ORDER-MARGINALIZED GENERATION OF SEMANTIC AND COLLABORATIVE IDS FOR GENERATIVE RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Recommendation (GR) increasingly incorporates collaborative signals into item tokenization to complement content semantics. When semantic and collaborative representations are separately quantized into two identifier chains, however, their cross-chain generation order is inherently unspecified. Their combination therefore forms a partially ordered target, whereas autoregressive generation requires a total order. Existing methods consequently rely on a prescribed cross-identifier serialization, an arbitrary choice that can materially affect recommendation performance. We propose **LatticeRec**, an order-marginalized framework that treats cross-identifier serialization as a latent variable rather than a prescribed generation target. The key idea is to represent partial generations by canonical states determined only by the revealed prefixes of the two identifiers, allowing different interleavings with equivalent revealed information to share the same predictive state. This organizes all valid interleavings into a compact lattice and enables exact marginalization over generation orders via dynamic programming, without explicit path enumeration. We further introduce coverage-aware training to prevent supervision from concentrating on only a few generation paths. Experiments on three benchmarks demonstrate the effectiveness of LatticeRec and its key components.

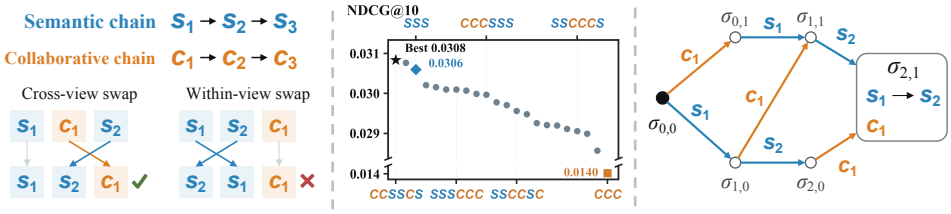
1 INTRODUCTION

Generative recommendation (GR) represents each item as a short discrete token sequence and predicts the next item by autoregressively generating its identifier (Rajput et al., 2023). Content-derived semantic IDs enable parameter sharing and generalization among semantically related items (Singh et al., 2024). However, content similarity does not always align with collaborative patterns in user interactions, motivating recent work to incorporate collaborative signals into item representation and tokenization (Liang et al., 2026; Wang et al., 2024a).

Existing approaches integrate semantic and collaborative information either during identifier construction or through separate modeling branches (Liu et al., 2025a; Wang et al., 2024b). A natural alternative is to quantize semantic and collaborative representations separately into two identifier chains and model their interaction during generation. This seemingly simple design, however, introduces a basic question absent from single-chain GR: *how should the two identifier chains be jointly generated in an autoregressive model?*

Let $S_y = (s_1, \ldots, s_m)$ and $C_y = (c_1, \ldots, c_n)$ denote the semantic and collaborative identifiers of item y. Residual quantization (Lee et al., 2022) specifies the within-view order of each identifier but leaves cross-view precedence unspecified. Figure 1(a) illustrates the resulting constraint: cross-view swaps are permitted as long as the within-view orders are preserved. The resulting dual-view target is therefore partially ordered, whereas autoregressive generation requires a total order. Since training supervision provides the target item y but no particular interleaving of S_y and C_y, any fixed cross-view serialization (Hong et al., 2025; Zhu et al., 2025; Qiao et al., 2026) introduces an additional modeling assumption and changes the autoregressive factorization. Figure 1(b) shows that recommendation performance is sensitive to the chosen fixed serialization. On Instruments with TIGER, most of the 20 evaluated orders score below the semantic-only baseline, while even the best yields only a small

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Partial-Order Constraints **(b) Fixed-Order Sensitivity** **(c) Canonical-State Sharing**

Figure 1: Motivation for order-marginalized generation. Panel (b) shows the results of all valid fixed cross-view serializations on Instruments with TIGER.

improvement. These observations motivate explicitly modeling cross-view order ambiguity rather than committing to a single fixed serialization.

A key observation is that different valid partial trajectories can reveal identical view-specific prefixes. In Figure 1(c), the three illustrated trajectories all reveal the semantic prefix (s_1, s_2) and the collaborative prefix (c_1), despite different cross-view arrival orders. We map trajectories with identical revealed prefixes to a shared canonical partial state that retains these prefixes while abstracting away their cross-view arrival order.

Building on this observation, we propose **LatticeRec**, an order-marginalized generative recommender that treats cross-view interleaving as a latent variable and marginalizes over all valid orders. Canonical-state sharing assigns the same next-step distribution to partial trajectories with identical revealed prefixes, allowing their probability masses to be aggregated. Under this shared-state parameterization, each target item's canonical-state lattice enables exact marginalization over all valid interleavings without explicitly enumerating individual paths. We further introduce coverage regularization to maintain supervision across valid interleavings. Experiments on multiple benchmarks demonstrate the effectiveness of LatticeRec and validate the contributions of our designs.

2 Problem Formulation

2.1 Preliminaries

Residual Quantization. Residual quantization represents a continuous item embedding $\mathbf{h}_y \in \mathbb{R}^{d_h}$ by an ordered sequence of discrete codes $Z(y) = (z_1, \ldots, z_D)$. Starting from $\mathbf{r}_0 = \mathbf{h}_y$, the d-th level selects a codeword from a level-specific codebook $\mathcal{C}_d$ and updates the residual as

$$z_d = \arg\min_k \|\mathbf{r}_{d-1} - \mathbf{e}_{d,k}\|_2^2, \qquad \mathbf{r}_d = \mathbf{r}_{d-1} - \mathbf{e}_{d,z_d}. \tag{1}$$

Because each level quantizes the residual left by preceding levels, the codes inherit an intrinsic within-identifier order $z_1 \prec \cdots \prec z_D$.

Generative Recommendation. Let $\mathcal{V}$ denote the item catalog, $x = (v_1, \ldots, v_t)$ a user interaction history, and $y \in \mathcal{V}$ the target item. Given an ordered identifier $Z(y) = (z_1, \ldots, z_D)$, a generative recommender factorizes its likelihood autoregressively:

$$P_\theta(Z(y) \mid x) = \prod_{d=1}^{D} p_\theta(z_d \mid x, z_{<d}). \tag{2}$$

When the identifier uniquely determines the item, we write this likelihood as $P_\theta(y \mid x)$ for brevity. Standard autoregressive GR therefore assumes a prescribed generation order over the identifier tokens.

2.2 Dual-View Identifiers and Serialization Ambiguity

Dual-View Partial Order. Let $S_y = (s_1, \ldots, s_m)$ and $C_y = (c_1, \ldots, c_n)$ denote the semantic and collaborative identifiers of item y. The semantic and collaborative representations are quantized separately, producing two identifiers with their respective residual-quantization orders:

$$s_1 \prec \cdots \prec s_m, \qquad c_1 \prec \cdots \prec c_n. \tag{3}$$

Their separate construction specifies no precedence between a semantic code s_i and a collaborative code c_j. The dual-view target therefore defines two internally ordered chains rather than a unique total generation order.

Valid Cross-View Interleavings. Autoregressive generation nevertheless requires a total order. A valid cross-view interleaving corresponds to a linear extension of the partial order induced by the two identifier chains. We represent such an interleaving by $\pi = (\pi_1, \ldots, \pi_{m+n}) \in \{S, C\}^{m+n}$, where π_t specifies which chain is extended at step t. Selecting a view emits its next ungenerated code, thereby preserving the within-view order of both chains. The set of valid interleavings is

$$\Pi_{m,n} = \left\{\pi \in \{S, C\}^{m+n} : |\{t : \pi_t = S\}| = m\right\}, \qquad |\Pi_{m,n}| = \binom{m+n}{m}. \tag{4}$$

Fixed Serialization. Let u_t^π denote the target code emitted at step t under a prescribed interleaving π. Under this fixed serialization, the target-item likelihood factorizes as

$$P_\theta^\pi(y \mid x) = \prod_{t=1}^{m+n} p_\theta\left(u_t^\pi \mid x, u_{<t}^\pi\right). \tag{5}$$

Different choices of π condition each code prediction on different revealed prefixes of the two views, inducing different autoregressive factorizations of the same dual-view target.

Crucially, recommendation supervision provides (x, y) rather than (x, y, π): the target item determines S_y and C_y, but does not specify any particular valid interleaving. A fixed serialization therefore introduces an additional modeling choice not determined by the observed label. The sensitivity in Figure 1(b) further shows that this otherwise unspecified choice can materially affect recommendation performance.

2.3 Latent Interleavings and Item-Level Likelihood

We therefore treat the cross-view interleaving $\pi \in \Pi_{m,n}$ as a latent variable. For an interleaving π, let u_t^π denote the target code emitted at step t, and define the partial generation trajectory before step t as

$$\rho_{t-1}^\pi = \left((\pi_1, u_1^\pi), \ldots, (\pi_{t-1}, u_{t-1}^\pi)\right). \tag{6}$$

When the serialization is fixed, the view choice at each step is predetermined. Once the interleaving is latent, however, each transition must account for both which view is extended and which code is emitted. The joint probability of target item y and interleaving π is therefore

$$P_\theta(y, \pi \mid x) = \prod_{t=1}^{m+n} p_\theta\left(\pi_t, u_t^\pi \mid x, \rho_{t-1}^\pi\right). \tag{7}$$

Since the cross-view order is unobserved, we define the item-level probability by marginalizing over all valid interleavings:

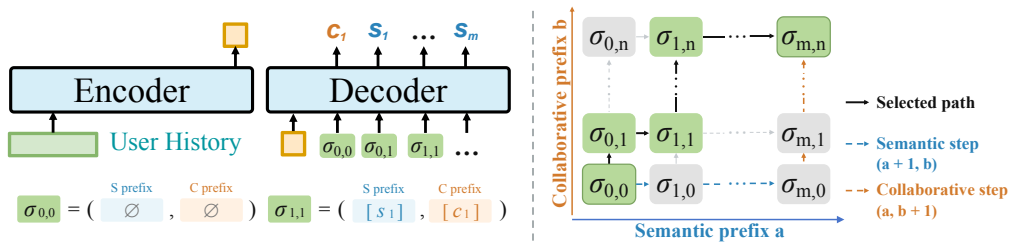
$$P_\theta(y \mid x) = \sum_{\pi \in \Pi_{m,n}} P_\theta(y, \pi \mid x), \qquad \mathcal{L}_{\text{marg}}(x, y) = -\log P_\theta(y \mid x). \tag{8}$$

This formulation avoids designating any particular cross-view serialization as the observed target.

Importantly, different partial trajectories can reveal the same semantic and collaborative prefixes through different cross-view generation orders. This motivates a canonical state defined by the revealed view-specific prefixes rather than the order in which they were generated.

