A NON-CONTRASTIVE LEARNING FRAMEWORK FOR SEQUENTIAL RECOMMENDATION WITH PREFERENCE-PRESERVING PROFILE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Contrastive Learning (CL) proves to be effective for learning generalizable user representations in Sequential Recommendation (SR), but it suffers from high computational costs due to its reliance on negative samples. To overcome this limitation, we propose the first Non-Contrastive Learning (NCL) framework for SR, which eliminates computational overhead of identifying and generating negative samples. However, without negative samples, it is challenging to learn uniform representations from only positive samples, which is prone to representation collapse. Furthermore, the alignment of the learned representations may be substantially compromised because existing ad-hoc augmentations can produce positive samples that have inconsistent user preferences. To tackle these challenges, we design a novel preference-preserving profile generation method to produce high-quality positive samples for non-contrastive training. Inspired by differential privacy, our approach creates augmented user profiles that exhibit high diversity while provably retaining consistent user preferences. With larger diversity and consistency of the positive samples, our NCL framework significantly enhances the alignment and uniformity of the learned representations, which contributes to better generalization. The experimental results on various benchmark datasets and model architectures demonstrate the effectiveness of the proposed method. Finally, our investigations reveal that both uniformity and alignment play a vital role in improving generalization for SR. Interestingly, in our data-sparse setting, alignment is usually more important than uniformity.

1 INTRODUCTION

Contrastive Learning (CL) proves to be an effective approach to learn generalizable user representations for sequential recommendation (SR) (Wang et al., 2022b; Liu et al., 2021; Xie et al., 2022). A canonical CL-based recommendation framework usually employs positive samples (e.g., augmented user profiles or items) to promote representation alignment and negative samples (e.g., profiles of other users or irrelevant items) for representation uniformity (Wang & Isola, 2020; Zhuo et al., 2023). With enhanced alignment and uniformity, CL-based methods achieve state-of-the-art performance.

However, CL-based methods inevitably suffer from high computational costs, because they heavily rely on negative samples to avoid representation collapse during training (Grill et al., 2020; Chen & He, 2021). For example, CL-based methods either require large batch sizes during training or use sophisticated retrieval strategies to identify hard negative samples (Robinson et al., 2020), leading to challenges on hardware memories and computational efficiency (Kulatilleke et al., 2023).

In a sharp comparison, Non-Contrastive Learning (NCL) emerges as a promising solution that avoids using negative samples (Zhuo et al., 2023). Unlike CL using negative samples for uniformity, NCL is capable of learning uniform representations using only positive samples. Therefore, in this work, we aim to learn uniformity using only positive samples for SR. The rationale behind NCL is: such uniformity can best represent the information of individual data points, because a probability distribution that best represents the current state of knowledge about a system is the one with the largest entropy according to information theory (Jaynes, 1957; Guiasu & Shenitzer, 1985). Nevertheless, despite the success of NCL for Computer Vision (Grill et al., 2020; Zbontar et al., 2021) and Natural

Language Processing (Cho et al., 2022; Zhou et al., 2023), NCL for recommendation is unexplored. Furthermore, learning generalizable user representations from recommendation data through NCL is much more complex and difficult than learning representations for images and texts.

The key challenge stems from obtaining high-quality positive user profile pairs to enable non-contrastive training. On one hand, within an NCL framework, the absence of negative samples makes it more challenging for the recommender to learn uniform representations from the sparse recommendation data than from dense CV or NLP data. This is because the sparsity of the recommendation data increases the risk of representation collapse Guo et al. (2023). In recommendation, the collapsed user representations are mapped nearby in the latent space, even if they represent drastically different user preferences. As a consequence, such collapsed representation lacks expressiveness to capture unique preferences of different users, making it difficult for the recommender to generate meaningful recommendations. On the other hand, the representation alignment may be substantially compromised as well, due to the inconsistent user preferences encoded in the positive samples. We note that existing operations are too ad-hoc to generate consistent positive user profile pairs. For instance, data augmentations are primary methods to generate positive samples, but they are usually random (e.g., random crop, random swap, or random mask (Xie et al., 2022)) or heuristic (correlation-based substitution or insertion (Liu et al., 2021)). Obviously, such ad-hoc operations may fail to preserve the genuine user preferences in the positive samples. These inconsistencies confuse the model, when the model is trained to align profiles that reflect different, even conflicting preferences for the same user. As a consequence, the representation alignment is substantially compromised, and the model fails to capture the true user preferences, degrading the performance.

Therefore, targeting at the limitations of CL-based SR methods and NCL methods, we propose the first **N**on-**C**ontrastive **L**earning framework for **S**equential **R**ecommendation powered by preference-preserving user profile generation: **NCL-SR**. The proposed NCL-SR eliminates the computational overhead of identifying and generating negative samples in CL. Furthermore, we design a novel preference-preserving profile generation to address the representation collapse issue and preference inconsistency within the training data. Inspired by Differential Privacy (DP), our approach creates augmented user profiles that retain consistent user preferences with provable guarantees. To address representation collapse, our approach generates highly diversified positive samples with controllable randomness. The diversity of positive samples helps the model explore distinct and larger portions of the representation space, which effectively reduces the risk of representation collapse. In addition, the preference-preserving design of our approach also addresses the inconsistent preference issue within the positive samples by preserving the genuine user preferences within the generated positive samples. Finally, note that the DP augmentation serves as the key pre-requisite to promote both alignment and uniformity under our NCL framework. With the generated augmented user profiles, the proposed framework computes a non-contrastive alignment loss and a non-contrastive uniformity loss from matrix information theory (Zhang et al., 2023), to improve the generalization of the learned user representations. Overall, our contribution can be summarized as follows:

- We design and present the very first self-supervised non-contrastive learning framework for sequential recommendation: NCL-SR. NCL-SR learns generalizable and robust user representations for sequential recommendation.
- We present a novel data augmentation operation with theoretical guarantees to enable non-contrastive training. Unlike existing ad-hoc augmentation operations, our approach is guaranteed to preserve user preferences in the augmented user profiles, addressing the representation collapse issue and preference inconsistency issue within the training data.
- We conduct extensive experiments encompassing diverse datasets. Our experimental results suggest NCL-SR consistently exhibits superior performance over state-of-the-art SR models and CL-based SR methods. Our investigations reveal that uniformity and alignment of the representations play a vital role in SR. Interestingly, we observe that alignment may be more important than uniformity.

2 Related Work

Self-Supervised Learning for Sequential Recommendation. There are three types of self-supervised learning schemes for sequential recommendation: generative, adversarial and contrastive

(Ren et al., 2024). Generative SR models primarily focus on generating sequence item data as recommendations (Ye et al., 2023; Wu et al., 2023; Wang et al., 2022d). An exemplar generative SR model is BERT4Rec (Sun et al., 2019), where the recommender is trained to generate and reconstruct the masked items within user histories. In comparison, adversarial SR models (Ni et al., 2023; Lv et al., 2021) are mainly GAN(Goodfellow et al., 2020)-like in the sense that they usually train discriminators that distinguishes between real and generated item sequences, while optimizing generators to produce realistic recommendations. Compared to generative and adversarial SR models, contrastive SR models comprises the majority of self-supervised SR methods and are state-of-the-art. Existing CL-based SR methods focus on designing sophisticated rules and augmentation operations to construct contrastive pairs. For instance, CLS4Rec (Xie et al., 2022) adopts three randomized operations to generate augmented views of user history. Based on CLS4Rec, CoSeRec (Liu et al., 2021) further proposes two informative augmentations: correlation-based item substitution and insertion. Liu et al. (2021) present EC4Rec, an explanation guided CL framework that identifies positive/negative items using training gradients. In comparison, DUORec (Qiu et al., 2022) showcases a supervised contrastive learning framework with model-level dropout augmentations to further improve representation learning. In (Zheng et al., 2022), aiming at the Cross-Domain Sequential Recommendation challenge, the authors propose a novel model via dual dynamic graph modeling and hybrid metric training with contrastive learning integration. In (Su et al., 2023), a novel hierarchy-aware dual clustering graph network (HADCG) model is designed to explore the inherent hierarchy structures from both item popularity and collaborations. With information regularizer for intra-session clustering and contrastive training for inter-session clustering, HADCG achieves substantial performance improvements in session-based recommendation. Finally, Shi et al. (2024) present self contrastive learning (SCL) to enhance the uniformity of the learned item representations. However, CL-based methods usually poses challenges on hardware memories and computational efficiency, because they either require large batch sizes or sophisticated retrieval strategies to identify hard negatives (Robinson et al., 2020). Furthermore, augmentation operations of CL-based methods may also introduce preference inconsistency in the positive samples, because they may fail to preserve user preferences. This makes CL-based methods sensitive to data augmentation.

Non-Contrastive Self-supervised Learning. Recently, there is an increasing interest in Non-Contrastive Learning (NCL) for its capability of learning robust representations without negative samples. Unlike CL pursing both alignment and uniformity in representation learning, NCL tends to focus more on learning uniform data representations. For instance, BYOL (Grill et al., 2020) and SimSiam (Chen & He, 2021) introduce asymmetry in the network architecture and parameter updates to avoid representation collapse. Barlow Twins (Zbontar et al., 2021) enhances uniformity by making the cross-correlation matrix of two twin representations as close to the identity matrix as possible. Liu et al. (2022) present a theoretical framework based on the maximum entropy encoding principle from information theory, encompassing all previous NCL losses into a generalized uniformity loss. Finally, harmonizing alignment and uniformity, Zhang et al. (2023) propose matrix information theory, and design a matrix-theoretical self-supervised framework. Unfortunately, NCL demonstrates effectiveness only for CV and NLP tasks where the data is dense and the training signals are based on explicit feedback, while its effectiveness for SR is unexplored. Furthermore, the sparsity of the recommendation data leads to higher risk of representation collapse for NCL.

3 PRELIMINARIES

Data. We focus on sequential recommendation (SR). In an SR dataset $\mathcal{D}$, a data sample is a sequence of interacted items x (sorted by timestamps) from a user's history. We denote a list of interacted items as $x = [x_1, x_2, ..., x_l]$. For x, its ground-truth label is the item at the next time step: $y = x_{l+1}$. Each element in x and y belong to the item scope $\mathcal{I}$: $x_i \in \mathcal{I}$.

