CT4Rec: Simple yet Effective Consistency Training for Sequential Recommendation

Chong Liu*
Tencent Inc.
China
nickcliu@tencent.com

Xiaoyang Liu*
OPPO Inc.
China
liuxiaoyang@oppo.com

Rongqin Zheng
Tencent Inc.
China
leonezheng@tencent.com

Lixin Zhang
Tencent Inc.
China
lixinzhang@tencent.com

Xiaobo Liang
Soochow University
China
xbliang3@stu.suda.cn

Juntao Li[†]
Soochow University
China
ljt@suda.edu.cn

Lijun Wu
Microsoft Research Asia
China
lijunwu@microsoft.com

Min Zhang
Soochow University
China
minzhang@suda.edu.cn

Leyu Lin
Tencent Inc.
China
goshawklin@tencent.com

ABSTRACT

Sequential recommendation methods are increasingly important in cutting-edge recommender systems. Through leveraging historical records, the systems can capture user interests and perform recommendations accordingly. State-of-the-art sequential recommendation models proposed very recently combine contrastive learning techniques for obtaining high-quality user representations. Though effective and performing well, the models based on contrastive learning require careful selection of data augmentation methods and pretext tasks, efficient negative sampling strategies, and massive hyper-parameters validation. In this paper, we propose an ultra-simple alternative for obtaining better user representations and improving sequential recommendation performance. Specifically, we present a simple yet effective **C**onsistency **T**raining method for sequential **Rec**ommendation (CT4Rec) in which only two extra training objectives are utilized without any structural modifications and data augmentation. Experiments on three benchmark datasets and one large newly crawled industrial corpus demonstrate that our proposed method outperforms SOTA models by a large margin and with much less training time than these based on contrastive learning. Online evaluation on real-world content recommendation system also achieves 2.717% improvement on the click-through rate and 3.679% increase on the average click number per capita. Further exploration reveals that such a simple method

has great potential for CTR prediction. Our code is available at https://github.com/ct4rec/CT4Rec.git.

CCS CONCEPTS

• **Information systems → Information retrieval**.

KEYWORDS

Recommender Systems, Sequential Recommendation, Consistency Training

1 INTRODUCTION

Recommendation systems have been extensively applied in online platforms nowadays, e.g., Amazon [30], Google [1, 5, 22] and Facebook [19]. Due to the dynamic interactions between users and items, it is essential to capture evolving user interests from users' historical records. To accurately represent user interests and make an appropriate recommendation, many efforts have been paid to study sequential recommendation methods [15, 24, 45, 50].

Generally, the sequential recommendation task aims to characterize user representation from users' historical behaviors and predict the expected item accordingly. In viewing the great success of deep learning for sequential dependency modeling and representation learning, many methods based on deep neural networks [14] have been introduced and proposed to solve this task, covering RNN-Based frameworks [15, 32, 44], different CNNs blocks and structures [45, 46], Graph Neural Networks (GNNs) [37, 50, 51], and also model variants [24, 27, 43] relied on the powerful multi-head self-attention [47]. Though performing well, these methods might suffer from the data sparsity problem [24, 42, 63] for sequential recommendation, especially for models built on the multi-head

---

*Both authors contributed equally to this paper.
[†]Juntao Li is the corresponding author.

self-attention mechanism, where only one single item prediction loss is used to optimize the full model parameters for capturing all possible correlations in input interaction sequences.

To address the above challenge, various self-supervised learning strategies are introduced [55, 62]. Among these, the recently introduced contrastive learning (CL) objective [53] achieves very promising results. Through combing with effective data augmentation strategies and cooperating with the vanilla sequential prediction objective, the CL-based method can learn better sequence-level user representations and enhance the performance of sequential recommendations. However, the effectiveness of CL-based approaches is subject to the correlated data augmentation methods, pretext tasks, efficient negative sampling, and hyper-parameters selection (e.g., the temperature in NCE and InfoNCE losses) [20]. To mitigate the above drawbacks, many efforts have been made to simplify contrastive learning. [8] propose a very simple yet effective contrastive learning scheme that utilizes dropout noise as data augmentation to construct high-quality positive samples for sequence-level representation learning. Although such a scheme is effective for unsupervised representation learning, adapting it into the sequential recommendation task will still encounter the dilemma in which a higher proportion of CL objective in model training will lead to better representations that can easily distinguish positive and negative samples but might result in worse item prediction performance and vice versa. In other words, there is a noticeable gap between the discrimination of positive and negative samples and the task objective for the sequential recommendation. Thus, the key to obtaining better user representations mainly for the item prediction task is designing more training strategies that can address the inherent issues of sequential recommendation models.

Inspired by the recent observation on the multi-head attention model that a very simple regularization strategy imposed on the output space of supervised tasks yields striking performance improvement [29] (achieving SOTA on many challenging tasks), we propose to thoroughly explore the effect of consistency training for the sequential recommendation task. We first introduce the simple bidirectional KL divergence regularization into the output space to constrain the inconsistency between two forward passes with different dropouts. Unlike previous machine translation, summarization, and natural language understanding tasks, we argue that the introduced consistency regularization merely in the output space is not enough for the data sparsity setting of sequential recommendation. We then design a novel and simple regularization objective in the representation space. Unlike previous studies that utilize cosine [8] and L2 [35, 64] distance to regularize the representation space, we propose to regularize the distributed probability of each user representation over others. Thus, we can extend and leverage the effective bidirectional KL loss to regularize the representation inconsistency. Experiments on three public benchmarks and one newly collected large-scale corpus indicate that our proposed simple consistency training for sequential recommendation (CT4Rec) outperforms state-of-the-art methods based on contrastive learning by a large margin. Extensive experiments further prove that our proposed consistency training can be easily extended to the data side. The Online A/B test also confirms the effectiveness of our method. Besides, we extend our consistency training method to the

CTR prediction task, and experiments conducted on two newly constructed industrial datasets further prove the effectiveness of our method. In a nutshell, we mainly have the following contributions:
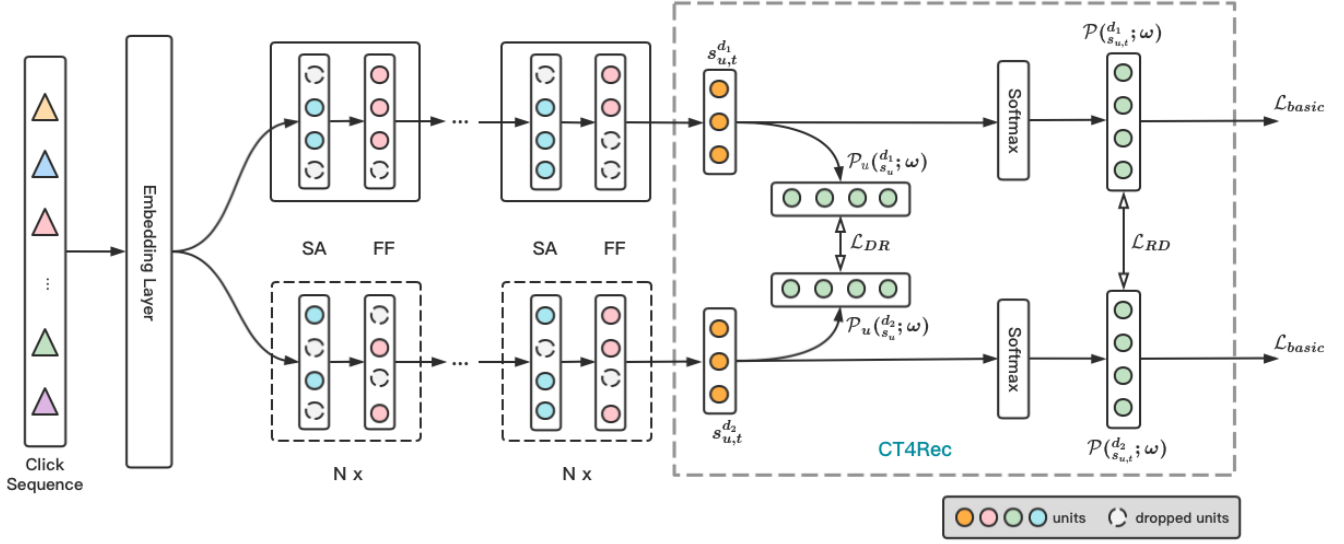
- We propose a simple (with only two bidirectional KL losses) yet very effective consistency training method for sequential recommendation systems. To the best of our knowledge, this is the first work to thoroughly study the effect of consistency training from different perspectives and with a unified training objective for the sequential recommendation task.
- Our proposed consistency training method can be easily extended to other inconsistency scenarios and tasks, e.g., data augmentation and CTR prediction.
- Extensive experiments on four offline datasets show the effectiveness of our proposed CT4Rec over SOTA models based on contrastive learning with much better performance and faster convergence time. The Online A/B test also shows significant improvement over the strong ensemble model.

