Meta-optimized Contrastive Learning for Sequential Recommendation

Xiuyuan Qin*
Soochow University, China
20215227016@stu.suda.edu.cn

Huanhuan Yuan*
Soochow University, China
hhyuan@stu.suda.edu.cn

Pengpeng Zhao†
Soochow University, China
ppzhao@suda.edu.cn

Junhua Fang
Soochow University, China
jhfang@suda.edu.cn

Fuzhen Zhuang
Institute of Artificial Intelligence &
SKLSDE, School of Computer Science
Beihang University, China
zhuangfuzhen@buaa.edu.cn

Guanfeng Liu
Macquarie University, Australia
guanfeng.liu@mq.edu.au

Yanchi Liu
Rutgers University, United States
yanchi.liu@rutgers.edu

Victor Sheng
Texas Tech University, United States
victor.sheng@ttu.edu

ABSTRACT

Contrastive Learning (CL) performances as a rising approach to address the challenge of sparse and noisy recommendation data. Although having achieved promising results, most existing CL methods only perform either hand-crafted data or model augmentation for generating contrastive pairs to find a proper augmentation operation for different datasets, which makes the model hard to generalize. Additionally, since insufficient input data may lead the encoder to learn collapsed embeddings, these CL methods expect a relatively large number of training data (e.g., large batch size or memory bank) to contrast. However, not all contrastive pairs are always informative and discriminative enough for the training processing. Therefore, a more general CL-based recommendation model called Meta-optimized Contrastive Learning for sequential Recommendation (MCLRec) is proposed in this work. By applying both data augmentation and learnable model augmentation operations, this work innovates the standard CL framework by contrasting data and model augmented views for adaptively capturing the informative features hidden in stochastic data augmentation. Moreover, MCLRec utilizes a meta-learning manner to guide the updating of the model augmenters, which helps to improve the quality of contrastive pairs without enlarging the amount of input data. Finally, a contrastive regularization term is considered to encourage the augmentation model to generate more informative augmented views and avoid too similar contrastive pairs within the meta updating. The experimental results on commonly used datasets validate the effectiveness of MCLRec[1].

CCS CONCEPTS

• **Information systems → Recommender systems.**.

KEYWORDS

Sequential Recommendation, Contrastive Learning, Meta Learning

1 INTRODUCTION

Sequential Recommendation (SR) models are designed to predict a user's next interacted item based on his/her historical interaction sequence [38, 47]. Compared with other types of recommender systems, SR could accurately characterize users' dynamic interest in the long- and short-term, and capture the sequential pattern hidden in the users' behaviors. Although many of them, such as GRU4Rec [16], SASRec [17], and BERT4Rec [35], have achieved immense performance improvements, they are limited by the sparse and noisy data. Recently, Contrastive Learning (CL) based recommender systems, which leverage the information from different views to boost the effectiveness of learned representations, are introduced to cope with these problems [42].

We recap the basic idea of Contrastive Learning (CL), which is to learn an expressiveness embedding by maximizing agreement between the augmented views of the same sequence and pushing away the views of different sequences. Thus, the choice of augmentation operations becomes one of the most crucial problems for every CL recommendation model. Particularly, according to the different augmentation operations, CL-based recommender systems

*These authors contributed equally to this work.
†Corresponding author.

[1] Our code is available at https://github.com/QinHsiu/MCLRec.

can be divided into three categories. The first category [5, 26, 45] generates different views of the same sequential data by manually choosing random 'mask', 'crop', or 'reorder' operations on the data level. And the second one [30] produces contrastive pairs by 'dropout' on the model level. And the third one [25] combines three model augmentation methods (i.e., 'neural mask', 'layer drop', and 'encoder complement') with data augmentation for constructing view pairs. Most of them are designed as auxiliary tasks to help the primary task of improving the recommendation accuracy.

Despite the recent advances, almost all current CL-based models produce contrastive pairs by manually identifying which random augmentation operations are conducted on either the data level or model level. Nevertheless, the need for augmentation operations in practice for different datasets always be diverse and evolving. Even though sequence augmentation methods utilizing random sequence or model perturbations (including 'crop', and 'mask' operations) have been widely used and shown great superiority, only relying on such unlearnable operations often requires domain expertise and hand-crafted design, which may not be enough to search for a suitable augmentation operation in such a setting.

Furthermore, self-supervised contrastive learning does not require labeled data, but insufficient input data may lead the encoder to learn collapsed embeddings [12, 48]. Conventionally, contrastive methods enlarge the batch (or memory bank) and increase the number of augmented views to promote the performance of models for better representations, but many contrastive pairs maybe not be too informative to guide the training, i.e., the representations of positive pairs are pretty close, and negative pairs are already very apart in the latent space [22]. Such pairs may have few contributions to the optimization and lead to contrastive methods further to pursue the large numerous input data to collect informative ones. Moreover, simply enlarging the batch size will highly promote the cost and reduce training efficiency.

In this paper, we propose a general sequential recommendation model with a meta-learning algorithm, which we call Meta-optimized Contrastive Learning for sequential Recommendation (MCLRec). Firstly, auxiliary contrastive learning is chosen to complement the primary task in both the data and model perspectives. A learnable model augmentation method is combined with data augmentation methods in MCLRec for extracting more expressive features. In this way, model augmented views can serve as additional contrastive pairs and be contrasted with data augmented views during training. Additionally, the parameters of the model augmenters could adaptively adjust to different datasets. Secondly, we leverage a meta manner to update the parameters of the augmentation model according to the performance of the encoder. By using such a learning paradigm, the augmentation model could learn discriminative augmented views based on a relatively restricted amount of interactions (e.g., small batch size). Finally, a contrastive regularization term is considered in MCLRec by injecting a margin between the similarities of similar pairs for avoiding feature collapse and generating more informative and discriminative features. In a short, the major contributions of MCLRec are as follows:

- A learnable contrastive learning method MCLRec is proposed for sequential recommendation. MCLRec extracts additional helpful information from the existing positive and negative samples (generated by data augmentation) by combining data augmentation and learnable model augmentation.
- A meta-optimized manner is leveraged in the proposed model MCLRec to guide the training of learnable model augmenters and help learn more discriminative features for recommendation.
- Extensive experiments on different public benchmark datasets demonstrate that MCLRec can significantly outperform the state-of-the-art sequential methods.

2 PRELIMINARIES

2.1 Problem Definition

Sequential Recommendation (SR) is to recommend the next item that the user will interact with based on his/her historical interaction data. Assuming that user sets and item sets are $\mathcal{U}$ and $\mathcal{I}$ respectively, user $u \in \mathcal{U}$ has a sequence of interacted items $S^u = \{i_1^u, ..., i_{|S^u|}^u\}$ and $i_k^u \in \mathcal{I} (1 \leq k \leq |S^u|)$ represents an interacted item at position k of user u within the sequence, where $|S^u|$ denotes the sequence length. Given the historical interactions S^u, the goal of SR is to recommend an item from the set of items $\mathcal{I}$ that the user u may interact with at the $|S^u| + 1$ step:

$$\arg \max_{i \in \mathcal{I}} P(i_{|S^u+1|}^u = i | S^u) \tag{1}$$

2.2 Sequential Recommendation Model

The backbone SR model used in our model contains three parts, (1) embedding layer, (2) representation learning layer, and (3) next item prediction layer.

