Retrieve-then-Adapt: Test-Time Retrieval-Augmentation for Sequential Recommendation

Xing Tang*
School of Artificial Intelligence
Shenzhen Technology University
Shenzhen, China

Jingyang Bin*
Shenzhen Technology University
Shenzhen, China

Ziqiang Cui*
City University of Hong Kong
Hong Kong, Hong Kong SAR

Xiaokun Zhang
City University of Hong Kong
Hong Kong, Hong Kong SAR

Fuyuan Lyu
McGill University
Montreal, Canada

Jingyan Jiang
Shenzhen Technology University
Shenzhen, China

Dugang Liu
Shenzhen University
Shenzhen, China

Chen Ma
City University of Hong Kong
Hong Kong, Hong Kong SAR

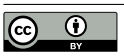
Ruiming Tang
Kuaishou Technology
Beijing, China

Xiuqiang He†
Shenzhen Technology University
Shenzhen, China

Abstract

The sequential recommendation (SR) task aims to predict the next item based on users' historical interaction sequences. Typically trained on historical data, SR models are frozen after deployment, limiting their ability to handle test-time patterns that are rare during training. Existing approaches to address this limitation include test-time training, test-time augmentation, and retrieval-augmented fine-tuning. However, these methods either introduce significant computational overhead, rely on random augmentation strategies, or require a carefully designed two-stage training paradigm. In this paper, we argue that the key to effective test-time augmentation lies in achieving both effective retrieval and efficient fusion. To this end, we propose Retrieve-then-Adapt (**ReAd**), a novel framework that augments a deployed SR model through retrieved user preference signals. Specifically, given a trained SR model, ReAd first retrieves collaboratively similar items for a test user from a constructed collaborative memory. A lightweight retrieval learning module then integrates these items into an informative augmentation embedding that captures both collaborative signals and prediction-refinement cues. Finally, the initial SR prediction is refined via a fusion mechanism that incorporates this embedding. Extensive experiments across five benchmark datasets demonstrate that ReAd consistently outperforms existing SR methods. The source code is publicly available at https://github.com/xingt-tang/ReAd.

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



CIKM '26, Rome, Italy

ACM ISBN 979-8-4007-2539-5/2026/11
https://doi.org/10.1145/3799682.3841085

CCS Concepts

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

Keywords

Retrieval-Augmented, Test-time Retrieval Augmentation, Sequential Recommendation

ACM Reference Format:
Xing Tang, Jingyang Bin, Ziqiang Cui, Xiaokun Zhang, Fuyuan Lyu, Jingyan Jiang, Dugang Liu, Chen Ma, Ruiming Tang, and Xiuqiang He. 2026. Retrieve-then-Adapt: Test-Time Retrieval-Augmentation for Sequential Recommendation. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management (CIKM '26), November 07–11, 2026, Rome, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3799682.3841085

1 Introduction

In recent years, sequential recommendation (SR) has attracted considerable attention for its ability to learn user preferences from sequential behavioral data [41, 44]. Early SR research primarily focused on designing increasingly sophisticated architectures to capture implicit user patterns from historical interactions [11, 13, 26, 27, 31, 40]. However, due to the limited number of interactions between users and items, training SR models often suffers from data sparsity, making it challenging to learn high-quality user representations [20]. To mitigate this issue, some recent studies have turned to self-supervised learning (SSL), leveraging data augmentation and contrastive loss to align different augmented views of the data [5]. With the success of contrastive learning, SR models can now incorporate additional knowledge during training, thereby enhancing the quality of user representations.

Despite efforts to train the SR model, improving its performance at test time or the inference stage remains an open challenge. As illustrated in Figure 1, deployed SR models rely on frozen parameters inherited from training. This static representation poses challenges when test-time patterns differ from those seen during training. First,

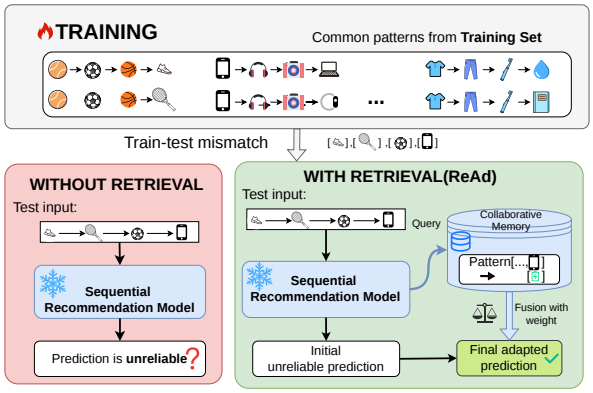


Figure 1: Illustration of the training-test mismatch in sequential recommendation. The training set contains common patterns (e.g., sports or electronics sequences). However, the test input may contain items or combinations that are rare in the training data. Without retrieval (left), the frozen model produces unreliable predictions. With ReAd (right), the test sequence queries a collaborative memory, retrieves relevant patterns, and fuses them with the initial prediction to produce an adapted output.

training data is often dominated by popular items and frequent interaction patterns, while test-time sequences may contain long-tail items or rare combinations [37, 42]. This mismatch between training and test distributions is a fundamental challenge for frozen models. For instance, consider the example in Figure 1: during training, the model frequently encounters patterns such as `[tennis racket, soccer ball, basketball] -> sneakers`. However, at test time, the input sequence `[sneakers, tennis racket, soccer ball, smartphone]` contains a smartphone, which rarely appears in such sports-related combinations in the training set. Second, trained models encode collaborative signals into their parameters, which may be insufficient for rare patterns. In particular, interactions involving long-tail items often lack adequate representation and are easily dominated by popular items. When encountering a test-time sequence, SR models rely heavily on fixed parameters that encode historical collaborative signals. However, these signals may not generalize to long-tail items or rare combinations, leading to a mismatch between training and test distributions. Together, these challenges highlight a fundamental issue in SR, how to help a frozen model generalize to test-time patterns that are rare or absent in training.

Recent studies have explored test-time learning paradigms to improve sequential recommendation during inference. Test-time training has been introduced into recommendation systems [37, 38, 42], leveraging an auxiliary task to facilitate real-time model updating. However, as SR architectures become increasingly complex [2, 39, 41], the computational overhead of such online updates can no longer be overlooked during deployment. Alternatively, lightweight test-time augmentation methods have been explored, which enhance sequential recommendation by augmenting input sequences at inference time and aggregating predictions without introducing additional tasks or structures [6, 12]. Yet, because it relies on random augmentation operators and simple averaging, this method often lacks robustness and may fail to generalize effectively. Another line of work addresses training-test distribution mismatch through retrieval-augmented fine-tuning, which accelerates adaptation by combining recommendation and retrieval learning during pre-training [43]. This approach, however, requires a carefully structured two-stage training pipeline that jointly incorporates both objectives. Overall, there remains a clear need for a low-cost, efficient, and model-agnostic approach that effectively enhances sequential recommendation performance at test time.

In this paper, we propose a novel framework, **Re**trieve-then-**Ad**apt (ReAd), to address the challenges outlined above. Unlike prior test-time adaptation methods that update parameters or rely on random augmentation, ReAd performs test-time retrieval augmentation while keeping the SR model completely frozen. Designing such a retrieval-augmented framework for sequential recommendation presents two main challenges. First, unlike language models [8] or vision-language models [15], SR lacks access to external knowledge sources for retrieval-based enhancement. Defining a suitable knowledge base is thus crucial. We must identify what knowledge is essential and establish a feasible retrieval mechanism. As noted earlier, collaborative signals encoded within model parameters are often weak, especially for long-tail items. To this end, we construct a collaborative memory from the training data that maps sequence representations to their next items. This design enables the retrieval of relevant items based on historically similar behavior sequences, thereby augmenting predictions explicitly at test time—without relying solely on implicit parametric knowledge. The second challenge lies in effectively integrating the retrieved items. Similar to retrieval-augmented generation [8], which retrieves multiple documents per query, our method also retrieves several items per query. To enable efficient augmentation for each test sample, we aim to refine predictions directly from these retrieved items without incurring high computational costs. To this end, we introduce a lightweight retrieval learning module that fuses the retrieved items into a unified representation via cross-attention, coupled with an entropy-based fusion mechanism that dynamically balances the initial prediction and the retrieved signal.

We summarize our major contributions as follows:

- We systematically study test-time retrieval augmentation for sequential recommendation, characterizing when and why it helps and revealing that its effectiveness correlates with interaction density and item popularity.
- We propose ReAd, a lightweight framework that retrieves relevant patterns from a collaborative memory at test time and fuses them with frozen-model predictions, without parameter updates or retraining.
- Extensive experiments are conducted on five public datasets. The experimental results demonstrate the effectiveness and efficiency of the proposed method.