Model. Classic sequential recommenders are ID-based models that represent items with discrete item IDs. They take sequences of item IDs as inputs, and predict new item IDs as recommendations. Given x, an ID-based recommender computes scores for all items from $\mathcal{I}$, and selects items with highest scores for recommendations. In comparison, text-based recommenders represent items with textual descriptions (e.g., title, category, reviews), and generate recommended items with free-form texts (e.g., item titles). In this work, we use a text-based model to build NCL-SR for the generality

of textual features. For clarity and consistency, we also use $x = [x_1, x_2, ..., x_l]$ to denote the input texts of a text-based recommender, where each element x_i is the textual description of an item.

Contrastive Learning. Existing CL-based recommendation methods usually employ the following contrastive loss to train recommendation models (Xie et al., 2022; Zhang et al., 2024; Shi et al., 2024). Formally, for a batch of training data X, the contrastive loss is computed with:

$$\mathcal{L}_{\text{CL}} = \frac{1}{|X|} \sum_{x^{(a)} \sim X} -\log \frac{s(x^{(a)}, x^{(p)})}{s(x^{(a)}, x^{(p)}) + \sum_{k=1}^{N} s(x^{(a)}, x_k^{(n)})}, \tag{1}$$

where $x^{(a)}$ denotes an anchor training sample within the batch, $x^{(p)}$ denotes a positive sample of $x^{(a)}$, and $x^{(n)}$ denotes a negative sample of $x^{(a)}$ (assume N negative samples for $x^{(a)}$). s represents an arbitrary similarity scoring function (e.g., cosine similarity) in a canonical form. However, CL suffers from high computational costs due to its reliance on negative samples and is sensitive to data augmentation. The above limitations of CL motivate our NCL framework.

4 METHODOLOGY

In this work, we propose a novel NCL-SR framework via preference-preserving profile generation to promote uniformity and alignment in user representations.

Uniformity ensures that that individual user preferences are preserved as much as possible in the user representations. However, within an NCL framework, the absence of negative samples makes it challenging to learn uniformity from sparse recommendation data. This is because such sparsity increases the risk of representation collapse (Guo et al., 2023) , where different user preferences are mapped nearby in the latent space. Consequently, it becomes difficult for the recommender to capture unique preferences of different users and generate meaningful recommendations. *Alignment* enforces that two users with similar preferences should be mapped to nearby representations, which makes representations robust against undesired noise factors. Nevertheless, the inconsistency within the training data hinders learning such alignment (e.g., implicit feedback, inconsistent user preferences in the augmented data). These inconsistencies confuse the model, when the model is trained to align profiles that reflect different, conflicting preferences for the same user.

Therefore, we design the preference-preserving profile generation to generate high-quality positive user profiles to facilitate non-contrastive training. To address representation collapse, our approach generates diversified positive samples. This helps the model explore distinct portions of the representation space, reducing the risk of representation collapse. In addition, with the reference-preserved positive samples the data inconsistency issue is effectively addressed.

4.1 PREFERENCE-PRESERVING PROFILE AUGMENTATION VIA DIFFERENTIAL PRIVACY

To build the NCL framework for SR, we first propose the preference-preserving user profile generation. Inspired by Differential Privacy (DP) (Dwork et al., 2006), our approach creates augmented user profiles that retain original user preferences with provable guarantees. The preference-preserved augmentation generates diverse and consistent positive samples to overcome the representation collapse and inconsistency within the training data, contributing to better alignment and uniformity of the learned user representations. For a better understanding of our preference-preserving profile generation, we begin by formally defining preference-preserving augmentation. We provide further insights about Differential Privacy (DP) in Appendix A.

Definition 1 (Preference-Preserving Augmentation) *For a user profile x, its augmented view $x^{'}$ is considered preference-preserving if $x^{'}$ leads to the same expected purchase, or the expected top-1-ranked item remains the same, when x' is fed into the same recommendation mechanism $\mathcal{M}$.*

With the definition of preference-preserving augmentation, we then present how DP guarantees preference-preservation in Theorem 1.

Theorem 1 (Guaranteed Preference-Preservation with Differential Privacy) *Suppose a recommendation mechanism $\mathcal{M}$ satisfies differential privacy with the privacy parameter ϵ (i.e., ϵ-DP). For*

any user sequence x, if $\mathcal{M}$ successfully recommends the ground-truth item y, which is the j-th item within the item scope $\mathcal{I}$: $y := x_j$, and the predictive score for the ground-truth item is greater than the second-largest runner-up score of another item with a small multiplicative factor $e^{2\epsilon}$:

$$\mathbb{E}(\mathcal{M}(x)_j) > e^{2\epsilon} \max_{k:k\neq j} \mathbb{E}(\mathcal{M}(x)_k), \tag{2}$$

then for its augmented view x', its predicted top-1 item remains the same if there is only limited modification from x to $x^{'}$:

$$\mathbb{E}(\mathcal{M}(x^{'})_j) > \max_{k:k\neq j} \mathbb{E}(\mathcal{M}(x^{'})_k). \tag{3}$$

Note that Theorem 1 is a specific adaptation of a general DP property for our SR problem. The proof is adapted from (Wang et al., 2021), which we relegate to Appendix A. Since the recommendation data considered in this work is discrete (i.e., item texts), we use the DP exponential mechanism to build our DP augmentation pipeline, which is formally defined below.

Definition 2 (Exponential Mechanism) *The exponential mechanism $\mathcal{M}_E(u, \Delta_u, \hat{P})$ is characterized by three components: a scoring function u, the sensitivity Δ_u and a sampling distribution $\hat{P}$. The scoring function $u(X, X_c)$ computes scores for each pair of inputs (X) and a candidate (X_c). The sensitivity Δ_u is defined as $\Delta_u := \max_{X_c} \max_X |u(X, X_c) - u(X^{'}, X_c)|$. The exponential mechanism $\mathcal{M}_E(u, \Delta_u, \hat{P})$ is $\epsilon-$differentially private if it outputs the candidate X_c with probability $\hat{P}$ proportional to $e^{\frac{\epsilon u(X, X_c)}{2\Delta_u}}$.*

User Profile Augmentation via Exponential Mechanism. When implementing the exponential mechanism for NCL-SR, we first craft a set of synonym dictionary for all items. The synonym items will be used to perturb the original user history following the exponential mechanism. For a specific item text $x_i \in \mathcal{I}$, its synonym items are other items with top-k highest cosine similarity in the representation space. Formally, the synonym set $\mathcal{S}(x_i)$ of an item x_i with top-k similarity is computed with:

$$\mathcal{N}_k(x_i) = \arg \max_{\mathcal{N}\subset\mathcal{I}, |\mathcal{N}|=k, x_i\notin\mathcal{N}} \sum_{x_j\in\mathcal{N}} \text{CosSim}(f(x_i), f(x_j)), \tag{4}$$

where f represents a text-based recommender model that encodes user/item texts into latent representations. After replacing randomly selected items within $x = [x_1, x_2, ..., x_l]$ with their synonyms, we obtain a set of candidate user profiles $X^{'} = \{x^{'}\}$. In terms of the exponential mechanism, the scoring function u is then defined as the cosine similarity between embedded user profile x and any other candidate user profiles x':

$$u(x, x^{'}) = e^{\text{CosSim}(f(x), f(x^{'}))} \tag{5}$$

Note that the candidate user profiles x' are perturbed compared to the original user profile x, and the perturbations can be multiple on different positions. With the definition of Equation 5, the sensitivity of the utility score $u(x, x^{'})$ is $\Delta_u = e - 1/e$, with e being maximum of $u(x, x^{'})$ and $1/e$ being the minimum. Finally, the exponential mechanism selects and outputs $x^{'}$ with sampling probability $e^{\frac{\epsilon u(t, t^{'})}{2\Delta_u}}$. For computational efficiency, the exponential mechanism returns the expected **latent representation** of $X^{'}$ by computing the weighted sum of $x^{'}$, with the importance weight for each $x^{'}$ being the normalized sampling probability:

$$z_{\hat{X'}} = \sum_{x^{'}\in X^{'}} \hat{P_{x'}} \cdot f(x^{'}), \quad \text{where} \quad \hat{P_{x'}} = \frac{e^{\frac{\epsilon u(x, x^{'})}{2\Delta_u}}}{\sum_{x''\in X'} e^{\frac{\epsilon u(x, x^{''})}{2\Delta_u}}}. \tag{6}$$

To this end, we obtain the augmented user profile for a training user. The DP user profiles effectively preserves the original user preferences and serves as the foundation for the NCL framework.

Note that a critical caveat here is that the search space grows exponentially w.r.t. the length of user histories if the augmentation is defined at user-level: $\mathcal{O}(k^l)$. This is because each item

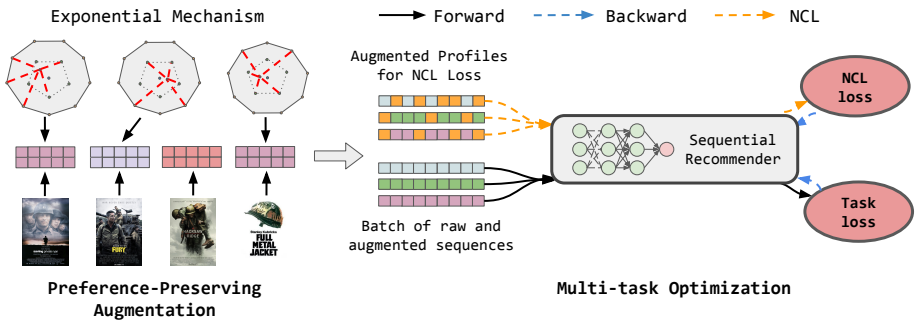


Figure 1: The overview of the proposed NCL-SR. NCL-SR first performs preference-preserving augmentation to generate high-quality positives. Then, NCL-SR optimizes a combined training loss of NCL losses and the main task loss (e.g., cross-entropy or other ranking loss).

$x_i \in x = [x_1, x_2, ..., x_l]$ has k replacements. As such, there are k^l possible candidate user profiles for exponential mechanism to sample from, i.e., $|X'| = k^l$. Given the sparsity of recommendation data, computing DP augmentations for user sequences introduces prohibitive computation complexity and noises. Therefore, in our experiments, we re-define Equation 5 and Equation 6 at item-level, successfully reducing the complexity to be linear: $\mathcal{O}(k \cdot l)$. To see this, we highlight that when applying the exponential mechanism to each item $x_i \in x = [x_1, x_2, ..., x_l]$, the exponential mechanism just to need to sample from k candidates, and repeat for all items within the history l times, leading to $\mathcal{O}(k \cdot l)$. More details (guarantees, proofs and implementation) about the efficient design is documented in Appendix B.

