Rethinking Semantic ID Construction for Generative Recommendation: SimHash with Parallel Decoding and Semantic Alignment

Yuqing Liu[1,*] **Huiyuan Chen**[2,*,†] **Yibo Wang**[1] **Wooseong Yang**[1] **Philip S. Yu**[1]
[1]University of Illinois Chicago [2]Amazon
{yliu363,ywang633,wyang73,psyu}@uic.edu huiyuach@amazon.com

Abstract

Semantic ID-based generative recommendation represents each item as a sequence of discrete tokens, enabling structured modeling of item semantics. A critical challenge is constructing semantic IDs that are both semantically expressive and computationally efficient. While recent approaches favor complex learned quantization, simple hashing-based methods such as SimHash are widely regarded as fundamentally inferior. In this work, we challenge this consensus by showing that the apparent performance gap does not stem from inherent limitations of hashing, but rather from a structural mismatch with autoregressive decoding, coupled with the inevitable information loss during rigid discretization. Based on this insight, we propose FLASH, a two-stage framework that revitalizes training-free SimHash tokenization through parallel decoding and explicit semantic alignment. Despite its simplicity, FLASH achieves state-of-the-art performance across multiple datasets without requiring any tokenizer training, while exhibiting stronger generalization in cold-start scenarios. Notably, we demonstrate that semantic alignment acts as a universally effective mechanism across diverse paradigms. Our findings suggest that, with compatible decoding and semantic grounding, simple and efficient tokenizers can achieve performance comparable to complex learned counterparts in generative recommendation. Our project is available at https://github.com/KevinC2015/Flash.

1 Introduction

Generative recommendation has emerged as a promising paradigm that frames next-item prediction as a sequence generation problem [25, 45, 43], moving beyond traditional approaches that rely on learned atomic item embeddings and inner product-based retrieval [23, 40, 38, 7]. A key direction within this paradigm is semantic ID (SID)-based recommendation, where each item is represented as a sequence of discrete tokens derived from its content, and a sequence model is trained to generate the semantic ID of the next item [18, 47, 37, 17, 42]. This shift from traditional item IDs to composable semantic codes enables knowledge sharing across items, improves generalization to rare or unseen items, and eliminates the need for large item-specific embedding tables. A central question is how to construct semantic IDs that are both expressive and well aligned with the decoding process.

Among existing approaches, residual quantization (RQ) [24, 32] and product quantization (PQ) [21] are two dominant strategies for semantic ID construction. RQ-based methods, represented by TIGER [37], adopt a Residual-Quantized Variational Autoencoder (RQ-VAE) [24] to produce hierarchically dependent codes and rely on autoregressive (AR) decoding. However, they require training a dedicated quantizer and are typically limited to short semantic ID sequences (e.g., 4 tokens per

[*]Equal contribution. [†]Work done prior to joining Amazon.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

item) due to the high inference cost of AR decoding, restricting their ability to capture fine-grained item semantics. In contrast, PQ-based methods, represented by RPG [17], apply optimized product quantization (OPQ) [11] to generate unordered codes and adopt parallel decoding, thereby improving efficiency and enabling longer semantic IDs. Despite these differences, both approaches depend on learned quantizers, motivating the exploration of training-free alternatives such as Locality-Sensitive Hashing (LSH) [20, 19], which maps item embeddings to discrete codes via random projections. While training-free and efficient, this crude discretization inevitably discards fine-grained semantic information compared to optimized quantizers. Furthermore, TIGER's ablation study evaluates SimHash [4] and finds that it performs significantly worse than RQ-VAE [37]. This observation has led to the prevailing view that LSH-based tokenization is less effective in generative recommendation.

In this work, we revisit this conclusion and show that the performance gap can be attributed to a mismatch between tokenization structure and decoding paradigm, rather than the tokenization itself. Specifically, SimHash produces codes that do not admit a natural ordering and lack inherent dependencies across positions, whereas autoregressive decoding imposes sequential dependencies. Our preliminary study (Figure 1) shows that replacing autoregressive decoding with parallel decoding significantly improves the performance of LSH-based tokenization, indicating that such incompatibility introduces harmful inductive bias. However, this improvement alone does not fully close the gap with learned quantization methods, suggesting that additional factors beyond decoding compatibility play an important role.

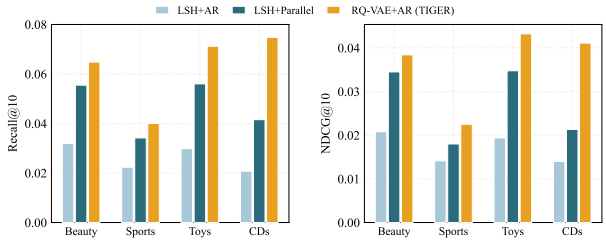


Figure 1: Preliminary study. Replacing autoregressive (AR) decoding with parallel decoding substantially improves LSH-based tokenization and narrows the gap with RQ-VAE. The remaining performance gap suggests that structural compatibility alone is insufficient, motivating our introduction of semantic grounding. See Appendix A.3 for detailed setup.

Building on this insight, we derive two design principles for effective semantic ID-based generative recommendation: (1) **structural compatibility**, which requires the decoding paradigm to align with the dependency structure induced by the tokenization method. Crucially, enforcing this compatibility via parallel decoding inherently removes the inference bottleneck and ID length constraint of autoregressive generation, enabling the use of substantially longer semantic ID sequences to boost representational capacity. (2) **semantic grounding**, which explicitly compensates for the loss of fine-grained information introduced by discrete tokenization, particularly for training-free methods.

Guided by these principles, we propose **FLASH** (**F**ast **L**LM-**A**ligned **S**im**H**ash for Generative Recommendation), a simple yet effective two-stage framework that integrates training-free SimHash tokenization with parallel decoding and LLM-based semantic alignment. In the first stage, items are mapped to semantic IDs using SimHash over pre-computed LLM text embeddings, requiring no tokenizer training. In the second stage, a sequence model encodes user interaction histories. To ensure structural compatibility with SimHash codes, it predicts the semantic ID of the target item using parallel decoding under a multi-token prediction (MTP) objective. To compensate for the semantic information loss introduced by hashing, we incorporate an alignment loss that anchors the learned representations to the original LLM embedding space during training.

Overall, our results demonstrate that effective semantic ID-based generative recommendation depends on both structural compatibility and semantic grounding, and that LSH-based methods can perform strongly with appropriate decoding and semantic supervision. Beyond this core insight, FLASH combines competitive recommendation performance with high efficiency through training-free tokenization, parallel inference, and support for long semantic IDs. Furthermore, FLASH exhibits strong generalization in cold-start settings. Our contributions are summarized as follows:

- By revisiting prior conclusions about the limited effectiveness of LSH-based tokenization, we derive two key design principles for semantic ID-based generative recommendation: structural compatibility and semantic grounding.
- We propose FLASH, a simple yet effective unified framework that instantiates these principles by integrating training-free SimHash tokenization, parallel decoding, and LLM-based semantic alignment.