The remainder of this paper is structured as follows. Section 2 reviews related work in sequential recommendation, test-time learning paradigms, and retrieval-augmented recommendation. Section 3 introduces the preliminary definitions and task formulation for sequential recommendation. Our proposed ReAd framework is presented in detail in Section 4, followed by experimental results and

analysis in Section 5. Finally, Section 6 concludes the paper and outlines potential directions for future work.

2 Related Work

In this section, we provide a brief review of related work. Our study builds upon three key research lines: sequential recommendation, test-time strategies for sequential recommendation, and retrieval-augmented recommendation.

2.1 Sequential Recommendation

The learning paradigm of sequential recommendation (SR) models has progressed from traditional models [10, 24] to deep learning approaches [11, 13, 26, 27, 46]. For instance, GRU4Rec [11] and Caser [27] employed Gated Recurrent Units (GRU) and convolutional networks, respectively, to capture the dynamics of user preferences. Following the success of attention mechanisms in language modeling [29], transformer-based SR models have gained prominence [13, 26]. More recently, inspired by large foundation models in language processing, several studies have explored scaling up the parameter sizes of SR models [39, 41]. Another important research direction addresses data sparsity in sequential recommendation by integrating self-supervised learning (SSL) techniques [18, 23, 34, 45]. A key aspect of these approaches is data augmentation, which has evolved from heuristic operations—such as cropping, masking, and dropout—to model-enhanced strategies, including diffusion models [5, 32] and large language model-based augmentation [4]. Additional efforts focus on improving semantic alignment across augmented views through techniques such as contrastive learning and invariant representation learning [3, 21, 47].

Despite these advances, existing methods primarily operate during training and may struggle with test-time patterns that are rare in the training data, limiting their effectiveness in real-world recommendation scenarios.

2.2 Test-Time Learning Paradigms for SR

Building on the preceding discussion, recent research has increasingly focused on test-time strategies for sequential recommendation, particularly test-time training (TTT) [17] and test-time augmentation (TTA) [25]. For example, TTT4Rec [38] adapts model parameters during inference by leveraging additional real-time data. Following this direction, T^{2}ARec [42] and PCRec [33] further explore ways to update trained models at test time. T^{2}ARec introduces a state-space model with two alignment modules to capture shifts in user interest distributions, while PCRec incorporates real-time hidden-state inference and performs one-step optimization during deployment. Despite their effectiveness, these approaches require specialized architectures and incur non-negligible computational overhead during inference. In parallel, TTA-based operators such as TNoise and TMask [6] investigate data augmentation applied directly at test time. While these enhancements yield performance gains over baseline models, they remain susceptible to failure due to the inherent randomness of augmentation and potential misalignment with model architecture.

2.3 Retrieval-Augmented Recommendation

Retrieval-Augmented Generation (RAG) has gained prominence in large language models (LLMs) for its ability to incorporate external knowledge from structured databases [8, 9]. Inspired by this paradigm, several works have introduced retrieval-augmented strategies into recommendation systems to enhance performance [1, 4, 28, 30, 35, 43]. These approaches retrieve external knowledge from diverse sources, such as LLM-generated content [4, 30], cross-domain information [28], or historical user behavior [43]. Among these, RaSeRec [43] is particularly relevant to our work. It proposes a two-stage framework involving pre-training and fine-tuning to mitigate training-test mismatch. However, like other similar methods, it operates primarily during the training phase and cannot adapt dynamically during inference without retraining, thus limiting its applicability in real-time recommendation settings.

To better understand the differences among these paradigms, Table 1 summarizes the key characteristics of test-time training (TTT4Rec), test-time augmentation (TTA), training-time retrieval (RaSeRec), and our proposed ReAd.

3 Preliminaries

In this section, we first introduce the standard sequential recommendation model, followed by the formal problem formulation for test-time retrieval augmentation.

3.1 Sequential Recommendation Model

We begin with the notation used in sequential recommendation. The goal of sequential recommendation is to predict the next item a user is likely to interact with based on their historical behavior sequence. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ denote the set of users and $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ denote the set of items, where $|\mathcal{U}|$ and $|\mathcal{V}|$ are the number of users and items, respectively. For each user $u \in \mathcal{U}$, the historical interaction sequence in chronological order is represented as $\mathbf{s}^u = \{v_1^u, v_2^u, \ldots, v_{|\mathbf{s}^u|}^u\}$, where $v_t^u \in \mathcal{V}$ is the item that user u interacted with at time step t. Given $\mathbf{s}^u$, a sequential recommendation model M is trained to maximize the likelihood of the next item:

$$\arg\max_{v^* \in \mathcal{V}} p\left(v_{|\mathbf{s}|+1} = v^* \mid \mathbf{s}^u\right), \tag{1}$$

where $p(\cdot \mid \mathbf{s}^u)$ denotes the output probability distribution of the model, representing the likelihood of candidate items given the user's historical sequence.

3.2 Problem Formulation

At test time, our objective is to augment the trained sequential model M to improve next-item prediction for a given user u_t and her input sequence $\mathbf{s}^{u_t}$, while leveraging item embeddings $\{\mathbf{e}_j \mid j \in \mathcal{V}\}$. To enable retrieval-based augmentation, a key subproblem is to construct an indexed collaborative memory $\mathcal{D}$ that supports efficient lookup of relevant items. Formally, the augmentation step can be expressed as:

$$f_{\text{aug}} : (\mathcal{D}, M; \mathbf{s}^{u_t}) \longrightarrow \mathbf{e}_{\text{aug}}, \tag{2}$$

where f_{aug} is a retrieval-augmentation function that maps the memory $\mathcal{D}$ and the input sequence $\mathbf{s}^{u_t}$ to an augmented embedding $\mathbf{e}_{\text{aug}}$.

Method	Test-time operation	Retrieval enhanced	Test-time param update	Requires retraining	Plug-and-play	Computational overhead
TTT4Rec	✓	✗	✓	✗	✗	High
TTA	✓	✗	✗	✗	✓	Low
RaSeRec	✗	✓	✗	✓	✗	High
ReAd (Ours)	✓	✓	✗	✗	✓	Low

Table 1: Learning paradigms for sequential recommendation: test-time training, test-time augmentation, training-time retrieval, and ReAd.

Based on this augmented representation, the final prediction refinement follows:

$$v^* = g_{\text{refine}}\Big(p_{\text{init}}, p_{\text{aug}}\Big), \tag{3}$$

where g_{refine} denotes the refinement operator applied at test time, and v^* is the final predicted item based on the initial prediction p_{init} and the augmented prediction p_{aug}.

Note that during test-time inference, the original training data is not accessible, and the forward computation through M is assumed to be computationally efficient. The core contributions of this paper focus on designing effective instantiations of the two functions f_{aug} and g_{refine}.

4 Methodology

In this section, we present the proposed **ReAd** (Retrieve-then-Adapt) framework in detail. ReAd is designed as a test-time retrieval augmentation method. It does not modify the pre-trained SR model's parameters. Instead, it constructs an offline collaborative memory of training patterns and retrieves from it at test time to augment the model's prediction. We begin with a high-level overview of its workflow, followed by a step-by-step explanation of the retrieval-augmentation and refinement mechanisms in subsequent subsections.

4.1 The Overview of ReAd

Figure 2 presents the overall framework of the proposed ReAd method. The framework operates in two main steps: offline preparation (completed once before deployment) and online inference (forward pass only, no gradient updates).

In the offline stage (Section 4.2), we construct a collaborative memory $\mathcal{D}$ from the training data. To effectively integrate the retrieved candidate items, we design a retrieval learning module that processes candidate item embeddings. Notably, the parameters of the trained sequential model remain fixed during this stage, ensuring that the retrieval process remains lightweight and does not introduce additional training overhead.

During the online inference stage (Section 4.3), for a given test user sequence, the retrieval module selects the top-k most relevant items from $\mathcal{D}$ based on sequence similarity and fuses their embeddings into a retrieved representation. The parameters of the SR model remain frozen throughout; no gradient updates are performed at test time. The retrieved representation is then used to refine the model's initial prediction, ultimately yielding the final augmented prediction through a learnable fusion mechanism.

4.2 Collaborative Memory Construction

During the training stage, the sequential recommendation model M is optimized using historical user behavior data to learn and encode collaborative signals. To provide necessary context for the subsequent adaptation process, we first outline the standard training procedure.