2 RELATED WORK

Early sequential recommendation (SR) methods are usually based on Markov Chain (MC), including adopting first-order MC [40] and high-order MCs [12, 13]. As for current deep learning approaches, they can be generally divided into four categories, i.e., RNN-based [15, 16, 21, 26, 31, 32, 38, 44], CNN-based [45, 46, 52], attention-based [24, 27, 43, 56, 57, 60] and GNN-based [25, 37, 48, 50, 51, 61] methods. Concretely, GRU4Rec [15] applies RNN to SR and many variants have been proposed based on GRU4Rec by adding data augmentation GRU4Rec+ [44], hierarchical RNN [38] and attention module [2, 26, 34]. However, RNN-based methods usually exhibit worse performance than CNN-based and attention-based methods for the data sparsity setting. Caser [45] proposes a convolutional sequence embedding recommendation model, and [46] use 3-dimensional CNNs to achieve character-level encoding of input data. Meanwhile, attention mechanisms [47] is used to model user behavior sequences and have achieved outstanding performance [7, 23, 24, 27, 33, 43]. Besides, many researches [37, 50, 51] combine attention mechanisms and GNNs to solve the SR task. Memory networks [3, 18], data augmentation [9, 54] and session-based [17] models are also utilized to improve the performance of SR.

Recently, the combination of contrastive learning and attention mechanisms [53, 62] is widely used in SR and achieves great success. CauseRec [59] performs contrastive learning by contrasting the counterfactual with the observational. StackRec [49] utilizes stacking and fine-tuning to develop a deep but easy-to-train SR model. ICAI-SR [58] focuses on the complex relations between items and categorical attributes in SR. Unlike previous researches, we only introduce two consistency training objectives in the representation and output spaces without any structure modifications, extra data, and heuristic patterns from tasks. Though extremely simple, our method outperforms recent SOTA models by a large margin and is more efficient for model training.

There are only a few related researches in the fields of natural language process [8] and machine learning [29, 35, 64]. Specifically, [8] introduce dropout as the alternative of data augmentation into contrasting learning for sequence representation learning while we focus on the inconsistency introduced by dropout in the data

Figure 1: Model structure of CT4Rec. It takes user click sequences as input and outputs user representations for item retrieval in the matching stage of recommendation. The input sequences are transformed into vector representations via the embedding layer and then encoded by N transformers with different hidden dropout masks. In addition, Distributed Regularization Loss and Regularized Dropout Loss are introduced to restrain these representations generated by different dropout masks.

sparsity SR task. Our proposed method is also extremely simple compared with the paradigm of combining contrastive learning with the traditional item prediction objective. [35, 64] mainly focus on the gap between training and testing and utilize L2 for regularizing the representation space, which is less effective in the data sparsity setting, represented by the marginal to none performance improvements in Section 5. Different from introducing a regularization objective in the output space to constrain the randomness of sub-models brought by dropout [29], we focus on the consistency training of the data sparsity SR task from both the representation and output space. We also propose a simple yet effective regularization strategy in the representation space to compensate and align the output space consistency loss.

3 THE CT4REC MODEL

The overall structure of our model is illustrated in Figure 1. Before elaborating our proposed CT4Rec, we first present some necessary notations to describe the sequential item prediction task. Let $\mathcal{U} = (u_1, u_2, ..., u_{|\mathcal{U}|})$ denote a set of users, and $\mathcal{V} = (v_1, v_2, ..., v_{|\mathcal{V}|})$ denote a set of items. The sequence for user $u \in \mathcal{U}$ is denoted as $s_u = (v_1^{(u)}, v_2^{(u)}, ..., v_t^{(u)}, ..., v_{|s_u|}^{(u)})$, where $v_t^{(u)} \in \mathcal{V}$ is the item that user u interacts at time step t and $|s_u|$ is the length of sequence s_u. Given the historical sequence s_u, the task of sequential recommendation is to predict the probability of all alternative items to be interacted by user u at time step $|s_u| + 1$, which is formulated as $P(v_{|s_u|+1}^{(u)} = v|s_u)$.

3.1 Backbone Model

Since our proposed consistency training method does not involve structural modification and extra data utilization, we apply it to the widely used SASRec model [24]. Following the original setting in SASRec, the transformer encoder contains three parts, i.e., an embedding layer, the stacked multi-head self-attention blocks, and a prediction layer. Then, we can obtain user representation $s_u = f(s_u)$, where $f(\cdot)$ indicates the transformer encoder.

To learn the relation between users and items in sequential recommendation, a similarity function, e.g., inner product, is applied to measure distances between user representation and item representation. Thus, for user representation $s_{u,t}$ of user u at time step t, we can get a similarity distribution $\mathcal{P}(s_{u,t}) = \mathcal{P}(v_{t+1}^{(u)}|s_{u,t})$ to predict the item that user u will interact at time step $t + 1$. Then, the basic loss function with positive item v_{t+1}^+ and randomly sampled negative items $v_{t+1}^- \in \mathcal{V}$ is denoted as:

$$\mathcal{P}(s_{u,t}; \omega) = \frac{exp(s_{u,t}v_{t+1}^+)}{exp(s_{u,t}v_{t+1}^+) + \sum_{v_{t+1}^- \in \mathcal{V}} exp(s_{u,t}v_{t+1}^-)} \quad (1)$$

$$\mathcal{L}_{basic}(s_{u,t}; \omega) = -log\mathcal{P}(s_{u,t}; \omega) \quad (2)$$

, where ω refers to all trainable parameters of the model.

3.2 Consistency Training

Since the over-fitting problem extensively exists in deep neural network models, regularization methods, including dropout, are widely used to alleviate this problem. Commonly, dropout can reduce over-fitting and co-adapting by randomly removing a certain rate of units in the whole deep neural network. Also, dropout can be treated as a method to generate and combine exponentially sub-models, which always effectively enhances model performance. Considering the above advantages and the randomness of dropout, we propose our CT4Rec based on dropout to regularize both the
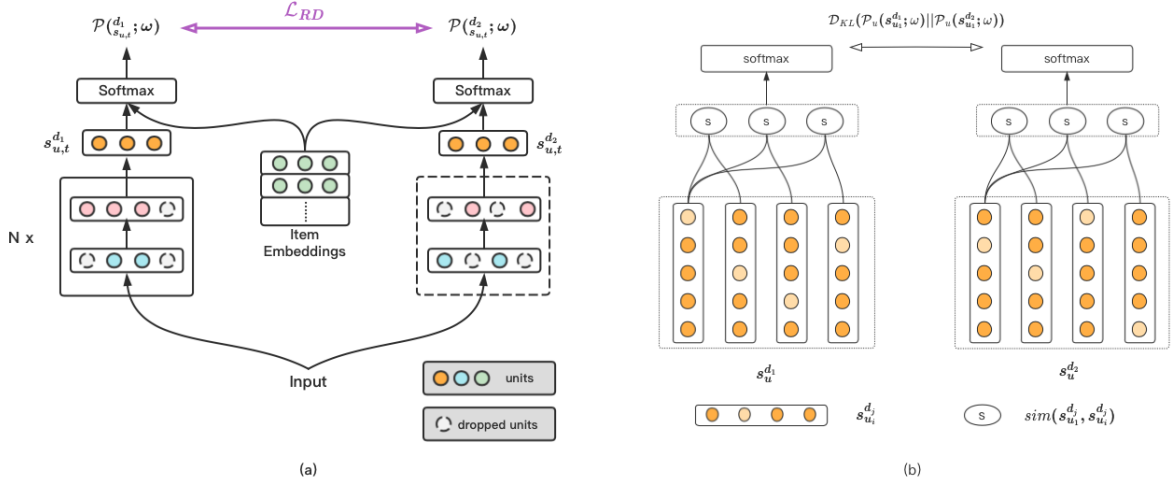
Figure 2: Illustration of (a) RD loss and (b) DR loss.

output space and the representation space of models. Inspired by recent studies on dropout [29], we enhance the user representation from the perspective of reducing the model inconsistency and gap between training and testing.

Concretely, we forward twice with different dropouts and learn the consistency between these two representations for each user, i.e., each user interaction sequence s_u passing the forward network twice and obtain two representations $s_{u,t}^{d_1}$ and $s_{u,t}^{d_2}$. Since dropout randomly removes units in a model, the two representations are actually generated from two sub-models of the same model.

