for each dataset according to the results in Section 4.3 and keep other parameters fixed to make a fair comparison.

Figure 5 shows the evaluation results. Note that with larger value λ, $\mathcal{L}_{\text{cl}}$ contributes more heavily in the $\mathcal{L}_{\text{total}}$. We observe that performance substantially deteriorates when λ increases

Table V: Performance comparison between SASRec, $\text{SASRec}_{\text{aug}}$, and CL4SRec on the metrics HR@10 and NDCG@10. $\text{SASRec}_{\text{aug}}$ is a variant of SASRec which directly applies our proposed data augmentations on the original interactions during training procedure.

Aug	Method	Sports		Yelp	
		HR@10	NDCG@10	HR@10	NDCG@10
N/A	SASRec	0.0271	0.0119	0.0252	0.0124
Crop	$\text{SASRec}_{\text{aug}}$	0.0288	0.0136	0.0292	0.0142
	CL4SRec	0.0312	0.0161	0.0331	0.0161
Mask	$\text{SASRec}_{\text{aug}}$	0.0275	0.0132	0.0269	0.0129
	CL4SRec	0.0286	0.0138	0.0291	0.0144
Reorder	$\text{SASRec}_{\text{aug}}$	0.0253	0.0122	0.0262	0.0127
	CL4SRec	0.0263	0.0129	0.0296	0.0144

over certain threshold. However, with proper λ, CL4SRec is consistently better than SASRec. This observation implies that when contrastive learning loss dominates the learning process, it may decrease the performance on the sequence prediction task. We will analyze this impact carefully in future work.

F. Ablation Study (RQ5)

We perform an ablation study on CL4SRec by showing how the augmentation methods and contrastive learning loss affect its performance. To verify the effectiveness of each component, we conduct experiments on the variant of SASRec, called $\text{SASRec}_{\text{aug}}$, which enhances the behavior sequences using our proposed augmentation methods during the training procedure. All experiments are also conducted under the best hyperparameter settings for each method as mentioned above.

Table V shows the results of SASRec, $\text{SASRec}_{\text{aug}}$, and CL4SRec. We observe a few trends in this table. On the one hand, we find that $\text{SASRec}_{\text{aug}}$ outperforms SASRec in most datasets with most of augmentations. This indicates that our augmentation methods are very useful for basic models by adding random noises. We also observe that some augmentations deteriorate the performance on some datasets, which demonstrates that different augmentations suits different scenarios. And we will further analyze this phenomenon in the future work. On the other hand, with our proposed contrastive learning loss, CL4SRec consistently outperforms $\text{SASRec}_{\text{aug}}$ in all datasets with all augmentations. It verifies the effectiveness of our contrastive learning component for providing significant self-supervised signals, as illustrated above.

G. Quality of User Representations (RQ6)

As mentioned above, CL4SRec can infer better representations in the user behavior sequence level compared with other state-of-the-art baselines. In this subsection, we attempt to verify whether CL4SRec can really cope with this problem. We utilize the friend relations provided by Yelp dataset to explore whether users are closer to each other in the latent space if they are friends. Cosine function is applied to measure the similarity. Note that we do not use this kind of information during the training procedure.

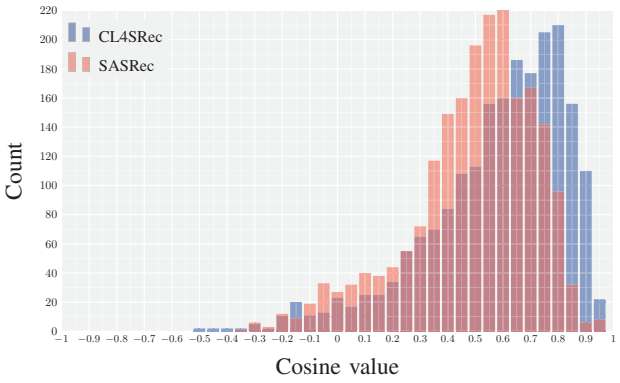


Figure 6: Comparison of cosine similarity distribution between friends in the Yelp dataset. Each bar at the range $[x, x+0.05)$ demonstrates the count of users whose similarity score with her/his friends falls in this range.