4.2.1 Model Training. We construct a trainable item embedding matrix $M_{\mathbf{E}} \in \mathbb{R}^{|\mathcal{V}|\times d}$ for the entire item set $\mathcal{V}$, mapping each item to a d-dimensional dense vector. During training, each user sequence $\mathbf{s}^u$ is split into subsequence–label pairs $(\{v_1^u, \cdots, v_j^u\}, v_{j+1}^u)$, where the latter denotes the item to be predicted. Given a subsequence $\{v_1^u, \cdots, v_j^u\}$, the sequential model M first computes its representation $\mathbf{h}_j^u \in \mathbb{R}^{1\times d}$ via the representation encoder, denoted M_θ. The similarities between $\mathbf{h}_j^u$ and all item embeddings are then obtained as:

$$\mathbf{r} = \mathbf{h}_j^u M_{\mathbf{E}}^\top, \tag{4}$$

where $\mathbf{r} \in \mathbb{R}^{|\mathcal{V}|}$, and the value of each entry r_i indicates the unnormalized affinity between the sequence representation and item v_i, and the ranking of r_i determines the predicted order of candidate items. While inference relies on the top-k ranking derived from $\mathbf{r}$, training optimizes the negative log-likelihood over the full item set via softmax:

$$\mathcal{L}_{\text{train}} = -\sum_{u\in\mathcal{U}}\sum_{j=1}^{|\mathbf{s}^u|}\log(e^{\mathbf{h}_j^u \mathbf{e}_{v_{j+1}^u}^\top}/\Sigma_{v_i\in\mathcal{V}} e^{\mathbf{h}_j^u \mathbf{e}_{v_i}^\top}). \tag{5}$$

Here, $\mathbf{e}_{v_{j+1}^u}$ denotes the embedding of the target item v_{j+1}^u, and $\mathbf{e}_{v_i}$ represents the embedding of any item in $\mathcal{V}$. The model parameters are updated by minimizing $\mathcal{L}_{\text{train}}$, thereby encoding collaborative signals in a parametric manner.

4.2.2 Collaborative Memory. Based on the trained representation encoder M_θ, we construct a collaborative memory $\mathcal{D}$ that explicitly stores collaborative signals. Specifically, for each training sequence $\mathbf{s}^u$, we compute its representation $\mathbf{h}^u$ using M_θ. The embedding of the corresponding next item $\mathbf{e}_{v_u}$ is associated with $\mathbf{h}^u$ to form a pair $< \mathbf{h}^u, \mathbf{e}_{v_u} >$, which encapsulates a sequential pattern in user behavior. Here, $\mathbf{h}^u$ serves as the index in $\mathcal{D}$, and the collection of such pairs comprises the entries of the memory. Sequences with similar representations $\mathbf{h}$ typically correspond to similar target items, reflecting that collaborative signals are explicitly encoded and accessible in this memory structure.

During inference, given a test user sequence $\mathbf{s}^{u_t}$, we first derive its representation $\mathbf{h}^{u_t}$ via M_θ. This representation is then used to retrieve the top-k most relevant item embeddings from $\mathcal{D}$ based on cosine similarity:

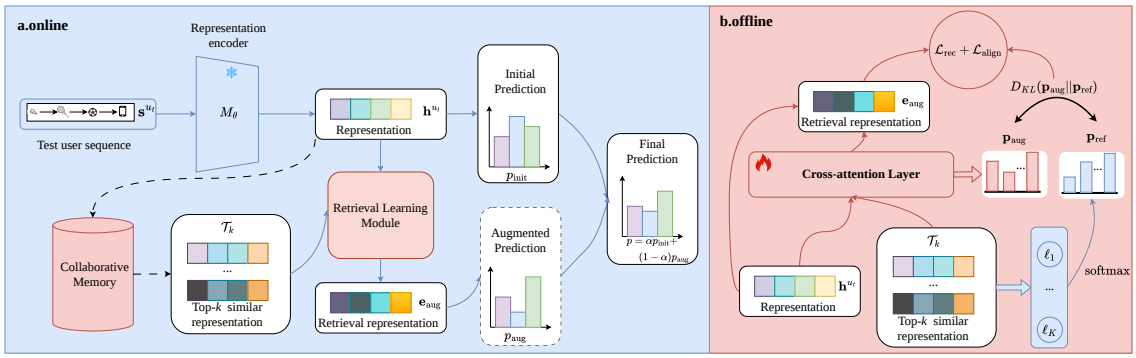


Figure 2: Overview of ReAd. (a) Online inference retrieves collaborative memories and fuses them with the initial prediction. (b) Offline retrieval learning trains cross-attention with recommendation and KL-based alignment losses. Blue and red boxes denote online and offline components, respectively.

$$\mathcal{T}_K = \{\mathbf{e}_k | s_{u_t,k} \in \text{Top}-k(\mathbf{h}^{u_t}, \mathcal{D}, K)\}, \quad (6)$$
$$\text{where} \quad s_{u_t,k} = \cos(\mathbf{h}^{u_t}, \mathbf{h}^{u}) = \frac{\mathbf{h}^{u_t} \cdot \mathbf{h}^{u}}{\|\mathbf{h}^{u_t}\| \|\mathbf{h}^{u}\|},$$

where $\text{Top}-k(.;K)$ is a function that selects the K highest values from the memory regarding the representation as a query. For efficiency, the retrieval process can be accelerated using approximate nearest-neighbor search libraries such as FAISS [7].

4.2.3 Retrieval Learning. To utilize the retrieved item embeddings from $\mathcal{T}_K$, a simple strategy is to average predictions based on Equation 4, which is commonly employed in test-time augmentation methods [6]. While efficient, this strategy introduces increasing noise as K grows. Another straightforward alternative is to manually weight the embeddings, which is both subjective and difficult to generalize.

We propose a lightweight, learning-based weighting scheme, illustrated in the offline part of Figure 2, which requires minimal additional computation during retrieval. Specifically, we treat the sequence representation $\mathbf{h}^{u_t}$ as the query and the retrieved item set $\mathcal{T}_K$ as both the keys and the values. The retrieved embedding is then obtained via cross-attention:

$$\mathbf{e}_{\text{aug}} = \text{softmax}(\frac{QK^\top}{\sqrt{d}})V,$$
$$\text{where} \quad Q = \mathbf{h}^{u_t}\mathbf{W}_q, K = \mathbf{e}_K\mathbf{W}_k, V = \mathbf{e}_K\mathbf{W}_v. \quad (7)$$

Here, $\mathbf{e}_K \in \mathbb{R}^{K\times d}$ stacks the retrieved embeddings, and $\mathbf{W}_q \in \mathbb{R}^{d\times d'}$, $\mathbf{W}_k \in \mathbb{R}^{d\times d'}$ and $\mathbf{W}_v \in \mathbb{R}^{d\times d}$ are learnable projection matrices. The attention distribution over the retrieved items is given by:

$$\mathbf{p}_{\text{aug}} = \text{softmax}(\frac{QK^\top}{\sqrt{d}}), \quad (8)$$

which reflects the similarity between the query and each retrieved item under the learned transformations.

The projection matrices are optimized through two complementary loss terms: a recommendation loss that ensures the augmented representation improves prediction accuracy, and an alignment loss that calibrates the attention weights based on each retrieved item's intrinsic predictive utility.