Regularized Dropout Loss (RD). We first apply a regularized dropout loss to constrain the output space of sub-models from dropout. Considering two representations $s_{u,t}^{d_1}$ and $s_{u,t}^{d_2}$ for user u, as mentioned in Function 1, we can get two similarity distributions $\mathcal{P}(s_{u,t}^{d_1};\omega)$ and $\mathcal{P}(s_{u,t}^{d_2};\omega)$. Then, we introduce a bidirectional KL-divergence loss to regularize the above two distributions:

$$\mathcal{L}_{RD}(s_{u,t};\omega) = \frac{1}{2}(\mathcal{D}_{KL}(\mathcal{P}(s_{u,t}^{d_1};\omega)||\mathcal{P}(s_{u,t}^{d_2};\omega)) \\ + \mathcal{D}_{KL}(\mathcal{P}(s_{u,t}^{d_2};\omega)||\mathcal{P}(s_{u,t}^{d_1};\omega))) \quad (3)$$

As shown in Figure 2(a), the dropped units of the left model to generate user presentation $s_{u,t}^{d_1}$ are different from that of the right model to generate $s_{u,t}^{d_2}$. Therefore, the similarity distributions $\mathcal{P}(s_{u,t}^{d_1};\omega)$ and $\mathcal{P}(s_{u,t}^{d_2};\omega)$ are also varied for the same input sequence s_u.

Distributed Regularization Loss (DR). To better regularize the representation space, we propose a distributed regularization method in which each user is represented by its correlations with other users rather than directly utilizing user representations for consistency regularization. In this paper, we compare users in each mini-batch, e.g., n users $(u_1, u_2, ..., u_n)$ and two representations for each user generated by dropout denoted as $(s_{u_1}^{d_1}, s_{u_2}^{d_1}, ..., s_{u_n}^{d_1})$ and $(s_{u_1}^{d_2}, s_{u_2}^{d_2}, ..., s_{u_n}^{d_2})$. As shown in Figure 2(b), for user u_1, we calculate the similarities between $s_{u_1}^{d_j}$ and all the other user representation

$s_{u_i}^{d_j}$, defined as $sim(s_{u_1}^{d_j}, s_{u_i}^{d_j})$, and obtain the similarity distribution

$$\mathcal{P}_u(s_{u_1}^{d_j};\omega) = softmax(sim(s_{u_1}^{d_j}, s_{u_2}^{d_j}), ..., sim(s_{u_1}^{d_j}, s_{u_n}^{d_j})) \quad (4)$$

Then, a bidirectional KL-divergence loss is applied to regularize the two distributions $\mathcal{P}_u(s_{u_1}^{d_1};\omega)$ and $\mathcal{P}_u(s_{u_1}^{d_2};\omega)$, defined as:

$$\mathcal{L}_{DR}(s_u;\omega) = \frac{1}{2}(\mathcal{D}_{KL}(\mathcal{P}_u(s_u^{d_1};\omega)||\mathcal{P}_u(s_u^{d_2};\omega)) \\ + \mathcal{D}_{KL}(\mathcal{P}_u(s_u^{d_2};\omega)||\mathcal{P}_u(s_u^{d_1};\omega))) \quad (5)$$

Final Objective. As shown in Figure 1, we train the above two objectives together with the task-specific loss in the backbone model. The task-specific loss and final training objective are as:

$$\mathcal{L}_{basic}(s_{u,t};\omega) = -\frac{1}{2}(log\mathcal{P}(s_{u,t}^{d_1};\omega) + log\mathcal{P}(s_{u,t}^{d_2};\omega)) \quad (6)$$

$$\mathcal{L}_{final} = \mathcal{L}_{basic} + \alpha\mathcal{L}_{RD} + \beta\mathcal{L}_{DR} \quad (7)$$

where α and β are the coefficient weights to control $\mathcal{L}_{RD}$ and $\mathcal{L}_{DR}$. Thus, our CT4Rec can control the influence of dropout and constrain the model space. Compared with Equation 2, our CT4Rec only adds two losses $\mathcal{L}_{RD}$ and $\mathcal{L}_{DR}$ with model structures unchanged, which can also be widely applied on various model structures. We present the training algorithm in the Appendix A.2.

4 EXPERIMENTS

4.1 Datasets

We conduct extensive experiments on three public benchmark datasets that are widely used in recent literature [53] and a new large-scale dataset collected from a real-world recommendation scenario, i.e., PC-WeChat Top Stories. These datasets are very different in domains, platforms, and data scale, where their detailed statistics are presented in Table 1.

- **Amazon:** a series of datasets comprise product reviews, which are crawled from one of the largest E-Commerce platforms, i.e., *Amazon.com*. As introduced in SASREC [24, 36], these datasets are separated by top-level product categories.

Table 1: Dataset statistics of three public benchmarks and an offline corpus from real-world application, where *avg.* refers to the average actions per user.

Dataset	#users	#items	#actions	avg.	density
Beauty	52,024	57,289	0.4M	7.6	0.01%
Sports	25,598	18,357	0.3M	8.3	0.05%
Yelp	30,431	20,033	0.3M	10.4	0.05%
WeChat	749,452	211,004	9.5M	12.7	0.006%

We follow one of the most recent researches [53] to utilize the **Beauty** and **Sports** categories for comparison.

- **Yelp:** it is a widely acknowledged dataset for the business recommendation, which is collected from the Yelp platform[1]. Following [63], we leverage the data after January 1st, 2019, and treat business categories as attributes.

- **WeChat:** this dataset is constructed from WeChat platform for PC Top Stories recommendation (denoted as **WeChat** to distinguish datasets from other platforms), which consists of interaction records from 7th to 13rd, June 2021. Each interaction provides positive feedback (i.e., click) of an item from a user. Concretely, we collect 9.5 million interactions from 0.74 million users on 0.21 million items and regard data from the first few days as the train set and the rest for testing.

4.2 Baselines

To verify the effectiveness of our proposed method, we introduce four representative baselines and three very recent methods.

- **GRU4Rec [15].** It utilizes GRU modules to model user action sequences for the session-based recommendation. We consider each user's click sequence as a session.

- **SASRec [24].** It applies the multi-head self-attention mechanism to solve the sequential recommendation task, which is commonly treated as one of the state-of-the-art baselines. In this paper, we utilize SASRec as the backbone of our CT4Rec.

- **TiSASRec [27].** Based on SASRec, it further introduces time interval aware self-attention mechanism to encode the user's interaction sequence, where the positions and time interval between any two items are considered.

- **BERT4Rec [43].** It uses the deep bidirectional self-attention mechanism to model user interaction history in the sequential recommendation and trains the model like BERT [6].

- **CL4SRec [53].** It generates different views of the same user interaction sequence by using data augmentation methods and adds contrastive learning objective to the original objective of SASRec for sequential recommendation tasks.

- **CLRec.** [62] design a queue-based contrastive learning method named CLRec to de-bias deep candidate generation in the recommendation system and further propose Multi-CLRec for multi-intention aware bias reduction. In this work, we only compare the CLRec for fairness.

- **StackRec [49].** It first uses a stacking operation on the pre-trained layers/blocks to transfer knowledge from a shallow model to a deep model and then utilizes iterative stacking to obtain a deeper but easier-to-train recommendation model.

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

4.3 Settings

All models are implemented based on TensorFlow. For baselines with official codes, we utilize the implementations provided by authors. As for models without open-accessible codes from the original paper, we prefer the well-tested version from the open-source community. Specifically, we use code from https://github.com/Songweiping/GRU4Rec_TensorFlow as the implementation of GRU4Rec [15]. We implement CL4SRec and CLRec based on the model descriptions and experimental settings of the correlated papers since there is no official code or popular implementation from the open-source community. For fair comparisons, the embedding dimension size is set to 50, and all models are optimized by Adam.

Recall that our proposed CT4Rec method does not modify the model architecture and increases the model scale of the backbone model. Instead, it only involves two effective consistency regularization strategies. Thus, we follow the backbone method, i.e., SASRec [24], to implement our CT4Rec. We use two self-attention layers and set the head number to 2. The maximum sequence length is 50 for all datasets. We optimize the parameters with the learning rate of 0.001 and the batch size as 128. The dropout rate of turning off neurons is set to 0.5. To verify the effect of each component and their combination, we fix the structure and other hyper-parameters of the model and only adjust the values of α and β, where α and β are selected from $\{0.1, 0.3, 0.5, 1.0, 2.0, 3.0\}$. Other details for reproducing our experiments can be found in our anonymous code.

