

Personalized LLM Recommendation with Semantic IDs: Autoregressive Beam Search vs. Multi-Token Prediction at Industry Scale

Sejoon Oh* Fernando Amat Gil Evan Smothers Dan Iter Mark Thornburg
Dawit Mureja Argaw Sourabh Medapati Thea Wang Ria Verma Qiuling Xu
Harshit Syal Ehsan Gholami Moumita Bhattacharya Ashish Rastogi

Netflix Personalization Foundations Team

Abstract

Large-scale personalization is shifting from embedding-table retrieval to LLM-based recommenders that *generate* the *Semantic ID* (SID) of the item to recommend, where each SID is a short sequence of discrete hierarchical content codes. A single instruction-tuned model can then unify retrieval, content understanding, and language, and recommend brand-new titles from their content alone using SID; then, *how* should the model generate the SID? Traditionally, models rely on a single language modeling head that generates tokens autoregressively, typically using constrained beam search. In contrast, an emerging body of work explores multi-token prediction (MTP), where dedicated heads predict all tokens in parallel, substantially reducing the decoding latency associated with beam search. We report an *industry-scale, apples-to-apples* comparison of these two *generation paradigms*: single-head autoregressive decoding vs. MTP. We introduce a new auxiliary ranking loss, defined over the joint SID score, to train the MTP objective. We evaluate the SID training objectives on both quality and training/inference cost, inside a multi-task LLM that jointly does personalized retrieval, content question-answering (QA), and instruction following over a large streaming-media catalog. Our experiments show **single-head beam search consistently beats MTP on most product-critical metrics**: ranking and content QA, and trains at roughly **2.6x** lower dollar cost on *Qwen3-8B* scale; MTP wins only on raw inference throughput. Around this finding, we detail an efficient training and inference stack and report end-to-end gains over the *previous non-SID baseline equipped with an auxiliary ranking head*: **+77%** item cold-start ranking, **+22%** content-QA accuracy, **+2.8x** faster training, with language metric regression traced to SID vocabulary expansion. We address the language regression by reinforcement learning post training.

1 Introduction

Recent large-scale personalization systems increasingly leverage a large language model (LLM) to unify retrieval, ranking, content understanding, and natural-language understanding. Such an LLM ingests a member’s interaction history and context and must (i) *retrieve* relevant catalog entities, (ii) *understand* title content well enough to *answer* natural-language questions about it (content QA), and (iii) *follow* general instructions. We treat content understanding as a first-class training and evaluation objective alongside ranking, since it lets the model reason about *why* a title fits a member,

*Project lead / corresponding author: sejoono@netflix.com.

ground recommendations in metadata (not just popularity), and generalize to titles with little or no engagement history.

Modern systems refer to catalog entities not by arbitrary integer ids but by *Semantic IDs* (SIDs): short sequences of discrete codes produced by residual quantization [15, 20, 17, 18] of a content embedding, so that semantically similar titles share early-level codes. The model then *generates* the target’s SID on its own output head, which removes the catalog-sized classification head, unifies retrieval with the language and content-understanding tasks, and greatly improves cold-start because a new title inherits a meaningful (non-random) SID from its content embedding. This recipe now underlies several industrial systems (YouTube’s PLUM [7], Spotify’s GLIDE [3], Kuaishou’s OneRec [5], Meta’s generative recommenders [26]), which report substantial online gains.

This raises the question at the focus of this paper: *how should the LLM generate SID?* **(a) Single-head autoregressive** decoding emits the codes one at a time on the standard LM head with constrained beam search, which is also the classical recipe of DSI and TIGER [20, 17] and the fine-tuning stage of PLUM [7] and GLIDE [3]. **(b) Multi-token prediction (MTP)** attaches L dedicated heads to a shared trunk and predicts all codes in parallel [6]. Leveraging this parallel decoding, MTP can generate SID faster than the beam search in inference time. Only a few studies such as RPG [9] and STAMP [27] have applied MTP in generative recommendations. Those studies achieve higher ranking metrics or training efficiency by leveraging parallel SID prediction.

However, these arguments are often made on small, retrieval-only academic benchmarks. Using an open-source model with an industry-scale streaming dataset as a controlled testbed, we reach the conclusion in the regime that matters for deployment: **single-head constrained beam search beats MTP on most product-critical metrics** (ranking and content QA), with a significantly lower training cost on *Qwen3-8B* scale. Meanwhile, MTP outperforms on inference throughput only.