2.2.1 *Embedding Layer.*
Firstly, the whole item sets $\mathcal{I}$ are embedded into the same space [17, 35] and generate the item embedding matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{I}| \times d}$. Given the input sequence S^u, the embedding of the sequence S^u is initialized to $\mathbf{e}^u \in \mathbb{R}^{n \times d}$ and $\mathbf{e}^u = \{\mathbf{m}_{s_1} + \mathbf{p}_1, \mathbf{m}_{s_2} + \mathbf{p}_2, ..., \mathbf{m}_{s_n} + \mathbf{p}_n\}$, where $\mathbf{m}_{s_k} \in \mathbb{R}^d$ represents the item's embedding at the position k in the sequence, $\mathbf{p}_k \in \mathbb{R}^d$ represents the position embedding in the sequence and n represents the length of the sequence.

2.2.2 *Representation Learning Layer.*
Given the sequence embedding $\mathbf{e}^u$, a deep neural network model (e.g., SASRec [17]) represented as $f_\theta(\cdot)$ is utilized to learn the representation of the sequence. Where θ represents the parameters of the sequential model. The output representation $\mathbf{H}^u \in \mathbb{R}^{n \times d}$ is calculated as:

$$\mathbf{H}^u = f_\theta(\mathbf{e}^u). \tag{2}$$

The last vector $\mathbf{h}_n^u \in \mathbb{R}^d$ in $\mathbf{H}^u = [\mathbf{h}_0^u, \mathbf{h}_1^u, ..., \mathbf{h}_n^u]$ is chosen as the representation of the sequence [30].

2.2.3 *Next Item Prediction Layer.*
Finally, the probability of each item $\hat{\mathbf{y}} = \text{softmax}(\mathbf{h}_n^u \mathbf{M}^\top)$, where $\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{I}|}$. A cross-entropy loss is optimized to maximize the probability of correct prediction:

$$\mathcal{L}_{rec} = -1 * \hat{\mathbf{y}}[g] + \log(\sum_{i \in \mathcal{I}} \exp(\hat{\mathbf{y}}[i]))), \tag{3}$$

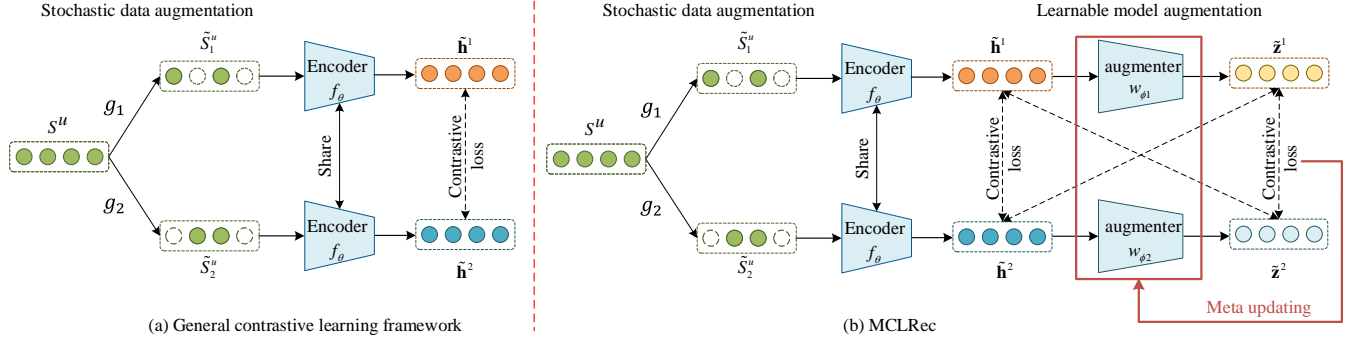where $g \in \mathcal{I}$ represents the ground-truth of user u.

Figure 1: General contrastive learning framework (a) *vs*. MCLRec (b). In general contrastive learning, the applied data augmentation operations are randomly chosen and the generated augmented views are directly contrasted with each other. In MCLRec, we utilize the learnable augmenters to generate two more model augmented views for contrastive learning and leverage the contrastive loss to guide the training of these augmenters in a meta-optimized manner.

3 METHODOLOGY

As shown in Figure 1(a), a general contrastive learning framework commonly consists of a stochastic data augmentation module, a user representation encoder, and a contrastive loss function [45]. Different from this general CL paradigm that only relies on data augmentation operation, MCLRec further leverages two learnable augmenters to find the suitable augmentation operation adaptively. The whole framework of the MCLRec is depicted in Figure 1(b). Especially, MCLRec consists of three main parts, (1) augmentation module, (2) meta-learning training strategy, and (3) contrastive regularization. All these three modules would be elaborated on in the following subsections.

3.1 Augmentation Module

Our augmentation module mainly contains two parts, a stochastic data augmentation module, and a learnable model augmentation module. The former is to generate two different augmented sequences from the same sequence, and the latter is to capture more informative features according to these augmented sequences.

*3.1.1 **Stochastic Data Augmentation Module**.* As shown in Figure 1(b), the first half of our model is the same as the general contrastive learning framework [26, 45]. The stochastic data augmentation operations in MCLRec could be any classical augmentation e.g., 'mask', 'crop', or 'reorder' operations, to create two positive views of a sequence. Given a sequence S^u, and a pre-defined data augmentation function set $\mathcal{G}$, we denote the generation of two positive views as follows:

$$\tilde{S}_1^u = g_1(S^u), \tilde{S}_2^u = g_2(S^u), \ s.t. \ g_1, g_2 \sim \mathcal{G}, \tag{4}$$

where g_1 and g_2 represent the data augmentation functions sampled from $\mathcal{G}$, $\tilde{S}_1^u$ and $\tilde{S}_2^u$ denote the different augmented sequences. Taking $\tilde{S}_1^u$ and $\tilde{S}_2^u$ as inputs, the data augmentation views $\tilde{\mathbf{h}}^1$ and $\tilde{\mathbf{h}}^2$ are generated according to Eq. (2).

*3.1.2 **Learnable Model Augmentation Module**.* General contrastive methods only rely on data augmentations. More recent emerging contrastive methods leverage different dropouts to generate different augmentation models for constructing contrastive

loss [30], which provides a new way to produce augmentation views and inspires us to take advantage of both data and model augmentation. However, no matter the previous data or model augmentation, their inflexible and random augmentations are hard to generalize in practice. That makes the adaptive and learnable augmentation needed for the CL framework. Hence, we propose to use two learnable augmenters to capture the informative features hidden in the stochastic data augmented views.

As shown in Figure 1(b), $\tilde{\mathbf{h}}^1$ and $\tilde{\mathbf{h}}^2$ are fed into the augmentation model $w_{\phi 1}(\cdot)$ and $w_{\phi 2}(\cdot)$, respectively. The model augmentation views $\tilde{\mathbf{z}}^1$ and $\tilde{\mathbf{z}}^2$ are calculated as:

$$\tilde{\mathbf{z}}^1 = w_{\phi 1}(\tilde{\mathbf{h}}^1), \tilde{\mathbf{z}}^2 = w_{\phi 2}(\tilde{\mathbf{h}}^2), \tag{5}$$

where $\phi 1$ and $\phi 2$ represent the parameters of two augmenters, which enable the augmentation operation to be learned end to end and adaptively find optimal augmenters for different datasets. The newly generated $\tilde{\mathbf{z}}^1$ and $\tilde{\mathbf{z}}^2$ could act as contrastive pairs to produce more augmentation views without enlarging batch size. Due to the powerful capability of approximating function, the simple Multi-Layer Perceptron (MLP) [46] is chosen as the augmentation model of MCLRec. We leave other neural network models such as self-attention for future work studies.

