OnePiece: The Great Route to Generative Recommendation —A Case Study from Tencent Algorithm Competition

Abstract

While Generative Recommendation is gaining traction for its potential to adhere to Scaling Laws, it faces critical challenges in ranking precision and inference efficiency. Purely probabilistic generation (e.g., beam search) struggles to provide the precise ranking required by industrial systems and is computationally expensive. To address this, we propose a Unified Scalable Cascade Framework. This framework synergizes generative and discriminative tasks within a shared, high-capacity HSTU+MoE backbone. Specifically, we first leverage the generative head (predicting SIDs) for efficient **Semantic Pruning**; subsequently, we utilize the next-item embedding, predicted by the **same backbone**, to perform high-precision **Vector Ranking** on the candidate set. A **core empirical finding** of our work is that both the discriminative (InfoNCE) and generative (SID) losses strictly adhere to power-law Scaling Laws (R2>0.9) within our unified architecture. This proves our framework allows recall and ranking to benefit simultaneously from model scaling. Furthermore, our designed Collaborative Tokenizer reduces hash collisions on Top-50 items to 7.86%. Experiments on TencentGR-10M dataset demonstrate the superiority of our method, and we list the 9th rank in the competition leaderboard.

1 Introduction

Background. Scaling Laws [1] have emerged as a core driver of breakthroughs in domains such as Natural Language Processing (NLP) and Computer Vision (CV). However, conventional recommender system paradigms, particularly those based on discriminative models, have struggled to replicate similar success. A typical observation is that despite substantial efforts from both industry and academia to devise increasingly sophisticated architectures—evolving from early models like DIN [10] to MIMN [6] and SIM [7]—the performance enhancements exhibit a clear trend of diminishing returns. This phenomenon of performance saturation strongly suggests that merely increasing the complexity of discriminative models may be approaching a fundamental paradigm ceiling. In stark contrast to the dilemma faced by discriminative models, generative models, epitomized by Large Language Models (LLMs), have demonstrated remarkable scalability. It shifts the research focus from the traditional discriminative task of "predicting click-through rates" to exploring the generative task of "next interactive item"[5].

Related work. Recommender systems have long been dominated by discriminative models. Over the past decade, this paradigm has dominated the landscape, evolving from early Matrix Factorization to deep architectures like DeepFM [4], and subsequently to attention-based models such as DIN [10] and SIM [7] for modeling user behavior sequences. Inspired by the remarkable scaling effects of LLMs in NLP, pioneering works like P5 [3] unify recommendation into a text-based sequence generation task. Subsequent approaches, such as OneRec [2], further reformulate recommendation as an autoregressive generation process, validating continuous performance growth with larger parameter scales. These advancements suggest that generative architectures adhering to Scaling Laws are poised to become the core paradigm for next-generation recommender systems.

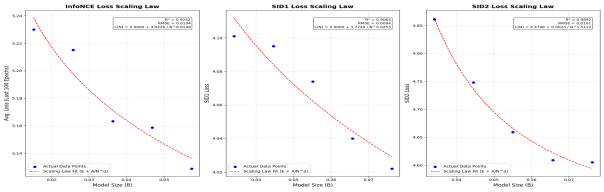

Figure 1: The InfoNCE/Semantic ID prediction Scaling Laws we tested in Tencent competition.

Motivation. While generative recommendation (e.g., OneRec [2]) has successfully adopted Scaling Laws to capture open-ended user interests, its reliance on probabilistic sampling (e.g., beam search) inherently limits both ranking precision and inference speed. In contrast, traditional vector-based retrieval offers efficient and precise ranking but struggles to match the continuous scalability of generative backbones due to capacity constraints. This dichotomy presents a clear research gap: there is a need for a framework that retains the massive capacity of generative backbones while enforcing the precision and efficiency of vector-based retrieval mechanisms suitable for industrial deployment.

