MLPs ARE EFFICIENT DISTILLED GENERATIVE RECOMMENDERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation models employing Semantic IDs (SIDs) exhibit strong potential, yet their practical deployment is bottlenecked by the high inference latency of beam-expanded autoregressive decoding. In this work, we identify that standard attention-heavy Transformer decoders represent a structural overkill for this task: the hierarchical nature of SIDs makes prediction difficulty drop sharply after the first token, rendering repeated attention computations highly redundant. Driven by this insight, we propose SID-MLP, a lightweight MLP-centric distillation framework that fundamentally simplifies the decoding paradigm for GR. Instead of executing complex, step-by-step attention mechanisms, our approach captures the global user context in a single operation, decoupled from sequential token prediction. We then distill the heavy autoregressive teacher into position-specific MLP heads, eliminating the dense attention overhead while preserving prefix and context dependencies. Extensive experiments demonstrate that SID-MLP matches the accuracy of teacher models while accelerating inference by $8.74\times$. Crucially, this distillation strategy can serve as a plug-and-play accelerator for different backbones and tokenizer settings. Furthermore, we introduce SID-MLP++, extending our distillation framework to replace the Transformer encoder, unlocking further latency reductions. Ultimately, our work reveals that decoder-side MLPs distillation is an effective acceleration path for structured SID recommendation, while full encoder replacement offers an additional speed–accuracy trade-off. Our code is available at: https://anonymous.4open.science/r/SID-MLP.

1 INTRODUCTION

Generative recommendation (GR) approaches (Tay et al., 2022; Rajput et al., 2023; Zheng et al., 2024; Deng et al., 2025) diverge from the conventional sequential recommendation paradigm (Hidasi et al., 2016; Kang & McAuley, 2018), which models user histories with atomic item IDs; instead, each item is often represented by a semantic ID (SID), an ordered tuple of discrete tokens from a compact shared vocabulary. This formulation enables semantically similar items to share token prefixes (improving generalization) and avoid large embedding tables which scale with item cardinality (alleviating embedding sparsity). Using this formulation, prior work (Rajput et al., 2023; Zheng et al., 2024; Deng et al., 2025; He et al., 2026) frames next-item prediction as autoregressive SID generation and reports promising performance.

However, the high inference latency of GR has hindered its deployment in production systems. Consider a GR model that represents each item with a 4-token SID. Unlike conventional models, which can produce top-K predictions with a single model forward pass (Hidasi et al., 2016; Kang & McAuley, 2018), GR models usually rely on beam search to generate multiple candidate SID sequences, with each sequence requiring four model forward passes due to the autoregressive generation mechanism. To achieve competitive performance, the beam size B is often set even larger than K, leading to a total of roughly $4 \times B$ model forward passes during inference. This cost can be prohibitive in recommender systems, where latency is critical to user experience.

Developing broadly applicable methods for accelerating GR is non-trivial. Speculative decoding (Leviathan et al., 2023; Cai et al., 2024; Li et al., 2024b) has been widely used to accelerate large language model inference. However, GR requires generating top-K candidate sequences, which makes the verification step difficult to adapt, as it would need to verify and rank multiple

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.

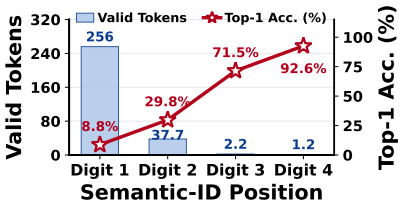


Figure 1: SID prefix-conditioned branching on Instruments. Bars show the average valid next-token choices after a valid prefix, distinct from marginal codebook utilization.

Table 1: Decoder depth redundancy on Instruments. Teacher generates first m digits; a 1-layer decoder predicts the remaining $4-m$.

Variant	N@10	Δ vs teacher
TIGER	0.0323	—
m=3	0.0323	±0.0%
m=2	0.0320	−0.9%
m=1	0.0320	−0.9%
m=0	0.0318	−1.5%

Table 2: Attention ablation on the m=0 1-layer decoder on Instruments. SA and CA denote self-attention and cross-attention, respectively.

SA	CA	N@10	Δ vs baseline
✓	✓	0.0318	—
×	✓	0.0295	−7.2%
✓	×	0.0130	−59.1%
×	×	0.0111	−65.1%

candidate sequences simultaneously (Lin et al., 2025b). Multi-token, parallel, and non-autoregressive methods (Hou et al., 2025a; Ren et al., 2024) remove the left-to-right decoding dependency, but often require jointly fine-tuning the teacher model or relying on a specific tokenizer, limiting their applicability to general model architectures and other recent advances in the field. Knowledge distillation (KD) (Hinton et al., 2015; Zhang et al., 2022) offers a promising pathway by transferring knowledge from a teacher model to a more efficient student model. However, standard same-family KD (*e.g.*, distilling into a smaller Transformer) only reduces model size while retaining much of the decoding time complexity.

Note that the decoding space of GR is typically well structured, leaving room for inference acceleration. Unlike large language models (Zhao et al., 2026), whose generations are open-ended and variable-length, GR models produce short and fixed-length outputs, such as 4-token SIDs in TIGER (Rajput et al., 2023), with a limited set of valid token sequences. As empirically observed in Figure 1, once the first few tokens are determined, the number of valid continuations conditioned on the prefix rapidly drops to only a few choices, far below the number of all theoretically possible combinations. This prefix-conditioned valid-continuation sparsity motivates a natural question: *Is an attention-based Transformer decoder unnecessarily heavy for inference in GR?*

To this end, we investigate whether a simpler model architecture can be used for efficient GR inference. Our hypothesis is that a heavy model is still needed to learn from raw token sequences, but the structural regularity of SID generation enables a much cheaper approximation at inference time. We begin by replacing the original Transformer decoder with one of the simplest neural architectures: multilayer perceptrons (MLPs) (Zhou et al., 2022; Zhang et al., 2022). However, naive MLPs do not perform well, since these regularities help only after the user-dependent first tokens are predicted correctly, as also demonstrated by our empirical results in Table 2.

Taking the above considerations together, we propose **SID-MLP**, an MLP-centric distillation framework for efficient GR inference. Given the importance of accurately predicting the first token, we retain a lightweight one-layer attention module to obtain a user-aware summary representation. Cascaded, prefix-conditioned MLP heads then generate the remaining tokens sequentially. Since SID positions have heterogeneous conditional branching and output distributions, we use a separate MLP with independent parameters for each position. To condition each position on the previously generated prefix, the input to each MLP is constructed by concatenating the embeddings of the generated prefix tokens. Experiments on public benchmarks demonstrate that SID-MLP matches Transformer-based teacher model's performance with an average of $8.74\times$ speedup. We further introduce **SID-MLP++**, which extends the distillation to the encoder side and increases the speedup to $10.25\times$ while maintaining competitive performance.

2 Motivation Study

2.1 Preliminary: Generative Recommendation

Task formulation. Let $\mathcal{U}$ and $\mathcal{V}$ denote the sets of users and items, respectively. For a user $u \in \mathcal{U}$, their historical interaction sequence is represented as $\mathbf{X}_u = [v_1, v_2, \ldots, v_n]$, where $v_i \in \mathcal{V}$. Each item v is uniquely mapped to a Semantic ID, which is a tuple of L discrete tokens: $c^{(v)} = (c_1, c_2, \ldots, c_L)$.

For ease of notation, we assume $L = 4$ in our subsequent examples, yielding $c^{(v)} = (c_1, c_2, c_3, c_4)$. In TIGER, the first three tokens are residual quantization codes of item embeddings from a shared codebook $\mathcal{C}$ ($|\mathcal{C}| = 256$), and the fourth is a conflict avoidance token that keeps every full SID unique (Appendix D.4). TIGER serializes $\mathbf{X}_u$ into an encoder token sequence $\mathbf{s}_u$ containing S_u non-padding tokens. A multi-layer T5 encoder-decoder then generates the next item's SID by factorizing the joint probability autoregressively: $p_\theta(c^{(v)} \mid \mathbf{s}_u) = \prod_{j=1}^{L} p_\theta(c_j \mid c_{<j}, \mathbf{s}_u)$.

The autoregressive mechanism and latency explosion. Autoregressive SID generation uses beam search. For SID length L, beam size B, and an N-layer decoder, caching still requires about LBN decoder-block evaluations. Across all digits, it performs $\mathcal{O}(BNL^2)$ prefix self-attention and $\mathcal{O}(LBNS_u)$ cross-attention over encoder states. Because digit c_j depends on prefix $c_{<j}$, SID positions cannot be decoded in parallel, so sequential decoding is the main inference bottleneck.

2.2 Motivation: Rethinking the Decoder's Necessity

Given the inference latency caused by autoregressive decoding, we investigate the structural necessity of the full Transformer decoder through data-level and architecture-level analyses.

SID branching concentrates uncertainty in early digits. We first characterize the inherent difficulty of the SID prediction task. As shown in Figure 1, the average number of valid next-token choices conditioned on a valid prefix drops sharply. On the Instruments dataset, the conditional branching factor drops from $256 \rightarrow \sim 38 \rightarrow 2.2 \rightarrow \sim 1.2$. Appendix B.1 shows the same pattern on Scientific and Games and confirms that high marginal codebook utilization can coexist with low prefix-conditioned branching; Appendix D.3 covers longer, heterogeneous, and multimodal SIDs. The recommendation problem remains history-dependent, but decoder-side uncertainty is highly uneven: most branching is concentrated in the early digits, while later valid digits are strongly constrained by prefixes. For c_3 and c_4, the full N-layer decoder spends computationally expensive forward passes on a nearly sparse valid-continuation set.

Decoder depth is heavily over-parameterized. Given this sharp drop in prediction difficulty, we investigate how much decoder depth is actually utilized. We train a minimal 1-layer student decoder distilled from the frozen TIGER teacher (training recipe in Section 4.1). The teacher generates the first $m \in \{0, 1, 2, 3\}$ digits, and the student predicts the remaining $4 - m$ digits. As shown in Table 1, a 1-layer student perfectly matches the 4-layer teacher when predicting only c_4 (m=3), and loses a mere 1.5% NDCG@10 even when predicting all four digits (m=0). This suggests that much of the decoder depth is not essential for SID-level ranking once the teacher's encoder representation and distillation signal are available. We focus on distillation because the accuracy gap between scratch-training and distillation widens as the architecture simplifies into MLPs (Appendix B.2). Depth reduction alone, however, raises throughput only from 424 to 1,021 samples/s, because every digit still runs the decoder (Appendix E.1).

User-history context must be preserved. Since decoder depth can be safely reduced, we investigate whether we can further decompose the remaining Transformer block to remove redundant internal components. Table 2 evaluates the impact of removing specific attention modules from the m=0 1-layer decoder. Removing prefix self-attention causes a 7.2% NDCG@10 drop, indicating the short SID prefix still needs a position-aware representation. In contrast, removing cross-attention severs the decoder's access to the encoded user history, causing a massive 59.1% drop. Removing both yields a 65.1% loss, proving that a purely context-free MLP decoder is insufficient. Cross-dataset replications confirm this pattern (Appendix B.3).

Target: bridging efficiency and context-awareness. These ablations present a clear structural dilemma. A context-free MLP decoder offers high inference efficiency by eliminating attention overhead, but fails severely because it loses both the fine-grained historical alignment and the position-aware prefix dependency. Conversely, the standard Transformer decoder captures this necessary context but remains impractically slow: even with caching, each active beam still runs decoder-block updates and cross-attention reads at every digit. Any successful decoder replacement must explicitly preserve both the dynamic historical context and the prefix structure while avoiding these repeated computations. *Can we achieve MLP-level inference speeds while retaining the structural benefits of both self- and cross-attention?* This objective motivates our design of SID-MLP in Section 3.

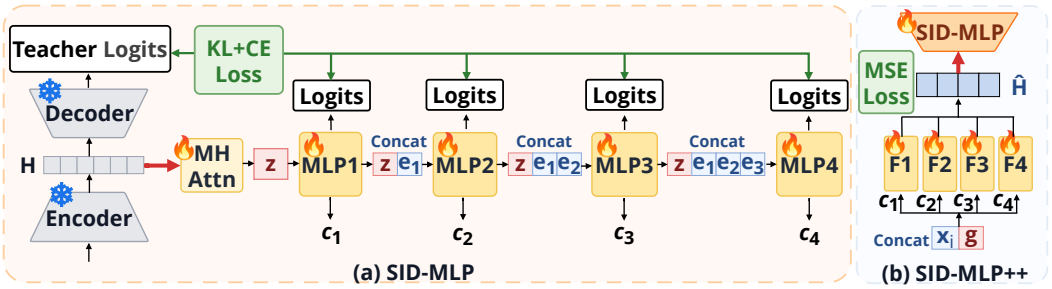


Figure 2: **Architecture.** SID-MLP is shown in yellow (left), and SID-MLP++ in blue (right). MH Attn denotes multi-head attention, and $\mathbf{e}_i = e(c_i)$ is the prefix embedding. Snowflakes mark frozen modules; flames mark trainable ones.

3 SID-MLP: DISTILLING AUTOREGRESSIVE DECODING TO MLPS

3.1 OVERVIEW OF SID-MLP

To resolve the autoregressive latency and heavy attention overhead identified in Section 2, we propose **SID-MLP**. Our core idea is to replace the redundant attention operations in autoregressive generative recommendation with lightweight MLP heads. As illustrated in Figure 2, we train these heads via knowledge distillation from a frozen Transformer teacher. This enables SID-MLP to achieve highly efficient inference while preserving the teacher's strong performance.

In this section, we first detail the extraction of a global user context from the frozen encoder (Section 3.2), then introduce the prefix-conditioned MLP heads (Section 3.3). Finally, we describe how we transfer the teacher's knowledge to MLPs via knowledge distillation (Section 3.4). We conclude with an extension, SID-MLP++, that distills the encoder for further acceleration (Section 3.5).

3.2 ONE-SHOT MULTI-HEAD ATTENTION CONTEXT

The motivation study shows that encoder-conditioned user context is essential. SID-MLP therefore keeps an explicit multi-head attention readout, but computes it once outside the beam loop.

Let $\mathbf{H}_u = E_T(\mathbf{s}_u) \in \mathbb{R}^{S_u \times d_h}$ be the frozen TIGER encoder states for user u. We mean-pool these states, project the result into a query, and use a multi-head attention block to produce a context vector that is reused across all SID digit steps:

$$\mathbf{q} = \mathrm{MeanPool}(\mathbf{H}_u)\, W_q, \quad \tilde{\mathbf{z}} = \mathrm{LN}\big(\mathbf{q} + \mathrm{MHA}(\mathbf{q}, \mathbf{H}_u, \mathbf{H}_u)\big), \quad \mathbf{z} = \mathrm{LN}\big(\tilde{\mathbf{z}} + \mathrm{FFN}(\tilde{\mathbf{z}})\big). \tag{1}$$

where MHA is multi-head attention, LN is layer normalization, and FFN is feed-forward network.