We define the combined representation as the average of the query representation and the augmented embedding: $\mathbf{h} = \frac{1}{2}(\mathbf{h}^{u_t} + \mathbf{e}_{\text{aug}})$, and train it to better predict the target item $\mathbf{e}_t$. This simple equal weighting suffices, as the subsequent entropy-based fusion mechanism at test time will dynamically adjust the final contribution. The recommendation loss then ensures the augmented representation to improve the prediction:

$$\mathcal{L}_{\text{rec}} = -\log \frac{\exp(\mathbf{h}\mathbf{e}_t^\top)}{\sum_{v_i\in\mathcal{V}} \exp(\mathbf{h}\mathbf{e}_{v_i}^\top)}. \quad (9)$$

While $\mathbf{p}_{\text{aug}}$ in Equation 8 captures similarity between the query and retrieved items, this similarity alone may not reflect an item's true utility for refining the final prediction. For instance, a highly similar popular item could provide redundant information, whereas a moderately similar but discriminative long-tail item might be more beneficial for correcting the prediction. To address this gap, we introduce an alignment loss, which calibrates $\mathbf{p}_{\text{aug}}$ by aligning it with a reference distribution $\mathbf{p}_{\text{ref}}$ that explicitly encodes each retrieved item's predictive usefulness. We first measure how well each retrieved item $\mathbf{e}_k$ independently predicts the target item $\mathbf{e}_t$:

$$\ell(\mathbf{e}_k, \mathbf{e}_t) = -\log \frac{(\exp(\mathbf{e}_k\mathbf{e}_t^\top)}{\sum_{v_i\in\mathcal{V}} \exp(\mathbf{e}_k\mathbf{e}_{v_i}^\top)}. \quad (10)$$

A lower $\ell(\cdot)$ indicates stronger predictive capability. The reference distribution is then defined as:

$$\mathbf{p}_{\text{ref}} = \text{softmax}(\{-\ell(\mathbf{e}_k, \mathbf{e}_t)\}_{k=1}^K), \quad (11)$$

assigning a higher probability to items that are more predictive of the target. The alignment loss aims to pull the attention distribution toward this utility-aware reference. Finally, the alignment loss is defined as the Kullback–Leibler divergence between the attention distribution $\mathbf{p}_{\text{aug}}$ and this utility-aware reference:

$$\mathcal{L}_{\text{align}} = D_{KL}(\mathbf{p}_{\text{aug}} || \mathbf{p}_{\text{ref}}). \quad (12)$$

Minimizing this term encourages the learned attention weights to not only reflect similarity but also align with each item's intrinsic contribution to predicting the target. In effect, the KL-divergence

acts as a calibrator: it adjusts the similarity-based attention toward a distribution that prioritizes items with high predictive value, thereby ensuring that the aggregation step focuses on information that is truly beneficial for the final recommendation.

The overall training objective combines the recommendation loss and the alignment loss:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \mathcal{L}_{\text{align}}, \tag{13}$$

where the λ balances two terms. Notice that this design ensures the retrieval-learning module satisfies the following two complementary criteria:

(1) Preserving collaborative signals through the attention mechanism. The cross-attention module naturally captures the similarity between the query (user sequence representation) and each retrieved item. This similarity reflects historical co-occurrence and interaction patterns encoded in the training data—i.e., the collaborative signals.
(2) Emphasizing predictive utility via the KL-driven alignment with $\mathbf{p}_{\text{ref}}$. While similarity provides a useful prior, it does not guarantee that an item will help refine the current prediction. The KL-divergence term introduces a helpful objective: it aligns the attention distribution with a reference distribution that directly measures each retrieved item's ability to individually predict the target. This alignment effectively re-weights the retrieved items, amplifying those that are not only similar but also predictively discriminative for the specific target item.

Notice that the extra computation in our retrieval learning is marginal, requiring only the optimization of three matrices using the training data, and is model-agnostic, requiring no architectural assumptions about the underlying sequential recommendation model.

4.3 Test-Time Retrieval Augmentation

At test time, the SR model parameters remain frozen. The retrieval and fusion process operates entirely in forward pass, requiring no gradient computation or parameter updates. Once the augmented embedding $\mathbf{e}_{\text{aug}}$ is obtained, we first compute the corresponding augmentation-based prediction probability p_{aug} via the same similarity computation used in Equation 4. Specifically:

$$p_{\text{aug}}(v = v^* \mid \mathbf{s}^{u_t}) = \frac{\exp\left(\mathbf{e}_{\text{aug}} \mathbf{e}_{v^*}^\top\right)}{\sum_{v_i \in \mathcal{V}} \exp\left(\mathbf{e}_{\text{aug}} \mathbf{e}_{v_i}^\top\right)}. \tag{14}$$

With both the initial and augmentation predictions available, the next step is to refine the final prediction by effectively combining these two signals, as formulated in Equation 3. We implement this refinement as follows:

$$\begin{aligned} p\big(v = v^* \mid \mathbf{s}^{u_t}\big) = \alpha \times p_{\text{init}}\big(v = v^* \mid \mathbf{s}^{u_t}\big) \\ + (1 - \alpha)\, p_{\text{aug}}\big(v = v^* \mid \mathbf{s}^{u_t}\big), \end{aligned} \tag{15}$$

where p_{init} is the initial prediction and p_{aug} is the augmentation-based prediction, respectively. To adaptively determine the mixing coefficient α, we propose an entropy-based dynamic fusion mechanism that weights each prediction based on its uncertainty across the entire item set.

4.3.1 Uncertainty Estimation via Entropy. We quantify the uncertainty of a prediction distribution by its entropy. For a probability vector $\mathbf{p} = [p_{v_1}, \cdots, p_{v_{|\mathcal{V}|}}]$ obtained via softmax as Equation 14, the entropy is computed as:

$$H(\mathbf{p}) = -\sum_{v_i \in \mathcal{V}} p_{v_i} \log(p_{v_i} + \epsilon), \tag{16}$$

where $\epsilon = 10^{-8}$ ensures numerical stability. Because the item set is typically large and follows a long-tail distribution, computing entropy over all items can dilute the discriminative signal. To focus on the most plausible candidates, we restrict the entropy calculation to the top-ρ items with the highest logit scores:

$$H_{\text{top}}(\mathbf{p}) = -\sum_{v_i \in \tau_{\text{top}}(\mathbf{p})} p_{v_i} \log\big(p_{v_i} + \epsilon\big), \tag{17}$$

where $\tau_{\text{top}}(\mathbf{p})$ denotes the set of items whose logits rank in the top ρ fraction. This truncation serves two purposes: (1) it concentrates the uncertainty measure on the region that matters most for ranking, and (2) it amplifies the relative difference in entropy between the two predictions, making the fusion weights more sensitive to genuine confidence variations.

4.3.2 Confidence-Driven Fusion Weight. Let $H_{\text{top}}^{\text{init}}$ and $H_{\text{top}}^{\text{aug}}$ be the truncated entropies of the initial and augmented predictions, respectively. the fusion weight α is computed as:

$$\alpha = \frac{\exp\left(\frac{1}{1+H_{\text{top}}^{\text{init}}}\right)}{\exp\left(\frac{1}{1+H_{\text{top}}^{\text{init}}}\right) + \exp\left(\frac{1}{1+H_{\text{top}}^{\text{aug}}}\right)}. \tag{18}$$

The rationale behind this formulation is that lower entropy corresponds to higher prediction confidence. When a model outputs a sharply peaked distribution (low H), it expresses strong discriminative certainty among the top candidates, and should therefore receive a larger weight in the fusion. Conversely, a flat distribution (high H) reflects uncertainty and is assigned a smaller weight.

This entropy-guided weighting scheme enables ReAd to dynamically balance the contributions of the original sequential representation and the retrieval-augmented signal, seamlessly adapting to the varying confidence levels across different user sequences at test time.

5 Experiments

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

- **RQ1**: Does ReAd consistently outperform existing sequential recommendation baselines?
- **RQ2**: How well does ReAd generalize across different backbone SR architectures?
- **RQ3**: How do key hyperparameters in the retrieval learning module, i.e., the number of retrieved items K and the fusion weight λ, affect the performance of ReAd?
- **RQ4**: What is the contribution of each core component in ReAd, and is the introduced computational overhead acceptable for real-time inference?
- **RQ5**: Does ReAd indeed retrieve semantically relevant and helpful items during test-time adaptation?

5.1 Experimental Settings

5.1.1 Datasets. We evaluate our method using five public datasets that represent diverse recommendation scenarios, which are widely adopted in previous sequential recommendation research. Four Amazon subsets—Office, Beauty, Sports, and Home—span distinct product categories and exhibit varied interaction sparsity, reflecting real-world e-commerce environments. The ML-1M dataset provides a widely adopted benchmark in the movie domain with denser user activity. Following common practice in sequential recommendation [6, 23], we filter out users and items with fewer than five interactions to ensure data quality and mitigate extreme sparsity. Table 2 summarizes the resulting dataset statistics.

Table 2: Statistics of the datasets used in experiments.

Statistic	Office	Beauty	Sport	Home	ML-1M
#Users	4,906	22,364	35,599	66,520	6,040
#Items	2,421	12,102	18,358	28,238	3,706
#Interactions	53,258	198,502	296,337	551,682	1,000,209
Avg. Actions	10.86	8.88	8.32	8.29	165.60
Sparsity	99.55%	99.93%	99.95%	99.97%	95.53%

5.1.2 Baselines. To conduct a comprehensive evaluation, we include twelve baselines that fall into three categories as outlined in Section 2. The first group comprises architectural models designed to capture sequential patterns, including **GRU4Rec** [11], **SASRec** [13], and **BERT4Rec** [26].