Our Work. To bridge this gap, we propose a unified scalable cascade framework. Our design is grounded in a critical empirical observation (as illustrated in Figure 1): within a unified backbone, the discriminative objective (InfoNCE) exhibits robust scaling behavior highly consistent with the generative objective (SID Loss), with both strictly adhering to power-law distributions ($R^2 > 0.9$). Guided by this insight, we optimize both tasks within a shared **HSTU+MoE backbone** via a **coarse-to-fine strategy**:

(1) **Stage 1: Generative Pruning.** We use the backbone's generative capacity to rapid prune the search space via beam search (retrieving Top-K SIDs). This step exploits the scaling laws of generative loss (SID1/SID2) to capture broad semantic structures.
(2) **Stage 2: Vector-Based Ranking.** We utilize the **same backbone** to output the user's vector representation for precise dot-product ranking. Driven by our observation that vector representation quality scales predictably with model size (InfoNCE scaling), this step effectively corrects generative hallucinations and ensures high-precision ranking as the model expands.

Contributions. The main contributions of our work are:

- **Unified Scaling Paradigm:** We establish a unified architecture where semantic generation and vector retrieval share a single scalable backbone, allowing both tasks to benefit simultaneously from Scaling Laws.

- **Collaborative Tokenizer:** By fusing multi-modal and structural features, we design a residual quantization scheme that significantly reduces hash collisions (7.86% for Top-50 items), providing high-resolution targets for the generative model.
- **Efficient Cascade Inference:** The proposed cascade inference mechanism combines generative pruning with vector refinement, addressing the ranking imprecision of pure generative models while maintaining efficient system response latency.

2 Preliminary

2.1 Problem Statement

Our task is to predict the next item i_{t+1} a user $u \in \mathcal{U}$ is most likely to interact with, given their historical interaction sequence $S_u = (i_1, i_2, \ldots, i_t)$, where $i \in \mathcal{I}$ is an item from the full corpus. To enable scalable generative recommendation, we first define a semantic tokenizer $\mathcal{T}$ that maps each item i to a set of hierarchical semantic codes (SIDs), $\mathbf{c}_i = (c_i^1, c_i^2)$. Our cascade framework decomposes this problem into two sub-tasks:

- **Generative Pruning:** Given S_u, generate a high-quality candidate set of Top-K SIDs.
- **Vector Ranking:** Given S_u, predict a next-item embedding $\mathbf{h}_{t+1}$, and use this embedding to perform precise vector-based retrieval and ranking on the candidate set.

2.2 Data Challenges

Table 1: Item Coverage Statistics

Member	Covered Items	Total Items	Coverage Rate
81	16693655	19099627	87.403%
82	16732879	19099627	87.600%
83	16733606	19099627	87.612%
85	7534474	19099627	39.448%
86	7161580	19099627	37.495%

The raw multi-modal embeddings for item tokenization exhibit severe coverage inconsistencies. As shown in Table 1, some modalities (e.g., 82, 85) cover less than 40% of the item corpus. This sparsity renders any single modality unreliable for tokenization, as it would cause massive **out-of-vocabulary (OOV)** issues. Addressing this data sparsity via a robust fusion strategy is therefore a critical prerequisite for building our generative model.

3 Methodology

The architecture of our framework (Figure 2) consists of three integral components: a **Semantic Tokenizer** for item discretization, a scalable **HSTU+MoE Sequence Encoder**, and a **Hybrid Objective** mechanism.

3.1 Semantic Tokenizer

We employ **Residual K-means** to quantize item i into $\mathbf{c}_i = (c_i^1, c_i^2)$ with codebooks of size 16, 384. Since single-modality embeddings suffer from limited coverage (leading to high collisions), we propose a **Collaborative** strategy to fuse multi-modal signals. This

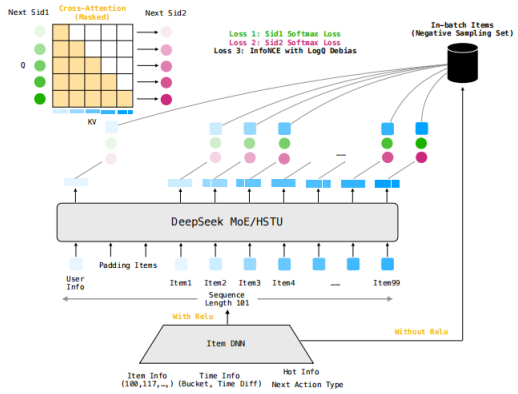

Figure 2: The training workflow of OnePiece.

dense representation enables our **Greedy Re-assignment** mechanism (searching **Top-50** neighbors) to effectively resolve conflicts, achieving the highest resolution (17.6M unique pairs) and a minimal conflict rate of 7.86%, as detailed in Table 2.