Our contributions are:

- **The industry-scale, apples-to-apples comparison of single-head autoregressive decoding vs. MTP as SID generation paradigms.** A *quality-and-cost* comparison of training objectives, inside a unified multi-task LLM and large-scale streaming dataset, on ranking, content QA, language, and training/inference performance (Section 4.2).
- **An auxiliary ranking loss for MTP**, a new objective that optimizes the item-level joint SID score so that MTP’s parallel heads produce calibrated, rankable item scores despite their conditional-independence assumption (Section 3.2.2).
- **An efficient training and inference stack:** Muon [11], on-the-fly packing with cross-rank batch mixing, Cut Cross-Entropy [22], and HSDP [28] for training ($> 2\times$ faster at -73% cost); and **vLLM-based constrained, catalog-aware beam-search decoding** for inference [13], whose trie- and catalog-level constraints guarantee every candidate names a real, request-eligible entity while prefix caching reuses the long member-history KV cache across beams (Sections 3.3–3.4).
- **A comprehensive evaluation and ablations** covering new user and item recommendations, plus controlled ablations of LoRA [10] vs. full fine-tuning, Muon [11] vs. AdamW [16] optimizers, and content QA tasks (Section 4).

2 Related Work

Generative retrieval and semantic IDs. The Differentiable Search Index [20] first framed retrieval as sequence-to-sequence generation of document identifiers. TIGER [17] introduced *semantic IDs* for recommendation, obtaining discrete codes from an RQ-VAE [15] over item content embeddings and autoregressively generating the next item’s id. [18] show semantic IDs generalize better than random atomic ids for ranking. At industrial scale, PLUM [7] adapts a pre-trained LLM to generative retrieval via continued pre-training on SID-tokenized behavior, GLIDE [3] frames podcast discovery as instruction-following over a SID-discretized catalog, and OneRec [5] unifies retrieval and ranking in one generative model. We adopt this representation but place the SID vocabulary on a general-purpose instruction-tuned LLM’s own head, so that retrieval, content QA, and language tasks share a single decoder, rather than training a bespoke retrieval-only transformer.

Autoregressive beam search vs. parallel/MTP decoding. MTP was introduced as an auxiliary objective that predicts n future tokens with n independent heads on a shared trunk, improving sample efficiency and enabling self-speculative decoding [6]; it is now a component of frontier LLMs [4]. In generative recommendation (GR), a recent wave applies this idea to SIDs to escape the autoregressive bottleneck. RPG [9] predicts long (up to 64-token) *unordered* SIDs in parallel with an MTP loss and graph-constrained decoding for higher efficiency and ranking quality; STAMP [27] introduces an auxiliary multi-token prediction objective to densify supervision, improving representation learning while semantic pruning reduces training cost. [23] question constrained decoding’s generalization theoretically. We revisit these conclusions *empirically and at industry scale* (Section 4.2): within a unified multi-task LLM, and even after strengthening MTP with an auxiliary ranking loss, single-head beam search delivers better ranking and content-QA quality at lower training cost, while MTP’s advantage is limited to inference throughput and full-catalog ranking capability².

Speculative and self-speculative decoding. A distinct line of work uses multi-token or draft heads purely as a *lossless accelerator*: Medusa [2] adds parallel decoding heads verified by tree attention, Speculative Streaming [1] folds future n -gram drafting into the target model, and [25] combine Medusa with distillation for an industrial travel application. Crucially, these methods speed up inference while *preserving* the underlying autoregressive model’s output distribution, so they do not trade quality for speed; they are orthogonal to, and can be layered on top of, our single-head decoder rather than being an alternative to it. NEZHA [21] makes the distinction explicit: it accelerates SID beam search with an *autoregressive* self-drafting head plus a hash-set verifier, and reports that a plain MTP decoder is faster but markedly worse in quality because it cannot exploit the sequential dependency inherent in the SID. We instead compare single-head vs. MTP as *generation paradigms* (a quality-and-cost question), whereas this line targets quality-preserving acceleration (a speed question).

Efficient LLM training and inference. Our stack combines existing components: Muon [11], Cut Cross-Entropy [22], sequence packing [12], and HSDP [28] for both single-head and MTP training, and vLLM’s PagedAttention and prefix caching [13] for inference with the single-head and beam search. Our contribution here is not any single component but their integration and measured joint effect on a large-scale run, plus the trie- and catalog-constrained decoding layer we build on vLLM.

3 Method

We first define how entities are represented and generated as Semantic IDs (Section 3.1), then the two *training* paradigms we compare: single-head and MTP, the latter equipped with our auxiliary ranking loss (Section 3.2). We also introduce their corresponding *inference* procedures (Section 3.3), and finally the efficient training/inference stack (Section 3.4).