Complexity comparison. Standard decoders (*e.g.*, TIGER) compute cross-attention at every digit step, beam, and decoder layer, incurring $\mathcal{O}(LBNS_u)$ historical alignment. SID-MLP instead extracts the global context once with the aggregated query $\mathbf{q}$. Caching $\mathbf{z}$ removes cross-attention from the beam-search loop and reduces this cost to $\mathcal{O}(S_u)$.

3.3 PREFIX CONCATENATION AND PER-DIGIT MLP HEADS

After $\mathbf{z}$ is cached, each digit prediction only requires the current SID prefix. SID-MLP completely removes standard self-attention. Instead, at step t, each beam simply concatenates the context $\mathbf{z}$ with $t-1$ frozen token embeddings $e(\cdot)$ retrieved from the teacher. Because SID digits have heterogeneous conditional branching and output distributions, each position $t \in \{1, \ldots, L\}$ uses a dedicated 1-hidden-layer MLP head f_t to predict the next token:

$$\mathbf{p}_t = \big[\mathbf{z}\,;\, e(c_1)\,;\ldots;\, e(c_{t-1})\big] \in \mathbb{R}^{d_h + (t-1)d_e}, \quad \boldsymbol{\ell}_t = f_t(\mathbf{p}_t) \in \mathbb{R}^C. \tag{2}$$

where $[\cdot;\cdot]$ is concatenation, $c_1, \ldots, c_{t-1}$ are the available prefix digits, d_e is the frozen embedding dimension, and C (typically 256) is the codebook size. The input dimension of f_t grows with t and reaches at most $d_h + (L-1)d_e$.

Algorithm 1 SID-MLP: Distillation training and batched beam-search inference.

Require: Frozen E_T and $e(\cdot)$; batches $(\mathbf{s}_u, \mathbf{c}^\star, \tilde{\boldsymbol{\ell}}_{1:L})$; beam B; cutoff K

Training (teacher forcing; parallel digit heads)

for each training batch **do**
  $\mathbf{z} \leftarrow \mathrm{GlobalContext}(E_T(\mathbf{s}_u))$
  For all t, form $\mathbf{p}_t$ from the ground-truth prefix and compute $\boldsymbol{\ell}_t = f_t(\mathbf{p}_t)$ in parallel
  Update trainable modules with $\sum_{t=1}^{L} \mathcal{L}_t$ from Equation (3)
end for

Inference (sequential digits; batched beams)

$\mathbf{z} \leftarrow \mathrm{GlobalContext}(E_T(\mathbf{s}_u))$, $\mathcal{B}_0 \leftarrow \{\text{empty prefix}\}$
for $t = 1, \ldots, L$ **do**
  Batch-compute $\boldsymbol{\ell}_t = f_t(\mathbf{p}_t)$ for all prefixes in $\mathcal{B}_{t-1}$
  Apply the valid-prefix mask, expand, and retain top-B candidates as $\mathcal{B}_t$
end for
return top-K items decoded from $\mathcal{B}_L$

Complexity comparison. Standard decoders process prefixes via step-wise self-attention, yielding an $\mathcal{O}(BNL^2)$ cost alongside incremental KV-cache updates. Conversely, SID-MLP scores all active beams via batched dense MLP operations over static concatenated embeddings. Prefix processing thus becomes an update-free, constant-size operation per beam (4.88→0.46 GFLOP per example).

3.4 Distillation Training and Inference

We train SID-MLP via offline knowledge distillation from the frozen teacher. Only the single multi-head attention block and the MLP heads are trainable; the TIGER encoder is frozen. Each MLP head predicts the codebook slice for its digit, and the teacher logits are sliced to the same support. During teacher-forced training, Equation (2) takes the ground-truth SID prefix. Let $\tilde{\boldsymbol{\ell}}_t$ denote the teacher logits and $c_t^\star$ the ground-truth codebook index. We train each digit with a Kullback–Leibler distillation term plus cross-entropy:

$$\mathcal{L}_t = \alpha\tau^2 D_{\mathrm{KL}}\Big(\sigma(\tilde{\boldsymbol{\ell}}_t/\tau) \,\|\, \sigma(\boldsymbol{\ell}_t/\tau)\Big) + (1-\alpha)\mathrm{CE}(\boldsymbol{\ell}_t, c_t^\star) \tag{3}$$

where τ is the distillation temperature, σ is softmax, and $\alpha \in [0,1]$ balances teacher mimicry and task grounding. The total loss is $\mathcal{L} = \sum_{t=1}^{L} \mathcal{L}_t$.

During inference, the context vector $\mathbf{z}$ is computed once. Beam search then evaluates the MLP heads sequentially while batching active prefixes at each digit. A valid-prefix mask built from the fixed item-to-SID mapping removes invalid prefixes before expansion; its lookup time stays flat from 25K to 5M SIDs (Appendix E.3). Algorithm 1 gives the complete training and inference procedure.

3.5 Extension: Encoder Distillation (SID-MLP++)

SID-MLP removes the Transformer decoder stack but retains the teacher encoder. SID-MLP++ extends distillation to this encoder. It replaces the teacher encoder with an MLP encoder and produces encoder hidden states $\hat{\mathbf{H}}_u = G_\phi(\mathbf{s}_u) \in \mathbb{R}^{S_u \times d_h}$; the following SID-MLP structure is unchanged.

SID-MLP++ architecture. The encoder input is the serialized user-history sequence $[\texttt{user}, c_1^{(v_1)}, \ldots, c_L^{(v_1)}, \ldots, \texttt{EOS}]$. We assign four role-specific MLPs (F_1, F_2, F_3, F_4) to process the tokens, where F_4 handles c_4, `user`, and `EOS`, and the others handle their respective c_j digits. Let $a_i \in \{1,2,3,4\}$ denote the role index for token i. The initial state $\mathbf{x}_{u,i}^{(0)}$ combines the frozen token embedding and learnable position embeddings. To replace self-attention, we achieve global context modeling at each layer r by mean-pooling all non-padding token states into a global vector $\mathbf{g}_u^{(r)}$. Each token state $\mathbf{x}_{u,i}^{(r)}$ is then concatenated with this global vector and passed through its role-specific MLP to form a residual update:

$$\mathbf{g}_u^{(r)} = \frac{1}{S_u}\sum_{i=1}^{S_u} \mathbf{x}_{u,i}^{(r)}, \quad \mathbf{x}_{u,i}^{(r+1)} = \mathbf{x}_{u,i}^{(r)} + F_{a_i}\big([\mathbf{x}_{u,i}^{(r)}; \mathbf{g}_u^{(r)}]\big). \tag{4}$$

Two-stage encoder distillation. Because direct logit distillation exhibits optimization instability for the encoder (ablation details in Appendix G.2), we adopt a two-stage process:

1. **Stage 1 (Representation matching):** We pre-train the SID-MLP++ encoder to match the frozen teacher's final encoder states $\mathbf{H}_u$ using a token-level Mean Squared Error (MSE) loss.
2. **Stage 2 (Logit distillation):** We freeze the student encoder, pass its output $\hat{\mathbf{H}}_u$ to the SID-MLP decoder, and train the decoder using the per-digit KL+CE loss (Equation (3)).

At inference, SID-MLP++ replaces the teacher encoder states $\mathbf{H}_u = E_T(\mathbf{s}_u)$ with student states $\hat{\mathbf{H}}_u = G_\phi(\mathbf{s}_u)$. The cached context computation and sequential MLP heads are unchanged.

4 EXPERIMENTS

4.1 SETUP

Datasets and baselines. We instantiate SID-MLP on **TIGER** (Rajput et al., 2023), a T5-based autoregressive generative recommender where items are tokenized into 4-digit semantic IDs. SID-MLP freezes the TIGER encoder and replaces decoder-side generation with prefix-conditioned MLP heads. We evaluate on three categories from the latest Amazon Reviews 2023 dataset (Hou et al., 2026b): Musical Instruments **(Instruments)**, Industrial & Scientific **(Scientific)**, and Video Games **(Games)**. Full dataset statistics are in Appendix C.1. We organize baselines into three distinct groups.

i *Teacher:* The TIGER model serves as our autoregressive upper bound, using a four-layer T5 to generate SIDs token-by-token via beam search. We report both the standard no-cache teacher TIGER and TIGER-kv, which enables the decoder key–value cache during generation. We further evaluate our method on the LC-Rec teacher, with experimental details in Appendix D.2.

ii *LLM-style accelerators ported to TIGER:* These baselines add draft/verification modules or jointly fine-tune TIGER, whereas SID-MLP keeps TIGER frozen. (1) **AtSpeed** (Lin et al., 2025b), originally instantiated on LC-Rec (Zheng et al., 2024), uses a compact draft model for speculative decoding and teacher verification; we port it with a compact T5 draft and evaluate strict verification (-S) and relaxed sampling (-R). (2) **EARN** (Yang et al., 2025a) uses boundary register tokens to retain early-layer history information and prunes other states; we port it to the TIGER encoder with joint fine-tuning. (3) **NEZHA** (Wang et al., 2026b) uses SID placeholders and an autoregressive draft head with recurrent state updates for self-drafting; we feed the placeholders to the TIGER encoder and use the draft head instead of the T5 decoder, with joint fine-tuning. Table 4 compares these models.

iii *State space decoders:* These baselines test whether linear-recurrent sequence models can replace the TIGER decoder. They keep the TIGER encoder frozen and feed its hidden states, a bridge token, and prefix embeddings into a causal SSM stack. The stack outputs digit logits at the added positions and uses the same KL+CE objective as SID-MLP. We evaluate (4) **GatedDeltaNet (GDN)** (Yang et al., 2025b) and (5) **Mamba2** (Dao & Gu, 2024).

Evaluation & implementation. We report Recall@K and NDCG@K ($K \in \{5, 10\}$) over valid 4-digit SIDs. For the main SID-MLP row, performance metrics are averaged over random seeds $\{42, 43, 44\}$; Section C.3 reports mean±std and a competitive performance against the fixed TIGER-kv teacher. All methods use matched TIGER checkpoints. Throughput is end-to-end samples/s on the test split with batch size 32 and beam size B=50; speedup is relative to TIGER-kv. Hyperparameters are in Appendix C.2; baseline adaptation and diagnostic details are in Appendices F.1, F.2, and F.3; hardware profiling details are in Appendix E.1.

4.2 MAIN RESULTS

Table 3 compares SID-MLP against all baselines on the three categories. We highlight four findings:

(1) SID-MLP matches teacher performance without target-model verification. SID-MLP matches or exceeds TIGER NDCG@10 on all datasets while keeping the TIGER encoder frozen and requiring no target-model verification. This supports the motivation study: for short hierarchical SIDs, per-digit MLP heads can recover the teacher decoder's ranking behavior when the student preserves the

Table 3: **Main results.** Ranking metrics are mean values; Section C.3 reports SID-MLP seed stability over $\{42, 43, 44\}$. Tput is the averaged throughput across three datasets, and Spd. is the speedup relative to TIGER-kv. **Bold** = best, underline = second-best.

			Instruments				Scientific				Games			
Method	Tput	Spd.	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
TIGER	320	0.75×	0.0386	0.0606	0.0252	0.0323	0.0295	0.0457	0.0191	0.0243	**0.0612**	0.0951	**0.0403**	**0.0512**
TIGER-kv	424	1.00×	0.0386	0.0606	0.0252	0.0323	0.0295	0.0457	0.0191	0.0243	**0.0612**	0.0951	**0.0403**	**0.0512**
GDN	510	1.20×	0.0385	0.0598	0.0253	0.0321	0.0278	0.0439	0.0183	0.0235	0.0585	0.0914	0.0380	0.0486
Mamba2	517	1.22×	0.0388	0.0605	0.0256	0.0326	**0.0300**	0.0468	**0.0193**	0.0247	0.0589	0.0931	0.0386	0.0496
AtSpeed-R	94	0.22×	0.0246	0.0375	0.0163	0.0204	0.0216	0.0320	0.0144	0.0178	0.0436	0.0637	0.0296	0.0361
AtSpeed-S	286	0.68×	0.0386	0.0606	0.0252	0.0323	0.0295	0.0457	0.0191	0.0243	0.0612	0.0951	0.0403	0.0512
EARN	822	1.94×	0.0383	0.0596	0.0250	0.0319	0.0288	0.0454	0.0186	0.0239	0.0584	0.0921	0.0377	0.0486
NEZHA	3,082	7.27×	0.0371	0.0567	0.0245	0.0308	0.0254	0.0402	0.0164	0.0212	0.0560	0.0865	0.0368	0.0467
SID-MLP	3,706	8.74×	**0.0396**	**0.0620**	**0.0259**	**0.0332**	0.0297	**0.0472**	**0.0193**	**0.0250**	0.0610	**0.0953**	0.0402	**0.0512**
SID-MLP++	**4,347**	**10.25×**	0.0395	0.0612	0.0257	0.0327	0.0295	0.0459	0.0192	0.0244	0.0578	0.0916	0.0378	0.0486

Table 4: **Comparison of generative recommendation acceleration paradigms.** *Plug-and-Play*: acts as a drop-in accelerator for already deployed models without requiring modifications to the original tokenizers, or fine-tuning of the base model. *Verify-Free*: requires no target-model verification passes. *Attn-Free Decoding*: isolates attention computation from the autoregressive loop, eliminating repeated self-attention and cross-attention during generation. *Update-Free Beam*: executes beam expansion as batched matrix multiplications without KV-cache or recurrent-state updates.

Paradigm	Plug-and-Play	Verify-Free	Attn-Free Decoding	Update-Free Beam
Speculative (AtSpeed)	✓	×	×	×
State Pruning (EARN)	×	✓	×	×
Self-Drafting (NEZHA)	×	✓	✓	×
SID-MLP (Ours)	✓	✓	✓	✓

encoded user context and ordered SID prefix. Appendix G.1 provides hyperparameter analyses; Appendix D extends this result to three more datasets, 1M–13M teachers, and varied SID structures.

(2) The speedup comes from removing repeated decoder work. SID-MLP reaches 3,706 samples/s (an 8.74× speedup over TIGER-kv), with a 95.7% peak-memory reduction shown in Appendix E.1. After one encoder pass and a multi-head attention readout, beam expansion relies solely on batched MLP projections. It eliminates decoder blocks, KV-cache updates, repeated attention reads, and draft verification. Distilling the encoder (SID-MLP++) further pushes the speedup to 10.25× with a minimal accuracy tradeoff. SID-MLP's offline cost is offset after ∼1M examples (Appendix E.2).