An illustrative example of DP profile generation is shown in Figure 1. In Figure 1, each polygon represents the synonym set for an item. The vertices within a polygon represent the embeddings of the synonym items w.r.t. the original item. For instance, for the first item (i.e., the movie "Save the Private Ryan ") of the user's history, our method first encodes this item into an embedding vector. Then, we compute the synonym set for this item with Equation 4. Therefore, all vertices in the polygon represent the embedded items. Note that there is no corresponding polygon for the third item in Figure 1, because our approach only perturbs a part of the user profiles, which reduces computation costs of the exponential mechanism.

Note this efficient design is feasible and remain DP, because of the two key properties of DP. First, DP has the post-processing property: any computation applied to the output of a DP algorithm remains DP. As such, if the augmentation operation is DP, then recommendation process based on the augmented user profile is also DP. Second, the expected output stability property of DP reduces the computational costs of DP augmentation. This property allows us to generate an expected augmented profile for each user to preserve user preferences in the augmented profiles, instead of generating massive augmented samples for Monte-Carlo sampling.

4.2 Overview of NCL-SR

With the proposed preference-preserving profile generation, we then introduce the overall NCL-SR. As shown in Figure 1, our proposed NCL-SR consists of 3 components.

The first component is a text-based recommender f that encodes user/item texts and returns relevance scores over candidate items. For any user sequence x, the recommender f encodes user/item texts into representations and then returns a probability distribution over the item scope $\mathcal{I}$:

$$P = P(f, x, \mathcal{I}) = \text{Softmax}\left(\left[\text{CosSim}(q, p_1), \text{CosSim}(q, p_2), ..., \text{CosSim}(q, p_{|\mathcal{I}|})\right]\right), \quad (7)$$

where $q = f_T(x)$ is the embedded user profile, and $p_j = f_T(x_j), x_j \in \mathcal{I}$ is the embedded description of an item x_j. The items associated with highest probabilities are retrieved as recommendations.

The second component is multi-task optimization, where we optimize the model w.r.t. the ranking task, alignment learning task and uniformity learning task. The multi-task optimization benefits recommendation performance, because both the recommendation task and non-contrastive training are

modeling user/item relationships in the representation space. To design a general NCL framework for sequential recommendation, we adopt Matrix Cross Entropy (MCE) (Zhang et al., 2023), which generalizes multiple non-contrastive losses including SimSiam (Chen & He, 2021), Barlow Twin (Zbontar et al., 2021), MEC (Liu et al., 2022), etc. Formally, the matrix cross entropy between two matrices U and V is defined as

$$\mathrm{MCE}(U, V) = \mathrm{tr}(-U \log V + V), \tag{8}$$

where $\log$ denotes the principal matrix logarithm (Higham, 2008) and is approximated via Taylor expansion. Based on the definition of matrix cross entropy, one may promote uniformity in representation learning by aligning the covariance matrix of representations with the identity matrix:

$$\mathcal{L}_{\text{uniform}}(Z, Z^{'}) = \mathrm{MCE}\big(\frac{1}{d} I_d, C(Z, Z^{'})\big), \tag{9}$$

where $Z = f(X)$ and $Z^{'} = f(X^{'})$ denote encoded, d-dimensional, $l2$-normalized representations of batched user profiles and their preference-preserving augmentation, respectively. $C(Z, Z^{'})$ represents the centered covariance matrix of Z and $Z^{'}$: $C(Z, Z^{'}) = \frac{1}{B} Z H_B Z'^{T}, H_B = I_B - \frac{1}{B} 1_B 1_B^T$. Intuitively, minimizing $\mathcal{L}_{\text{uniform}}(Z, Z^{'})$ promotes the uniformity in representation learning, because the minimization makes the covariance matrix of the embedded (augmented) user representations to be close to the identity matrix, which increases the entropy of the encoded representations.

Note that Equation 9 as well as other existing non-contrastive methods only promotes uniformity in representation learning. However, it has been widely shown that learning aligned user representations also play a vital role in improving model generalization for sequential recommendation. This is because aligned representations are more robust against the noises caused by data inconsistency (Wang et al., 2022b; Xie et al., 2022; Qiu et al., 2022). Therefore, we propose to integrate another alignment loss to train the model w.r.t. the alignment learning loss:

$$\mathcal{L}_{\text{align}}(Z, Z^{'}) = -\mathrm{tr}(C(Z, Z^{'})) + \gamma \cdot \mathrm{MCE}\big(C(Z, Z), C(Z^{'}, Z^{'})\big). \tag{10}$$

Intuitively, since Z and $Z^{'}$ are derived from the same group of users as in Equation 9, minimizing $\mathcal{L}_{\text{align}}(Z, Z^{'})$ enforces the similarity of the learned representations from the same user. To this end, the multi-task optimization is a combination of losses of three tasks:

$$\mathcal{L}_{\text{NCL-SR}} = \mathcal{L}_{\text{task}} + \lambda_1 \cdot \mathcal{L}_{\text{uniform}} + \lambda_2 \cdot \mathcal{L}_{\text{align}}. \tag{11}$$

The third component of NCL-SR is the preference-preserving profile generation introduced the previous section. We highlight that both Equation 9 and Equation 10 require the augmented views of original user profiles. In other words, generating high-quality augmented data is the prerequisite for non-contrastive training. This component can effectively preserve the user preferences in the diverse, augmented user profiles, which overcomes the representation collapse issue and the inconsistency within the training data.

5 Experiments

5.1 Experimental Setup

Datasets. We adopt 6 public benchmark recommendation datasets to evaluate NCL-SR: Beauty, Games, Sports, Toys, Office and Auto (He & McAuley, 2016; McAuley et al., 2015)[1]. These datasets cover different application domains, and are characterized with different sparsity and sequence lengths. Following Yue et al. (2022), we use the 5-core processing to filter out infrequent items and users. To simulate the data sparsity and cold-start users, the split ratio of the training, validation, and test sets is 2:2:6. Such a split ratio is selected because we aim to explore the model's performance under a limited quantity of training data and the model's generalization on cold-start users, following the recent studies Wu et al. (2024); Qian et al. (2020); Wang et al. (2022a); Lin et al. (2025). More details about datasets statistics are in summarized Appendix C[2].

[1]The datasets are available at http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/.

[2]We submit the code in the supplemental material, and will release the code upon acceptance

Dataset	Metric	ID-Based				Text-Based				Improv.
		BERT4Rec	SASRec	NARM	LRURec	P5	UniSRec	RecFmr.	NCL-SR	(%)
Beauty	R@10 ↑	0.0429	0.0599	0.0493	0.0715	0.0227	0.0437	<u>0.0729</u>	**0.0791**	8.50%
	N@10 ↑	0.0232	0.0362	0.0281	<u>0.0439</u>	0.0133	0.0249	0.0407	**0.0440**	0.23%
	R@20 ↑	0.0617	0.0812	0.0719	0.0931	0.0282	0.0656	<u>0.1035</u>	**0.1135**	9.66%
	N@20 ↑	0.0279	0.0415	0.0338	<u>0.0493</u>	0.0147	0.0304	0.0484	**0.0526**	6.69%
Games	R@10 ↑	0.0564	0.0934	0.0661	<u>0.1087</u>	0.0120	0.0691	0.0867	**0.1140**	4.88%
	N@10 ↑	0.0294	0.0490	0.0350	<u>0.0606</u>	0.0063	0.0365	0.0465	**0.0611**	0.83%
	R@20 ↑	0.0932	0.1323	0.1036	<u>0.1540</u>	0.0196	0.1024	0.1339	**0.1683**	9.29%
	N@20 ↑	0.0386	0.0587	0.0444	<u>0.0720</u>	0.0083	0.0449	0.0583	**0.0748**	3.89%
Sports	R@10 ↑	0.0195	0.0259	0.0272	0.0327	0.0057	0.0203	<u>0.0331</u>	**0.0441**	33.2%
	N@10 ↑	0.0111	0.0154	0.0147	<u>0.0201</u>	0.0028	0.0115	0.0172	**0.0237**	17.9%
	R@20 ↑	0.0291	0.0330	0.0389	0.0464	0.0085	0.0317	<u>0.0518</u>	**0.0660**	27.4%
	N@20 ↑	0.0135	0.0172	0.0177	<u>0.0235</u>	0.0036	0.0143	0.0218	**0.0292**	24.3%
Toys	R@10 ↑	0.0278	0.0379	0.0299	0.0655	0.0253	0.0382	<u>0.0845</u>	**0.0941**	11.4%
	N@10 ↑	0.0166	0.0226	0.0173	0.0420	0.0139	0.0221	<u>0.0475</u>	**0.0537**	13.1%
	R@20 ↑	0.0379	0.0507	0.0444	0.0858	0.0319	0.0548	<u>0.1164</u>	**0.1286**	10.5%
	N@20 ↑	0.0192	0.0258	0.0210	0.0471	0.0156	0.0262	<u>0.0555</u>	**0.0623**	12.3%
Office	R@10 ↑	0.0453	0.0524	0.0654	<u>0.0926</u>	0.0528	0.0629	0.0645	**0.1047**	13.1%
	N@10 ↑	0.0231	0.0325	0.0333	<u>0.0496</u>	0.0255	0.0300	0.0350	**0.0534**	7.66%
	R@20 ↑	0.0782	0.0996	0.1155	<u>0.1462</u>	0.0915	0.1018	0.1025	**0.1625**	11.2%
	N@20 ↑	0.0314	0.0445	0.0459	<u>0.0631</u>	0.0351	0.0398	0.0445	**0.0681**	7.92%
Auto	R@10 ↑	0.0415	0.0439	0.0585	<u>0.1220</u>	0.0431	0.0646	0.1085	**0.1354**	11.0%
	N@10 ↑	0.0202	0.0216	0.0287	<u>0.0632</u>	0.0216	0.0355	0.0571	**0.0714**	13.0%
	R@20 ↑	0.0732	0.0780	0.1024	<u>0.1829</u>	0.0866	0.1098	0.1768	**0.2085**	14.0%
	N@20 ↑	0.0282	0.0304	0.0396	<u>0.0785</u>	0.0325	0.0469	0.0745	**0.0899**	14.5%

Table 1: Main results on recommendation performance of different SR models. The best results are highlighted in bold and the second best results are highlighted with underline.