Figure 6 shows the evaluation results. The average cosine similarity is 0.5118 and 0.6021 for SASRec and CL4SRec, respectively. We can observe that the representations of similar users inferred by CL4SRec are much closer in the latent space than SASRec. This demonstrates that CL4Rec can really capture user preference hidden in her/his interaction sequences, thus deriving better representations in the user behavior sequence level.

V. Conclusion

In this paper, we propose a novel model called Contrastive Learning for Sequential Recommendation (CL4SRec), which can alleviate the data sparsity problem and learn the effective user representations. It is equipped with the contrastive learning to extract self-supervised signals just from raw data. In addition, we propose three data augmentation approaches to construct contrastive tasks and exploit the effects of the composition of different augmentations. The proposed method is verified on four public datasets. Extensive experiment results show that our CL4SRec achieves significant improvements and outperforms the start-of-the-art baselines.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant (No. 61832001), Beijing Academy of Artificial Intelligence (BAAI), PKU-Baidu Fund 2019BD006.

References

[1] J. Zhao, P. Zhao, L. Zhao, Y. Liu, V. S. Sheng, and X. Zhou, "Variational self-attention network for sequential recommendation," in *37th IEEE International Conference on Data Engineering, ICDE 2021*. IEEE, 2021, pp. 1559–1570.

[2] Z. Deng, C. Li, S. Liu, W. Ali, and J. Shao, "Knowledge-aware group representation learning for group recommendation," in *37th IEEE International Conference on Data Engineering, ICDE 2021*. IEEE, 2021, pp. 1571–1582.

[3] Y. Tan, C. Yang, X. Wei, Y. Ma, and X. Zheng, "Multi-facet recommender networks with spherical optimization," in *37th IEEE International Conference on Data Engineering, ICDE 2021*. IEEE, 2021, pp. 1524–1535.

[4] G. Li, H. Liu, G. Li, S. Shen, and H. Tang, "Lstm-based argument recommendation for non-api methods," *Science China Information Sciences*, vol. 63, no. 9, pp. 1–22, 2020.

[5] Q. Wang, W. Min, D. He, S. Zou, T. Huang, Y. Zhang, and R. Liu, "Discriminative fine-grained network for vehicle re-identification using two-stage re-ranking," *Science China Information Sciences*, vol. 63, no. 11, pp. 1–12, 2020.

[6] X. Miao, H. Zhang, Y. Shi, X. Nie, Z. Yang, Y. Tao, and B. Cui, "Het: Scaling out huge embedding model training via cache-enabled distributed framework," 2022.

[7] X. Miao, L. Ma, Z. Yang, Y. Shao, B. Cui, L. Yu, and J. Jiang, "Cuwide: Towards efficient flow-based training for sparse wide models on gpus," *IEEE Transactions on Knowledge & Data Engineering*, pp. 1–1, 2020.

[8] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *4th International Conference on Learning Representations*.

[9] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. ACM, 2010, p. 811–820.

[10] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, ser. WSDM '18. ACM, 2018, p. 565–573.

[11] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 197–206.

[12] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19. ACM, 2019, p. 1441–1450.

[13] F. Fu, Y. Shao, L. Yu, J. Jiang, H. Xue, Y. Tao, and B. Cui, "Vf^{2}boost: Very fast vertical federated gradient boosting for cross-enterprise learning," in *SIGMOD '21: International Conference on Management of Data*. ACM, 2021, pp. 563–576.

[14] F. Fu, Y. Hu, Y. He, J. Jiang, Y. Shao, C. Zhang, and B. Cui, "Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 3304–3314.