3.2 Meta-Learning Training Strategy

After the introduction of learnable model augmenters, there are two modules that with parameters need to be updated, each with its own objective (i.e., multi-task learning for the encoder and contrastive task for the augmenters). Since there is possibly a gap between these two objectives, directly updating their parameters using joint learning may lead to suboptimal solutions [18]. Therefore, we follow [11, 24] to perform a meta-learning strategy to guide the training of two augmenters, which is beneficial for the model to mine discriminative augmentation views from the sequence. The whole training process can be concluded in two stages.

In the first stage, we contrast all four augmented views (i.e., $\tilde{\mathbf{h}}^1$, $\tilde{\mathbf{h}}^2$, $\tilde{\mathbf{z}}^1$ and $\tilde{\mathbf{z}}^2$) in different ways and unite the recommendation loss to update the parameters of encoder $f_\theta(\cdot)$. In the second stage, we utilize the learned encoder $f_{\theta'}(\cdot)$ to re-encode the sequence and

Table 1: Comparison with other contrastive learning models.

Augmentation Type		CL4SRec	CoSeRec	LMA4Rec	ICLRec	DuoRec	SRMA	Ours
Stochastic	Data Level	✓	✓	✓	✓	✗	✓	✓
	Model Level	✗	✗	✗	✗	✓	✓	✗
Learnable	Model Level	✗	✗	✓	✗	✗	✗	✓

use the contrastive loss related to the augmenters to optimized update $w_{\phi 1}(\cdot)$ and $w_{\phi 2}(\cdot)$. Concretely, $f_\theta(\cdot)$, $w_{\phi 1}(\cdot)$ and $w_{\phi 2}(\cdot)$ are iteratively trained until convergence. Especially, in the first stage, we randomly initialize the parameters of encoder $f_\theta(\cdot)$ and two augmenters $w_{\phi 1}(\cdot)$ and $w_{\phi 2}(\cdot)$. After getting all four augmented views, we calculate the recommendation loss by Eq. (3) and joint contrastive losses to update the encoder $f_\theta(\cdot)$ by back-propagation, which can be calculated as:

$$\mathcal{L}_0 = \mathcal{L}_{rec} + \lambda \mathcal{L}_{cl1} + \beta \mathcal{L}_{cl2}, \tag{6}$$

where λ and β are hyper-parameters that need to be tuned. $\mathcal{L}_{cl1}$ and $\mathcal{L}_{cl2}$ denote the two kinds of contrastive losses. The first kind of contrastive loss acts as the same role as infoNCE loss of general contrastive learning [3, 15], called $\mathcal{L}_{cl1}$, which depends only on data augmented view $\tilde{\mathbf{h}}^1$ and $\tilde{\mathbf{h}}^2$. It can be formulated as:

$$\mathcal{L}_{cl1} = \mathcal{L}_{con}(\tilde{\mathbf{h}}^1, \tilde{\mathbf{h}}^2), \tag{7}$$

and

$$\mathcal{L}_{con}(\mathbf{x}^1, \mathbf{x}^2) = -\log \frac{e^{s(\mathbf{x}^1, \mathbf{x}^2)}}{e^{s(\mathbf{x}^1, \mathbf{x}^2)} + \sum_{\mathbf{x} \in neg} e^{s(\mathbf{x}^1, \mathbf{x})}}$$
$$-\log \frac{e^{s(\mathbf{x}^2, \mathbf{x}^1)}}{e^{s(\mathbf{x}^2, \mathbf{x}^1)} + \sum_{\mathbf{x} \in neg} e^{s(\mathbf{x}^2, \mathbf{x})}}, \tag{8}$$

where $(\mathbf{x}^1, \mathbf{x}^2)$ represents a pair of positive sample's embedding, $s(\cdot)$ represents inner product and neg indicates the negative sample embedding set. The positive pairs obtained from the same sequence and other $2(|\mathbf{B}|-1)$ views within the same batch are treated as negative samples, where $|\mathbf{B}|$ denotes the batch size. The second kind of contrastive learning loss called $\mathcal{L}_{cl2}$, is generated from both data and model augmentation views. It can be calculated as:

$$\mathcal{L}_{cl2} = \mathcal{L}_{con}(\tilde{\mathbf{z}}^1, \tilde{\mathbf{z}}^2) + \mathcal{L}_{con}(\tilde{\mathbf{h}}^1, \tilde{\mathbf{z}}^2) + \mathcal{L}_{con}(\tilde{\mathbf{h}}^2, \tilde{\mathbf{z}}^1). \tag{9}$$

In the second stage, we fix the parameters of encoder $f_\theta(\cdot)$ and optimize $w_{\phi 1}(\cdot)$ and $w_{\phi 2}(\cdot)$ with respect to the performance of the encoder. Denote θ' is the learned parameters by back-propagation at the first stage, we use the learned encoder $f_{\theta'}(\cdot)$ to re-encode the augmented sequence by Eq. (2), recompute $\mathcal{L}_{cl2}$ by Eq. (9), and then leverage back-propagation to update the augmenters. The loss is calculated as follows:

$$\mathcal{L}_1 = \mathcal{L}_{cl2}. \tag{10}$$

Then we get learned augmenters $w_{\phi 1'}(\cdot)$ and $w_{\phi 2'}(\cdot)$, where $\phi 1'$ and $\phi 2'$ represent the learned parameters by back-propagation at second stage. With this meta-learning paradigm, the difference between the dimensions of learned views is more significant, and such informative and discriminative features promote the effectiveness of contrastive learning. In addition, the two modules (i.e., encoder and augmenters) are tightly coupled. More details can be seen in 4.4 of the experimental section.

Algorithm 1 The MCLRec Algorithm

Input: Training dataset $\{S_u\}_{u=1}^{|U|}$, learning rate l and l', hyper-parameters λ, β, γ;
Initialize: θ for encoder $f_\theta(\cdot)$, $\phi 1$ for augmenter $w_{\phi 1}(\cdot)$ and $\phi 2$ for augmenter $w_{\phi 2}(\cdot)$;

1: **repeat**
2:     **for** t-th training iteration **do**
3:         $\mathcal{L}_0 = \mathcal{L}_{rec} + \lambda \mathcal{L}_{cl1} + \beta \mathcal{L}_{cl2} + \gamma \cdot \mathcal{R}$
4:         $\theta \leftarrow \theta - l \triangle_\theta \mathcal{L}_0$
5:         Update encoder $f_\theta(\cdot)$ by minimizing $\mathcal{L}_0$
6:     **end for**
7:     **for** t-th training iteration **do**
8:         $\mathcal{L}_1 = \mathcal{L}_{cl2} + \gamma \cdot \mathcal{R}$
9:         $\phi 1 \leftarrow \phi 1 - l' \triangle_{\phi 1} \mathcal{L}_1$
10:        $\phi 2 \leftarrow \phi 2 - l' \triangle_{\phi 2} \mathcal{L}_1$
11:        Update $w_{\phi 1}(\cdot)$ and $w_{\phi 2}(\cdot)$ by minimizing $\mathcal{L}_1$
12:     **end for**
13: **until** $\theta, \phi 1, \phi 2$ converge