Baselines and Implementation. There are two sets of baselines in our experiments. The first group of baselines are state-of-the-art SR models, including ID-based models (i.e., NARM (Li et al., 2017), LRURec (Yue et al., 2023), BERT4Rec (Sun et al., 2019), SASRec (Kang & McAuley, 2018)) and text-based SR model: P5 (Geng et al., 2022), RecFmr (Li et al., 2023) and UniSR$_{\mathrm{T}}$ (Hou et al., 2022). The second group baselines are state-of-the-art CL-based recommendation methods, including CLS4Rec (Xie et al., 2022), CoSeRec (Liu et al., 2021), EC4Rec (Liu et al., 2021), DUORec (Qiu et al., 2022) and SCL (Shi et al., 2024). Note that some of the CL-based baselines are developed based on item IDs, and we modify such baselines into the text-based setting for a fair comparison. We use E5 (`e5-base-v2`) (Wang et al., 2022c) to implement NCL-SR and CL-based baselines. When generating DP augmentations, we randomly perturb 3 items for each training user from datasets. For evaluation, we use normalized discounted cumulative gain (NDCG@N) and recall (Recall@N) with $N \in [10, 20]$. The predictions are ranked against all items in the dataset.

5.2 Comparing against SR models

The first set of experiments is conducted to compare NCL-SR against state-of-the-art sequential recommenders. The performance results are reported in Table 1. The last column of Table 1 is relative improvement NCL-SR compared to the best-performing baseline method (i.e., Improv.). We observe: (1) NCL-SR consistently outperforms baseline methods across all metrics and datasets, with an average performance improvement of 11.93% compared to the second best method. Such an observation demonstrates the generality of NCL-SR, regardless of data domains. (2) The performance gains of NCL-SR are more pronounced on the sparsest dataset (i.e., Sports), where NCL-SR can go up to 33.2% on Recall@10. This demonstrates the substantial benefits of applying NCL-SR in sparse data domains. (3) NCL-SR generally demonstrates better retrieval performance (i.e., Recall) than ranking performance (i.e., NDCG). For instance, there is a significant increase of 16.7% in the average Recall@10 scores with NCL-SR, while the relative improvement on NDCG@10 is slightly lower (8.78%). This is expected, as our text-based model has a strong ability to understand and match the semantic meanings of item descriptions, which provides advantages for the retrieval task. We also implement the SR models with E5 for additionl comparison, whose results are reported in Appendix F.

Dataset	Metric	CL-Based					NCL	Improv.
		CLS4Rec	CoSeRec	DUORec	EC4Rec	SCL	NCL-SR	(%)
Beauty	R@10 ↑	0.0690	0.0698	0.0706	0.0628	0.0709	**0.0791**	11.6%
	N@10 ↑	0.0367	0.0375	0.0383	0.0344	0.0375	**0.0440**	14.9%
	R@20 ↑	0.1008	0.1007	0.1052	0.0971	0.1026	**0.1135**	7.89%
	N@20 ↑	0.0447	0.0453	0.0470	0.0430	0.0456	**0.0526**	11.9%
Games	R@10 ↑	0.0992	0.1057	0.1041	0.0998	0.1012	**0.1140**	7.85%
	N@10 ↑	0.0521	0.0553	0.0543	0.0524	0.0533	**0.0611**	10.5%
	R@20 ↑	0.1458	0.1560	0.1545	0.1516	0.1499	**0.1683**	7.89%
	N@20 ↑	0.0638	0.0680	0.0669	0.0654	0.0655	**0.0748**	10.0%
Sports	R@10 ↑	0.0325	0.0327	0.0332	0.0337	0.0314	**0.0441**	30.9%
	N@10 ↑	0.0173	0.0169	0.0174	0.0180	0.0163	**0.0237**	31.7%
	R@20 ↑	0.0529	0.0501	0.0504	0.0511	0.0473	**0.0660**	24.8%
	N@20 ↑	0.0223	0.0213	0.0217	0.0223	0.0203	**0.0292**	30.9%
Toys	R@10 ↑	0.0898	0.0868	0.0882	0.0871	0.0895	**0.0941**	4.79%
	N@10 ↑	0.0511	0.0490	0.0501	0.0497	0.0511	**0.0537**	5.09%
	R@20 ↑	0.1221	0.1199	0.1232	0.1201	0.1249	**0.1286**	2.96%
	N@20 ↑	0.0592	0.0573	0.0589	0.0580	0.0600	**0.0623**	3.83%
Office	R@10 ↑	0.0974	0.0967	0.0974	0.0913	0.0980	**0.1047**	6.84%
	N@10 ↑	0.0498	0.0487	0.0518	0.0465	0.0502	**0.0534**	3.09%
	R@20 ↑	0.1471	0.1561	0.1497	0.1459	0.1500	**0.1625**	4.10%
	N@20 ↑	0.0623	0.0635	0.0647	0.0603	0.0634	**0.0681**	5.26%
Auto	R@10 ↑	0.1159	0.1171	0.1098	0.1073	0.1122	**0.1354**	15.6%
	N@10 ↑	0.0575	0.0576	0.0572	0.0535	0.0559	**0.0714**	24.0%
	R@20 ↑	0.1829	0.1902	0.1927	0.1793	0.1780	**0.2085**	8.20%
	N@20 ↑	0.0742	0.0761	0.0781	0.0713	0.0724	**0.0899**	15.1%

Table 2: Main results on recommendation performance of different CL and NCL methods.

5.3 Comparing against CL-based Methods

We compare NCL-SR against state-of-the-art CL-based SR methods. The performance results are reported in Table 2. We observe: (1) our method demonstrates consistent performance improvements over the baseline CL methods across all metrics and datasets, with an average improvement of 12.48% compared to the second best method. (2) On the most sparse dataset (i.e., Sports), NCL-SR still achieves the most performance gain, suggesting that NCL is more efficient that CL-based methods in terms of learning generalizable representations from sparse data. Recall Due to the space limit, we provide additional experiments on exploring the relationship between CL, NCL and data augmentation in Appendix D, and compare their memory consumption in Appendix G.

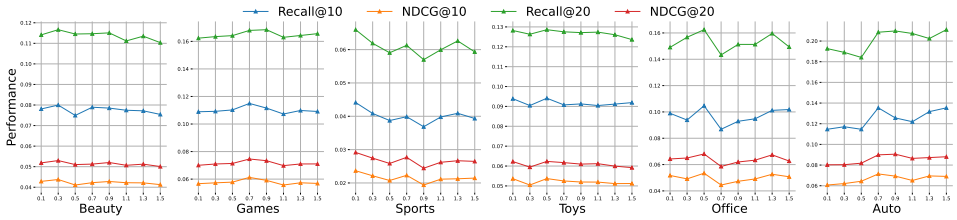
5.4 Ablation Study

Alignment and Uniformity. We conduct an ablation to understand the effects of learning aligned and uniform representations for our SR task. In particular, we set $\lambda_1 = 0$ in Equation 11 to remove the uniformity (Ours w/o Uni.) or set $\lambda_2 = 0$ to remove the alignment (Ours w/o Align.), respectively. The experimental results are reported in Table 3. From Table 3, it is observed that in general, both alignment and uniformity play a vital role in learning generalizable user representations for SR. Specifically, when removing the alignment loss in Equation 11, the averaged performance on Recall@10 and NDCG@10 across all datasets would drop by 6.10% and 6.47%, respectively. In comparison, there is an averaged drop of 4.00% on Recall@10 and 3.92% on NDCG@10 if uniformity is removed. Furthermore, better recommendation performance can be observed when only learn the aligned user representations. For instance, on Beauty, there is a noticeable increase on both Recall@20 and NDCG@20 without uniformity loss. To this end, we conclude that promoting alignment may be more important than uniformity in learning generalizable representations for SR applications. This further necessitates the design and merit of the proposed preference-preserving augmentation, which addresses the preference inconsistency caused by ad-hoc augmentations.

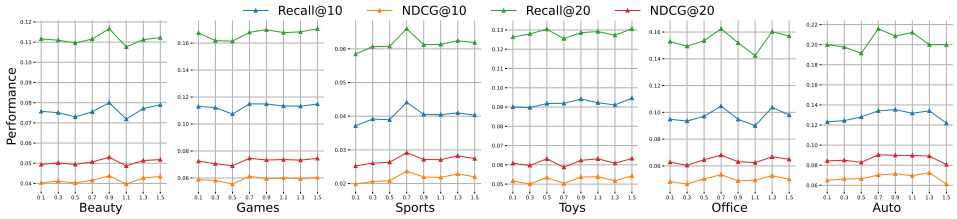
DP augmentation and NCL loss. We then conduct another ablation to investigate the effects of the proposed DP augmentation and the NCL losses. Specifically, we mask out either module by (1) replacing our DP augmentation with other data augmentation operations that are randomly sampled from the CL-based baselines (Ours w/o DP Aug.) or (2) replacing the NCL loss with the

Variants	Metric	Beauty	Games	Sports	Toys	Office	Auto
		Recall/NDCG	Recall/NDCG	Recall/NDCG	Recall/NDCG	Recall/NDCG	Recall/NDCG
NCL-SR (Ours)	@10	**0.0791 / 0.0440**	**0.1140 / 0.0611**	**0.0441 / 0.0237**	**0.0941 / 0.0537**	0.1047 / **0.0534**	**0.1354 / 0.0714**
	@20	0.1135 / 0.0526	**0.1683 / 0.0748**	**0.0660 / 0.0292**	0.1286 / **0.0623**	**0.1625 / 0.0681**	0.2085 / **0.0899**
(1) Ours w/o Align.	@10	0.0761 / 0.0408	0.1133 / 0.0590	0.0384 / 0.0205	0.0916 / 0.0504	0.0954 / 0.0524	0.1293 / 0.0691
	@20	0.1124 / 0.0499	0.1673 / 0.0725	0.0606 / 0.0261	0.1270 / 0.0594	0.1580 / 0.0680	0.1988 / 0.0865
(2) Ours w/o Uni.	@10	0.0784 / 0.0437	0.1128 / 0.0592	0.0404 / 0.0221	0.0936 / 0.0529	0.0983 / 0.0519	0.1280 / 0.0661
	@20	**0.1149 / 0.0529**	0.1667 / 0.0728	0.0630 / 0.0277	**0.1304** / 0.0622	0.1519 / 0.0652	**0.2171** / 0.0885
(1) Ours w/o DP Aug.	@10	0.0717 / 0.0394	0.1029 / 0.0552	0.0364 / 0.0189	0.0911 / 0.0517	**0.1053** / 0.0530	0.1329 / 0.0705
	@20	0.1046 / 0.0477	0.1520 / 0.0675	0.0543 / 0.0234	0.1264 / 0.0606	0.1554 / 0.0656	0.2049 / 0.0886
(2) Ours w/o NCL	@10	0.0727 / 0.0394	0.1014 / 0.0529	0.0369 / 0.0198	0.0834 / 0.0456	0.0980 / 0.0527	0.1268 / 0.0678
	@20	0.1077 / 0.0481	0.1529 / 0.0659	0.0553 / 0.0245	0.1160 / 0.0538	0.1465 / 0.0647	0.2073 / 0.0879

Table 3: Ablation Study on different components of NCL-SR.