3 LatticeRec: Order-Marginalized Generative Recommendation

Building on the latent-interleaving formulation, LatticeRec represents partial generation trajectories using canonical states defined by the revealed semantic and collaborative prefixes. The key principle is that trajectories exposing the same view-specific prefixes should share a predictive state, rather than remain distinguished by their cross-view arrival order. Under this path-invariant transition model, partial trajectories reaching the same canonical state share the same next-step distribution, allowing their probability masses to be aggregated exactly. Finally, we present coverage-aware marginal training and state-merged decoding.



(a) Canonical-State Generation **(b) Order-Marginalization Lattice**

Figure 2: Overview of LatticeRec. (a) The user history is encoded once, and generation proceeds through canonical states determined by the revealed semantic and collaborative prefixes. At each state, the decoder predicts the next view and corresponding code. (b) For a target item, target-consistent transitions form an order-marginalization lattice. Interleavings reaching the same canonical state share the same continuation model, allowing their probability masses to be merged exactly.

3.1 Canonical Partial States and Path-Invariant Encoding

Canonical Partial States. A key observation is that different generation paths need not correspond to different predictive states. Consider two partial trajectories that reveal the same semantic prefix $S_{1:a}$ and collaborative prefix $C_{1:b}$ through different cross-view arrival orders. Although these partial trajectories differ in their cross-view generation order, they reveal the same view-specific prefixes. Motivated by the absence of specified cross-view precedence, we map such trajectories to a shared canonical partial state:

$$\sigma_{a,b} = (S_{1:a}, C_{1:b}), \qquad 0 \le a \le m,\ 0 \le b \le n. \tag{9}$$

For example, (s_1, c_1, s_2) and (c_1, s_1, s_2) both map to $\sigma_{2,1}$. Canonicalization preserves the revealed semantic and collaborative prefixes while abstracting away only their cross-view arrival order.

From an order-theoretic perspective, each canonical state $\sigma_{a,b}$ corresponds to an order ideal of the partial order induced by the two identifier chains, while each valid interleaving corresponds to a maximal chain in the resulting ideal lattice. A formal characterization of this correspondence is provided in Appendix B.

Path-Invariance Condition. For a canonical state to determine the next-step distribution, that distribution must depend on the revealed view-specific prefixes rather than their cross-view generation order. Let $\sigma(\rho)$ denote the canonical state associated with partial trajectory ρ. We impose the following path-invariance condition:

$$p_\theta(\mu, z \mid x, \rho) = p_\theta(\mu, z \mid x, \sigma(\rho)), \qquad \mu \in \{S, C\}. \tag{10}$$

This condition makes the canonical state sufficient for next-step prediction. Here, path invariance concerns the next-step distribution at a fixed canonical state; it does not require different complete interleavings to have equal probabilities.

Path-Invariant State Encoding. Given a user history x, we first encode it once as

$$H_x = \mathrm{Encoder}_\theta(x). \tag{11}$$

We enforce path invariance by constructing the decoder input directly from the canonical state. For $\sigma_{a,b}$, revealed codes are placed into fixed view- and level-specific slots:

$$\mathrm{Canon}(\sigma_{a,b}) = [s_1, \ldots, s_a, \underbrace{\texttt{[MASK]}, \ldots, \texttt{[MASK]}}_{m-a} \mid c_1, \ldots, c_b, \underbrace{\texttt{[MASK]}, \ldots, \texttt{[MASK]}}_{n-b}]. \tag{12}$$

Here, [MASK] denotes an unrevealed slot whose code-content embedding is zero; its view and level embeddings are retained. The resulting slot embeddings are transformed into a state query $\mathbf{q}_{a,b}$ for the canonical state. The decoder then combines this state query with the encoded user history:

$$\mathbf{h}_{a,b} = \mathrm{Decoder}_\theta\left(\mathbf{q}_{a,b}, H_x\right). \tag{13}$$

Because $\mathrm{Canon}(\sigma_{a,b})$ depends only on the canonical state, all trajectories mapped to $\sigma_{a,b}$ induce the same state query $\mathbf{q}_{a,b}$ and hence the same continuation distribution. Eq. (10) is therefore satisfied by construction.

3.2 Exact Order Marginalization

Joint View-and-Code Prediction. Once the interleaving is latent, the view choice itself becomes part of the generation process. From canonical state $\sigma_{a,b}$, LatticeRec jointly predicts which identifier chain to extend and which code to generate. We factorize the transition probability as

$$p_\theta(\mu, z \mid x, \sigma_{a,b}) = g_\theta(\mu \mid \mathbf{h}_{a,b}) q_\theta(z \mid \mu, \mathbf{h}_{a,b}). \tag{14}$$

Here, g_θ predicts the next view, and q_θ defines a distribution over code indices in the codebook for the selected view and its next ungenerated level. Unavailable views are masked before normalizing g_θ: semantic transitions are allowed only when $a < m$, and collaborative transitions only when $b < n$. The initial transition is handled identically from $\sigma_{0,0}$, requiring no serialization-specific start rule.

Lossless Path Merging. Under Eq. (10), trajectories reaching the same canonical state share the same continuation distribution, so their probability masses can be aggregated without loss. We define the aggregated probability mass reaching $\sigma_{a,b}$ as

$$\alpha_{a,b} = \sum_{\rho:\, \sigma(\rho) = \sigma_{a,b}} P_\theta(\rho \mid x), \qquad \alpha_{0,0} = 1. \tag{15}$$

Theorem 1 (Lossless Path Merging via Canonical States)**.** *Suppose the transition model satisfies Eq.* (10)*. Then, for every* $(a, b) \neq (0, 0)$*,*

$$\alpha_{a,b} = \mathbb{I}[a > 0] \alpha_{a-1,b} p_\theta(S, s_a \mid x, \sigma_{a-1,b}) + \mathbb{I}[b > 0] \alpha_{a,b-1} p_\theta(C, c_b \mid x, \sigma_{a,b-1}). \tag{16}$$

At the terminal state,

$$\alpha_{m,n} = \sum_{\pi \in \Pi_{m,n}} P_\theta(y, \pi \mid x) = P_\theta(y \mid x). \tag{17}$$

Hence, merging trajectories by canonical state preserves the exact item-level marginal likelihood.

The proof is provided in Appendix C.1. Under Eq. (10), the theorem guarantees that canonical-state aggregation exactly recovers the item-level marginal likelihood over all valid interleavings. Thus, the $\binom{m+n}{m}$ interleavings can be compressed into $(m+1)(n+1)$ canonical states, enabling exact dynamic programming in $O(mn)$ arithmetic operations once the neural transition probabilities are available. We implement the recursion in log space using log-sum-exp; details are given in Appendix C.2.

3.3 Marginal Training with Coverage Regularization

Posterior-Weighted Gradients. We further observe that, although marginalization removes the need for fixed-order supervision, it does not guarantee non-negligible supervision for every valid interleaving during training. Specifically, the gradient contribution of each path is weighted by its posterior probability under the current model. For a valid path π, define

$$r_\theta(\pi \mid x, y) = \frac{P_\theta(y, \pi \mid x)}{P_\theta(y \mid x)}. \tag{18}$$

The marginal-likelihood gradient can then be written as

$$\nabla_\theta \mathcal{L}_{\text{marg}} = - \sum_{\pi \in \Pi_{m,n}} r_\theta(\pi \mid x, y) \nabla_\theta \log P_\theta(y, \pi \mid x). \tag{19}$$

Thus, a path with higher posterior probability receives a larger coefficient in the gradient decomposition, whereas the coefficient of a low-posterior path can become arbitrarily small. Because the path posterior has no model-independent positive lower bound, some valid interleavings can receive negligible gradient weights during training.

Coverage Regularization. We introduce coverage regularization to lower-bound the gradient weight of every valid interleaving when $\lambda > 0$, without imposing exact equality of path posterior probabilities. We therefore introduce a uniform reference distribution $U_y(\pi) = 1/K$ over the $K = |\Pi_{m,n}|$ valid interleavings and optimize

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{marg}} + \lambda D_{\text{KL}} \left(U_y \| r_\theta(\cdot \mid x, y) \right), \qquad 0 \leq \lambda \leq 1. \tag{20}$$

Proposition 2 (Coverage as Jensen-Gap Control)**.** *For the uniform-path loss* $\mathcal{L}_{\text{uni}} = -\frac{1}{K}\sum_{\pi\in\Pi_{m,n}} \log P_\theta(y, \pi \mid x)$*, the coverage regularizer satisfies*

$$D_{\text{KL}}(U_y \| r_\theta) = \mathcal{L}_{\text{uni}} - (\mathcal{L}_{\text{marg}} + \log K)\,. \tag{21}$$

Consequently,

$$\mathcal{L}_{\text{train}} = (1-\lambda)\mathcal{L}_{\text{marg}} + \lambda\mathcal{L}_{\text{uni}} - \lambda \log K. \tag{22}$$

Proposition 2 identifies the coverage regularizer with the Jensen gap between $\mathcal{L}_{\text{uni}}$ and $\mathcal{L}_{\text{marg}} + \log K$. Up to an additive constant independent of θ, λ interpolates between marginal-likelihood training and uniform-path training.

More explicitly, letting $\gamma_\pi = \log P_\theta(y, \pi \mid x)$ and $r_\pi = r_\theta(\pi \mid x, y)$, we have

$$\frac{\partial \mathcal{L}_{\text{train}}}{\partial \gamma_\pi} = -w_\pi, \qquad w_\pi = (1-\lambda) r_\pi + \frac{\lambda}{K} \geq \frac{\lambda}{K}. \tag{23}$$

For $\lambda > 0$, every valid interleaving receives a gradient weight of at least λ/K. For $0 < \lambda < 1$, the weights also retain a posterior-dependent component; at $\lambda = 1$, they become uniform. The proof of Proposition 2, the gradient-weight analysis, and further discussion of the $\lambda = 1$ endpoint are provided in Appendix D. The lattice-based computation of $\mathcal{L}_{\text{uni}}$ is derived in Appendix E.