4.4 Offline Evaluation

4.4.1 Evaluation Protocols. Following many previous studies [14, 24, 53, 63], we employ the leave-one-out strategy to evaluate model performance. Specifically, for each user, we take the last interacted item for test. Similar to [24, 63], we randomly sample 500 items from the whole dataset for each positive item, and rank them by similarity scores. The model performances are evaluated by top-k Normalizedrand Discounted Cumulative Gain (NDCG@k) and top-k Hit Ratio (HR@k), which are both commonly used in top-k recommendation systems. Specifically, we report HR@k and NDCG@k with $k = \{5, 10, 20\}$ for all datasets.

4.4.2 Experimental Results. As shown in Table 2, our proposed CT4Rec outperforms all other baseline methods, including multiple representative models and state-of-the-art sequential recommendation solutions, on three benchmark datasets and one large industrial corpus. Compared with other strong methods that utilize data augmentation or/and contrastive learning to obtain better user representations and mitigate the incompatibility between the single item prediction task and the vast amounts of parameters in data sparsity scenarios [24], our CT4Rec is simpler and more effective without requiring any augmented data or delicate training strategies in which only two extra objectives are introduced. The performance advance of CT4Rec over SOTA models based on contrastive learning is confirmed by the significant improvements of HR@k, and NDCG@k scores over CLRec and CL4SRec. We also observe that the training objective in recent baselines performs much better than the original binary cross-entropy in *SASRec*, which is demonstrated by the results that *SASRec** increase offline scores by a large margin over *SASRec*. Notice that our CT4Rec is implemented by introducing two training objectives into *SASRec**. We

Table 2: Model performance of baselines and our proposed CT4Rec on four offline datasets, where '*' refers to modifying the original binary cross-entropy loss in SASRec with the training objective in recent baselines, e.g., CLRec, CL4SRec, StackRec. Our CT4Rec is implemented on SASRec*. *Improv.* and *Improv. refer to the relative improvement of CT4Rec over SASRec and SASRec*, respectively. The performance improvement over baselines is statistically significant with $p < 0.01$, in which we present the experimental results of extra two runs with different random seeds in the Appendix A.4.**

Datasets	Metric	SASRec	SASRec*	GRU4Rec	BERT4Rec	TiSASRec	StackRec	CLRec	CL4SRec	CT4Rec	Improv.	Improv.*
Beauty	HR@5	0.2109	0.2194	0.1179	0.0860	0.2024	0.1725	0.1498	0.2275	**0.2556**	21.19%	16.50%
	HR@10	0.2759	0.2748	0.1582	0.1357	0.2746	0.2225	0.1816	0.2896	**0.3200**	15.98%	16.45%
	HR@20	0.3546	0.3392	0.2146	0.2036	0.3508	0.2818	0.2218	0.3624	**0.3891**	9.73%	14.71%
	NDCG@5	0.1523	0.1661	0.0869	0.0572	0.1413	0.1286	0.1177	0.1701	**0.1924**	26.33%	15.83%
	NDCG@10	0.1733	0.1840	0.1000	0.0732	0.1646	0.1447	0.1279	0.1901	**0.2132**	23.02%	15.87%
	NDCG@20	0.1932	0.2003	0.1141	0.0903	0.1839	0.1596	0.1380	0.2085	**0.2307**	19.41%	15.18%
Sports	HR@5	0.1912	0.1966	0.0961	0.0766	0.1703	0.1318	0.1498	0.2084	**0.2196**	14.85%	11.70%
	HR@10	0.2747	0.2683	0.1499	0.1267	0.2456	0.1962	0.2178	0.2834	**0.3010**	9.57%	12.19%
	HR@20	0.3751	0.3547	0.2295	0.2055	0.3352	0.2780	0.3072	0.3721	**0.3950**	5.31%	11.36%
	NDCG@5	0.1289	0.1398	0.0631	0.0494	0.1159	0.0902	0.1044	0.1488	**0.1556**	20.71%	11.30%
	NDCG@10	0.1558	0.1629	0.0804	0.0654	0.1402	0.1110	0.1263	0.1729	**0.1817**	16.62%	11.54%
	NDCG@20	0.1811	0.1847	0.1003	0.0852	0.1628	0.1315	0.1488	0.1953	**0.2055**	13.47%	11.26%
Yelp	HR@5	0.2834	0.3216	0.1457	0.1567	0.2935	0.2230	0.2545	0.3173	**0.3462**	22.16%	7.65%
	HR@10	0.4221	0.4469	0.2546	0.2623	0.4257	0.3397	0.3881	0.4451	**0.4784**	13.34%	7.05%
	HR@20	0.5975	0.5989	0.4257	0.4312	0.5839	0.4927	0.5670	0.5993	**0.6309**	5.59%	5.34%
	NDCG@5	0.1889	0.2283	0.0890	0.0996	0.2009	0.1484	0.1702	0.2236	**0.2443**	29.33%	7.01%
	NDCG@10	0.2335	0.2687	0.1239	0.1336	0.2435	0.1859	0.2132	0.2647	**0.2869**	22.87%	6.77%
	NDCG@20	0.2778	0.3070	0.1668	0.1759	0.2834	0.2244	0.2582	0.3036	**0.3253**	17.10%	5.96%
WeChat	HR@5	0.2756	0.3069	0.1836	0.1943	0.3193	0.2975	0.2849	0.3105	**0.3406**	25.58%	10.98%
	HR@10	0.4103	0.4366	0.2231	0.2247	0.4406	0.4173	0.3985	0.4511	**0.4861**	18.47%	11.34%
	HR@20	0.5291	0.5484	0.2884	0.2907	0.5539	0.5197	0.4852	0.5507	**0.5979**	13.00%	9.03%
	NDCG@5	0.1948	0.2131	0.1272	0.1266	0.2036	0.2082	0.1939	0.2195	**0.2361**	21.20%	10.79%
	NDCG@10	0.2357	0.2743	0.1447	0.1394	0.2615	0.2576	0.2371	0.2827	**0.3057**	29.70%	11.40%
	NDCG@20	0.2869	0.3013	0.1693	0.1526	0.2957	0.2811	0.2639	0.3089	**0.3314**	15.51%	9.99%

compare *CT4Rec* with *SASRec** to calibrate the effect of our consistency training method. The universal enhancement of *CT4Rec* over *SASRec** on four datasets and all HR@k and NDCG@k (relative improvements ranging from 5.34% to 16.50%) verify the superiority of our proposed simple method.

4.5 Online Evaluation

4.5.1 Evaluation Protocols. We further perform an online A/B test that resembles [11] to evaluate our CT4Rec in a real-world system. Commonly, an online recommendation system is divided into four stages, including matching, pre-ranking, ranking, and re-ranking. We have deployed CT4Rec for *PC Wechat Top Stories* recommendation in the matching stage. To calibrate the effect of CT4Rec for the online system, CT4Rec is implemented as an additional channel in the current matching module with the rest of the system unchanged, where the existing matching module refers to an ensemble model that combines multiple methods, e.g., rule-based, reinforcement-based, sequence-based, DSSM, self-distillation. In the online A/B test, we utilize two metrics, i.e., click-through rate (CTR) and average click number per capita (ACN), to evaluate model performance. The online test lasts for 7 days and involves nearly 3 million users.

4.5.2 Experimental Results. The performance improvement of CT4Rec on real-world recommendation service is reported in Table 3 with
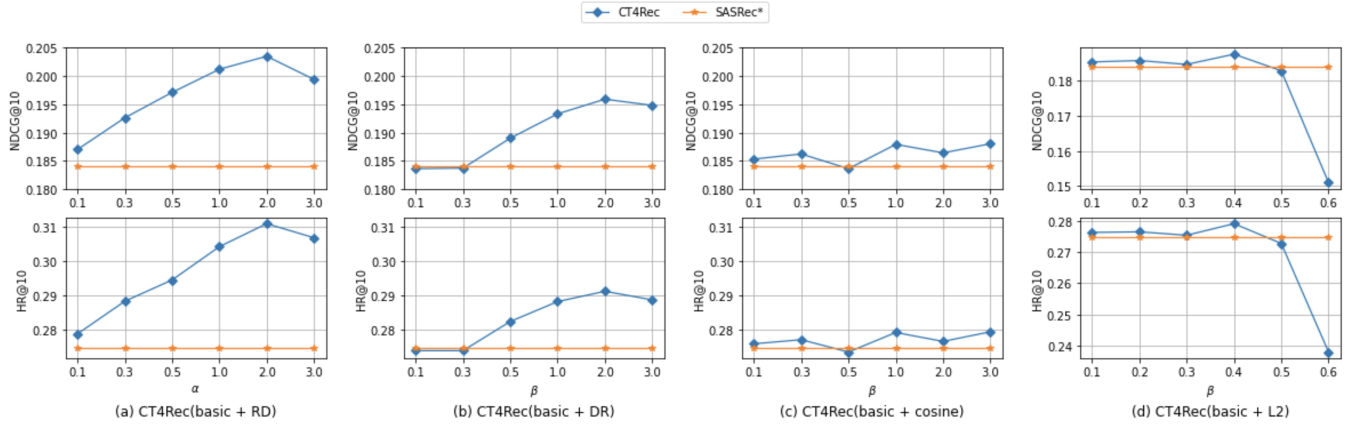
Table 3: Performance improvement of online A/B test.