The second group consists of contrastive learning-based methods that address data sparsity through augmentation and alignment. **CL4SRec** [34] and **DuoRec** [23] use sequence augmentation strategies. **CoSeRec** [18] and **MCLRec** [21] further improve augmentation and alignment. **S^3-Rec** [45], **ICLRec** [3], and **ICSRec** [22] instead align different sequence views to improve representation learning.

The third group covers test-time and retrieval-augmented enhancement methods: **TTA** [6] applies random perturbations at inference time, while **RaSeRec** [43] incorporates retrieval during training but is included as a strong retrieval-based baseline.

5.1.3 Evaluation Metrics. To ensure an unbiased evaluation, we adopt the leave-one-out strategy to split each user's interaction sequence into training, validation, and test segments. During evaluation, we rank all items in the catalog for every test sequence and compute metrics over the full ranking list, avoiding the potential bias introduced by negative-item sampling. We employ two widely used ranking-based metrics: Hit Ratio@K (HR@K) and Normalized Discounted Cumulative Gain@K (ND@K) with $K \in \{5, 10, 20\}$. Higher values of both metrics indicate better ranking performance.

5.1.4 Implementation Details. For all baseline models, we use the open-source implementations provided in RecBole [36] under their recommended settings. Our proposed ReAd framework is intentionally model-agnostic. To demonstrate this property, we instantiate it on representative models from both the architectural group (SASRec) and the contrastive-learning group (DuoRec), thereby showing that ReAd can be flexibly combined with different types of sequential recommendation backbones.

We set the embedding dimension to 64 and the batch size to 256 across all experiments. All other hyperparameters for the baselines are carefully tuned following the configurations described in their original papers. Training is performed with the Adam optimizer [14] using a learning rate of 0.001 The three related hyperparameters in our method are the K values in Equation 6 and λ in Equation 13, which balance the recommendation and alignment objectives. The top ratio in Equation 17, which controls the fraction of items considered in the entropy-based fusion. We hence search $K \in \{1, 3, 5, 10, 15, 20\}$, $\lambda \in \{0, 0.5, 1, 1.5, 2\}$, and top ratio in $\rho \in \{10\%, 5\%, 1\%, 0.5\%, 0.1\%\}$. All experiments are conducted on an NVIDIA GeForce RTX 4090D and run for ten runs, reporting the average results for all methods.

5.2 Overall Performance (RQ1)

We compare our ReAd with the mentioned baselines, and the overall results are illustrated in Table 3. We make several observations as follows.

- When built upon DuoRec as the backbone, ReAd consistently achieves the best results across all five datasets, demonstrating its effectiveness. The performance gap between ReAd (+DuoRec) and both ReAd (+SASRec) and the test-time baselines (TTA and RaSeRec) also highlights the benefit of incorporating contrastive learning during training. Notably, RaSeRec—which introduces retrieval through a dedicated pre-training architecture but does not use contrastive learning—performs comparably to TTA, while ReAd (+DuoRec) outperforms both. This confirms that ReAd is model-agnostic and can flexibly leverage different training paradigms (e.g., contrastive learning) to further boost performance.
- Compared with the original SASRec, both TTA and ReAd (+SASRec) improve the performance, confirming the value of test-time augmentation. However, ReAd (+SASRec) consistently surpasses TTA, indicating that our retrieval-augmented refinement, which integrates both retrieval-based augmentation and confidence-aware prediction fusion, is more effective than TTA's purely input-level augmentation.
- This pattern suggests that test-time retrieval augmentation is particularly beneficial when historical signals are sparse, where the frozen model alone is insufficient. Both RaSeRec and ReAd, which employ retrieval mechanisms, show greater improvements on sparse data, reinforcing the importance of augmenting sparse sequences with externally retrieved collaborative signals.

5.3 Generalize to Other Architectures (RQ2)

To further verify the generalization capability of ReAd, we apply it to SR models with different architectural designs: SASRec and DuoRec use causal (unidirectional) self-attention; BERT4Rec employs regular (bidirectional) self-attention; and GRU4Rec relies on gated recurrent units [19]. The results are summarized in Table 4, along with the earlier SASRec results in Table 3.

ReAd consistently improves recommendation performance across all tested architectures, confirming its model-agnostic design. Moreover, the magnitude of improvement varies, with most gains exceeding 10%, although a few remain modest. This is attributed to the inherent capacity of the backbone model: stronger base

Table 3: Performance on five public datasets. Bold and underlined values denote the best and second-best results; ReAd's gains are significant under paired t-tests ($p < 0.01$).

Dataset	Metric	GRU4Rec	SASRec	BERT4Rec	S^3-Rec	CL4SRec	CoSeRec	ICLRec	DuoRec	MCLRec	ICSRec	RaSeRec	TTA	ReAd (+SASRec)	ReAd (+DuoRec)
Office	HR@5	0.0277	0.0544	0.0376	0.0443	0.0480	0.0577	0.0564	0.0644	0.0671	0.0651	0.0669	0.0548	0.0585	**0.0698**
	HR@10	0.0532	0.0899	0.0666	0.0658	0.0665	0.0811	0.0815	0.1011	0.1071	0.1051	0.1042	0.0896	0.0907	**0.1090**
	HR@20	0.0985	0.1388	0.1166	0.1221	0.1121	0.1346	0.1195	0.1552	0.1648	0.1576	0.1641	0.1389	0.1427	**0.1668**
	ND@5	0.0169	0.0369	0.0233	0.0296	0.0308	0.0394	0.0345	0.0410	0.0407	0.0416	0.0429	0.0371	0.0384	**0.0456**
	ND@10	0.0252	0.0483	0.0326	0.0383	0.0385	0.0500	0.0439	0.0527	0.0538	0.0525	0.0549	0.0492	0.0489	**0.0582**
	ND@20	0.0365	0.0605	0.0452	0.0523	0.0502	0.0611	0.0534	0.0663	0.0691	0.0679	0.0699	0.0625	0.0620	**0.0727**
Beauty	HR@5	0.0206	0.0377	0.0340	0.0395	0.0471	0.0506	0.0498	0.0538	0.0563	0.0540	0.0570	0.0416	0.0447	**0.0582**
	HR@10	0.0325	0.0598	0.0526	0.0619	0.0654	0.0726	0.0741	0.0825	0.0869	0.0841	0.0865	0.0629	0.0687	**0.0874**
	HR@20	0.0499	0.0877	0.0789	0.0937	0.0985	0.1031	0.1059	0.1184	0.1212	0.1198	0.1221	0.0921	0.0967	**0.1243**
	ND@5	0.0127	0.0237	0.0222	0.0251	0.0273	0.0339	0.0331	0.0340	0.0344	0.0338	0.0369	0.0266	0.0290	**0.0386**
	ND@10	0.0166	0.0308	0.0282	0.0323	0.0350	0.0410	0.0405	0.0433	0.0446	0.0435	0.0461	0.0329	0.0367	**0.0481**
	ND@20	0.0209	0.0378	0.0349	0.0403	0.0410	0.0488	0.0488	0.0523	0.0539	0.0525	0.0554	0.0419	0.0438	**0.0573**
Sport	HR@5	0.0107	0.0216	0.0170	0.0220	0.0256	0.0285	0.0291	0.0310	0.0322	0.0316	0.0315	0.0239	0.0233	**0.0331**
	HR@10	0.0178	0.0326	0.0281	0.0336	0.0382	0.0426	0.0429	0.0471	0.0486	0.0479	0.0487	0.0398	0.0339	**0.0496**
	HR@20	0.0279	0.0479	0.0444	0.0510	0.0553	0.0636	0.0639	0.0692	0.0720	0.0712	0.0711	0.0637	0.0500	**0.0726**
	ND@5	0.0068	0.0148	0.0109	0.0147	0.0150	0.0179	0.0181	0.0193	0.0204	0.0202	0.0196	0.0168	0.0163	**0.0221**
	ND@10	0.0091	0.0184	0.0144	0.0185	0.0217	0.0231	0.0238	0.0245	0.0260	0.0245	0.0251	0.0214	0.0197	**0.0274**
	ND@20	0.0116	0.0222	0.0185	0.0229	0.0244	0.0275	0.0286	0.0300	0.0311	0.0301	0.0308	0.0266	0.0237	**0.0332**
Home	HR@5	0.0055	0.0081	0.0083	0.0103	0.0136	0.0153	0.0146	0.0190	0.0198	0.0198	0.0196	0.0108	0.0105	**0.0203**
	HR@10	0.0104	0.0131	0.0143	0.0155	0.0198	0.0232	0.0225	0.0278	0.0286	0.0289	0.0293	0.0175	0.0164	**0.0298**
	HR@20	0.0180	0.0204	0.0241	0.0260	0.0282	0.0336	0.0345	0.0402	0.0419	0.0418	0.0409	0.0266	0.0248	**0.0426**
	ND@5	0.0034	0.0052	0.0052	0.0074	0.0079	0.0107	0.0101	0.0117	0.0122	0.0128	0.0121	0.0069	0.0071	**0.0133**
	ND@10	0.0049	0.0068	0.0072	0.0092	0.0100	0.0133	0.0133	0.0145	0.0150	0.0152	0.0152	0.0091	0.0090	**0.0163**
	ND@20	0.0068	0.0086	0.0096	0.0115	0.0121	0.0159	0.0161	0.0176	0.0185	0.0179	0.0181	0.0113	0.0111	**0.0196**
ML-1M	HR@5	0.0462	0.1364	0.1364	0.1192	0.1163	0.1117	0.1312	0.1909	0.1889	0.1913	0.1932	0.1359	0.1644	**0.1956**
	HR@10	0.0654	0.2119	0.2156	0.2079	0.2006	0.1878	0.2196	0.2859	0.2832	0.2799	0.2844	0.2136	0.2382	**0.2897**
	HR@20	0.0980	0.3144	0.3164	0.3181	0.3121	0.2997	0.3357	0.3839	0.3796	0.3844	0.3879	0.3175	0.3404	**0.3897**
	ND@5	0.0299	0.0894	0.0902	0.0748	0.0738	0.0693	0.0844	0.1297	0.1288	0.1287	0.1328	0.0814	0.1080	**0.1341**
	ND@10	0.0360	0.1135	0.1156	0.1120	0.0989	0.0981	0.1116	0.1603	0.1600	0.1588	0.1646	0.1105	0.1317	**0.1653**
	ND@20	0.0442	0.1393	0.1410	0.1347	0.1289	0.1269	0.1371	0.1850	0.1833	0.1851	0.1888	0.1372	0.1575	**0.1907**