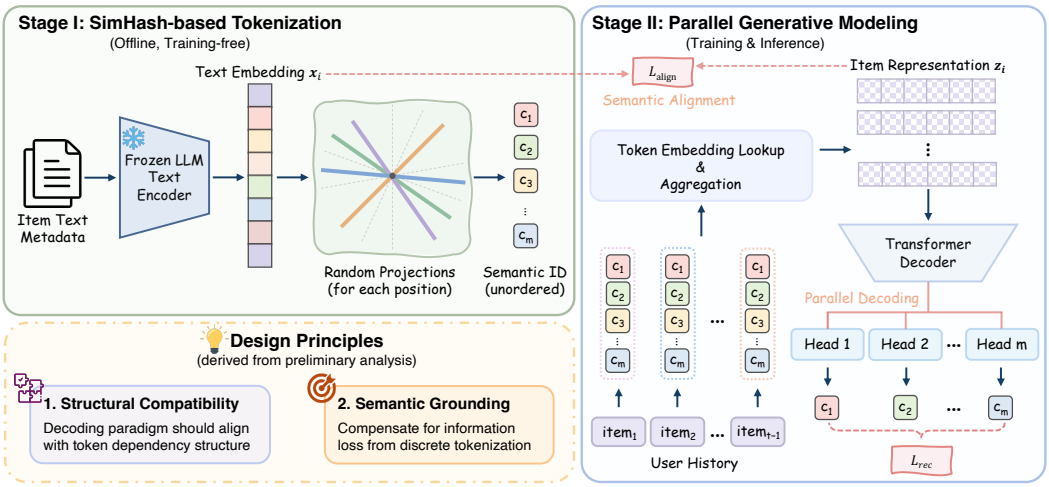


Figure 2: Overview of FLASH. Stage I constructs semantic IDs using SimHash over pre-computed LLM text embeddings in a training-free manner. Stage II performs parallel generative modeling with a multi-token prediction objective and incorporates an alignment loss for semantic grounding. The design is guided by structural compatibility and semantic grounding, aligning the decoding paradigm with the tokenization structure while compensating for information loss.

- We demonstrate that FLASH achieves competitive performance across multiple datasets without requiring tokenizer training, and exhibits superior generalization capabilities in cold-start scenarios. Furthermore, explicit semantic alignment serves as a general and effective regularization strategy across different quantization and decoding paradigms.

2 Methodology

We formulate the task as a sequential recommendation problem. Given a user's historical interaction sequence $\mathcal{S} = (i_1, \ldots, i_{t-1})$ over an item set $\mathcal{I}$, the goal is to predict the next item i_t. In semantic ID-based generative recommendation, each item $i \in \mathcal{I}$ is represented by a tuple of discrete tokens $c^{(i)} = (c_1^{(i)}, \ldots, c_m^{(i)})$, where m is the number of codebooks, and each token $c_j^{(i)} \in \mathcal{C}_j = \{1, \ldots, M\}$ is drawn from the j-th codebook. The recommendation task is thus reformulated as generating the semantic ID $(c_1^{(t)}, \ldots, c_m^{(t)})$ of the next item. Guided by the design principles of structural compatibility and semantic grounding, we propose FLASH, a two-stage framework consisting of: (1) training-free SimHash tokenization (§2.1), and (2) parallel generative modeling (§2.2). In particular, Stage II enforces structural compatibility through parallel decoding (§2.2.2) and achieves semantic grounding via an alignment objective (§2.2.3). An overview of the framework is shown in Figure 2.

2.1 Stage I: SimHash-based Semantic ID Construction

Semantic ID Construction. We construct semantic IDs by applying SimHash [4] to pre-computed LLM text embeddings. Each item i is first encoded into a dense representation $\mathbf{x}_i \in \mathbb{R}^{d_{\text{text}}}$ using a pre-trained LLM based on its textual metadata (e.g., title, description, category). These embeddings are computed once offline and kept fixed. For each code position $j \in \{1, \ldots, m\}$, we sample $B = \lceil \log_2 M \rceil$ random hyperplanes $\{\mathbf{r}_{j,b}\}_{b=1}^{B} \subset \mathbb{R}^{d_{\text{text}}}$ from a standard Gaussian distribution, where M is the codebook size. The code at position j for item i is obtained by projecting $\mathbf{x}_i$ onto these independent sets of hyperplanes and converting the resulting sign bits into an integer:

$$c_j^{(i)} = \left(\sum_{b=1}^{B} \mathbf{1}[\mathbf{x}_i \cdot \mathbf{r}_{j,b} > 0] \cdot 2^{b-1} \right) \bmod M + 1. \tag{1}$$

We restrict the codebook size M to be a power of 2, so that the modulo mapping distributes binary sequences more uniformly across the codebook. The modulo operation yields values in $\{0, \ldots, M-1\}$, and we shift the result by $+1$ to obtain 1-indexed semantic IDs. The hyperplanes are fixed and shared across all items, resulting in a training-free tokenization process.

Structural Properties and Design Implications. SimHash-based semantic IDs exhibit structural properties that inform model design. First, codes across positions are generated individually without residual dependencies. Moreover, SimHash imposes no intrinsic ordering over positions, making autoregressive decoding structurally mismatched. Finally, the discretization process introduces information loss despite preserving coarse-grained similarity. These properties motivate the design of FLASH in §2.2, where parallel decoding aligns with the unordered structure and semantic alignment compensates for information loss.

2.2 Stage II: Parallel Generative Modeling

2.2.1 Semantic ID-based Item Representation and Sequence Modeling

Item Representation. Given an item with semantic ID $(c_1, \ldots, c_m)$, we associate each position j with a learnable codebook embedding table $\mathbf{E}^{(j)} \in \mathbb{R}^{M \times d}$, where d is the hidden embedding dimension. Each code c_j is mapped to its corresponding embedding $\mathbf{e}_{c_j}^{(j)} \in \mathbb{R}^d$ via table lookup. The resulting code embeddings $\{\mathbf{e}_{c_j}^{(j)}\}_{j=1}^m$ are aggregated into a single item representation:

$$\mathbf{z} = \mathrm{Aggr}(\mathbf{e}_{c_1}^{(1)}, \ldots, \mathbf{e}_{c_m}^{(m)}), \tag{2}$$

where $\mathrm{Aggr}(\cdot)$ denotes an aggregation function. In this work, we instantiate $\mathrm{Aggr}(\cdot)$ as a linear projection over the concatenation of code embeddings, mapping the md-dimensional input to a d-dimensional representation. This design treats all code positions uniformly at the structural level, consistent with the unordered nature of SimHash-based SIDs. At the same time, it allows the model to learn distinct contributions from different codebook embeddings. Furthermore, to mitigate over-reliance on specific code positions and improve robustness, we apply position-level codebook dropout during training by randomly masking entire code positions (i.e., removing the corresponding code embeddings) and rescaling the remaining embeddings to preserve their expected magnitude.