3.4 State-Merged Decoding

At inference time, we perform beam search over canonical states and aggregate the probability masses of surviving partial trajectories that reach the same state. Aggregation within a retained canonical state is exact, while beam pruning makes the overall inference procedure approximate. Let $\mathcal{E}_t$ denote the partial trajectories represented by the expanded beam at step t, and let LSE denote the log-sum-exp operator. Their scores are aggregated as

$$s_t(\sigma) = \text{LSE}_{\rho\in\mathcal{E}_t:\,\sigma(\rho)=\sigma} \log P_\theta(\rho \mid x). \tag{24}$$

We then retain the top-B canonical states, allocating the beam budget to distinct states rather than multiple trajectories with the same revealed prefixes. For latency-sensitive inference, we optionally early-terminate a state once its semantic and collaborative prefixes jointly identify a unique catalog item, assigning its accumulated log-mass to that item. Scores from multiple terminated states identifying the same item are combined by log-sum-exp.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on three datasets: Instruments and Games from Amazon Reviews 2023 (Hou et al., 2024), and Steam (Kang & McAuley, 2018). Each user's review history is grouped into a single sequence and sorted chronologically. We adopt the widely used leave-one-out strategy (Kang & McAuley, 2018) for data splitting, using the most recent interaction for testing, the second most recent for validation, and the remainder for training. Dataset statistics and preprocessing details are provided in Appendix G.1.

Implementation. We separately quantize the semantic and collaborative item representations using RQ-Kmeans (Ju et al., 2025). We adopt a T5-based encoder–decoder (Rajput et al., 2023) as our recommendation backbone. We report Recall@5/10 and NDCG@5/10, with detailed implementation and evaluation settings provided in Appendix G.

4.2 Main Results

Table 1 compares LatticeRec with sequential and generative recommendation baselines. LatticeRec achieves the best performance across all three datasets and all reported metrics, with an average relative improvement of 5.18% on NDCG@10 over the strongest baseline on each dataset. Notably, LatticeRec also outperforms generative methods such as ActionPiece (Hou et al., 2025b) and LETTER (Wang et al., 2024a), which enrich item tokenization with semantic features and interaction-derived signals. These comparisons suggest that explicitly modeling cross-view generation order can provide benefits beyond incorporating semantic and collaborative information, a hypothesis further examined through the controlled ablations below.

Table 1: Overall recommendation performance. The best and second-best results are highlighted in **bold** and underlined, respectively. "Δ" denotes the relative improvement of LatticeRec over the strongest baseline, and "*" denotes statistically significant improvements ($p < 0.05$) over the best baseline according to a paired t-test.

Methods	Instruments				Games				Steam			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
GRU4Rec	0.0303	0.0478	0.0195	0.0251	0.0491	0.0782	0.0316	0.0409	0.0576	0.0903	0.0377	0.0482
SASRec	0.0335	0.0531	0.0207	0.0270	0.0558	0.0892	0.0328	0.0435	0.0533	0.0862	0.0338	0.0444
BERT4Rec	0.0247	0.0400	0.0158	0.0207	0.0424	0.0687	0.0275	0.0359	0.0512	0.0822	0.0331	0.0430
FDSA	0.0387	0.0604	0.0250	0.0321	0.0587	0.0916	0.0376	0.0481	0.0559	0.0870	0.0360	0.0459
HSTU	0.0275	0.0444	0.0165	0.0219	0.0515	0.0856	0.0311	0.0422	0.0461	0.0762	0.0281	0.0378
TIGER	0.0368	0.0580	0.0238	0.0306	0.0571	0.0902	0.0374	0.0481	0.0571	0.0916	0.0379	0.0489
LLaDA-Rec	0.0392	0.0596	0.0258	0.0323	0.0605	0.0938	0.0401	0.0508	0.0560	0.0878	0.0370	0.0472
LETTER	0.0310	0.0481	0.0200	0.0255	0.0476	0.0740	0.0312	0.0398	0.0525	0.0833	0.0349	0.0448
EAGER	0.0331	0.0518	0.0212	0.0272	0.0531	0.0839	0.0343	0.0442	0.0588	0.0923	0.0391	0.0498
ActionPiece	0.0394	0.0611	0.0256	0.0325	0.0602	0.0941	0.0392	0.0501	0.0569	0.0889	0.0380	0.0483
SETRec++	0.0389	0.0616	0.0249	0.0322	0.0577	0.0918	0.0369	0.0479	0.0583	0.0940	0.0379	0.0494
MTGRec	0.0395	0.0618	0.0263	0.0335	0.0611	0.0943	0.0404	0.0510	0.0557	0.0857	0.0374	0.0471
SCRec	0.0362	0.0546	0.0237	0.0296	0.0483	0.0757	0.0315	0.0403	0.0515	0.0816	0.0331	0.0427
LatticeRec	**0.0406***	**0.0619**	**0.0271***	**0.0339***	**0.0641***	**0.0964***	**0.0432***	**0.0536***	**0.0644***	**0.0979***	**0.0436***	**0.0544***
Δ	+2.78%	+0.16%	+3.04%	+1.19%	+4.91%	+2.23%	+6.93%	+5.10%	+9.52%	+4.15%	+11.51%	+9.24%

Table 2: Ablation study of the core designs in LatticeRec.

Variants	Instruments				Games				Steam			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
w/o Marginal Inference	0.0388	0.0593	0.0259	0.0325	0.0610	0.0930	0.0408	0.0511	0.0617	0.0926	0.0419	0.0519
w/o Canonical-State Sharing	0.0396	0.0586	0.0261	0.0322	0.0603	0.0912	0.0401	0.0500	0.0605	0.0948	0.0416	0.0525
w/o Coverage Regularization	0.0354	0.0556	0.0234	0.0299	0.0608	0.0921	0.0406	0.0506	0.0559	0.0873	0.0365	0.0466
LatticeRec	**0.0406**	**0.0619**	**0.0271**	**0.0339**	**0.0641**	**0.0964**	**0.0432**	**0.0536**	**0.0636**	**0.0961**	**0.0430**	**0.0535**

4.3 Ablation Study

We ablate the three core components of LatticeRec. Table 2 summarizes the results.

- **Marginal Inference.** We replace marginal aggregation with max aggregation during inference while keeping the trained checkpoint unchanged, confirming the benefit of aggregating probability mass across alternative generation orders.
- **Canonical-state sharing.** Allowing the next-step distribution to depend on cross-view generation order, even when the revealed prefixes are identical, reduces recommendation performance.
- **Coverage regularization.** Removing coverage regularization substantially reduces performance across all three datasets, highlighting the importance of maintaining gradient weights across valid interleavings.

4.4 Cross-View Order Analysis

We further examine how the choice of cross-view serialization affects recommendation performance on Games, where three semantic and three collaborative codes yield $K = \binom{6}{3} = 20$ valid interleavings.

Fixed-order sensitivity. We train a separate model for each valid interleaving while keeping the identifiers, history inputs, model architecture, and training configuration unchanged. The prescribed interleaving fixes the selected view at each step, so the model predicts only the corresponding code. As shown in Figure 3, NDCG@10 varies noticeably across the 20 fixed orders. This sensitivity confirms that imposing a particular cross-view serialization is a consequential modeling choice. All 20 models use the

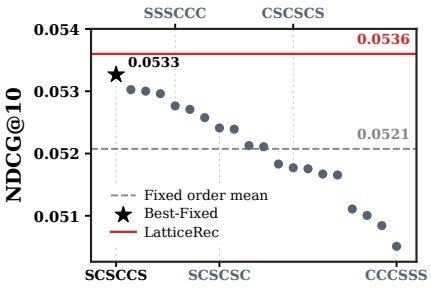


Figure 3: 20 valid fixed cross-view serializations (sorted by NDCG@10) on Games.

same semantic and collaborative identifiers and preserve both within-view orders; they differ only in cross-view serialization.

Comparison with Best-Fixed. To construct Best-Fixed, we train one model for each of the 20 valid interleavings and select the model with the highest validation NDCG@10. Best-Fixed therefore provides a strong fixed-order reference, but requires explicitly searching over candidate serializations. In contrast, LatticeRec marginalizes over all valid interleavings within a single model and achieves higher test NDCG@10 without committing to any particular cross-view order. Together, these results support treating cross-view order as latent rather than imposing a single externally specified serialization.

4.5 In-Depth Analysis

Effect of coverage regularization. We vary $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ to examine how coverage regularization affects posterior concentration and recommendation performance. We report posterior entropy $H(r)$ and the effective number of paths $\exp(H(r))$. Formal definitions are given in Appendix I.

As shown in Figure 4, marginal-likelihood training alone ($\lambda = 0$) produces a concentrated path posterior, reflected in low posterior entropy and a small effective number of paths. Increasing λ from 0 to 0.2 accounts for most of the observed increase in posterior entropy and the effective number of paths, while also improving NDCG@10 on all three datasets. Further increasing λ continues to broaden path coverage, whereas recommendation performance changes only mildly and non-monotonically. These results suggest that coverage regularization discourages excessive posterior concentration, while recommendation performance is relatively insensitive to the precise choice of a positive λ.

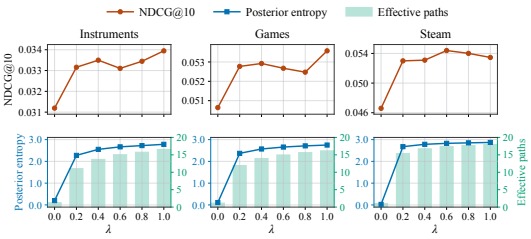


Figure 4: Effect of λ.

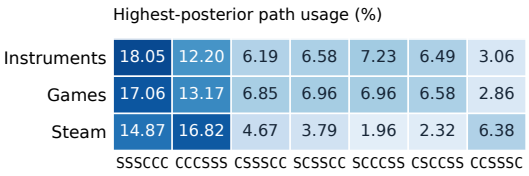


Figure 5: Highest-posterior path usage on the three datasets.

Highest-posterior interleavings. For each test example (x, y), we identify the interleaving with the highest posterior probability under the learned model. Figure 5 reports how frequently each displayed interleaving attains the highest posterior probability. No interleaving is the posterior maximizer for more than 18.05% of test examples on any dataset. This shows that the highest-posterior interleaving varies across examples under the learned model, rather than being represented by one globally dominant serialization. Full statistics over all 20 interleavings are provided in Appendix H.

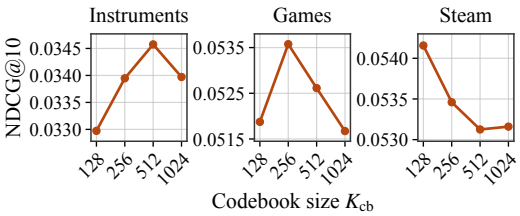


Figure 6: Effect of codebook size K_{cb} on recommendation performance.

Table 3: Effect of identifier depth L.