3.1 Semantic-ID Generation

Let $\mathcal{E}$ be the catalog of entities. Each entity $e \in \mathcal{E}$ is mapped to a *Semantic ID*

$$\text{SID}(e) = (c_1(e), c_2(e), \dots, c_L(e)), \quad c_\ell(e) \in \{1, \dots, K_\ell\}, \quad (1)$$

a length- L sequence of discrete codes drawn from per-level codebooks of sizes $K_1, \dots, K_L$, obtained by residual quantization [15] of a content embedding $\phi(e) \in \mathbb{R}^d$. The quantizer is trained so that the first code captures the coarsest semantic partition and later codes refine it, so entities close in content space share early-level codes. The framework is agnostic to the SID *shape*: level sizes may be uniform (e.g., 1024×4 , meaning $K_\ell=1024$ for all $L=4$ levels) or pyramidal (e.g., $1024 \times 512 \times 256 \times 128$, spending more capacity on the coarse levels); our experiments use the uniform 1024×4 shape. The $\sum_\ell K_\ell$ SID tokens are appended to the LLM vocabulary $\mathcal{V}$, giving $\mathcal{V}^+ = \mathcal{V} \cup \mathcal{S}$ with $|\mathcal{S}| = \sum_{\ell=1}^L K_\ell$. While SID collisions rarely happen (e.g., $\leq 1\%$), we resolve the collision by returning the first entity in the SID bucket.

We place the SID vocabulary on the LM head and *generate* the target’s codes, rather than scoring a catalog-sized auxiliary head as the baseline did. This gives (i) a *simpler architecture* that has one model, one head, with no catalog-sized layer to provision or re-train as the catalog grows; (ii) *stronger*

²It is computationally infeasible for the beam search to generate probabilities across entire catalog.

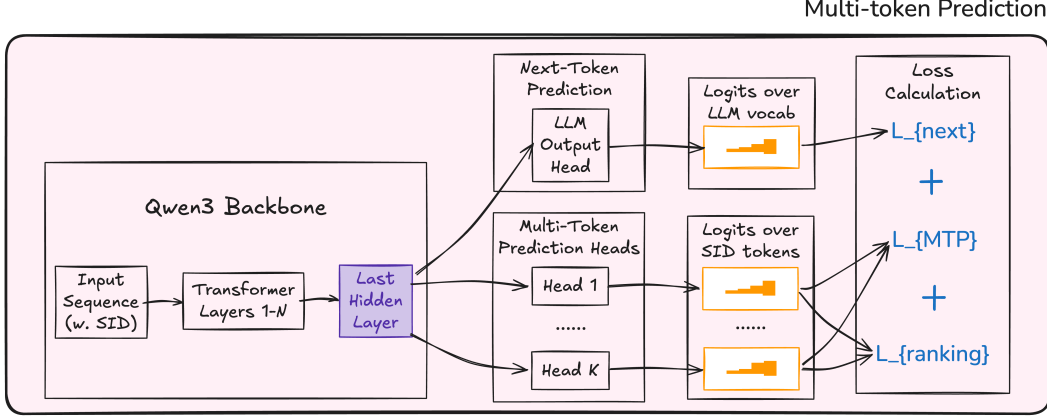


Figure 1: **Multi-token-prediction (MTP) training.** While the single-head approach uses a LM head with a next-token loss (Eq. 2), MTP adds L parallel SID heads trained with per-code cross-entropy and our auxiliary ranking loss over the joint SID score (Eq. 5).

cold-start, since a new title’s SID is computed from its content embedding and can be recommended before any member interacts with it, by sharing SID prefixes with established titles³, and (iii) *shared representations*, since retrieval, content QA, and instruction following reuse the same head.

3.2 Training

Every training example (e.g., retrieval, content QA, or instruction following) is a token sequence over $\mathcal{V}^+$: retrieval targets are written as the target’s SID tokens, content QA references titles by both text and SID, and instruction data is unchanged. We compare two ways of producing the SID codes; both share the same backbone, tokenizer, and data (Figure 1).

3.2.1 Single-Head (LM Head) Training

The classical design uses the single next-token LM head to emit SID codes *autoregressively*, so training is one next-token problem whose per-example loss splits into text and SID positions,

$$\mathcal{L}^{\text{SH}} = - \sum_{i \in \mathcal{T}_{\text{text}}} \log p_{\theta}(y_i | y_{<i}) - \lambda \sum_{i \in \mathcal{T}_{\text{sid}}} \log p_{\theta}(y_i | y_{<i}), \quad (2)$$

where $\mathcal{T}_{\text{text}}$ and $\mathcal{T}_{\text{sid}}$ index text and SID target positions and $\lambda \geq 1$ up-weights the sparser SID positions. Crucially, this autoregressive training enforces dependencies between the SID levels, where predicting the current code c_{ℓ} is conditioned on previous codes $c_{<\ell}$. During inference, beam search retains multiple early prefixes that later scores can rerank.