(3) Direct ports expose different bottlenecks. AtSpeed-S and AtSpeed-R are both slower than TIGER-kv (0.68× and 0.22×) because even our smallest single layer Transformer draft is too large relative to the 4.59M TIGER teacher, unlike the 68M-vs-7B LC-Rec setting (Lin et al., 2025b). EARN reaches only 1.94× because it only compresses the encoder states, leaving the autoregressive decoder blocks and per-step computation intact. Its quality also drops because TIGER's bidirectional encoder lacks the head/tail attention-sink pattern that motivates EARN's registers. NEZHA reaches 7.27× by replacing the decoder with a draft-head path, but it still loses up to 13% relative NDCG@10. SID-MLP is 1.20× faster than NEZHA and preserves teacher quality. This highlights the architectural mismatch of directly porting LLM accelerators to GR.

(4) SSMs are limited by recurrent memory updates overhead. Mamba2 (1.22×) and GDN (1.20×) are only slightly faster than TIGER-kv and much slower than SID-MLP despite their linear sequence complexity. In four-digit beam search, each step must update, fork, and gather recurrent states across active beams, so state movement dominates. SID-MLP is update-free after the context readout and scores all active prefixes with batched dense projections.

4.3 Ablation Study

We ablate SID-MLP along three design axes. Table 5 reports NDCG@10 and Recall@10 across Instruments, Scientific and Games; Δ columns show relative NDCG@10 change vs SID-MLP.

Table 5: **Ablation on three datasets.** Recall@10, NDCG@10, and relative NDCG@10 change.

	Instruments			Scientific			Games		
Variant	R@10	N@10	Δ	R@10	N@10	Δ	R@10	N@10	Δ
SID-MLP	**0.0620**	**0.0332**	—	**0.0472**	**0.0250**	—	**0.0953**	**0.0512**	—
G1: Head Architecture									
w/o prefix conditioning	0.0175	0.0092	−72.0%	0.0160	0.0086	−65.5%	0.0352	0.0182	−64.3%
summed prefix embeddings	0.0603	0.0324	−2.4%	0.0457	0.0243	−2.8%	0.0950	0.0503	−1.8%
shared MLP head	0.0564	0.0301	−9.3%	0.0423	0.0225	−10.0%	0.0908	0.0481	−6.1%
cascaded hidden state	0.0613	0.0328	−1.2%	0.0457	0.0245	−2.0%	0.0945	0.0503	−1.8%
G2: Context Module									
w/o multi-head attention	0.0597	0.0316	−4.8%	0.0442	0.0234	−6.4%	0.0929	0.0496	−3.1%
per-digit context readout	0.0618	0.0330	−0.4%	0.0467	0.0249	−0.3%	0.0959	0.0515	+0.7%
w/o context FFN	0.0616	0.0330	−0.7%	0.0468	0.0248	−0.8%	0.0953	0.0507	−1.0%
G3: Distillation Design									
w/o KL	0.0602	0.0323	−2.7%	0.0449	0.0240	−4.0%	0.0900	0.0478	−6.6%
full-vocab logits	0.0619	0.0330	−0.6%	0.0470	0.0250	−0.0%	0.0956	0.0510	−0.5%
w/o teacher embeddings	0.0616	0.0330	−0.6%	0.0467	0.0248	−0.8%	0.0955	0.0511	−0.2%

G1: Head architecture. The variant without prefix conditioning predicts all four digits from the same context vector in parallel, without feeding prefix embeddings into later heads. It loses 64–72% NDCG@10, showing that the prediction must condition on the ordered SID prefix. Summing prefix embeddings instead of concatenating them loses 1.8–2.8%, so the heads benefit from seeing the prefix as an ordered tuple. On the summed-prefix input, sharing one classifier across all digits drops by 6.1–10.0%, suggesting that different digits require dedicated heads. The cascaded-state variant passes the newest prefix token and the hidden activation to the next head, instead of giving every head the fixed context and the full prefix. It does not help, so direct access to the cached context and explicit prefix is better for four-digit SIDs.

G2: Context module. The variant without multi-head attention replaces the one-shot attention readout with a linear projection of the mean-pooled encoder states. It loses 3.1–6.4% NDCG@10, confirming that SID-MLP still needs token-level history alignment. A per-digit prefix-conditioned context readout changes NDCG@10 by -0.4% to $+0.7\%$, so repeated cross-attention at each step provides no consistent gain, confirming our observation that attention becomes redundant in SID-based GR. Removing the FFN after multi-head attention costs at most 1.0%, indicating that the attention readout carries most of the useful context.

G3: Distillation design. Removing KL distillation and training only on hard SID labels loses 2.7–6.6%, showing that the teacher's dense soft labels help in the sparse valid-SID space. Predicting the full 1027-token vocabulary gives no benefit over the 256-way digit codebook. Replacing the frozen teacher embeddings with randomly initialized trainable embeddings changes NDCG@10 by at most 0.8%, so the teacher embeddings are useful but not the main source of SID-MLP's accuracy.

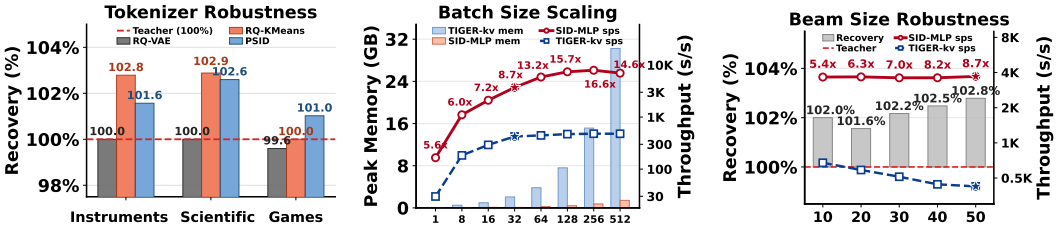
4.4 Performance of SID-MLP under Different Settings

We evaluate SID-MLP under three settings: SID tokenizer, batch size, and beam size (Figure 3). A temporal item-shift diagnostic is reported in Appendix D.5.

Tokenizer sensitivity. We distill SID-MLP from teachers trained with popular tokenizers: RQ-KMeans (Deng et al., 2025; Ju et al., 2025), RQ-VAE (Lee et al., 2022; Rajput et al., 2023), and PSID (Zhang et al., 2025). As shown in Figure 3(a), SID-MLP retains 99.6% to 102.9% of teacher NDCG@10 across all tokenizer–dataset pairs, with no systematic loss.

Batch size. Figure 3(b) plots throughput and peak GPU memory as batch size scales from 1 to 512. TIGER-kv saturates early because each digit step still runs the Transformer decoder over all active beams. SID-MLP instead turns the post-context computation into batched dense projections. Its throughput scales until batch size reaches 256, where speedup peaks at $16.6\times$. At batch size 512, SID-MLP keeps peak memory below 1.5 GB, compared with up to 30 GB for TIGER-kv.

Beam size. Figure 3(c) shows that TIGER's throughput drops by 37% from beam size 10 to 50 because it repeats autoregressive decoding. After one context readout, SID-MLP only widens the batched MLP projections, adding no sequential steps. Its throughput stays stable and NDCG@10 exceeds the teacher at every beam size.



(a) Tokenization strategies. (b) Batch size. (c) Beam size.

Figure 3: **SID-MLP robustness across settings.** (a) NDCG@10 recovery is the ratio between SID-MLP and its teacher across tokenizers. (b) Peak GPU memory (bars) and throughput (lines) as batch size changes. (c) NDCG@10 recovery (bars) and throughput (lines) as beam size changes.

5 Related Work

Generative recommendation with identifiers. Generative retrieval stores items as discrete identifiers and retrieves them by generating these identifiers autoregressively, as in DSI (Tay et al., 2022). TIGER (Rajput et al., 2023) brings this idea to recommendation with RQ-VAE semantic IDs and a T5 encoder–decoder, and later work extends the backbone to language models (Geng et al., 2022; Zheng et al., 2024) or scales GR to industrial systems (Zhai et al., 2024; Deng et al., 2025). Most SID studies improve the identifiers with collaborative, context-aware, or multimodal codes (Wang et al., 2024b; Hou et al., 2025b; Liu et al., 2024a), or with tokenizers trained jointly with the recommender (Wang et al., 2024a; Liu et al., 2025a), while other studies analyze how SID models scale and generalize (Liu et al., 2026a; Ding et al., 2026a). These methods change how IDs are built or how the recommender is trained (see Section H). SID-MLP complements them: given a trained TIGER-style tokenizer and encoder, it replaces repeated Transformer decoding in constrained beam search.

Efficient generation for recommender IDs. Existing GR accelerators reduce latency in different ways. AtSpeed (Lin et al., 2025b) and SpecGR (Ding et al., 2026b) draft candidate IDs and still use the target model to verify them. NEZHA (Wang et al., 2026b) adds self-drafting heads and a hash-based validity check, while EARN (Yang et al., 2025a) compresses the LLM context with register tokens. Other methods change the generation procedure with parallel or non-autoregressive decoding (Hou et al., 2025a; Ren et al., 2024) or discrete diffusion (Liu et al., 2026b; Shi et al., 2025), and tree-based retrievers learn a new item index (Zhu et al., 2018; Gao et al., 2020). SID-MLP instead keeps the existing 4-digit SID space and replaces online decoder calls with prefix-conditioned MLP heads, without target-model verification.

Distillation and lightweight sequence models. Knowledge distillation transfers teacher behavior to smaller students (Hinton et al., 2015), and recommendation and retrieval mainly use it to transfer quality (Cui et al., 2024; Li et al., 2024a). GLNN (Zhang et al., 2022) and LLP (Guo et al., 2023) show that an MLP student can absorb a teacher's structural inductive bias while removing inference-time dependencies. LLM accelerators such as speculative decoding, KV-cache serving, and draft heads (Leviathan et al., 2023; Kwon et al., 2023; Cai et al., 2024) target open-ended text and usually keep a verification step. State-space models (Gu & Dao, 2024; Dao & Gu, 2024) provide efficient recurrent backbones but do not use the low-branching SID prefix structure.

6 Conclusion and Future Works

In this paper, we propose SID-MLP to eliminate the inference latency of autoregressive generative recommendation. Instead of relying on a heavy Transformer decoder, our key idea is to distill the generative process into cascaded (sequentially evaluated, prefix-conditioned, position-specific), lightweight MLP heads. By computing the multi-head attention context exactly once, SID-MLP reduces beam expansion to efficient batched dense MLP operations without requiring repeated state/cache update. Extensive experiments demonstrate that SID-MLP matches the teacher's performance, achieving an $8.74\times$ throughput speedup and a 95.7% peak-memory reduction. We also introduce SID-MLP++, an encoder-distilled variant that achieves a $10.25\times$ speedup. In future work, we plan to apply SID-MLP to industrial-scale systems with millions of items to further investigate search-space dynamics and encoder-side efficiency.

AI USE STATEMENT

Generative AI tools were used only to assist with language polishing of the manuscript. The authors reviewed all AI-assisted edits and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Our code is available at the anonymous repository linked in the abstract, together with launch scripts for feature extraction, training, and inference. Appendix C.2 gives the training recipe and tuning ranges, Appendix C.3 the random seeds and non-inferiority test, and Appendix E.1 the hardware and throughput protocol. Dataset statistics and splits are in Appendix C.1, the additional datasets and teacher sizes in Appendix D.1, the LC-Rec setup in Appendix D.2, and the baseline adaptations in Appendices F.1–F.3.

REFERENCES

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. 2024.

Yimeng Bai, Chang Liu, Yang Zhang, Dingxian Wang, Frank Yang, Andrew Rabinovich, Wenge Rong, and Fuli Feng. Bi-level optimization for generative recommendation: Bridging tokenization and generation. *arXiv preprint arXiv:2510.21242*, 2025.

Vladimir Baikalov, Iskander Bagautdinov, and Sergey Muravyov. Mitigating collaborative semantic id staleness in generative retrieval. *arXiv preprint arXiv:2604.13273*, 2026.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Runjin Chen, Mingxuan Ju, Ngoc Bui, Dimosthenis Antypas, Stanley Cai, Xiaopeng Wu, Leonardo Neves, Zhangyang Wang, Neil Shah, and Tong Zhao. Enhancing item tokenization for generative recommendation through self-improvement. *arXiv preprint arXiv:2412.17171*, 2024.

Yu Cui, Feng Liu, Pengbo Wang, Bohao Wang, Heng Tang, Yi Wan, Jun Wang, and Jiawei Chen. Distillation matters: Empowering sequential recommenders to match the performance of large language models. In *18th ACM Conference on Recommender Systems*, pp. 507–517. ACM, October 2024.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*, 2025.

Yijie Ding, Zitian Guo, Jiacheng Li, Letian Peng, Shuai Shao, Wei Shao, Xiaoqiang Luo, Luke Simon, Jingbo Shang, Julian McAuley, et al. How well does generative recommendation generalize? *arXiv preprint arXiv:2603.19809*, 2026a.

Yijie Ding, Jiacheng Li, Julian McAuley, and Yupeng Hou. Inductive generative recommendation via retrieval-based speculation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 14675–14683, 2026b.

Junchen Fu, Xuri Ge, Alexandros Karatzoglou, Ioannis Arapakis, Suzan Verberne, Joemon M Jose, and Zhaochun Ren. Differentiable semantic id for generative recommendation. *arXiv preprint arXiv:2601.19711*, 2026a.

Kairui Fu, Tao Zhang, Shuwen Xiao, Ziyang Wang, Xinming Zhang, Chenchi Zhang, Yuliang Yan, Junjun Zheng, Xiangheng Kong, Shengyu Zhang, et al. 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, 2026b.

Weihao Gao, Xiangjun Fan, Chong Wang, Jiankai Sun, Kai Jia, Wenzhi Xiao, Ruofan Ding, Xingyan Bin, Hui Yang, and Xiaobing Liu. Deep retrieval: learning a retrievable structure for large-scale recommendations. *arXiv preprint arXiv:2007.07203*, 2020.

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

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. 2022.

Zhichun Guo, William Shiao, Shichang Zhang, Yozen Liu, Nitesh V Chawla, Neil Shah, and Tong Zhao. Linkless link prediction via relational distillation. In *International conference on machine learning*, pp. 12012–12033. PMLR, 2023.

Ruining He, Lukasz Heldt, Lichan Hong, Raghunandan Keshavan, Shifan Mao, Nikhil Mehta, Zhengyang Su, Alicia Tsai, Yueqi Wang, Shao-Chuan Wang, et al. Plum: Adapting pre-trained language models for industrial-scale generative recommendations. In *Proceedings of the ACM Web Conference 2026*, pp. 8093–8104, 2026.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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 V. 2*, pp. 956–966, 2025a.