Dataset	L	R@5	N@5	R@10	N@10
Instruments	2	0.0399	0.0264	0.0596	0.0327
	3	0.0406	0.0271	**0.0619**	0.0339
	4	**0.0408**	**0.0275**	0.0618	**0.0342**
Games	2	0.0633	0.0424	0.0952	0.0527
	3	**0.0641**	**0.0432**	**0.0964**	**0.0536**
	4	0.0614	0.0410	0.0930	0.0512
Steam	2	**0.0640**	0.0426	**0.0985**	0.0536
	3	0.0636	0.0430	0.0961	0.0535
	4	0.0639	**0.0435**	0.0973	**0.0542**

Effect of codebook size and depth. We vary the codebook size K_{cb} and the number of quantization levels per view L, with $m = n = L$. As shown in Figure 6 and Table 3, increasing either K_{cb} or L does not monotonically improve recommendation performance, and the preferred configuration

varies across datasets. Increasing the codebook size or identifier depth does not consistently improve recommendation performance, and the best-performing configuration differs across datasets.

Effect of marginal inference. We further examine the value of marginalizing valid generation orders under different identifier depths and codebook sizes on Steam. For each configuration, Marginal and Max-Path use the same trained checkpoint and differ only in the score-aggregation rule during decoding: log-sum-exp versus maximum. As shown in Table 4, Marginal consistently improves Recall@10 and NDCG@10 across all configurations, indicating that useful item probability mass is distributed over multiple valid paths. These results support summing, rather than maximizing, probability mass over alternative interleavings during decoding.

Table 4: Comparison of marginal and max-path inference across different identifier configurations on Steam.

Metric	Inference	L3-K128	L3-K256	L3-K512	L4-K256
R@10	Marginal	**0.0982**	**0.0961**	**0.0950**	**0.0973**
	Max-Path	0.0975	0.0947	0.0940	0.0955
N@10	Marginal	**0.0542**	**0.0535**	**0.0531**	**0.0542**
	Max-Path	0.0538	0.0527	0.0524	0.0528

Table 5: Collision rates (%) of semantic, collaborative, and joint identifiers before disambiguation (3-level RQ-Kmeans).

Dataset	Semantic	Collaborative	Joint
Instruments	16.777	0.053	**0.000**
Games	18.566	0.105	**0.004**
Steam	2.875	0.028	**0.000**

Identifier collisions. As shown in Table 5, collaborative IDs have substantially lower collision rates than semantic IDs, while using the identifier pair (S_y, C_y) as the joint identifier reduces the collision rate to nearly zero. In this experiment, collaborative IDs distinguish many items that share the same semantic ID. This indicates that collaborative quantization provides strong item-level discrimination that is unavailable when items are represented by semantic IDs alone. The two views are therefore complementary not only in information content, but also in their ability to uniquely identify items. For items sharing the same identifier pair (S_y, C_y), we append a disambiguation token after both identifier chains have been generated.

Generation efficiency. Dual-view identifiers provide richer item representations but introduce additional decoding steps. For comparison, TIGER generates three semantic tokens followed by a disambiguation token to uniquely identify the target item. We reduce the additional decoding overhead using the early-termination rule described in Section 3.4. As shown in Table 6, early termination consistently reduces the effective generation length across all three datasets, while Figure 7 further shows a substantial reduction in measured inference latency. Although dual-view identifiers contain more tokens, early termination avoids generating tokens that are no longer needed to identify an item.

Table 6: Average number of decoding steps.

Method	Instruments	Games	Steam
TIGER	4.00	4.00	4.00
Ours w/o accel.	6.00	6.00	6.00
Ours	**5.00**	**5.04**	**4.99**

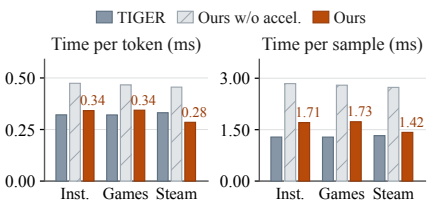


Figure 7: Mean inference time per token and sample. Inst. denotes Instruments.

5 Conclusion

In this work, we study cross-view serialization ambiguity in generative recommendation with separately quantized semantic and collaborative identifiers. Rather than imposing a fixed serialization, LatticeRec treats the cross-view interleaving as a latent variable and marginalizes over all valid interleavings. Our key observation is that partial trajectories with the same revealed semantic and collaborative prefixes can be represented by a shared canonical state. This representation enables exact dynamic-programming marginalization for a target item and probability aggregation over surviving trajectories during beam decoding. Coverage regularization further maintains a positive gradient weight for every valid interleaving when $\lambda > 0$. Experiments on multiple recommendation benchmarks consistently validate the effectiveness of the proposed framework. In future work, we plan to extend this framework to richer multi-view identifier structures.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing and improving the clarity and readability of the manuscript. These tools were not used to generate research ideas, experimental results, or scientific conclusions. All AI-assisted edits were carefully reviewed and revised by the authors. We take full responsibility for the final content of this work.

REFERENCES

Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. RecSys challenge 2018: Automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 527–528. Association for Computing Machinery, 2018. doi: 10.1145/3240323.3240342.

Jiayi Dan, Weijian Li, Yongqi Liu, and Kaiqiao Zhan. Addressing cross-stage decoupling of semantic and collaborative signals in generative recommendation. In *Proceedings of the 20th ACM Conference on Recommender Systems*, RecSys '26. Association for Computing Machinery, 2026. doi: 10.1145/3773078.3831736.

Junchen Fu, Xuri Ge, Alexandros Karatzoglou, Ioannis Arapakis, Suzan Verberne, Joemon M. Jose, and Zhaochun Ren. Differentiable semantic ID for generative recommendation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 369–379. ACM, 2026. doi: 10.1145/3805712.3809641.

Alex Graves. Sequence transduction with recurrent neural networks. In *ICML 2012 Workshop on Representation Learning*, Edinburgh, Scotland, 2012.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 369–376. ACM, 2006. doi: 10.1145/1143844.1143891.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676, 2019. doi: 10.1162/tacl_a_00292.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

Minjie Hong, Yan Xia, Zehan Wang, Jieming Zhu, Ye Wang, Sihang Cai, Xiaoda Yang, Quanyu Dai, Zhenhua Dong, Zhimeng Zhang, and Zhou Zhao. EAGER-LLM: Enhancing large language models as recommenders through exogenous behavior-semantic integration. In *Proceedings of the ACM on Web Conference 2025*, pp. 2754–2762. ACM, 2025. doi: 10.1145/3696410.3714933.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. Generating long semantic IDs in parallel for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025a. doi: 10.1145/3711896.3736979.

Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H. Chi, Julian McAuley, and Derek Zhiyuan Cheng. ActionPiece: Contextually tokenizing action sequences for generative recommendation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24004–24024. PMLR, 2025b.

Yupeng Hou, Haven Kim, Clark Mingxuan Ju, Eduardo Escoto, Neil Shah, and Julian McAuley. Expressiveness limits of autoregressive semantic ID generation in generative recommendation. *arXiv preprint arXiv:2605.06331*, 2026.

Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. How to index item IDs for recommendation foundation models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pp. 195–204. ACM, 2023. doi: 10.1145/3624918.3625339.

Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvesh Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. Generative recommendation with semantic IDs: a practitioner's handbook. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 6420–6425, 2025.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206. IEEE, 2018. doi: 10.1109/ICDM.2018.00035.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11523–11532. IEEE, 2022.

Yu Liang, Zhongjin Zhang, Yuxuan Zhu, Kerui Zhang, Zhiluohan Guo, Wenhang Zhou, Zonqi Yang, Kangle Wu, Yabo Ni, Anxiang Zeng, Cong Fu, Jianxin Wang, and Jiazhi Xia. Rethinking generative recommender tokenizer: RecSys-native encoding and semantic quantization beyond llms. *arXiv preprint arXiv:2602.02338*, 2026.

Xinyu Lin, Chuanyu Zhang, Yufan Liu, Haihan Shi, Wenjie Wang, Fuli Feng, Qifan Wang, See-Kiong Ng, and Tat-Seng Chua. SETRec++: Scaling order-agnostic identifier for large language model-based generative recommendation. *ACM Transactions on Recommender Systems*, 2026. doi: 10.1145/3838805.

Chang Liu, Yimeng Bai, Xiaoyan Zhao, Yang Zhang, Fuli Feng, and Wenge Rong. DiscRec: Disentangled semantic-collaborative modeling for generative recommendation. *arXiv preprint arXiv:2506.15576*, 2025a.

Enze Liu, Bowen Zheng, Cheng Ling, Lantao Hu, Han Li, and Wayne Xin Zhao. Generative recommender with end-to-end learnable item tokenization. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 729–739. ACM, 2025b. doi: 10.1145/3726302.3729989.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-T5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 1864–1874. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.146.

Shutong Qiao, Wei Yuan, Tong Chen, Xiangyu Zhao, Quoc Viet Hung Nguyen, and Hongzhi Yin. When text-as-vision meets semantic IDs in generative recommendation: An empirical study. *arXiv preprint arXiv:2601.14697*, 2026.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Teng Shi, Chenglei Shen, Weijie Yu, Shen Nie, Chongxuan Li, Xiao Zhang, Ming He, Yan Han, and Jun Xu. LLaDA-Rec: Discrete diffusion for parallel semantic ID generation in generative recommendation. *arXiv preprint arXiv:2511.06254*, 2025.

Anima Singh, Trung Vu, Nikhil Mehta, Raghunandan Keshavan, Maheswaran Sathiamoorthy, Yilin Zheng, Lichan Hong, Lukasz Heldt, Li Wei, Devansh Tandon, et al. Better generalization with semantic IDs: A case study in ranking for recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pp. 1039–1044, 2024.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5976–5985. PMLR, 2019.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1441–1450. ACM, 2019. doi: 10.1145/3357384.3357895.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 2400–2409. ACM, 2024a. doi: 10.1145/3627673.3679569.

Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, and Zhenhua Dong. EAGER: Two-stream generative recommender with behavior-semantic collaboration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3245–3254. ACM, 2024b. doi: 10.1145/3637528.3671775.

Yidan Wang, Zhaochun Ren, Weiwei Sun, Jiyuan Yang, Zhixiang Liang, Xin Chen, Ruobing Xie, Su Yan, Xu Zhang, Pengjie Ren, Zhumin Chen, and Xin Xin. Content-based collaborative generation for recommender systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 2420–2430. ACM, 2024c. doi: 10.1145/3627673.3679692.

Junwei Yin, Senjie Kou, Changhao Li, Shuli Wang, Xue Wei, Yinqiu Huang, Yinhua Zhu, Haitao Wang, and Xingxing Wang. DOS: Dual-flow orthogonal semantic IDs for recommendation in meituan. In *Proceedings of the ACM Web Conference 2026*, pp. 8305–8308. ACM, 2026. doi: 10.1145/3774904.3792845.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58484–58509. PMLR, 2024.

Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. Feature-level deeper self-attention network for sequential recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 4320–4326, 2019. doi: 10.24963/ijcai.2019/600.