(a) Uniformity sensitivity: changing λ_1 with fixed λ_2 in Equation 11.



(b) Alignment sensitivity: changing λ_2 with fixed λ_1 in Equation 11.

Figure 2: Sensitivity Analysis w.r.t. Alignment and Uniformity.

contrastive loss (Ours w/o NCL). The results are reported in Table 3. It is observed that removing either module generally leads to performance degradation, indicating that both DP augmentation and NCL representation learning are necessary for improving recommendation performance.

5.5 Sensitivity Analysis

We finally study the sensitivity of NCL-SR w.r.t. the key hyperparameters λ_1 and λ_2 in Equation 11, where λ_1 controls the strength of uniformity and λ_2 controls the strength of alignment. When performing sensitivity analysis, we select and fix the best λ_1, and change λ_2 to understand the sensitivity w.r.t. alignment. Similarly, λ_2 is fixed and λ_1 is changed for the sensitivity analysis w.r.t. uniformity. The performance is visualized in Figure 2. We observe that there exist different optimal configurations for different datasets. Additional sensitivity analysis can be found at Appendix E.

6 Conclusion

In this work, we design the very first Non-Contrastive Learning framework for Sequential Recommendation: NCL-SR. In particular, we present a principled data augmentation operation to produce preference-preserving user profiles, based on which we define uniformity and alignment losses to learn user representations. Our investigations reveal that learning uniform and aligned user representations play a vital role in the SR task, and alignment may be more important than uniformity. Our extensive experimental results suggest NCL-SR consistently exhibits superior performance over state-of-the-art SR models and CL-based SR methods.

REFERENCES

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021.

Jaejin Cho, Jesús Villalba, Laureano Moro-Velazquez, and Najim Dehak. Non-contrastive self-supervised learning for utterance-level information extraction from speech. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1284–1295, 2022.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pp. 265–284. Springer, 2006.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 299–315, 2022.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Silviu Guiasu and Abe Shenitzer. The principle of maximum entropy. *The mathematical intelligencer*, 7:42–48, 1985.

Xingzhuo Guo, Junwei Pan, Ximei Wang, Baixu Chen, Jie Jiang, and Mingsheng Long. On the embedding collapse when scaling up recommendation models. *arXiv preprint arXiv:2310.04400*, 2023.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pp. 507–517, 2016.

D Hemkumar and Pvn Prashanth. A weighted privacy mechanism under differential privacy. In *2023 4th International Conference on Intelligent Technologies (CONIT)*, pp. 1–6. IEEE, 2024.

Nicholas J Higham. *Functions of matrices: theory and computation*. SIAM, 2008.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 585–593, 2022.

Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

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

Gayan K Kulatilleke, Marius Portmann, and Shekhar S Chandra. Efficient block contrastive learning via parameter-free meta-node approximation. *Neurocomputing*, 561:126850, 2023.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pp. 656–672. IEEE, 2019.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. Text is all you need: Learning language representations for sequential recommendation. *arXiv preprint arXiv:2305.13731*, 2023.

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

Jianghao Lin, Xinyi Dai, Rong Shan, Bo Chen, Ruiming Tang, Yong Yu, and Weinan Zhang. Large language models make sample-efficient recommender systems. *Frontiers of Computer Science*, 19(4):194328, 2025.

Xin Liu, Zhongdao Wang, Ya-Li Li, and Shengjin Wang. Self-supervised learning via maximum entropy coding. *Advances in Neural Information Processing Systems*, 35:34091–34105, 2022.

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

Yao Lv, Jiajie Xu, Rui Zhou, Junhua Fang, and Chengfei Liu. Ssrgan: A generative adversarial network for streaming sequential recommendation. In *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part III 26*, pp. 36–52. Springer, 2021.

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

Shuang Ni, Wei Zhou, Junhao Wen, Linfeng Hu, and Shutong Qiao. Enhancing sequential recommendation with contrastive generative adversarial network. *Information Processing & Management*, 60(3):103331, 2023.

Tieyun Qian, Yile Liang, Qing Li, and Hui Xiong. Attribute graph neural networks for strict cold start recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3597–3610, 2020.

Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pp. 813–823, 2022.

Xubin Ren, Wei Wei, Lianghao Xia, and Chao Huang. A comprehensive survey on self-supervised learning for recommendation. *arXiv preprint arXiv:2404.03354*, 2024.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

Zhengxiang Shi, Xi Wang, and Aldo Lipani. Self contrastive learning for session-based recommendation. In *European Conference on Information Retrieval*, pp. 3–20. Springer, 2024.

Jiajie Su, Chaochao Chen, Weiming Liu, Fei Wu, Xiaolin Zheng, and Haoming Lyu. Enhancing hierarchy-aware graph networks with deep dual clustering for session-based recommendation. In *Proceedings of the ACM Web Conference 2023*, pp. 165–176, 2023.

Rishabh Subramanian. Have the cake and eat it too: Differential privacy enables privacy and precise analytics. *Journal of Big Data*, 10(1):117, 2023.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1441–1450, 2019.

Chunyang Wang, Yanmin Zhu, Haobing Liu, Tianzi Zang, Jiadi Yu, and Feilong Tang. Deep meta-learning in recommendation systems: A survey. *arXiv preprint arXiv:2206.04415*, 2022a.

Lei Wang, Ee-Peng Lim, Zhiwei Liu, and Tianxiang Zhao. Explanation guided contrastive learning for sequential recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2017–2027, 2022b.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. corr abs/2212.03533 (2022), 2022c.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pp. 9929–9939. PMLR, 2020.

Wenjie Wang, Pengfei Tang, Jian Lou, and Li Xiong. Certified robustness to word substitution attack with differential privacy. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1102–1112, 2021.

Yu Wang, Hengrui Zhang, Zhiwei Liu, Liangwei Yang, and Philip S Yu. Contrastvae: Contrastive variational autoencoder for sequential recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2056–2066, 2022d.

Xuansheng Wu, Huachi Zhou, Yucheng Shi, Wenlin Yao, Xiao Huang, and Ninghao Liu. Could small language models serve as recommenders? towards data-centric cold-start recommendation. In *Proceedings of the ACM on Web Conference 2024*, pp. 3566–3575, 2024.

Zihao Wu, Xin Wang, Hong Chen, Kaidong Li, Yi Han, Lifeng Sun, and Wenwu Zhu. Diff4rec: Sequential recommendation with curriculum-scheduled diffusion augmentation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 9329–9335, 2023.

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pp. 1259–1273. IEEE, 2022.

Yaowen Ye, Lianghao Xia, and Chao Huang. Graph masked autoencoder for sequential recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 321–330, 2023.

Zhenrui Yue, Huimin Zeng, Ziyi Kou, Lanyu Shang, and Dong Wang. Defending substitution-based profile pollution attacks on sequential recommenders. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 59–70, 2022.

Zhenrui Yue, Yueqi Wang, Zhankui He, Huimin Zeng, Julian McAuley, and Dong Wang. Linear recurrent units for sequential recommendation. *arXiv preprint arXiv:2310.02367*, 2023.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pp. 12310–12320. PMLR, 2021.

Yabin Zhang, Zhenlei Wang, Wenhui Yu, Lantao Hu, Peng Jiang, Kun Gai, and Xu Chen. Soft contrastive sequential recommendation. *ACM Transactions on Information Systems*, 2024.

Yifan Zhang, Zhiquan Tan, Jingqin Yang, Weiran Huang, and Yang Yuan. Matrix information theory for self-supervised learning. *arXiv preprint arXiv:2305.17326*, 2023.

Xiaolin Zheng, Jiajie Su, Weiming Liu, and Chaochao Chen. Ddghm: Dual dynamic graph with hybrid metric training for cross-domain sequential recommendation. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 471–481, 2022.

Jinghao Zhou, Li Dong, Zhe Gan, Lijuan Wang, and Furu Wei. Non-contrastive learning meets language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11028–11038, 2023.

Zhijian Zhuo, Yifei Wang, Jinwen Ma, and Yisen Wang. Towards a unified theoretical understanding of non-contrastive learning via rank differential mechanism. *arXiv preprint arXiv:2303.02387*, 2023.

APPENDIX A: DIFFERENTIAL PRIVACY AND PROOF FOR THEOREM 1

Differential privacy is a privacy mechanism that introduces randomness to prevent information leakage of individuals in datasets (Dwork et al., 2006).

Definition 3 (Differential Privacy) *Formally, a randomized mechanism $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ satisfies $\epsilon-$DP if for all neighboring inputs $X \sim X^{'}$ and for all sets of outputs $Y \subseteq \mathcal{Y}$, the following inequality holds:*

$$P[\mathcal{M}(X) \in Y] \leq e^{\epsilon} P[\mathcal{M}(X^{'}) \in Y]. \tag{12}$$

In the above definition, $\epsilon > 0$ is the privacy parameter quantifying the privacy guarantees. Specifically, the smaller ϵ is, the tighter the bound is, indicating less change is allowed in the output distribution when modifying X to X'. Furthermore, if the modification from X to X' is too large, leading to a large change in P, then ϵ may also be large to make $\mathcal{M}$ ϵ-DP. In this case, the privacy is less protected.

From a high-level point of view, we regard each user profile in our recommendation problem as a database in the definition of DP. Consequently, each interacted item within a profile can be regarded as a record in a database. According to the definition of DP, if a mechanism satisfies DP, an observer analyzing its output cannot tell whether a particular data record was used in the computation. As such, applying DP to user profile generation, the recommender functions as the observer. By ensuring DP, the recommender cannot tell whether a particular item from the user's history was used to compute the item scores. Therefore, when we formulate the above process into our user profile generation, it is straightforward that: with limited perturbations, the generated user profile will not cause a significant change in the resulting scores if the entire generation satisfies DP. Finally, the above process can also be prescribed by the mathematical definition of DP: the output distribution of a DP mechanism will exhibit a limited change, if the input is perturbed with a limited budget.

In this appendix, we provide proof for Theorem 1, which is adapted from Wang et al. (2021).