3.3 Contrastive Regularization

To prevent creating collapsed augmented views and avoid two augmenters generating too similar contrastive pairs, we further propose a contrastive regularization within updating parameters [22]. Given two augmented views $\tilde{\mathbf{z}}^1$ and $\tilde{\mathbf{z}}^2$, we calculate the similarity scores between them by the inner product and then split the output into positive and negative score sets, σ^+ and σ^-, which is calculated as:

$$\sigma^+, \sigma^- = contrast(\tilde{\mathbf{z}}^1, \tilde{\mathbf{z}}^2), \tag{11}$$

where $contrast$ represents the inner product and split operations. The scores calculated from the same sequence are split into positive score sets σ^+, and others are split into negative score sets σ^-. After that, the following formula is used to calculate the regularization:

$$o_{min} = \min(\{\min(\sigma^+), \max(\sigma^-)\}),$$
$$o_{max} = \max(\{\min(\sigma^+), \max(\sigma^-)\}), \tag{12}$$

and

$$\mathcal{R} = \frac{1}{|\sigma^+|} \sum ([\sigma^+ - o_{min}]_+) + \frac{1}{|\sigma^-|} \sum ([o_{max} - \sigma^-]_+), \tag{13}$$

where $|\sigma^+|$ represents the number of positive samples and $|\sigma^-|$ represents the number of negative samples. $[\cdot]_+$ denotes the cut-off-at-zero function, which is defined as $[a]_+ = \max(a, 0)$. Then the Eq. (6) can be rewritten as follows:

$$\mathcal{L}_0 = \mathcal{L}_{rec} + \lambda \mathcal{L}_{cl1} + \beta \mathcal{L}_{cl2} + \gamma \mathcal{R}. \tag{14}$$

The Eq. (10) can be rewritten as follows:

$$\mathcal{L}_1 = \mathcal{L}_{cl2} + \gamma \mathcal{R}, \tag{15}$$

where γ is a weight to balance the contrastive regularization and other losses. The whole training process is detailed by Algorithm 1.

3.4 Discussion

*3.4.1 **Connections with Contrastive SSL in SR**.* Recent methods [5, 13, 25, 26, 30, 45] mainly take the contrastive objective as an auxiliary task to complement the main recommendation task. Among them, CL4SRec [45], CoSeRec [26], and ICLRec [5] augment

Table 2: Statistical information of experimented datasets.

Datasets	#users	#items	#actions	avg.length	sparsity
Sports	33598	18357	296337	8.3	99.95%
Beauty	22363	12101	198502	8.8	99.93%
Yelp	30431	20033	316354	10.4	99.95%

the input sequence at data level with cropping, masking, and reordering. DuoRec [30] conducts neural masking augmentation on the input sequence at the model level. LMA4Rec [13] introduces Learnable Bernoulli Dropout (LBD [1]) to the encoder and combines it with stochastic data augmentation to construct contrastive views. SRMA [25] introduces three model augmentation methods (neural masking, layer dropping, and encoder complementing) and combines them with data augmentation for constructing view pairs. However, all the above models use either stochastic data augmentation or stochastic model augmentation. Different from the above-mentioned models, our model can be viewed as a two-stage process combining both data and model augmentation operations. In the first stage, stochastic data augmentation is applied to obtain two pairwise contrastive views, and adaptively learn more informative features from these views by using two learnable augmenters in the second stage. Compared with other CL models, MCLRec leverages data and model augmentation views to enlarge the number of contrastive pairs without increasing the input data, thus extracting more informative features for the model training. Meanwhile, the meta-learning optimized approach is also implemented to guide the training of learnable augmenters, which is an alternative way to fuse these two augmentation manners. The main differences are summarized in Table 1.

3.4.2 ***Time Complexity Analysis of MCLRec.*** The complexity of our model mainly comes from the training and the testing. During training, the computation costs of our proposed method are mainly from the optimization of θ, ϕ_1, and ϕ_2 with multi-task learning in two stages. For stage one, since we have four objectives to optimize the network f_θ, the time complexity is $O(|\mathcal{U}|^2 d + |\mathcal{U}| d^2)$.

For stage two, we have two objectives to optimize the augmenters, the time complexity is $O(|\mathcal{U}| d^2)$. The overall complexity is dominated by the term $O(|\mathcal{U}|^2 d)$, where $|\mathcal{U}|$ represents the number of users. In the testing phase, the proposed augmenters and contrastive objectives are no longer needed, which enables the model to have the time complexity as the encoder, e.g., SASRec ($O(d|\mathcal{I}|)$), where $|\mathcal{I}|$ represents the number of items. Based on the above analysis, our MCLRec achieves comparable time complexity when computing with state-of-the-art contrastive SR models [5, 45].

4 EXPERIMENT

In this section, we conduct extensive experiments with three real-world datasets, investigating the following research questions (**RQ**s).

- **RQ1**: How does MCLRec perform compared to state-of-the-art sequential recommendation models?
- **RQ2**: How effective are the key model components (e.g., stochastic data augmentation, learnable model augmentation, contrastive regularization) in MCLRec?

- **RQ3**: How does the meta-learning training strategy affect the recommendation performance?
- **RQ4**: How does the robustness (e.g., training on small batch size, adding noise on test datasets) of MCLRec?

4.1 Experimental Settings

4.1.1 ***Datasets.*** To verify the effectiveness of our methods, we evaluate the model on three real-world benchmark datasets: Amazon (Beauty and Sports)[2] and Yelp[3]. Amazon dataset collects user review data from amazon.com, which is one of the largest e-commerce websites in the world. We use two sub-categories, Amazon-Beauty and Amazon-Sports, in our experiments. Yelp is a dataset for business recommendations. Following [26, 30, 49] for preprocessing, the users and items that have less than five interactions are removed. The statistics of the prepared datasets are summarized in Table 2.

4.1.2 ***Baseline Methods.*** We compare our models with the following three groups of sequential recommendation models:

- **Non-sequential models**: BPR [31] uses Bayesian Personalized Ranking (BPR) loss to optimize the matrix factorization model.
- **General sequential models**: GRU4Rec [16] uses Gated Recurrent Unit (GRU) to model for the sequential recommendation. Caser [36] uses both horizontal and vertical Convolution Neural Networks (CNN) to model sequential behaviors. SASRec [17] for the first time to use the attention mechanism to the sequential recommendation and achieve a good performance.
- **Self-supervised based sequential models**: BERT4Rec [35] uses the deep bidirectional self-attention to capture the potential relationships between items and sequences in Cloze task [6]. S^3-Rec [49] uses self-supervised learning to capture the correlations between items. Since there is no attribute information in our experiments, only the MIP (Masked Item Prediction) task, called S^3-Rec$_{MIP}$, is used for training. CL4SRec [45] uses both data augmentation and contrastive learning in the sequential recommendation for the first time. CoSeRec [26] further proposes two more informative data augment methods (i.e., 'insert' and 'substitute') to improve the performance of contrastive learning. LMA4Rec [13] improves CoSeRec by introducing a Learnable Bernoulli Dropout (LBD [1]) to the encoder, which is to extract more signals from the stochastic augmented views. ICLRec [5] learns users' latent intents from the behavior sequences through clustering and integrates the learned intents into the model via an auxiliary contrastive SSL loss. DuoRec [30] proposes a sampling strategy to formulate positive samples and uses dropout [34] to conduct the model-level augmentation. SRMA [25] introduces three model augmentation methods (i.e., 'neural mask', 'layer drop', and 'encoder complement') and combines them with data augmentation for constructing view pairs.

4.1.3 ***Evaluation Metrics.*** For evaluation purposes, we split the data into training, validation, and testing datasets based on timestamps given in the datasets [5, 17, 30]. Specifically, the last item is used for testing, the second-to-last item is used for validation, and the rest for training. Following [21, 39], we rank the whole item

---

[2]https://jmcauley.ucsd.edu/data/amazon/
[3]https://www.yelp.com/dataset

Table 3: Performance comparisons of different methods. Where the bold score is the best in each row and the second-best baseline is underlined. The last column is the relative improvements compared with the best baseline results.