Sequence Modeling. The sequence of aggregated item representations $\{\mathbf{z}_1, \ldots, \mathbf{z}_{t-1}\}$ (derived from Eq. 2) is encoded by a Transformer backbone, producing contextual hidden states $\{\mathbf{h}_i\}_{i=1}^{t-1}$, where each $\mathbf{h}_i \in \mathbb{R}^d$ summarizes the user interaction history up to position i. We use the final hidden state $\mathbf{h}_{t-1}$ as the representation of the user interaction history for next-item prediction.

2.2.2 Parallel Decoding for Structural Compatibility

Given an encoded user interaction history $\mathbf{h}_{t-1}$, the recommendation task is to model the conditional probability distribution over the semantic ID of the next item: $P((c_1^{(t)}, \ldots, c_m^{(t)}) \mid \mathbf{h}_{t-1})$.

A key design principle in generative recommendation is the alignment between the decoding paradigm and the dependency structure of semantic IDs. SimHash-based semantic IDs do not impose sequential dependencies or an intrinsic ordering across code positions. Motivated by this unordered structure, we adopt a factorized formulation that enables all m code positions to be predicted simultaneously:

$$P((c_1^{(t)}, \ldots, c_m^{(t)}) \mid \mathbf{h}_{t-1}) = \prod_{j=1}^{m} p_j(c_j^{(t)} \mid \mathbf{h}_{t-1}). \tag{3}$$

We parameterize each factor using a dedicated prediction head for code position j, yielding m parallel heads. This corresponds to a multi-token prediction (MTP) setting, where all code positions are predicted in parallel. Concretely, each prediction head produces a probability distribution over M codes at position j via dot-product similarity with a learnable temperature τ_j:

$$p_j(c \mid \mathbf{h}_{t-1}) = \frac{\exp\Big(f_j(\mathbf{h}_{t-1})^\top \cdot \mathbf{e}_c^{(j)} \,/\, \tau_j\Big)}{\sum_{c' \in \mathcal{C}_j} \exp\Big(f_j(\mathbf{h}_{t-1})^\top \cdot \mathbf{e}_{c'}^{(j)} \,/\, \tau_j\Big)}, \tag{4}$$

where each $f_j(\cdot)$ is a lightweight MLP projection head [8, 2] that maps contextual representations to the codebook space $\mathbf{E}^{(j)}$, shared between input and output. Here $c \in \mathcal{C}_j$ denotes a candidate code, and the target probability is obtained by evaluating at $c = c_j^{(t)}$. Given a target item with SID $(c_1^{(t)}, \ldots, c_m^{(t)})$, the recommendation loss is defined as the average cross-entropy across all m heads:

$$\mathcal{L}_{\text{rec}} = -\frac{1}{m} \sum_{j=1}^{m} \log p_j(c_j^{(t)} \mid \mathbf{h}_{t-1}). \tag{5}$$

This MTP objective jointly optimizes all code positions under the factorized formulation. Moreover, parallel decoding supports longer semantic ID sequences by avoiding autoregressive constraints.

2.2.3 Semantic Alignment for Information Loss

While SimHash preserves coarse-grained similarity through its locality-sensitive property, the discretization process inevitably discards fine-grained semantic information. Unlike RQ-VAE, which implicitly anchors item representations through its reconstruction objective, SimHash lacks the semantic grounding signal. To compensate for this loss, we introduce a semantic alignment objective that regularizes the input representation space toward the original LLM embedding space, thereby preserving semantic consistency. Specifically, we introduce a trainable linear projector $\phi\colon \mathbb{R}^{d_{\text{text}}} \to \mathbb{R}^{d}$ and align each input item representation in the sequence with its corresponding (frozen) LLM text embedding by minimizing the negative cosine similarity, providing an explicit semantic anchor:

$$\mathcal{L}_{\text{align}} = -\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \cos(\mathbf{z}_i,\ \phi(\mathbf{x}_i))\,. \tag{6}$$

The overall training objective can be represented as

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_{\text{align}}\, \mathcal{L}_{\text{align}}, \tag{7}$$

where λ_{align} is the alignment loss weight. The alignment loss improves semantic fidelity, while the recommendation loss supervises the predictive distribution over next-item SIDs, providing complementary signals for representation quality and predictive accuracy. Overall, the design enforces structural compatibility at the architectural level and semantic grounding at the representation level.

2.3 Efficient Inference

At inference, given the final hidden state $\mathbf{h}_{t-1}$, we rank all candidate items without enumerating all possible SID combinations, and return the top-k items by score as predictions.

Parallel decoding enables item scoring by aggregating position-wise probabilities, with per-head probabilities computed once and cached. Specifically, we first compute the probability distribution over all M codebook entries at each position, following the cached logit strategy in [17]:

$$\mathbf{q}^{(j)} = \operatorname{softmax}\Big(\mathbf{E}^{(j)} \cdot f_j(\mathbf{h}_{t-1}) \,/\, \tau_j\Big) \in \mathbb{R}^{M}. \tag{8}$$

This decouples the computation from the number of items N, as the per-head probabilities are computed once and reused for all items, resulting in $O(mMd)$ time complexity. Given these cached probabilities, the score of a candidate item can be computed by summing the corresponding log-probabilities across code positions. Specifically, for an item i with semantic ID $(c_1, \ldots, c_m)$, its score is given by

$$\text{score}(i) = \sum_{j=1}^{m} \log \mathbf{q}^{(j)}_{c_j}. \tag{9}$$

Consequently, inference reduces to an $O(Nm)$ gather-and-sum operation, leading to a total cost of $O(mMd + Nm)$, compared to $O(Nmd)$ for naive per-item similarity evaluation.

3 Experiments

3.1 Experimental Setup

Datasets. We evaluate FLASH on four widely used categories from the Amazon Product Reviews dataset [34]: "Beauty", "Toys and Games" (Toys), "Sports and Outdoors" (Sports), and "CDs and Vinyl" (CDs). We follow standard preprocessing and evaluation protocols commonly used in prior work [37, 17]. Detailed dataset statistics and preprocessing steps are provided in Appendix A.1.

Baselines. We compare FLASH against three categories of baselines: (1) *Item ID-based methods*: Caser [41], GRU4Rec [15], HGN [33], BERT4Rec [40], SASRec [23], and S^3-Rec [48], which represent items using unique identifiers and learn embeddings from interaction data; (2) *RQ-based semantic ID*: TIGER [37], LETTER [42], and DIGER [10], which construct semantic IDs via learned

Table 1: Overall performance comparison among baselines and FLASH across four datasets. The best result is in **bold** and the second-best result is underlined. FLASH achieves statistically significant improvements over the best baseline under a paired t-test ($p < 0.05$).