Theorem 1 (Preference-Preservation with Guarantees) *Suppose a recommendation mechanism $\mathcal{M}$ satisfies ϵ-DP. For any user sequence x, if $\mathcal{M}$ successfully recommends the ground-truth item y, which is the j-th item within the item scope $\mathcal{I}$: $y := x_j$, and the predictive score for the ground-truth item is greater than the second-largest runner-up score of another item with a small multiplicative factor $e^{2\epsilon}$:*

$$\mathbb{E}(\mathcal{M}(x)_j) > e^{2\epsilon} \max_{k:k\neq j} \mathbb{E}(\mathcal{M}(x)_k),$$

then for its augmented view x', its predicted top-1 item remains the same if there is only limited modification from x to $x^{'}$:

$$\mathbb{E}(\mathcal{M}(x^{'})_j) > \max_{k:k\neq j} \mathbb{E}(\mathcal{M}(x^{'})_k).$$

.

Proof 1

$$\begin{aligned}
\mathbb{E}(\mathcal{M}(x)_j) &= \int_0^1 \mathbb{P}(\mathcal{M}(x)_j) > t)dt \\
&\leq \int_0^1 \mathbb{P}(e^{\epsilon}\mathcal{M}(x')_j) > t)dt \\
&= e^{\epsilon} \int_0^1 \mathbb{P}(\mathcal{M}(x')_j) > t)dt \\
&= e^{\epsilon}\mathbb{E}(\mathcal{M}(x^{'})_j).
\end{aligned} \tag{13}$$

Therefore, we have

$$\begin{aligned}
\mathbb{E}(\mathcal{M}(x)_j) &\leq e^{\epsilon}\mathbb{E}(\mathcal{M}(x^{'})_j) \\
\mathbb{E}(\mathcal{M}(x^{'})_k) &\leq e^{\epsilon}\mathbb{E}(\mathcal{M}(x)_k),\ k \neq j
\end{aligned} \tag{14}$$

Finally, we have

$$\begin{aligned}\mathbb{E}(\mathcal{M}(x^{'})_j) &\geq \frac{\mathbb{E}(\mathcal{M}(x)_j)}{e^{\epsilon}} \\ &\geq \frac{e^{2\epsilon}\max_{k:k\neq j}\mathbb{E}(\mathcal{M}(x)_k)}{e^{\epsilon}} \\ &= e^{\epsilon}\max_{k:k\neq j}\mathbb{E}(\mathcal{M}(x)_k) \\ &\geq \max_{k:k\neq j}\mathbb{E}(\mathcal{M}(x^{'})_k).\end{aligned} \tag{15}$$

Finally, our approach is closely related to the findings in the cited (Wang et al., 2021; Lecuyer et al., 2019). In (Wang et al., 2021; Lecuyer et al., 2019), DP is used to provide certified robustness of classifiers against adversarial examples. That is, in these studies, the goal is to ensure that a classifier's predictions remain consistent even if the input is deliberately perturbed. Inspired by [1,2], our method leverages DP to generate augmented user profiles that preserve user preferences. In the context of recommendation, this means that the model should produce consistent recommendations for the original user profile and the augmented one if they represent the same user preference. Additionally, recent studies (Subramanian, 2023; Hemkumar & Prashanth, 2024) have explored the trade-off between the analysis accuracy and perturbations under DP. Such studies do not necessarily only focus on the privacy benefits of DP, but also aim to maintain high accuracy under noises or perturbations. In summary, inspired by (Wang et al., 2021; Lecuyer et al., 2019; Subramanian, 2023; Hemkumar & Prashanth, 2024), we propose to use DP to augment user profiles for recommendation systems. Our goal is to preserve user preferences during the augmentation process: a perturbed user profile should yield the same recommendations as its original counterpart. This consistency indicates that user preferences are well-preserved in the augmented data, ensuring high-quality positive training samples to compute the NCL losses.

APPENDIX B: EFFICIENT DESIGN

Guarantee with Empirical Expectation. In Theorem 1, the expectation $\mathbb{E}(\mathcal{M}(x)_j)$ is usually estimated via Monte Carlo sampling. That is, one must repeat the inference of the exponential mechanism to draw candidate profiles with different augmentations and compute $\hat{\mathbb{E}}(\mathcal{M}(x)_j) = \frac{1}{m}\sum_{m=1}^{m} f(x_m^{'})$. In this appendix, we adapt the proof from Lecuyer et al. (2019) to show that the user preference could still be preserved with approximated empirical expectation. We first obtain an upper bound and a lower bound for $\mathbb{E}(\mathcal{M}(x)_j)$. Specifically, with Hoeffding's inequality with probability η, we have

$$\begin{aligned}\hat{\mathbb{E}}^{lb}(\mathcal{M}(x)) &= \hat{\mathbb{E}}(\mathcal{M}(x)) - \sqrt{\frac{1}{2m} ln(\frac{2|\mathcal{I}|}{1-\eta})} \\ \hat{\mathbb{E}}^{ub}(\mathcal{M}(x)) &= \hat{\mathbb{E}}(\mathcal{M}(x)) + \sqrt{\frac{1}{2m} ln(\frac{2|\mathcal{I}|}{1-\eta})}\end{aligned} \tag{16}$$

Theorem 2 (Guarantee with Empirical Expectation) *Suppose a recommendation mechanism $\mathcal{M}$ satisfies ϵ-DP. For any user sequence x, if $\mathcal{M}$ successfully recommends the ground-truth item y, which is the j-th item within the item scope $\mathcal{I}$: $y := x_j$, and the predictive score for the ground-truth item is great than the second-largest runner-up score of another item with a small multiplicative factor $e^{2\epsilon}$:*

$$\hat{\mathbb{E}}^{lb}(\mathcal{M}(x)_j) > e^{2\epsilon} \max_{k:k\neq j} \hat{\mathbb{E}}^{ub}(\mathcal{M}(x)_k), \tag{17}$$

then for its augmented view x', with probability higher than η, we have

$$\hat{\mathbb{E}}(\mathcal{M}(x^{'})_j) > \max_{k:k\neq j} \hat{\mathbb{E}}(\mathcal{M}(x^{'})_k). \tag{18}$$

.

Proof 2

$$\begin{aligned}\hat{\mathbb{E}}(\mathcal{M}(x^{'})_j) &\geq \frac{\hat{\mathbb{E}}(\mathcal{M}(x)_j)}{e^{\epsilon}} \\ &\geq \frac{\hat{\mathbb{E}}^{lb}(\mathcal{M}(x)_j)}{e^{\epsilon}}.\end{aligned} \tag{19}$$

Similarly, we have

$$\hat{\mathbb{E}}(\mathcal{M}(x^{'})_k) \leq e^{\epsilon} \max_{k:k\neq j} \hat{\mathbb{E}}^{ub}(\mathcal{M}(x)_k) \tag{20}$$

Finally, we have

$$\begin{aligned}\hat{\mathbb{E}}(\mathcal{M}(x^{'})_j) &\geq \frac{\hat{\mathbb{E}}(\mathcal{M}(x)_j)}{e^{\epsilon}} \\ &\geq \frac{\hat{\mathbb{E}}^{lb}(\mathcal{M}(x)_j)}{e^{\epsilon}} \\ &\geq \frac{e^{2\epsilon} \max_{k:k\neq j} \hat{\mathbb{E}}^{ub}(\mathcal{M}(x)_k)}{e^{\epsilon}} \\ &= e^{\epsilon} \max_{k:k\neq j} \hat{\mathbb{E}}^{ub}(\mathcal{M}(x)_k) \\ &\geq \max_{k:k\neq j} \mathbb{E}(\mathcal{M}(x^{'})_k).\end{aligned} \tag{21}$$

Exponential Mechanism at Item Level. Recall a critical caveat of designing the exponential mechanism at the user-level is that the search space grows exponentially w.r.t. the length of user histories if the DP augmentation is defined at user-level: $\mathcal{O}(k^l)$. This is because each item $x_i \in x = [x_1, x_2, ..., x_l]$ has k replacements. As such, there are (k^l) possible candidate user profiles for exponential mechanism to sample from, i.e., $|X'| = k^l$.

Given the sparsity of recommendation data, designing DP augmentations for user sequences introduces prohibitive computation complexity and noises. Therefore, we propose to define the DP augmentation at item level to generate DP augmented user profiles. In this case, the search space grows linearly w.r.t. the length of user histories. To see this, we note that when applying the exponential mechanism to each item $x_i \in x = [x_1, x_2, ..., x_l]$, the exponential mechanism just to need to sample from k candidates, and repeat for all items within the history l times, leading to $\mathcal{O}(k \cdot l)$. Specifically, we compute a DP item for each to-be-replaced item from the original user history using the exponential mechanism. Then, the augmented user profile is generated by replacing the selected items within original user history with the calculated DP items.

To achieve linear complexity, we propose to define the exponential mechanism at item level to generate augmented user profiles, we compute a DP item for each to-be-replaced item from the original user history using the exponential mechanism. Then, the augmented user profile is generated by replacing the selected items within original user history with the calculated DP items. For an item x_i, its scoring function w.r.t. a synonym item x_j is defined as:

$$u(x_i, x_j) = e^{\mathrm{CosSim}(f(x_i), f(x_j))}. \tag{22}$$

Correspondingly, the embedding of the DP item for x_i is computed with

$$\hat{z_{x_i}} = \sum_{x_j \in \mathcal{N}_k(x_i)} \frac{e^{\frac{\epsilon u(x_i, x_j)}{2\Delta_u}}}{\sum_{x_k \in \mathcal{N}_k(x_i)} e^{\frac{\epsilon u(x_i, x_k)}{2\Delta_u}}} \cdot f(x_j). \tag{23}$$

We then replace x_i with the DP item to generated the augmented user profile, which is visualized in Figure 1.

Note this efficient design is feasible and remain DP, because of the two key properties of DP. First, DP has the post-processing property: any computation applied to the output of a DP algorithm remains DP. As such, if the augmentation operation is DP, then recommendation process based on the augmented user profile is also DP. Second, the expected output stability property of DP reduces the computational costs of DP augmentation. This property allows us to generate an expected augmented profile for each user to preserve user preferences in the augmented profiles, instead of generating massive augmented samples for Monte-Carlo sampling.

APPENDIX C: DATASETS STATISTICS

In this section, we provide further details about our datasets. We adopt 6 public benchmark recommendation datasets to evaluate NCL-SR: Beauty, Games, Sports, Toys, Office and Auto He & McAuley (2016); McAuley et al. (2015). These datasets cover different application domains, and are characterized with different sparsity and sequence lengths. Following Yue et al. (2022), we use the 5-core processing to filter out infrequent items and users. The user-item interactions are sorted chronologically to construct training sequences.

