

X-Rec Technical Report

TikTok-Data-Content Intelligence & TikTok-Data-Feed Quality

Abstract

Recent advances in generative modeling have reshaped recommender systems by formulating recommendation as a next-item generation problem. Existing retrieval approaches primarily follow two paradigms: user-to-item (U2I) methods represent user context using one or a few deterministic embeddings, which limits the ability to capture diverse and multi-mode interests, while semantic-ID-based autoregressive (SID-AR) methods model more expressive distributions but suffer from quantization errors and the low throughput of sequential decoding. To address these limitations, we propose X-Rec to directly learn the recommendation distribution in the continuous item embedding space through flow matching and generate embedding triggers for approximate nearest neighbor retrieval. X-Rec incorporates three key designs to make this formulation effective and efficient. First, we introduce anchor conditioning to decompose generation into coarse semantic-region selection and fine-grained refinement. Second, we adopt Riemannian flow matching to align generative trajectories with the hyperspherical geometry of item embeddings. Third, we design a late-interaction diffusion Transformer that restricts repeated velocity-field estimation to the final Transformer layer. On a streaming benchmark, X-Rec substantially outperforms U2I baselines, matches the retrieval quality of SID-AR methods, and delivers 3.46× higher inference throughput than SID-AR. X-Rec has also been deployed as a new retrieval source for a specific vertical content on TikTok, where two consecutive launches have yielded significant improvements in both vertical engagement (+4.1484%) and general engagement (+0.0111%).

Contents

1	Introduction	3
2	Formulations of Different Paradigms	4
2.1	U2I	5
2.2	SID-AR	6
2.3	X-Rec	7
3	X-Rec Framework	8
3.1	Preliminary: Flow Matching	9
3.2	Algorithm	10
3.3	Model Architecture	11
3.4	Training Pipeline	13
4	Offline Experiments	15
4.1	Experimental Settings	15
4.2	Overall Performance	17
4.3	Ablation Study	18
4.4	Analysis	20
4.5	Case Study	23
5	Online Results	24
6	Conclusion & Future Work	25
7	Contributors	29
A	Appendix	30
A.1	Complete Results on Streaming Benchmark	30
A.2	Architectures of X-Rec-AR and X-Rec-U2I-K	31
A.3	Additional Multi-Interest Case Studies	32

1. Introduction

Recommendation systems have become a core infrastructure for modern content platforms, connecting users with videos, services, merchants, and local consumption opportunities on an industrial scale (Ricci et al., 2010; 2021). In large-scale retrieval, a central challenge is to efficiently recall compact and diverse items that reflects a user’s evolving interests before downstream ranking.

Recommendation can be formulated as a next-item generation task that models the recommendation distribution $p(x | c)$, where x denotes the next item and the user context c typically comprises the user’s historical interaction sequence s and the expected target attribute f^1 (Covington et al., 2016; Kang and McAuley, 2018; Rajput et al., 2023). To this end, existing approaches can be broadly categorized into two paradigms. The first is two-tower user-to-item (U2I) retrieval (Cen et al., 2020; Chen et al., 2024; Zhang et al., 2024), which represents each item x as a continuous embedding and encodes the user context c into one or a few fixed user embeddings for approximate nearest neighbor (ANN) retrieval. Although widely adopted in industrial systems (Covington et al., 2016), this paradigm is limited by its intrinsic *poor expressiveness*: U2I models approximate the target distribution $p(x | c)$ with a delta distribution. Such a distribution is too simple to capture diverse user interests intuitively and has limited model capacity theoretically, e.g., in terms of VC dimension (Csikós et al., 2019). Although recent multi-interest U2I approaches alleviate this problem by introducing more embeddings to encode c , the underlying distributional formulation remains unchanged. Another line of work is SID-AR (Rajput et al., 2023; Deng et al., 2025; Zhou et al., 2025b), which discretizes item embeddings into semantic identifiers (SIDs), thereby transforming recommendation into an autoregressive next-token prediction problem. This approach defines tractable distributions over SIDs and can represent substantially more flexible distributions than the delta distribution. However, it also faces several practical and modeling challenges. The first challenge is what we refer to as *flawed SIDs*, which manifests in two aspects: 1) Discretizing continuous item representations inevitably introduces quantization error (Lee et al., 2022), leading to information loss in the item representation space; and 2) SIDs are ideally expected to preserve a near one-to-one correspondence with item IDs, which is difficult to satisfy in practice.² The second challenge is *low throughput*: autoregressive models generate SID tokens sequentially, limiting inference throughput and posing a significant bottleneck for large-scale retrieval scenarios (Stern et al., 2018; Kwon et al., 2023).

In this paper, we represent each item by its embedding and directly model the conditional recommendation distribution $p(x | c)$ using flow matching (FM)-based diffusion (Lipman et al., 2022; Liu et al., 2022). We term this approach **X-Rec**, emphasizing a return to x itself as the fundamental object of recommendation. This design directly addresses the limitations of existing paradigms. First, to overcome the poor expressiveness of U2I, X-Rec treats recommendation as universal distribution estimation in the continuous item embedding space. Utilizing FM, it can capture complex, diverse, and multi-mode user interest distributions, rather than collapsing them into one or a few deterministic retrieval points. Second, to overcome flawed SIDs of SID-AR, X-Rec operates directly on item embeddings, avoiding the quantization errors introduced by SID discretization, and eliminating the need to enforce a fragile one-to-one correspondence between SIDs and item IDs. Third, to overcome the low throughput of SID-AR, X-Rec achieves high inference throughput by generating a single item embedding through diffusion, rather than autoregressively generating multiple SID tokens, making it an efficient alternative for large-scale retrieval.

¹For example, f can denote the expected target action to be modeled.

²A common workaround is to append a non-semantic hash token to the SID sequence (Rajput et al., 2023; Fu et al., 2026), but this weakens the semantic structure of the code itself.

Built upon diffusion transformers (DiT) (Peebles and Xie, 2023; Wu et al., 2025), X-Rec incorporates three key designs to make continuous generative retrieval both effective and efficient. First, we introduce anchor conditioning to enable coarse-to-fine generation. Before generating the target item embedding, X-Rec first predicts an anchor, i.e., the cluster index of the target embedding, and then conditions the embedding generation process on this anchor. This design reduces the difficulty of directly generating fine-grained item embeddings from scratch, while also making the generation process more controllable. In particular, by manipulating the anchor distribution, X-Rec can explicitly encourage broader coverage over different regions of the item space, thereby improving the diversity of retrieved candidates. Empirically, this anchor-guided generation improves Recall @20 by 2.05% percentage points, demonstrating its effectiveness in enhancing retrieval quality. Second, we adopt Riemannian flow matching (RFM) (Chen and Lipman, 2024; Kumar and Patel, 2026), where the generative trajectory evolves along geodesics on the hypersphere. This formulation is well aligned with the geometry of multi-mode item embeddings, which are typically learned through contrastive objectives under cosine similarity and thus naturally lie on a spherical manifold (Radford et al., 2021). Compared with rectified FM in Euclidean space, RFM avoids wasting model capacity on forcing generated embeddings back onto the sphere, and instead directly models the intrinsic geometry of the item embedding space. Experiments show that this geometry-aware formulation improves Recall @20 by 1.78% in absolute, highlighting the benefit of modeling item embeddings on their native manifold. Third, we employ a late-interaction architecture for efficient velocity-field estimation. Instead of applying the full Transformer at every diffusion step, X-Rec uses only the final Transformer layer to compute the velocity field. Empirically, compared to the full-layer denoising setup, the one-layer design reduces Recall @20 by only 1.05% percentage points, while improving generation throughput by approximately 8.28 $\times$. This makes X-Rec more suitable for large-scale retrieval scenarios where both recommendation quality and serving efficiency are critical.

In offline evaluation, X-Rec substantially outperforms strong baselines. Compared to U2I methods, it can capture the diverse, multi-mode structure of user interests more precisely, leading to significant recall performance improvements. Compared to SID-AR, it achieves comparable recall performance, while improving the inference throughput by 3.46 $\times$, making it a more practical generative retrieval solution for large-scale industrial deployment. X-Rec has also been deployed as a new retrieval source for a specific vertical content on TikTok. The online A/B tests show that two consecutive launches deliver significant gains in both vertical engagement (+4.1484%) and general engagement metrics (+0.0111%).

In the remainder of this paper, we first examine different generative retrieval paradigms in §2, describing their formulations, advantages and limitations. Next, we elaborate on the proposed X-Rec framework in §3, including its training pipeline and model architecture. Then, we conduct offline experiments to validate the effectiveness of X-Rec and its key components in §4, followed by online evaluation results in §5. Finally, we conclude this work and propose potential future directions in §6.

2. Formulations of Different Paradigms

Let X denote the item set. Given a user context c , our goal is to model the conditional target item distribution $p_\theta(x | c)$ over X , where θ denotes the model parameters. Following standard sequential recommendation, the context c contains a chronologically ordered historical interaction sequence s (Kang and McAuley, 2018; Rajput et al., 2023). In our formulation, it additionally contains expected

target attributes³ $f = (f^1, f^2, \dots, f^F) \in \mathcal{F}$, where f^i denotes a sub-attribute. The behavior sequence s consists of a series of interaction tuples, each containing an interacted item and its associated attributes:

$$s = (i_1, i_2, \dots, i_N), \quad i_n = (x_n, f_n) \quad (1)$$

where $f_n = (f_n^1, f_n^2, \dots, f_n^F)$ denotes the attribute vector associated with item x_n , for $n \in \{1, \dots, N\}$.

During inference, we generate multiple retrieval triggers $\tilde{x}_{1:K}$ from $p_\theta(x | c)$ and use them to retrieve candidate items from $\mathcal{X}$. Depending on how the item x and the recommendation distribution $p_\theta(x | c)$ are represented, existing retrieval methods can be instantiated under different paradigms. In the following, we compare these paradigms in terms of both modeling capability and computational complexity. For the complexity analysis, we assume that all methods are built on a Transformer backbone (Vaswani et al., 2017) with L layers and hidden dimension d , where the condition length is N and the number of generated retrieval triggers is K . Table 1 summarizes the formalization and computational complexity of different methods.

Paradigms	Formalization		Complexity	
	x	$p(x c)$	Time	Space
U2I	Embeddings	Train: Sampled Categorical; Test: Delta	$O(N^2)$	$O(1)$
SID-AR	SIDs	Discrete Categorical	$O(N(N + KM))$	$O(LN)$
X-Rec	Embeddings	Continuous	$O(N(N + M))$	$O(N)$

Table 1 | Formalization and complexity of different generative retrieval paradigms.

2.1. U2I

The U2I approach (Cen et al., 2020; Chen et al., 2024) uses an embedding $x^{(e)}$ to represent item x . During training, it is commonly optimized with a contrastive objective, such as InfoNCE (Oord et al., 2018; Radford et al., 2021), which can be interpreted as modeling a sampled categorical distribution:

$$p_\theta^{\text{u2i-train}}(x|c) = \frac{\exp\left(\frac{e_\theta(c)^\top x^{(e)}}{\tau}\right)}{\exp\left(\frac{e_\theta(c)^\top x^{(e)}}{\tau}\right) + \sum_{x' \in \mathcal{X}_{\text{neg}}} \exp\left(\frac{e_\theta(c)^\top x'^{(e)}}{\tau}\right)} \quad (2)$$

Here, τ denotes the temperature hyperparameter, and $\mathcal{X}_{\text{neg}}$ is a collection of negative samples, usually constructed via random sampling strategies such as in-batch negatives (Chen et al., 2020). Because its denominator is normalized only over $\mathcal{X}_{\text{neg}}$ rather than the full item universe $\mathcal{X}$, this objective is a sampled and biased approximation to the full-catalog softmax distribution (Blanc and Rendle, 2018). At inference time, U2I degenerates to a delta distribution:

$$p_\theta^{\text{u2i-test}}(x | c) = \begin{cases} 1, & x^{(e)} = e_\theta(c), \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Accordingly, the recall trigger has only one choice:

$$\tilde{x}^{(e)} = e_\theta(c) \quad (4)$$

³We do not distinguish between item features, e.g., item type, and user-item interaction features, e.g., the user's action on the item. Both are included as attributes.

The recommendation items are then retrieved from $\mathcal{X}$ via ANN. As seen, U2I concentrates all probability mass on a single retrieval point in the item embedding space. This formulation is overly restrictive, as a user’s true interest distribution can be diverse and multi-mode, which lies beyond the representational capacity of a single-point U2I model (Cen et al., 2020).

To address this limitation, prior works such as ComiRec (Cen et al., 2020) encode the user context c into multiple triggers during training. The model then selects one of these triggers for contrastive learning according to a matching strategy⁴. At inference time, the model can output multiple triggers, each of which is used to retrieve candidates from the item embedding space. However, this solution still remains within the same U2I framework. It increases only the number of deterministic retrieval points, rather than explicitly modeling the full manifold of the conditional distribution $p(x|c)$. Moreover, the number of user embeddings must be predefined as a hyperparameter, making it difficult to adaptively capture the intrinsic complexity and varying number of modes in a user’s interest distribution. Therefore, U2I approaches exhibit poor expressiveness.

Since U2I is only made up of an encoder, its inference cost comes from the feed-forward and self-attention computations over the condition tokens:

$$C_{\text{U2I}}^{\text{time}} = \underbrace{O(Ld^2N)}_{\text{FFN of condition tokens}} + \underbrace{O(LdN^2)}_{\text{attention of condition tokens}} \sim O(N^2) \quad (5)$$

Moreover, U2I is KV cache-free, since it does not need to maintain KV caches for subsequent computations. Thus, its space complexity is:

$$C_{\text{U2I}}^{\text{space}} = O(1) \quad (6)$$

2.2. SID-AR

SID-AR (Rajput et al., 2023; Deng et al., 2025; Zhou et al., 2025b) represents each item x as a sequence of SIDs $x_{1:M}^{(s)}$, thereby factorizing the recommendation distribution into an autoregressive sequence model:

$$p_{\theta}^{\text{SID-AR}}(x|c) = \prod_{m=1}^M p_{\theta} \left(x_m^{(s)} \mid c, x_{1:m-1}^{(s)} \right) \quad (7)$$

This formulation allows the model to be trained in the same way as language models, i.e., using cross-entropy supervision over the SID sequence. At inference time, the retrieval triggers are obtained via beam search:

$$\tilde{x}_{1:K}^{(s)} = \text{Beamsearch}_K \left(p_{\theta}^{\text{SID-AR}}(x|c) \right) \quad (8)$$

In contrast to U2I, SID-AR defines a compositional distribution over discrete SID sequences and is therefore more flexible in representing multi-mode user interests (Uria et al., 2016).

Despite this advantage, the SID-AR formulation introduces two important drawbacks. Firstly, its retrieval quality is highly dependent on the construction of SIDs. Since SIDs are obtained by quantizing continuous item embeddings, the discretization process inevitably discards fine-grained information in the original representation space (Lee et al., 2022; Rajput et al., 2023). Moreover, the discrete code space must be organized such that generated SID sequences can be reliably mapped back to valid and distinguishable items, a requirement that is non-trivial to satisfy at industrial scale. Secondly, SID-AR inherits the efficiency bottleneck of autoregressive decoding. Generating a retrieval

⁴For example, selecting the trigger that is closest to the ground-truth item embedding.

trigger requires multiple sequential prediction steps and KV cache maintenance, making the inference cost grow with the SID length and limiting throughput in large-scale retrieval systems (Stern et al., 2018; Kwon et al., 2023). Specifically, the complexity of SID-AR is:

$$\begin{aligned}
C_{\text{SID-AR}}^{\text{time}} = & \underbrace{O(Ld^2N)}_{\text{FFN of condition tokens}} + \underbrace{O(LdN^2)}_{\text{Attention of condition tokens}} \\
& + \underbrace{O(Ld^2K(M-1))}_{\text{FFN of generated tokens}} + \underbrace{O\left(\frac{LdK(2N+M-2)(M-1)}{2}\right)}_{\text{Attention of generated tokens}} \sim O(N(N+KM)) \quad (9) \\
C_{\text{SID-AR}}^{\text{space}} = & O(L(N+M))
\end{aligned}$$

As we can see, compared to U2I, SID-AR has additional time complexity from generated tokens and additional space complexity from maintaining the KV cache, which significantly limits its inference throughput.

Overall, SID-AR approaches suffers from flawed SIDs and low throughput.