Dataset	Metric	BPR	GRU4Rec	Caser	SASRec	BERT4Rec	S³-Rec$_{MIP}$	CL4SRec	CoSeRec	LMA4Rec	ICLRec	DuoRec	SRMA	MCLRec	Improv.
Sports	HR@5	0.0123	0.0162	0.0154	0.0214	0.0217	0.0121	0.0231	0.0290	0.0297	0.0290	0.0312	0.0299	**0.0328**	5.13%
	HR@10	0.0215	0.0258	0.0261	0.0333	0.0359	0.0205	0.0369	0.0439	0.0439	0.0437	0.0466	0.0447	**0.0501**	7.51%
	HR@20	0.0369	0.0421	0.0399	0.0500	0.0604	0.0344	0.0557	0.0636	0.0634	0.0646	0.0696	0.0649	**0.0734**	5.46%
	NDCG@5	0.0076	0.0103	0.0114	0.0144	0.0143	0.0084	0.0146	0.0196	0.0197	0.0191	0.0192	0.0199	**0.0204**	2.51%
	NDCG@10	0.0105	0.0142	0.0135	0.0177	0.0190	0.0111	0.0191	0.0244	0.0245	0.0238	0.0244	0.0246	**0.0260**	5.69%
	NDCG@20	0.0144	0.0186	0.0178	0.0224	0.0251	0.0146	0.0238	0.0293	0.0293	0.0291	0.0302	0.0297	**0.0319**	5.63%
Beauty	HR@5	0.0178	0.0180	0.0251	0.0377	0.0360	0.0189	0.0401	0.0504	0.0511	0.0500	0.0559	0.0503	**0.0581**	3.94%
	HR@10	0.0296	0.0284	0.0342	0.0624	0.0601	0.0307	0.0642	0.0725	0.0735	0.0744	0.0825	0.0724	**0.0871**	5.58%
	HR@20	0.0474	0.0427	0.0643	0.0894	0.0984	0.0487	0.0974	0.1034	0.1047	0.1058	0.1193	0.1025	**0.1243**	4.19%
	NDCG@5	0.0109	0.0116	0.0145	0.0241	0.0216	0.0115	0.0268	0.0339	0.0342	0.0326	0.0340	0.0318	**0.0352**	2.92%
	NDCG@10	0.0147	0.0150	0.0226	0.0342	0.0300	0.0153	0.0345	0.0410	0.0414	0.0403	0.0425	0.0398	**0.0446**	4.94%
	NDCG@20	0.0192	0.0186	0.0298	0.0386	0.0391	0.0198	0.0428	0.0487	0.0493	0.0483	0.0518	0.0474	**0.0539**	4.05%
Yelp	HR@5	0.0127	0.0152	0.0142	0.0160	0.0196	0.0101	0.0227	0.0241	0.0233	0.0239	0.0429	0.0243	**0.0454**	5.83%
	HR@10	0.0216	0.0248	0.0254	0.0260	0.0339	0.0176	0.0384	0.0395	0.0387	0.0409	0.0614	0.0395	**0.0647**	5.37%
	HR@20	0.0346	0.0371	0.0406	0.0443	0.0564	0.0314	0.0623	0.0649	0.0636	0.0659	0.0868	0.0646	**0.0941**	8.41%
	NDCG@5	0.0082	0.0091	0.0080	0.0101	0.0121	0.0068	0.0143	0.0151	0.0147	0.0152	0.0324	0.0154	**0.0332**	2.47%
	NDCG@10	0.0111	0.0124	0.0113	0.0133	0.0167	0.0092	0.0194	0.0205	0.0196	0.0207	0.0383	0.0207	**0.0394**	2.87%
	NDCG@20	0.0143	0.0145	0.0156	0.0179	0.0223	0.0127	0.0254	0.0263	0.0258	0.0270	0.0447	0.0266	**0.0467**	4.47%

set without negative sampling. In order to evaluate the model effectively, we use two widely-used evaluation metrics, including Hit Ratio @k (HR@k) and Normalized Discounted Cumulative Gain @k (NDCG@k), where $k \in \{5, 10, 20\}$. Intuitively, the HR metric considers whether the ground-truth is ranked amongst the top k items while the NDCG metric is a position-aware ranking metric.

4.1.4 ***Implementation Details***. The implementations of Caser, S³-Rec, BERT4Rec, CoSeRec, LMA4Rec, ICLRec, DuoRec, and SRMA are provided by the authors. BPR, GRU4Rec, SASRec, and CL4SRec are implemented based on public resources. All parameters in these methods are used as reported in their papers and the optimal settings are chosen based on the model performance on validation data. For MCLRec, we use transformer [17] as the encoder, and the number of the self-attention blocks and attention heads is set as 2. The augmenters are 3-layer fully connected MLPs. We set d as 64, n as 50, the learning rate l as 0.001, l' as 0.001, and the batch size as 256. λ, β are selected from $\{0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and γ is $0.1 \times \beta$. The whole model is optimized with the Adam [20] optimizer. We train the model with an early stopping strategy based on the performance of validation data. All experiments are implied on NVIDIA GeForce RTX 2080 Ti GPU.

4.2 Overall Performances (RQ1)

We compare the performance of all baselines with MCLRec for Sequential Recommendation. Table 3 shows the experimental results of the compared models on three datasets, and the following findings can be seen through it:

- The self-supervised based models perform more effectively than classical models, such as BPR, GRU4Rec, Caser, and SASRec. Among them, different from BERT4Rec and S³-Rec$_{MIP}$ that use MIP tasks to train the model, CL4SRec, CoSeRec, LMA4Rec, ICLRec, DuoRec, and SRMA utilize data augmentation and contrastive learning for training, which lead to generally better results than BERT4Rec and S³-Rec$_{MIP}$. That indicates contrastive learning paradigm may generate more expressive embeddings for users and items by maximizing the mutual information.

Table 4: Ablation study with key modules. Where HR and NDCG indicate HR@20 and NDCG@20.

Model	Dataset					
	Sports		Beauty		Yelp	
	HR	NDCG	HR	NDCG	HR	NDCG
(A) MCLRec	**0.0734**	**0.0319**	**0.1243**	**0.0539**	**0.0941**	**0.0467**
(B) w/o $\mathcal{L}_{cl1}$	0.0705	0.0299	**0.1243**	**0.0539**	0.0918	0.0462
(C) w/o $\mathcal{L}_{cl2}$	0.0557	0.0238	0.1056	0.0394	0.0623	0.0254
(D) w/o $\mathcal{R}$	0.0691	0.0291	0.1236	0.0529	0.0873	0.0445
(E) share	0.0707	0.0299	0.1231	0.0532	0.0923	0.0456

- Compared to SRMA and CL4SRec, we can find that introducing model augmentation can further improve performance. In addition, DuoRec performs better than other baselines on all datasets. Compared with the previous SSL-based sequential models, DuoRec utilizes both supervised data augmentation and random model augmentation, and thus improves the performance by a large margin. That motivates us to combine two types of augmentation operations within the training of CL.

- Benefiting from the meta-optimized model augmentation operation, MCLRec significantly outperforms other methods on all metrics across the different datasets. For instance, MCLRec improves over the second-best result w.r.t. HR and NDCG by 3.94-8.41% and 2.47-5.69% on three datasets, respectively. The reasons are concluded as: (1) Our learnable augmentation module adaptively learns appropriate augmentation representations for contrastive learning. (2) The meta-learning manner acts as an effective way for training the augmentation model as well as boosting recommendation accuracy. The results support that our contrastive recommendation framework can enable different models to learn more informative representations.