Model	Beauty				Toys and Games				Sports and Outdoors				CDs and Vinyl			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
Item ID-based																
Caser	0.0205	0.0131	0.0347	0.0176	0.0166	0.0107	0.0270	0.0141	0.0116	0.0072	0.0194	0.0097	0.0116	0.0073	0.0205	0.0101
GRU4Rec	0.0164	0.0099	0.0283	0.0137	0.0097	0.0059	0.0176	0.0084	0.0129	0.0086	0.0204	0.0110	0.0195	0.0120	0.0353	0.0171
HGN	0.0325	0.0206	0.0512	0.0266	0.0321	0.0221	0.0497	0.0277	0.0189	0.0120	0.0313	0.0159	0.0259	0.0153	0.0467	0.0220
BERT4Rec	0.0203	0.0124	0.0347	0.0170	0.0116	0.0071	0.0203	0.0099	0.0115	0.0075	0.0191	0.0099	0.0326	0.0201	0.0547	0.0271
SASRec	0.0387	0.0249	0.0605	0.0318	0.0463	0.0306	0.0675	0.0374	0.0233	0.0154	0.0350	0.0192	0.0351	0.0177	0.0619	0.0263
S^3-Rec	0.0387	0.0244	0.0647	0.0327	0.0443	0.0294	0.0700	0.0376	0.0251	0.0161	0.0385	0.0204	0.0213	0.0130	0.0375	0.0182
RQ-based Semantic ID																
TIGER	0.0454	0.0321	0.0648	0.0384	0.0521	0.0371	0.0712	0.0432	0.0264	0.0181	0.0400	0.0225	0.0492	0.0329	0.0748	0.0411
LETTER	0.0489	0.0364	0.0689	0.0402	0.0544	0.0379	0.0727	0.0437	0.0293	0.0187	0.0420	0.0232	0.0481	0.0314	0.0723	0.0398
DIGER	0.0503	0.0370	0.0692	0.0421	0.0540	0.0383	0.0719	0.0442	0.0284	0.0192	0.0388	0.0241	0.0470	0.0312	0.0717	0.0369
PQ-based Semantic ID																
RecJPQ	0.0311	0.0167	0.0482	0.0222	0.0331	0.0182	0.0484	0.0231	0.0141	0.0076	0.0220	0.0102	0.0075	0.0046	0.0138	0.0066
VQ-Rec	0.0457	0.0317	0.0664	0.0383	0.0497	0.0346	0.0737	0.0423	0.0208	0.0144	0.0300	0.0173	0.0352	0.0238	0.0520	0.0292
RPG	0.0550	0.0381	0.0809	0.0464	0.0592	0.0401	0.0869	0.0490	0.0314	0.0216	0.0463	0.0263	0.0498	0.0338	0.0735	0.0415
LSH-based Semantic ID																
FLASH	**0.0576**	**0.0402**	**0.0832**	**0.0489**	**0.0650**	**0.0459**	**0.0914**	**0.0544**	**0.0337**	**0.0237**	**0.0487**	**0.0285**	**0.0536**	**0.0370**	**0.0769**	**0.0446**
Improv.	*+4.73%*	*+5.51%*	*+2.84%*	*+5.39%*	*+9.80%*	*+14.46%*	*+5.18%*	*+11.02%*	*+7.32%*	*+9.72%*	*+5.18%*	*+8.37%*	*+7.63%*	*+9.47%*	*+2.81%*	*+7.47%*

residual quantization and model them with autoregressive decoding; and (3) *PQ-based semantic ID*: RecJPQ [36], VQ-Rec [16], and RPG [17], which derive semantic IDs using product quantization and differ in modeling strategies. See Appendix A.2 for detailed descriptions of baselines.

Evaluation. We evaluate recommendation performance using two widely adopted metrics, Recall (R@K) and NDCG (N@K). Following prior work [37], we set $K = 5, 10$ by default.

Implementation Details. We adopt most baseline results from TIGER [37] and RPG [17], and implement the remaining methods using their publicly available source code. For all methods, hyperparameters are carefully tuned following the original papers for fair comparison. For FLASH, we perform a grid search over the number of codebooks $m \in \{4, 8, 16, 32, 64\}$, codebook size $M \in \{128, 256, 512, 1024\}$, and alignment loss weight $\lambda_{\text{align}} \in [0, 1]$. The learning rate is tuned within $[0.001, 0.01]$, with a fixed position-level codebook dropout ratio of 0.2. Models are trained with a batch size of 256 for up to 300 epochs, with early stopping if validation performance does not improve for 20 consecutive epochs. Additional details are provided in Appendix A.3.

3.2 Overall Performance

We compare FLASH with three categories of baselines across four public datasets and report the overall performance in Table 1. FLASH consistently achieves the best performance on all datasets and metrics, with relative improvements on N@5 ranging from 5.51% to 14.46%, demonstrating its effectiveness and robustness.

Compared to item ID-based methods, FLASH achieves substantial improvements across all datasets, suggesting that representations grounded in item semantic information provide more informative signals than discrete item identifiers for recommendation. Within the semantic ID-based paradigm, FLASH outperforms both RQ-based and PQ-based methods. For example, compared to RPG, the strongest PQ-based baseline, FLASH achieves gains of 5.18% on R@10 and 11.02% on N@10 on the Toys dataset. This result is particularly notable given that FLASH employs a training-free tokenization scheme, in contrast to learned quantization approaches that require additional training objectives and model parameters. FLASH is also more efficient than TIGER due to its training-free tokenization and parallel prediction (see Figure 3 and Appendix B.1).

The superior performance of FLASH can be attributed to three key design choices. First, SimHash enables efficient semantic ID construction while preserving coarse-grained semantic similarity from LLM-based embeddings. Second, parallel decoding aligns naturally with the unordered structure of LSH-based tokens, avoiding the mismatch introduced by autoregressive decoding. Third, the semantic alignment objective compensates for information loss introduced by discrete hashing, retaining fine-grained semantic signals. Together, these findings highlight the importance of structural compatibility and semantic grounding in semantic ID-based generative recommendation.

Table 2: Effect of tokenization methods under unified parallel decoding and semantic alignment. T denotes wall-clock tokenization time (s). †RQ-VAE and RQ-KMeans are measured on GPU; others on CPU. Best recommendation results are in **bold** and second-best are underlined.