2.3. X-Rec

Combining the advantages of U2I and SID-AR, our proposed X-Rec represents each item x as a continuous embedding $x^{(e)}$ and directly models the recommendation distribution in the continuous embedding space with FM-based diffusion (Lipman et al., 2022; Liu et al., 2022). In essence, FM learns a time-dependent velocity field $v_\theta^t(\cdot | c)$ that transports a noise sample $\epsilon \sim r(\epsilon)$ toward the target item embedding $x^{(e)}$. During training, the model is optimized with the FM objective, which regresses the predicted velocity field to the ground-truth conditional velocity. During inference, the model generates triggers by denoising a noise step by step, which corresponds to sampling from the learned recommendation distribution⁵:

$$\tilde{x}_{1:K}^{(e)} \sim p_\theta^{\text{X-Rec}}(x|c) \quad (10)$$

Compared with U2I, this formulation can model general continuous distributions, providing sufficient expressiveness to capture diverse and multi-mode user interests, as illustrated in Figure 1⁶. For a case study, please refer to §4.5.

Compared with SID-AR, X-Rec operates directly on item embeddings, avoiding the quantization errors introduced by discretization and eliminating the need to enforce a fragile one-to-one correspondence between SIDs and item IDs. Meanwhile, the time complexity of X-Rec is:

$$\begin{aligned}
C_{\text{X-Rec}}^{\text{time}} = & \underbrace{O(Ld^2N)}_{\text{FFN of condition tokens}} + \underbrace{O(LdN^2)}_{\text{attention of condition tokens}} + \underbrace{O(L'Td^2K)}_{\text{FFN of generated embeddings}} + \underbrace{O(L'TdKN)}_{\text{attention of generated embeddings}} \quad (11)
\end{aligned}$$

⁵The form of recommendation distribution $p_\theta^{\text{X-Rec}}(x|c)$ is implicitly defined by the velocity field: $p_\theta^{\text{X-Rec}}(x|c) = r(\epsilon) \exp\left(-\int_0^1 \nabla \cdot v_\theta^t(\phi_\theta^t(\epsilon|c)|c)dt\right)$, where $\frac{d\phi_\theta^t(\cdot|c)}{dt} = v_\theta^t(\phi_\theta^t(\cdot|c)|c)$ (Lai et al., 2025).

⁶From a decision-boundary perspective, retrieval with multiple triggers corresponds to selecting candidates $x \in \mathcal{X}$ such that $f(x) = 1 \left(\max_{k \in [K]} \left(\tilde{x}_k^{(e)\top} x^{(e)} \right) > \tau \right)$. Thus, the positive retrieval region is a union of K halfspaces induced by the generated triggers. The VC dimension of this class is $\Theta(dK \log K)$ (Csikós et al., 2019), indicating that increasing the number of triggers directly increases the complexity of the representable decision boundary. This offers a theoretical explanation for the stronger expressiveness of X-Rec over single-vector U2I retrieval: multiple triggers allow X-Rec to represent a richer, multi-region candidate space rather than a single local neighborhood.

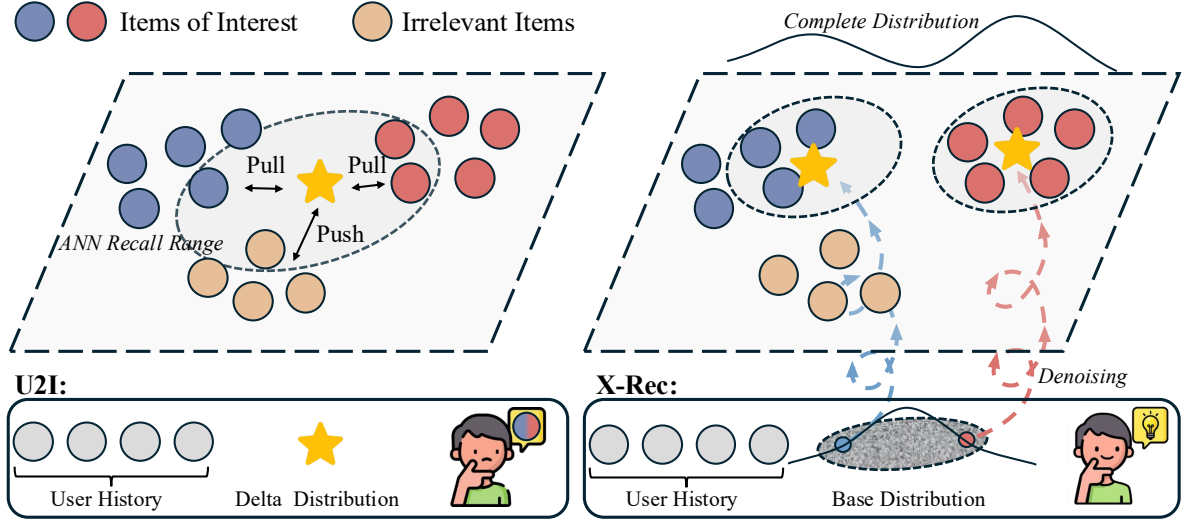


Figure 1 | Illustration of the expressiveness of U2I and X-Rec. In the left panel, U2I produces a single retrieval trigger. Its training objective pulls this trigger toward items of interest while pushing it away from irrelevant items. Due to geometric constraints in the embedding space, however, a single trigger can easily fall into a compromised position, making it difficult to retrieve all relevant items without also retrieving irrelevant ones. In the right panel, X-Rec directly models the distribution of items of interest and generates multiple retrieval triggers. These triggers can cover different interest regions, enabling more expressive and precise retrieval of relevant items.

Here, L' denotes the number of Transformer layers used in each velocity field evaluation, and T denotes the number of denoising steps. As shown in §4.3.3, we empirically find that using only the final Transformer layer at each denoising step, i.e., $L' = 1$, and setting the number of denoising steps equal to the total number of Transformer layers, i.e., $T = L$, provides a favorable quality–efficiency trade-off. Under this efficient configuration, the time complexity of X-Rec becomes:

$$C_{\text{X-Rec}}^{\text{time}} = \underbrace{O(Ld^2N)}_{\text{FFN of condition tokens}} + \underbrace{O(LdN^2)}_{\text{attention of condition tokens}} + \underbrace{O(Ld^2K)}_{\text{FFN of generated embeddings}} + \underbrace{O(LdKN)}_{\text{attention of generated embeddings}} \sim O(N(N+K)) \quad (12)$$

With this design, X-Rec only need to store the last-layer KV cache of the condition tokens, i.e.,

$$C_{\text{X-Rec}}^{\text{space}} = O(N) \quad (13)$$

Therefore, the complexity of X-Rec lies between U2I and SID-AR. More importantly, X-Rec achieves a favorable trade-off between expressiveness and efficiency: it significantly improves upon U2I by modeling a flexible continuous recommendation distribution, while avoiding the expensive token-by-token autoregressive decoding required by SID-AR. This makes X-Rec expressive enough to capture complex user interest distributions and efficient enough for large-scale retrieval.

3. X-Rec Framework

In this section, we present the proposed X-Rec framework in detail. We begin with a brief overview of FM, then describe the high-level training and inference algorithms of X-Rec, followed by the concrete

implementation of the model architecture. Finally, we introduce the overall training pipeline.

3.1. Preliminary: Flow Matching

Given a recommendation context $\hat{c}$, let $\hat{x}^{(e)} \in \mathcal{X}^{(e)}$ denote the item embedding of the ground truth item $\hat{x}^7$, sampled from an empirical recommendation distribution $p_{\text{data}}(x | \hat{c})$. The key idea of FM is to learn a time-dependent flow function $\phi_{\theta}^t(\cdot | \hat{c})$, where $t \in [0, 1]$, that transforms a simple continuous random variable $\epsilon \sim r(\epsilon)$, i.e., noise, into a sample from the target recommendation distribution (Lipman et al., 2023). Specifically, the transformed random variable $\phi_{\theta}^1(\epsilon | \hat{c})$ induces a model distribution $p_{\theta}(x^{(e)} | \hat{c})$, which is expected to approximate $p_{\text{data}}(x^{(e)} | \hat{c})$. Therefore, the learned flow should satisfy

$$\phi_{\theta}^0(\epsilon | \hat{c}) = \epsilon \sim r(\epsilon), \quad \phi_{\theta}^1(\epsilon | \hat{c}) \sim p_{\text{data}}(x^{(e)} | \hat{c}) \quad (14)$$

Once such a flow is learned, we can sample from $p_{\theta}(x^{(e)} | \hat{c})$ by first drawing noise from $r(\epsilon)$ and then denoising it through the learned flow. However, directly constructing a flow that satisfies the above requirement is challenging. Fortunately, it is straightforward to define a conditional flow when the target embedding $\hat{x}^{(e)}$ is given. This conditional flow $\phi^t(\epsilon | \hat{x}^{(e)}, \hat{c})$ can be constructed by interpolating from the noise ϵ to the target embedding $\hat{x}^{(e)}$ over time t :

$$\phi^0(\epsilon | \hat{x}^{(e)}, \hat{c}) = \epsilon \sim r(\epsilon), \quad \phi^1(\epsilon | \hat{x}^{(e)}, \hat{c}) = \hat{x}^{(e)} \quad (15)$$

In most cases, the conditional flow is independent of $\hat{c}$. For example, Euclidean FM can use the linear interpolation $\phi^t(\epsilon | \hat{x}^{(e)}, \hat{c}) = (1 - t)\epsilon + t\hat{x}^{(e)}$ (Lipman et al., 2023). For convenience, we abbreviate $\phi^t(\epsilon | \hat{x}^{(e)}, \hat{c})$ as $\tilde{x}_t^{(e)}$. Taking the derivative of the conditional flow with respect to time yields the corresponding conditional velocity field:

$$v^t(\tilde{x}_t^{(e)} | \hat{x}^{(e)}, \hat{c}) = \frac{d}{dt} \tilde{x}_t^{(e)} \quad (16)$$

The FM objective then regresses a neural velocity field $v_{\theta}^t(\tilde{x}_t^{(e)} | \hat{c})$ to match this conditional velocity field in expectation over data, noise, and timestep:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{\substack{\hat{c}, \hat{x}^{(e)} \sim p_{\text{data}}, \\ \epsilon \sim r, t \sim \mathcal{U}(0,1)}} \left[\left\| v_{\theta}^t(\tilde{x}_t^{(e)} | \hat{c}) - \frac{d}{dt} \tilde{x}_t^{(e)} \right\|_{\mathcal{X}^{(e)}}^2 \right] \quad (17)$$

Lipman et al. (2023) have proven that the optimal regressed velocity field $v_{\theta^*}^t(\cdot | \hat{c})$ induces a flow function that satisfies the construction requirement in eq. (14). More precisely, the induced flow $\phi_{\theta^*}^t(\epsilon | \hat{c})$, which we refer to as $\tilde{x}_t^{(e)}$, is defined as the solution to the following ordinary differential equation (ODE)⁸:

$$\begin{aligned} \frac{d}{dt} \tilde{x}_t^{(e)} &= v_{\theta^*}^t(\tilde{x}_t^{(e)} | \hat{c}) \\ \tilde{x}_0^{(e)} &= \epsilon \sim r(\epsilon) \end{aligned} \quad (18)$$

At inference time, we numerically integrate the above learned flow on the representation space $\mathcal{X}^{(e)}$ using Euler discretization, as we shall see in §3.2.2.

⁷In this paper, $\hat{\cdot}$ indicates a sample drawn from an empirical distribution.

⁸In this paper, $\tilde{\cdot}$ indicates a model-estimated counterpart of the underlying random variable.

3.2. Algorithm

This section presents the learning and generation procedures of X-Rec. During training, the model jointly learns coarse semantic anchor prediction and anchor-conditioned FM on the hypersphere. During inference, X-Rec first identifies an anchor from the recommendation context and then integrates the learned vector field on the hypersphere to produce recall triggers. We describe the training and inference stages in turn.

3.2.1. Training

Anchor Loss. Although FM provides a general simulation-free framework for learning continuous transport paths (Lipman et al., 2023), directly generating fine-grained item embeddings remains challenging due to the complexity and multi-mode nature of the item representation space. To address this issue, we introduce anchor conditioning, which enables a coarse-to-fine generation process. Concretely, X-Rec first predicts a coarse semantic region of the target item and then generates the recall-trigger embedding conditioned on this region. We represent the coarse region by the cluster-center index $\hat{x}^{(a)} \in \mathcal{A}$ of the target item embedding $\hat{x}^{(e)}$. We refer to $\hat{x}^{(a)}$ as an ‘‘anchor’’ because it provides a coarse localization of the target embedding. This decomposition reduces the difficulty of continuous embedding generation and thus makes the generation process more stable and controllable.

Naturally, we optimize anchor prediction with the cross-entropy loss:

$$\mathcal{L}_{\text{anchor}} = -\mathbb{E}_{(\hat{c}, \hat{x}^{(a)}) \sim p_{\text{data}}} [\log p_{\theta}(\hat{x}^{(a)} \mid \hat{c})] \quad (19)$$

where $\hat{x}^{(a)}$ denotes the ground-truth anchor of the target item under context $\hat{c}$.

RFM Loss. Conditioned on both the user context and anchor, X-Rec then generates the final recall trigger by progressively denoising the noise. As aforementioned, each item $\hat{x} \in \mathcal{X}$ is represented by a embedding $\hat{x}^{(e)} \in \mathcal{X}^{(e)}$. The embedding model is trained with contrastive representation learning and ℓ_2 normalization, so item embeddings naturally lie on the unit hypersphere, i.e., $\mathcal{X}^{(e)} = \mathcal{S}^{d_e-1}$, where d_e is the embedding dimension. In this setting, directly applying FM in Euclidean space using linear interpolation and additive denoising may cause intermediate and generated states deviate from the hypersphere. Such off-manifold generated triggers are misaligned with the normalized item embeddings. Therefore, we instantiate the general formulation in Section 3.1 as Riemannian Flow Matching (RFM) (Chen and Lipman (2024); Kumar and Patel (2026)), ensuring that the flow trajectory remains on the hypersphere.

Specifically, RFM constructs the conditional flow through geodesic interpolation:

$$\bar{x}_t^{(e)} = \frac{\sin((1-t)\omega)}{\sin \omega} \epsilon + \frac{\sin(t\omega)}{\sin \omega} \hat{x}^{(e)}, \quad \omega = \arccos(\epsilon^\top \hat{x}^{(e)}) \quad (20)$$

where $\epsilon \sim r(\epsilon)$ is sampled from the base distribution on the hypersphere, and ω is the geodesic distance between ϵ and $\hat{x}^{(e)}$. The corresponding conditional velocity field is obtained by differentiating the geodesic path with respect to t :

$$\frac{d}{dt} \bar{x}_t^{(e)} = \frac{\omega}{\sin \omega} \left(\cos(t\omega) \hat{x}^{(e)} - \cos((1-t)\omega) \epsilon \right) \quad (21)$$

Substituting this conditional velocity into the flow-matching objective in Equation (17), we can then obtain the final RFM loss.

$$\mathcal{L}_{\text{RFM}} = \mathbb{E}_{\substack{(\hat{c}, \hat{x}^{(a)}, \hat{x}^{(e)}) \sim p_{\text{data}}, \\ \epsilon \sim r(\epsilon), \\ t \sim \mathcal{U}(0,1)}}} \left\| v_{\theta}^t \left(\frac{\sin((1-t)\omega)}{\sin \omega} \epsilon + \frac{\sin(t\omega)}{\sin \omega} \hat{x}^{(e)} \mid \hat{c}, \hat{x}^{(a)} \right) - \frac{\omega}{\sin \omega} \left(\cos(t\omega) \hat{x}^{(e)} - \cos((1-t)\omega) \epsilon \right) \right\|^2 \quad (22)$$

$\omega = \arccos(\epsilon^\top \hat{x}^{(e)})$

TotalLoss. In practice, for each training example, we sample multiple noise vectors ϵ and timesteps t , and use the resulting empirical mean to estimate the expectation in the RFM objective. The final training objective of X-Rec is a weighted sum of the above two terms:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{anchor}} + \beta \mathcal{L}_{\text{RFM}} \quad (23)$$

where α and β are weighting coefficients.

3.2.2. Inference

During inference, X-Rec first predicts an anchor $\tilde{x}^{(a)} \sim p_{\theta}(x^{(a)} \mid \hat{c})$. Then, it initializes the state from the base distribution $\tilde{x}_0^{(e)} \sim r(\epsilon) = U_{S^{d_e-1}}(\epsilon)$, where $U_{S^{d_e-1}}(\epsilon)$ denotes the uniform distribution on the unit hypersphere. Next, it denoises this state progressively via Riemannian Euler integration with timestep interval Δt :

$$\tilde{x}_{t+\Delta t}^{(e)} = \cos \left(\left\| v_{\theta}^t(\tilde{x}_t^{(e)} \mid \hat{c}, \tilde{x}^{(a)}) \right\| \Delta t \right) \tilde{x}_t^{(e)} + \sin \left(\left\| v_{\theta}^t(\tilde{x}_t^{(e)} \mid \hat{c}, \tilde{x}^{(a)}) \right\| \Delta t \right) \frac{v_{\theta}^t(\tilde{x}_t^{(e)} \mid \hat{c}, \tilde{x}^{(a)})}{\left\| v_{\theta}^t(\tilde{x}_t^{(e)} \mid \hat{c}, \tilde{x}^{(a)}) \right\|} \quad (24)$$

Once t reaches 1, the terminal state $\tilde{x}_1^{(e)}$ is used as the final recall trigger.