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. *arXiv preprint arXiv:2502.13581*, 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*, 2026a.

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking llms as semantic encoders. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3251–3265, 2026b.

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, 2023.

Jie Jiang, Xinxun Zhang, Enming Zhang, Yuling Xiong, Jun Zhang, Jingwen Wang, Huan Yu, Yuxiang Wang, Hao Wang, Xiao Yan, et al. End-to-end semantic id generation for generative advertisement recommendation. *arXiv preprint arXiv:2602.10445*, 2026.

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.

Clark Mingxuan Ju, Tong Zhao, Leonardo Neves, Liam Collins, Bhuvesh Kumar, Jiwen Ren, Lili Zhang, Wenfeng Zhuo, Wen Zhang, Xiao Bai, et al. Semantic ids for recommender systems at snapchat: Use cases, technical challenges, and design choices. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 4694–4699, 2026.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Sein Kim, Hongseok Kang, Kibum Kim, Jiwan Kim, Donghyun Kim, Minchul Yang, Kwangjin Oh, Julian McAuley, and Chanyoung Park. Lost in sequence: Do large language models understand sequential recommendation? In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 1160–1171, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

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. 11513–11522. IEEE, 2022.

Geon Lee, Bhuvesh Kumar, Mingxuan Ju, Tong Zhao, Kijung Shin, Neil Shah, and Liam Collins. Sequential data augmentation for generative recommendation. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, pp. 303–312, 2026.

Simon Lepage, Jeremie Mary, and David Picard. Closing the performance gap in generative recommenders with collaborative tokenization and efficient modeling. *arXiv preprint arXiv:2508.14910*, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

Jialei Li, Yang Zhang, Yimeng Bai, Shuai Zhu, Ziqi Xue, Xiaoyan Zhao, Dingxian Wang, Frank Yang, Andrew Rabinovich, and Xiangnan He. Unigrec: Unified generative recommendation with soft identifiers for end-to-end optimization. *arXiv preprint arXiv:2601.17438*, 2026a.

Yongqi Li, Zhen Zhang, Wenjie Wang, Liqiang Nie, Wenjie Li, and Tat-Seng Chua. Distillation enhanced generative retrieval. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11119–11129, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems*, 38:136737–136756, 2026b.

Xinyu Lin, Haihan Shi, Wenjie Wang, Fuli Feng, Qifan Wang, See-Kiong Ng, and Tat-Seng Chua. Order-agnostic identifier for large language model-based generative recommendation. In *Proceedings of the 48th international ACM SIGIR conference on research and development in information retrieval*, pp. 1923–1933, 2025a.

Xinyu Lin, Chaoqun Yang, Wenjie Wang, Yongqi Li, Cunxiao Du, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Efficient inference for large language model-based generative recommendation. In *International Conference on Learning Representations*, volume 2025, pp. 91672–91697, 2025b.

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, 2025a.

Enze Liu, Bowen Zheng, Wayne Xin Zhao, and Ji-Rong Wen. Bridging textual-collaborative gap through semantic codes for sequential recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 1788–1798, 2025b.

Han Liu, Yinwei Wei, Xuemeng Song, Weili Guan, Yuan-Fang Li, and Liqiang Nie. Mmgrec: Multimodal generative recommendation with transformer model. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2024a.

Jingzhe Liu, Liam Collins, Jiliang Tang, Tong Zhao, Neil Shah, and Mingxuan Ju. Understanding generative recommendation with semantic ids from a model-scaling view. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 3234–3245, 2026a.

Zhao Liu, Yichen Zhu, Yiqing Yang, Xiao Lv, Guoping Tang, Rui Huang, Qiang Luo, Ruiming Tang, and Guorui Zhou. Diffgrm: Diffusion-based generative recommendation model. In *Proceedings of the ACM Web Conference 2026*, pp. 5853–5864, 2026b.

Zihan Liu, Yupeng Hou, and Julian McAuley. Multi-behavior generative recommendation. In *Proceedings of the 33rd ACM international conference on information and knowledge management*, pp. 1575–1585, 2024b.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

Aleksandr V Petrov and Craig Macdonald. Generative sequential recommendation with gptrec. *arXiv preprint arXiv:2306.11114*, 2023.

Haohao Qu, Shanru Lin, Yujuan Ding, Yiqi Wang, and Wenqi Fan. Diffusion generative recommendation with continuous tokens. In *Proceedings of the ACM Web Conference 2026*, pp. 7259–7270, 2026.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

Yuxin Ren, Qiya Yang, Yichun Wu, Wei Xu, Yalong Wang, and Zhiqiang Zhang. Non-autoregressive generative models for reranking recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5625–5634, 2024.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. 2015.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

Kulin Shah, Bhuvesh Kumar, Neil Shah, and Liam Collins. Masked diffusion for generative recommendation. 2025.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. 2019.

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.

Zhenmei Shi, Yifei Ming, Xuan-Phi Nguyen, Yingyu Liang, and Shafiq Joty. Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13839–13857, 2026.

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.

Zhengyang Su, Isay Katsman, Yueqi Wang, Ruining He, Lukasz Heldt, Raghunandan Keshavan, Shao-Chuan Wang, Xinyang Yi, Mingyan Gao, Onkar Dalal, et al. Vectorizing the trie: Efficient constrained decoding for llm-based generative retrieval on accelerators. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 8009–8020, 2026.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. *Advances in neural information processing systems*, 35:21831–21843, 2022.

Huanjie Wang, Xinchen Luo, Honghui Bao, Zhang Zixing, Lejian Ren, Yunfan Wu, Hongwei Zhang, Liwei Guan, and Guang Chen. Pit: A dynamic personalized item tokenizer for end-to-end generative recommendation. *arXiv preprint arXiv:2602.08530*, 2026a.

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, 2024a.

Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, et al. 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, 2024b.

Yejing Wang, Shengyu Zhou, Jinyu Lu, Ziwei Liu, Langming Liu, Maolin Wang, Wenlin Zhang, Feng Li, Wenbo Su, Pengjie Wang, et al. Nezha: A zero-sacrifice and hyperspeed decoding architecture for generative recommendations. In *Proceedings of the ACM Web Conference 2026*, pp. 8073–8082, 2026b.

Tianxin Wei, Xuying Ning, Xuxing Chen, Ruizhong Qiu, Yupeng Hou, Yan Xie, Shuang Yang, Zhigang Hua, and Jingrui He. Cofirec: Coarse-to-fine tokenization for generative recommendation. *arXiv preprint arXiv:2511.22707*, 2025.

Chaoqun Yang, Xinyu Lin, Wenjie Wang, Yongqi Li, Teng Sun, Xianjing Han, and Tat-Seng Chua. EARN: efficient inference acceleration for llm-based generative recommendation by register tokens. *CoRR*, 2025a.

Guang Yang, Yu Zhou, Xiangyu Zhang, Wei Cheng, Ke Liu, Xiang Chen, Terry Yue Zhuo, and Taolue Chen. Less is more: Towards green code large language models via unified structural pruning. *Information Processing & Management*, 63(4):104580, 2026.

Liu Yang, Fabian Paischer, Kaveh Hassani, Jiacheng Li, Shuai Shao, Zhang Gabriel Li, Yun He, Xue Feng, Nima Noorshams, Sem Park, et al. Unifying generative and dense retrieval for sequential recommendation. *arXiv preprint arXiv:2411.18814*, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. 2025b.

Jianyang Zhai, Zi-Feng Mai, Chang-Dong Wang, Feidiao Yang, Xiawu Zheng, Hui Li, and Yonghong Tian. Multimodal quantitative language for generative recommendation. In *International Conference on Learning Representations*, volume 2025, pp. 74424–74441, 2025a.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

Penglong Zhai, Yifang Yuan, Fanyi Di, Jie Li, Yue Liu, Chen Li, Jie Huang, Sicong Wang, Yao Xu, and Xin Li. A simple contrastive framework of item tokenization for generative recommendation. *arXiv preprint arXiv:2506.16683*, 2025b.

Fuwei Zhang, Xiaoyu Liu, Dongbo Xi, Jishen Yin, Huan Chen, Peng Yan, Fuzhen Zhuang, and Zhao Zhang. Multi-aspect cross-modal quantization for generative recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 16271–16279, 2026a.

Qian Zhang, Lech Szymanski, Haibo Zhang, and Jeremiah D Deng. How reliable are semantic-id tokenizer comparisons in generative recommendation? *arXiv preprint arXiv:2605.25330*, 2026b.

Ruohan Zhang, Jiacheng Li, Julian McAuley, and Yupeng Hou. Purely semantic indexing for llm-based generative recommendation and retrieval. *arXiv preprint arXiv:2509.16446*, 2025.

Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. 2022.

Zhen Zhang, Jujia Zhao, Xinyu Ma, Xin Xin, Maarten de Rijke, and Zhaochun Ren. Cold-starts in generative recommendation: A reproducibility study. *arXiv preprint arXiv:2603.29845*, 2026c.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, et al. A survey of large language models. *Frontiers of Computer Science*, 20(12):2012627, 2026.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1435–1448. IEEE, 2024.

Qiyong Zhong, Jiajie Su, Yunshan Ma, Julian McAuley, and Yupeng Hou. Pctx: Tokenizing personalized context for generative recommendation. *arXiv preprint arXiv:2510.21276*, 2025.

Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. Filter-enhanced mlp is all you need for sequential recommendation. In *Proceedings of the ACM web conference 2022*, pp. 2388–2399, 2022.

Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1079–1088, 2018.

Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. Joint optimization of tree-based index and deep model for recommender systems. *Advances in Neural Information Processing Systems*, 32, 2019.

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, 2024.

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.

A NOTATIONS

Table 6 summarizes the notations used throughout the paper.

Table 6: Notations and explanations.

Notation	Explanation
$\mathcal{U}, \mathcal{V}$	User and item sets; u and v index users and items
$\mathbf{X}_u = [v_1, \ldots, v_n]$	Historical item sequence of user u
$\mathbf{s}_u, S_u$	Serialized encoder tokens for $\mathbf{X}_u$ and its non-padding length
$c^{(v)} = (c_1, \ldots, c_L)$	Semantic ID of item v; c_t is digit t and $c_{<t}$ its prefix
$\mathcal{C}, C, L$	Per-digit codebook, codebook size, and SID length; TIGER uses $C = 256$ and $L = 4$
$p_\theta(\cdot)$	Teacher autoregressive distribution over SID digits
B, K, m	Beam size, recommendation cutoff, and teacher-generated prefix length in m-mode
$E_T(\cdot), G_\phi(\cdot)$	Frozen TIGER encoder and SID-MLP++ student encoder; both map serialized user-history tokens to encoder states
$\mathbf{H}_u, \hat{\mathbf{H}}_u$	Teacher states and SID-MLP++ encoder states
d_h, d_e	Encoder hidden-state size and frozen token-embedding size
$e(\cdot)$	Frozen SID-token embedding lookup
$W_q, \mathbf{q}, \tilde{\mathbf{z}}, \mathbf{z}$	Query projection/query and cached one-shot context vectors
$f_t(\cdot)$	Digit-t MLP head mapping the prefix-conditioned input to sliced student logits
$\mathbf{p}_t, \boldsymbol{\ell}_t$	Input vector and student logits for digit t
$\tilde{\boldsymbol{\ell}}_t, c_t^\star$	Teacher logits and ground-truth codebook index for digit t
$\alpha, \tau, \mathcal{L}_t, \mathcal{L}$	Distillation weight, temperature, per-digit loss, and total loss
$\mathcal{B}_t$	Active beam set after digit t
N, i, r	Teacher decoder layer count, encoder token position, and SID-MLP++ layer index
$\mathbf{x}_{u,i}^{(r)}, \mathbf{g}_u^{(r)}$	SID-MLP++ token state and pooled global vector at encoder layer r
$a_i, F_{a_i}(\cdot)$	Digit-role id for token position i and the corresponding role-specific MLP

B MOTIVATION DETAILS

B.1 PER-DATASET CONDITIONAL BRANCHING FACTOR STATISTICS

Table 7 reports the per-dataset conditional branching factors behind Figure 1. For each valid prefix $c_{<t}$, we count how many next-token values c_t can still lead to a valid item SID, and then average over valid prefixes. This statistic is different from marginal codebook utilization, which counts how many code values appear at a position without conditioning on previous digits. In each evaluated catalog, the first digit has the full 256-way choice, the second digit has about 38 valid continuations on average, the third digit has about 2.2, and the fourth digit is close to singleton. Later digits still have occasional multi-choice prefixes, but their average valid-continuation set is already near singleton within each dataset. The teacher's own predictions show the same concentration: given the ground-truth prefix, its predictive entropy falls from about 6 bits at the first digit to at most 0.35 bits at the last (Table 8), so most of its uncertainty is resolved within the first two digits.

This sparsity is expected in over-complete fixed-length SID spaces. Let $\mathcal{P}_t$ denote the set of valid prefixes of length t in an item catalog, with $\mathcal{P}_0 = \{\emptyset\}$. The average conditional branching at position t is $b_t = |\mathcal{P}_t|/|\mathcal{P}_{t-1}|$. For unique length-$L$ SIDs, $\prod_{t=1}^{L} b_t = |\mathcal{P}_L| = |\mathcal{V}|$. Therefore, when the full capacity C^L is much larger than the catalog size $|\mathcal{V}|$, broadly spreading items across early prefixes necessarily leaves low average branching for later positions. Our Amazon Reviews 2023 categories contain about 25K items, while TIGER's 4-digit 256-way SID space has $256^4 \approx 4.3$B possible combinations. Later digits nevertheless remain user-dependent: within a shared semantic prefix, the model still ranks items by the current user's history (Section D.4).

Table 7: **Average conditional branching factor by digit.** Number of valid continuations at digit t given a valid prefix $c_{<t}$, computed separately within each dataset's item vocabulary. No cross-dataset average is used.

Dataset	# items	Avg. $\lvert c_1 \rvert$	Avg. $\lvert c_2 \rvert c_1 \rvert$	Avg. $\lvert c_3 \rvert c_{1:2} \rvert$	Avg. $\lvert c_4 \rvert c_{1:3} \rvert$
Instruments	24,587	256	37.7	2.19	1.17
Scientific	25,848	256	37.7	2.24	1.19
Games	25,612	256	38.1	2.19	1.20

Table 8: **Teacher predictive entropy by digit.** Entropy (bits) of the TIGER teacher's 256-way distribution at each digit given the ground-truth prefix, averaged over validation examples. The maximum is 8 bits.