Table 4: ReAd improvements on different backbones (all $p < 0.01$).

Dataset	Metric	GRU4Rec			BERT4Rec		
		Orig	ReAd	Imp.	Orig	ReAd	Imp.
Office	HR@5	0.0277	0.0320	15.52%	0.0376	0.0437	16.22%
	HR@10	0.0532	0.0595	11.84%	0.0666	0.0742	11.41%
	ND@5	0.0169	0.0199	17.75%	0.0233	0.0271	16.31%
	ND@10	0.0252	0.0288	14.29%	0.0326	0.0369	13.19%
Beauty	HR@5	0.0206	0.0225	9.22%	0.0340	0.0372	9.41%
	HR@10	0.0325	0.0344	5.85%	0.0526	0.0585	11.22%
	ND@5	0.0127	0.0147	15.75%	0.0222	0.0242	9.01%
	ND@10	0.0166	0.0186	12.05%	0.0282	0.0311	10.28%
Sport	HR@5	0.0107	0.0119	11.21%	0.0170	0.0190	11.76%
	HR@10	0.0178	0.0189	6.18%	0.0281	0.0305	8.54%
	ND@5	0.0068	0.0078	14.71%	0.0109	0.0124	13.76%
	ND@10	0.0091	0.0100	9.89%	0.0144	0.0161	11.81%
Home	HR@5	0.0055	0.0059	7.27%	0.0083	0.0093	12.05%
	HR@10	0.0104	0.0109	4.81%	0.0143	0.0159	11.19%
	ND@5	0.0034	0.0036	5.88%	0.0052	0.0058	11.54%
	ND@10	0.0049	0.0052	6.12%	0.0072	0.0079	9.72%
ML-1M	HR@5	0.0462	0.0505	9.31%	0.1364	0.1412	3.52%
	HR@10	0.0654	0.0733	12.08%	0.2156	0.2243	4.04%
	ND@5	0.0299	0.0338	13.04%	0.0902	0.0929	2.99%
	ND@10	0.0360	0.0409	13.61%	0.1156	0.1196	3.46%

models may leave less room for relative improvement. Additionally, dataset characteristics play a role: the gains are larger on datasets where sequential patterns are sparse or exhibit greater diversity, as test-time retrieval augmentation provides complementary signals that the frozen model lacks. Overall, these experiments demonstrate that ReAd effectively enhances a diverse set of sequential recommendation architectures, validating its robustness and practical applicability.

5.4 Hyperparameter Sensitivity (RQ3)

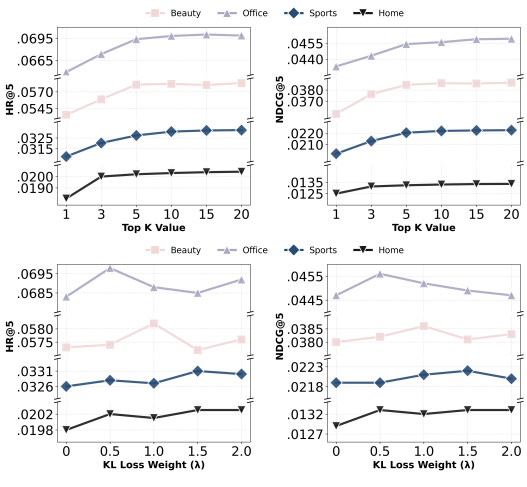


Figure 3: Effects of the number of retrieved items K and KL-alignment coefficient λ.

As discussed in Section 4, ReAd involves three key hyperparameters. The first two are the number of retrieved items K and the alignment-loss coefficient λ in Equation 13, which govern retrieval

learning. As shown in Figure 3 (top), increasing K initially improves performance, peaking at $K = 10$ for most datasets, after which performance gradually declines. This confirms that retrieving too many items introduces irrelevant or noisy signals. The bottom subfigure shows that varying λ has a relatively modest impact, suggesting that the alignment loss plays a supportive role when retrieved items are already collaboratively relevant.

Across different backbone models, performance follows a consistent trend: it initially improves with larger K, peaks at an optimal value, and then declines. This pattern can be explained as follows: a moderate increase in K enriches the augmentation signal with more collaborative information, but beyond a certain point, irrelevant or noisy items are introduced, which hinder the learning process. Therefore, selecting an appropriate K is crucial for achieving the best improvement. In contrast, varying λ does not lead to significant changes in performance. We hypothesize that this is because the retrieved set is relatively small and the items are already collaboratively relevant, so strictly aligning the attention distribution with the reference distribution yields diminishing returns.

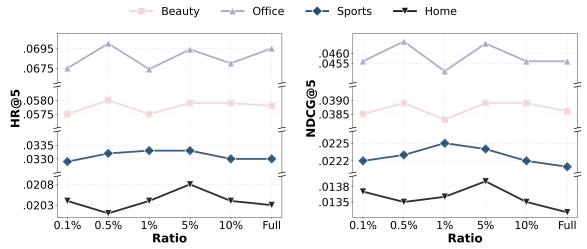


Figure 4: Effect of top-ratio ρ in τ_{top}.

The fraction ρ of top-ranked items considered in τ_{top} directly influences the entropy-based fusion weight α in Equation 3, and consequently the final refined prediction. As shown in Figure 4, a larger top-ratio generally leads to better performance, whereas using the full item set (i.e., top-ratio = 100%) consistently degrades results, though more stable. This can be explained by the long-tail nature of item catalogs: when entropy is computed over all items, the numerous low-probability tail items dilute the discriminative signal, making the entropies of the initial and augmented predictions too similar.

5.5 Ablation Study (RQ4)

Table 5: Ablation results on four Amazon datasets with DuoRec.

Method	HR@10				NDCG@10			
	Beauty	Home	Office	Sports	Beauty	Home	Office	Sports
w/o Att	0.0786	0.0268	0.0901	0.0436	0.0413	0.0127	0.0484	0.0207
w/o KL	0.0863	0.0293	0.1074	0.0489	0.0473	0.0159	0.0571	0.0269
w/o Rec	0.0643	0.0218	0.0752	0.0345	0.0332	0.0095	0.0381	0.0198
w/o α	0.0845	0.0279	0.1044	0.0479	0.0451	0.0149	0.0536	0.0243
ReAd	**0.0874**	**0.0298**	**0.1090**	**0.0496**	**0.0481**	**0.0163**	**0.0582**	**0.0274**

In this section, we conduct ablation studies to investigate the contribution of each component in ReAd. All results are reported with DuoRec as the backbone (i.e., DuoRec+ReAd) on four Amazon datasets. The variants and their results are summarized in Table 5:

- **w/o Rec**: Removes $\mathcal{L}_{\text{rec}}$. The retrieval module is trained only with $\mathcal{L}_{\text{align}}$, without direct supervision to predict the target item.
- **w/o KL**: Removes $\mathcal{L}_{\text{align}}$. The retrieval module is trained only with $\mathcal{L}_{\text{rec}}$, without calibration against predictive utility.
- **w/o Att**: Replaces learnable cross-attention with simple averaging of retrieved embeddings.
- **w/o α**: Fixes $\alpha = 0.5$, disabling entropy-based dynamic fusion.