Bowen Zheng, Enze Liu, Zhongfu Chen, Zhongrui Ma, Yue Wang, Wayne Xin Zhao, and Ji-Rong Wen. Pre-training generative recommender with multi-identifier item tokenization. *arXiv preprint arXiv:2504.04400*, 2025.

Bowen Zheng, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. Universal item tokenization for transferable generative recommendation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2509–2520. ACM, 2026. doi: 10.1145/3805712.3809720.

Jieming Zhu, Mengqun Jin, Qijiong Liu, Zexuan Qiu, Zhenhua Dong, and Xiu Li. CoST: Contrastive quantization based semantic tokenization for generative recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pp. 969–974. ACM, 2024. doi: 10.1145/3640457.3688178.

Jing Zhu, Mingxuan Ju, Yozen Liu, Danai Koutra, Neil Shah, and Tong Zhao. Beyond unimodal boundaries: Generative recommendation with multimodal semantics. *arXiv preprint arXiv:2503.23333*, 2025.

Table 7: Summary of the main notation used in the theoretical analysis.

Notation	Explanation
x	User interaction history.
y	Target item.
$S_y = (s_1, \ldots, s_m)$	Semantic identifier of item y.
$C_y = (c_1, \ldots, c_n)$	Collaborative identifier of item y.
π	A valid cross-view interleaving.
$\Pi_{m,n}$	Set of valid interleavings preserving both within-view orders.
K	Number of valid interleavings, $K = \lvert\Pi_{m,n}\rvert = \binom{m+n}{m}$.
ρ	A target-consistent partial generation trajectory.
$\sigma_{a,b}$	Canonical partial state revealing $S_{1:a}$ and $C_{1:b}$.
$\alpha_{a,b}$	Aggregated probability mass reaching $\sigma_{a,b}$.
$A_{a,b}$	Log aggregated probability mass, $A_{a,b} = \log \alpha_{a,b}$.
$P_\theta(y, \pi \mid x)$	Joint probability of target item y and interleaving π.
r_π	Posterior probability of interleaving π, $r_\pi = r_\theta(\pi \mid x, y)$.
$U_y(\pi)$	Uniform reference distribution over valid interleavings, $U_y(\pi) = 1/K$.
λ	Strength of the coverage regularization.
γ_π	Path log-score, $\gamma_\pi = \log P_\theta(y, \pi \mid x)$.
w_π	Coefficient of interleaving π in the path-gradient decomposition.
$\ell^S_{a,b}$, $\ell^C_{a,b}$	Log-probabilities of the target-consistent semantic and collaborative transitions from $\sigma_{a,b}$.
$F(\sigma)$	Number of valid path prefixes from $\sigma_{0,0}$ to σ.
$R(\sigma)$	Number of valid path suffixes from σ to $\sigma_{m,n}$.

A Notation

For ease of reference, Table 7 summarizes the main notation used in order marginalization and coverage regularization.

B Order-Theoretic Characterization of Canonical States

For a target item y, consider the partially ordered set (poset) $\mathcal{P}_y$ formed by the two identifier chains

$$s_1 \prec \cdots \prec s_m, \qquad c_1 \prec \cdots \prec c_n, \tag{25}$$

with no precedence relation between elements from different views. Here, each s_i and c_j denotes a view- and level-specific code occurrence. A valid interleaving is therefore precisely a linear extension of $\mathcal{P}_y$.

Proposition 3 (Canonical States as Order Ideals)**.** *For every (a, b) with $0 \le a \le m$ and $0 \le b \le n$, the canonical state*

$$\sigma_{a,b} = (S_{1:a}, C_{1:b}) \tag{26}$$

corresponds uniquely to the order ideal

$$I_{a,b} = \{s_1, \ldots, s_a, c_1, \ldots, c_b\}. \tag{27}$$

Moreover, valid interleavings of the two identifier chains are in bijection with maximal chains in the lattice of order ideals $J(\mathcal{P}_y)$. Consequently,

$$|J(\mathcal{P}_y)| = (m+1)(n+1), \qquad |\Pi_{m,n}| = \binom{m+n}{m}. \tag{28}$$

Proof. Because the within-view orders are fixed, any order ideal of $\mathcal{P}_y$ must contain a prefix of the semantic chain and a prefix of the collaborative chain. It is therefore uniquely determined by a pair (a, b) and has the form $I_{a,b}$. Conversely, every $I_{a,b}$ is downward closed and hence is an order ideal. This establishes a bijection between canonical states and order ideals.

Given a valid interleaving, revealing its elements sequentially produces a strictly increasing sequence of order ideals from $I_{0,0}$ to $I_{m,n}$, with exactly one element added at each step. This sequence is therefore a maximal chain in $J(\mathcal{P}_y)$. Conversely, every maximal chain from $I_{0,0}$ to $I_{m,n}$ adds exactly one available element at each step, and the resulting sequence of added elements is a linear extension of $\mathcal{P}_y$. Thus, valid interleavings and maximal chains are in bijection.

Since an order ideal is uniquely indexed by $(a, b) \in \{0, \dots, m\} \times \{0, \dots, n\}$, we have $|J(\mathcal{P}_y)| = (m + 1)(n + 1)$. Finally, a valid interleaving is determined by choosing the m semantic positions among the $m + n$ generation steps, yielding $|\Pi_{m,n}| = \binom{m+n}{m}$. □

C Canonical-State Dynamic Programming for Exact Order Marginalization

C.1 Proof of Theorem 1

Theorem 1 follows from the path-invariance condition in Eq. (10), under which trajectories reaching the same canonical state share the same continuation distribution.

Proof. Fix a training pair (x, y) with semantic and collaborative identifiers $S_y = (s_1, \dots, s_m)$ and $C_y = (c_1, \dots, c_n)$. Let $\mathcal{R}_y$ denote the set of all target-consistent partial trajectories.

For any $\rho \in \mathcal{R}_y$, let $a(\rho)$ and $b(\rho)$ denote the numbers of semantic and collaborative codes revealed by ρ, respectively. Since the within-view orders are fixed, its canonical state is

$$\sigma(\rho) = \sigma_{a(\rho),b(\rho)} = \big(S_{1:a(\rho)}, C_{1:b(\rho)}\big). \tag{29}$$

For each canonical state $\sigma_{a,b}$, define the set of trajectories reaching it as

$$\mathcal{R}_{a,b} = \{\rho \in \mathcal{R}_y : \sigma(\rho) = \sigma_{a,b}\}, \tag{30}$$

and let their total probability mass be

$$\alpha_{a,b} = \sum_{\rho \in \mathcal{R}_{a,b}} P_\theta(\rho \mid x). \tag{31}$$

For any non-initial state, partition the trajectories reaching it according to the view selected by their last transition.

$$\mathcal{R}_{a,b} = \mathcal{R}^S_{a,b} \dot\cup \mathcal{R}^C_{a,b}, \tag{32}$$

where an empty subset is omitted at the corresponding boundary.

For $a > 0$, every trajectory in $\mathcal{R}^S_{a,b}$ is obtained by extending a unique trajectory in $\mathcal{R}_{a-1,b}$ with the target-consistent semantic transition (S, s_a). By the path-invariance condition,

$$\sum_{\rho \in \mathcal{R}^S_{a,b}} P_\theta(\rho \mid x) = \sum_{\bar\rho \in \mathcal{R}_{a-1,b}} P_\theta(\bar\rho \mid x)\, p_\theta(S, s_a \mid x, \sigma_{a-1,b}) \tag{33}$$

$$= \alpha_{a-1,b}\, p_\theta(S, s_a \mid x, \sigma_{a-1,b}). \tag{34}$$

Similarly, for $b > 0$,

$$\sum_{\rho \in \mathcal{R}^C_{a,b}} P_\theta(\rho \mid x) = \alpha_{a,b-1}\, p_\theta(C, c_b \mid x, \sigma_{a,b-1}). \tag{35}$$

Since the two subsets are disjoint, their probability masses add, yielding the canonical-state recurrence

$$\alpha_{a,b} = \mathbb{I}[a > 0]\, \alpha_{a-1,b}\, p_\theta(S, s_a \mid x, \sigma_{a-1,b}) + \mathbb{I}[b > 0]\, \alpha_{a,b-1}\, p_\theta(C, c_b \mid x, \sigma_{a,b-1}), \tag{36}$$

with $\alpha_{0,0} = 1$.

At the terminal state $\sigma_{m,n}$, target-consistent complete trajectories are in bijection with the valid interleavings $\Pi_{m,n}$ because both within-view orders are fixed. Therefore,

$$\alpha_{m,n} = \sum_{\rho \in \mathcal{R}_{m,n}} P_\theta(\rho \mid x) = \sum_{\pi \in \Pi_{m,n}} P_\theta(y, \pi \mid x) = P_\theta(y \mid x). \tag{37}$$

Thus, all trajectories reaching the same canonical state can be merged by summing their probability mass without changing the exact item-level marginal likelihood. □

C.2 Log-Space Dynamic Programming and Complexity

For numerical stability, let $A_{a,b} = \log \alpha_{a,b}$ and define the target-consistent edge log-probabilities as

$$\ell^S_{a,b} = \log p_\theta(S, s_{a+1} \mid x, \sigma_{a,b}), \qquad \ell^C_{a,b} = \log p_\theta(C, c_{b+1} \mid x, \sigma_{a,b}). \tag{38}$$

Initialize $A_{0,0} = 0$. For $a > 0$ and $b > 0$,

$$A_{a,b} = \mathrm{LSE}\left(A_{a-1,b} + \ell^S_{a-1,b}, A_{a,b-1} + \ell^C_{a,b-1}\right), \tag{39}$$

where $\mathrm{LSE}(u, v) = \log(e^u + e^v)$. The boundary states satisfy

$$A_{a,0} = A_{a-1,0} + \ell^S_{a-1,0}, \qquad A_{0,b} = A_{0,b-1} + \ell^C_{0,b-1}. \tag{40}$$

The exact marginal negative log-likelihood is therefore

$$\mathcal{L}_{\mathrm{marg}} = -A_{m,n}. \tag{41}$$

The canonical-state lattice contains $(m+1)(n+1)$ states and

$$m(n+1) + n(m+1) = 2mn + m + n \tag{42}$$

directed edges. Once the neural edge scores are available, exact marginalization requires $O((m+1)(n+1)) = O(mn)$ arithmetic operations, rather than explicit enumeration of the $\binom{m+n}{m}$ valid interleavings.