3.2.2 Multi-Token Prediction (MTP) Training

Following [6], MTP keeps the next-token LM head and adds L parallel heads $g_1, \dots, g_L$ on the shared last-hidden state h_x for query x , with head g_{ℓ} responsible for code c_{ℓ} . Because all codes are predicted *in parallel* from h_x , the standard MTP retrieval term is a sum of independent per-code cross-entropies,

$$\mathcal{L}_{\text{code}} = - \sum_{\ell=1}^L \log p_{g_{\ell}}(c_{\ell}(e^+) | h_x), \quad (3)$$

for the positive item e^+ . This factorization is exactly the weakness of MTP for ranking: it optimizes each code’s *marginal* likelihood but never the *joint* item score, so an off-target item whose individual codes are each independently probable can outscore the true target.

³For “extreme” cold-start entities whose SIDs are never exposed during training, we fall back to retrieval-augmented prompting or lightweight daily fine-tuning.

Auxiliary ranking loss. We address this with a new auxiliary objective that directly optimizes the joint, item-level score. Define the MTP score of a candidate e as the sum of its per-code log-probabilities,

$$s_{\text{MTP}}(e | x) = \sum_{\ell=1}^L \log p_{g_\ell}(c_\ell(e) | h_x), \quad (4)$$

and train the heads so the positive item ranks above a set $\mathcal{N}(x)$ of sampled in-batch/hard negatives from the eligible catalog, via a sampled-softmax⁴ ranking loss:

$$\mathcal{L}_{\text{rank}} = -\log \frac{\exp(s_{\text{MTP}}(e^+ | x))}{\sum_{e \in \{e^+\} \cup \mathcal{N}(x)} \exp(s_{\text{MTP}}(e | x))}, \quad \mathcal{L}^{\text{MTP}} = \mathcal{L}_{\text{next}} + \alpha \mathcal{L}_{\text{code}} + \beta \mathcal{L}_{\text{rank}}, \quad (5)$$

with importance hyperparameters α, β , and $\mathcal{L}_{\text{next}}$ the text next-token loss retained from the LM head. Unlike standard MTP [6], which uses per-token cross-entropy only, and RPG [9], which pairs MTP with graph-constrained decoding at inference, $\mathcal{L}_{\text{rank}}$ teaches the *independent* heads to produce jointly calibrated, rankable item scores at *training* time, which partially compensates for the missing autoregressive coupling and gives MTP its strongest form. We empirically confirm that having $\mathcal{L}_{\text{rank}}$ improves ranking metrics, while increasing training cost.

3.3 Inference

3.3.1 Single-Head: Constrained, Catalog-Aware Beam Search

A single-head retrieval call generates the target SID token-by-token. Let $\mathcal{T}$ be the SID *trie* of all prefixes realized by some entity, and $\mathcal{C}_x \subseteq \mathcal{E}$ the per-request eligible catalog for query x . At step ℓ with prefix $c_{<\ell}$, we restrict the next-token distribution to the admissible set

$$\mathcal{A}(c_{<\ell}, x) = \{ k : (c_{<\ell}, k) \in \mathcal{T} \text{ and } \exists e \in \mathcal{C}_x, \text{SID}(e) \succ (c_{<\ell}, k) \}, \quad (6)$$

where $\succ$ denotes “has as a prefix”. *Constrained* decoding guarantees we never emit a SID that names no real entity; *catalog-aware* decoding guarantees every candidate is eligible for the request, sharpening local ranking metrics without post-hoc filtering. Beam search keeps the top- N partial hypotheses per level, scoring a complete SID by its length-normalized log-probability

$$s(e | x) = \frac{1}{L} \sum_{\ell=1}^L \log p_\theta(c_\ell(e) | x, c_{<\ell}(e)), \quad (7)$$

and returns the top- K entities. We implement this on vLLM [13] that reuses the long member-history prompt’s KV cache across beams and decoding steps, so wide beams are affordable, and the trie/catalog constraints are applied as a logit mask at each step.

3.3.2 MTP: Parallel Decoding

MTP produces all L per-code distributions from h_x in a *single* forward pass and scores candidates by s_{MTP} (Eq. 4); the top- K eligible entities are selected by evaluating s_{MTP} over the trie-restricted candidate set, without sequential beam expansion. This removes the per-code forward passes of beam search, which is the source of MTP’s inference-throughput advantage. However, it also removes the autoregressive coupling crucial for accurate retrieval and ranking. Thus, we utilize the ranking loss $\mathcal{L}_{\text{rank}}$ to recover the retrieval/ranking ability. MTP is particularly useful for scenarios requiring dense, full-catalog scores with non-zero probability mass on every title.