	Beauty			Toys			Sports			CDs		
Tokenizer	R@10	N@10	T (s) ↓	R@10	N@10	T (s) ↓	R@10	N@10	T (s) ↓	R@10	N@10	T (s) ↓
Random	0.0432	0.0261	0.0177	0.0612	0.0324	0.0144	0.0361	0.0198	0.0220	0.0197	0.0098	0.0353
RQ-VAE†	0.0832	0.0483	67.08	0.0856	0.0481	77.41	0.0466	0.0278	73.65	0.0751	0.0438	147.26
RQ-KMeans†	0.0812	0.0462	8.19	0.0861	0.0490	12.07	0.0457	0.0263	20.90	0.0760	0.0441	132.87
OPQ	**0.0851**	**0.0498**	292.89	0.0905	0.0523	111.08	0.0460	0.0265	151.90	**0.0782**	**0.0455**	162.98
SimHash	0.0832	0.0489	0.0556	**0.0914**	**0.0544**	0.0869	**0.0487**	**0.0285**	0.1340	0.0769	0.0446	0.4608

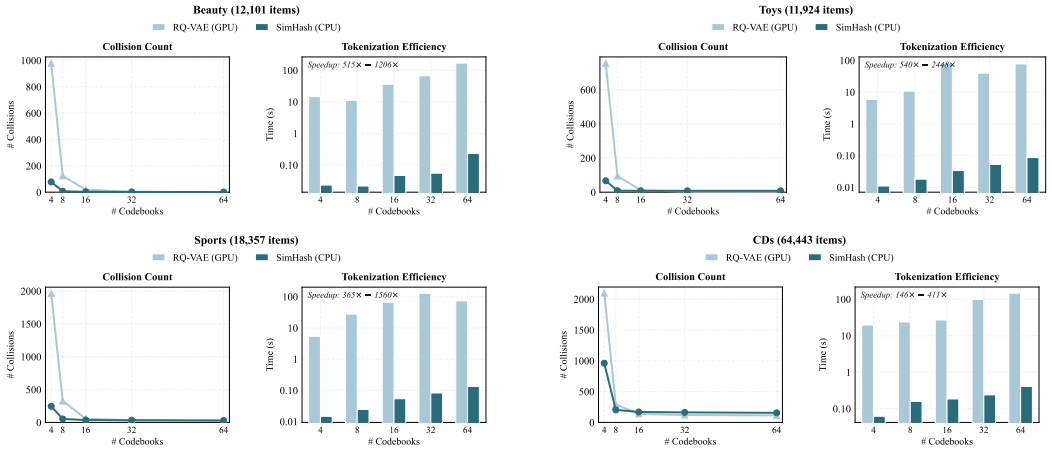


Figure 3: Collision statistics and tokenization time for SimHash and RQ-VAE across different numbers of codebooks with a fixed codebook size of 256.

3.3 Ablation Study

3.3.1 Analysis of Structural Compatibility

To investigate the impact of structural compatibility, we replace SimHash with alternative tokenization methods while keeping the downstream sequence modeling pipeline fixed, following the dataset-specific configuration of FLASH reported in Table 4. Specifically, we consider **Random** code assignment with no semantic basis and three learned quantizers: **RQ-VAE**, **RQ-KMeans**, and **OPQ**. The results are reported in Table 2.

SimHash achieves competitive or superior performance compared to learned quantization methods across most datasets, despite requiring no tokenizer training. Random assignment performs substantially worse, with over 40% lower N@10 than SimHash on average, showing that structural compatibility alone is insufficient without meaningful semantic information. Among learned tokenizers, OPQ generally outperforms the residual methods under the same downstream modeling pipeline, further suggesting the importance of compatibility between the tokenization structure and the decoding paradigm. However, the relatively small performance gaps between learned tokenizers and SimHash indicate that parallel decoding and semantic grounding can partially mitigate differences in tokenization. Notably, SimHash substantially reduces tokenization cost, requiring less than 0.5 seconds on CPU across all datasets, compared with tens to hundreds of seconds for learned quantizers.

To further evaluate the properties of SimHash beyond recommendation performance, we compare its collision behavior and tokenization efficiency with RQ-VAE across different codebook configurations. As shown in Figure 3, SimHash exhibits limited collisions relative to dataset size, with collision counts decreasing as the number of codebooks increases. Moreover, for short semantic IDs, SimHash yields far fewer collisions than RQ-VAE (e.g., 78 vs. 978 on Beauty at $m = 4$). This further highlights the benefit of structural compatibility, as parallel decoding enables longer semantic IDs with negligible collisions, whereas autoregressive decoding is typically constrained to shorter semantic IDs with higher collision rates. SimHash also achieves a speedup of up to 2,448$\times$ over RQ-VAE in tokenization, despite running on a CPU rather than a GPU. These results show that SimHash retains the advantages of simplicity and efficiency without sacrificing code distinguishability.

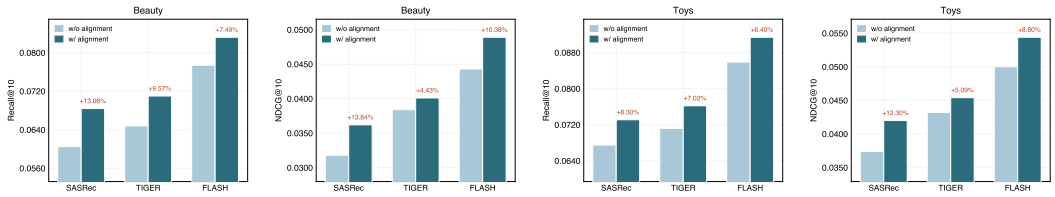


Figure 4: Effect of semantic alignment across models. "w/" and "w/o" denote training with and without the alignment loss, respectively. For each model, both variants share the same backbone and training settings. Further details are provided in Appendix A.3.

3.3.2 Analysis of Semantic Grounding

To evaluate the effect of semantic grounding, we compare model performance with and without semantic alignment while preserving the original backbone architecture of each model. For clarity, we present results on the Beauty and Toys datasets in the main text, while results on the remaining datasets are provided in Appendix B.2 and exhibit similar trends.

As shown in Figure 4, incorporating the alignment loss consistently improves performance across models and metrics. This demonstrates that explicitly anchoring item representations to the original LLM embedding space effectively compensates for the loss of fine-grained semantic information introduced by hashing. Moreover, this trend generalizes across different models, suggesting that semantic alignment serves as a general and effective mechanism for enhancing representation quality. For example, on the Beauty dataset, alignment improves R@10 by 13.06%, 9.57%, and 7.49% for SASRec, TIGER, and FLASH, respectively. Overall, these results highlight the importance of semantic grounding in mitigating information loss introduced by discretization, complementing structural compatibility in determining model effectiveness. Further qualitative analysis of the learned code embeddings is provided in Appendix B.3.

Taken together, these results demonstrate that structural compatibility and semantic grounding are both essential, while decoding capacity further contributes to the remaining performance gap.

3.4 Hyperparameter Analysis

We analyze the impact of key hyperparameters on the Beauty dataset for clarity. Results on the remaining datasets are provided in Appendix B.4, which show similar trends.

Alignment loss weight λ_{align}. The alignment weight controls the strength of semantic grounding by balancing the alignment loss against the recommendation objective. We vary λ_{align} in the range $[0, 1]$. As shown in Figure 5 (left), introducing a moderate alignment weight (e.g., 0.1–0.2) consistently improves performance, validating the effectiveness of semantic grounding in compensating for the information loss introduced by discrete tokenization. However, excessively large alignment weights (e.g., $\lambda_{\text{align}} \geq 0.5$) fail to provide further improvements and may even lead to performance degradation on other datasets. This suggests that overly strong alignment may dominate the training objective and weaken the contribution of the recommendation signal.