Based on the results, we make the following observations. First, *w/o Rec* causes the largest drop, confirming that retrieval must be guided by the prediction objective. Second, *w/o Att* underperforms, showing that simple averaging fails to capture the varying predictive utility of retrieved items. Third, *w/o* α also harms performance, underscoring the importance of dynamic fusion. Finally, *w/o KL* shows the smallest drop, as retrieved items are already relevant via similarity.

To directly address the concern of whether retrieval is necessary, we compare ReAd against two ablation strategies: (i) **Random Retrieval**, which randomly samples K items from the item set; and (ii) **Popularity Retrieval**, which always retrieves the K most popular items in the training set. Table 6 summarizes the results. Several observations can be given. First, random retrieval consistently degrades performance across all datasets, confirming that indiscriminate retrieval introduces noise. Second, popularity retrieval shows mixed results: it underperforms ReAd on Beauty and Office, but slightly outperforms on the extremely sparse Home dataset, where popularity bias provides a simple but effective alternative when similarity-based retrieval struggles due to data sparsity. Third, ReAd achieves the best or comparable results across all datasets, demonstrating that similarity-based retrieval is the most robust strategy for test-time augmentation.

Table 6: Retrieval-strategy comparison on four Amazon datasets (best in bold).

Dataset	Metric	SASRec	Random	Popularity	**ReAd**
Beauty	NDCG@10	0.0308	0.0212	0.0226	**0.0367**
	HR@10	0.0598	0.0421	0.0453	**0.0687**
Office	NDCG@10	0.0483	0.0360	0.0378	**0.0489**
	HR@10	0.0899	0.0734	0.0744	**0.0907**
Sports	NDCG@10	0.0184	0.0128	0.0196	**0.0197**
	HR@10	0.0326	0.0224	0.0321	**0.0339**
Home	NDCG@10	0.0068	0.0050	0.0090	0.0090
	HR@10	0.0131	0.0100	0.0163	**0.0164**

5.6 Item Popularity Stratified Analysis (RQ5)

To answer RQ5 and investigate whether ReAd merely retrieves popular items or genuinely supports long-tail recommendations, we conduct a stratified analysis by item popularity. Following prior work [16], we group items into four quartiles (Q1-Q4) by their frequency in the training set, where Q1 contains the least popular (long-tail) items and Q4 the most popular ones. We then evaluate SASRec and ReAd (+SASRec) on each group separately, reporting NDCG@10 as the primary metric.

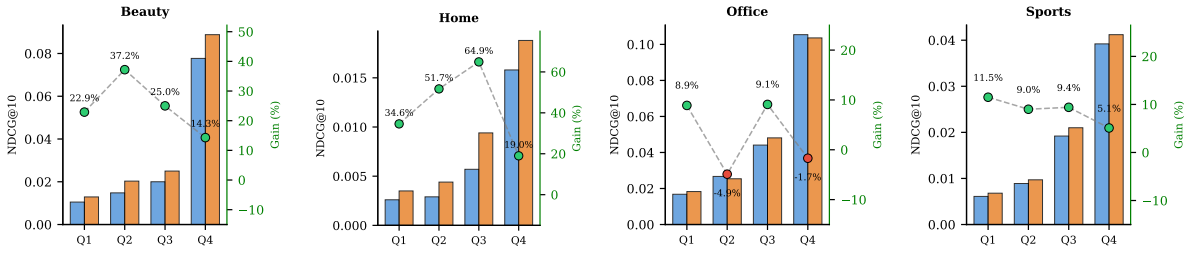


Figure 5: Item popularity stratified results on four Amazon datasets (NDCG@10). Each subplot shows absolute NDCG@10 (bars: SASRec in blue, ReAd in orange) and relative gain (green/red dots with dashed line) across item popularity quartiles (Q1: long-tail, Q4: popular). ReAd achieves substantial improvements on long-tail items (Q1) and moderately popular items (Q2-Q3).

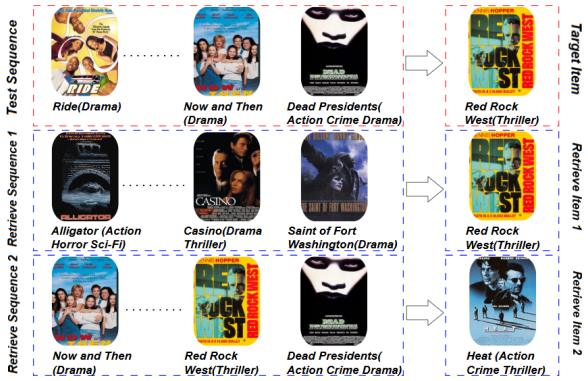


Figure 6: MovieLens case study: red dashed lines denote the test sequence and target item; blue dashed lines denote the two retrieved sequences and corresponding items.

Figure 5 presents item popularity stratified results. ReAd consistently improves over SASRec across all popularity quartiles. Notably, substantial gains are observed on long-tail items (Q1), confirming that ReAd does not merely amplify popular items.

The gains follow an inverted-U shape: they are substantial for Q1-Q3 but diminish for the most popular items (Q4). This is because moderately popular items have sufficient yet not excessive collaborative signals in memory, whereas popular items already benefit from abundant training data. These results reveal a fundamental property: test-time retrieval augmentation is most beneficial when collaborative signals are present but not overwhelming. This characterizes the capability boundary of ReAd and, more broadly, of test-time retrieval augmentation in sequential recommendation.

5.7 Case Study and Efficiency (RQ4&RQ5)

The Figure 6 provides a case study on MovieLens, illustrating how ReAd enhances test-time prediction. The test sequence (red) transitions from drama to thriller.

Retrieved sequences (blue) not only share overlapping items such as *Now and Then*, *Dead Presidents*, and *Red Rock West* — confirming collaborative relevance — but also supply complementary items like *Casino* and *Heat*, which reinforce the emerging thriller preference while bridging earlier drama context (e.g., *Alligator*, *Saint of Fort Washington*).

This example highlights two key advantages of ReAd. First, the model does not rely solely on the most recent interactions; instead, it retrieves sequences that are historically similar and collectively reflect both past and emerging interests. Second, retrieved items are not merely popular or globally similar; they are selectively aligned with the local transition in the test sequence (from drama to thriller), enabling the fusion module to refine predictions toward the actual next item, *Red Rock West*. Thus, ReAd enriches test-time representations by dynamically retrieving collaborative signals.

Meanwhile, we also compare inference efficiency on the ML-1M dataset with a batch size. DuoRec is the fastest (*1.06 ms per batch*), followed by TTA (*1.48 ms*). ReAd incurs moderate overhead (*4.76 ms*), but is substantially faster than RaSeRec (*6.20 ms*) while achieving superior accuracy. This confirms that ReAd offers a favorable trade-off between accuracy and efficiency.

6 Conclusion

In this work, we proposed ReAd, a test-time retrieval augmentation framework for sequential recommendation. To address the challenge of training-test mismatch on sequential patterns, ReAd introduces a collaborative memory to retrieve historically relevant items, a lightweight retrieval learning module that fuses retrieved items into an augmentation embedding, and an entropy-based refinement mechanism that dynamically balances the original prediction with the augmented signal. Extensive experiments on five benchmark datasets demonstrate that ReAd consistently improves recommendation performance across different backbone architectures. Our fine-grained analysis reveals that ReAd's effectiveness correlates with interaction density and item popularity, achieving consistent gains across all popularity levels, with the most pronounced improvements on moderately popular items. Ablation studies validate the contribution of each component, and comparisons of retrieval strategies confirm that similarity-based retrieval is essential. Furthermore, ReAd maintains competitive inference efficiency with moderate overhead. Overall, ReAd provides a practical, model-agnostic solution for enhancing sequential recommendation at test time, with its effectiveness tied to the richness of available collaborative signals.

Acknowledgments

This work was supported by the Shenzhen Technology University School-level project (No.20251061020002).

GenAI Usage Disclosure

We used GenAI to help polish our writing, such as catching grammar mistakes, fixing typos, and improving sentence flow. Aside from this light language editing, we did not use AI for any other part of this paper.