3.4 Efficient Training and Inference Stack

The comparison above is enabled by a stack that makes large-scale fine-tuning and inference cheap. For *training*, we combine four standard components and report their integrated effect (Section 4.3): the **Muon** optimizer [11], which orthogonalizes matrix gradient updates and stores only momentum, roughly halving optimizer-state memory and converging in fewer steps than our AdamW [16]

⁴If the entire catalog is small (e.g., $\leq 1M$), normal Softmax without any sampled negatives can be used here.

baseline; **on-the-fly sequence packing** with block-diagonal masks [12] plus *cross-rank batch mixing* that assigns different tasks to different data-parallel ranks in one global batch, improving throughput and stability; **Cut Cross-Entropy** [22], a fused kernel that avoids materializing the large-vocabulary logits tensor; and **HSDP** [28], which shards state within a node and replicates across nodes, confining the heaviest communication to high-bandwidth intra-node links ($\sim 7\%$ absolute MFU gain on the 8B backbone). For *inference* we serve on vLLM (Section 3.3.1).

4 Experiments

4.1 Experimental Setup

Confidentiality and training data. We study the streaming-media recommendation domain. By *industry-scale* we mean an 8B-parameter production backbone fine-tuned over a proprietary catalog on $\geq 10M$ sampled examples *per task* drawn from production logs, evaluated under a production training- and inference-cost budget rather than on a small retrieval-only academic benchmark. To respect confidentiality we report *relative* rather than absolute figures for proprietary quantities (catalog size, corpus sizes, and dollar costs) and reported metrics (e.g., MRR, IFEval, MMLU). A retrieval task uses member interaction history data ($\geq 10M$) snapshotted at a fixed date-hour; a content QA task uses an internal content QA corpus ($\geq 10M$) derived from our rich content metadata; an instruction-following task uses a mix of open-source datasets [19, 14] ($\geq 10M$). SIDs come from the latest quantizer based on RQ-VAE [17] snapshot.

Models and baseline. All the experiments are based on open-weight *Qwen3-8B* backbone [24] with an optional 0.6B variant. We fine-tune such pre-trained models by supervised fine-tuning (SFT) plus LoRA [10] on our training data, and we conduct additional reinforcement learning (RL) after the SFT for language metric enhancement. We use up to 256 NVIDIA H200 and A100-80GB GPUs for our model training and inference. The *baseline* model in this paper refers to the previous **non-SID** model: a LoRA-only recipe with a catalog-sized auxiliary head for ranking. The additional head was necessary for the baseline to generate logits across our entire catalog. However, this ranking head can degrade training efficiency and increase maximum memory usage.

Content QA corpus. Content QA answers natural-language questions about a title (e.g., “What is *Movie A* about?”). We internally developed a large-scale *Content QA* corpus in which each title is referenced by *both* its text and its SID, so the entity representation matches what the model sees during retrieval. The corpus is generated synthetically by prompting a frontier teacher model (e.g., *GPT 5.2*) with internal title+SID context and spans four complementary question types: title-grounded descriptions, SID-centric semantics, member-like query grounding, and two-title comparisons. To reduce sensitivity to prompt style, we sample system prompts at training time, including a 20% probability of an empty system message.

Evaluation. Ranking is measured with normalized Mean Reciprocal Rank (nMRR). For queries Q ,

$$\text{MRR} = \frac{1}{|Q|} \sum_{x \in Q} \frac{1}{\text{rank}_x}, \quad \text{nMRR} = \frac{\text{MRR}}{\text{MRR}_{\text{ideal}}}, \quad (8)$$

where rank_x is the position of the first relevant entity and the normalizer accounts for the number of eligible relevant items per query. We report the **average** ranking performance using an internal test set across 7 different offsets (i.e., predict +1 – +7 days future interactions). We also offer detailed metrics computed only on discoveries (i.e., watching new content) and new users (e.g., users with $\leq K$ engagement) and items (e.g., titles launched after training cutoff), along with overall ranking. Content QA is scored by an LLM judge (*Qwen2.5-32B-Instruct*) [29] on a stratified test set. Among numerous language benchmarks, we report IFEval [30] and MMLU [8] in this paper.

4.2 Core Study: Single-Head Beam Search vs. Multi-Token Prediction

This is our central result. We take the *same* backbone, SID tokenizer, data mixture, and training budget per token and train the model in two different ways: (1) single-head autoregressive training (Eq. 2) and constrained beam search inference (Eq. 7) and (2) multi-token prediction with our auxiliary ranking loss (Eq. 5), i.e., MTP in its strongest form. This isolates the generation paradigm as the only variable, making the comparison apples-to-apples at production scale.

Table 1: **Single-head beam search vs. multi-token prediction.** Relative quality change of **the single-head over the MTP**; positive values favor the single-head. Metrics are grouped by task family.