3.3. Model Architecture

As introduced in §3.2, X-Rec consists of two key modules: an anchor predictor $p_{\theta}(x^{(a)} \mid c)$ and a velocity predictor $v_{\theta}^t(\tilde{x}_t^{(e)} \mid c, x^{(a)})$. We first describe how the features in the recommendation context are encoded, and then detail the architectures of the two modules.

3.3.1. Feature Encoding

The recommendation context c consists of the historical interaction sequence s and the expected target attribute f . We now describe how s and f are represented in our framework. First, the attribute is represented by projecting its sub-attribute embeddings to the model dimension d_m :

$$f^{(m)} = W_f f^{(e)} = W_f [f^{1(e)}; f^{2(e)}; \dots; f^{F(e)}] \in \mathbb{R}^{d_m} \quad (25)$$

where W_f is the learnable attribute projection matrix. Each interaction in the historical interaction sequence s , i.e., a tuple $i = (x, f) \in h$, is represented by fusing the information of item and attributes:

$$i^{(m)} = x^{(m)} + f^{(m)} = W_x x^{(e)} + W_f f^{(e)} \in \mathbb{R}^{d_m} \quad (26)$$

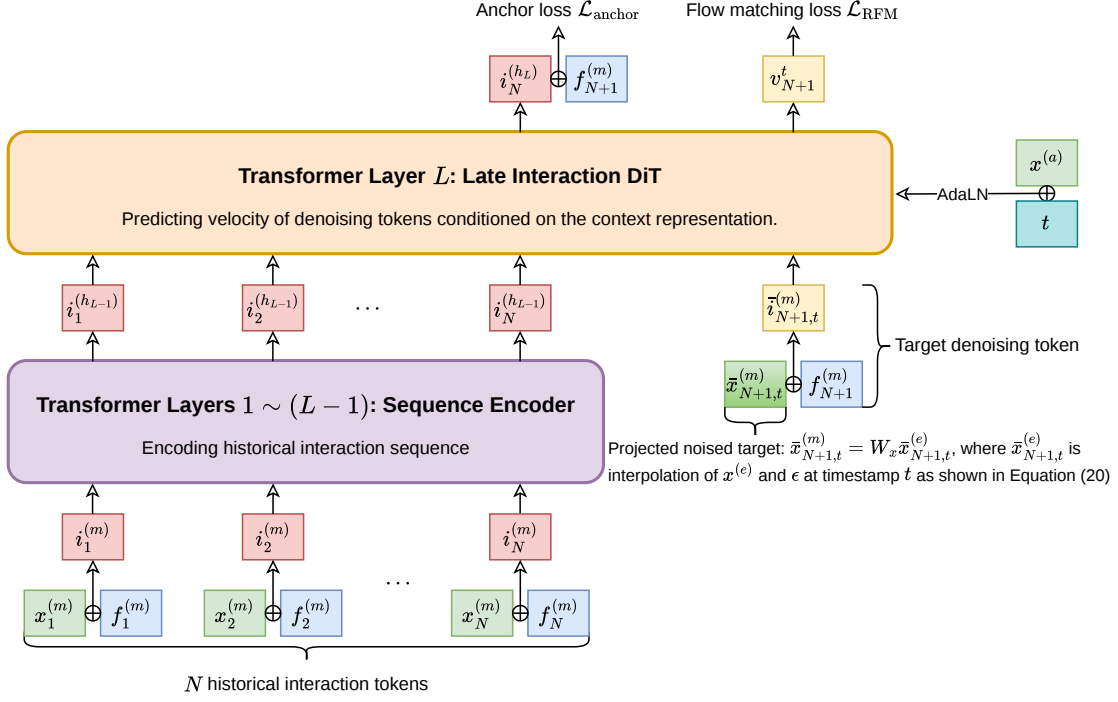


Figure 2 | Model architecture of X-Rec. Historical interaction tokens are first encoded by the first $L-1$ Transformer layers, and the target denoising tokens are then fed only into the last Transformer layer to predict the velocity. Pretraining and SFT share the same architecture but differ in context–target construction and attention visibility.

where W_x is the item-embedding projection matrix. The encoded historical interaction sequence s is therefore represented as a sequence of interaction tokens:

$$s^{(m)} = (i_1^{(m)}, i_2^{(m)}, \dots, i_N^{(m)}) \in \mathbb{R}^{d_m \times N} \quad (27)$$

As illustrated in Figure 2, the encoded interaction sequence $s^{(m)}$ is then fed into an L -layer Transformer to produce multi-level contextualized user representations:

$$\begin{aligned} s^{(h_1)} &= \text{TransformerLayer}_1(s^{(m)}) = (i_1^{(h_1)}, i_2^{(h_1)}, \dots, i_N^{(h_1)}) \in \mathbb{R}^{d_m \times N} \\ s^{(h_2)} &= \text{TransformerLayer}_2(s^{(h_1)}) = (i_1^{(h_2)}, i_2^{(h_2)}, \dots, i_N^{(h_2)}) \in \mathbb{R}^{d_m \times N} \\ &\dots \\ s^{(h_L)} &= \text{TransformerLayer}_L(s^{(h_{L-1})}) = (i_1^{(h_L)}, i_2^{(h_L)}, \dots, i_N^{(h_L)}) \in \mathbb{R}^{d_m \times N} \end{aligned} \quad (28)$$

We adopt the Qwen3-style (Qwen3 et al. (2025); Qwen3.5 (2026)) Transformer backbone, where RoPE (Su et al. (2024)) is used for positional encoding and QK-normalization is applied to stabilize training. The user sequence is encoded with a causal attention mask, making $s_N^{(h_L)}$ a valid contextualized representation of the historical interaction sequence. Moreover, this masking strategy allows the resulting KV cache to be reused during the subsequent velocity-prediction stage, which significantly improves inference throughput.

3.3.2. Anchor Predictor

Since the anchor $x^{(a)}$ is discrete, the anchor predictor is naturally a softmax classifier based on the historical-sequence embedding $i_N^{(h_L)}$ and the expected target-attribute embedding $f^{(m)}$:

$$p_\theta(x^{(a)}|c) = \frac{\exp\left(w_a^\top (i_N^{(h_L)} + f^{(m)})\right)}{\sum_{a' \in \mathcal{A}} \exp\left(w_{a'}^\top (i_N^{(h_L)} + f^{(m)})\right)}, \quad w_a \in \mathbb{R}^{d_m} \quad (29)$$

where w_a is the classifier weight vector for anchor a .

3.3.3. Velocity Predictor: Late Interaction DiT

The velocity predictor $v_\theta^t(\tilde{x}_t^{(e)} | c, x^{(a)})$ largely follows the DiT design (Peebles and Xie, 2023; Wu et al., 2025), with one important modification: we introduce a late-interaction (Khattab and Zaharia, 2020) structure for efficient velocity computation. Given the current denoising state $\tilde{x}_t^{(e)}$, we first incorporate the target attribute information $f^{(m)}$ into the denoising state in the same way of the interaction tokens:

$$\tilde{i}_t^{(m)} = \tilde{x}_t^{(m)} + f^{(m)} = W_x \tilde{x}_t^{(e)} + W_f f^{(e)} \in \mathbb{R}^{d_m} \quad (30)$$

Instead of feeding the denoising token $\tilde{i}_t^{(m)}$ through all Transformer layers, we insert it only into the last layer. This design allows the model to reuse the contextualized historical representations computed in the prefill stage, while using the final layer to estimate the velocity field for the current denoising state. Formally,

$$\begin{aligned} \left[s^{(h_L)}; \tilde{i}_t^{(h_L)} \right] &= \text{TransformerLayer}_L \left(\left[s^{(h_{L-1})}; \tilde{i}_t^{(m)} \right], t, x^{(a)} \right) \in \mathbb{R}^{(N+1) \times d_m} \\ v_\theta^t(\tilde{x}_t^{(e)} | c, x^{(a)}) &= W_v \tilde{i}_t^{(h_L)} \in \mathbb{R}^{d_e} \\ v_\theta^t(\tilde{x}_t^{(e)} | c, x^{(a)}) &= \underbrace{v_\theta^t(\tilde{x}_t^{(e)} | c, x^{(a)}) - \left(v_\theta^t(\tilde{x}_t^{(e)} | c, x^{(a)})^\top \tilde{x}_t^{(e)} \right) \tilde{x}_t^{(e)}}_{\text{projection onto } T_{\tilde{x}_t^{(e)}} \mathcal{S}^{d_e-1}} \in T_{\tilde{x}_t^{(e)}} \mathcal{S}^{d_e-1} \end{aligned} \quad (31)$$

where $(v_\theta^t)^\top \tilde{x}_t^{(e)} \tilde{x}_t^{(e)}$ is the radial velocity component along the current state, and $T_{\tilde{x}_t^{(e)}} \mathcal{S}^{d_e-1} = \{u \in \mathbb{R}^{d_e} : u^\top \tilde{x}_t^{(e)} = 0\}$ denotes the tangent space of the unit hypersphere at $\tilde{x}_t^{(e)}$. The projection removes the radial component of the velocity, ensuring that the predicted velocity lies in the tangent space.

As illustrated in Figure 2, the flow timestep t and anchor $x^{(a)}$ condition the final Transformer layer through AdaLN (Peebles and Xie, 2023): their embeddings are added up, and jointly modulate the attention and MLP blocks. A causal mask is applied when encoding the historical interaction sequence, allowing its KV cache to be reused across iterative velocity evaluations. As shown in §4.3.3, this late-interaction design substantially improves generation throughput, with only a marginal decrease in Recall@20 compared with full-layer denoising.

3.4. Training Pipeline

Training X-Rec includes two stages: pretraining and supervised fine-tuning (SFT). The pretraining stage aims to capture general sequential transition structure and knowledge from users' historical

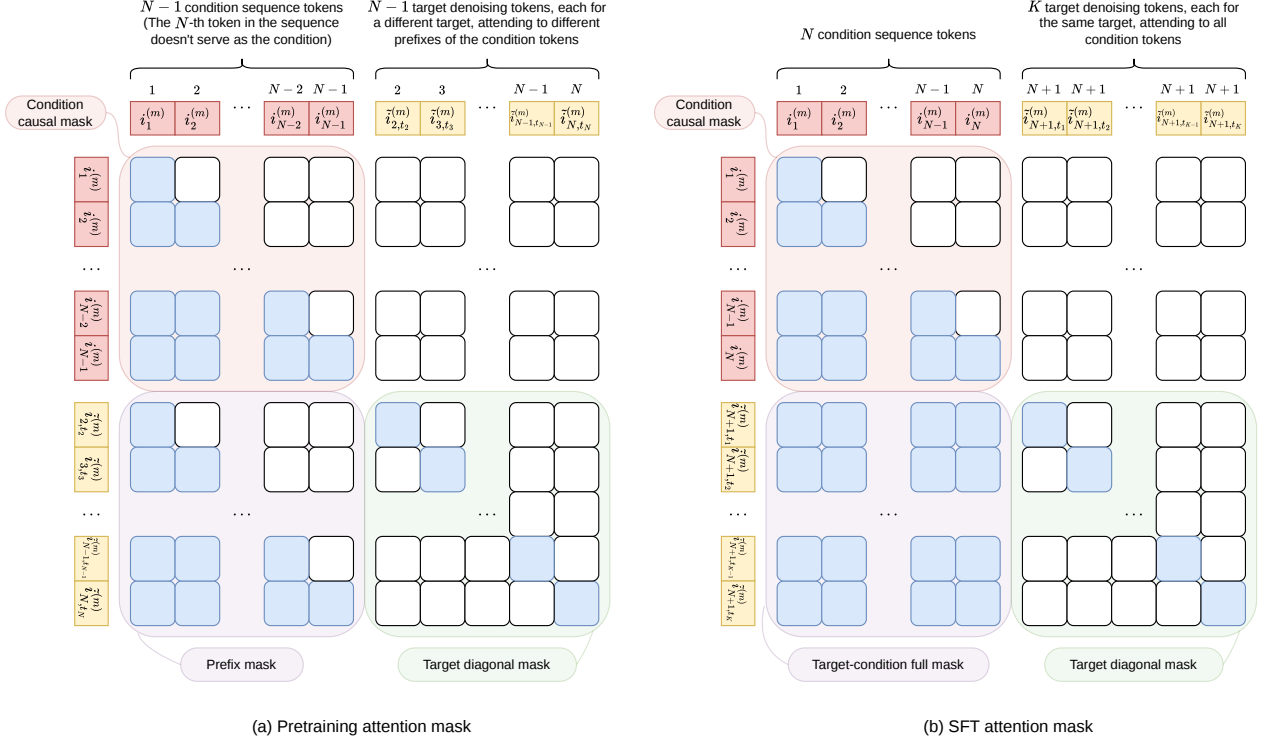


Figure 3 | Attention masks used in X-Rec. During pretraining, each denoising token corresponds to a different target and attends only to its designated history prefix and itself. Accordingly, the position index of each denoising token is set to the length of its corresponding prefix plus one. During SFT, all denoising tokens correspond to the same target and attend to the complete condition sequence. Their position indices are therefore identical, i.e., the length of the condition sequence plus one. In both stages, condition tokens are encoded with a causal mask, allowing the last-layer KV cache to be reused across denoising tokens at different timesteps. Meanwhile, denoising tokens remain mutually invisible to one another.

interaction sequences and item embeddings, enabling the model to learn broadly transferable representations. The SFT stage further adapts the pretrained model to specific recommendation scenarios, such as particular content types or expected target attributes.

3.4.1. Pretraining

Given a chronological user historical interaction sequence $s = (i_1, i_2, \dots, i_N)$, $i_n = (x_n, f_n)$, the pretraining stage models the following autoregressive factorization of the item distribution:

$$\prod_{n=1}^N p_{\theta}(x_n | i_{1:n-1}, f_n) \quad (32)$$

Here, each factor $p_{\theta}(x_n | i_{1:n-1}, f_n)$ can be optimized using the X-Rec loss. Thus, a single sequence yields N next-item prediction pairs according to different prefix lengths.

Specifically, for each target item x_n , the model samples a noise vector and a timestep, and evaluates the loss defined in eq. (23). To improve training efficiency, we introduce a specialized attention mask that allows the losses for all sequence positions to be computed within a single prefill pass. As illustrated in Figure 3, this mask restricts each denoising token to attend only to its corresponding

historical prefix, while keeping different denoising tokens isolated from one another. As a result, X-Rec can train all prefix–target pairs in parallel without changing the semantics of sequential pretraining.

After pretraining, the model captures general item-transition patterns. In the next stage, SFT, it further adapts the context and target semantics to the downstream serving task.

3.4.2. Supervised Fine-Tuning

SFT adapts the pretrained transition model to the retrieval events used by the downstream service. For each eligible target item x with attribute f , we construct its sequence s by chronologically concatenating two types of user history: a historical positive sequence and a recent interaction sequence. The historical positive sequence consists of items with which the user has had meaningful engagement, whereas the recent interaction sequence contains the user’s most recent activity before the target event. These two sources provide complementary signals: the former reflects stable and persistent user preferences, while the latter captures immediate and transient intent that may not appear in the long-term positive history (Yu et al., 2019).

For each target item, we draw eight independent noise samples and timesteps to estimate the RFM objective. As shown in Figure 3, we also use a specialized attention mask, where each denoising token attends to the complete composed history and the same expected target attribute, while different noised copies remain mutually isolated. This allows multiple denoising tokens for the same target to be trained in parallel without introducing information leakage across replicas.

Overall, the SFT stage adapts the pretrained model to the downstream retrieval task, enabling it to generate retrieval triggers that better match the serving-time user context and target attribute.

4. Offline Experiments

In this section, we evaluate X-Rec in offline settings. We first describe the experimental setup, including the chronological streaming benchmark, evaluation metrics, and baselines. We then compare X-Rec with representative methods and analyze its key design choices through ablation studies.

4.1. Experimental Settings

4.1.1. Data

We evaluate X-Rec on an industrial streaming benchmark constructed from TikTok platform. All models are first pretrained on large-scale historical interaction sequences and then trained and evaluated on the stream, following the continual-update setting used in production. For pretraining and streaming data, the item attribute set f includes only user actions. Therefore, the task is to generate items that are aligned with the user’s demonstrated interests, conditioned on the user’s historical interaction sequence.