To simulate the data sparsity and cold-start users, the split ratio of the training, validation, and test sets is 2:2:6 in our paper. Such a split ratio is selected because we aim to explore the model's performance under a limited quantity of training data and the model's generalization on cold-start users, following the recent studies Wu et al. (2024); Qian et al. (2020); Wang et al. (2022a); Lin et al. (2025). For instance, in Lin et al. (2025), there was only 10% training data used to train the model for the recommendation task. However, in our experiments, it is observed that many baselines could not converge under extreme data sparsity. Therefore, we increase the number of training samples for the baselines to converge (20%). This enables a fairer comparison between our method against the baselines. The details datasets statistics are summarized in Table 4.

Datasets	users	Items	Interaction	Length	Density
Beauty	22332	12086	198K	8.88	7e-4
Games	15264	7676	147K	9.69	1e-3
Sports	35265	18173	293K	8.32	5e-4
Toys	19124	11758	165K	8.64	7e-4
Office	4895	2414	53K	10.86	4e-3
Auto	1281	844	8K	6.70	8e-3

Table 4: Overall dataset statistics.

APPENDIX D: ADDITIONAL EXPERIMENTAL RESULTS

In this section, we further investigate the relationship between CL, NCL and data augmentation.

In particular, we first replace the contrastive loss with NCL losses for CL-based baselines. The results are reported in Table 5. In Table 5, each column represent a training scheme being a CL-based baseline or its NCL version. The last column includes the performance of NCL-SR as reference. The comparison is made two-column-wise, with better performance highlighted in bold. In addition, we also make a comparison per row, with the second-best performance highlighted with underlines. From Table 5, we observe that NCL indeed has the potential to improve the recommendation performance while eliminating the need of negative samples for CL-baselines. For instance, the performance of CLS4Rec is generally improved with non-contrastive training over all 6 datasets. For remaining CL-based methods, performance improvement could be observed over half of the datasets. There exists inconsistent performance improvement for certain baselines across different datasets. Such inconsistency is expected, because the data augmentation operations of the CL-baselines are usually ad-hoc and may fail to preserve user preferences. Therefore, the preference inconsistency of the augmented data may degrade the effectiveness of NCL training. Finally, it is observed the performance of NCL-SR is still better than NCL with ad-hoc augmentations. This suggests that our proposed preference-preserving augmentation can indeed better exploit the capacity of NCL training.

Dataset	Metric	CLS4Rec	CLS4Rec + NCL	CoSeRec	CoSeRec + NCL	DUORec	DUORec + NCL	EC4Rec	EC4Rec + NCL	Ours
Beauty	R@10 ↑	0.0690	**0.0693**	**0.0698**	0.0683	0.0706	**0.0717**	0.0628	**0.0665**	**0.0791**
	N@10 ↑	0.0367	**0.0373**	**0.0375**	0.0373	0.0383	**0.0394**	0.0344	**0.0358**	**0.0440**
	R@20 ↑	**0.1008**	0.1007	0.1007	**0.1023**	**0.1052**	0.1046	0.0971	**0.0996**	**0.1135**
	N@20 ↑	0.0447	**0.0451**	0.0453	**0.0458**	0.0470	**0.0477**	0.0430	**0.0441**	**0.0526**
Games	R@10 ↑	0.0992	**0.1028**	**0.1057**	0.1031	**0.1041**	0.1002	0.0998	**0.1029**	**0.1140**
	N@10 ↑	0.0521	**0.0534**	**0.0553**	0.0539	**0.0543**	0.0520	0.0524	**0.0552**	**0.0611**
	R@20 ↑	0.1458	**0.1550**	**0.1560**	0.1548	**0.1545**	0.1519	0.1516	**0.1520**	**0.1683**
	N@20 ↑	0.0638	**0.0665**	**0.0680**	0.0669	**0.0669**	0.0650	0.0654	**0.0675**	**0.0748**
Sports	R@10 ↑	0.0325	**0.0353**	0.0327	**0.0352**	0.0332	**0.0364**	**0.0337**	0.0258	**0.0441**
	N@10 ↑	0.0173	**0.0189**	0.0169	**0.0183**	0.0174	**0.0189**	**0.0180**	0.0133	**0.0237**
	R@20 ↑	0.0529	**0.0552**	0.0501	**0.0543**	0.0504	**0.0543**	**0.0511**	0.0421	**0.0660**
	N@20 ↑	0.0223	**0.0239**	0.0213	**0.0231**	0.0217	**0.0234**	**0.0223**	0.0174	**0.0292**
Toys	R@10 ↑	0.0898	**0.0911**	0.0868	**0.0888**	**0.0882**	0.0876	0.0871	**0.0874**	**0.0941**
	N@10 ↑	0.0511	**0.0517**	0.0490	**0.0502**	**0.0501**	0.0481	**0.0497**	0.0494	**0.0537**
	R@20 ↑	0.1221	**0.1264**	0.1199	**0.1235**	**0.1232**	0.1206	**0.1201**	0.1194	**0.1286**
	N@20 ↑	0.0592	**0.0606**	0.0573	**0.0589**	**0.0589**	0.0564	**0.0580**	0.0575	**0.0623**
Office	R@10 ↑	**0.0974**	0.0970	0.0967	**0.1005**	0.0974	**0.1053**	0.0913	**0.1012**	0.1047
	N@10 ↑	0.0498	**0.0512**	0.0487	**0.0509**	0.0518	**0.0530**	0.0465	**0.0503**	**0.0534**
	R@20 ↑	0.1471	**0.1484**	**0.1561**	0.1503	0.1497	**0.1554**	**0.1459**	0.1452	**0.1625**
	N@20 ↑	0.0623	**0.0642**	**0.0635**	0.0634	0.0647	**0.0656**	0.0603	**0.0614**	**0.0681**
Auto	R@10 ↑	0.1159	**0.1256**	0.1171	**0.1220**	0.1098	**0.1329**	0.1073	**0.1305**	**0.1354**
	N@10 ↑	0.0575	**0.0630**	0.0576	**0.0645**	0.0572	**0.0705**	0.0535	**0.0629**	**0.0714**
	R@20 ↑	0.1829	**0.1878**	**0.1902**	0.1841	0.1927	**0.2049**	0.1793	**0.2012**	**0.2085**
	N@20 ↑	0.0742	**0.0786**	0.0761	**0.0802**	0.0781	**0.0886**	0.0713	**0.0808**	**0.0899**

Table 5: Replacing the contrastive loss of CL-based baselines with NCL losses.

Next, we replace the ad-hoc augmentations with our proposed preference-preserving augmentation. In this context, CLS4Rec, CoSeRec and EC4Rec converge to the same CL framework. In comparison, the modified DUORec is also a new baseline, because DUORec has the additional dropout mask. The results are reported in Table 6. In Table 6, each column represent a training scheme being a CL-based baseline or its DP version. The last column includes the performance of NCL-SR as reference. The first comparison is made with first four columns and the second comparison is made with the next two columns, with better performance highlighted in bold. In addition, we also make a comparison per row, with the second-best performance highlighted with underlines. From Table 6, we observe that despite our preference-preserving augmentation, CL-based methods still suffer from generalization issue because of the reliance on negative samples. Due to the limited GPU memory, the training batch size is 4. Under such a small batch size, it is challenging for the CL-based methods to promote uniformity in the learned user/item representations. As a consequence, the trained recommenders may still suffer from generalization issue. Such an observation shows that the per-

formance of CL-based methods is sensitive to negative samples. This is a systematic loophole of CL, which necessiates the developmennt of our NCL framework.

		CLS4Rec	CoSeRec	EC4Rec	+ DP	DUORec	+ DP	Ours
Beauty	R@10 ↑	0.0690	0.0698	0.0628	**0.0727**	**0.0706**	0.0674	**0.0791**
	N@10 ↑	0.0367	0.0375	0.0344	**0.0394**	**0.0383**	0.0363	**0.0440**
	R@20 ↑	0.1008	0.1007	0.0971	**0.1077**	**0.1052**	0.0984	**0.1135**
	N@20 ↑	0.0447	0.0453	0.0430	**0.0481**	**0.0470**	0.0440	**0.0526**
Games	R@10 ↑	0.0992	**0.1057**	0.0998	0.0989	**0.1041**	0.1014	**0.1140**
	N@10 ↑	0.0521	**0.0553**	0.0524	0.0514	**0.0543**	0.0529	**0.0611**
	R@20 ↑	0.1458	**0.1560**	0.1516	0.1513	**0.1545**	0.1529	**0.1683**
	N@20 ↑	0.0638	**0.0680**	0.0654	0.0645	**0.0669**	0.0659	**0.0748**
Sports	R@10 ↑	0.0325	0.0327	0.0337	**0.0369**	0.0332	**0.0358**	**0.0441**
	N@10 ↑	0.0173	0.0169	0.0180	**0.0198**	0.0174	**0.0191**	**0.0237**
	R@20 ↑	0.0529	0.0501	0.0511	**0.0553**	0.0504	**0.0541**	**0.0660**
	N@20 ↑	0.0223	0.0213	0.0223	**0.0245**	0.0217	**0.0237**	**0.0292**
Toys	R@10 ↑	**0.0898**	0.0868	0.0871	0.0834	**0.0882**	0.0796	**0.0941**
	N@10 ↑	**0.0511**	0.0490	0.0497	0.0456	**0.0501**	0.0440	**0.0537**
	R@20 ↑	**0.1221**	0.1199	0.1201	0.1160	**0.1232**	0.1141	**0.1286**
	N@20 ↑	**0.0592**	0.0573	0.0580	0.0538	**0.0589**	0.0527	**0.0623**
Office	R@10 ↑	**0.0974**	0.0967	0.0913	0.0958	0.0974	**0.0980**	**0.1047**
	N@10 ↑	**0.0498**	0.0487	0.0465	0.0495	0.0518	**0.0527**	**0.0534**
	R@20 ↑	0.1471	**0.1561**	0.1459	0.1398	**0.1497**	0.1465	**0.1625**
	N@20 ↑	0.0623	**0.0635**	0.0603	0.0606	**0.0647**	**0.0647**	**0.0681**
Auto	R@10 ↑	0.1159	0.1171	0.1073	**0.1317**	0.1098	**0.1268**	**0.1354**
	N@10 ↑	0.0575	0.0576	0.0535	**0.0644**	0.0572	**0.0678**	**0.0714**
	R@20 ↑	0.1829	0.1902	0.1793	**0.2000**	0.1927	**0.2073**	**0.2085**
	N@20 ↑	0.0742	0.0761	0.0713	**0.0814**	0.0781	**0.0879**	**0.0899**

Table 6: Applying DP augmentation for CL-based methods. The best results are highlighted in bold and the second best results are highlighted with underline.