Dataset	c_1	c_2	c_3	c_4
Instruments	6.30	4.13	1.27	0.27
Scientific	6.26	4.02	1.35	0.35
Games	6.03	3.26	1.01	0.28

Table 9: **Marginal utilization versus conditional branching across tokenizers.** Marginal code usage and entropy are computed independently at each SID position, while branching counts valid next-token choices conditioned on a valid prefix. PSID-OPQ uses all 256 codes at every position with high marginal entropy, yet still exhibits low conditional branching.

Tokenizer	Dataset	Marginal codes	Norm. entropy	Avg. valid continuations
RQ-KMeans	Instruments	$256/255/256/24$	$0.96/0.86/0.97/0.10$	$256 \to 37.71 \to 2.19 \to 1.17$
RQ-KMeans	Scientific	$256/255/256/30$	$0.97/0.83/0.97/0.12$	$256 \to 37.71 \to 2.24 \to 1.19$
RQ-KMeans	Games	$256/256/256/25$	$0.96/0.88/0.97/0.12$	$256 \to 38.08 \to 2.19 \to 1.20$
RQ-VAE	Instruments	$92/256/256/8$	$0.79/0.98/0.98/0.06$	$92 \to 102.45 \to 2.40 \to 1.09$
RQ-VAE	Scientific	$58/256/256/17$	$0.72/0.98/0.97/0.10$	$58 \to 123.72 \to 3.10 \to 1.16$
RQ-VAE	Games	$148/256/256/9$	$0.88/0.98/0.98/0.06$	$148 \to 77.11 \to 2.07 \to 1.08$
PSID-OPQ	Instruments	$256/256/256$	$0.97/0.98/0.97$	$256 \to 43.72 \to 2.20$
PSID-OPQ	Scientific	$256/256/256$	$0.98/0.98/0.98$	$256 \to 52.15 \to 1.94$
PSID-OPQ	Games	$256/256/256$	$0.97/0.96/0.97$	$256 \to 47.50 \to 2.11$

Table 10: **Distillation versus scratch-training on Instruments.** Values are NDCG@10. The relative drop is measured against the distillation row.

Decoder Family	Distill	Scratch	Drop vs. Distill
1-layer Transformer decoder	0.0318	0.0313	-1.6%
MLP decoder (SID-MLP)	0.0332	0.0323	-2.7%

B.2 Distillation Versus Scratch Training

We compare distillation against scratch-training to justify our focus on the distilled setting. Table 10 evaluates a 1-layer Transformer decoder (m=0) and our SID-MLP decoder on Instruments dataset. For scratch-training, we train exclusively on hard SID labels by removing the KL term.

Without distillation, the 1-layer Transformer loses 1.6% NDCG@10. However, for the simpler MLP decoder, this gap widens to 2.7%. This confirms that as the architecture becomes lighter, it relies more heavily on the teacher's soft labels to maintain accuracy. Therefore, in this work, we focus on distillation to achieve MLP-level efficiency without severe performance degradation.

Table 11: **Cross-dataset attention ablation on the m=0 1-layer decoder.** SA and CA denote self-attention and cross-attention, respectively. Values are N@10; relative drops are computed against the first row within each dataset.

SA	CA	Instruments	Scientific	Games
✓	✓	0.0318	0.0236	0.0480
×	✓	0.0295 (−7.2%)	0.0205 (−13.1%)	0.0449 (−6.5%)
✓	×	0.0130 (−59.1%)	0.0088 (−62.7%)	0.0123 (−74.4%)
×	×	0.0111 (−65.1%)	0.0072 (−69.5%)	0.0095 (−80.2%)

Table 12: **Dataset statistics.** Amazon Reviews 2023 rows are computed from our 5-core preprocessed splits (McAuley-Lab/amazon-reviews-2023, `last_out` split), with four training targets per user sequence (rolling next-item); evaluation uses leave-one-out. The last three rows are the additional datasets in Section D.1.

Dataset	# Users	# Items	# Interactions	Sparsity	Avg. Seq. Len
Musical Instruments	57,439	24,587	511,836	99.96%	8.91
Industrial & Scientific	50,985	25,848	412,947	99.97%	8.10
Video Games	94,762	25,612	814,586	99.97%	8.60
Steam	47,761	10,971	599,620	99.89%	12.55
Yelp	30,431	20,033	316,354	99.95%	10.40
Office Products	223,308	77,551	1,800,878	99.99%	8.06

B.3 Cross-Dataset Attention Ablation

Table 11 checks whether the attention-ablation pattern in Table 2 is specific to Instruments dataset. We repeat the m=0 1-layer decoder ablation on all three datasets. Removing self-attention causes a modest loss, while removing cross-attention causes a large loss on every dataset. This supports the design choice in SID-MLP: retain an encoder-conditioned context readout and remove repeated decoder attention.

C Experimental Setup and Reproducibility

C.1 Dataset Statistics

Table 12 summarizes the Amazon Reviews 2023 (Hou et al., 2026b) datasets used in the main experiments, together with the three additional datasets in Section D.1. User sequences are chronologically ordered, truncated to 20 items, and split via the leave-last-out protocol.

C.2 Implementation Details and Hyperparameters

This subsection details the training configurations deferred from Section 4.1.

SID-MLP training. We optimize the trainable modules using AdamW with a cosine learning-rate schedule and select the checkpoint with the best validation NDCG@10. Offline student training takes approximately one hour on average for each dataset (Instruments, Scientific, and Games). This offline cost is separate from online serving and is excluded from throughput measurements. Throughput profiling uses the trained checkpoints under the protocol in Appendix E.1.

Hyperparameters. Across all datasets, most values are fixed: learning rate $5\text{e}{-5}$, dropout 0.2, distillation temperature τ=1, and 4 attention heads. The practical tuning space reduces to $\alpha \in \{0.7, 0.8\}$, weight decay $\in \{1\text{e}{-4}, 1\text{e}{-3}\}$, attention inner dimension $\in \{384, 512\}$, and head hidden size $\in \{512, 768, 1536\}$. The wider one-variable sweeps in Appendix G.1 serve as a one-time robustness check.

SID-MLP++ training. For the encoder distillation (Section 3.5), both Stage 1 (MSE pre-training) and Stage 2 (logit distillation) use the same AdamW optimizer as SID-MLP.

Table 13: **Random-seed stability and non-inferiority test on NDCG@10.** Values are computed over seeds $\{42, 43, 44\}$. The symbol † indicates passing the non-inferiority test against TIGER-kv with a 1% relative margin ($p < 0.05$).

Dataset	N@10 (42)	N@10 (43)	N@10 (44)	Mean±std	TIGER-kv	Diff.	p
Instruments	0.0333	0.0332	0.0331	0.0332±0.0001†	0.0323	+0.0009	0.001
Scientific	0.0252	0.0250	0.0248	0.0250±0.0002†	0.0243	+0.0007	0.007
Games	0.0512	0.0512	0.0511	0.0512±0.0001†	0.0512	+0.0000	0.006

Hyperparameters. SID-MLP++ adds one extra capacity choice for the encoder, but it does not multiply the base search space: Stage 2 reuses the per-dataset base SID-MLP configuration, and Stage 1 adds only learning rate 1e−3, depth $\in \{4, 6\}$, and FFN dimension $\in \{1024, 2048, 8192\}$ (8192 is merely the sweep's upper bound for the encoder, never a requirement of base SID-MLP).

C.3 Random Seeds and Teacher-Matching Significance

This appendix provides the multi-seed stability analysis for SID-MLP. Our core claim is quality preservation: SID-MLP replaces the autoregressive decoder without degrading the accuracy of the original TIGER-kv checkpoint. Table 13 reports the seed-level NDCG@10 values and a formal non-inferiority test.

Seed protocol. We train SID-MLP across three random seeds ($\{42, 43, 44\}$). The seeds control the model initialization, data loader order, and minibatch stochasticity. TIGER-kv is evaluated using the single fixed teacher checkpoint from the main paper, serving as a deterministic reference.

Non-inferiority test. Since our objective is to match rather than outperform the teacher, we employ a one-sided non-inferiority t-test. Let x_i denote the SID-MLP NDCG@10 for seed i, b denote the deterministic TIGER-kv baseline, and $\epsilon = 0.01b$ define a 1% relative non-inferiority margin. The hypotheses are:

$$H_0 : \mathbb{E}[x_i - b] \leq -\epsilon, \qquad H_1 : \mathbb{E}[x_i - b] > -\epsilon.$$

The test statistic is $t = (\bar{x} - b + \epsilon)/(s/\sqrt{n})$, with $n = 3$ and degrees of freedom $\mathrm{df} = 2$, where s is the sample standard deviation. A p-value < 0.05 rejects the null hypothesis, formally confirming that SID-MLP does not incur a statistically significant performance drop beyond the 1% margin.

As shown in Table 13, all three datasets strongly satisfy the $p < 0.05$ criterion. This multi-seed evaluation statistically grounds our claim: SID-MLP successfully preserves the teacher's ranking quality while delivering the substantial speedups reported in Table 3.

D Generalization of SID-MLP

The main experiments use three Amazon Reviews 2023 categories, a ∼5M-parameter TIGER teacher, and the default RQ-KMeans 3×256 SID. To test whether SID-MLP's ability to match its teacher depends on these choices, this section changes one of them at a time and compares SID-MLP with its matched teacher, mainly through recovery, the ratio of SID-MLP's metric to the teacher's. We vary the data domain and catalog scale together with the TIGER teacher size (Section D.1), the teacher architecture with a 7B LC-Rec teacher (Section D.2), and the SID structure, including tokenizer, length, per-position width, and modality (Section D.3). We then examine two properties of the SID itself and of the data: prefix conflict, where a high share of items share a semantic prefix (Section D.4), and temporal item shift, where test targets are newly introduced items (Section D.5).

D.1 Datasets and Teacher Sizes

Diverse and larger datasets. To separately examine cross-domain generalization and larger public-dataset scale, we add Steam (gaming), Yelp (local services), and the larger Amazon Office Products; SID-MLP preserves matched-teacher performance on all three (Table 14). Office Products has 223K users, 77.6K items, and 1.80M interactions, about 2.4×, 3.0×, and 2.2× the largest corresponding values among our main datasets (Table 12). Across all three new datasets SID-MLP recovers 99.3%–

Table 14: **Additional datasets.** Metrics are percentages. "Rec." is the recovery rate, the student metric divided by the corresponding teacher metric; it is computed from unrounded values.

			R@10			N@10		
Dataset	# Items	# Inter.	Teacher	SID-MLP	Rec.	Teacher	SID-MLP	Rec.
Steam	10,971	599,620	19.44	19.51	100.4	15.51	15.40	99.3
Yelp	20,033	316,354	4.86	5.08	104.5	2.57	2.65	103.3
Office Products	77,551	1,800,878	4.44	4.47	100.7	2.54	2.54	100.3

Table 15: **TIGER teachers of varying sizes.** N@10 values are percentages. Recovery is SID-MLP N@10 divided by teacher N@10.

Dataset	Teacher parameters	Teacher N@10	SID-MLP N@10	Recovery (%)
Instruments	1M	3.07	3.16	102.9
	5M	3.23	3.32	102.8
	13M	3.16	3.31	104.5
Scientific	1M	2.23	2.27	101.9
	5M	2.43	2.50	102.9
	13M	2.39	2.48	103.9
Games	1M	4.25	4.54	106.8
	5M	5.12	5.12	100.0
	13M	5.06	5.07	100.3

Table 16: **Dataset statistics (Amazon Reviews 2018).** Users and items with fewer than five interactions are filtered.

Dataset	# Users	# Items	# Interactions	Sparsity	Avg. Seq. Len
Musical Instruments	24,773	9,923	206,153	99.92%	8.32
Arts Crafts and Sewing	45,142	20,957	390,832	99.96%	8.66
Video Games	50,547	16,860	452,989	99.95%	8.96

103.3% of teacher N@10, so its ability to match the teacher holds beyond the Amazon categories and small scales of the main experiments.

TIGER teachers of varying sizes. On the three Amazon-2023 categories, we add controlled experiments with ∼1M/5M/13M TIGER (Rajput et al., 2023) teachers. SID-MLP preserves matched-teacher N@10 in all nine dataset–size combinations (Table 15), so the decoder replacement does not depend on the default 5M teacher. Since teacher quality does not increase monotonically with size on these benchmarks (consistent with (Liu et al., 2026a)), we focus on recovery.

Results for different tokenizers, SID lengths, codebook widths, and heterogeneous SIDs are reported in Section D.3, showing the conclusion does not depend on a single TIGER tokenizer either. Together with Section D.2, our verified teachers span two representative GR architectures, encoder–decoder T5 (TIGER, 1M–13M) and decoder-only LLaMA 7B (LC-Rec (Zheng et al., 2024)), over nine datasets covering e-commerce, gaming, and local services.

D.2 APPLICATION TO LC-REC

This appendix checks whether SID-MLP also applies to an LLM-based generative recommender. LC-Rec (Zheng et al., 2024) releases public checkpoints on Amazon Reviews 2018 (Ni et al., 2019), so we follow its protocol on Musical Instruments, Arts Crafts and Sewing **(Arts)**, and Video Games. The dataset statistics are shown in Table 16.

Setup. Two design philosophies exist for accelerating LLM-based generative recommenders: (i) jointly fine-tune the LLM backbone with a draft head (e.g. NEZHA (Wang et al., 2026b)), which breaks plug-and-play deployment; or (ii) keep the LLM frozen and train a plug-and-play student,

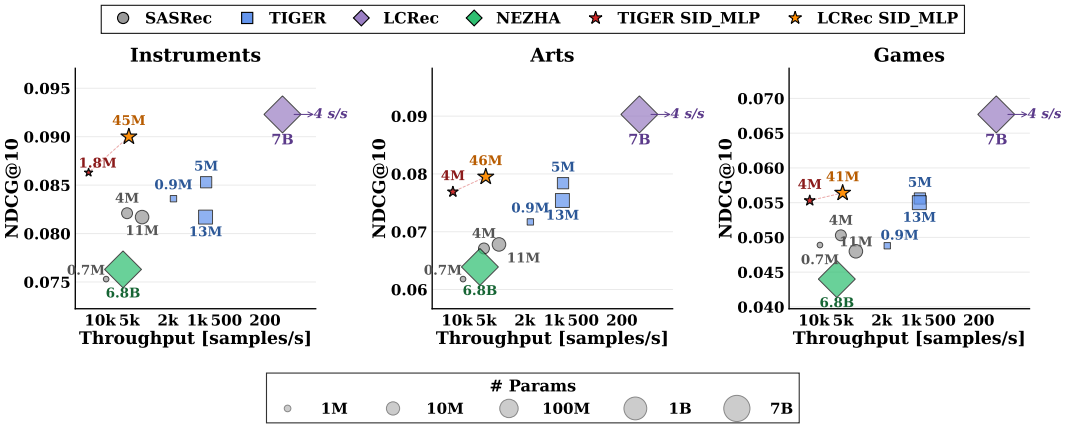


Figure 4: **Cross-scale Pareto on Amazon Reviews 2018.** NDCG@10 vs throughput (samples/s, log scale) across Instruments, Arts, and Games. All throughputs are *decode-only*. Baselines span SASRec (Kang & McAuley, 2018) (∼1M/5M/13M), TIGER (∼1M/5M/13M), and LC-Rec 7B (LLaMA-2, off-axis at ∼4 samples/s); students are TIGER SID-MLP and LC-Rec SID-MLP, distilled from the 5M and 7B teachers respectively. Marker size encodes the trainable parameter count for students and the total parameter count for all other models.

preserving compatibility with existing checkpoints. SID-MLP follows (ii). We distill SID-MLP from **LC-Rec**, a 7B LLaMA-2-based generative recommender. Since LC-Rec only releases public checkpoints on Amazon Reviews 2018, we adopt its protocol; retraining TIGER at ∼1M/5M/13M on the same categories gives a single cross-scale Pareto view (Figure 4).