Task family	Metric	<i>Qwen3-8B</i>	<i>Qwen3-0.6B</i>
Ranking	Overall ranking (nMRR)	↑+2.0%	↑+3.5%
	Discovery (nMRR)	↑+7.3%	↑+12.6%
	New user (nMRR)	↑+2.1%	↑+4.2%
	New item (nMRR)	↓-0.8%	↑+6.8%
Content Understanding	QA Accuracy	↑+2.0%	↑+12.7%
Language	IFEval (accuracy)	↑+3.3%	↑+2.3%
	MMLU (accuracy)	↑+5.7%	↑+4.8%

Table 2: Single-head vs. MTP: training and inference system metrics. Relative quality change of **the single-head over the MTP**. Single-head trains cheaper; MTP only wins inference throughput.

Backbone	Training Runtime ↓	MFU ↑	Tokens/sec ↑	Dollar Cost ↓	Inference/sec ↑
<i>Qwen3-8B</i>	0.37×	1.15×	1.18×	0.38×	0.57×
<i>Qwen3-0.6B</i>	0.50×	1.21×	1.20×	0.97×	0.18×

Quality: single-head wins where it counts. Single-head beam search beats MTP on all ranking and content-QA metrics (Table 1). On the *Qwen 3-8B* model, it improves discovery (↑7.3%), new user recommendation (↑2.1%), overall retrieval (↑2.0%), content understanding (↑2.0%), and general-language benchmarks (IFEval ↑3.3%, MMLU ↑5.7%). On the *Qwen3-0.6B* model, the single-head advantage is uniform and larger including *gains* on discovery (↑12.6%) and new item recommendations (↑6.8%). Across both scales, we attribute the ranking/QA boost of the single-head to autoregressive coupling: letting code c_ℓ condition on $c_{<\ell}$ lets the model correct early mistakes, whereas MTP’s parallel heads must commit to all codes from a single hidden state under a conditional-independence assumption.

Cost: single-head is far cheaper to train, but MTP is much faster at inference. The quality advantage does not come with a training-cost premium; in fact, the reverse (Table 2). For the 8B model, single-head trains $2.7\times$ faster than MTP with $2.6\times$ lower dollar cost, and higher model FLOPs utilization (MFU) & token per second. The *only* win for MTP is raw inference throughput (↑75%), consistent with the efficiency motivation of the MTP literature [6, 9]. In our deployment, this throughput edge does not offset the quality and training-cost gaps, so we **adopt single-head beam search as the default** and keep MTP only for applications that need dense full-catalog scores.

Reconciling with prior work. We confirm MTP’s inference advantage but overturn the *quality* claim that parallel prediction matches autoregressive generation. Two features of our regime explain the gap: a moderate SID length ($L=4$), where autoregressive decoding is cheap and its coupling most valuable (vs. the long $L\geq 64$ SIDs that make beam search prohibitive in RPG); and multi-task training, which lets the single head share representations across retrieval, content QA, and language.

4.3 Single-head Beam Search with SID vs. the non-SID Baseline

Having established the generation paradigm, we report the end-to-end gains of the full recipe (single-head + native SID + new training/inference stack) over the previous *non*-SID baseline.

Overall comparison. The single-head model with SID outperforms the non-SID baseline across ranking and content QA (Table 3). The headline gains are ↑77.3% on new item nMRR, ↑21.7% on content-QA accuracy, and ↑~5% on overall ranking & discovery. The significant new item nMRR boost is direct evidence that SID prefix sharing lets the model recommend titles it never saw engagement for. We observe regression on language metrics such as IFEval (↓7.3%) and MMLU (↓10.0%). The key reason is that expanding the vocabulary with LK SID tokens (assuming L -level codebook with K tokens per level) and co-training on them perturbs the instruction-following

Table 3: **Overall metric comparison (*Qwen3-8B*) of the single-head with SID vs. the non-SID baseline with an auxiliary ranking head.** Relative quality change of the SID model over the baseline; positive values favor the SID model. Metrics are grouped by task family. We observe enhanced ranking and content understanding, while language regression due to SID vocab extension. We confirm the language regression can be recovered by RL post training (e.g., IFEval $\uparrow+12.8\%$).

Task family	Metric	Single-head with SID vs. Non-SID baseline
Ranking	Overall ranking (nMRR)	$\uparrow+4.7\%$
	Discovery (nMRR)	$\uparrow+5.3\%$
	New user (nMRR)	$\uparrow+4.7\%$
	New item (nMRR)	$\uparrow+77.3\%$
Content Understanding	QA Accuracy	$\uparrow+21.7\%$
Language	IFEval (accuracy)	$\downarrow-7.3\%$
	MMLU (accuracy)	$\downarrow-10.0\%$

Table 4: Training system metrics between the single-head with SID vs. the non-SID baseline, where both models are based on *Qwen3-8B*. Multipliers are relative to the non-SID baseline.

Model	Runtime $\downarrow$	MFU $\uparrow$	Tokens/sec $\uparrow$	Dollar Cost $\downarrow$
Single-head with SID	$0.36\times$	$4.75\times$	$4.0\times$	$0.27\times$

distribution. To address this regression, we perform reinforcement learning (RL) with instruction-following datasets after the SFT stage, which yields $\uparrow+12.8\%$ improvement of the IFEval metric.