4.3 Ablation Study (RQ2)

To analyze the effectiveness of each component of our model, we conduct several ablation experiments about MCLRec. HR@20 and NDCG@20 performances of different variants are shown in Table 4,
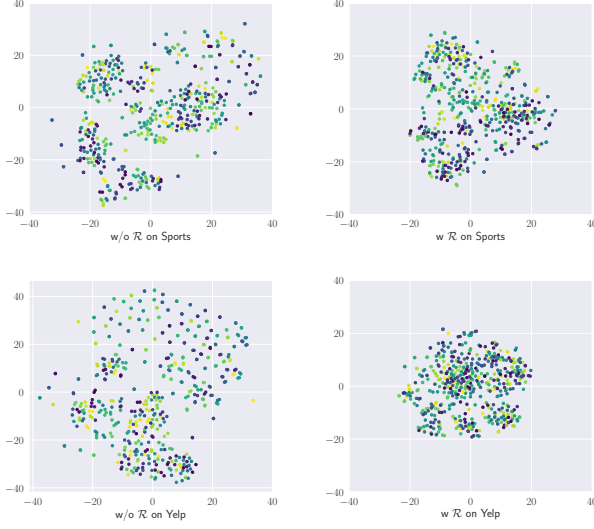
Figure 2: T-SNE visualization of the model augmentation views $\tilde{z}^1$ and $\tilde{z}^2$ trained with w/o $\mathcal{R}$ and w $\mathcal{R}$ on Sports and Yelp. Where different colors represent negative pairs.

Figure 3: Comparison of two versions MCLRec (have different train strategies) with CL4SRec and DuoRec on all datasets.

Figure 4: Performances comparison w.r.t. Batch Size.

where w/o denotes without, (A) represents MCLRec, (B) removes the $\mathcal{L}_{cl1}$ by setting λ to 0 in Eq. (14), (C) removes the $\mathcal{L}_{cl2}$ (, which is equivalent to CL4SRec), (D) removes the contrastive regularization component, and (E) denotes the two augmenters that share parameters (i.e., $\phi 1 = \phi 2$). From this table, we can find that MCLRec achieves the best results on all datasets, which indicates all components are effective for our framework and the meta-optimized contrastive learning enhances the model's ability to learn more expressive representations. By comparing (A) with (C) and (D), we find that learnable model augmentation and contrastive regularization could significantly improve the model accuracy, which is consistent with our statements. By comparing (B) and (C), it can be observed that learnable augmentation is much more efficient than random data augmentation. By comparing (A) and (B), the combination of data and model level augmentation could further boost model performance. By comparing (A) and (E), we can find that sharing parameters of augmenters will decrease the results. This may be the fact that using the same augmenter may further lead to a high similarity of learned augmentation views, thus making the performance degraded. As shown in Table 4, after removing the regular term, the performance of our model decreases on all three datasets, which indicates the effectiveness of the regular term. To further analyze the effect of the regular term on the model, we visualize the learned augmentation views $\tilde{z}^1$ and $\tilde{z}^2$ via T-SNE [37]. To simplify, we denote without as w/o and with as w. We use both w/o $\mathcal{R}$ and w $\mathcal{R}$ to train our model for 300 epochs in an end-to-end manner respectively and utilize T-SNE to reduce the augmented embeddings into two-dimensional space. Limited by the space, the results of Sports and Yelp are presented in Figure 2. We find that w/o $\mathcal{R}$ allows the enhancer to learn collapsed view representations (i.e., the representations of both positive and negative pairs are too
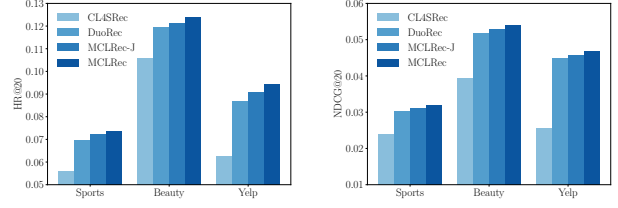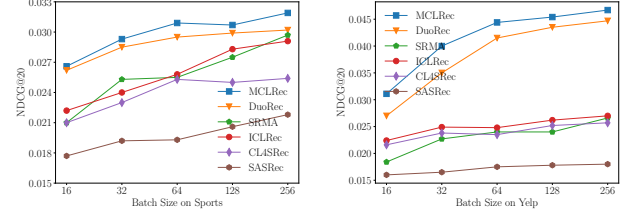
"dispersed"), and w $\mathcal{R}$ allows the augmenters to learn more discriminative features (i.e., the positive pairs are "close" enough and the negative pairs are relative "far away"). This further demonstrates the effectiveness of the regular term. In addition, we found that simply adding $\mathcal{R}$ to other models (e.g., CL4SRec and DuoRec) will make the performance worse. The reason might be that $\mathcal{R}$ is mainly designed to constrain model augmenters.

4.4 Effectiveness of Meta Optimization (RQ3)

We conduct several experiments based on MCLRec to analyze the effectiveness of meta optimization. We first compare the performance of the joint-learning strategy with MCLRec, called MCLRec-J, where the joint-learning strategy means the whole model is trained according to $\mathcal{L}_0$ (Eq. (14)) in one step. As shown in Figure 3, our meta-optimized based manner outperforms others on all datasets. Specifically, *on the one hand*, MCLRec-J performs better than CL4SRec and DuoRec, which demonstrates the effectiveness of learnable model augmentation. *On the other hand*, MCLRec beats the MCLRec-J. The main reason is that meta-learning help CL to learn more discriminative augmentation views. To specify this point, we visualize the learned augmentation views $\tilde{h}^1$ and $\tilde{h}^2$ via T-SNE [37]. We use the joint-learning strategy and meta-learning strategy to train our model for 300 epochs in an end-to-end manner respectively and utilize T-SNE to reduce the augmented embeddings into two-dimensional space. Limited by the space, the results of Sports and Yelp are presented in Figure 5. We intuitively observe that the representations of the negative pairs generated by MCLRec are more "scattered" and the representations of positive pairs generated by MCLRec are "close" than that of MCLRec-J, which indicates that meta-learning strategy helps avoid collapsed results and outputs more informative representations for recommendation. The main reason may be that there are two modules (i.e., encoder and augmenters) with parameters that need to be updated, and the target objects of the two modules are different, which leads to a possible
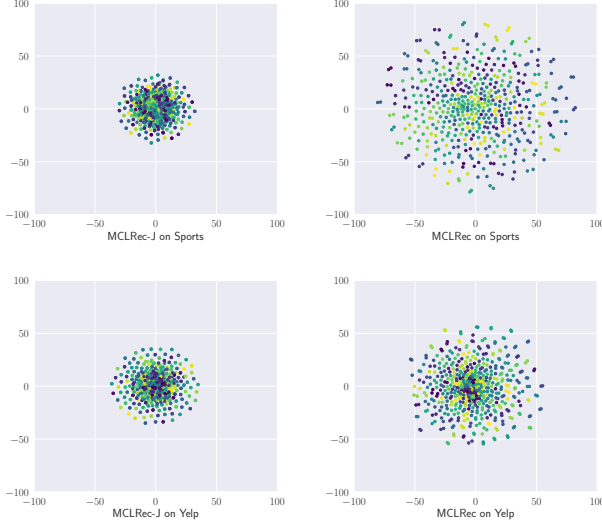
Figure 5: Comparison of the data augmentation views, $\tilde{\mathbf{h}}^1$ and $\tilde{\mathbf{h}}^2$, trained with different strategies on Sports and Yelp datasets. The dimensions are reduced via T-SNE, where different colors represent negative pairs.