CTR	ACN
+2.717%	+3.679%

the significance level $p < 0.01$, which can be seen that only implementing CT4Rec as an additional channel in the matching module with the rest of the system unchanged can yield very impressive performance improvement over the original ensemble model. Our method increases CTR and ACN metrics by 2.717% and 3.679%, respectively. This indicates that our proposed simple method is not only effective for offline benchmarks and evaluation metrics but also generalizes well to the real-world online system. Since CT4Rec restricts the uncertainty of sub-models without involving extra model structure, the computation cost of CT4Rec for online serving is identical to the original backbone model. Thus, CT4Rec can significantly enhance online performance without adding online computation costs, which is a definite advantage for online serving, especially considering the machine costs.

5 ANALYSIS

We have demonstrated the impressive performance of our proposed simple CT4Rec on offline benchmarks and the online A/B test. In

Figure 3: Evaluation results for ablation study and analysis of α and β. (a) CT4Rec with $\mathcal{L}_{basic}$ and $\mathcal{L}_{RD}$. (b) CT4Rec with $\mathcal{L}_{basic}$ and $\mathcal{L}_{DR}$. (c) and (d) replace $\mathcal{L}_{DR}$ with cosine and L2 loss, respectively, where the orange lines are the performance of SASRec*.

this section, we launch extensive experiments to understand and analyze CT4Rec from different perspectives. For convenience, these studies are performed on *Beauty*. More specifically, we mainly focus on: **1)** the effect of each introduced objective (*Ablation Study*), **2)** the influence of several important hyper-parameters (*Hyper-Parameter Analysis*), **3)** the extension of CT4Rec to data augmentation (*Extension to Data Augmentation*), **4)** the change of training process and cost resulted by CT4Rec (*Training and Cost Analysis*)
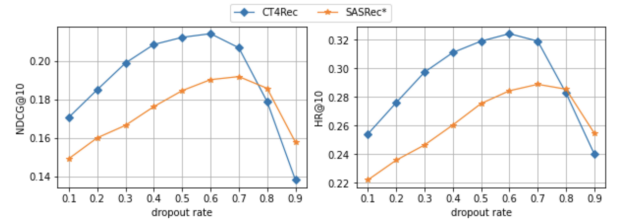
5.1 Ablation Study

We perform an ablation study to explore the effect of two objectives in Section 3, i.e., regularized dropout (RD) and distributed regularization (DR), where the results are illustrated in Figure 3.

RD Objective. As shown in the left sub-figure (a), the introduced RD objective achieves significant performance improvement over the backbone *SASRec** model. Compared with DR objective and its variants, RD loss contributes most to the performance increase, which concludes that launching consistency regularization in the model output space is the most beneficial to *SASRec**, which is similar to the observations in other tasks [29]. The possible reason might be that RD consistency regularization affects directly on the same space of the model output probability distribution.

DR Objective. We can see that the unsupervised DR loss can also yield substantial performance gains, which points out that unsupervised consistency regularization in user representation space for data sparsity sequential recommendation task is also essential. We also adapt two typical unsupervised strategies in previous studies [8, 35, 64] to regularize user representations, including cosine similarity and L2 distance. As shown in the right two sub-figures in Figure 3, these two methods have not brought meaningful improvement upon the backbone method, which further proves that our designed DR objective is more preferable for consistency regularization in the representation space.

5.2 Hyper-Parameter Analysis

We mainly consider several critical hyper-parameters in this part, including α and β in Equation 7, and the dropout rate.

Figure 4: The impact of different dropout rates for CT4Rec and SASRec* on the Beauty dataset.

The Effect of α. Figure 3 also gives the results of different α values. Considering that there are many feasible combinations of the (α, β) grid, we temporarily remove the DR objective and only examine the influence of α for RD. It can be observed that a small value of α can bring meaningful performance improvement. With the increase of α, the performance improvement further increases in which the best result is achieved when $\alpha = 2.0$. These results further confirm that the consistency regularization directly performed on the output space is very effective even when it only takes a small proportion for the final training objective. With a further increase of α, the model will pay more attention to the consistency of model outputs, which will dilute the original item prediction objective, resulting in worse performance (e.g., $\alpha = 2.0$ v.s., $\alpha = 3.0$).

The Influence of β. Similar to the analysis of α, we only study the impact of β for each single unsupervised regularization loss (i.e., DR, cosine, and L2). Different from α, the DR loss has no influence on the overall performance when the value of β<0.3. This is probably because the consistency regularization on the user representations is overwhelmed and adapted when passing to the output space. With the increase of β, DR loss gradually produces a more important role and consistently improves model performance until β>2.0. And also, a large β value will force the model to focus on the consistency of user representation rather than perform the item prediction task. For the cosine objective, different β values have a limited influence on evaluation results. As for the L2 loss, a large β value will cause inferior performance.
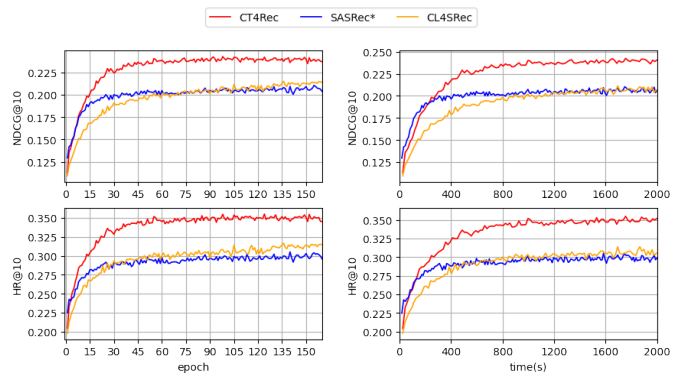
Table 4: Performance comparison of different consistency regularization strategies in the data augmentation setting.

Aug.[53]	Metrics	SASRec	SASRec*	+CL	+L2	+Cos	+DR	+RD	+CT4Rec	Improv.	Improv.*
Reorder	HR@10	0.2759	0.2748	0.2768	0.2867	0.2856	0.2905	0.2857	**0.3076**	11.49%	11.94%
	HR@20	0.3546	0.3392	0.3407	0.3552	0.3542	0.3558	0.3546	**0.3776**	6.49%	11.32%
	NDCG@10	0.1523	0.1661	0.1856	0.1892	0.1900	0.1954	0.1899	**0.2047**	34.41%	23.24%
	NDCG@20	0.1932	0.2003	0.2017	0.2064	0.2073	0.2119	0.2072	**0.2224**	15.11%	11.03%
Mask	HR@10	0.2759	0.2748	0.2803	0.2831	0.2898	0.2892	0.2891	**0.3140**	13.81%	14.26%
	HR@20	0.3546	0.3392	0.3452	0.3484	0.3553	0.3542	0.3609	**0.3868**	9.08%	14.03%
	NDCG@10	0.1523	0.1661	0.1861	0.1879	0.1941	0.1948	0.1886	**0.2061**	35.33%	24.08%
	NDCG@20	0.1932	0.2003	0.2025	0.2044	0.2106	0.2111	0.2067	**0.2244**	16.15%	12.03%

Analysis on Dropout Rates. Besides the above analysis, we also study how the dropout rate affects the effectiveness of our CT4Rec since different dropout rates will lead to varying degrees of inconsistency. As demonstrated in Figure 4, dropout rates have a significant influence on the performance of the backbone model, and our proposed consistency training is applicable and effective when the dropout rate<0.7. However, our proposed CT4Rec will lead to a negative effect when the dropout rate>0.8. We speculate that the reason behind this might be the data sparsity issue and irreconcilable inconsistency.