Pretraining Data. We construct pre-training data from large-scale user–item interaction logs on the platform. The sequences are divided into chunks containing at most 200 interactions.

Streaming Evaluation Benchmark. To approximate production retrieval, where models are continuously updated with fresh user feedback, we construct a streaming benchmark comprising 47 consecutive partitions. Each partition includes 2.6M sampled training examples and 10K test examples, where each example consists of a historical positive-interaction sequence and a recent-interaction sequence, both of bounded length. We collect the target items from each partition to construct the corresponding ANN recall source. Models are trained and evaluated sequentially over the partitions: the final checkpoint from one partition is used to initialize training on the next, preserving the temporal order of both data and model updates. This protocol captures model performance under continual adaptation and provides a more faithful offline approximation of online retrieval behavior than a static evaluation split.

4.1.2. Evaluation Metrics

For each test example (c, x) , the model generates K embedding triggers $t_{1:K}$. Each trigger is used to retrieve its top- R nearest items from the ANN index. We define Recall @ $(K \times R)$ as

$$\text{Recall @}(K \times R) = \frac{1}{|\mathcal{D}|} \sum_{(c,x) \in \mathcal{D}} \mathbb{I} \left[x \in \bigcup_{k=1}^K \text{ANN}_R(t_k) \right], \quad (33)$$

where $\text{ANN}_R(t)$ denotes the set of top- R items retrieved by trigger t , and $\mathcal{D}$ denotes the testing set. This metric measures the fraction of test examples for which the ground-truth item is included in the union of retrieved candidates. By default, we report Recall @20×1, which we abbreviate as Recall @20. §A.1 reports the complete per-snapshot results.

We also evaluate generation efficiency using throughput, measured as requests per second on a single production-grade GPU. Each request generates $K = 20$ triggers, matching the default retrieval setting. The throughput measurement includes only trigger generation and excludes the ANN lookup stage.

4.1.3. Baseline Methods

We compare X-Rec against a broad range of representative open-source baselines from two categories. The first category comprises U2I methods, including SASRec (Kang and McAuley, 2018), LRURec (Yue et al., 2024), and DreamRec (Yang et al., 2023)⁹. The second category comprises SID-based generative methods, including TIGER (Rajput et al., 2023), EAGER (Wang et al., 2024), LATTE (Hou et al., 2026), and LLaDARec (Shi et al., 2025). For all open-source baselines, we follow the model configurations reported in their original papers.

We additionally implement two controlled variants of X-Rec to disentangle the effects of the learning paradigm and generation architecture. X-Rec-U2I is trained with a contrastive objective and produces a single trigger embedding for retrieval, whereas X-Rec-AR employs a decoder-only architecture to generate SIDs autoregressively. Both variants use the same context-encoder configuration as X-Rec, as detailed in §A.2.

For U2I methods, which produce a single retrieval embedding, we compute Recall @20 under the 1×20 setting. For SID-based methods and X-Rec, which support multiple retrieval triggers, we use the 20×1 setting by default. We denote both settings as Recall @20 when the underlying configuration is clear from context.

⁹Although DreamRec is diffusion-based, its original implementation generates only a single embedding for retrieval. We therefore follow this setting and categorize DreamRec as a U2I method.

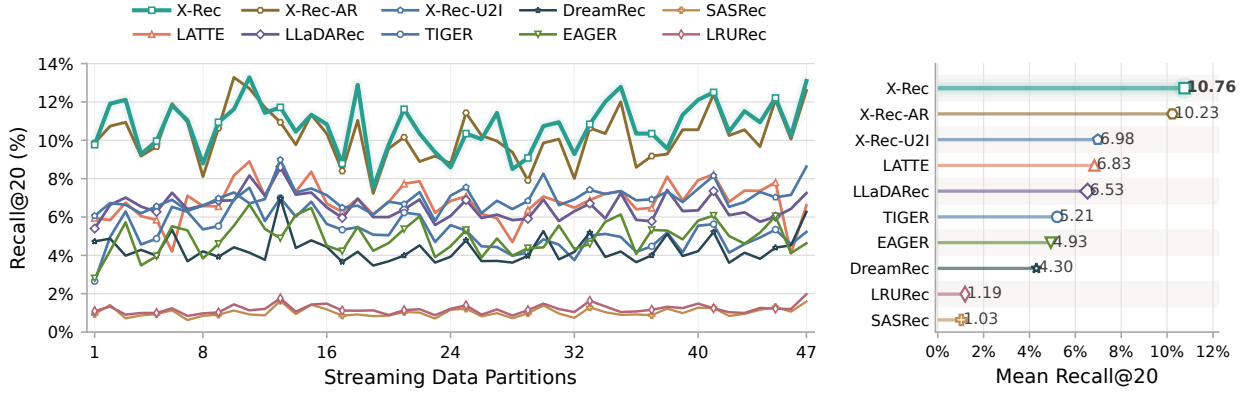


Figure 4 | Comparison of different models on the streaming benchmark. The left panel shows the results for each partition, while the right panel ranks the methods by their average performance.

All models follow the same pretraining-then-streaming-SFT pipeline. To ensure a consistent comparison between SID-based and embedding-based retrieval methods, we adopt a unified ANN evaluation protocol. Specifically, the SIDs generated by SID-based methods are decoded into embeddings through their respective RQ codebooks, and the resulting embeddings are used as triggers for ANN retrieval.

4.1.4. Implementations

X-Rec adopts Qwen3-0.6B (Qwen3 et al., 2025) as its Transformer backbone. After removing the token embedding layer, the resulting model contains approximately 0.46B parameters. Each item is a video, and is represented by a 128-dimensional multimodal embedding normalized to the unit hypersphere. SIDs are obtained by quantizing item embeddings with K-means residual quantization using 4 codebooks of size 4096 (Zhou et al., 2025a), yielding an average reconstruction cosine similarity of 0.94.

During pretraining, we train the model for one epoch (200K steps) using AdamW (Loshchilov and Hutter, 2017) with a batch size of 4096. The learning rate is linearly warmed up to 5×10^{-4} over the first 2K steps and then decayed to 1×10^{-4} following a cosine schedule. During streaming SFT, we train the model for three epochs using AdamW with a constant learning rate of 1×10^{-4} and a batch size of 512. The maximum length of the user interaction history is set to 200 in both stages. We set the loss coefficients to $\alpha = 0.1$ and $\beta = 1$. To enable classifier-free guidance (CFG) (Ho and Salimans, 2022) through marginal velocity estimation, we randomly drop the conditioning context with probability 5% during training.

At inference time, X-Rec first samples anchors with a temperature of 1.4 and then generates retrieval triggers over 30 denoising steps. We apply CFG with a guidance scale of 3.0 throughout the denoising process.

4.2. Overall Performance

As shown in Figure 4, on our streaming benchmark, X-Rec substantially outperforms most baselines. After controlling for model configuration with the two X-Rec variants, we make the following two main observations.

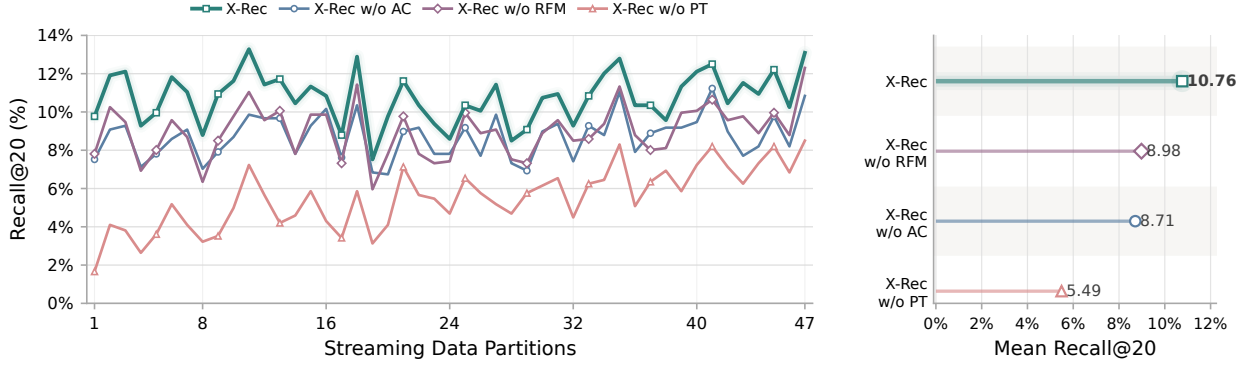


Figure 5 | Ablation results of anchor conditioning (AC), RFM, and pretraining (PT) on the streaming benchmark.

Models	SID-1 Recall@20 (%)	SID-1 Recall@50 (%)
X-Rec w/o AC	28.57	35.77
X-Rec	41.37	53.55

Table 2 | Average Recall@20 and Recall@50 of the first SID digit across all streaming partitions. Recall is defined as the fraction of test examples for which the first SID digit of the ground-truth item is covered by the first SID digits of the generated triggers after quantization.

First, X-Rec achieves slightly higher recall than X-Rec-AR. On average, X-Rec reaches 10.65%, outperforming X-Rec-AR by 0.53% percentage points. Across the 47 streaming partitions, X-Rec performs better than X-Rec-AR on 38 partitions, with one tie. This suggests that, in terms of distribution modeling capacity, FM and SID-AR are comparable. More importantly, as we will show later, X-Rec achieves this performance with substantially higher generation throughput, since its triggers can be generated in parallel rather than decoded sequentially token-by-token.

Second, X-Rec clearly outperforms X-Rec-U2I. X-Rec-U2I achieves an average recall of only 6.98%, trailing X-Rec by 3.67% percentage points. Since X-Rec-U2I relies on a single retrieval embedding, the gap highlights the importance of multi-mode interest modeling. X-Rec addresses this by generating multiple triggers for different interest modes. Moreover, as shown in §4.4.1, even when X-Rec generates only one trigger and is evaluated under the same Recall@1×20 protocol as X-Rec-U2I, it still achieves better performance.

4.3. Ablation Study

We further conduct ablation studies to examine the contributions of several key components, including anchor conditioning, RFM, late interaction, and pretraining.

4.3.1. Effect of Anchor Conditioning

We implement an X-Rec variant without anchor conditioning, denoted as X-Rec w/o AC. In this variant, the model directly generates the recall triggers without the guidance of anchors, and all other settings remain identical to those of X-Rec. As shown in Figure 5, anchor conditioning plays a crucial role in X-Rec. Removing it reduces the averaged Recall@20 from 10.76% to 8.71%, i.e., a dropping of 2.05%

percentage points. Meanwhile, X-Rec w/o AC is inferior to X-Rec on all 47 streaming partitions.

To further understand the role of anchor conditioning, we quantize the generated triggers of X-Rec and X-Rec w/o AC back into SIDs, and examine whether the first SID digit matches that of the ground-truth item. As shown in Table 2, X-Rec achieves substantially higher first-digit recall than X-Rec w/o AC. Specifically, the recall improves from 28.57% to 41.37% at top-20 and from 35.77% to 53.55% at top-50, corresponding to absolute gains of 12.80% and 17.78% percentage points, respectively. These results indicate that anchor conditioning substantially improves the model’s ability to identify the correct coarse semantic region, which in turn leads to better end-to-end retrieval performance.

Overall, anchor conditioning effectively decomposes trigger generation into coarse semantic localization followed by fine-grained refinement. This decomposition simplifies the generation task and consistently improves retrieval performance, while introducing negligible additional computational cost.

4.3.2. Effect of RFM

We implement a variant of X-Rec without RFM, denoted as X-Rec w/o RFM, by replacing RFM with rectified FM in Euclidean space (Liu et al., 2022), while keeping all other settings unchanged. As shown in Figure 5, the w/o RFM variant reaches an average Recall@20 of 8.98%, trailing X-Rec by 1.78% percentage points. X-Rec achieves higher recall on all 47 streaming partitions, indicating that the geometry-aware flow design consistently improves trigger quality.

This improvement comes from explicitly grounding the generation process on the normalized item embedding manifold. Since item embeddings are ℓ_2 -normalized, valid retrieval triggers should lie on the unit hypersphere. RFM incorporates this constraint into the generative dynamics by transporting samples along spherical geodesics and constraining the velocity to the tangent space at each state. As a result, the model can devote its full capacity to learning meaningful movement within the item manifold. In contrast, under Euclidean FM, the velocity field must implicitly handle two coupled objectives: moving toward the target item and correcting deviations from the hypersphere. This additional burden makes the learned dynamics less efficient and less aligned with the normalized retrieval space, leading to less accurate retrieval triggers.

4.3.3. Effect of Late-Interaction

X-Rec uses late interaction to restrict each velocity-field evaluation to the last Transformer layer, i.e., $L' = 1$. Here, we study the performance–efficiency trade-off by varying the number of velocity evaluation layers $L' \in \{1, 7, 14, 28\}$. For all configurations, the number of denoising steps is fixed to 30. We also include X-Rec-AR as a reference. Retrieval performance is measured by the average Recall@20×1 across streaming partitions, while efficiency is measured by the maximum observed generation throughput on a single production-grade GPU in BF16. When benchmarking throughput, each request contains a condition sequence of length 199 and requires the model to generate 20 triggers.

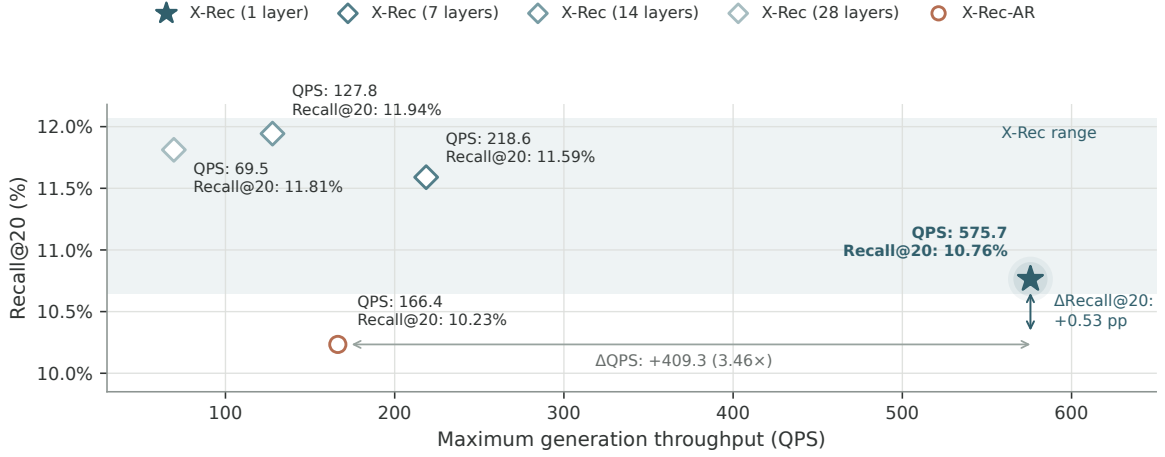


Figure 6 | Performance-throughput trade-off of different models. Each point reports average Recall@20 over all streaming partitions and the maximum generation throughput across the tested batch sizes.

As shown in Figure 6, increasing L' initially improves retrieval performance, with the 14-layer variant achieving the highest mean Recall @20 of 11.94%, which is only 1.05% percentage points higher than that of the 1-layer variant. Further increasing L' to 28, however, leads to a slight performance decline, indicating that using more layers for denoising does not necessarily improve retrieval performance. In contrast, the 1-layer variant is substantially more efficient, achieving 2.63 $\times$, 4.50 $\times$, and 8.28 $\times$ the throughput of the 7-, 14-, and 28-layer variants, respectively. Furthermore, the 1-layer X-Rec outperforms X-Rec-AR in both retrieval performance and generation efficiency, improving mean Recall @20 by 0.53 percentage points and throughput by 3.46 $\times$.

These results show that the one-layer late-interaction design offers the best deployment trade-off: it preserves strong retrieval performance while greatly reducing the repeated Transformer computation during generation.

4.3.4. Effect of Pretraining

To isolate the contribution of pretraining, we compare X-Rec with an otherwise identical variant trained on the streaming benchmark from scratch, i.e., without pretraining. As shown in Figure 5, removing the pretraining stage causes a substantial and persistent performance degradation across all the 47 streaming partitions. The mean Recall @20 decreases from 10.76% to 5.49%, an absolute drop of 5.27% percentage points. Although the non-pretrained variant gradually improves during streaming SFT, it remains consistently underperforms the pretrained model. These results demonstrate that pretraining provides a strong and transferable initialization that cannot be fully recovered through downstream streaming supervision alone, highlighting its critical role in learning robust representations for effective retrieval.