System efficiency. The training optimizations we introduced transform training economics (Table 4). Compared to the non-SID baseline without any training optimizations (Section 3.4), single-head model trains $2.8\times$ faster, raises MFU $\sim 4.75\times$ and tokens/sec $\sim 4.0\times$, and cuts training cost by 73%. These savings let us launch more model and ablation runs per unit budget.

4.4 Ablation Studies

LoRA vs. full fine-tuning (FT). We choose LoRA by default compared to full FT, as LoRA ranking metrics are on-par-or-better than full FT (Table 5), especially the cold-start item recommendation metric. According to Table 6, full FT is on-par-or-better than LoRA because its standard dense matrix operations avoid the extra kernel launch overheads and lower arithmetic intensity introduced by LoRA. On the 8B model, LoRA shows a language-metric drop relative to full FT because we apply LoRA to the output layer, whereas on the 0.6B model (where LoRA is not applied to the output layer) the effect reverses. We plan to revisit full FT given its simplicity and system performance.

Muon vs AdamW optimizer. Holding everything else fixed, we compare two popular optimizers: Muon [11] and AdamW [16]. As shown in Table 7, they reach similar quality, while Muon adds a small item cold-start gain, but Muon’s system performance is clearly superior; e.g., $\sim 20\%$ higher MFU and throughput, lower memory consumption, and larger training batch size. Thus, we set Muon as the default for both single-head and MTP training.

Contribution of content-QA data. Table 7 tests whether content understanding helps ranking. Removing *all* content-QA tasks leaves ranking nearly unchanged while collapsing content QA accuracy notably. We do not observe the synergy between ranking and content QA tasks in the current SFT-only recipe. This motivates moving content QA into a CPT stage followed by ranking-focused SFT, which is a direction for future iterations, and one consistent with the CPT+SFT of PLUM [7].

Other ablations. We also found that introducing a separate weight for SID tokens in (2) and having the auxiliary ranking loss for MTP (5) significantly improve ranking metrics. Note that we leave other non-critical ablations such as various SID codebook shape (e.g., fan-in or fan-out), SID level (e.g., large 2-level or small 8-level), or different task mixture ratios as future work.

Table 5: **LoRA vs. full fine-tuning on the single-head setup.** Relative quality change of **LoRA over full fine-tuning**; positive values favor LoRA. Metrics are grouped by task family. We choose LoRA by default because of better ranking metrics, especially the cold-start item recommendation.

Task family	Metric	<i>Qwen3-8B</i>	<i>Qwen3-0.6B</i>
Ranking	Overall ranking (nMRR)	↑+0.3%	↑+0.1%
	Discovery (nMRR)	↑+1.9%	0.0%
	New user (nMRR)	↑+0.9%	0.0%
	New item (nMRR)	↑+6.2%	↑+4.5%
Content Understanding	QA Accuracy	0.0%	0.0%
Language	IFEval (accuracy)	↓-5.2%	↑+3.3%
	MMLU (accuracy)	↓-6.1%	↑+3.1%

Table 6: System metrics for the LoRA FT vs. full FT ablation. **Multipliers are relative to LoRA** of the same backbone. Full FT is on-par-or-better than LoRA across all system metrics and model sizes, as it avoids the runtime memory overhead and kernel launch latencies introduced by LoRA.

Backbone	Variant	Runtime ↓	MFU ↑	Tokens/sec ↑	Dollar Cost ↓
<i>Qwen3-8B</i>	Full FT	1.0×	1.13×	1.02×	1.0×
<i>Qwen3-0.6B</i>	Full FT	0.87×	1.12×	1.17×	0.9×

Table 7: **Optimizer and content-QA-data ablations against single-head LoRA training.** The columns report Muon relative to AdamW and all tasks relative to all tasks without content QA, respectively; positive values favor the first variant. Metrics are grouped by task family.

Task family	Metric	Muon vs. AdamW	All tasks vs. no Content QA
Ranking	Overall ranking (nMRR)	↓-0.1%	↓-0.2%
	Discovery (nMRR)	↓-0.9%	0.0%
	New user (nMRR)	↑+0.1%	↓-0.1%
	New item (nMRR)	↑+5.2%	↑+4.0%
Content QA	Accuracy	↑+0.1%	↑+211.4%
Language	IFEval (accuracy)	↓-1.9%	↓-3.2%
	MMLU (accuracy)	↓-0.6%	↓-1.8%