3.2 Feature Engineer with ItemDNN

The **ItemDNN** module fuses multifaceted item features (static info, time info, hot info) via an MLP. As shown in Figure 2, we use a dual-path (ReLU and linear) design to preserve both non-linear and raw feature signals for the encoder.

3.3 Sequence Encoder

3.3.1 HSTU Backbone. We adopt **HSTU** [9] as the backbone, which, unlike standard $O(L^2)$ Transformers, achieves near-linear scaling to efficiently process long user sequences.

3.3.2 Sparse Mixture-of-Experts (MoE). To efficiently scale capacity, we also implement the **Sparse MoE** layers for a large parameters tuning with limited training resource (e.g., 7-card H20(96GB)). The layer dynamically routes input $\mathbf{x}$ to the top-k experts:

$$\text{MoE}(\mathbf{x}) = \sum_{j=1}^{N} g_j(\mathbf{x}) E_j(\mathbf{x}), \tag{1}$$

where $g(\cdot)$ is the gating function.

Load Balancing. To prevent expert collapse, we monitor the Gini coefficient of expert usage. As shown in Figure 3, the rapid descent to a low stable range (0.1–0.4) across all layers confirms effective **load balancing**, fully leveraging the expanded capacity.

3.4 Hybrid Training Objectives

Our framework optimizes a joint objective that unifies discriminative representation learning with generative semantic modeling.

3.4.1 Discriminative Matching with Correction. To align the sequence representation $\mathbf{h}_T$ with the target item $\mathbf{i}^+$, we employ the InfoNCE loss. Crucially, to mitigate the popularity bias inherent in

Table 2: Collision Analysis of Semantic Tokenizer. We compare single-modality embeddings against our Collaborative strategy, which is designed to resolve the low coverage and sparsity issues in individual modalities. 'Standard' denotes direct quantization, while 'Re-assigned' demonstrates the significant conflict reduction achieved by our Greedy Re-assignment strategy.

Modality	Training Loss		Standard Quantization			w/ Greedy Re-assignment		
	Loss1	Loss2	Conflicts	Conflict Rate	Unique Pairs	Conflicts	Conflict Rate	Unique Pairs
81	0.0184	0.0085	12.6M	75.88%	4.02M	2.15M	12.88%	14.54M
82	0.2722	0.2220	13.7M	81.95%	3.02M	1.37M	8.19%	15.36M
83	0.1653	0.1291	14.9M	89.43%	1.76M	7.11M	42.51%	9.61M
85	0.1545	0.1177	5.45M	72.34%	2.08M	0.49M	6.51%	7.04M
86	0.1608	0.1262	5.36M	74.98%	1.79M	0.59M	8.34%	6.56M
Collaborative	0.4179	0.2968	14.4M	75.46%	2.27M	1.49M	**7.86%**	**17.60M**

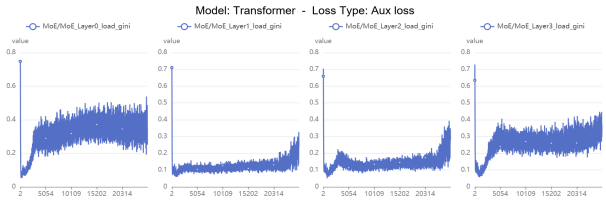

Figure 3: Gini coefficient over training steps for all MoE layers. The rapid drop and stabilization at low values (0.1–0.4) indicate successful expert load balancing.

in-batch sampling, we incorporate a **LogQ correction** [8]:

$$\mathcal{L}_{con} = -\log \frac{\exp(\mathbf{h}_T^\top \mathbf{e}_{i^+} - \log Q(i^+))}{\sum_{j \in \mathcal{B}} \exp(\mathbf{h}_T^\top \mathbf{e}_j - \log Q(j))}, \tag{2}$$

where $Q(i)$ denotes the estimated item popularity and $\mathbf{e}$ represents item embeddings.