Number of codebooks m. This parameter determines the length of the semantic ID by specifying the number of code positions used to represent each item, thereby affecting the expressiveness of the representation. We experiment with $m \in \{4, 8, 16, 32, 64\}$ while fixing the codebook size. Figure 5 (middle) shows that performance improves as m increases, since longer semantic IDs provide greater representational capacity to capture fine-grained item semantics. The gains are substantial when increasing m from 4 to 32, while further increasing m to 64 yields only marginal improvements. On other datasets, the performance peak occurs at different values of m, indicating that the optimal number of codebooks depends on the dataset scale and interaction sparsity, reflecting a trade-off between representation capacity and modeling complexity.

Codebook size M. The codebook size defines the number of discrete entries available at each code position, controlling the granularity of the semantic representation. We evaluate $M \in \{128, 256, 512, 1024\}$ while fixing the number of codebooks. From Figure 5 (right), we observe that increasing M leads to modest performance improvements, as larger codebooks allow

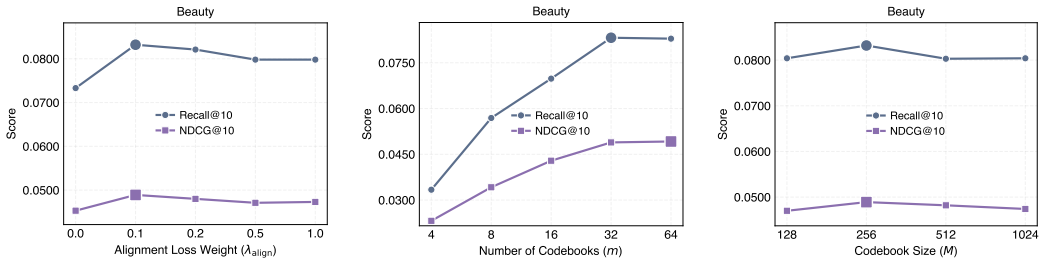


Figure 5: Performance of FLASH over different hyperparameters on the Beauty dataset.

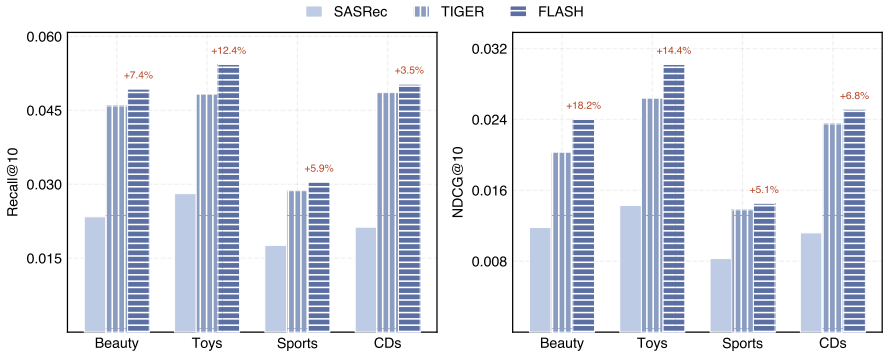


Figure 6: Performance comparison under the cold-start setting, where target items are unseen during training. FLASH achieves the best performance across all datasets. Percentages denote relative improvements of FLASH over TIGER.

for finer-grained distinctions between items. Performance improves from $M = 128$ to $M = 256$, after which it slightly declines. This suggests that while larger codebooks increase representational granularity, excessively large M makes learning more difficult due to data sparsity. Larger codebooks also require higher memory and computational costs, making moderate M more practical. Compared to increasing the number of codebooks m, which directly expands representational capacity through longer semantic ID sequences, the impact of M is relatively limited. This indicates that capacity is more effectively improved by increasing sequence length rather than per-position granularity.

3.5 Cold-Start Generalization

To evaluate the generalization ability of FLASH, we adopt a cold-start setting following TIGER [37], where a subset of target items is held out from training and used only for evaluation. Full details are provided in Appendix A.4. From Figure 6, we can observe that FLASH consistently outperforms all baselines across all datasets. Compared to SASRec, the performance gap highlights the advantage of semantic ID-based modeling over conventional item ID-based methods in handling unseen items. Compared to TIGER, which employs learned quantization, FLASH still achieves better performance. This improvement can be attributed to two key factors. First, the training-free SimHash tokenization preserves coarse-grained semantic similarity derived from LLM-based text embeddings, enabling stable and meaningful representations for unseen items without relying on interaction data or learned codebooks. Second, the semantic alignment objective further anchors item representations to the original embedding space, improving generalization under distribution shift. Overall, the superior performance of FLASH demonstrates its ability to generalize to unseen items and highlights the importance of semantic grounding for robust recommendation in cold-start scenarios.

4 Related Work

LLM-based Generative Recommendation. Traditional sequential recommendation relies on item ID-based paradigms, where each item is represented as a unique identifier and predicted via embedding matching [23, 15, 40, 41, 33, 14, 6, 3, 27, 5]. Although effective, such approaches lack semantic

compositionality and struggle to generalize to new or infrequent items. With the advancement of large language models (LLMs), generative recommendation has emerged as a promising paradigm that formulates recommendation as sequence generation [45, 13, 26, 29, 30, 22]. Some methods [12, 46, 1] treat recommendation as a text generation task, leveraging pre-trained semantic knowledge. However, operating directly in natural-language space without explicit discrete item representations makes efficient large-scale retrieval challenging. Semantic ID-based methods address this limitation by representing items as discrete token sequences from a shared vocabulary [37, 17, 9], enabling compositional representations and scalable retrieval. Despite their success, existing approaches differ significantly in how semantic IDs are constructed and decoded.

Semantic ID-based Generative Recommendation. Existing approaches to semantic ID construction can be broadly categorized into two dominant quantization-based strategies. RQ-based methods [37, 42, 10, 28, 32] produce hierarchically dependent codes typically paired with autoregressive decoding. For example, TIGER [37] employs RQ-VAE, which iteratively quantizes item embedding residuals across multiple codebooks to construct a multi-level tuple of discrete codewords as the item's semantic ID. In contrast, PQ-based methods [17, 16, 36] generate unordered codes compatible with parallel prediction. Despite these differences, both approaches rely on data-dependent quantizers learned from item embeddings, introducing additional training complexity and potential misalignment with recommendation objectives. Recent work further explores semantic IDs tailored to bidirectional or diffusion-based decoding [39, 31, 35]; see Appendix C.1 for discussion. As an alternative to learned quantization, Locality-Sensitive Hashing (LSH) methods [20, 19, 44], such as SimHash [4], provide a training-free mechanism for constructing semantic IDs via random projections, without requiring quantizer learning. However, prior work [37] reports inferior performance of hashing-based tokenization, leading to the common belief that such methods are less effective. In this work, we revisit this conclusion by examining the interaction between tokenization structure and decoding strategy in generative recommendation.