[15] X. Nie, S. Cao, X. Miao, L. Ma, J. Xue, Y. Miao, Z. Yang, Z. Yang, and B. Cui, "Dense-to-sparse gate for mixture-of-experts," *arXiv preprint arXiv:2112.14397*, 2021.

[16] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 346–353.

[17] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, 2014.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017.

[19] R. Qiu, J. Li, Z. Huang, and H. YIn, "Rethinking the item order in session-based recommendation with graph neural networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 2019, p. 579–588.

[20] T. Yao, X. Yi, D. Z. Cheng, F. X. Yu, A. K. Menon, L. Hong, E. H. Chi, S. Tjoa, J. Kang, and E. Ettinger, "Self-supervised learning for deep models in recommendations," *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2007.12865

[21] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen, *S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization*, ser. CIKM '20. ACM, 2020, p. 1893–1902.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. ACL, Jun. 2019, pp. 4171–4186.

[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[24] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *CoRR*, vol. abs/2002.05709, 2020.

[25] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in

European conference on computer vision. Springer, 2016, pp. 69–84.

[26] R. He and J. J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 191–200.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17. ACM, 2017, p. 152–160.

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

[30] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. ACM, 2017, p. 1419–1428.

[31] T. Chen, H. Yin, H. Chen, R. Yan, Q. V. H. Nguyen, and X. Li, "Air: Attentional intention-aware recommender systems," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 304–315.

[32] F. Lv, T. Jin, C. Yu, F. Sun, Q. Lin, K. Yang, and W. Ng, "Sdm: Sequential deep matching model for online large-scale recommender system," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19. ACM, 2019, p. 2635–2643.

[33] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17. ACM, 2017, p. 130–137.

[34] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, p. 108–116.

[35] J. Huang, W. X. Zhao, H. Dou, J.-R. Wen, and E. Y. Chang, "Improving sequential recommendation with knowledge-enhanced memory networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '18. ACM, 2018, p. 505–514.

[36] L. Hu, L. Cao, S. Wang, G. Xu, J. Cao, and Z. Gu, "Diversifying personalized recommendation with user-session context," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. ijcai.org, 2017, pp. 1858–1864.

[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations*. OpenReview.net, 2017.

[38] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, "Graph contextualized self-attention network for session-based recommendation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 3940–3946.

[39] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*. OpenReview.net, 2018.

[40] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1422–1430.

[41] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*. Springer, 2016, pp. 649–666.

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*, 2013, pp. 3111–3119.

[43] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL, 2020, pp. 7871–7880.

[44] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3733–3742.

[45] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9726–9735.

[46] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '21. ACM, 2021, p. 726–735.

[47] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. ACL, Aug. 2016, pp. 10–21.

[48] A. M. Dai and Q. V. Le, "Semi-supervised sequence

learning," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, vol. 28. Curran Associates, Inc., 2015, pp. 3079–3087.

[49] S. G, U. N, and K. G, "Lawbo: A smart lawyer chatbot," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, ser. CoDS-COMAD '18. ACM, 2018, p. 348–351.

[50] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: Challenges, progress and prospects," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 6332–6338.

[51] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, p. 191–198.

[52] M. Artetxe, G. Labaka, E. Agirre, and K. Cho, "Unsupervised neural machine translation," in *6th International Conference on Learning Representations*, 2018.

[53] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato, "Unsupervised machine translation using monolingual corpora only," in *6th International Conference on Learning Representations*. OpenReview.net, 2018.

[54] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. ACM, 2015, p. 43–52.

[55] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[56] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. ACM, 2017, p. 173–182.

[57] W. Krichene and S. Rendle, *On Sampled Metrics for Item Recommendation*. ACM, 2020, p. 1748–1757.

[58] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, p. 452–461.

[59] C. Wang, M. Zhang, W. Ma, Y. Liu, and S. Ma, "Make it a chorus: Knowledge- and time-aware item modeling for sequential recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2020, p. 109–118.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015.

[61] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.