D Proof of Proposition 2

Proof. For a training pair (x, y), denote

$$P_\pi = P_\theta(y, \pi \mid x), \qquad Z = \sum_{\pi \in \Pi_{m,n}} P_\pi, \qquad r_\pi = \frac{P_\pi}{Z}. \tag{43}$$

Here, $Z = P_\theta(y \mid x)$ and $r_\pi = r_\theta(\pi \mid x, y)$. Let $K = |\Pi_{m,n}|$ and $U_\pi = 1/K$.

By definition,

$$D_{\mathrm{KL}}(U \| r) = \sum_{\pi \in \Pi_{m,n}} U_\pi \log \frac{U_\pi}{r_\pi} \tag{44}$$

$$= \frac{1}{K} \sum_{\pi \in \Pi_{m,n}} \log \frac{1/K}{P_\pi / Z} \tag{45}$$

$$= -\log K - \frac{1}{K} \sum_{\pi \in \Pi_{m,n}} \log P_\pi + \log Z. \tag{46}$$

Using

$$\mathcal{L}_{\mathrm{marg}} = -\log Z, \qquad \mathcal{L}_{\mathrm{uni}} = -\frac{1}{K} \sum_{\pi \in \Pi_{m,n}} \log P_\pi, \tag{47}$$

we obtain

$$D_{\mathrm{KL}}(U_y \| r) = \mathcal{L}_{\mathrm{uni}} - \left(\mathcal{L}_{\mathrm{marg}} + \log K\right), \tag{48}$$

which proves Eq. (21).

Substituting Eq. (48) into Eq. (20) gives

$$\mathcal{L}_{\mathrm{train}} = \mathcal{L}_{\mathrm{marg}} + \lambda D_{\mathrm{KL}}(U_y \| r) \tag{49}$$

$$= (1 - \lambda)\mathcal{L}_{\mathrm{marg}} + \lambda \mathcal{L}_{\mathrm{uni}} - \lambda \log K, \tag{50}$$

which proves Eq. (22). □

Jensen-Gap Interpretation. Since

$$\frac{1}{K}\sum_{\pi\in\Pi_{m,n}} P_\pi = \frac{Z}{K}, \tag{51}$$

the convexity of $-\log(\cdot)$ gives

$$\mathcal{L}_{\text{uni}} = -\frac{1}{K}\sum_{\pi\in\Pi_{m,n}} \log P_\pi \tag{52}$$

$$\geq -\log\left(\frac{1}{K}\sum_{\pi\in\Pi_{m,n}} P_\pi\right) \tag{53}$$

$$= \mathcal{L}_{\text{marg}} + \log K. \tag{54}$$

Hence,

$$\mathcal{L}_{\text{uni}} - (\mathcal{L}_{\text{marg}} + \log K) = D_{\text{KL}}(U_y \| r) \geq 0. \tag{55}$$

The gap is zero if and only if $r_\pi = 1/K$ for every $\pi \in \Pi_{m,n}$.

Lower-Bounded Path-Gradient Weights. Let $\gamma_\pi = \log P_\theta(y, \pi \mid x)$. Differentiating Eq. (22) with respect to γ_π gives

$$\frac{\partial \mathcal{L}_{\text{train}}}{\partial \gamma_\pi} = (1-\lambda)\frac{\partial \mathcal{L}_{\text{marg}}}{\partial \gamma_\pi} + \lambda \frac{\partial \mathcal{L}_{\text{uni}}}{\partial \gamma_\pi} \tag{56}$$

$$= -(1-\lambda)\frac{\exp \gamma_\pi}{\sum_{\pi'\in\Pi_{m,n}} \exp \gamma_{\pi'}} - \frac{\lambda}{K} \tag{57}$$

$$= -\left[(1-\lambda)r_\pi + \frac{\lambda}{K}\right]. \tag{58}$$

Therefore, the effective gradient weight is

$$w_\pi = (1-\lambda)r_\pi + \frac{\lambda}{K}. \tag{59}$$

Since $\sum_\pi r_\pi = 1$, these weights satisfy

$$\sum_{\pi\in\Pi_{m,n}} w_\pi = 1, \qquad w_\pi \geq \frac{\lambda}{K}. \tag{60}$$

For $\lambda > 0$, every valid interleaving receives a gradient weight of at least λ/K. A posterior-dependent component is retained when $\lambda < 1$, whereas $\lambda = 1$ gives uniform path weights.

Interpretation of the $\lambda = 1$ endpoint. When $\lambda = 1$, the training objective reduces to $\mathcal{L}_{\text{uni}} - \log K$, and all valid interleavings receive equal gradient weight. This corresponds to the uniform-path endpoint of the coverage objective. More generally, λ controls how supervision is distributed across valid interleavings, rather than whether cross-view order is treated as latent. Thus, $\lambda = 1$ changes the training weights over interleavings but does not designate any particular interleaving as the target or reduce the model to a fixed-order formulation. All valid interleavings remain represented through the shared canonical-state lattice, and uniform training weights do not require their learned path probabilities to be uniform. At inference time, item scores are still obtained by aggregating probability mass over alternative interleavings through the state-merged decoding procedure in Section 3.4. As shown in Table 4, marginal inference consistently outperforms Max-Path inference under the same checkpoint, indicating that aggregation over multiple interleavings remains beneficial even at the uniform-path-training endpoint. Moreover, the method does not rely on $\lambda = 1$: Figure 4 shows performance gains across a broad range of positive λ, including substantial improvements at $\lambda = 0.2$. Therefore, $\lambda = 1$ should be viewed as a valid uniform-path-training endpoint of the latent-interleaving framework, rather than a return to fixed-order supervision.

E UNIFORM-PATH LOSS ON THE CANONICAL-STATE LATTICE

Although $\mathcal{L}_{\text{uni}}$ is defined by averaging over $K = |\Pi_{m,n}| = \binom{m+n}{m}$ complete interleavings, it can be computed exactly on the canonical-state lattice without explicit path enumeration.

For notational convenience, we use π both for a valid interleaving and for its corresponding complete path through the target-consistent lattice. Let $\mathcal{E}_y$ denote the set of target-consistent lattice edges. For each edge $e : \sigma \to \sigma' \in \mathcal{E}_y$, let ℓ_e denote its transition log-probability. Under the path-invariance condition, ℓ_e depends only on the source canonical state and the target-consistent transition, rather than on the particular arrival history leading to that state. Hence, for every valid interleaving $\pi \in \Pi_{m,n}$,

$$\log P_\theta(y, \pi \mid x) = \sum_{e \in \pi} \ell_e. \tag{61}$$

Substituting Eq. (61) into the definition of the uniform-path loss gives

$$\mathcal{L}_{\text{uni}} = -\frac{1}{K} \sum_{\pi \in \Pi_{m,n}} \sum_{e \in \pi} \ell_e. \tag{62}$$

Lemma 4 (Numbers of Path Prefixes and Suffixes)**.** *For a canonical state $\sigma_{a,b}$, the numbers of valid path prefixes from $\sigma_{0,0}$ to $\sigma_{a,b}$ and valid path suffixes from $\sigma_{a,b}$ to $\sigma_{m,n}$ are*

$$F(\sigma_{a,b}) = \binom{a+b}{a}, \qquad R(\sigma_{a,b}) = \binom{m+n-a-b}{m-a}. \tag{63}$$

Proof. Any path prefix reaching $\sigma_{a,b}$ contains exactly a semantic steps and b collaborative steps. Choosing the positions of the a semantic steps among the $a+b$ transitions uniquely determines the prefix, yielding $\binom{a+b}{a}$ possibilities. Likewise, any suffix from $\sigma_{a,b}$ contains exactly $m-a$ semantic steps and $n-b$ collaborative steps, yielding $\binom{m+n-a-b}{m-a}$ possibilities. □

Proposition 5 (Exact Edge Decomposition of the Uniform-Path Loss)**.** *For a target-consistent edge $e : \sigma \to \sigma'$, define its uniform-path inclusion probability as*

$$w_e = \Pr_{\pi \sim U_y}(e \in \pi) = \frac{F(\sigma) R(\sigma')}{K}. \tag{64}$$

Then the uniform-path loss admits the exact lattice decomposition

$$\mathcal{L}_{\text{uni}} = -\sum_{e \in \mathcal{E}_y} w_e \ell_e. \tag{65}$$

Proof. A complete lattice path containing an edge $e : \sigma \to \sigma'$ is uniquely determined by a path prefix from $\sigma_{0,0}$ to σ, the edge e, and a path suffix from σ' to $\sigma_{m,n}$. Therefore, exactly $F(\sigma)R(\sigma')$ complete interleavings contain e, which gives Eq. (64) after division by K.

Exchanging the two finite sums in Eq. (62) gives

$$\mathcal{L}_{\text{uni}} = -\frac{1}{K} \sum_{\pi \in \Pi_{m,n}} \sum_{e \in \pi} \ell_e \tag{66}$$

$$= -\sum_{e \in \mathcal{E}_y} \left(\frac{1}{K} \sum_{\pi \in \Pi_{m,n}} \mathbf{1}\{e \in \pi\} \right) \ell_e \tag{67}$$

$$= -\sum_{e \in \mathcal{E}_y} w_e \ell_e, \tag{68}$$

which proves Eq. (65). □

The coefficients w_e are edge-inclusion probabilities rather than a probability distribution over edges. Indeed,

$$\sum_{e \in \mathcal{E}_y} w_e = \mathbb{E}_{\pi \sim U_y} \left[\sum_{e \in \mathcal{E}_y} \mathbf{1}\{e \in \pi\} \right] = m + n, \tag{69}$$

since every complete interleaving contains exactly $m + n$ edges.

For fixed identifier lengths (m, n), the coefficients w_e depend only on the lattice geometry and can therefore be precomputed once. Moreover, the edge log-probabilities ℓ_e are precisely the target-consistent transition scores already required by the marginal-likelihood dynamic program. Thus, once these edge scores have been computed, evaluating $\mathcal{L}_{\text{uni}}$ requires no explicit interleaving enumeration and no additional model evaluations, but only a deterministic weighted reduction over the canonical-state lattice edges.

F Related Work

Generative recommendation. Semantic-ID-based generative recommendation represents each item with a discrete identifier and predicts the target item by generating its identifier, with TIGER establishing a representative paradigm based on residual-quantized semantic IDs and autoregressive decoding (Rajput et al., 2023). Subsequent studies have improved how such identifiers are constructed and optimized for recommendation. CoST augments semantic tokenization with contrastive quantization to better preserve item relationships (Zhu et al., 2024), while ETEGRec couples item tokenization with the downstream recommender through recommendation-oriented alignment and alternating optimization (Liu et al., 2025b). More recent work further develops recommendation-native representation learning and quantization (Liang et al., 2026), differentiable semantic IDs jointly optimized with recommendation objectives (Fu et al., 2026), and multi-tokenizer pre-training for generative recommendation (Zheng et al., 2025). Another line of research revisits the generation process itself. RPG removes sequential dependencies among identifier tokens to enable parallel prediction (Hou et al., 2025a), while LLaDA-Rec replaces fixed left-to-right decoding with discrete diffusion and adaptive generation order (Shi et al., 2025). Latte analyzes the expressiveness limitations induced by autoregressive semantic-ID trees and introduces latent tokens to diversify decoding paths (Hou et al., 2026). These studies improve identifier construction or reconsider token dependencies and decoding order, but do not address the serialization ambiguity that arises when multiple internally ordered identifier chains are jointly generated. Our work treats the unspecified cross-view interleaving as a latent variable and marginalizes over all valid interleavings, rather than imposing a fixed serialization.