APPENDIX E: ADDITIONAL SENSITIVITY ANALYSIS

Sensitivity Analysis w.r.t. Alignment. In the alignment loss (Equation 10), there exists an additional hyperparameter γ that affects the computation of $\mathcal{L}_{align}$. Firstly, we acknowledge that different values of γ could indeed affect the model performance. This is revealed in our sensitivity analysis w.r.t. λ_2 as in Figure 2b. To see this, note that λ_2 is equivalent to a rescaled γ in our implementation. As shown in [1], $\mathcal{L}_{uniform} + \mathcal{L}_{align}$ could be expanded as follows:

$$\begin{aligned}
&\mathcal{L}_{\text{uniform}}(Z, Z^{'}) + \mathcal{L}_{\text{align}}(Z, Z^{'}) \\
&= \text{MCE}\big(\frac{1}{d} I_d, C(Z, Z^{'})\big) - \text{tr}\big(C(Z, Z^{'})\big) + \gamma \cdot \text{MCE}\big(C(Z, Z), C(Z^{'}, Z^{'})\big) \\
&= -\text{tr}\big((\frac{1}{d} I_d)\text{log}(C(Z, Z^{'}))\big) - \gamma \cdot \text{tr}\big(C(Z, Z)\text{log}(C(Z^{'}, Z^{'}))\big) + \gamma \cdot \text{tr}\big(C(Z^{'}, Z^{'})\big) + \text{const.}
\end{aligned} \tag{24}$$

In the last expansion in the third row, it is observed that both the second term and the third term are derived from $\mathcal{L}_{align}$. More importantly, both these two terms are re-scaled by γ. Since the second term and third term correspond to $\mathcal{L}_{align}$, we can instead use another scaling factor that directly adjusts the trade-off between $\mathcal{L}_{uniform}$ and $\mathcal{L}_{align}$. Specifically, we used λ_1 to re-scale $\mathcal{L}_{uniform}$ and λ_2 to re-scale $\mathcal{L}_{align}$ with γ being absorbed into λ_2. Finally, such an implementation (i.e., absorbing γ into λ_2) reduces the number of hyperparameters required for tuning.

Additional Sensitivity Analysis w.r.t. Number of Perturbations. We conduct a sensitivity analysis w.r.t. the number of perturbations in the user profile generation. Recall, the preference-preserving user profiles are generated by injecting item-level perturbations into the original user profile. In this set of experiments, we increase the number of perturbations from 1 to 8. We stopped at 8, because the averaged length of user profiles in most datasets is around 8. We report the results of sensitivity analysis in Table 7. According to Table 7, it is observed that in general, the model performance is stable when varying the number of perturbations from 1 to 4. Moreover, the performance of the model starts degrading after the number of perturbations is larger than 5. This is expected, because, with too many perturbations, the initial model may generate some wrong user preferences at the beginning. During training, such wrong user preferences are preserved, so the model is then trained to learn wrong user preferences, which degrades the performance.

Dataset	Num. of Perturbations	1	2	3	4	5	6	7	8
Beauty	R@10 ↑	0.0772	0.0779	0.0791	0.0742	0.0783	0.0749	0.0726	0.0747
	N@10 ↑	0.0415	0.0427	0.0440	0.0396	0.0430	0.0408	0.0400	0.0404
	R@20 ↑	0.1130	0.1139	0.1135	0.1098	0.1165	0.1141	0.1089	0.1113
	N@20 ↑	0.0505	0.0517	0.0526	0.0486	0.0526	0.0507	0.0492	0.0496
Games	R@10 ↑	0.1089	0.1123	0.1140	0.1101	0.1076	0.1107	0.1104	0.1085
	N@10 ↑	0.0574	0.0600	0.0611	0.0572	0.0561	0.0582	0.0573	0.0583
	R@20 ↑	0.1656	0.1664	0.1683	0.1650	0.1626	0.1647	0.1629	0.1607
	N@20 ↑	0.0716	0.0736	0.0748	0.0710	0.0700	0.0717	0.0705	0.0714
Sports	R@10 ↑	0.0397	0.0387	0.0441	0.0406	0.0382	0.0351	0.0358	0.0394
	N@10 ↑	0.0210	0.0206	0.0237	0.0209	0.0201	0.0191	0.0192	0.0212
	R@20 ↑	0.0593	0.0592	0.0660	0.0617	0.0596	0.0565	0.0549	0.0611
	N@20 ↑	0.0259	0.0258	0.0292	0.0262	0.0255	0.0244	0.0240	0.0266
Toys	R@10 ↑	0.0930	0.0958	0.0941	0.0942	0.0926	0.0908	0.0922	0.0928
	N@10 ↑	0.0511	0.0540	0.0537	0.0540	0.0525	0.0515	0.0521	0.0522
	R@20 ↑	0.1248	0.1309	0.1286	0.1314	0.1281	0.1238	0.1261	0.1302
	N@20 ↑	0.0592	0.0628	0.0623	0.0634	0.0615	0.0598	0.0607	0.0615
Office	R@10 ↑	0.0983	0.0989	0.1047	0.1028	0.0983	0.0875	0.0989	0.0980
	N@10 ↑	0.0521	0.0524	0.0534	0.0529	0.0506	0.0431	0.0498	0.0509
	R@20 ↑	0.1561	0.1535	0.1625	0.1570	0.1554	0.1459	0.1519	0.1551
	N@20 ↑	0.0662	0.0662	0.0681	0.0665	0.0649	0.0579	0.0631	0.0653
Auto	R@10 ↑	0.1390	0.1280	0.1354	0.1268	0.1220	0.1268	0.1256	0.1220
	N@10 ↑	0.0690	0.0660	0.0714	0.0667	0.0630	0.0677	0.0672	0.0626
	R@20 ↑	0.2207	0.2098	0.2085	0.2037	0.2049	0.1976	0.2024	0.1902
	N@20 ↑	0.0894	0.0867	0.0899	0.0860	0.0839	0.0856	0.0863	0.0800

Table 7: Changing number of perturbations for preference-preserving profile generation.

APPENDIX F: IMPLEMENTING ID-BASED SR MODELS WITH E5

We further implement SR models with E5, where E5 is used as the embedding model to embed the item texts. We compare our method against the E5-based SR models in the table below. It is observed that our method can outperform the E5-based SR models. We also note that it is non-trivial to apply E5 to these models, because such ID-based models originally take item IDs as inputs, whereas E5 is a text-based model.

Dataset	Metric	E5-Based				
		BERT4Rec	SASRec	NARM	LRURec	NCL-SR (Ours)
Beauty	R@10 ↑	0.0420	0.0504	0.0348	0.0446	**0.0791**
	N@10 ↑	0.0226	0.0249	0.0189	0.0210	**0.0440**
	R@20 ↑	0.0672	0.0784	0.0486	0.0614	**0.1135**
	N@20 ↑	0.0290	0.0319	0.0224	0.0292	**0.0526**
Games	R@10 ↑	0.0739	0.1061	0.0422	0.0641	**0.1140**
	N@10 ↑	0.0385	0.0556	0.0236	0.0351	**0.0611**
	R@20 ↑	0.1150	0.1568	0.0563	0.0946	**0.1683**
	N@20 ↑	0.0488	0.0682	0.0272	0.0429	**0.0748**
Sports	R@10 ↑	0.0287	0.0253	0.0105	0.0164	**0.0441**
	N@10 ↑	0.0159	0.0126	0.0064	0.0095	**0.0237**
	R@20 ↑	0.0431	0.0372	0.0144	0.0228	**0.0660**
	N@20 ↑	0.0195	0.0156	0.0074	0.0111	**0.0292**
Toys	R@10 ↑	0.0379	0.0533	0.0127	0.0198	**0.0941**
	N@10 ↑	0.0197	0.0268	0.0088	0.0133	**0.0537**
	R@20 ↑	0.0576	0.0739	0.0158	0.0256	**0.1286**
	N@20 ↑	0.0247	0.0320	0.0096	0.0147	**0.0623**
Office	R@10 ↑	0.0342	0.0307	0.0425	0.0466	**0.1047**
	N@10 ↑	0.0170	0.0156	0.0210	0.0210	**0.0534**
	R@20 ↑	0.0549	0.0434	0.0836	0.0782	**0.1625**
	N@20 ↑	0.0222	0.0187	0.0313	0.0289	**0.0681**
Auto	R@10 ↑	0.0378	0.0439	0.0402	0.0512	**0.1354**
	N@10 ↑	0.0206	0.0279	0.0207	0.0275	**0.0714**
	R@20 ↑	0.0646	0.0537	0.0683	0.0817	**0.2085**
	N@20 ↑	0.0282	0.0303	0.0227	0.0352	**0.0899**

Table 8: Implementing SR models with E5.

APPENDIX G: COMPUTATIONAL COSTS AND MEMORY CONSUMPTION BETWEEN CL AND NCL

From a theoretical point of view, the time complexity of CL is $\mathcal{O}(n^2)$, if the training batch size is n and the loss is computed with in-batch negative samples. In comparison, the time complexity of NCL is $\mathcal{O}(n)$, because there are no negative samples at all. Therefore, NCL is more computationally efficient than CL (Zhuo et al., 2023; Zbontar et al., 2021; Grill et al., 2020; Cho et al., 2022).

Furthermore, we highglight that in our method, the computation of the logarithm in Equation 8 is approximated with Taylor expansion as in (Zhang et al., 2023). In general, the time complexity of this operation is the time complexity is $\mathcal{O}(md^2)$ (under our implementation), with m being the order of the expansion and d being the dimensionality of the matrix V. In our implementation, we set $m = 4$ for computational efficiency and $d = 768$ as given by the dimensionality of E5 outputs:

$$\log \mathrm{V} = \sum_{i=1}^{m} (-1)^{(m+1)} \frac{(V - I)^m}{m}.$$

Finally, we conducted additional experiments to evaluate the computational cost and memory consumption of computing CL and NCL losses. In this set of experiments, we ensure that all methods use exactly the same training configuration (i.e., batch size, model, CUDA version). The hardware used to evaluate memory consumption is NVIDIA A40. We report the results in the table

below. In the Table below, we report the relative memory consumption between NCL and CL: $\eta_{memory} = \frac{\text{memory consumption of NCL}}{\text{memory consumption of CL}}$:

Batch Size	1	2	3	4	5
η_{memory}	0.72	0.57	0.51	0.51	CL out of memory

Table 9: Memory consumption for computing CL and NCL losses with different batch sizes.