Adaptation to decoder-only LLaMA. LC-Rec has no separate encoder, requiring us to redesign the multi-head attention sources in Equation (1). Recent studies on LLMRec identify a layer-wise attention sparsity inversion: early layers retain dense, informative patterns that encode broad user history, while later layers become highly sparse and redundant (Yang et al., 2025a; Shi et al., 2026). Building on this insight, we extract the key–value (KV) states from all history-item tokens at an early layer of the frozen prefill (e.g., the 8th of 32). For the query, we extract the last-layer (32nd) hidden state of the final prompt token, as it is directly optimized for next-token prediction and closely aligns with the immediate recommendation intent. The per-digit MLP heads, trie-constrained beam search, and KL+CE objective remain identical to the T5-based SID-MLP.

SID-MLP training on LC-Rec. The training protocol follows the TIGER SID-MLP setup in Appendix C.2. To accommodate the LLaMA backbone, we simply project the 4096-dimensional Q and KV states down to d_h=768 via learnable linear layers. For hyperparameters, we sweep the number of stacked attention readout layers over $\{1, 2, 3\}$, FFN dimension $\in \{512, 1024, 2048, 3072\}$, and head hidden size $\in \{512, 1024, 2048, 3072\}$.

Findings. SID-MLP sits on the Pareto frontier at both teacher scales. At the TIGER scale, it matches the teacher while delivering an $8.74\times$ throughput speedup. At the 7B LC-Rec scale, SID-MLP recovers 83.3–97.5% of the teacher's accuracy using only 44.1M trainable parameters (0.6% of the teacher). It accelerates end-to-end throughput to 38.04 samples/s ($17.9\times$ speedup) and decode-only throughput to 5,194 samples/s ($1{,}262.8\times$ speedup). The remaining accuracy gap to the frozen backbone varies by category. In contrast, NEZHA requires 6.8B trainable parameters ($153.2\times$ more than SID-MLP) yet only recovers 65.0–82.7% of the LC-Rec teacher. As detailed in Appendix F.3, our NEZHA reproductions consistently exhibit larger quality drops than SID-MLP across both architectures.

d_h width scaling. If the remaining gap were mainly caused by an overly narrow $4096 \to 768$ projection, raising d_h to 1024/2048 should consistently improve accuracy. We ran the scale-up on all three datasets (Table 17). Widening yields no consistent cross-dataset improvement: some points above 768 help on one dataset (e.g., Games 1024/2048, Arts 1024) but not others, and all values stay within ∼2.6% relative, so widening does not close the teacher gap. Even at d_h=4096, the gap remains. Under the original d_h=768, SID-MLP distills a 7B LC-Rec teacher into 44.1M

Table 17: d_h **width scaling on LC-Rec.** N@10 values are percentages under the Amazon-2018 LC-Rec protocol.

d_h	Instruments	Games	Arts
512	8.93	5.70	7.92
768	9.00	5.55	8.01
1024	8.91	5.65	8.02
2048	8.87	5.68	7.95
3072	8.88	5.57	7.85
4096	8.86	5.58	7.94

Table 18: **Tokenizers and codebook widths on Amazon-2023 Instruments.** N@10 values are percentages.

		N@10		
SID setting	Valid continuations by position	Teacher	SID-MLP	Rec. (%)
RQ-VAE: 3×256	$92 \rightarrow 102.45 \rightarrow 2.40 \rightarrow 1.09$	3.20	3.20	100.0
PSID-OPQ: 3×256	$256 \rightarrow 43.72 \rightarrow 2.20$	3.19	3.24	101.5
RQ-KMeans: 3×512	$512 \rightarrow 28.24 \rightarrow 1.56 \rightarrow 1.09$	3.13	3.20	102.3
RQ-KMeans: 3×256 (default)	$256 \rightarrow 37.71 \rightarrow 2.19 \rightarrow 1.17$	3.23	3.32	102.8
RQ-KMeans: 3×128	$128 \rightarrow 44.12 \rightarrow 3.36 \rightarrow 1.30$	3.28	3.33	101.6

trainable parameters (only 0.6% of the 7B backbone), and the order-of-magnitude capacity gap remains. SID-MLP recovers 83.3–97.5% of teacher N@10 and lies on the accuracy–throughput Pareto frontier among all evaluated methods (Figure 4). Taken together, these results indicate that the remaining gap marks the boundary of this aggressive speed–quality trade-off.

D.3 TOKENIZERS, SID LENGTH, AND HETEROGENEOUS AND MULTIMODAL SIDS

In this subsection, Recovery is SID-MLP N@10 divided by teacher N@10 and is computed from unrounded values. Each $n \times W$ lists semantic levels only. For every TIGER (Rajput et al., 2023)-style setting, we append a final 256-way conflict-avoidance token, whereas PSID-OPQ (Zhang et al., 2025) produces unique three-token SIDs without this extra token. Unless noted otherwise, all settings use Amazon-2023 Instruments and the ∼5M-parameter TIGER architecture, and each SID mapping has its own matched teacher.

Tokenizers and prefix structure. Figure 3(a) reports 99.6%–102.9% teacher N@10 recovery across RQ-VAE (Lee et al., 2022), RQ-KMeans, and PSID-OPQ on the three main datasets. On Amazon-2023 Instruments, we further vary RQ-KMeans codebook width; branching reports the average number of valid continuations at each position given a valid prefix (Table 18). These settings span clearly different early- and late-position branching (e.g., RQ-VAE's second position averages 102 valid continuations), and SID-MLP matches its teacher under all of them. Once early prefixes have separated most items, later valid branching naturally becomes small; we therefore report the measured branching profiles and do not assume identical sparsity for every tokenizer.

SID length. We use a matched teacher–student protocol. The 2×256–3×256 comparison isolates semantic depth at a fixed width of 256; the 6×64 setting follows TIGER's Appendix E (Rajput et al., 2023) as a literature-grounded longer-SID stress test (Table 19). Each TIGER-style SID appends one conflict-avoidance token, so decoding steps equal the number of semantic levels plus one. Recent studies also adopt three semantic tokens and find that longer SIDs usually reduce quality (Liu et al., 2026a); our paper therefore focuses on short SIDs and controls SID length when comparing tokenizers and codebook widths. Even the long 6-level (7-step) SID still recovers 99.8% of teacher N@10, so SID-MLP's ability to match the teacher is not limited to short SIDs.

Heterogeneous SIDs. SID-MLP requires neither equal-width positions nor text-only SIDs. We add two heterogeneous settings, $[64, 128, 256]$ and $[256, 128, 64]$, each with a final 256-way conflict-avoidance token appended (Table 20 lists semantic positions only). These are a catalog-scaled

Table 19: **SID length on Amazon-2023 Instruments.** N@10 values are percentages.

SID setting	Semantic levels	Decoding steps	Teacher N@10	SID-MLP N@10	Recovery (%)
2×256	2	3	3.29	3.37	102.4
3×256 (default)	3	4	3.23	3.32	102.8
6×64	6	7	3.24	3.23	99.8

Table 20: **Heterogeneous SIDs on Amazon-2023 Instruments.** N@10 values are percentages.

Semantic widths	Teacher N@10	SID-MLP N@10	Recovery (%)
$[256, 256, 256]$	3.23	3.32	102.8
$[64, 128, 256]$	3.35	3.36	100.3
$[256, 128, 64]$	3.35	3.40	101.4

Table 21: **Multimodal SIDs.** N@10 values are percentages.

		N@10		
SID setting	Dataset / protocol	Teacher	SID-MLP	Rec. (%)
Unified multimodal SID	Amazon-2023 Instruments	3.17	3.17	100.1
MACRec dual-branch	Amazon-2018 Instruments (MACRec)	10.25	10.27	100.2

Table 22: **Prefix conflict on Amazon-2023 Instruments.** Full structures and branching profiles are in Section D.3.

SID setting (semantic)	Prefix-conflict items (%)	N@10 recovery (%)
PSID-OPQ: 3×256	0.0	101.5
RQ-VAE: 3×256	14.1	100.0
RQ-KMeans: 2×256	76.7	102.4
RQ-KMeans: 3×256	23.6	102.8
RQ-KMeans: 3×128	35.5	101.6
RQ-KMeans: 3×512	14.1	102.3
FORGE: $[64, 128, 256]$	49.0	100.3
FORGE: $[256, 128, 64]$	29.0	101.4

structural adaptation of FORGE (Fu et al., 2026b). Both coarse-to-fine and fine-to-coarse layouts match their teachers, so SID-MLP does not rely on equal-width positions.

Multimodal SIDs. For multimodal IDs, we test early-fusing both modalities into one RQ-KMeans SID. Using MACRec's (Zhang et al., 2026a) released embeddings and cross-modal quantization, we also distill its text/image SID branches separately (4 RQ codes each, no separate conflict-avoidance token) and apply its official item-score fusion. The datasets and protocols differ, so recovery, not absolute N@10, is comparable (Table 21). Across both multimodal settings SID-MLP recovers $\sim$100% of teacher N@10, so the method is robust to heterogeneous and multimodal SIDs and does not depend on text-only, uniform-width IDs.

D.4 SID Collision and Prefix Conflict

ZCR (Zhang et al., 2026b) concerns full-SID collisions: SID-level metrics can overestimate item-level performance when multiple items share the same full SID. TIGER (Rajput et al., 2023) guarantees full-SID uniqueness via the final conflict-avoidance token, and SID-MLP retains it; PSID-OPQ (Zhang et al., 2025) needs no such token and its full-SID collision rate is 0. Our SIDs therefore have no complete-SID collisions. Table 22 (all on Amazon-2023 Instruments) instead reports prefix-conflict items, the share of items in non-singleton semantic-prefix groups once the conflict-avoidance token is removed, which is prefix conflict, not full-SID collision.

Table 23: **Replacing the current user's history.** Random-history values are averaged over 3 seeds.

Check	User history	TIGER	SID-MLP
(a) End-to-end N@10 (%)	Real	3.292	3.370
(a) End-to-end N@10 (%)	Random other user	0.543	0.569
(b) Within-prefix pairwise (%)	Real	55.382	54.217
(b) Within-prefix pairwise (%)	Random other user	50.012	50.010

Table 24: **Final-position objective in the high-conflict setting.** Metrics are percentages.

Final-position objective	N@10	R@10	Within-prefix pairwise
$0.7\,\mathrm{KL} + 0.3\,\mathrm{CE}$	3.370	6.341	54.217
CE only	3.303	6.203	53.190

These settings cover 0%–76.7% prefix conflict, and SID-MLP preserves matched-teacher N@10 in every one. This also addresses the concern that the final conflict-avoidance token, which is not learned by the semantic tokenizer, may inject noise: across the full 0%–76.7% prefix-conflict range, SID-MLP consistently preserves that teacher's performance. So even if prefix conflict or the non-semantic final token affects the teacher's absolute recommendation quality, within our experimental scope it does not weaken SID-MLP's ability to recover that teacher.

Diagnostics in the hardest setting. All diagnostics below are run in the hardest setting for this concern: Instruments, RQ-KMeans $2 \times 256 + 1$ conflict-avoidance token, where 76.7% of items share a semantic prefix with another item.

(1) Replacing the user's history collapses performance, so the model relies on the current user rather than on token statistics. We replace the encoder input with the history of a random other user and keep everything else fixed, including the target and the valid-SID tree, in two settings. (a) End to end: beam search runs from the empty prefix as usual. (b) Final position: the correct semantic prefix is given, and for each pair of items A and B that share it, we compare the scores of their two final tokens on test cases whose target is A or B, weighting the two directions equally within each pair. Any user-independent rule, such as always preferring the more popular item or the smaller index, gives every user the same answer and therefore scores exactly 50%. The test contains 26,708 such pairs.

With another user's history, end-to-end N@10 falls by about 83% and the within-prefix accuracy returns to 50%, the level of a user-independent rule (Table 23). All accuracy above chance therefore comes from the current user's history, which no frequency ordering can provide. Such input-perturbation checks are established practice for testing whether a recommender uses the user's sequence (Kim et al., 2025). TIGER itself reaches 55.382%, so the within-prefix signal is about 5 points, and SID-MLP keeps most of it (a 1.165-point difference).

(2) High conflict makes the semantic prefix harder to predict, not easier. Top-1 accuracy at the first two semantic positions is 8.8%/29.8% in the default setting and drops to 7.883%/20.987% here (SID-MLP: 7.888%/21.332%). At the first position no prefix is available, and a context-free baseline emitting the most frequent training token gets only 2.521%; the models are 3.1× higher. SID-MLP matches its teacher exactly where prediction is hardest. The contraction of valid continuations after a correct prefix also holds here ($256 \to 41.41 \to 2.32$). Fewer candidates, however, do not make the final position a fixed lookup. The final position's high aggregate accuracy in Figure 1 pools in cases with only one valid candidate; restricted to cases with more than one, accuracy is 53.126%/52.820% for TIGER/SID-MLP.

(3) In SID-MLP training, the KL loss at the final position transfers a user-specific signal, not a sharper frequency prior. We change only the final collision position's objective from $0.7\,\mathrm{KL} + 0.3\,\mathrm{CE}$ to plain CE, with everything else fixed (Table 24). The within-prefix accuracy drops by 1.026 points (CIs exclude zero). Under random histories both versions fall back to 50% and the gap vanishes (50.010% vs. 50.064%), so the information added by the KL term is user-dependent.

Table 25: **Temporal item-shift diagnostic.** Test NDCG@10 on targets whose first interaction timestamp is after the global 80% or 90% timestamp. Δ is SID-MLP minus TIGER.