4.4. Analysis

We further analyze X-Rec in terms of trigger breadth, diversity, and the representation loss introduced by SID reconstruction.

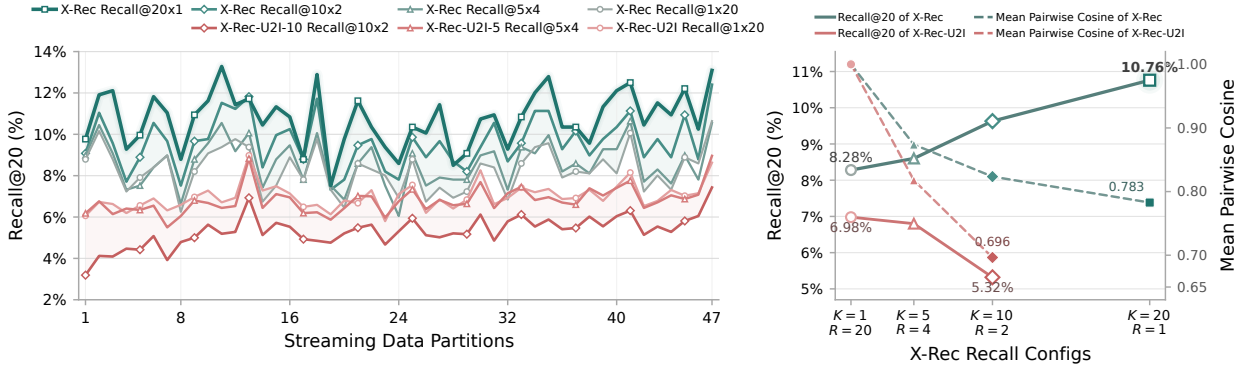


Figure 7 | Performance of different recall configurations of X-Rec and X-Rec-U2I-K.

4.4.1. Retrieval Breadth & Depth

Under a fixed candidate budget, increasing K distributes the budget across more generated triggers, broadening the coverage of user interests, whereas increasing R retrieves more nearest neighbors per trigger, enabling deeper exploration around each interest. To study this breadth–depth trade-off, we evaluate X-Rec under four Recall @20 configurations: 1×20 , 5×4 , 10×2 , and 20×1 . We also compare X-Rec with ComiRec-SA (Cen et al., 2020), a multi-interest U2I baseline that uses self-attention to derive multiple user-interest embeddings. For a fair comparison, ComiRec-SA equips the same context-encoder configuration as X-Rec, therefore we denote its resulting variants as X-Rec-U2I- K (See §A.2 for detailed implementations). We use the average pairwise cosine similarity among the generated triggers as a proxy for interest breadth, with a lower similarity indicating broader coverage. For configurations with $K = 1$, we define this pairwise cosine as 1.

As shown in Figure 7, for X-Rec, Recall @20 improves monotonically with the number of generated triggers K . The average Recall @20 increases from 8.28% under 1×20 to 8.60%, 9.63%, and 10.76% for 5×4 , 10×2 , and 20×1 , respectively. Reallocating the fixed budget of 20 candidates from 1×20 to 20×1 yields an absolute improvement of 2.48% percentage points. This gain is consistent across the evaluation stream: 20×1 outperforms 1×20 on every partition and achieves the best or tied-best result on 45 of the 47 partitions. Meanwhile, the average pairwise cosine similarity decreases as K increases, indicating broader coverage of user interests. Together, these results show that, under a fixed candidate budget, broadening interest coverage with more triggers is more effective than exploring deeper neighborhoods around fewer triggers. This highlights X-Rec’s ability to capture multiple modes of user interest.

X-Rec-U2I- K , however, exhibits a different trend. Although increasing K lowers the average pairwise cosine similarity and appears to broaden interest coverage, its Recall @20 drops substantially. Specifically, the average recall decreases from 6.98% under 1×20 to 5.32% under 10×2 , a drop of 1.66% percentage points. Moreover, under the same configuration, the ComiRec-based variants X-Rec-U2I- K perform substantially worse than X-Rec. Under 10×2 , for example, X-Rec-U2I- K has a lower average pairwise cosine similarity than X-Rec (0.696 vs. 0.823), yet achieves substantially lower recall (5.32% vs. 9.63%). This indicates that the apparent breadth of X-Rec-U2I- K is largely superficial: its interest embeddings are more dispersed but fail to effectively cover distinct user interests. This is because, although X-Rec-U2I- K generates multiple interest embeddings, it remains within the U2I contrastive-learning paradigm rather than explicitly modeling the full target distribution. Moreover, only the interest embedding selected for a given target is optimized during training. As a result, when

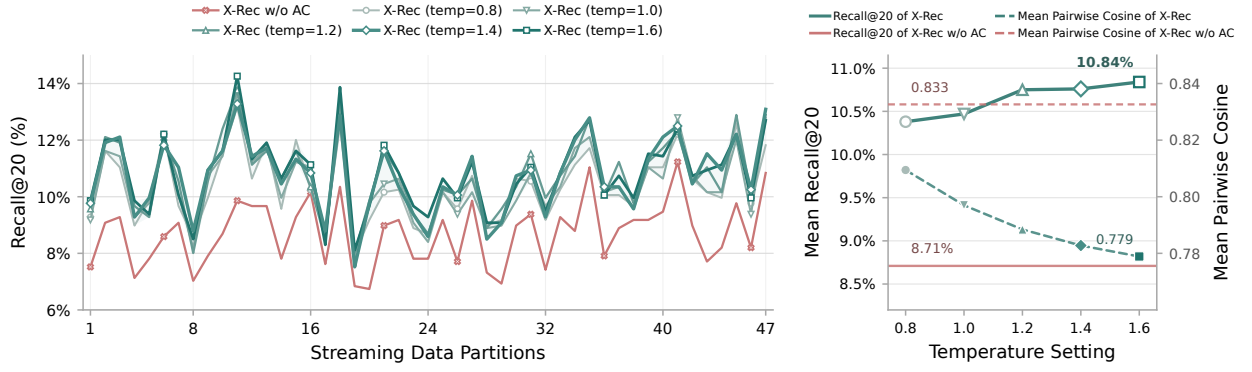


Figure 8 | Effect of anchor-sampling temperature across all streaming partitions.

K increases, the supervision received by each embedding becomes sparser, leading to performance degradation.

4.4.2. Adjusting Diversity with Anchor Temperature

Beyond simplifying the generation task, anchor conditioning provides a natural mechanism for controlling trigger diversity through the sampling temperature of the anchor distribution $p_{\theta}(x^{(a)} | c)$. As the temperature approaches zero, the anchor distribution degenerates into a delta distribution on the most probable anchor. In this case, all generated triggers are concentrated in a narrow semantic region, resulting in low diversity. As the temperature approaches infinity, the anchor distribution converges uniform over all candidate anchors. The generated triggers are then dispersed across different regions, leading to maximal diversity. In this section, we vary the temperature to study the relationship between trigger diversity and recall performance.

As shown in Figure 8, anchor conditioning improves both retrieval performance and trigger diversity. Even at a temperature of 0.8, compared with X-Rec w/o AC, X-Rec reduces the mean pairwise cosine similarity from 0.833 to 0.809, while increasing the mean Recall@20 from 8.71% to 10.38%. Increasing the temperature from 0.8 to 1.6 further decreases the cosine similarity monotonically from 0.809 to 0.779, coupled with an increase in Recall@20 from 10.38% to 10.84%. These results suggest that flattening the anchor distribution encourages the model to explore a broader range of semantic regions, thereby reducing redundancy among generated triggers and improving the coverage of relevant items.

Most recall improvement occurs when increasing the temperature from 0.8 to 1.2, after which the gain gradually saturates. In particular, temperatures of 1.2, 1.4, and 1.6 achieve Recall@20 values of 10.75%, 10.76%, and 10.84%, respectively. Thus, the default temperature of 1.4 provides a near-saturated operating point. Overall, anchor temperature provides an effective mechanism for controlling trigger diversity and balancing concentrated exploitation against broader retrieval coverage.

4.4.3. Reconstructed Targets Expose Representation Loss

Here, we examine the performance degradation caused by SID quantization errors. Instead of supervising X-Rec with the original continuous item embeddings, we construct a variant that uses SID-reconstructed embeddings as training targets. Specifically, each target item is first mapped to its

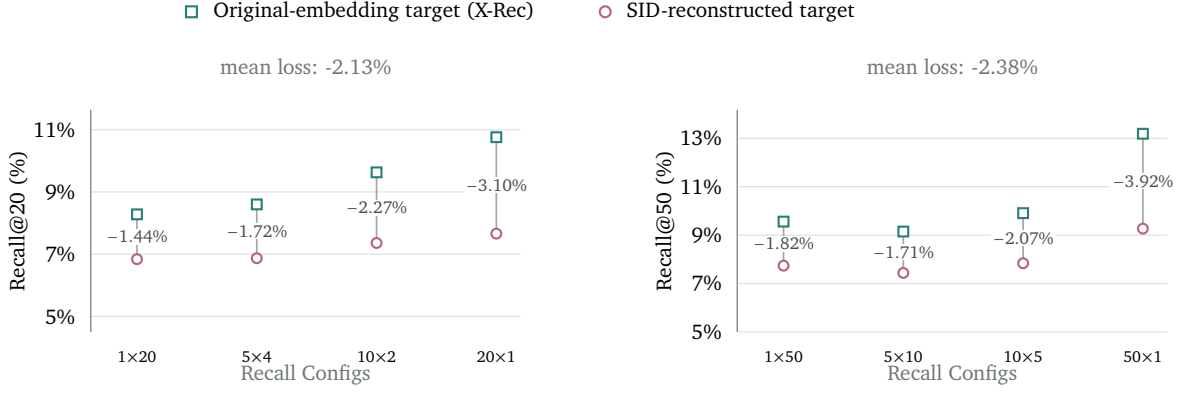


Figure 9 | Effect of using SID-reconstructed embeddings as SFT targets under different recall configurations. Each marker reports the average performance over all the streaming partitions. Numbers inside the connectors denote absolute losses relative to the original-embedding target, in percentage points.

SID, and the SID is then decoded through the RQ codebooks to obtain the reconstructed training target for SFT.

As shown in Figure 9, replacing original item embeddings with SID-reconstructed targets consistently hurts performance across different recall configurations. Concretely, the average recall drops by 1.44%–3.10% percentage points for Recall @20 and by 1.71%–3.92% points for Recall @50, with average gaps of 2.13% and 2.38% points, respectively. The degradation is largest for 20x1 and 50x1, where each generated trigger retrieves only its nearest ANN neighbor. In this setting, a small shift in the target representation can directly change the nearest-neighbor result. This suggests that SID reconstruction distorts the fine-grained neighborhoods of the original embedding space and introduces a non-negligible information loss. X-Rec avoids this reconstruction bottleneck by learning directly from continuous item-embedding targets.

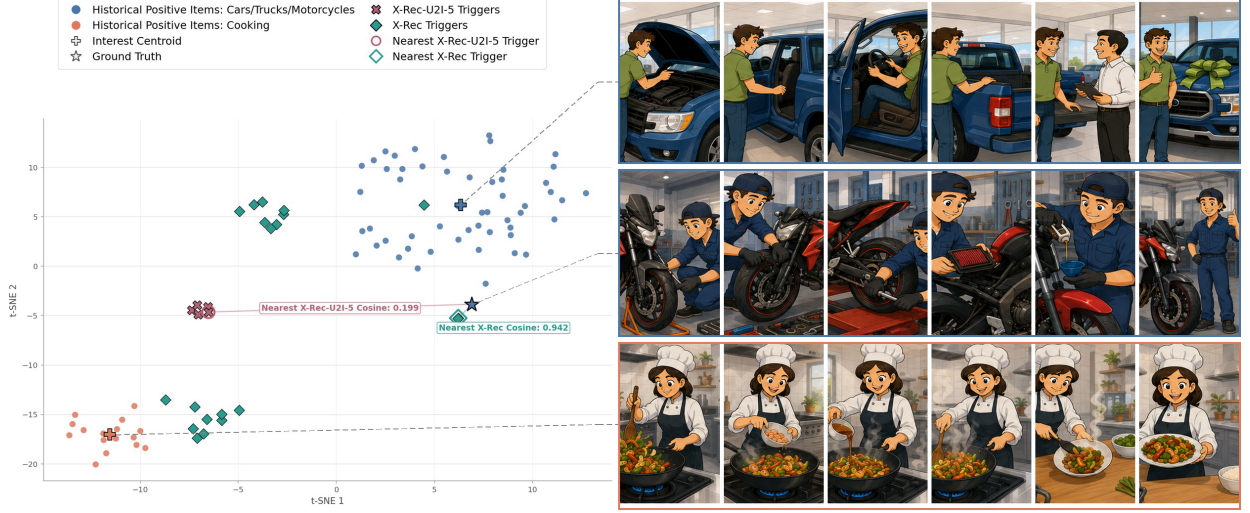
4.5. Case Study

In this section, we present case studies of the generated results. Specifically, we use t-SNE (Cai and Ma, 2022) to project the 128-dimensional embeddings into two dimensions and visualize the user’s historical positive items together with the triggers generated by X-Rec and X-Rec-U2I-K. Representative examples¹⁰ are shown in Figure 10, and additional cases are provided in §A.3.

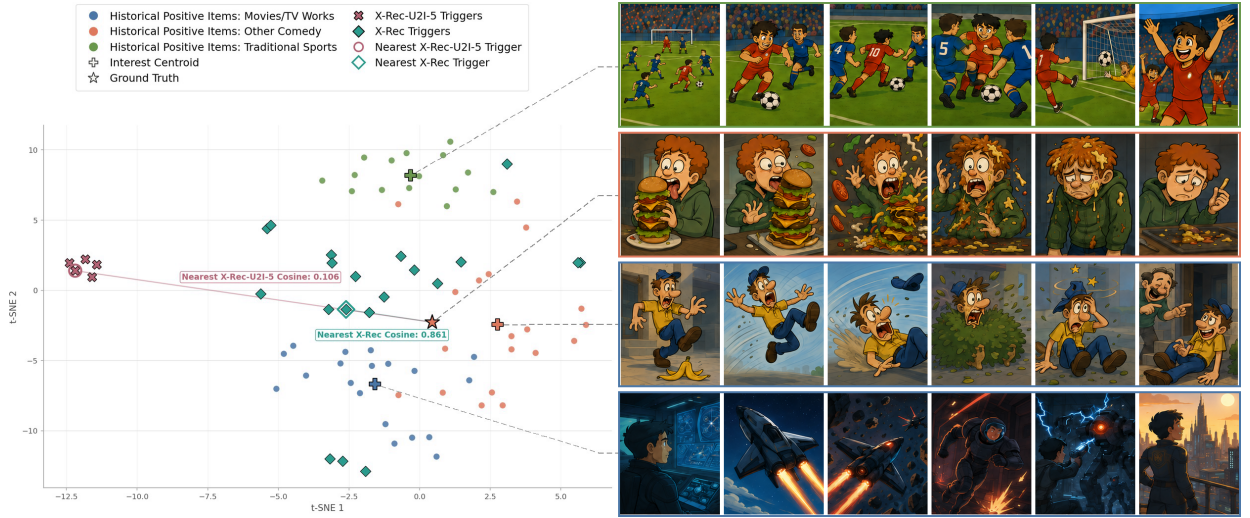
As shown, user interests, represented by items with historical positive interactions, often exhibit a multi-mode structure. For example, in Figure 10a, the positively interacted items in the user’s history mainly fall into two categories: Cars/Trucks/Motorcycles and Cooking, forming two distinct modes in the video embedding space. The user’s next positively interacted ground-truth video is also likely to lie near one of these modes: in this case, it belongs to the Cars/Trucks/Motorcycles category. X-Rec closely approximates this distribution. The manifold formed by its generated triggers aligns well with that of the historical positive items, and the closest cosine similarity between the generated triggers and the ground-truth item reaches 0.942. In contrast, X-Rec-U2I-5 produces an averaged representation. Although it generates five distinct triggers, its manifold collapses to a single mode

¹⁰For illustrative purposes and to meet compliance requirements, all example videos are AI-generated based on the corresponding video categories and do not correspond to real user videos.

located in the valley between the user’s two interest modes. As a result, it fails to capture the user’s multi-mode interest distribution, and its closest cosine similarity to the ground-truth item is only 0.199. This visualization further demonstrates that, compared with U2I methods, X-Rec is able to model diverse user interests more effectively .



(a) Active interests: Cars/Trucks/Motorcycles and Cooking. The ground-truth item belongs to Cars/Trucks/Motorcycles.