5.3 Extension to Data Augmentation

The above studies prove the effectiveness of each component of our CT4Rec and its capability of leveraging more consistency regularization signals. We then study its generalization ability to other inconsistency scenarios. More concretely, we utilize our consistency training method to regularize the inconsistency brought by data other than the above-mentioned model-side inconsistency, i.e., we replace dropout with data augmentation methods to create two different user representations, and the corresponded task outputs for each user so as to conduct consistency regularization. To align with recent sequential recommendation methods based on constrastive learning, we leverage two easily implemented augmentation strategies from CL4SRec [24], i.e., *Reorder* and *Mask*. In doing so, we can directly compare the effects of different training objectives (besides the item prediction one for the sequential recommendation) on the augmented data, including unsupervised methods (i.e., CL, cosine, L2, DR), the supervised regularization in the output space (RD), and the combination of DR and RD (CT4Rec). Table 4 presents the experimental results for the data augmentation setting. We can observe that: 1) Our introduced DR objective is the most effective compared with other single methods, i.e., the consistency regularization on the representation space is the most preferable for than data augmentation scenario, which is in contrast with the observation for the dropout setting. We speculate that the label-invariant data augmentation methods can lead to permuted and perturbed representation variants, which need more consistency regularization, while the label-invariant strategy does not deteriorate the inconsistency in the output space. 2) The combination of the consistency regularization in the representation space and the output space (CT4Rec) still performs the best with consistent and significant performance over other training objectives.

Figure 5: NDCG@10 and HR@10 curves on the valid set along with training epoch and time on the Beauty dataset.

5.4 Training and Cost Analysis

Since our CT4Rec does not modify the model structure of the backbone model or introduce extra augmented data, we mainly analyze the changes in the training process. We plot the curves of HR@10and NDCG@10 scores on the valid set along the training epoch number and time (seconds) for *SASRec**, *CL4SRec* and our CT4Rec models, shown in Figure 5. At the early training stage, the backbone *SASRec** model converges quickly with the same training epochs (around 15 epochs) and model performance as CT4Rec, but our CT4Rec can continuously improve the performance on the well-trained *SASRec** model. It concludes that our consistency training objectives do not lead to more training epochs on the backbone *SASRec**. But for *CL4SRec* that combines contrastive learning objective with *SASRec**, it converges with much more training epochs and only moderate performance improvement. As for convergence time (seconds), our model indeed is slower than the backbone model since our consistency training objectives need an extra forward process (dropout) in each training step but achieves a much superior performance. Compared with *CL4SRec*, our CT4Rec is much more efficient and effective with a much better final optimum and less convergence time even when we haven't calculated the time cost of data augmentation and negative sampling for *CL4SRec*.

Through the analysis, we can conclude that: 1) CT4Rec is more efficient and effective than the method based on contrastive learning even without counting its time cost of data augmentation and negative sampling; 2) CT4Rec indeed introduces extra training time

Table 5: Dataset statistics of two corpora from real-world recommendation system in WeChat platform.

Dataset	#Instances	#Fields	#Features
Wechat-Video	41M	39	40.28M
Wechat-Article	138M	25	21.52M

for the backbone model, which can be mitigated by early stop insomuch as our CT4Rec can quickly surpass the backbone model in the early stage and with a much better final convergence performance.

6 EXTENSION TO CTR PREDICTION

Besides the application of CT4Rec on the matching stage, we further explore the effectiveness of our **C**onsistency **T**raining for the **CTR** prediction task denoted as CT4CTR, serving on the ranking stage. Concretely, for each instance $(x, y) \in \mathcal{D}$, x denotes a multi-filed feature vector input, and label $y \in \{0, 1\}$ indicates whether the user clicks the item. The CTR prediction is to obtain the probability $\hat{y}$ that a user will click a certain item in a given context.

We utilize a widely used structure DeepFM in this section, which simply applies two components (i.e., FM component and deep component) and still achieves promising performance in the industry. For each input x, we forward it twice in the deep component of DeepFM with different dropouts and learn the consistency between these two sub-models. Similar to Sec. 3, two regularization methods (i.e. RD loss, DR loss) are utilized to constrain the two sub-models generated from dropout. Concretely, input x passes the deep component twice and obtains $\hat{y}_{DNN}^{d_1}$ and $\hat{y}_{DNN}^{d_2}$ with different dropouts. Thus, with the unchanged part $\hat{y}_{FM}$, the final prediction can be defined as $\hat{y}^{d_1}$ and $\hat{y}^{d_2}$ and a two-dimensional distribution $\mathcal{P}(x^{d_i}) = (\hat{y}^{d_i}, 1-\hat{y}^{d_i})$ can be formulated to calculate RD loss. Meanwhile, we compare the deep representations of instances in each mini-batch, e.g., $(x_1^{d_1}, x_2^{d_1}, \ldots, x_n^{d_1})$ and $(x_1^{d_2}, x_2^{d_2}, \ldots, x_n^{d_2})$. Similar to the matching task, for instance, x_1, we can calculate the similarity $sim(x_1^{d_j}, x_i^{d_j}), i = 2, 3, \ldots, n$, so as to obtain similarity distribution between deep representations denoted as $\mathcal{P}_s(x_1^{d_j})$. Then, similar to Eq. 5, a bidirectional KL-divergence loss to regularize the two distributions $\mathcal{P}_s(x_1^{d_1})$ and $\mathcal{P}_s(x_1^{d_2})$ can be denoted as DR loss.

We compare CT4CTR with: (1) LR [41], a simple baseline model for CTR prediction, which only models the linear combination of raw features. (2) FM [39]. Since LR fails to capture non-linear feature interactions, the factorization machine (FM) has been proposed to model second-order feature interactions. It is notable that FM only has a linear time complexity in terms of the number of features. (3) Wide&Deep[4]. With the development of deep models, Google achieves great improvement by combining a wide (or shallow) network and a deep network. This is a general learning framework that can achieve the advantages of both wide networks and deep networks. (4) DeepFM [10], which extends Wide&Deep by substituting LR with FM to precisely model second-order feature interactions. (5) xDeepFM[28], which captures high-order interactions by its core module, Compressed Interaction Network (CIN). CIN takes an outer product of a stacked feature matrix in a vector-wise way. (6) AutoInt[42], which automatically models the high-order interactions of input features by using self-attention networks.

To evaluate the performance of CT4CTR, we build two private industrial datasets from the WeChat ecosystem: Wechat-Video and Wechat-Article, and both of them are collected from a real-world recommendation scenario, i.e., WeChat Subscriptions. We choose them because they are sampled from real click logs in production, and both have tens of millions of samples, making the results meaningful to industrial practitioners. We split Train and Test sets in chronological order, and Table 5 summarizes the detailed statistics.

Table 6: Offline performance of CT4CTR in the CTR prediction task, with DeepFM as backbone.

WeChat-Video		WeChat-Article	
Model	AUC	Model	AUC
LR	0.7569	LR	0.7401
FM	0.7608	FM	0.7465
Wide&Deep	0.7695	Wide&Deep	0.7538
DeepFM	0.7719	DeepFM	0.7552
AutoInt	0.7703	AutoInt	0.7519
xDeepFM	0.7738	xDeepFM	0.7560
CT4CTR	0.7766	CT4CTR	0.7593

As shown in Table 6, our proposed CT4CTR outperforms all other baseline methods in CTR prediction on AUC, including multiple state-of-the-art CTR prediction solutions, on two large industrial corpora. Besides, we have deployed CT4CTR on a real-world recommendation system that serves nearly one billion users with dramatically high online machine costs. In the online A/B test, we achieve +1.203% on the Video scenario and +2.341% on the Article scenario, as for the CTR metric. Moreover, it is a remarkable success that CT4CTR can enhance online performance without involving extra computation costs and machine costs, which is essential for online serving. Besides, the improvement of such a mature system that already has stable and advanced online models is extremely challenging, which further proves the effectiveness of CT4CTR.

Table 7: Performance improvement of online A/B test.

Scenario	Video	Article
CTR	+1.203%	+2.341%

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed a simple yet very effective consistency training method for the sequential recommendation task, namely CT4Rec, which only involves two bidirectional KL losses. We first introduce a top-performed regularization in the output space by minimizing the bidirectional KL loss of two different outputs. We then design a novel consistency training term in the representation space by minimizing the distributed probability of two user representations. Extensive experiments and analysis demonstrate its effectiveness, efficiency, generalization ability, and compatibility. Besides experiments on the recall task in recommendation systems, further exploration reveals that the introduced consistency training strategies (i.e., DR and RD) are still very effective for the CTR prediction task. In the near future, we will thoroughly study the pros and cons of consistency training for the CTR task.

REFERENCES

[1] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[2] Tong Chen, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li. 2019. Air: Attentional intention-aware recommender systems. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 304–315.

[3] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[7] Xinyan Fan, Zheng Liu, Jianxun Lian, Wayne Xin Zhao, Xing Xie, and Ji-Rong Wen. 2021. Lighter and better: low-rank decomposed self-attention networks for next-item recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1733–1737.