	Temporal-80				Temporal-90			
Dataset	# Test	TIGER	SID-MLP	Δ	# Test	TIGER	SID-MLP	Δ
Instruments	4,272	0.0076	0.0076	+0.0000	2,051	0.0079	0.0078	-0.0001
Scientific	5,573	0.0152	0.0158	+0.0006	3,122	0.0134	0.0149	+0.0015
Games	21,212	0.0193	0.0194	+0.0001	10,434	0.0190	0.0191	+0.0001
Average	–	0.0140	0.0143	+0.0003	–	0.0134	0.0139	+0.0005

(4) Hits come from selecting the right item within a prefix group rather than filling the Top-10 list with that group. Among SID-MLP's 3,642 Recall@10 hits, 87.04% contain no second item from the target's prefix group (TIGER: 86.39%), and its lists carry slightly more distinct prefixes (9.347 vs. 9.325). The model therefore has to identify the right item within the group.

(5) The conclusion holds on a SID with no collision position at all. All of the above uses an appended conflict-avoidance token. By contrast, all three positions of PSID-OPQ's 3×256 SID are semantic codes: every complete SID is already unique, so there is no arbitrary position where frequency could substitute for user modelling. SID-MLP still recovers 101.5% of teacher N@10 there. Matching the teacher here requires reproducing its user-conditioned distribution at every position, which is what the KL objective transfers.

Finally, on collision handling: ZCR (Zhang et al., 2026b) studies how to build collision-free SIDs, while SID-MLP takes the mapping as given, and our results cover both a mapping with an appended conflict-avoidance token and one that is collision-free by construction.

D.5 Temporal Item-Shift Diagnostic

We evaluate SID-MLP's robustness to temporal item shift, where test targets represent recently introduced items. Following the timestamp-based evaluation protocols of SpecGR (Ding et al., 2026b) and ColdGenRec (Zhang et al., 2026c), we compute each item's first interaction timestamp in the chronological log. A test case is assigned to the Temporal-80 or Temporal-90 subset if its target item's first interaction occurs after the global 80% or 90% timestamp, respectively. The 90% cutoff strictly matches ColdGenRec's item cold-start definition, while the 80% cutoff provides a broader diagnostic subset. Since this diagnostic uses the existing TIGER checkpoints, SID tokenizer, and candidate sets from the main experiments, it assesses temporal shift robustness rather than strict zero-shot item insertion.

Table 25 shows that SID-MLP seamlessly tracks the frozen TIGER teacher on temporally shifted targets. Across all three datasets, SID-MLP performs identically or slightly better, yielding average NDCG@10 improvements of $+0.0003$ and $+0.0005$ on the 80% and 90% splits, respectively. This confirms that replacing the autoregressive Transformer decoder with prefix-conditioned MLP heads introduces no measurable degradation on newer items.

E Efficiency and Deployment Cost

Table 3 reports online throughput. This section adds the costs that a deployment also pays: the hardware profile behind the throughput numbers (Section E.1), the one-time offline distillation compute compared with the online compute it saves (Section E.2), and the memory and lookup time of the valid-prefix index as the catalog grows to millions of items (Section E.3).

E.1 Hardware Profiling

Reproducibility setup. All experiments run on a single NVIDIA RTX A6000 GPU (48GB) using PyTorch 2.1, CUDA 11.8, and Python 3.10.

Profiling protocol. Table 26 details the runtime metrics behind the main results in Table 3. The timed inference region starts after model initialization and data loading, encompassing GPU forward passes,

Table 26: **Hardware profile.** Throughput, speedup, and amortized time are averaged across the three datasets; NDCG@10 and peak GPU memory are measured on Instruments. TIGER-kv serves as the speedup and memory-ratio baseline. **Bold** = best and underline = second-best per metric.

Method	NDCG@10	samples/s	Speedup	Amort. ms/sample	Peak Mem. (GB)	Mem./TIGER-kv
TIGER	0.0323	320	0.75×	3.13	0.84	40.6%
TIGER-kv	0.0323	424	1.00×	2.36	2.07	100.0%
NEZHA	0.0308	3,082	7.27×	0.32	**0.07**	**3.4%**
EARN	0.0319	822	1.94×	1.22	0.21	10.1%
Mamba2	0.0326	517	1.22×	1.93	2.55	123.2%
GDN	0.0321	510	1.20×	1.96	9.67	467.1%
AtSpeed-S	0.0323	286.3	0.68×	3.49	0.87	42.0%
AtSpeed-R	0.0204	94.0	0.22×	10.64	0.93	44.9%
SID-MLP	**0.0332**	3,706	8.74×	0.27	0.09	4.3%
SID-MLP++	0.0327	**4,347**	**10.25**×	**0.23**	0.43	20.8%

Table 27: **Decoder depth on Instruments.** Throughput follows the protocol of Table 3 (batch size 32, beam size 50).

Method	N@10	Throughput (samples/s)
TIGER-kv	0.0323	424
1-layer distilled decoder	0.0318	1,021
SID-MLP	**0.0332**	**3,706**

beam expansion, valid-SID masking, final item lookup, and metric computation. Offline teacher feature extraction for SID-MLP training is excluded from this online serving profile. We use batch size 32 and beam size B=50 across all tested methods. All runs use FP32 with TF32 disabled under `torch.inference_mode`. Each run starts with a two-batch warm-up, CUDA synchronization brackets the timed region, and we report the median of three full test-split runs. Throughput and speedup are averaged across the three datasets, while NDCG@10 and peak memory are reported from the Musical Instruments run (memory footprints remain stable across datasets).

Metric definitions. *NDCG@10* measures ranking quality. *samples/s* denotes end-to-end throughput, and *Amort. ms/sample* is the amortized time per request at batch size 32 ($1000/\text{samples/s}$). *Speedup* is relative to TIGER-kv. *Peak Mem.* captures the maximum allocated GPU memory during inference, and *Mem./TIGER-kv* reports this footprint as a percentage of the TIGER-kv baseline.

The profile reveals a practical Pareto frontier. SID-MLP achieves the best NDCG@10 while remaining 8.74× faster than TIGER-kv end-to-end and reducing peak memory from 2.07GB to a mere 0.09GB (a 95.7% reduction). When isolating the generation phase, SID-MLP yields a **16.25× decoder-only speedup** (13,130 vs. 808 samples/s for TIGER-kv). SID-MLP++ pushes the end-to-end throughput further to 4,347 samples/s (10.25× speedup) with a minor accuracy trade-off. While NEZHA has a slightly smaller absolute memory footprint (0.07GB), SID-MLP is substantially faster and more accurate. In contrast, SSM baselines (Mamba2 and GDN) incur much larger memory footprints and lower throughput, confirming that maintaining linear recurrent states is a poor latency-memory trade-off for short SID beam search.

Decoder depth. A distilled 1-layer Transformer decoder nearly matches the teacher (Table 1) and reaches 1,021 samples/s under the same protocol, a 2.4× speedup over TIGER-kv. SID-MLP is a further 3.6× faster with higher NDCG@10, so most of the gain comes from replacing the decoder with MLP heads rather than from reducing its depth (Table 27).

E.2 Offline Distillation Cost versus Serving Savings

Following lifecycle-compute accounting, Sardana et al. (2023) combine training and expected inference FLOPs, while Yang et al. (2026) divide one-time FLOPs by per-inference savings. After the shared TIGER teacher, SID-MLP adds 4.48 PFLOP once and reduces online compute from 4.88 to 0.46 GFLOP/example, saving 4.42 GFLOP/example. Thus, 4.48 PFLOP $\div$ 4.42 GFLOP $\approx$ 1.01M

Table 28: **One-time offline compute versus online savings.** N@10 and logical FLOPs use Amazon-2023 Instruments; speedup follows Table 3. Offline is the extra offline compute, excluding the shared teacher. Saved is the online compute saved per example, i.e., TIGER-kv's 4.877 GFLOP minus each method's full inference compute. Volume is the compute-equivalent volume, Offline divided by Saved.

Method	N@10 (%)	Speedup ↑	Offline (PFLOP) ↓	Saved (GFLOP/ex.) ↑	Volume (M ex.) ↓
SID-MLP	**3.32**	8.74×	**4.48**	4.42	**1.01**
EARN	3.19	1.94×	6.78	3.44	1.97
NEZHA	3.08	7.27×	17.58	4.50	3.91
Mamba2	3.26	1.22×	49.97	3.67	13.64
GDN	3.21	1.20×	66.17	3.37	19.66

Table 29: **Valid-prefix index from 25K to 5M unique SIDs.** Times are per batch (32 users × beam 50); the last-level mask lookup is a single batched lookup over all $32 \times 50 = 1{,}600$ active prefixes, including all mask lookups.

Unique SIDs	GPU index memory	After lossless bit-packing	Last-level mask lookup	Full beam search
25K	22.7 MiB	17.4 MiB	0.222 ms	3.650 ms
100K	42.8 MiB	21.4 MiB	0.217 ms	3.653 ms
1M	285.0 MiB	69.5 MiB	0.215 ms	3.608 ms
5M	1.32 GiB	284.5 MiB	0.221 ms	3.644 ms

examples (Table 28). SID-MLP offers the best quality, the lowest one-time offline compute, and the smallest compute-equivalent volume.

E.3 MILLION-ITEM PREFIX INDEX

We stress-tested the valid-prefix index with 25K–5M unique SIDs on an A6000 (batch 32, beam 50). This is a synthetic unique-SID stress test. The index stores, for every prefix, whether each next token is valid; "lossless bit-packing" packs the same Booleans into bits without deleting or approximating any prefix, so valid SIDs and recommendations are unchanged (Table 29).

The experiment shows that index memory grows with the number of unique indexed prefixes, and therefore linearly in this conservative synthetic construction, but lookups do not slow down: the last-level mask lookup stays around 0.22 ms and a full beam around 3.6 ms from 25K to 5M SIDs. The 1M/5M indexes take 285.0 MiB/1.32 GiB, reduced to 69.5 MiB/284.5 MiB by lossless bit-packing; relative to the vectorized dense implementation, bit unpacking adds approximately 0.057 ms (1.6% of a full beam). Scaling the catalog is therefore a memory question, not a latency one. More generally, STATIC (Su et al., 2026) implements the same validity constraint as an accelerator-native CSR-based trie index, reporting 0.033 ms/step (0.25% overhead) at industrial-scale deployment.

F BASELINE REPRODUCTIONS

F.1 ATSPEED ADAPTATION

Adaptation target. AtSpeed is adapted to TIGER rather than used in its original open-vocabulary causal-LM setting. TIGER uses a T5 encoder–decoder and emits four constrained SID digits.

Evaluation setting. The AtSpeed rows in Tables 3 and 26 use the same TIGER checkpoints and sentence-T5 SIDs as the main results. The profile uses the full test split and beam $B = 50$. AtSpeed-S uses strict tree verification: an accepted step requires the teacher top-B beams to be contained in the draft tree. AtSpeed-R uses relaxed verification: draft samples are accepted by rejection sampling against the teacher distribution, and missing beams are filled from the correction distribution.

Why AtSpeed is slow on TIGER. Original AtSpeed targets long autoregressive LM decoding. Its speedup comes from using a cheap draft to verify many future tokens of a costly teacher. TIGER does not match this setting. The teacher has only 4.59M parameters, and generation stops after

four constrained SID digits. AtSpeed-S reaches 286.3 samples/s, or $0.68\times$ TIGER-kv. It matches teacher quality because it verifies teacher greedy top-B containment, but it still pays for draft beam search and teacher tree verification. With only four decoder steps, this overhead is hard to amortize. AtSpeed-R reaches 94.0 samples/s, or $0.22\times$ TIGER-kv and trails TIGER by 27–37% NDCG@10 across the three datasets. This drop follows from the relaxed sampling target.

F.2 EARN and State-Space Decoder Adaptations

This subsection documents the remaining baseline ports used in Table 3. These adaptations keep the TIGER SID tokenizer and valid-SID constraints fixed, so the comparison isolates decoder-side acceleration under the same recommendation task. Unless noted otherwise, baseline hyperparameters follow the authors' recommendations unless adaptation to the TIGER backbone is required.

EARN adaptation. We adapt EARN to the TIGER encoder because TIGER stores the user history in encoder states before SID decoding. One prefix register and one suffix register are placed at the first and last valid encoder positions. After encoder layer k, all non-register states are pruned; the remaining encoder layers and the decoder cross-attention use only the register states. We jointly fine-tune the resulting T5 model and the register embeddings, sweep $k \in \{1, 2, 3\}$, and select checkpoints by validation NDCG@10.

EARN attention diagnostic. We also check whether TIGER supports EARN's head/tail register placement. On 1,280 validation samples, TIGER encoder attention does not form the dual sink reported by EARN: across the four encoder layers, the first three positions receive 10.1–14.2% of attention and the last three positions receive 11.0–11.7%, close to or below the uniform three-position baseline of 13.6%. The top-attended positions are mostly middle history tokens. This differs from the decoder-only premise used by EARN, where BOS is repeatedly visible through causal attention and the final prompt token can summarize the full history.

State-space decoder adaptation. GDN and Mamba2 use the same frozen TIGER encoder features as SID-MLP. Each model receives the encoder hidden states followed by a learned start token or a bridge token and prefix embeddings from the frozen TIGER token embedding. It outputs 256-way logits for each SID digit and applies the same valid-SID constraints during beam search. We train only the SSM-side modules, bridge/start parameters, and output head with the same KL+CE objective as SID-MLP. During beam search, each active prefix carries an SSM recurrent state; every digit step gathers parent states, updates them with the selected prefix token, and forks states for the expanded beams. This makes the short-SID setting dominated by state movement rather than long-sequence scan complexity.

F.3 NEZHA Reproduction and Adaptation Details

This appendix documents how we reproduced NEZHA (Wang et al., 2026b) in our two teacher settings. The key point is that we preserve NEZHA's hidden-state contract: a backbone pass produces one root hidden state and one hidden state per SID digit, and an MTP head predicts SID digits with per-position heads plus a recurrent state update and joint fine-tuning. The adaptation only maps NEZHA's placeholder construction to the teacher architecture: encoder-side placeholders for TIGER, response-side placeholders after the LC-Rec prompt for LC-Rec.