(b) Active interests: Movies/TV works, Other Comedy, and Traditional Sports. The ground-truth item belongs to Other Comedy.

Figure 10 | Visualization of historical positive items and triggers generated by X-Rec and X-Rec-U2I-5.

5. Online Results

We deploy X-Rec as a new retrieval source for vertical-content recommendation on TikTok. In the first launch (V1, i.e., X-Rec w/o anchor conditioning), X-Rec incorporated RFM and late interaction, achieving strong generation quality with high computational efficiency. In the second launch (V2, i.e., X-Rec), we further introduced anchor conditioning, which led to additional gains in retrieval performance. As shown in §5, X-Rec has yielded significant improvements in both vertical engagement

(+4.1484%) and general engagement metrics (+0.0111%). This result highlight the real-world impacts of our proposed retrieval method.

Metrics	V1	V2	Total
Vertical Engagement	+2.3779%	+1.7705%	+4.1484%
General Engagement	Not significant	+0.0111%	+0.0111%

Table 3 | The online A/B testing results of X-Rec. Since V1 serves as the baseline for V2 in the A/B test, the gains from the two tests can be safely combined to estimate the total improvement. Unless marked as “Not significant”, all results in the table are statistically significant.

6. Conclusion & Future Work

In this work, we presented X-Rec, an FM-based generative retrieval framework that directly models the recommendation distribution in the continuous item embedding space. Unlike U2I methods, which compress user interests into one or a few deterministic retrieval points, X-Rec generates multiple stochastic triggers to capture diverse interest modes. Unlike SID-AR methods, X-Rec avoids quantizing item embeddings into discrete identifiers and eliminates token-by-token decoding. To enable effective and efficient continuous generation at industrial scale, X-Rec incorporates three complementary designs: anchor conditioning decomposes generation into coarse semantic-region selection and fine-grained refinement; RFM aligns the transport paths with the hyperspherical geometry of normalized item embeddings; and the late-interaction DiT reuses cached condition representations while restricting repeated velocity-field estimation to the final Transformer layer.

Offline experiments on the streaming evaluation benchmark demonstrate that X-Rec achieves a favorable balance among retrieval quality, interest coverage, and generation efficiency. It outperforms both the U2I and SID-AR variants while achieving $3.46\times$ higher generation throughput than the latter. In online experiments, two consecutive launches on TikTok’s vertical content yield significant improvements on business metrics. These results demonstrate the potential of continuous generative retrieval as an expressive, efficient, and practical complement to existing industrial retrieval paradigms.

Several directions remain open for future work. First, the current sampler requires multiple denoising steps, motivating the exploration of few-step distillation, adaptive numerical solvers, and hardware-aware optimization to reduce end-to-end serving costs. Second, although the late-interaction design is computationally efficient, its scalability remains to be fully explored. Under strict serving-throughput constraints, practical systems must balance retrieval quality against generation cost. Future work could investigate how late interaction scales with larger backbones and deeper denoising modules, as well as adaptive computation mechanisms that dynamically balance quality and efficiency. Third, reward-aligned post-training, such as reinforcement learning, represents a promising direction for continuous generative retrieval. It may enable X-Rec to optimize for long-term user satisfaction and content relevance, objectives that are difficult to capture through supervised learning alone.

References

Ricci, F., L. Rokach, and B. Shapira (2010). “Introduction to recommender systems handbook”. In: *Recommender systems handbook*. Springer, pp. 1–35.

- Covington, P., J. Adams, and E. Sargin (2016). “Deep Neural Networks for YouTube Recommendations”. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, pp. 191–198. DOI: [10.1145/2959100.2959190](https://doi.org/10.1145/2959100.2959190).
- Uribe, B., M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle (2016). “Neural Autoregressive Distribution Estimation”. In: *Journal of Machine Learning Research* 17.205, pp. 1–37. URL: <https://jmlr.org/papers/v17/16-272.html>.
- Loshchilov, I. and F. Hutter (2017). “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101*.
- Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). “Attention is all you need”. In: *NeurIPS*. Vol. 30.
- Blanc, G. and S. Rendle (2018). “Adaptive Sampled Softmax with Kernel Based Sampling”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 590–599. URL: <https://proceedings.mlr.press/v80/blanc18a.html>.
- Kang, W.-C. and J. J. McAuley (2018). “Self-Attentive Sequential Recommendation”. In: *IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, pp. 197–206. DOI: [10.1109/ICDM.2018.00035](https://doi.org/10.1109/ICDM.2018.00035). URL: <https://doi.org/10.1109/ICDM.2018.00035>.
- Oord, A. v. d., Y. Li, and O. Vinyals (2018). “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748*.
- Stern, M., N. Shazeer, and J. Uszkoreit (2018). “Blockwise Parallel Decoding for Deep Autoregressive Models”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., pp. 10107–10116.
- Csikós, M., N. H. Mustafa, and A. Kupavskii (2019). “Tight Lower Bounds on the VC-dimension of Geometric Set Systems”. In: *Journal of Machine Learning Research* 20.81, pp. 1–8. URL: <https://jmlr.org/papers/v20/18-719.html>.
- Yu, Z., J. Lian, A. Mahmood, G. Liu, and X. Xie (2019). “Adaptive user modeling with long and short-term preferences for personalized recommendation”. In: *IJCAI*. Vol. 7, pp. 4213–4219.
- Cen, Y., J. Zhang, X. Zou, C. Zhou, H. Yang, and J. Tang (2020). “Controllable Multi-Interest Framework for Recommendation”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 2942–2951. DOI: [10.1145/3394486.3403344](https://doi.org/10.1145/3394486.3403344). URL: <https://doi.org/10.1145/3394486.3403344>.
- Chen, T., S. Kornblith, M. Norouzi, and G. Hinton (2020). “A Simple Framework for Contrastive Learning of Visual Representations”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 1597–1607. URL: <https://proceedings.mlr.press/v119/chen20j.html>.
- Khattab, O. and M. Zaharia (2020). “Colbert: Efficient and effective passage search via contextualized late interaction over bert”. In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 39–48.
- Radford, A., J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever (2021). “Learning Transferable Visual Models From Natural Language Supervision”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8748–8763. URL: <https://proceedings.mlr.press/v139/radford21a.html>.
- Ricci, F., L. Rokach, and B. Shapira (2021). “Recommender systems: Techniques, applications, and challenges”. In: *Recommender systems handbook*, pp. 1–35.
- Cai, T. T. and R. Ma (2022). “Theoretical foundations of t-sne for visualizing high-dimensional clustered data”. In: *Journal of Machine Learning Research* 23.301, pp. 1–54.

- Ho, J. and T. Salimans (2022). “Classifier-Free Diffusion Guidance”. In: *arXiv preprint arXiv:2207.12598*. arXiv: 2207.12598 [cs.LG]. URL: <https://arxiv.org/abs/2207.12598>.
- Lee, D., C. Kim, S. Kim, M. Cho, and W.-S. Han (2022). “Autoregressive Image Generation Using Residual Quantization”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 11513–11522. DOI: 10.1109/CVPR52688.2022.01123.
- Lipman, Y., R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le (2022). “Flow matching for generative modeling”. In: *arXiv preprint arXiv:2210.02747*.
- Liu, X., C. Gong, and Q. Liu (2022). “Flow straight and fast: Learning to generate and transfer data with rectified flow”. In: *arXiv preprint arXiv:2209.03003*.
- Kwon, W., Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica (2023). “Efficient Memory Management for Large Language Model Serving with PagedAttention”. In: *Proceedings of the 29th Symposium on Operating Systems Principles*. ACM, pp. 611–626. DOI: 10.1145/3600006.3613165. URL: <https://doi.org/10.1145/3600006.3613165>.
- Lipman, Y., R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le (2023). “Flow Matching for Generative Modeling”. In: *The Eleventh International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=PqvMRDCJT9t>.
- Peebles, W. and S. Xie (2023). “Scalable diffusion models with transformers”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, pp. 4172–4182.
- Rajput, S., N. Mehta, A. Singh, R. Hulikal Keshavan, T. Vu, L. Heldt, L. Hong, Y. Tay, V. Tran, J. Samost, M. Kula, E. Chi, and M. Sathiamoorthy (2023). “Recommender Systems with Generative Retrieval”. In: *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., pp. 10299–10315. DOI: 10.52202/075280-0452. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/20dcab0f14046a5c6b02b61da9f13229-Paper-Conference.pdf.
- Yang, Z., J. Wu, Z. Wang, X. Wang, Y. Yuan, and X. He (2023). “Generate What You Prefer: Reshaping Sequential Recommendation via Guided Diffusion”. In: *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., pp. 24247–24261. DOI: 10.52202/075280-1054. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/4c5e2bcbf21bdf40d75fddad0bd43dc9-Paper-Conference.pdf.
- Chen, J., L. Chi, B. Peng, and Z. Yuan (2024). “Hllm: Enhancing sequential recommendations via hierarchical large language models for item and user modeling”. In: *arXiv preprint arXiv:2409.12740*.
- Chen, R. T. Q. and Y. Lipman (2024). “Flow Matching on General Geometries”. In: *The Twelfth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=g7ohD1TITL>.
- Su, J., M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu (2024). “Roformer: Enhanced transformer with rotary position embedding”. In: *Neurocomputing* 568, p. 127063.
- Wang, Y., J. Xun, M. Hong, J. Zhu, T. Jin, W. Lin, H. Li, L. Li, Y. Xia, Z. Zhao, and Z. Dong (2024). “EAGER: Two-Stream Generative Recommender with Behavior-Semantic Collaboration”. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, pp. 3245–3254. DOI: 10.1145/3637528.3671775. URL: <https://doi.org/10.1145/3637528.3671775>.
- Yue, Z., Y. Wang, Z. He, H. Zeng, J. J. McAuley, and D. Wang (2024). “Linear Recurrent Units for Sequential Recommendation”. In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. ACM, pp. 930–938. DOI: 10.1145/3616855.3635760. URL: <https://doi.org/10.1145/3616855.3635760>.
- Zhang, C., S. Wu, H. Zhang, T. Xu, Y. Gao, Y. Hu, and E. Chen (2024). “Notellm: A retrievable large language model for note recommendation”. In: *Companion Proceedings of the ACM Web Conference 2024*, pp. 170–179.

- Deng, J., S. Wang, K. Cai, L. Ren, Q. Hu, W. Ding, Q. Luo, and G. Zhou (2025). “Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment”. In: *arXiv preprint arXiv:2502.18965*.
- Lai, C.-H., Y. Song, D. Kim, Y. Mitsufuji, and S. Ermon (2025). “The principles of diffusion models”. In: *arXiv preprint arXiv:2510.21890*.
- Qwen3, A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu (2025). “Qwen3 Technical Report”. In: *arXiv preprint arXiv:2505.09388*.
- Shi, T., C. Shen, W. Yu, S. Nie, C. Li, X. Zhang, M. He, Y. Han, and J. Xu (2025). *LLaDA-Rec: Discrete Diffusion for Parallel Semantic ID Generation in Generative Recommendation*. arXiv: 2511.06254 [cs.IR]. URL: <https://arxiv.org/abs/2511.06254>.
- Wu, C., J. Li, J. Zhou, J. Lin, K. Gao, K. Yan, S.-m. Yin, S. Bai, X. Xu, Y. Chen, et al. (2025). “Qwen-image technical report”. In: *arXiv preprint arXiv:2508.02324*.
- Zhou, G., J. Deng, J. Zhang, K. Cai, L. Ren, Q. Luo, Q. Wang, Q. Hu, R. Huang, S. Wang, W. Ding, W. Li, X. Luo, X. Wang, Z. Cheng, Z. Zhang, B. Zhang, B. Wang, C. Ma, C. Song, C. Wang, D. Wang, D. Meng, F. Yang, F. Zhang, F. Jiang, F. Zhang, G. Wang, G. Zhang, H. Li, H. Hu, H. Lin, H. Cheng, H. Cao, H. Wang, J. Huang, J. Chen, J. Liu, J. Jia, K. Gai, L. Hu, L. Zeng, L. Yu, Q. Wang, Q. Zhou, S. Wang, S. He, S. Yang, S. Yang, S. Huang, T. Wu, T. He, T. Gao, W. Yuan, X. Liang, X. Xu, X. Liu, Y. Wang, Y. Wang, Y. Liu, Y. Song, Y. Zhang, Y. Wu, Y. Zhao, and Z. Liu (2025a). *OneRec Technical Report*. arXiv: 2506.13695 [cs.IR]. URL: <https://arxiv.org/abs/2506.13695>.
- Zhou, G., H. Hu, H. Cheng, H. Wang, J. Deng, J. Zhang, K. Cai, L. Ren, L. Ren, L. Yu, et al. (2025b). “Onerec-v2 technical report”. In: *arXiv preprint arXiv:2508.20900*.
- Fu, K., T. Zhang, S. Xiao, Z. Wang, X. Zhang, C. Zhang, Y. Yan, J. Zheng, X. Kong, S. Zhang, et al. (2026). “Forge: Forming semantic identifiers for generative retrieval in industrial datasets”. In: *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 8858–8869.
- Hou, Y., H. Kim, C. M. Ju, E. Escoto, N. Shah, and J. McAuley (2026). *Expressiveness Limits of Autoregressive Semantic ID Generation in Generative Recommendation*. arXiv: 2605.06331 [cs.IR]. URL: <https://arxiv.org/abs/2605.06331>.
- Kumar, A. and V. M. Patel (2026). “Learning on the manifold: Unlocking standard diffusion transformers with representation encoders”. In: *arXiv preprint arXiv:2602.10099*.
- Qwen3.5 (2026). *Qwen3.5: Accelerating Productivity with Native Multimodal Agents*. URL: <https://qwen.ai/blog?id=qwen3.5>.

7. Contributors

Within each role, the contributors are listed in alphabetical order by their first names. Contributors marked with * had left the team before publication.

Corresponding Author. Kun Xǔ (daniel.chen28@bytedance.com)

Project Leader. Zhiwei Wang (heyisen@bytedance.com)

Core Contributors. Chenglei Shen, Chenzhe Huang, Dong Jiang, Hongjie Gao, Jue Zhang, Kun Xú, Lincan Cai, Nan Zhuang, Pan Zhang, Shi Chen, Shunchi Zhang, Xiaoyu Ye, Yang Jin, Yu Zhang, Zhenwei An, Zhongtao Jiang

Contributors. Ao Xu, Chenghao Liu*, Han Xia, Hantian Su, Haoyang Yao, Huixiu Jin, Jialin Li, Jianyuan Bo, Jie Li, Jingwei Li, Junwen Chen, Kailin Ding, Kangcheng Luo, Lin Chen, Linjie Wang, Mengshi Chen, Ming Zhang, Rengzhi Wang, Silong Yu, Song Jin*, Tingyan Li, Weixin Wang, Xin Chen, Xuhang Xiao, Yiming Jia, Yinong Lin, Yifeng Yao, YongKang Zhang, Yu Zhang*, Yuxiao Zhang, Zhangyi Chen, Zhao Wang, Zheng Li, Zixuan Wang

A. Appendix

A.1. Complete Results on Streaming Benchmark

Table 4 and Table 5 summarize the Recall @20 and Recall @50 results across different models. For U2I methods (SASRec, LRURec, DreamRec, and X-Rec-U2I), which produce a single retrieval embedding, we compute Recall @ K under the $1 \times K$ setting. For SID-based methods (TIGER, EAGER, LATTE, LLaDARec, and X-Rec-AR) and X-Rec, which support multiple retrieval triggers, we use the $K \times 1$ setting.