5 Conclusion

In this work, we demonstrate that the effectiveness of semantic ID-based generative recommendation depends less on tokenization complexity and more on structural compatibility with the decoding paradigm and alignment with the underlying semantic space. Specifically, our empirical results reveal that a lightweight, training-free SimHash tokenization scheme can achieve competitive performance when paired with parallel decoding and explicit semantic grounding. It also exhibits stronger generalization than multi-stage learned quantization pipelines. These findings suggest a critical shift in perspective for generative recommendation: cross-stage compatibility and semantic alignment may be more important than increasing tokenization complexity. This highlights the potential of simple, principled designs to rival or even outperform more complex alternatives.

Limitations. Despite its strong performance and efficiency, FLASH has several limitations. First, it relies on pre-computed text embeddings, so its effectiveness depends on the quality of the underlying representations and may degrade when textual information is sparse or noisy. Second, hashing-based tokenization may introduce collisions. Although our empirical analysis shows that collisions are limited under the evaluated configurations, they may become more pronounced at larger scales or under different codebook settings, potentially reducing the discriminability of semantic IDs. Third, our experiments are conducted on public recommendation benchmarks, and industrial-scale validation remains necessary to assess the scalability and robustness of FLASH in real-world recommendation systems. Finally, our study focuses on textual signals and does not explicitly incorporate other modalities such as images. Extending the framework to richer multimodal settings is an important direction for future work, particularly when item semantics cannot be fully captured by text alone.

Acknowledgments and Disclosure of Funding

This work is supported in part by NSF under grants III-2106758 and POSE-2346158. We thank the anonymous reviewers of NeurIPS 2026 for their valuable feedback and suggestions. The authors declare no competing interests.

References

[1] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014, 2023.

[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 *Proceedings of the 41st International Conference on Machine Learning*, pages 5209–5235, 2024.

[3] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Sequential Recommendation with Graph Neural Networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 378–387, 2021.

[4] Moses S Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 380–388, 2002.

[5] Huiyuan Chen, Yusan Lin, Menghai Pan, Lan Wang, Chin-Chia Michael Yeh, Xiaoting Li, Yan Zheng, Fei Wang, and Hao Yang. Denoising Self-Attentive Sequential Recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 92–101, 2022.

[6] Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. Structured Graph Convolutional Networks with Stochastic Masks for Recommender Systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 614–623, 2021.

[7] Huiyuan Chen, Zhe Xu, Chin-Chia Michael Yeh, Vivian Lai, Yan Zheng, Minghua Xu, and Hanghang Tong. Masked Graph Transformer for Large-Scale Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2502–2506, 2024.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020.

[9] Yingtong Dou, Zhimeng Jiang, Tianyi Zhang, Mingzhi Hu, Zhichao Xu, Huiyuan Chen, Shubham Jain, Uday Singh Saini, Xiran Fan, Jiarui Sun, et al. TransactionGPT: Toward Foundational Transaction Modeling. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 7185–7196, 2026.

[10] Junchen Fu, Xuri Ge, Alexandros Karatzoglou, Ioannis Arapakis, Suzan Verberne, Joemon M Jose, and Zhaochun Ren. Differentiable Semantic ID for Generative Recommendation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 369–379, 2026.

[11] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized Product Quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2014.

[12] 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*, pages 299–315, 2022.

[13] Ruidong Han, Bin Yin, Shangyu Chen, He Jiang, Fei Jiang, Xiang Li, Chi Ma, Mincong Huang, Xiaoguang Li, Chunzhen Jing, et al. MTGR: Industrial-Scale Generative Recommendation Framework in Meituan. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 5731–5738, 2025.

[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 639–648, 2020.

[15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-Based Recommendations with Recurrent Neural Networks. In *International Conference on Learning Representations*, 2016.

[16] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. In *Proceedings of the ACM Web Conference 2023*, pages 1162–1171, 2023.

[17] 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*, pages 956–966, 2025.

[18] 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*, pages 195–204, 2023.

[19] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[20] Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh Vempala. Locality-Preserving Hashing in Multidimensional Spaces. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 618–625, 1997.

[21] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[22] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. LLM Maybe LongLM: SelfExtend LLM Context Window Without Tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 22099–22114, 2024.

[23] Wang-Cheng Kang and Julian McAuley. Self-Attentive Sequential Recommendation. In *Proceedings of IEEE International Conference on Data Mining*, pages 197–206, 2018.

[24] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive Image Generation Using Residual Quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11513–11522, 2022.

[25] Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. Large Language Models for Generative Recommendation: A Survey and Visionary Discussions. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10146–10159, 2024.

[26] Wuchao Li, Rui Huang, Haijun Zhao, Chi Liu, Kai Zheng, Qi Liu, Na Mou, Guorui Zhou, Defu Lian, Yang Song, et al. DimeRec: A Unified Framework for Enhanced Sequential Recommendation via Generative Diffusion Models. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 726–734, 2025.

[27] Xiaohan Li, Yuqing Liu, Zheng Liu, and Philip S Yu. Time-Aware Hyperbolic Graph Attention Network for Session-Based Recommendation. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 626–635. IEEE, 2022.

[28] Zhao Li, FengYang Qi, Chuanyu Xu, Tao Zhang, Chengfu Huo, and Peng Zhang. LSIG: Long Semantic IDs for Generative Recommendation. In *Proceedings of the ACM Web Conference 2026*, pages 7779–7788, 2026.

[29] 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*, pages 1923–1933, 2025.

[30] Yuqing Liu, Yu Wang, Yuwei Cao, Lichao Sun, and Philip S Yu. Visual Summary Thought of Large Vision-Language Models for Multimodal Recommendation. In *2024 IEEE International Conference on Big Data (BigData)*, pages 456–461. IEEE, 2024.

[31] 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*, pages 5853–5864, 2026.

[32] Xinchen Luo, Jiangxia Cao, Tianyu Sun, Jinkai Yu, Rui Huang, Wei Yuan, Hezheng Lin, Yichen Zheng, Shiyao Wang, Qigen Hu, et al. QARM: Quantitative Alignment Multi-Modal Recommendation at Kuaishou. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 5915–5922, 2025.

[33] Chen Ma, Peng Kang, and Xue Liu. Hierarchical Gating Networks for Sequential Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 825–833, 2019.

[34] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52, 2015.

[35] Lingyu Mu, Hao Deng, Haibo Xing, Jinxin Hu, Yu Zhang, Xiaoyi Zeng, and Jing Zhang. Masked Diffusion Generative Recommendation. *arXiv preprint arXiv:2601.19501*, 2026.