References

[1] Shuqing Bian, Wayne Xin Zhao, Jinpeng Wang, and Ji-Rong Wen. 2022. A relevant and diverse retrieval-enhanced data augmentation framework for sequential recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2923–2932.

[2] Zheng Chai, Qin Ren, Xijun Xiao, Huizhi Yang, Bo Han, Sijun Zhang, Di Chen, Hui Lu, Wenlin Zhao, Lele Yu, et al. 2025. Longer: Scaling up long sequence modeling in industrial recommenders. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 247–256.

[3] Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*. 2172–2182.

[4] Ziqiang Cui, Yunpeng Weng, Xing Tang, Xiaokun Zhang, Dugang Liu, Shiwei Li, Peiyang Liu, Bowei He, Weihong Luo, Xiuqiang He, et al. 2025. Semantic Retrieval Augmented Contrastive Learning for Sequential Recommendation. *arXiv preprint arXiv:2503.04162* (2025).

[5] Ziqiang Cui, Haolun Wu, Bowei He, Ji Cheng, and Chen Ma. 2024. Context Matters: Enhancing Sequential Recommendation with Context-aware Diffusion-based Contrastive Learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (Boise, ID, USA) *(CIKM '24)*. Association for Computing Machinery, New York, NY, USA, 404–414.

[6] Yizhou Dang, Yuting Liu, Enneng Yang, Minhan Huang, Guibing Guo, Jianzhe Zhao, and Xingwei Wang. 2025. Data augmentation as free lunch: Exploring the test-time augmentation for sequential recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1466–1475.

[7] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[8] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting LLMs: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6491–6501.

[9] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* 2, 1 (2023).

[10] 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.

[11] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 843–852.

[12] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. 2023. Empowering Graph Representation Learning with Test-Time Graph Transformation. In *ICLR*.

[13] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation . In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, Los Alamitos, CA, USA, 197–206. doi:10.1109/ICDM.2018.00035

[14] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Youngjun Lee, Doyoung Kim, Junhyeok Kang, Jihwan Bang, Hwanjun Song, and Jae-Gil Lee. 2025. RA-TTA: Retrieval-Augmented Test-Time Adaptation for Vision-Language Models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

[16] Siyi Liu and Yujia Zheng. 2020. Long-tail Session-based Recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 509–514.

[17] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. TTT++: when does self-supervised test-time training fail or thrive?. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 1669, 13 pages.

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

[19] Aleksandr Petrov and Craig Macdonald. 2022. A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems* (Seattle, WA, USA) *(RecSys '22)*. Association for Computing Machinery, New York, NY, USA, 436–447.

[20] Xiuyuan Qin, Huanhuan Yuan, Pengpeng Zhao, Junhua Fang, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor Sheng. 2023. Meta-optimized Contrastive Learning for Sequential Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 89–98.

[21] Xiuyuan Qin, Huanhuan Yuan, Pengpeng Zhao, Junhua Fang, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor Sheng. 2023. Meta-optimized Contrastive Learning for Sequential Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 89–98.

[22] Xiuyuan Qin, Huanhuan Yuan, Pengpeng Zhao, Guanfeng Liu, Fuzhen Zhuang, and Victor S. Sheng. 2024. Intent Contrastive Learning with Cross Subsequences for Sequential Recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining* (Merida, Mexico) *(WSDM '24)*. Association for Computing Machinery, New York, NY, USA, 548–556.

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

[24] 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* (Raleigh, North Carolina, USA) *(WWW '10)*. Association for Computing Machinery, New York, NY, USA, 811–820.

[25] Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. 2021. Better aggregation in test-time augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1214–1223.

[26] 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* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1441–1450.

[27] 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* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 565–573.

[28] Xing Tang, Chaohua Yang, Yuwen Fu, Dongyang Ao, Shiwei Li, Fuyuan Lyu, Dugang Liu, and Xiuqiang He. 2025. Retrieval Augmented Cross-Domain Life-Long Behavior Modeling for Enhancing Click-through Rate Prediction. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 4891–4900.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[30] Junda Wu, Cheng-Chun Chang, Tong Yu, Zhankui He, Jianing Wang, Yupeng Hou, and Julian McAuley. 2024. CoRAL: Collaborative Retrieval-Augmented Large Language Models Improve Long-tail Recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 3391–3401.

[31] 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 Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence* (Honolulu, Hawaii, USA) *(AAAI'19/IAAI'19/EAAI'19)*. AAAI Press, Article 43, 8 pages.

[32] Zihao Wu, Xin Wang, Hong Chen, Kaidong Li, Yi Han, Lifeng Sun, and Wenwu Zhu. 2023. Diff4Rec: Sequential Recommendation with Curriculum-scheduled Diffusion Augmentation. In *Proceedings of the 31st ACM International Conference on Multimedia* (Ottawa ON, Canada) *(MM '23)*. Association for Computing Machinery, New York, NY, USA, 9329–9335.

[33] Wenjia Xie, Hao Wang, Minghao Fang, Ruize Yu, Wei Guo, Yong Liu, Defu Lian, and Enhong Chen. 2025. Breaking the Bottleneck: User-Specific Optimization and Real-Time Inference Integration for Sequential Recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 3333–3343.

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

[35] Jian Xu, Sichun Luo, Xiangyu Chen, Haoming Huang, Hanxu Hou, and Linqi Song. 2025. RALLRec: Improving Retrieval Augmented Large Language Model

Recommendation with Representation Learning. In *Companion Proceedings of the ACM on Web Conference 2025* (Sydney NSW, Australia) *(WWW '25)*. Association for Computing Machinery, New York, NY, USA, 1436–1440.

[36] Lanling Xu, Zhen Tian, Gaowei Zhang, Junjie Zhang, Lei Wang, Bowen Zheng, Yifan Li, Jiakai Tang, Zeyu Zhang, Yupeng Hou, Xingyu Pan, Wayne Xin Zhao, Xu Chen, and Ji-Rong Wen. 2023. Towards a More User-Friendly and Easy-to-Use Benchmark Library for Recommender Systems. In *SIGIR*. ACM, 2837–2847.

[37] Xihong Yang, Yiqi Wang, Jin Chen, Wenqi Fan, Xiangyu Zhao, En Zhu, Xinwang Liu, and Defu Lian. 2025. Dual Test-Time Training for Out-of-Distribution Recommender System. *IEEE Trans. on Knowl. and Data Eng.* 37, 6 (March 2025), 3312–3326.

[38] Zhaoqi Yang, Yanan Wang, and Yong Ge. 2024. TTT4Rec: A Test-Time Training Approach for Rapid Adaption in Sequential Recommendation. *arXiv preprint arXiv:2409.19142* (2024).

[39] Yufei Ye, Wei Guo, Jin Yao Chin, Hao Wang, Hong Zhu, Xi Lin, Yuyang Ye, Yong Liu, Ruiming Tang, Defu Lian, and Enhong Chen. 2025. FuXi-α: Scaling Recommendation Model with Feature Interaction Enhanced Transformer. In *Companion Proceedings of the ACM on Web Conference 2025* (Sydney NSW, Australia) *(WWW '25)*. Association for Computing Machinery, New York, NY, USA, 557–566.

[40] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Melbourne VIC, Australia) *(WSDM '19)*. Association for Computing Machinery, New York, NY, USA, 582–590.

[41] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, et al. 2024. Actions speak louder than words: trillion-parameter sequential transducers for generative recommendations. In *Proceedings of the 41st International Conference on Machine Learning*. 58484–58509.

[42] Changshuo Zhang, Xiao Zhang, Teng Shi, Jun Xu, and Ji-Rong Wen. 2025. Test-Time Alignment with State Space Model for Tracking User Interest Shifts in Sequential Recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25)*. Association for Computing Machinery, New York, NY, USA, 461–471.

[43] Xinping Zhao, Baotian Hu, Yan Zhong, Shouzheng Huang, Zihao Zheng, Meng Wang, Haofen Wang, and Min Zhang. 2024. Raserec: Retrieval-augmented sequential recommendation. *arXiv preprint arXiv:2412.18378* (2024).

[44] Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, et al. 2025. OneRec Technical Report. *arXiv preprint arXiv:2506.13695* (2025).

[45] 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.

[46] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 2388–2399.

[47] Peilin Zhou, Jingqi Gao, Yueqi Xie, Qichen Ye, Yining Hua, Jaeboum Kim, Shoujin Wang, and Sunghun Kim. 2023. Equivariant contrastive learning for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 129–140.