Data Partition	X-Rec	X-Rec-U2I	X-Rec-AR	TIGER	LATTE	LLaDARec	DreamRec	EAGER	SASRec	LRURec
1	9.7700	6.0645	9.8633	2.6562	5.9473	5.4004	4.7266	2.8027	0.9375	1.0938
2	11.9100	6.7285	10.7422	4.9609	5.8398	6.6309	4.8730	4.2383	1.4160	1.3281
3	12.1100	6.6309	10.9375	6.2891	6.7480	7.0117	3.9844	5.7227	0.7129	0.9082
4	9.2800	6.1914	9.1797	4.5703	6.0449	6.5430	4.2871	3.4766	0.8691	0.9961
5	9.9600	6.5625	9.6680	4.8730	5.8496	6.2695	3.9844	3.9551	0.9375	0.9961
6	11.8200	6.9043	11.9141	6.5332	4.1992	7.2656	5.3320	5.5176	1.1328	1.2402
7	11.0400	6.3184	10.9375	6.2695	7.1094	6.4062	3.6914	5.3027	0.6250	0.8398
8	8.7900	6.5918	8.1055	5.3516	6.5723	6.6113	4.2090	3.8379	0.8496	0.9766
9	10.9400	6.9727	10.6445	5.5273	6.5527	6.8555	3.9258	4.5996	0.9082	1.0254
10	11.6200	7.2754	13.2812	6.9043	8.1641	6.8652	4.4238	5.5469	1.1230	1.4453
11	13.2800	6.7090	12.6953	7.5293	8.8965	8.1641	4.1406	6.6504	0.9180	1.1328
12	11.4300	6.9336	11.7188	5.7910	7.0898	7.1094	3.7695	5.3809	0.8691	1.2109
13	11.7200	8.9844	10.9375	6.9922	8.5547	8.6230	6.9727	4.8926	1.6309	1.7578
14	10.4500	7.2754	9.7656	6.0156	7.3242	7.1680	4.3848	6.1035	0.9375	1.0645
15	11.3300	7.4902	11.3281	6.7969	8.3594	7.2754	4.7852	6.4746	1.4258	1.4355
16	10.8400	7.1387	10.3516	5.6348	6.6992	6.5039	4.4434	4.5020	1.1816	1.4844
17	8.7900	6.4941	8.3984	5.3320	6.2695	5.9570	3.6621	4.2188	0.8594	1.1230
18	12.8900	6.5918	11.0352	5.4688	6.9629	6.9531	4.1895	5.4883	0.9180	1.1133
19	7.5200	6.1328	7.2266	5.0781	6.1328	5.9961	3.4668	4.2383	0.8301	1.1328
20	9.7700	6.7969	9.5703	5.0488	6.7871	5.9961	3.6914	4.6387	0.8594	0.8887
21	11.6200	6.6699	10.1562	6.2305	7.7344	6.3965	3.9941	5.3711	1.0449	1.1426
22	10.3500	7.3047	8.8867	6.1230	7.8613	6.9141	4.5215	6.0352	1.0156	1.1914
23	9.3800	5.8008	9.1797	4.6973	6.2207	5.5859	3.6230	3.9062	0.7031	0.8789
24	8.5900	7.1094	8.7891	5.5762	6.8359	6.0547	3.9258	4.4727	1.1621	1.2109
25	10.3500	7.5488	11.4258	5.2930	7.0801	6.8848	4.7852	5.3223	1.2207	1.3965
26	10.0600	6.2012	10.2539	4.4727	6.1523	5.9473	3.7012	3.8867	0.8203	0.9082
27	11.4300	6.8457	9.9609	4.4336	5.9375	6.1230	3.7109	4.8828	0.9863	1.1816
28	8.5000	6.4062	9.3750	3.9844	4.6875	5.8398	3.6230	3.9844	0.7227	0.8594
29	9.0800	6.8359	7.9102	4.1504	6.3672	5.9082	3.9746	4.3750	0.9766	1.1426
30	10.7400	8.2617	9.8633	4.8340	7.0703	6.9043	5.2637	4.4141	1.3867	1.4844
31	10.9400	6.6211	10.0586	4.5605	6.7871	5.7910	3.7891	5.5566	0.9668	1.2012
32	9.2800	6.9238	8.0078	3.7500	6.4844	6.2988	4.1797	4.3359	0.7422	1.0449
33	10.8400	7.4219	10.6445	5.0293	6.8848	6.6992	5.1855	4.5898	1.2891	1.6309
34	12.0100	7.1973	10.3516	5.1172	7.2168	5.9277	3.9160	5.7520	1.0352	1.3379
35	12.7900	7.3633	12.0117	4.9609	7.3340	7.2363	4.1992	6.1328	0.8984	1.0449
36	10.3500	6.8652	8.5938	4.1797	6.4062	5.8496	3.6426	4.0723	0.9180	1.0742
37	10.3500	6.9238	9.1797	4.4727	6.4746	5.7813	4.0039	5.3223	0.8789	1.1621
38	9.5700	7.3242	9.2773	5.1855	8.1055	7.4121	5.1172	5.2930	1.2207	1.3184
39	11.3300	6.7773	10.5469	4.1602	6.8652	6.3086	3.9648	4.8340	0.9863	1.2500
40	12.1100	7.4707	10.5469	5.5469	7.9199	6.3477	4.2188	5.8105	1.2695	1.4844
41	12.5000	8.1543	12.4023	5.6250	8.2227	7.3535	5.2148	6.0742	1.2598	1.2402
42	10.4500	6.5332	10.2539	4.1406	6.7871	6.0938	3.6133	5.0098	0.8398	1.0352
43	11.5200	6.7773	10.5469	4.5898	7.3730	6.2402	4.1406	4.6094	0.9473	0.9961
44	10.9400	7.3145	9.6680	4.9316	7.3633	5.7520	3.8184	5.1855	1.1621	1.2598
45	12.2100	7.0312	12.1094	5.3418	7.8027	6.0156	4.4043	6.0449	1.2988	1.2207
46	10.2500	7.1484	10.0586	4.5801	4.1016	6.4258	4.5215	4.1211	1.0645	1.2012
47	13.0900	8.6328	12.5977	5.2246	6.6113	7.2461	6.2695	4.6289	1.5918	1.9727

Table 4 | Complete Recall @20 results on the streaming benchmark. The X-Rec column is highlighted with light shading.

Data Partition	X-Rec	X-Rec-U2I	X-Rec-AR	TIGER	LATTE	LLaDARec	DreamRec	EAGER	SASRec	LRURec
1	12.0020	8.4277	12.2070	3.4863	6.9824	6.0840	4.8730	3.6523	1.5820	1.7773
2	12.7344	9.1309	13.7695	6.4746	7.4316	7.5879	5.1660	5.8105	2.4316	2.3730
3	13.6523	9.3750	14.3555	7.9687	8.3984	8.2520	4.2578	7.8906	1.4941	1.7969
4	11.7578	8.7500	13.5742	6.5527	7.7148	7.7832	4.5996	5.1855	1.5039	1.7383
5	11.8164	9.0723	13.3789	6.9238	7.5293	7.2070	4.1602	5.4492	1.8945	2.0898
6	14.4629	9.4043	15.2344	8.3887	5.4688	8.8086	5.4688	7.5586	2.1777	2.3340
7	12.3242	8.8965	14.0625	8.2812	8.8867	7.5977	3.9355	7.3242	1.3574	1.6504
8	12.3535	9.1797	11.9141	7.1680	8.1934	7.7441	4.4043	5.6348	1.4648	1.7285
9	12.2461	9.3848	13.6719	7.3242	8.3301	8.1543	4.1797	6.3086	1.6113	1.7773
10	13.8672	9.8730	16.0156	9.0723	10.2246	8.5352	4.5410	7.8418	2.0605	2.5293
11	14.4043	9.3750	16.6992	9.9121	11.2012	9.4824	4.3945	9.2090	1.6797	2.0312
12	13.2812	9.6582	14.9414	8.3398	9.2578	8.6328	4.0137	7.5586	1.6211	2.1191
13	15.0195	11.4746	13.2812	9.1797	10.5469	10.0293	7.0801	6.7773	2.5488	2.7734
14	13.8281	10.3809	13.8672	7.8809	9.1016	8.5352	4.5801	8.5156	1.9434	2.2168
15	14.4727	10.1758	14.0625	8.8867	10.5371	8.8281	5.0293	8.7695	2.4219	2.5000
16	13.0762	9.9805	13.6719	7.7539	8.8574	7.8613	4.6094	6.8945	2.1973	2.7051
17	11.2988	9.1504	11.4258	6.9434	7.9883	7.0508	3.8477	5.8887	1.6504	1.9434
18	13.6719	9.2480	14.3555	7.5000	8.8574	8.3594	4.4727	7.4414	1.6895	2.0312
19	10.9180	8.4766	9.2773	6.7285	7.8711	7.1094	3.6426	5.9863	1.5723	1.8652
20	11.8066	9.0332	11.8164	6.9531	8.5352	7.2363	3.8867	6.6309	1.6406	1.7090
21	13.1250	9.7754	13.1836	8.2129	9.6875	7.8125	4.3164	7.8613	2.1582	2.2266
22	13.0078	9.7949	11.2305	8.1738	9.9414	8.5156	4.7559	8.3594	1.9531	2.0996
23	10.9668	8.6523	12.5000	6.4258	7.8027	6.5527	3.7891	5.5371	1.3965	1.7090
24	11.9043	9.4727	10.9375	7.3438	8.8965	7.2266	4.1602	6.2598	2.1777	2.1484
25	13.1348	10.6445	14.9414	7.2363	8.9453	7.9492	4.9805	7.3145	2.3047	2.5488
26	10.8301	8.7402	14.0625	5.9766	7.7441	6.9531	3.9160	5.4590	1.8164	1.9238
27	11.8652	9.7266	12.6953	6.2012	7.5977	7.3926	3.9746	6.6211	1.9043	2.1191
28	10.9863	8.9062	13.0859	5.3906	6.2988	6.6699	3.7793	5.5078	1.3672	1.7578
29	11.5137	9.5410	10.3516	5.6836	8.1250	6.9336	4.2383	5.8789	1.9336	2.1387
30	12.9199	11.1914	12.7930	6.6113	8.9062	7.9687	5.5176	6.5527	2.4316	2.7930
31	12.0898	9.2383	13.0859	6.1133	8.5938	6.7676	4.0625	7.4609	1.6406	2.2559
32	11.9531	9.7168	11.4258	5.2637	8.1836	7.3535	4.3848	6.0352	1.4062	1.9336
33	12.4805	10.3320	13.9648	6.5137	8.5547	7.7637	5.4199	6.1719	2.3145	2.8516
34	12.8223	10.5078	13.6719	7.0020	9.3945	7.3242	4.1406	7.9395	2.1973	2.5879
35	13.3301	10.4102	15.1367	7.2559	9.4727	8.3984	4.5508	8.1152	1.8457	2.1191
36	11.7969	9.6484	11.3281	5.7910	7.9687	7.0605	3.8184	5.7227	1.6309	1.9238
37	12.5488	9.7949	12.3047	6.1328	8.3008	7.0020	4.1504	7.2266	1.6602	2.3730
38	12.9492	9.6094	12.5000	6.7871	9.6973	8.2422	5.3906	7.1289	2.0801	2.2266
39	12.4805	9.8535	13.9648	5.7031	8.8672	7.3730	4.2676	6.6309	1.9141	2.2070
40	13.7695	10.6543	13.2812	6.9824	9.8340	7.5391	4.5020	7.9785	2.4609	2.7539
41	13.9453	10.6641	14.7461	7.5879	10.4688	8.7207	5.4883	8.2422	2.0996	2.1582
42	12.2461	9.1895	12.6953	5.7910	8.8672	7.3438	3.8281	7.0605	1.6895	1.9336
43	12.6172	9.4824	13.5742	6.2988	9.3359	7.2852	4.2578	6.4160	1.7578	2.0996
44	13.1348	10.0977	12.5977	6.6309	9.2383	7.0996	4.1016	7.4121	2.2363	2.3535
45	13.4863	9.9707	15.3320	7.1777	9.8242	7.1582	4.6094	8.1250	2.1875	2.2656
46	12.1289	9.4824	12.3047	5.9766	5.5566	7.5586	4.7070	5.9277	1.8555	2.1289
47	14.0039	11.2207	15.5273	6.7969	8.2617	8.3789	6.5332	6.3281	2.5098	3.1543

Table 5 | Complete Recall @50 results on the streaming benchmark. The X-Rec column is highlighted with light shading.

A.2. Architectures of X-Rec-AR and X-Rec-U2I-K

X-Rec, X-Rec-AR, and X-Rec-U2I-K share the same Qwen3-0.6B backbone, user context, and item-representation space. They differ only in how retrieval triggers are obtained.

X-Rec-AR. X-Rec-AR adopts an autoregressive architecture to generate the target SID. Specifically, it uses the context representation $s^{(m)}$ as a prepended prompt and models the distribution over the target 4-token SID sequence with a causal language-modeling head. The model is trained with a cross-entropy loss. During inference, we apply beam search to generate high-probability SIDs and reconstruct the retrieval triggers using the RQ codebooks.

X-Rec-U2I- K . X-Rec-U2I- K adopts the self-attentive multi-interest extraction module from ComiRec-SA (Cen et al., 2020) to derive K user-interest embeddings from the user context. Given the last-layer hidden representation $s^{(h_L)}$ of the context, we introduce learnable query vectors $q_k \in \mathbb{R}^{d_m}$, $k = 1, \dots, K$, to extract multiple interest triggers:

$$t_k = W_t \left(s^{(h_L)}^\top \text{softmax} \left(s^{(h_L)} q_k \right) \right) \in \mathbb{R}^{d_e}, \quad (34)$$

where W_t is a learnable projection matrix.

During training, the original ComiRec-SA implementation uses only the winning trigger, i.e., the trigger closest to the target item $\hat{x}^{(e)}$, to construct the contrastive learning loss:

$$\mathcal{L} = -\log \frac{\exp \left(\frac{t_{k^*}^\top \hat{x}^{(e)}}{\tau} \right)}{\exp \left(\frac{t_{k^*}^\top \hat{x}^{(e)}}{\tau} \right) + \sum_{x' \in \mathcal{X}_{\text{neg}}} \exp \left(\frac{t_{k^*}^\top \hat{x}'^{(e)}}{\tau} \right)}, \quad k^* = \arg \max_k t_k^\top \hat{x}^{(e)}. \quad (35)$$

However, we observe that this strategy is highly unstable and leads to poor performance on our streaming benchmark. We therefore adopt an improved variant. For each candidate item, including both the target item and negative items, we first select its nearest interest trigger and then compute the contrastive loss using the similarity between each item embedding and its corresponding nearest trigger. The improved loss is defined as:

$$\mathcal{L} = -\log \frac{\exp \left(\frac{t_{k_x}^\top \hat{x}^{(e)}}{\tau} \right)}{\exp \left(\frac{t_{k_x}^\top \hat{x}^{(e)}}{\tau} \right) + \sum_{x' \in \mathcal{X}_{\text{neg}}} \exp \left(\frac{t_{k_{x'}}^\top \hat{x}'^{(e)}}{\tau} \right)}, \quad k_x = \arg \max_k t_k^\top \hat{x}^{(e)}. \quad (36)$$

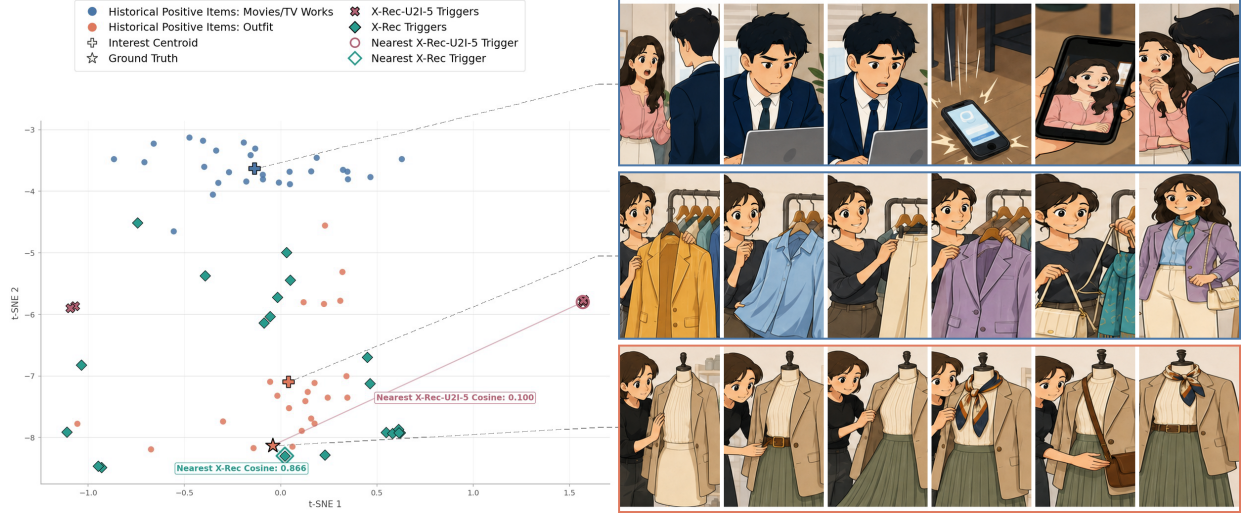
In this way, all interest triggers receive direct optimization signals, instead of only the winning trigger being updated. This substantially improves training stability and leads to much better performance on the streaming benchmark. During inference, the learned interest triggers are used to retrieve candidates from the ANN index. Therefore, X-Rec-U2I- K represents user interest diversity with a finite set of deterministic user vectors, rather than a continuous generative distribution.