Figure 6: Performances of MCLRec w.r.t. different weights assigned to the $\mathcal{L}_{cl1}$ and the $\mathcal{L}_{cl2}$ on all datasets.

Figure 7: Performance comparison w.r.t. different Noise Ratio on Sports and Yelp datasets.

gap between the two objects, thus directly using joint learning to update their parameters may lead to suboptimal results by lowering the performance of both modules [18].

4.5 Further Analysis (RQ4)

In this section, we conduct experiments on the Sports and Yelp datasets to verify the robustness of MCLRec. For all models in the following experiments, we only change one variable at a time while keeping other hyper-parameters optimal.

*4.5.1 **Impact of Batch Size**.* From Figure 4, we can see that reducing the batch size deteriorates the performance of all models. Comparing SASRec and other models, it can be shown that adding a self-supervised auxiliary task can significantly improve the model's performance with different batch sizes. Most importantly, MCLRec's performance with 64 batch size can outperform all other models with 256 batch size on Sports and Yelp. It indicts that, comparing MCLRec and Cl4SRec, our proposed method can preforms well without of large batch size. The reason can be concluded that the introduction of learnable model augmentation allows contrastive learning can be trained with more informative augmentation views including $\tilde{\mathbf{h}}^1$ and $\tilde{\mathbf{h}}^2$, $\tilde{\mathbf{z}}^1$ and $\tilde{\mathbf{z}}^2$.

*4.5.2 **Hyper-parameters Analysis**.* The final loss function of MCLRec in Eq. (14) is a multi-task learning loss. Figure 6 shows the impact of assigning different weights to β and λ on the model. We observe that the performance of MCLRec gets peak value to different β and λ, which demonstrates the effectiveness of the proposed framework and manifests that introducing suitable weights can boost the performance of recommendation. From these figures, $\beta = 0.4$ and $\lambda = 0.04$ for Sports, $\beta = 0.05$ and $\lambda = 0$ for Beauty, and $\beta = 0.1$ and $\lambda = 0.03$ for Yelp are generally proper to MCLRec. The
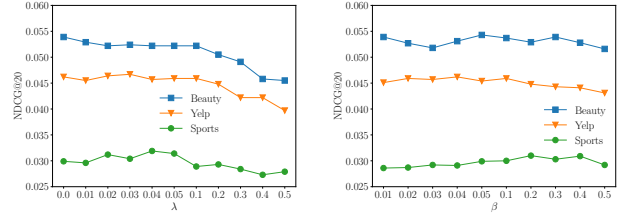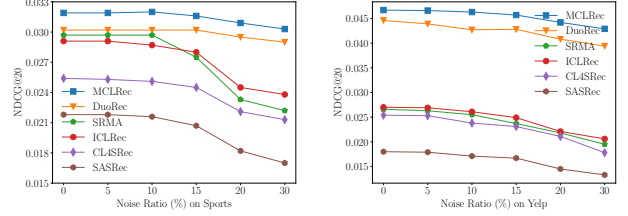
weight of $\mathcal{L}_{cl2}$, i.e., β, is commonly larger than λ, which demonstrates that learnable model augmentation generally gains more importance than stochastic data augmentation.

*4.5.3 **Robustness to Noise Data**.* To verify the robustness of MCLRec against noise interactions, we randomly add a certain proportion (i.e., 5%, 10%, 15%, 20%, 30%) of negative items into the input sequences during testing, and examine the final performance of MCLRec and other baselines. From Figure 7, we can see that adding noisy data deteriorates the performance of all models. By comparing SASRec and other models, it can be seen that adding a contrastive self-supervised auxiliary task can significantly improve the model's robustness to noise data. By comparing CL4SRec with other models, we can see that introducing model augmentation (e.g., DuoRec, SRMA, and MCLRec) or other auxiliary tasks (e.g., ICLRec) can further alleviate the noise data issues. By comparing MCLRec and other models, it can be seen that our model consistently performs better than other models. Especially, with 15% noise proportion, our model can even outperform other models without noise data on two datasets. It indicates that, comparing MCLRec with CL4SRec and SRMA, our proposed method can perform well against the noise data. The reason can be concluded that with the help of meta training strategy and regular term, our augmenters can adaptively learn appropriate representations from the stochastic augmented views for contrastive learning.

*4.5.4 **Robustness w.r.t. User Interaction Frequency**.* To further analyze the robustness of MCLRec against sparse data (e.g., limited historical behaviors), we divide the user behavior sequences into three groups based on their length and keep the total number of behavior sequences constant. The statistics of the prepared datasets are summarized in Table 5. And all models are trained and evaluated independently on each group of users. From Figure 8, we observe that reducing the interaction frequency deteriorates the

Table 5: Statistical information of experimented datasets.

DataSets	Sports			Yelp		
#length	=5	6-8	>8	=5	6-8	>8
#users	11416	14209	9973	8076	11109	11246
#items	18357	18357	18357	20032	20030	20033
#actions	57080	95564	143693	40380	75082	200892
sparsity	99.97%	99.96%	99.92%	99.98%	99.97%	99.91%

Figure 8: Performance comparison on different user groups among SASRec, CL4SRec, DuoRec and MCLRec.

performance of all models. By comparing MCLRec with SASRec and CL4SRec, we find that MCLRec can consistently perform better than SASRec and CL4SRec among all user groups. This demonstrates that MCLRec can further alleviate the data sparsity problem by introducing more informative augmentation features for contrastive learning, thus consistently benefiting the embedding representation learning even when the historical interactions are limited. By Comparing MCLRec with the best baseline model DuoRec, it can be seen that the improvement of MCLRec is mainly because it provides better recommendations to users with low interaction frequency. This shows that combining data augmentation and learnable model augmentation is beneficial, especially when the recommender system faces the problem of sparse data, where the information of each individual user sequence is limited.

5 RELATED WORK

5.1 Sequential Recommendation

Sequential recommendation system [38, 47] aims to predict successive preferences according to one's historical interactions, which has been heavily researched in academia and industry. Classical Markov Chains [32], Recurrent Neural Networks (RNN)-based [16], Convolutional Neural Networks (CNN)-based [36], Transformer-based [10, 17, 23, 35] and Graph Neural Networks (GNN)-based [2, 43] SR models concentrate on users' ordered historical interactions. However, these sequential models are commonly limited by the sparse and noisy problems in practical life.