Collaborative information in generative recommendation. A parallel line of research investigates how interaction-derived collaborative information can complement content semantics in generative recommendation. Early work systematically studies collaborative and hybrid item indexing for LLM-based recommendation (Hua et al., 2023). ColaRec constructs generative identifiers from pretrained collaborative representations and aligns content-based semantic representations with the collaborative identifier space (Wang et al., 2024c), while LETTER incorporates collaborative regularization directly into residual-quantization-based item tokenization (Wang et al., 2024a). UTGRec further integrates collaborative knowledge through co-occurrence alignment and reconstruction during universal item tokenization (Zheng et al., 2026), DOS leverages collaborative signals to align the semantic-ID codebook space with the downstream generation space (Yin et al., 2026), and SCRec incorporates collaborative signals into semantic tokenization while introducing semantic information into the generation stage (Dan et al., 2026). Other approaches model semantic and collaborative information more explicitly: EAGER employs separate behavioral and semantic decoding streams (Wang et al., 2024b), DiscRec disentangles the two signals through separate modeling branches followed by adaptive fusion (Liu et al., 2025a), and SETRec++ derives semantic and collaborative tokens from separate tokenizers and represents them as an order-agnostic set identifier (Lin et al., 2026). These studies primarily focus on injecting, aligning, or fusing collaborative information with semantic representations during identifier construction or recommendation modeling. In contrast, we preserve semantic and collaborative representations as two structurally distinct, internally ordered identifier chains and defer their interaction to the generation stage.

Flexible-order and latent-structure generation. Generation order has also been relaxed beyond conventional left-to-right decoding through insertion-based models that support flexible or adaptively inferred token orders (Stern et al., 2019; Gu et al., 2019). Relatedly, latent-alignment models such as CTC and RNN-T marginalize over multiple valid alignment paths using dynamic programming (Graves et al., 2006; Graves, 2012). Our setting differs in that the latent structure is the interleaving of two internally ordered identifier chains. By sharing predictive states across trajectories with identical revealed view-specific prefixes, our canonical-state formulation enables exact marginalization over all valid cross-view interleavings.

Table 8: Dataset statistics after preprocessing. Avg. T denotes the average user history length before the test target, without truncation.

Dataset	Users	Items	Interactions	Avg. T
Instruments	57,439	24,587	511,836	7.9109
Games	94,762	25,612	814,586	7.5961
Steam	32,919	14,330	1,436,230	42.6292

G Experimental Details

G.1 Data and Preprocessing

We evaluate on three datasets: Instruments and Games from the Musical Instruments and Video Games categories of Amazon Reviews 2023 (Hou et al., 2024), and Steam from the processed data released with SETRec++ (Lin et al., 2026), which is based on the Steam review dataset used in SASRec (Kang & McAuley, 2018). For the two Amazon datasets, we use the released 5-core subsets. For each user, we adopt a leave-one-out protocol: the last interaction is used for testing, the second-to-last for validation, and remaining interactions for training. For Steam, we group each user's review history into a single sequence, sort it chronologically, and then apply the same leave-one-out protocol. For all datasets, the model uses at most the 20 most recent interactions as history input.

G.2 Baselines

We compare LatticeRec with two groups of representative methods, including conventional sequential recommendation methods and generative recommendation methods.

Sequential recommendation. **GRU4Rec** (Hidasi et al., 2016) models user interaction sequences with gated recurrent units. **SASRec** (Kang & McAuley, 2018) applies causal self-attention for next-item prediction, while **BERT4Rec** (Sun et al., 2019) learns bidirectional sequence representations through masked-item prediction. **FDSA** (Zhang et al., 2019) jointly models item- and feature-level information with self-attention. **HSTU** (Zhai et al., 2024) models user behavior sequences with hierarchical sequential transduction units.

Generative recommendation. **TIGER** (Rajput et al., 2023) quantizes item representations into semantic identifiers and generates them autoregressively. **SETRec++** (Lin et al., 2026) incorporates semantic and collaborative information into item representation and generation. **SCRec** (Dan et al., 2026) enhances cross-stage coherence by incorporating collaborative signals into semantic tokenization and semantic information into the generation stage. **LLaDA-Rec** (Shi et al., 2025) models semantic-ID generation with discrete diffusion rather than strict left-to-right autoregressive generation. **LETTER** (Wang et al., 2024a) learns item tokenization with semantic, collaborative, and token-distribution objectives. **EAGER** (Wang et al., 2024b) employs a two-stream generative architecture with a shared encoder and separate decoders for behavioral and semantic tokens, promoting collaboration between the two information sources. **ActionPiece** (Hou et al., 2025b) constructs context-dependent tokens from item-feature action sequences. **MTGRec** (Zheng et al., 2025) augments pre-training sequences with multiple tokenizers and applies curriculum pre-training before single-tokenizer fine-tuning.

Table 9: Best hyperparameter settings of LatticeRec on the three datasets.

Hyperparameter	Instruments	Games	Steam
λ	1	1	0.6
Learning Rate	2×10^{-3}	3×10^{-3}	2×10^{-3}
Dropout	0.1	0.1	0.1
Weight Decay	0.01	0.05	0.05
Attention Heads	6	6	6
Encoder Layers	4	4	4
Decoder Layers	4	4	4
FFN Hidden Dimension	1024	1024	1024

For all baselines, we follow the configurations reported in the original papers and reproduce the results using their officially released implementations.

G.3 Implementation and Evaluation Settings

Identifier construction. Sentence-T5 (Ni et al., 2022) and a SASRec model trained only on training interactions provide semantic and collaborative item representations, respectively. We separately quantize the two types of representations using RQ-Kmeans. Each view uses three quantization levels ($m = n = 3$), with 256 codewords per level. The resulting semantic and collaborative identifiers are constructed offline and remain fixed throughout recommendation model training. For the rare cases where multiple items share the same identifier pair, we append a terminal disambiguation token after reaching $\sigma_{m,n}$. Since this token is conditioned on the same terminal canonical state for all interleavings, it does not affect the order-marginalization recurrence.

LatticeRec. We adopt a T5-style encoder–decoder architecture (Raffel et al., 2020). Specifically, both the history encoder and the state decoder consist of 4 Transformer blocks, with a hidden dimension of 128 and a feed-forward width of 1024. To construct the canonical-state representation, we use a SwiGLU module (Shazeer, 2020) with width 1024 to map the revealed semantic and collaborative prefixes into a state-query representation. Each user's history sequence is encoded only once. The code-content component of each unrevealed slot is set to zero, while its view and level embeddings are retained. Consequently, different generation paths reaching the same canonical state share the same state representation and reuse the same outgoing-edge scores.

Training and inference. We tune the learning rate from $\{2 \times 10^{-3}, 3 \times 10^{-3}, 5 \times 10^{-3}\}$, dropout from $\{0, 0.05, 0.1\}$, weight decay from $\{0, 0.01, 0.05\}$, and the warmup ratio from $\{10\%, 13.5\%, 15\%\}$. All hyperparameters are selected based on validation NDCG@10. Models are trained for 100 epochs without early stopping, and the checkpoint with the best validation NDCG@10 is used for testing. Linear warmup is followed by cosine learning-rate decay. The effective batch size is 3072 for all three datasets, and all experiments are conducted on two NVIDIA PRO 6000 GPUs. During inference, we use catalog-constrained state-merged beam search with a beam size of $B = 1000$, aggregating the probability masses of trajectories reaching the same canonical state before beam pruning. The main results use the validation-selected hyperparameters reported in Table 9. Unless otherwise specified, controlled analyses fix $\lambda = 1$ on all three datasets to isolate the factor under study.

Evaluation. We report Recall@k and NDCG@k for $k \in \{5, 10\}$. For LatticeRec and its controlled variants, checkpoints are selected using validation NDCG@10, and all test metrics are computed from the same selected checkpoint. Hyperparameters and fixed generation orders are selected without using test results. Baselines retain the validation-based checkpoint-selection criteria of their respective reproduction pipelines.

H Highest-Posterior Interleaving Frequencies

For each test target, we compute the posterior probability of every valid interleaving and record the path with the highest posterior probability. With three semantic and three collaborative codes, there

are $\binom{6}{3} = 20$ valid order-preserving interleavings. Figure 8 reports the fraction of test examples assigned to each highest-posterior path.

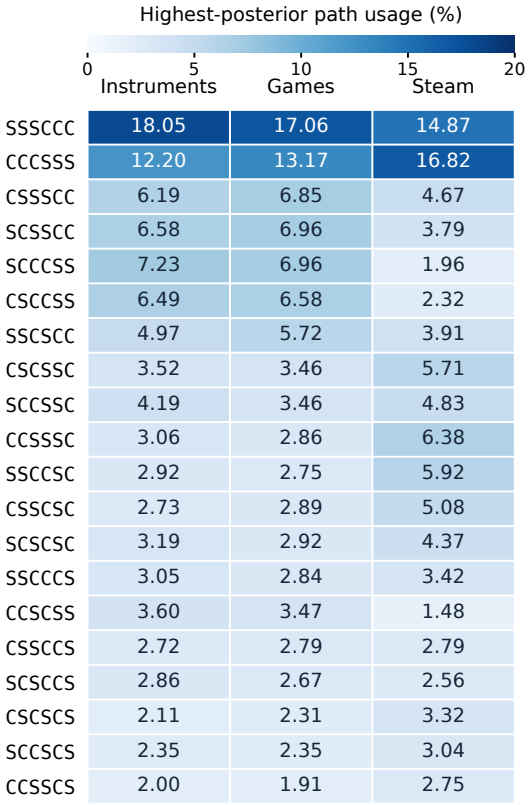


Figure 8: Highest-posterior path usage across all 20 valid interleavings on the three datasets.