A.3. Additional Multi-Interest Case Studies

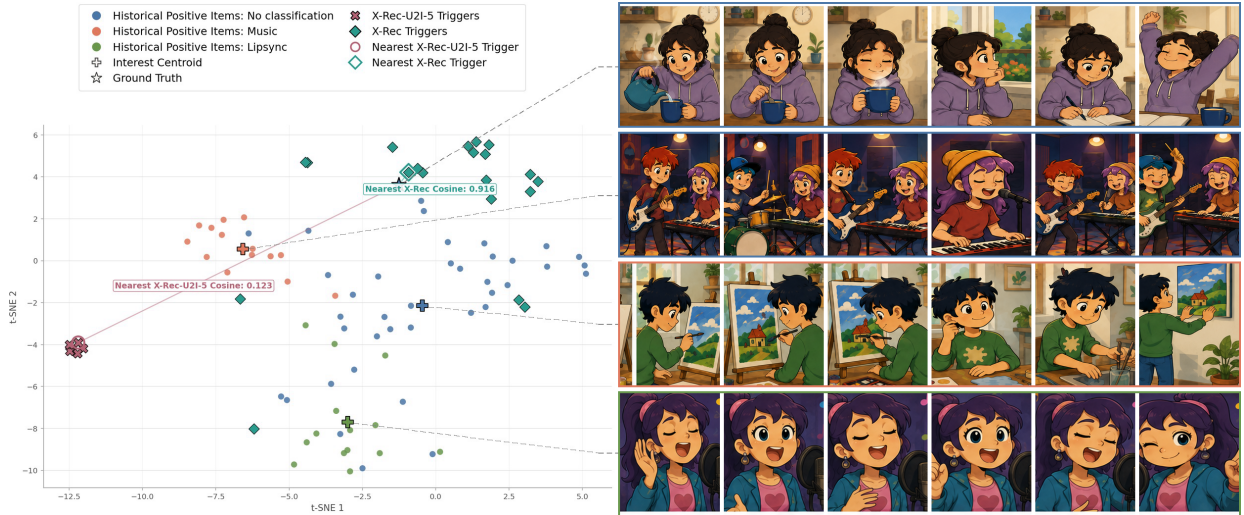
We provide additional qualitative examples of X-Rec-U2I-5 and X-Rec in Figure 11 and Figure 12.

Across Figure 11 and Figure 12, triggers of X-Rec-U2I-5 concentrate in a single local region, whereas X-Rec triggers cover multiple interest clusters. This observation supports the expressiveness claim in §2.3: by directly modeling the target recommendation distribution, X-Rec can capture diverse and multi-modal user interests more effectively than U2I methods.

To examine whether this pattern is caused by the five-trigger budget of X-Rec-U2I-5, we increase the trigger count to 10. As shown in Figure 13 and Figure 14, increasing the trigger count does not change the concentration pattern. Therefore, simply increasing the number of U2I triggers does not eliminate the observed expressiveness gap.

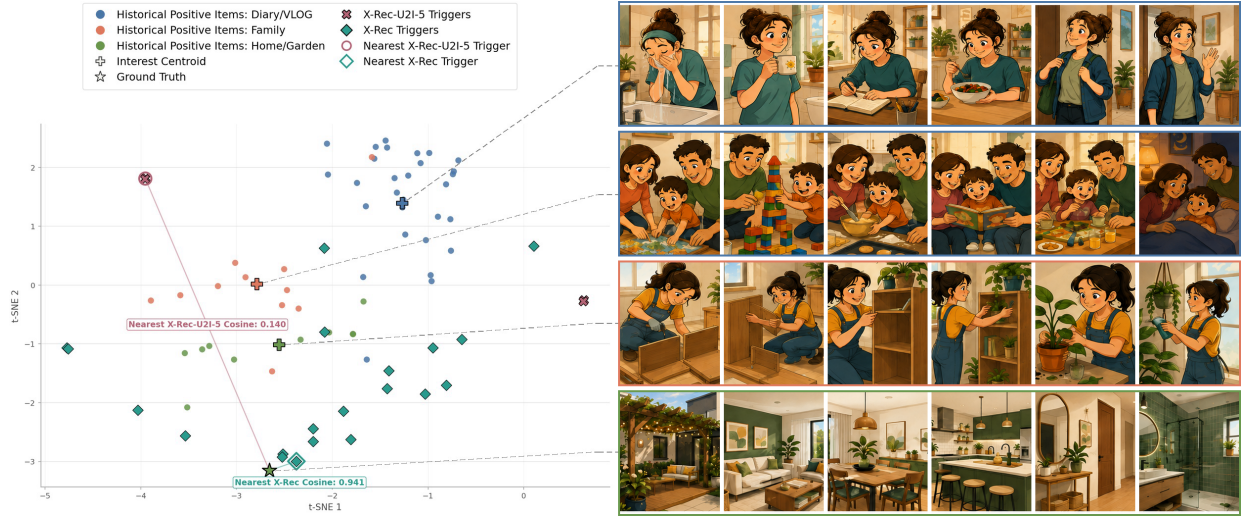


(a) Active interests: Movies/TV Works and Outfit. The ground-truth item belongs to Outfit.

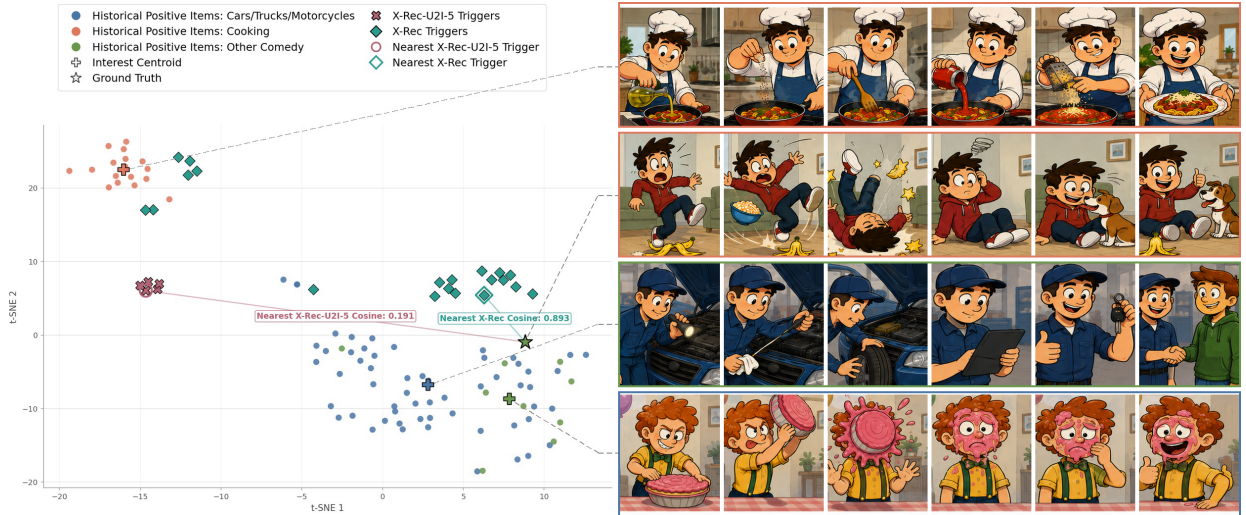


(b) Active interests: Unclassified content, Music, and Lipsync. The ground-truth item belongs to the unclassified mode.

Figure 11 | Visualization of historical positive items and triggers generated by X-Rec and X-Rec-U2I-5.

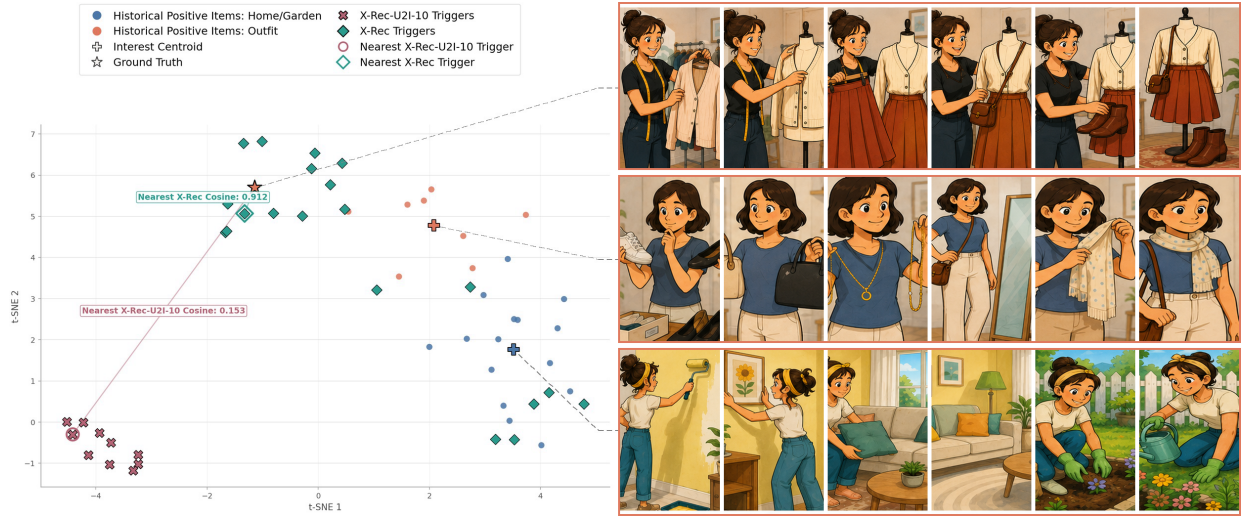


(a) Active interests: Diary/VLOG, Family, and Home/Garden. The ground-truth item belongs to Home/Garden.

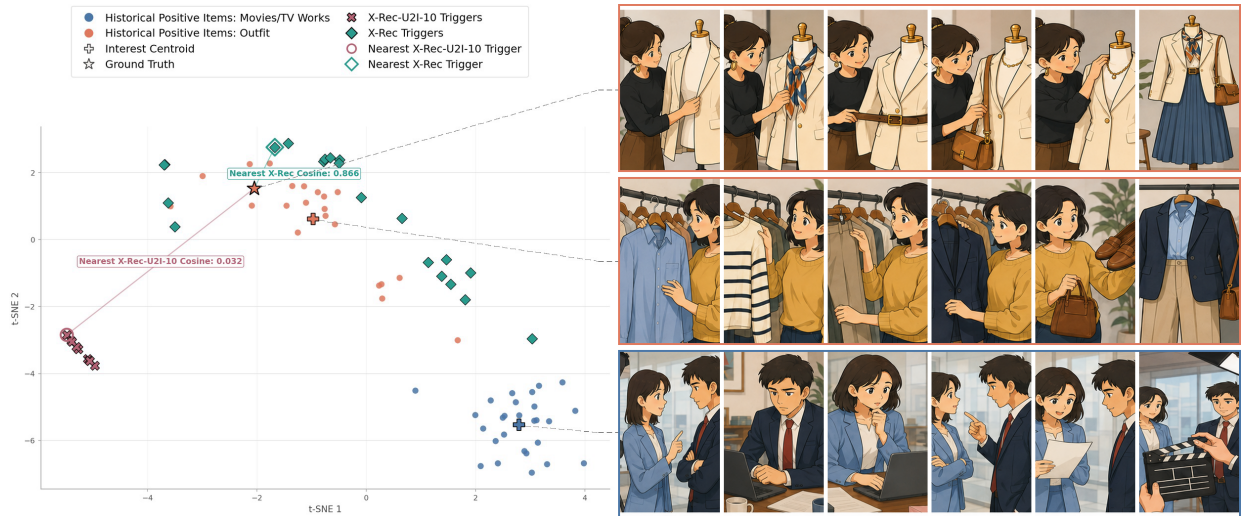


(b) Active interests: Cars/Trucks/Motorcycles, Cooking, and Other Comedy. The ground-truth item belongs to Other Comedy.

Figure 12 | Visualization of historical positive items and triggers generated by X-Rec and X-Rec-U2I-5.

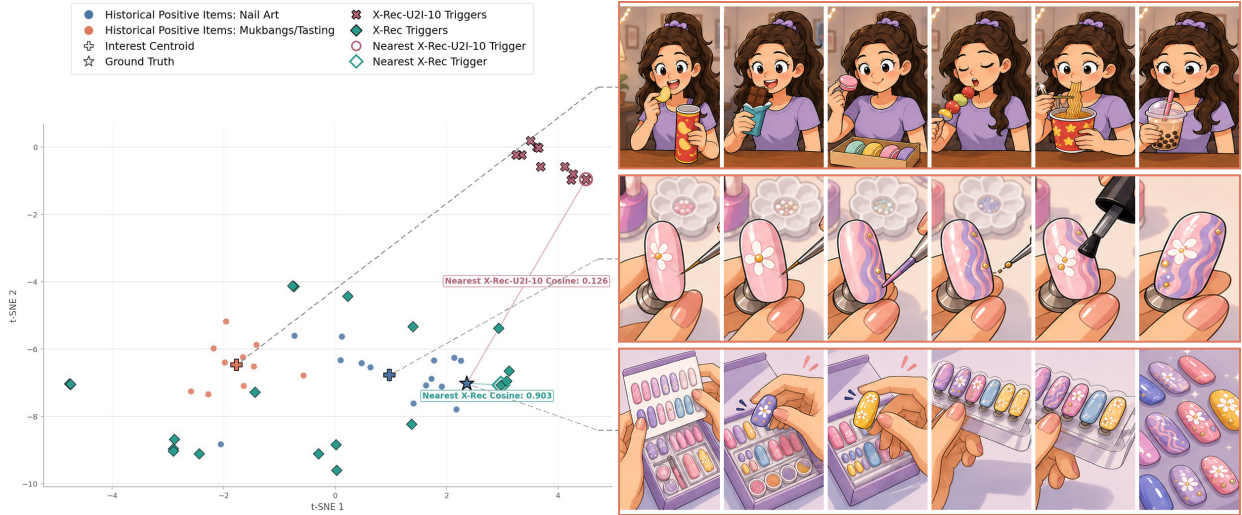


(a) Active interests: Home/Garden and Outfit. The ground-truth item belongs to Outfit.

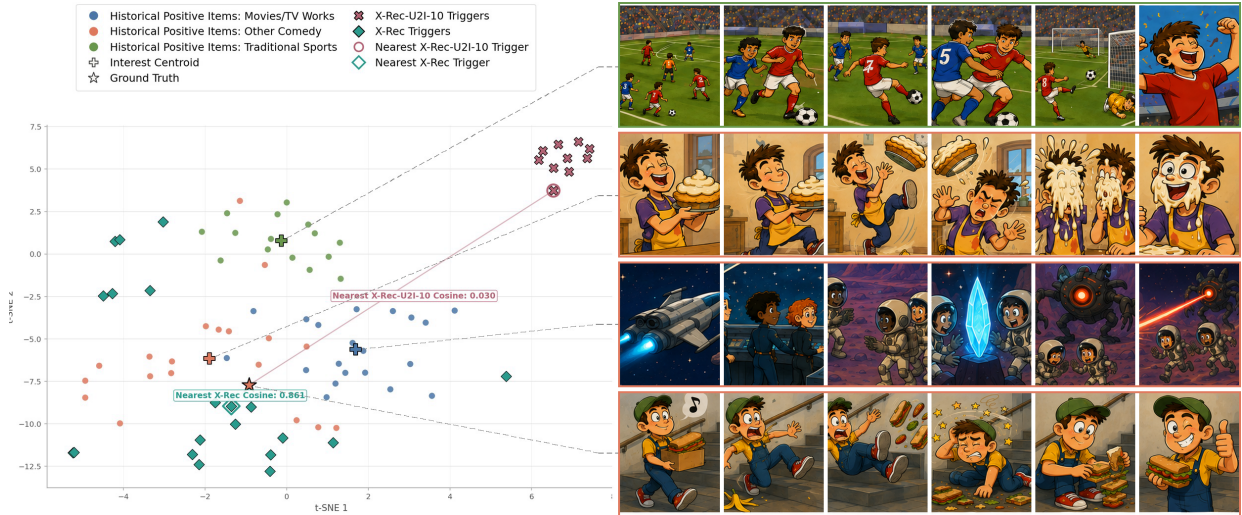


(b) Active interests: Movies/TV Works and Outfit. The ground-truth item belongs to Outfit.

Figure 13 | Visualization of historical positive items and triggers generated by X-Rec and X-Rec-U2I-10.



(a) Active interests: Nail Art and Mukbangs/Tasting. The ground-truth item belongs to Nail Art.



(b) Active interests: Movies/TV Works, Other Comedy, and Traditional Sports. The ground-truth item belongs to Other Comedy.

Figure 14 | Visualization of historical positive items and triggers generated by X-Rec and X-Rec-U2I-10.