5.2 Self-Supervised Learning for Recommendation

Motivated by the immense success of Self-Supervised Learning (SSL) in Natural Language Process (NLP) [7] and Computer Vision (CV) [8, 14], and its effectiveness in solving data sparsity problems, a growing number of works are now applying SSL to recommendation. Among them, some Bidirectional Encoder Representations from Transformer (BERT) like methods to introduce the self-supervised pre-training manner into recommendation [4, 35]. S^3-Rec [49] introduces four auxiliary self-supervised tasks to capture the sequential information. Meanwhile, the resurgence of Contrastive Learning (CL) significantly promotes the progress of SSL's research. CL4SRec [45] and CoSeRec [26] learns the representations of users by maximizing the agreement between differently augmented views. MMInfoRec [29] applies an item level contrastive learning for feature-based sequential recommendation. ICLRec [5] learns users' intent distributions from users' behavior sequences. More prior works also explore the application of CL to graph-based recommendation. SGL [42] generates two augmented views with graph
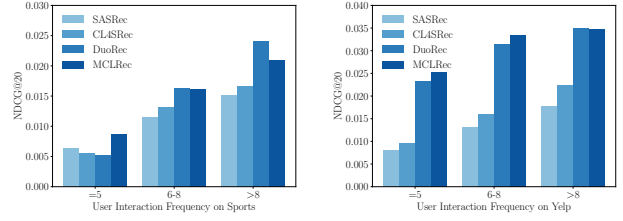
augmentation. GCA [50] explores adaptive topology-level and node-attribute-level augmentation operations. DHCN [44] employs contrastive tasks for hypergraph representation learning. Different from constructing views only by adopting data augmentation, DuoR-ec [30] chooses to construct view pairs with model augmentation. LMA4Rec [13] introduces Learnable Bernoulli Dropout (LBD [1]) to the encoder. SRMA [25] proposes three levels of model augmentation methods. However, these augmentation operations are all hand-crafted and cannot be learned end to end.

5.3 Meta-Learning for Recommendation

Meta-learning, which is known as learning to learn [11], has aroused comprehensive interest in recommender systems. Most meta-learning-based recommendation models are utilized to initialize the parameters for dealing with the cold-start problems in recommendation systems [9, 27, 33, 40]. Recently, some researchers [19, 28, 41] have also explored using meta-learning to find optimal hyper-parameters for recommendation. For example, MeLON [19] adaptively achieves a better learning rate for new coming user-item interactions. Related to meta-learning, our model is designed to update the parameters of learnable augmenters.

6 CONCLUSION

In this paper, we developed a novel contrastive learning-based model called meta-optimized contrastive learning (MCLRec) for sequential recommendation. We took the advantage of data and learnable model augmentation in contrastive learning to create more informative and discriminative features for recommendations. By applying meta-learning, the augmentation model could update its parameters in terms of the encoder's performance. Extensive experimental results showed that the proposed method outperforms the state-of-the-art contrastive learning based sequential recommendation models. In addition, due to the generalization of our framework, in the future, MCLRec could be applied to many other recommendation models and further improve their performance.

7 ACKNOWLEDGMENTS

REFERENCES

[1] Shahin Boluki, Randy Ardywibowo, Siamak Zamani Dadaneh, Mingyuan Zhou, and Xiaoning Qian. 2020. Learnable Bernoulli Dropout for Bayesian Deep Learning. In *AISTATS*. 3905–3916.

[2] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential Recommendation with Graph Neural Networks. In *SIGIR*. 378–387.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*. 1597–1607.

[4] Xusong Chen, Dong Liu, Chenyi Lei, Rui Li, Zheng-Jun Zha, and Zhiwei Xiong. 2019. BERT4SessRec: Content-based video relevance prediction with bidirectional encoder representations from transformer. In *MM*. 2597–2601.

[5] Yongjun Chen, Zhiwei Liu, Jia Li, Julian J. McAuley, and Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation. In *WWW*. 2172–2182.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*. 4171–4186.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[9] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential scenario-specific meta learner for online recommendation. In *KDD*. 2895–2904.

[10] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S. Yu. 2022. Sequential Recommendation via Stochastic Self-Attention. In *WWW*. 2036–2047.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. 1126–1135.

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*. 21271–21284.

[13] Yongjing Hao, Pengpeng Zhao, Xuefeng Xian, Guanfeng Liu, Deqing Wang, Lei Zhao, Yanchi Liu, and Victor S. Sheng. 2022. Learnable Model Augmentation Self-Supervised Learning for Sequential Recommendation. *arXiv preprint arXiv:2204.10128* (2022).

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *CVPR*. 16000–16009.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation Learning. In *CVPR*. 9726–9735.

[16] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*. 843–852.

[17] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. 197–206.

[18] Zhuoliang Kang, Kristen Grauman, and Fei Sha. 2011. Learning with Whom to Share in Multi-task Feature Learning. In *ICML*. 521–528.

[19] Minseok Kim, Hwanjun Song, Yooju Shin, Dongmin Park, Kijung Shin, and Jae-Gil Lee. 2022. Meta-learning for online update of recommender systems. In *AAAI*. 4065–4074.

[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[21] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *KDD*. 1748–1757.

[22] Jiangmeng Li, Wenwen Qiang, Changwen Zheng, Bing Su, and Hui Xiong. 2022. MetaAug: Contrastive learning via meta feature augmentation. In *ICML*. 12964–12978.

[23] Yang Li, Tong Chen, Peng-Fei Zhang, and Hongzhi Yin. 2021. Lightweight Self-Attentive Sequential Recommendation. In *CIKM*. 967–977.

[24] Shikun Liu, Andrew Davison, and Edward Johns. 2019. Self-supervised generalisation with meta auxiliary learning. In *NeurIPS*. 1677–1687.

[25] Zhiwei Liu, Yongjun Chen, Jia Li, Man Luo, and Caiming Xiong. 2022. Improving Contrastive Learning with Model Augmentation. *arXiv preprint arXiv:2203.15508* (2022).

[26] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S Yu, Julian McAuley, and Caiming Xiong. 2021. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479* (2021).

[27] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on heterogeneous information networks for cold-start recommendation. In *KDD*. 1563–1573.

[28] Mi Luo, Fei Chen, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Jiashi Feng, and Zhenguo Li. 2020. MetaSelector: Meta-Learning for Recommendation with User-Level Adaptive Model Selection. In *WWW*. 2507–2513.

[29] Ruihong Qiu, Zi Huang, and Hongzhi Yin. 2021. Memory Augmented Multi-Instance Contrastive Predictive Coding for Sequential Recommendation. In *ICDM*. 519–528.

[30] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive learning for representation degeneration problem in sequential recommendation. In *WSDM*. 813–823.

[31] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*. 811–820.

[33] Jiayu Song, Jiajie Xu, Rui Zhou, Lu Chen, Jianxin Li, and Chengfei Liu. 2021. CBML: A cluster-based meta-learning model for session-based recommendation. In *CIKM*. 1713–1722.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15, 1 (2014), 1929–1958.

[35] 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 *CIKM*. 1441–1450.

[36] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[37] Laurens Van Der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. *JMLR* 15, 1 (2014), 3221–3245.

[38] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. 2019. Sequential Recommender Systems: Challenges, Progress and Prospects. In *IJCAI*. 6332–6338.

[39] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[40] Tianxin Wei, Ziwei Wu, Ruirui Li, Ziniu Hu, Fuli Feng, Xiangnan He, Yizhou Sun, and Wei Wang. 2020. Fast adaptation for cold-start collaborative filtering with meta-learning. In *ICDM*. 661–670.

[41] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. 2022. Contrastive Meta Learning with Behavior Multiplicity for Recommendation. In *WSDM*. 1120–1128.

[42] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *SIGIR*. 726–735.

[43] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *AAAI*. 346–353.

[44] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. In *AAAI*. 4503–4511.

[45] 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).

[46] Eyal Yair and Allen Gersho. 1988. The Boltzmann perceptron network: A multi-layered feed-forward network equivalent to the Boltzmann machine. In *NeurIPS*. 116–123.

[47] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2022. Self-Supervised Learning for Recommender Systems: A Survey. *arXiv preprint arXiv:2203.15876* (2022).

[48] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*. 12310–12320.

[49] 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 *CIKM*. 1893–1902.

[50] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*. 2069–2080.