5 Conclusion

Inside a unified multi-task LLM at industry scale, *single-head autoregressive SID generation with constrained beam search beats multi-token prediction* on most product-critical metrics (ranking and content QA) and trains at less than half the cost on *Qwen3-8B* scale, while MTP leads only on inference throughput. Even after we strengthen MTP with a new auxiliary ranking loss, the single-head wins. Furthermore, the end-to-end recipe, from efficient training optimizations with SID to vLLM-served constrained catalog-aware beam search, more than halves training time and cost and lifts cold-start (↑+77%) and content-QA (↑+22%) over the non-SID baseline. Our findings have some limitations: (1) most experiments are based on offline training and test data, and online A/B test results might exhibit different trends, (2) the LLM could be overfit to our tasks and input format, and the output quality could be sub-optimal for diverse input format and tasks, and (3) missing continual pre-training (CPT) or mid-training stage that can warm-up the SID token embeddings and reduce the training data volume for the SFT stage. We will probe the crossover in SID shape and length, catalog size, and task mixture at which it tips. We will also extend both single-head and MTP frameworks to multi-stage training such as CPT+SFT and CPT+SFT+RL. Finally, we will diversify our input prompt style and format and utilize a larger number of tasks during the SFT, to make our LLM robust.

References

- [1] Nikhil Bhendawade, Irina Belousova, Qichen Fu, Henry Mason, Mohammad Rastegari, and Mahyar Najibi. Speculative streaming: Fast LLM inference without auxiliary models. In *NeurIPS Efficient Natural Language and Speech Processing (ENLSP) Workshop*, 2024.
- [2] 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. In *International Conference on Machine Learning (ICML)*, 2024.
- [3] Edoardo D’Amico, Marco De Nadai, Praveen Chandar, Gustavo Penha, Ghazal Fazelnia, Bernd Huber, Mounia Lalmas, Paul N. Bennett, et al. Deploying semantic id-based generative retrieval for large-scale podcast discovery at spotify. *arXiv preprint arXiv:2603.17540*, 2026.
- [4] DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- [5] 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.
- [6] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *International Conference on Machine Learning (ICML)*, 2024.
- [7] Ruining He, Lukasz Heldt, Lichan Hong, et al. PLUM: Adapting pre-trained language models for industrial-scale generative recommendations. *arXiv preprint arXiv:2510.07784*, 2025.
- [8] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.
- [9] 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 (KDD)*, 2025.
- [10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [11] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. <https://kellerjordan.github.io/posts/muon/>, 2024.
- [12] Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew W. Fitzgibbon. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. In *International Conference on Learning Representations*, 2023.
- [13] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.
- [14] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. 2025.
- [15] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

- [16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
- [17] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [18] Anima Singh, Trung Vu, Nikhil Mehta, Raghunandan Keshavan, Maheswaran Sathiamoorthy, Yilin Zheng, Lichan Hong, Lukasz Heldt, Li Wei, Devansh Tandon, Ed H. Chi, and Xinyang Yi. Better generalization with semantic ids: A case study in ranking for recommendations. *arXiv preprint arXiv:2306.08121*, 2024.
- [19] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*. <https://crfm.stanford.edu/2023/03/13/alpaca.html>, 3(6):7, 2023.
- [20] Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Garg, Yi-Hsuan Sung, Ming-Wei Wang, and Donald Metzler. Transformer memory as a differentiable search index. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [21] Yejing Wang, Shengyu Zhou, Jinyu Lu, Ziwei Liu, Langming Liu, Maolin Wang, Wenlin Zhang, Feng Li, Wenbo Su, Pengjie Wang, Jian Xu, and Xiangyu Zhao. NEZHA: A zero-sacrifice and hyperspeed decoding architecture for generative recommendations. In *Proceedings of the ACM Web Conference (WWW)*, 2026.
- [22] Erik Wijmans, Brody Huval, Alexander Hertzberg, Vladlen Koltun, and Philipp Krähenbühl. Cut your losses in large-vocabulary language models. *arXiv preprint arXiv:2411.09009*, 2024.
- [23] Shiguang Wu, Zhaochun Wei, Zhumin Chen, Pengjie Ren, and Maarten de Rijke. Constrained auto-regressive decoding constrains generative retrieval. *arXiv preprint arXiv:2504.09935*, 2025.
- [24] An Yang and Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- [25] Daniel Zagyva, Emmanouil Stergiadis, Laurens van der Maas, Aleksandra Dokic, Eran Fainman, Ilya Gusev, and Moran Beladev. Speed without sacrifice: Fine-tuning language models with medusa and knowledge distillation in travel applications. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL): Industry Track*, 2025.
- [26] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *International Conference on Machine Learning (ICML)*, 2024.
- [27] Tianyu Zhan, Kairui Fu, Chengfei Lv, Zheqi Lv, and Shengyu Zhang. Semantic trimming and auxiliary multi-step prediction for generative recommendation. *arXiv preprint arXiv:2604.05329*, 2026.
- [28] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12), 2023.
- [29] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [30] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.