I Posterior Statistics

For an evaluation instance (x, y), the posterior probability of a valid interleaving $\pi \in \Pi_{m,n}$ is

$$r_\theta(\pi \mid x, y) = \frac{P_\theta(y, \pi \mid x)}{P_\theta(y \mid x)}. \tag{70}$$

We measure the concentration of this posterior using its entropy

$$H(r) = -\sum_{\pi \in \Pi_{m,n}} r_\theta(\pi \mid x, y) \log r_\theta(\pi \mid x, y). \tag{71}$$

A low entropy indicates that posterior mass is concentrated on a small number of interleavings, whereas a higher entropy indicates that the mass is more broadly distributed.

For a more interpretable measure of posterior support, we additionally report the effective number of paths,

$$N_{\text{eff}} = \exp(H(r)). \tag{72}$$

This quantity can be interpreted as the number of equally probable interleavings that would yield the same posterior entropy. It ranges from 1 when all posterior mass is concentrated on a single path to $|\Pi_{m,n}|$ when the posterior is uniform over all valid interleavings. For dataset-level analysis, we compute $H(r)$ and N_{eff} separately for each evaluation instance and then average each statistic over instances.

Table 10: Comparison with random-order training on Games. Random-Order samples one valid interleaving for each training instance, whereas LatticeRec uses lattice-based supervision over all valid interleavings.

Method	R@5	R@10	N@5	N@10
Random-Order	0.0625	0.0940	0.0420	0.0521
LatticeRec ($\lambda = 1$)	**0.0641**	**0.0964**	**0.0432**	**0.0536**

Table 11: Recommendation performance with full decoding and early termination. Early termination produces negligible changes in ranking quality across all datasets. ΔN@10 is computed from unrounded scores.

Dataset	Inference	R@5	R@10	N@5	N@10	ΔN@10 (%)
Instruments	Full Decoding	0.0406	0.0619	0.0271	0.0339	–
	Early Termination	0.0406	0.0619	0.0270	0.0339	-0.20
Games	Full Decoding	0.0641	0.0964	0.0432	0.0536	–
	Early Termination	0.0639	0.0963	0.0431	0.0536	-0.03
Steam	Full Decoding	0.0644	0.0979	0.0436	0.0544	–
	Early Termination	0.0646	0.0979	0.0437	0.0544	$+0.02$

J DISCUSSION

Comparison with Random-Order Training. We further examine whether stochastic exposure to different valid serializations is sufficient to obtain the gains of lattice-based training. We conduct this controlled comparison on Games, which is also used for the fixed-order analysis in Section 4.4. Whenever a training pair (x, y) is sampled, Random-Order uniformly draws one valid interleaving $\pi \in \Pi_{m,n}$ and optimizes only its path loss,

$$\mathcal{L}_{\text{rand}} = -\log P_\theta(y, \pi \mid x), \qquad \pi \sim U_y. \tag{73}$$

Since

$$\mathbb{E}_{\pi \sim U_y}\left[\mathcal{L}_{\text{rand}}\right] = \mathcal{L}_{\text{uni}}, \tag{74}$$

Random-Order is a single-path Monte Carlo estimator of the uniform-path objective. For $\lambda = 1$, LatticeRec optimizes the same objective up to the constant $-\log K$, but computes supervision over all valid interleavings exactly on the canonical-state lattice. We keep the model architecture, identifiers, optimization settings, and state-merged inference unchanged, varying only how training supervision over interleavings is computed. As shown in Table 10, exact lattice-based training consistently outperforms Random-Order on Games. This indicates that stochastic random-order exposure alone does not match the effectiveness of explicitly incorporating all valid interleavings during each training update.

Effect of Early Termination. We further compare the results before and after applying the inference acceleration strategy introduced in Figure 7, and report the comparison in Table 11. Specifically, we compare full decoding with early termination, which stops expanding a state once its revealed semantic and collaborative prefixes uniquely identify a catalog item, while keeping all other inference settings unchanged. As shown in Table 11, early termination introduces only negligible changes in recommendation quality across all three datasets. The absolute relative change in NDCG@10 is within 0.20%, with similarly small differences in the other metrics. These results indicate that early termination effectively reduces unnecessary decoding with virtually no impact on ranking quality.

How Efficiently Does LatticeRec Converge? We compare the validation NDCG@10 curves of LatticeRec and TIGER on the three datasets. As shown in Figure 9, the convergence curves demonstrate faster convergence in terms of training epochs. On Instruments, Games, and Steam, LatticeRec surpasses TIGER's highest validation NDCG@10 among the reported evaluation points at epochs 24, 28, and 32, respectively, and continues to improve with further training. These results indicate that LatticeRec achieves strong recommendation performance with fewer training epochs.

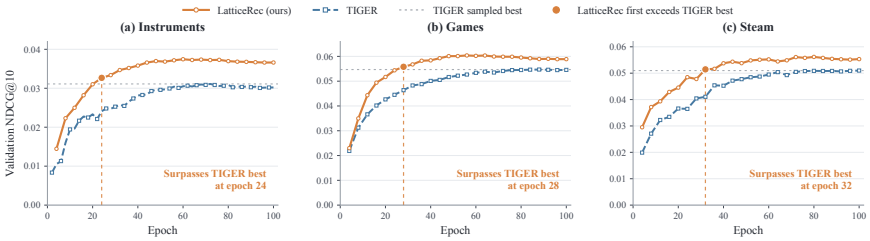


Figure 9: Validation NDCG@10 convergence curves of LatticeRec and TIGER on the three datasets. The marked points denote the first epochs at which LatticeRec surpasses TIGER's best validation performance.

Table 12: Effect of model hidden dimension on Instruments.

Model	R@5	N@5	R@10	N@10
LatticeRec (128-dim)	0.0406	0.0271	0.0619	0.0339
LatticeRec (144-dim)	0.0406	0.0271	0.0624	0.0341
LatticeRec (168-dim)	0.0409	0.0268	**0.0625**	0.0338
LatticeRec (196-dim)	**0.0416**	**0.0276**	0.0620	**0.0342**

Table 13: Performance comparison with TIGER under the original and parameter-matched settings. The best and second-best results on each dataset are highlighted in **bold** and underlined, respectively.

Dataset	Model	Params. (M)	FFN Dim.	R@5	N@5	R@10	N@10
Instruments	TIGER (Original)	4.5911	1024	0.0368	0.0238	0.0580	0.0306
	TIGER (Matched Params.)	6.7579	2082	0.0388	0.0251	0.0609	0.0322
	LatticeRec	6.7588	1024	**0.0406**	**0.0271**	**0.0619**	**0.0339**
Games	TIGER (Original)	4.5911	1024	0.0571	0.0374	0.0902	0.0481
	TIGER (Matched Params.)	6.7579	2082	0.0550	0.0360	0.0874	0.0464
	LatticeRec	6.7588	1024	**0.0641**	**0.0432**	**0.0964**	**0.0536**
Steam	TIGER (Original)	4.5911	1024	0.0571	0.0379	0.0916	0.0489
	TIGER (Matched Params.)	6.7579	2082	0.0564	0.0372	0.0888	0.0476
	LatticeRec	6.7588	1024	**0.0644**	**0.0436**	**0.0979**	**0.0544**

Can LatticeRec Benefit from Representation Scaling? LatticeRec performs subsequent generative modeling over multi-view representations, which raises a natural question: can the model further benefit from a larger representation space as the representation dimension increases?

To provide a preliminary investigation, we gradually increase the representation dimension from the default setting to 144, 168, and 196 on Instruments, while keeping the remaining model architecture and training configuration unchanged. As shown in Table 12, LatticeRec exhibits an overall trend of performance improvement as the representation dimension increases. This simple preliminary experiment suggests that LatticeRec may exhibit some potential for representation scaling, with additional room for performance improvement as representation capacity increases. This phenomenon warrants further investigation with larger representation scales and more datasets.

Does the Performance Gain Come from a Larger Parameter Budget? To examine this question, we increase the FFN size of TIGER so that its total parameter count is comparable to that of LatticeRec, while keeping the rest of TIGER unchanged. As shown in Table 13, under a matched parameter budget, LatticeRec outperforms TIGER across all four metrics on all three datasets.

Notably, increasing the model capacity of TIGER improves performance only on Instruments, while yielding no gains on Games or Steam. This suggests that simply enlarging model capacity does not consistently improve recommendation performance and cannot explain the gains achieved by LatticeRec. These results indicate that the advantage of LatticeRec is not primarily driven by a larger

parameter budget, and further support the effectiveness of its dual-view structural modeling and order-marginalization design.

Table 14: Statistics of the MPD subset after preprocessing.

Dataset	Playlists	Items	Interactions	Avg. T
MPD	9,025	64,967	323,564	35.85

Table 15: Performance comparison of different fixed orders and LatticeRec on MPD. Best and second-best results are boldfaced and underlined, respectively.

Generation Order	R@5	N@5	R@10	N@10
Playlist → Tag → Meta	0.0471	0.0312	0.0760	0.0405
Playlist → Meta → Tag	0.0507	0.0332	0.0783	0.0422
Tag → Playlist → Meta	0.0491	0.0340	0.0803	0.0440
Tag → Meta → Playlist	0.0509	0.0343	0.0790	0.0433
Meta → Playlist → Tag	<u>0.0547</u>	<u>0.0362</u>	0.0807	0.0445
Meta → Tag → Playlist	0.0533	0.0357	<u>0.0832</u>	<u>0.0452</u>
LatticeRec	**0.0551**	**0.0376**	**0.0901**	**0.0489**

Does LatticeRec Remain Effective Beyond Two Views? Canonical-state sharing and path marginalization in LatticeRec are not restricted to the dual-view setting. With additional views, the state can record the revealed prefix of each view, while each generation step can extend any unfinished view. To provide a preliminary evaluation of this extensibility, we conduct a three-view experiment on the Million Playlist Dataset (MPD) (Chen et al., 2018). Detailed statistics of the dataset are provided in Table 14. For each song, we construct three complementary views. The Playlist view is a collaborative embedding learned from song–playlist interactions, the Tag view is obtained by mean-pooling the embeddings of its associated tags, and the Meta view encodes the concatenation of the track title, artist name, and album name. Each view is quantized into three tokens with a codebook size of 256 at each level. In the three-view setting, this results in $(3+1)^3 = 64$ canonical states and $|\Pi_{3,3,3}| = \frac{9!}{3!3!3!} = 1680$ valid interleaving paths. We compare LatticeRec with all six block-wise fixed view orders, where the three tokens of each view are generated contiguously. For LatticeRec, we set $\lambda = 0.2$, while keeping the remaining configurations the same as those used on Instruments. As shown in Table 15, LatticeRec outperforms the strongest fixed order on all four metrics. These exploratory results provide preliminary evidence that canonical-state sharing and path marginalization remain effective in the three-view setting, and demonstrate the feasibility of extending LatticeRec to more general multi-view scenarios.