3.4.2 Hierarchical Generative Modeling. We introduce a generative task to predict the discrete codes $\mathbf{c} = (c^1, c^2)$ of the target item. This is formulated as a hierarchical auto-regressive process:

$$p(\mathbf{c}|\mathbf{h}_T) = p(c^1|\mathbf{h}_T) \cdot p(c^2|c^1, \mathbf{h}_T). \tag{3}$$

Specifically, the first-level code c^1 is predicted by attending to the encoder output H using $\mathbf{h}_T$ as the query. For the second level, to enforce semantic consistency, we employ a **Teacher-Forcing** strategy: the query $\mathbf{q}_2$ is constructed by fusing $\mathbf{h}_T$ with the embedding of the first code, i.e., $\mathbf{q}_2 = \text{MLP}([\mathbf{h}_T; \mathbf{E}(c^1_{\text{gt}})])$, where $\mathbf{E}(c^1_{\text{gt}})$ denotes the vector representation of the ground-truth code c^1 from the codebook. Both probabilities are optimized via cross-entropy losses, denoted as $\mathcal{L}_{c^1}$ and $\mathcal{L}_{c^2}$.

3.4.3 Joint Optimization. The final objective balances the retrieval accuracy and semantic reconstruction:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{con} + \lambda_1 \mathcal{L}_{c^1} + \lambda_2 \mathcal{L}_{c^2}. \tag{4}$$

4 Experiments

4.1 Inference Setting

Our inference process, illustrated in Figure 4, co-utilizes the model's auto-regressive (SID) and dual-tower (InfoNCE) capabilities.

(1) **Stage 1: Auto-Regressive Candidate Generation (SID Beam Search).** For a given user, we first use the model's auto-regressive head to decode the most probable (sid1, sid2) sequences via Beam Search. We set the beam width to $B = 20$. We compute the joint log-probability $P(\text{sid1}, \text{sid2})$ and select the Top-K' SID pairs with the highest scores. Here, K' = 384. We then reverse-map these SIDs to retrieve a candidate items, denoted as C_{sid}.

(2) **Stage 2: Dual-Tower Re-ranking (InfoNCE Re-ranking).** We compute the user's query embedding q_u and retrieve the item embeddings for all items within the candidate set C_{sid} generated in Stage 1. Instead of using the beam search generation scores, we re-rank the items in C_{sid} using the cosine similarity scores $s(q_u, k_i)$ from the InfoNCE dual-tower model.

(3) **Stage 3: Filtering Strategy.** Before returning the final Top-10 results, two key filtering strategies are applied:
- **Historical Behavior Filtering:** We explicitly filter out any item that has already appeared in the user's historical interaction sequence by setting its re-ranking score to $-\infty$.
- **Cold-Start Item Filtering:** During the creation of the full candidate embedding matrix, all cold-start items (i.e., items not present in the training set) are discarded.

The final recommended list consists of the Top-10 items from C_{sid} after InfoNCE re-ranking and filtering.

4.2 InfoNCE with Different Layers

To validate the scaling properties of our model, we evaluated the performance of the InfoNCE ranking task across different model depths.

We fixed other key hyperparameters: the model is an HSTU architecture with a hidden dimension $D = 128$, 8 attention heads, and Pre-LayerNorm.The maximum sequence length was 101 with a per-device batch size of 512.We used pure bfloat16 for training.For optimization, we employed the Muon optimizer with a learning rate of 0.08, momentum of 0.95, and a cosine annealing schedule with 2000 warmup steps. The InfoNCE loss used cosine similarity and a fixed temperature $\tau = 0.02$. We then varied the number of layers in the sequence encoder, ranging from 8 to 40.