[8] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *arXiv preprint arXiv:2104.08821* (2021).

[9] Chuan Guo, Ali Mousavi, Xiang Wu, Dan Holtmann-Rice, Satyen Kale, Sashank Reddi, and Sanjiv Kumar. 2019. Breaking the glass ceiling for embedding-based classifiers for large output spaces. (2019).

[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[11] Xiaobo Hao, Yudan Liu, Ruobing Xie, Kaikai Ge, Linyao Tang, Xu Zhang, and Leyu Lin. 2021. Adversarial Feature Translation for Multi-domain Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2964–2973.

[12] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 161–169.

[13] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[17] Liang Hu, Longbing Cao, Shoujin Wang, Guandong Xu, Jian Cao, and Zhiping Gu. 2017. Diversifying Personalized Recommendation with User-session Context.. In *IJCAI*. 1858–1864.

[18] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 505–514.

[19] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[20] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2021. A survey on contrastive self-supervised learning. *Technologies* 9, 1 (2021), 2.

[21] How Jing and Alexander J Smola. 2017. Neural survival recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 515–524.

[22] Manas R Joglekar, Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay K Adams, Pranav Khaitan, Jiahui Liu, and Quoc V Le. 2020. Neural input search for large scale recommendation models. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2387–2397.

[23] Taegwan Kang, Hwanhee Lee, Byeongjin Choe, and Kyomin Jung. 2021. Entangled Bidirectional Encoder to Autoregressive Decoder for Sequential Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research*

and Development in Information Retrieval. 1657–1661.

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

[25] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

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

[27] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.

[28] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.

[29] Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-Drop: Regularized Dropout for Neural Networks. *arXiv preprint arXiv:2106.14448* (2021).

[30] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[31] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1053–1058.

[32] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.

[33] Zhiwei Liu, Ziwei Fan, Yu Wang, and Philip S Yu. 2021. Augmenting Sequential Recommendation with Pseudo-Prior Items via Reversely Pre-training Transformer. *arXiv preprint arXiv:2105.00522* (2021).

[34] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2635–2643.

[35] Xuezhe Ma, Yingkai Gao, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard Hovy. 2017. Dropout with expectation-linear regularization. *International Conference on Learning Representations* (2017).

[36] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[37] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the item order in session-based recommendation with graph neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 579–588.

[38] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 130–137.

[39] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[40] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[41] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. 521–530.

[42] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[43] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[44] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 17–22.

[45] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[46] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D convolutional networks for session-based recommendation with content features. In *Proceedings of the eleventh ACM conference on recommender systems*. 138–146.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all

you need. In *Advances in neural information processing systems.* 5998–6008.

[48] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. 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.* 109–118.

[49] Jiachun Wang, Fajie Yuan, Jian Chen, Qingyao Wu, Min Yang, Yang Sun, and Guoxiao Zhang. 2021. StackRec: Efficient Training of Very Deep Sequential Recommender Models by Iterative Stacking. In *Proceedings of the 44th International ACM SIGIR conference on Research and Development in Information Retrieval.* 357–366.

[50] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[51] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation.. In *IJCAI*, Vol. 19. 3940–3946.

[52] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S Sheng S. Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent convolutional neural network for sequential recommendation. In *The world wide web conference.* 3398–3404.

[53] Fei Sun Xu Xie, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. 2021. Contrastive Learning for Sequential Recommendation. (2021).

[54] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al. 2020. Self-supervised Learning for Large-scale Item Recommendations. *arXiv preprint arXiv:2007.12865* (2020).

[55] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al. 2020. Self-supervised learning for deep models in recommendations. *arXiv e-prints* (2020), arXiv–2007.

[56] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based

on hierarchical attention network. In *IJCAI International Joint Conference on Artificial Intelligence*.

[57] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-order attentive ranking model for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5709–5716.

[58] Xu Yuan, Dongsheng Duan, Lingling Tong, Lei Shi, and Cheng Zhang. 2021. ICAI-SR: Item Categorical Attribute Integrated Sequential Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1687–1691.

[59] Shengyu Zhang, Dong Yao, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2021. Causerec: Counterfactual user sequence synthesis for sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 367–377.

[60] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. Atrank: An attention-based user behavior modeling framework for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[61] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable graph embedding for asymmetric proximity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[62] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive learning for debiased candidate generation in large-scale recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* 3985–3995.

[63] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management.* 1893–1902.

[64] Konrad Zolna, Devansh Arpit, Dendi Suhubdy, and Yoshua Bengio. 2018. Fraternal Dropout. In *International Conference on Learning Representations*.

A APPENDIX

A.1 Model structure of CT4CTR

The model structure of CT4CTR is shown in Figure 6. CT4CTR takes user features and item features as input and outputs click probabilities for each user and item pair in the ranking stage of recommendation. For the deep component of DeepFM, input features are transformed into vector representations via the embedding layer and deep neural network with different hidden dropout masks. In addition, Distributed Regularization Loss and Regularized Dropout Loss are introduced to restrain these representations generated by different dropout masks.

Figure 6: Model structure of CT4CTR.

A.2 Training Algorithm

Algorithm 1 CT4Rec algorithm

Input: Training data $\mathcal{D} = \left\{s_{u_i,t}\right\}_{i=1}^{N}$
Output: model parameters ω
  1: Initialization model with parameters ω
  2: **while** not converged **do**
  3:      $s_{u_i,t} \sim \mathcal{D}$
  4:      $s_{u_i,t}^{d_1} \leftarrow f(s_{u_i,t}; \omega)$ with first dropout
  5:      $s_{u_i,t}^{d_2} \leftarrow f(s_{u_i,t}; \omega)$ with second dropout
  6:      $g \leftarrow \nabla_\omega \mathcal{L}_{final}(s_{u_i,t}^{d_1}, s_{u_i,t}^{d_2}; \omega)$
  7:      $\omega \leftarrow GradientUpdate(\omega, g)$
  8: **end while**

The whole training process of CT4Rec is presented in Algorithm 1. As shown in Line 3-5, we obtain two user representations $s_{u_i,t}^{d_1}$

and $s_{u_i,t}^{d_2}$ by going forward the model twice for each user sequence $s_{u_i,t}$. Line 6-7 calculate the $\mathcal{L}_{final}$ according to the loss function 7, and update the model parameters. The training process will continue until convergence.

A.3 More Backbone

To verify the universal influence of CT4Rec, we also apply it to GRU4Rec [15] denoted as CT4Rec^G and evaluate its performance on four datasets. Table 8 shows that CT4Rec brings significant improvement compared with the original GRU4Rec on all datasets and all HR@k and NDCG@k scores (relative improvements ranging from 3.48% to 11.51%), which indicates that our method can be widely applied to different model structures and achieve enhancement.

Table 8: Performance of CT4Rec with GRU4Rec as backbone

Datasets	Metric	GRU4Rec	CT4Rec^G	Improv.
Beauty	HR@5	0.1149	0.1247	8.53%
	HR@10	0.1574	0.1665	5.78%
	HR@20	0.2157	0.2232	3.48%
	NDCG@5	0.0851	0.0923	8.46%
	NDCG@10	0.0987	0.1057	7.09%
	NDCG@20	0.1133	0.1200	5.91%
Sport	HR@5	0.0997	0.1057	6.02%
	HR@10	0.1558	0.1665	6.87%
	HR@20	0.2356	0.2504	6.28%
	NDCG@5	0.0656	0.0699	6.55%
	NDCG@10	0.0836	0.0894	6.94%
	NDCG@20	0.1037	0.1105	6.56%
Yelp	HR@5	0.1460	0.1628	11.51%
	HR@10	0.2570	0.2817	9.61%
	HR@20	0.4335	0.4601	6.14%
	NDCG@5	0.0903	0.1011	11.96%
	NDCG@10	0.1259	0.1392	10.56%
	NDCG@20	0.1702	0.1840	8.11%
WeChat	HR@5	0.1833	0.1946	6.61%
	HR@10	0.2235	0.2378	6.40%
	HR@20	0.2891	0.3042	5.22%
	NDCG@5	0.1267	0.1369	8.05%
	NDCG@10	0.1450	0.1581	9.03%
	NDCG@20	0.1694	0.1804	6.49%

A.4 More Experimental Results

Here, we present the experimental results of extra two runs with different random seeds.

Table 9: The second run of baselines and our proposed CT4Rec on four offline datasets, where '*' refers to modifying the original binary cross-entropy loss in SASRec with the training objective in recent baselines, e.g., CLRec, CL4SRec, StackRec. Our CT4Rec is implemented on SASRec*. *Improv.* and *Improv.** refer to the relative improvement of CT4Rec over SASRec and SASRec*, respectively. Except for the random seed, the other settings are the same as Table 2.