TIGER adaptation. TIGER encodes the user sequence with a T5 encoder and normally decodes the target SID with a T5 decoder. To instantiate NEZHA without changing the SID tokenizer, we use the same abstract input pattern on the encoder side:

$$[\mathrm{BOS}, q_{\mathrm{user}}, x_1, \ldots, x_t, \mathrm{SP}_1, \mathrm{SP}_2, \mathrm{SP}_3, \mathrm{SP}_4],$$

where q_{user} is TIGER's user/context token and $x_1, \ldots, x_t$ are the history SID tokens. The final context hidden state before SP_1 is used as h_0, and the four placeholder hidden states are used as $h_1, \ldots, h_4$. The T5 decoder is bypassed in the NEZHA path. The T5 encoder and the MTP head are jointly fine-tuned, matching NEZHA's joint backbone-plus-head training principle. The target remains exactly the four TIGER SID digits. For TIGER NEZHA, we tuned the base learning rate over $\{5\mathrm{e}{-5}, 1.25\mathrm{e}{-4}, 2.5\mathrm{e}{-4}, 5\mathrm{e}{-4}, 1\mathrm{e}{-3}\}$, swept weight decay over $\{1\mathrm{e}{-5}, 1\mathrm{e}{-3}, 5\mathrm{e}{-2}\}$, and tested NEZHA's official parameter-group learning-rate scaling; neither improved the reported TIGER result.

LC-Rec adaptation. LC-Rec is already a decoder-only LLaMA-based generative recommender trained with its own prompt template. For adaptation, the input is the same instruction prompt used by LC-Rec with an empty response field, followed by four SID placeholders:

$$[\mathrm{BOS}, q_{\mathrm{pre}}, x_1, \ldots, x_t, q_{\mathrm{resp}}, \mathrm{SP}_1, \mathrm{SP}_2, \mathrm{SP}_3, \mathrm{SP}_4],$$

where q_{pre} and q_{resp} are the LC-Rec prompt text before and after the rendered history; q_{resp} includes the response header. Thus the placeholders are appended after LC-Rec's response prompt, not directly after the final historical item SID. The hidden state of the final prompt token before SP_1 is h_0, and the four placeholder hidden states are $h_1, \ldots, h_4$. The labels at the placeholder positions are the original four LC-Rec SID tokens. This keeps the LC-Rec teacher's input semantics intact: the history and instruction remain in the prompt, while the response SID span is replaced by NEZHA placeholders.

LC-Rec NEZHA training. We fine-tuned the 7B LC-Rec backbone across four A6000 GPUs. Given the computational cost, we focused our sweep on the base learning rate $\{1\mathrm{e}{-6}, 5\mathrm{e}{-6}, 1\mathrm{e}{-5}, 5\mathrm{e}{-5}, 1\mathrm{e}{-4}\}$ and selected the best validation checkpoint. To ensure a rigorous reproduction, we extensively tested other high-impact factors: we compared raw-SID inputs versus prompt-preserving inputs, applied NEZHA's official parameter-group learning rate multipliers (logit head $1\times$, token embedding $100\times$, transition $10\times$), and swept the inference MTP search budget over $\{10, 20, 50, 512\}$. Despite these efforts, none of the configurations closed the accuracy gap to the LC-Rec teacher.

Interpretation. On both TIGER and LC-Rec, NEZHA acts as a valid high-speed joint-finetuning baseline but consistently suffers from accuracy degradation. For TIGER, the port keeps NEZHA's MTP mechanism but moves placeholders to the encoder and trains with hard SID labels. For LC-Rec, the prompt-preserving adaptation isolates the modification strictly to the response generation path. Despite mechanically well-defined ports and extensive hyperparameter sweeps, NEZHA fails to automatically recover the quality of the original teachers.

G Additional Analyses

G.1 Hyperparameter and m-Mode Analysis

This appendix reports the full hyperparameter and m-mode curves for Instruments, Scientific, and Games. Figure 5 is organized by analysis type: each row is one sweep, and each column is one dataset. All runs use the same distillation recipe as the main experiments. Each sweep changes one hyperparameter around the configuration reported in Table 3 and keeps all other settings fixed. The star marks the Table 3 result; every other point is a separately trained model whose checkpoint is selected by validation NDCG@10, and the plots report its test NDCG@10 to show sensitivity.

α sweep. All three datasets vary smoothly with α. For $\alpha \in [0.5, 0.8]$, test NDCG@10 stays within 1.4% of its maximum on every dataset and is higher than both pure CE (α=0) and pure KL (α=1), indicating that the KL+CE mixture is robust.

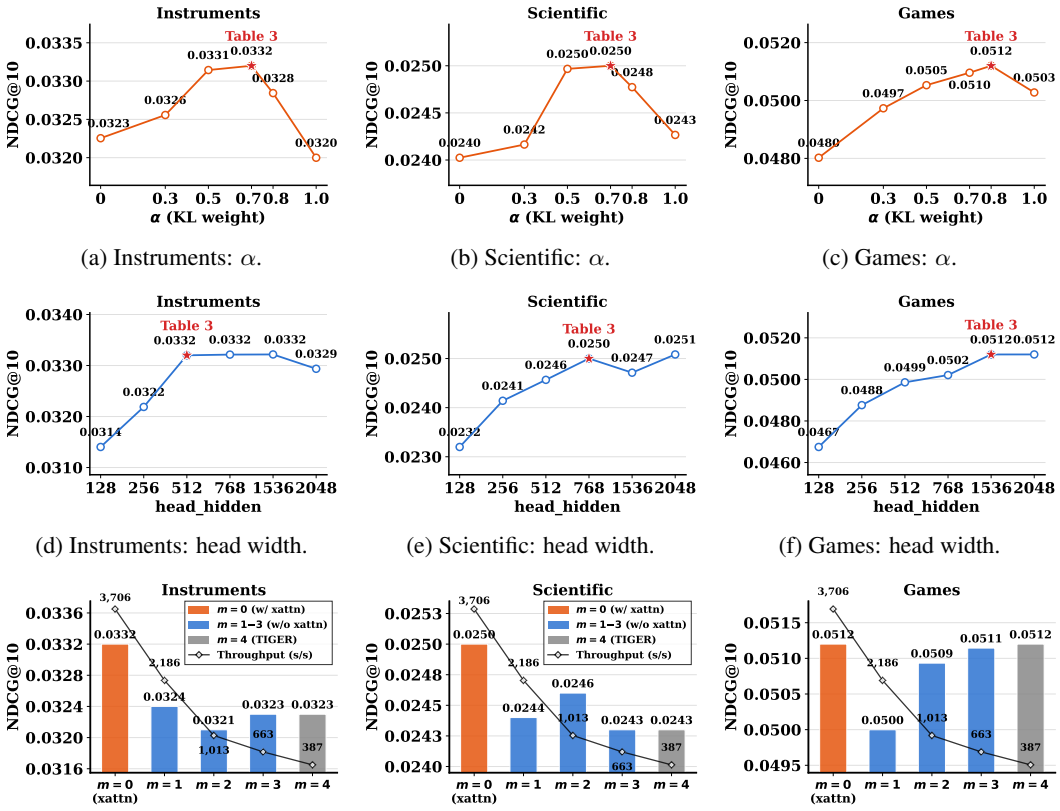
Scale-up. Test NDCG@10 flattens once the head is wide enough, while throughput decreases monotonically with head width. The configurations in Table 3 sit in this flat region rather than at the highest point: on each dataset a wider head reaches equal or slightly higher test NDCG@10 (e.g., 0.0251 at width 2048 versus 0.0250 at 768 on Scientific) at lower throughput.

m-mode tradeoff. All three datasets show the same tradeoff shape. m=0 is the fastest fully distilled setting and matches the teacher in the main results. m=1 anchors only the first SID digit with the teacher and provides an intermediate accuracy–speed point, consistent with the branching analysis in Section 2.2.

G.2 Distilled Encoder Ablations

We ablate the SID-MLP++ encoder distillation along two key design choices. Table 30 reports NDCG@10 across Instruments, Scientific, and Games; the Δ columns show the relative change versus the full SID-MLP++.

Position-specific MLPs. Sharing a single MLP across all positions consistently degrades NDCG@10 by 0.6–1.4%. This indicates that using dedicated MLPs for different token roles is beneficial.



(a) Instruments: α. (b) Scientific: α. (c) Games: α.

(d) Instruments: head width. (e) Scientific: head width. (f) Games: head width.

(g) Instruments: m-mode. (h) Scientific: m-mode. (i) Games: m-mode.

Figure 5: **Hyperparameter and m-mode analysis.** Top row: $\alpha \in \{0, 0.3, 0.5, 0.7, 0.8, 1.0\}$ sweep. Middle row: head-hidden width sweep. Bottom row: m-mode accuracy–throughput tradeoff. Columns correspond to Instruments, Scientific, and Games. In the top two rows, stars mark the configuration reported in Table 3. Points report test NDCG@10; checkpoints are selected on validation NDCG@10.

Table 30: **SID-MLP++ encoder distillation ablations.** Values are test NDCG@10 on Amazon Reviews 2023. Δ represents the relative change compared to the full SID-MLP++.

	Instruments		Scientific		Games	
Variant	N@10	Δ	N@10	Δ	N@10	Δ
SID-MLP++	**0.0327**	—	**0.0244**	—	**0.0486**	—
Shared MLP	0.0325	-0.6%	0.0242	-1.0%	0.0480	-1.4%
End2End	0.0308	-5.8%	0.0217	-11.3%	0.0454	-6.6%

Two-stage distillation. The *End2End* variant skips the Stage-1 hidden-state matching (MSE loss) and trains the encoder directly using only the final KL+CE ranking objective. This causes severe degradation: -5.8% on Instruments, -11.3% on Scientific, and -6.6% on Games. These substantial drops confirm that the final ranking loss is too sparse and indirect to optimize an MLP-based sequence encoder from scratch. Stage-1 MSE pre-training is crucial; it provides a dense, token-level supervision signal that aligns the lightweight encoder's representations before fine-tuning the prediction heads.

Performance difference between SID-MLP and SID-MLP++. SID-MLP++ additionally distills and replaces the teacher encoder, trading extra encoder approximation error for further acceleration; this is the source of the gap. To isolate it, we keep the same complete pretrained TIGER decoder, kept frozen, and only swap its input from the teacher encoder states to the SID-MLP++ Stage-1 distilled encoder states. This alone causes relative N@10 drops of 2.08%/2.25%/5.35% on the three datasets,

Table 31: **Source of the SID-MLP++ gap.** N@10 values are percentages on Amazon Reviews 2023.

Dataset	Frozen TIGER decoder: teacher encoder → distilled encoder	SID-MLP → SID-MLP++
Instruments	$3.23 \rightarrow 3.16$ (-2.08%)	$3.32 \rightarrow 3.27$ (-1.51%)
Scientific	$2.43 \rightarrow 2.37$ (-2.25%)	$2.50 \rightarrow 2.44$ (-2.40%)
Games	$5.12 \rightarrow 4.85$ (-5.35%)	$5.12 \rightarrow 4.86$ (-5.08%)

with the same direction, the same dataset ordering, and similar magnitude as the final SID-MLP →SID-MLP++ drops of 1.51%/2.40%/5.08% (Table 31). The gap therefore mainly comes from residual representation error introduced by encoder distillation, not from any failure of the SID-MLP decoder; Stage 2 retrains the heads on the distilled states and absorbs part of the interface mismatch.

Positioning. SID-MLP is the quality-first core method: it retains the teacher encoder and replaces only the decoder that beam search executes repeatedly. SID-MLP++ is an optional, more aggressive Pareto point: +17.3% throughput over SID-MLP ($8.74\times \rightarrow 10.25\times$) at the cost of two-stage offline training and a 1.5–5.1% relative N@10 drop. We recommend SID-MLP when accuracy is the priority and SID-MLP++ under stricter latency budgets.

H Extended Related Work

This section adds the works that Section 5 does not discuss and relates them to the experiments in this appendix.

Backbones and industrial systems. Besides T5-style encoder–decoders, GR has been built on GPT-style decoders (Petrov & Macdonald, 2023), on pretrained language models adapted to industrial-scale recommendation (He et al., 2026), and on hybrids that combine generative retrieval with dense retrieval (Yang et al., 2024). Our LC-Rec experiment (Section D.2) covers the decoder-only LLM case.

Item tokenization. Early work compares indexing schemes for recommendation foundation models (Hua et al., 2023). Many later tokenizers learn codes from contrastive or collaborative signals (Zhu et al., 2024; Zhai et al., 2025b; Liu et al., 2025b; Chen et al., 2024) or use coarse-to-fine quantization (Wei et al., 2025). PSID (Zhang et al., 2025) produces unique SIDs without an extra conflict-avoidance token. Other works extend SIDs to multiple behaviors (Liu et al., 2024b), personalized context (Zhong et al., 2025), and multiple modalities and aspects (Zhai et al., 2025a; Zhang et al., 2026a; Zhu et al., 2025). End-to-end methods train the tokenizer and the recommender together, through bi-level optimization, soft or differentiable identifiers, or personalized tokenizers (Bai et al., 2025; Fu et al., 2026a; Li et al., 2026a; Jiang et al., 2026; Wang et al., 2026a). SID-MLP takes the trained tokenizer as given. Figure 3(a) and Section D.3 evaluate it with RQ-VAE, RQ-KMeans, and PSID tokenizers, and with longer, heterogeneous, and multimodal SIDs.

Analysis and deployment of SIDs. Practical studies report that SIDs improve generalization in ranking (Singh et al., 2024), give practitioner recipes for SID-based GR (Ju et al., 2025), and describe SID use cases at Snapchat (Ju et al., 2026). Other studies analyze the expressiveness limits of autoregressive SID generation (Hou et al., 2026a), the staleness of collaborative SIDs as interactions change (Baikalov et al., 2026), and data augmentation for GR (Lee et al., 2026). Our temporal item-shift diagnostic (Section D.5) is related to the staleness setting.

Alternative generation procedures. SETRec (Lin et al., 2025a) uses order-agnostic identifiers, Marius (Lepage et al., 2025) combines collaborative tokenization with efficient modeling, and ContRec and MaskGR generate items with continuous-token diffusion and masked diffusion (Qu et al., 2026; Shah et al., 2025). JTM (Zhu et al., 2019) extends tree-based retrieval by optimizing the tree index and the deep model jointly. These designs define a new identifier, model, or index and train the recommender with it, whereas SID-MLP distills an already trained teacher.

LLM decoding acceleration. Multi-query attention shares keys and values across heads to reduce memory traffic during decoding (Shazeer, 2019). Hydra (Ankner et al., 2024) makes Medusa draft heads sequentially dependent, and EAGLE (Li et al., 2024b; 2026b) drafts at the feature level. These

methods accelerate the generation of one text sequence, while GR must return the top-K beams, which AtSpeed (Lin et al., 2025b) formulates as N-to-K verification.

Distillation and sequence models. FitNets (Romero et al., 2015) distills intermediate hidden states in addition to outputs; Stage 1 of SID-MLP++ (Section 3.5) similarly matches the teacher's encoder states. S4 (Gu et al., 2022) introduced the structured state-space layers that Mamba builds on, and Gated DeltaNet (Yang et al., 2025b) adds a delta-rule update to Mamba2. We use Mamba2 and GDN as decoder baselines in Table 3.