As shown in Table 3, we recorded the average training metrics for the last 100 batches after 1 epoch. The results clearly indicate a stable improvement in ranking performance (Hitrate and NDCG) as the number of layers increases, confirming our scaling hypothesis.

Table 3: Final Model Performance with Different Layer Configurations (Avg. of Last 100 Batches, 1-Epoch)

Layer Configuration	Model Size	Hitrate	NDCG
40 Layers	0.0573B	0.3219	0.1812
32 Layers	0.0468B	0.3153	0.1752
24 Layers	0.0363B	0.3027	0.1706
16 Layers	0.0257B	0.2936	0.1672
8 Layers	0.0152B	0.2770	0.1579

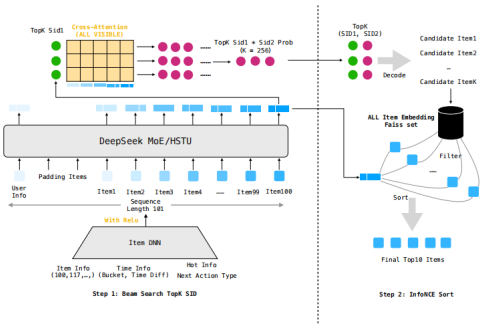

Figure 4: The hybrid inference pipeline of OnePiece(SID beam search + InfoNCE re-ranking).

Furthermore, we trained our optimal [40]-layer configuration for multiple epochs to observe its convergence and overfitting behavior. Table 4 details the training and validation metrics across epochs. While training metrics (like Hitrate) consistently rise, the validation metrics (Validation NDCG) peak at Epoch 6 (0.1064) before showing signs of slight overfitting.

Table 4: Performance Comparison across Epochs: Standard Validation vs. Hybrid Inference.

Epoch	Standard Valid.		Hybrid Inference	
	HR@10	NDCG@10	HR@10	NDCG@10
5	0.1885	0.1062	0.1307	0.0717
6	**0.1888**	**0.1064**	0.1313	0.0721
7	0.1885	0.1061	**0.1316**	**0.0723**
8	0.1879	0.1057	0.1313	0.0722

4.3 SID with Different Layers

We also evaluated the scaling properties of the auto-regressive SID generation task. We fixed key hyperparameters: the model is an HSTU architecture, $D = 128$, 8 attention heads, Pre-LN, AdamW optimizer with lr = 0.001, cosine annealing with 2000 warmup steps, and the InfoNCE loss was disabled.

As shown in Table 5, increasing model depth (from 4 to 20 layers) consistently improves the accuracy (HR@10) for SID1/SID2.

Table 5: HitRate@10 Performance with Different Layer Configurations (1 Epoch, Last 100 Batches Avg)

Layer Configuration	Model Size	SID1 HR@10	SID2 HR@10
20 Layer	0.0762B	0.5804	0.4419
16 Layers	0.0657B	0.5761	0.4345
12 Layers	0.0551B	0.5759	0.4280
8 Layers	0.0446B	0.5749	0.4149
4 Layers	0.0341B	0.5711	0.3979

We also trained the [20]-layer configuration for multiple epochs and applied our hybrid inference strategy (SID Beam Search + InfoNCE ranking) for evaluation. Table 4 shows the validation results. The performance peaks at Epoch 7 (NDCG@10 of 0.0723), suggesting the model achieves the best balance of SID generation and InfoNCE ranking capabilities at this point.

5 Conclusion

This work demonstrates that generative recall and discriminative ranking can be synergistically scaled within a unified backbone. Through our cascade framework, we empirically validated that both tasks adhere to power-law Scaling Laws, and effectively resolved high-frequency item collisions via a collaborative tokenizer. Future work will explore the behavior of Scaling Laws on larger-scale (e.g., billion-parameter) multi-modal foundation models and investigate end-to-end differentiable optimization strategies to further unify the training process.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv* (2023).

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

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

[4] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[5] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *IEEE international conference on data mining (ICDM)*.

[6] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2671–2679.

[7] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2685–2692.

[8] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM conference on recommender systems*. 269–277.

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

[10] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.