Datasets	Metric	SASRec	SASRec*	GRU4Rec	BERT4Rec	TiSASRec	StackRec	CLRec	CL4SRec	CT4Rec	Improv.	Improv.*
Beauty	HR@5	0.2127	0.2162	0.1184	0.0835	0.2087	0.1685	0.1528	0.2284	**0.2565**	20.59%	18.64%
	HR@10	0.2805	0.2707	0.1592	0.1323	0.2805	0.2167	0.1872	0.2894	**0.3211**	14.47%	18.62%
	HR@20	0.3583	0.3340	0.2157	0.1997	0.3569	0.2779	0.2288	0.3606	**0.3915**	9.27%	17.22%
	NDCG@5	0.1521	0.1639	0.0873	0.0550	0.1450	0.1272	0.1198	0.1704	**0.1927**	26.69%	17.57%
	NDCG@10	0.1740	0.1815	0.1004	0.0707	0.1682	0.1427	0.1309	0.1901	**0.2136**	22.76%	17.69%
	NDCG@20	0.1937	0.1975	0.1147	0.0876	0.1875	0.1581	0.1415	0.2081	**0.2314**	19.46%	17.16%
Sport	HR@5	0.1918	0.1963	0.0978	0.0738	0.1716	0.1321	0.1529	0.2052	**0.2226**	16.06%	13.40%
	HR@10	0.2753	0.2684	0.1543	0.1250	0.2457	0.1964	0.2187	0.2798	**0.3026**	9.92%	12.74%
	HR@20	0.3727	0.3550	0.2336	0.2048	0.3342	0.2797	0.3080	0.3687	**0.3964**	6.36%	11.66%
	NDCG@5	0.1302	0.1393	0.0651	0.0481	0.1166	0.0907	0.1064	0.1457	**0.1573**	20.81%	12.92%
	NDCG@10	0.1571	0.1626	0.0832	0.0645	0.1405	0.1111	0.1276	0.1697	**0.1831**	16.55%	12.61%
	NDCG@20	0.1816	0.1844	0.1032	0.0846	0.1627	0.1321	0.1501	0.1921	**0.2067**	13.82%	12.09%
Yelp	HR@5	0.2826	0.3180	0.1463	0.1518	0.2938	0.2204	0.2511	0.3159	**0.3485**	23.32%	9.59%
	HR@10	0.4252	0.4421	0.2589	0.2557	0.4225	0.3298	0.3855	0.4416	**0.4807**	13.05%	8.73%
	HR@20	0.5991	0.5941	0.4345	0.4202	0.5815	0.4620	0.5648	0.5967	**0.6342**	5.86%	6.75%
	NDCG@5	0.1870	0.2257	0.0906	0.0954	0.2020	0.1479	0.1680	0.2212	**0.2459**	31.50%	8.95%
	NDCG@10	0.2329	0.2657	0.1268	0.1287	0.2434	0.1831	0.2113	0.2618	**0.2886**	23.92%	8.62%
	NDCG@20	0.2767	0.3039	0.1708	0.1700	0.2835	0.2165	0.2565	0.3009	**0.3273**	18.29%	7.70%
WeChat	HR@5	0.2763	0.3067	0.1825	0.1924	0.3162	0.2981	0.2868	0.3103	**0.3416**	23.63%	11.38%
	HR@10	0.4113	0.4356	0.2238	0.2261	0.4401	0.4165	0.3993	0.4504	**0.4854**	18.02%	11.43%
	HR@20	0.5274	0.5481	0.2897	0.2926	0.5538	0.5180	0.4871	0.5511	**0.5981**	13.41%	9.12%
	NDCG@5	0.1961	0.2152	0.1287	0.1252	0.2060	0.2066	0.1947	0.2207	**0.2373**	21.01%	10.27%
	NDCG@10	0.2347	0.2762	0.1472	0.1416	0.2583	0.2583	0.2379	0.2817	**0.3070**	30.81%	11.15%
	NDCG@20	0.2882	0.3030	0.1683	0.1547	0.2977	0.2806	0.2659	0.3083	**0.3335**	15.72%	10.07%

Table 10: The third run of baselines and our proposed CT4Rec on four offline datasets, where '*' refers to modifying the original binary cross-entropy loss in SASRec with the training objective in recent baselines, e.g., CLRec, CL4SRec, StackRec. Our CT4Rec is implemented on SASRec*. *Improv.* and *Improv.** refer to the relative improvement of CT4Rec over SASRec and SASRec*, respectively. Except for the random seed, the other settings are the same as Table 2.

Datasets	Metric	SASRec	SASRec*	GRU4Rec	BERT4Rec	TiSASRec	StackRec	CLRec	CL4SRec	CT4Rec	Improv.	Improv.*
Beauty	HR@5	0.2087	0.2198	0.1149	0.0840	0.2062	0.1697	0.1528	0.2240	**0.2576**	23.43%	17.20%
	HR@10	0.2758	0.2740	0.1574	0.1319	0.2768	0.2157	0.1853	0.2842	**0.3208**	16.32%	17.08%
	HR@20	0.3517	0.3360	0.2157	0.2025	0.3522	0.2764	0.2255	0.3558	**0.3911**	11.20%	16.40%
	NDCG@5	0.1482	0.1674	0.0851	0.0550	0.1441	0.1272	0.1198	0.1679	**0.1933**	30.43%	15.47%
	NDCG@10	0.1699	0.1849	0.0987	0.0704	0.1668	0.1420	0.1303	0.1873	**0.2138**	25.84%	15.63%
	NDCG@20	0.1890	0.2005	0.1133	0.0880	0.1859	0.1573	0.1404	0.2054	**0.2315**	22.49%	15.46%
Sport	HR@5	0.1908	0.1968	0.0997	0.0781	0.1721	0.1334	0.1543	0.2082	**0.2198**	15.20%	11.69%
	HR@10	0.2734	0.2680	0.1558	0.1294	0.2447	0.1979	0.2215	0.2832	**0.2975**	8.81%	11.01%
	HR@20	0.3733	0.3546	0.2356	0.2090	0.3328	0.2802	0.3106	0.3731	**0.3915**	4.88%	10.41%
	NDCG@5	0.1308	0.1409	0.0656	0.0502	0.1169	0.0908	0.1077	0.1481	**0.1556**	18.96%	10.43%
	NDCG@10	0.1574	0.1638	0.0836	0.0666	0.1403	0.1115	0.1293	0.1723	**0.1807**	14.80%	10.32%
	NDCG@20	0.1825	0.1856	0.1037	0.0866	0.1625	0.1322	0.1517	0.1949	**0.2044**	12.00%	10.13%
Yelp	HR@5	0.2887	0.3194	0.1460	0.1522	0.2976	0.2202	0.2573	0.3214	**0.3475**	20.37%	8.80%
	HR@10	0.4313	0.4471	0.2570	0.2559	0.4294	0.3295	0.3911	0.4489	**0.4803**	11.36%	7.43%
	HR@20	0.6023	0.6011	0.4335	0.4227	0.5857	0.4628	0.5649	0.6013	**0.6326**	5.03%	5.24%
	NDCG@5	0.1922	0.2264	0.0903	0.0962	0.2045	0.1476	0.1725	0.2250	**0.2458**	27.89%	8.57%
	NDCG@10	0.2381	0.2675	0.1259	0.1294	0.2470	0.1828	0.2155	0.2661	**0.2886**	21.21%	7.89%
	NDCG@20	0.2812	0.3064	0.1702	0.1712	0.2865	0.2165	0.2592	0.3045	**0.3270**	16.29%	6.72%
WeChat	HR@5	0.2726	0.3073	0.1842	0.1937	0.3214	0.2958	0.2864	0.3112	**0.3409**	25.06%	10.93%
	HR@10	0.4091	0.4361	0.2261	0.2253	0.4417	0.4181	0.4003	0.4517	**0.4852**	18.60%	11.26%
	HR@20	0.5280	0.5472	0.2891	0.2926	0.5554	0.5208	0.4868	0.5502	**0.5965**	12.97%	9.01%
	NDCG@5	0.1932	0.2143	0.1277	0.1271	0.2051	0.2061	0.1953	0.2175	**0.2353**	21.79%	9.80%
	NDCG@10	0.2352	0.2747	0.1457	0.1422	0.2631	0.2562	0.2362	0.2819	**0.3067**	30.40%	11.65%
	NDCG@20	0.2834	0.3007	0.1706	0.1542	0.2973	0.2803	0.2647	0.3077	**0.3301**	16.48%	9.78%