[36] Aleksandr V Petrov and Craig Macdonald. RecJPQ: Training Large-Catalogue Sequential Recommenders. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 538–547, 2024.

[37] 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.

[38] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[39] 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.

[40] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1441–1450, 2019.

[41] Jiaxi Tang and Ke Wang. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 565–573, 2018.

[42] 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*, pages 2400–2409, 2024.

[43] 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*, pages 3245–3254, 2024.

[44] Chin-Chia Michael Yeh, Yujie Fan, Xin Dai, Uday Singh Saini, Vivian Lai, Prince Osei Aboagye, Junpeng Wang, Huiyuan Chen, Yan Zheng, Zhongfang Zhuang, et al. RPMixer: Shaking Up Time Series Forecasting with Random Projections for Large Spatial-Temporal Data. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3919–3930, 2024.

[45] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, et al. Actions Speak Louder Than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations. In *Proceedings of the 41st International Conference on Machine Learning*, pages 58484–58509, 2024.

[46] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. CoLLM: Integrating Collaborative Embeddings into Large Language Models for Recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2329–2340, 2025.

[47] 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 *IEEE 40th International Conference on Data Engineering*, pages 1435–1448. IEEE, 2024.

[48] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S^3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1893–1902, 2020.

A Additional Experimental Details

A.1 Dataset Statistics and Preprocessing

We conduct experiments on four widely used categories from the Amazon Product Reviews dataset [34]. Following prior work [37, 17], we construct user interaction sequences by treating each review as an interaction and sorting them chronologically. We filter out users and items with fewer than 5 interactions to ensure sufficient behavioral signals. We adopt the widely used leave-last-out evaluation protocol [23, 12, 47], where the last item in each sequence is used for testing, the second-to-last item for validation, and the remaining interactions for training. Table 3 summarizes the statistics of the processed datasets.

Table 3: Statistics of the processed datasets.

Datasets	# Users	# Items	# Interactions	Average Sequence Length
Beauty	22,363	12,101	198,502	8.87
Toys	19,412	11,924	167,597	8.63
Sports	35,598	18,357	296,337	8.32
CDs	75,258	64,443	1,097,592	14.58

A.2 Baselines

We compare FLASH against three categories of baselines:

(1) Item ID-based methods. These methods represent each item using a unique identifier and learn item embeddings solely from interaction data. **Caser** [41] models user–item interaction sequences using convolutional neural networks to capture both union-level and point-level sequential patterns. **GRU4Rec** [15] utilizes recurrent neural networks to model sequential dependencies in user behavior. **HGN** [33] introduces a hierarchical gating mechanism to adaptively combine long-term user preferences with short-term sequential signals. **BERT4Rec** [40] applies bidirectional Transformer encoders with a masked item prediction objective to capture contextual dependencies in interaction sequences. **SASRec** [23] employs a self-attention mechanism to model user sequences, enabling efficient capture of long-range dependencies with a unidirectional Transformer. **S^3-Rec** [48] enhances a SASRec-style Transformer with self-supervised pretraining objectives.

(2) RQ-based semantic ID methods. These methods construct semantic IDs via residual quantization and typically rely on autoregressive decoding. **TIGER** [37] adopts an RQ-VAE [24] to produce hierarchical semantic IDs and models them with autoregressive decoding, generating tokens sequentially with beam search. **LETTER** [42] learns semantic IDs via residual quantization with additional semantic, collaborative, and diversity constraints to improve token quality. **DIGER** [10] introduces differentiable residual quantization to jointly optimize tokenization and recommendation in an end-to-end framework.

(3) PQ-based semantic ID methods. These methods derive semantic IDs using product quantization, producing independent codes that enable flexible modeling. **RecJPQ** [36] applies joint product quantization to represent items as combinations of shared sub-embeddings. **VQ-Rec** [16] learns vector-quantized item representations via codebooks and integrates them into sequential recommendation models. **RPG** [17] generates long semantic IDs via optimized product quantization (OPQ) [11] and models them with parallel decoding to enable efficient and scalable inference.

A.3 Implementation Details

For all experiments, we use OpenAI's `text-embedding-3-large`[1] model to extract semantic embeddings for each item. Specifically, we construct descriptive sentences from item metadata (e.g., title, description, category), which are encoded into 3072-dimensional embeddings. These embeddings are used as inputs to different quantization methods (e.g., RQ-VAE, OPQ, LSH) to ensure a unified input representation, where SimHash is adopted as the specific LSH instantiation.

[1] `https://openai.com/index/new-embedding-models-and-api-updates/`

For the preliminary study illustrated in Figure 1, we follow TIGER's [37] default settings, using T5 as the sequence backbone with 4 codebooks, a codebook size of 256, and a maximum sequence length of 50. For the main experiments, we follow the original backbone configurations of the baselines, while FLASH employs a 2-layer, 2-head Transformer decoder with an embedding dimension $d \in \{448, 896\}$ and a feed-forward network dimension of 1024.

For the semantic grounding analysis in Section 3.3.2 (Figure 4), we apply different alignment strategies depending on the model architecture. For each model, the variants with and without semantic alignment use the same backbone and training configuration, differing only in whether the alignment objective is applied. For SASRec, which does not use a codebook, we directly align its item embeddings with the semantic embeddings. For TIGER, we introduce a linear projection layer to map the aggregated codebook representations to the semantic embedding space. This design encourages lower-level representations to capture semantic alignment, while higher layers focus on modeling item–item dependencies within the sequence. We keep the original architectures of SASRec and TIGER unchanged, and tune the hyperparameter $\lambda_{\text{align}} \in [0, 1]$ with a step size of 0.1.

All experiments are conducted on a single NVIDIA A100 (80GB) GPU. The best hyperparameters of FLASH are summarized in Table 4.

Table 4: Best hyperparameters for FLASH on each dataset.

Dataset	# Layers	# Codebooks (m)	Codebook Size (M)	Embedding Dim (d)	Codebook Dropout	Learning Rate	Alignment Weight (λ_{align})
Beauty	2	32	256	448	0.2	0.01	0.1
Toys	2	64	256	448	0.2	0.003	0.2
Sports	2	64	256	896	0.2	0.003	0.2
CDs	2	64	512	448	0.2	0.001	0.1

A.4 Cold-Start Evaluation Details

To construct the cold-start setting, we follow TIGER's evaluation protocol [37]. Specifically, we randomly sample 5% of target items (i.e., the last item in each test sequence) to form a cold-start item set. We then remove all interactions involving these items from the training set, ensuring that they remain completely unseen during training. Finally, we evaluate model performance on the subset of test sequences whose target items belong to this set, thereby simulating a strict cold-start scenario. For item ID-based baselines such as SASRec [23], embeddings of unseen items are randomly initialized at evaluation time, as they are not updated during training.

B Additional Experimental Results

B.1 Further Analysis of Training Efficiency

We follow the standard configurations for each method. TIGER adopts residual quantization with a small number of codebooks (e.g., 4 tokens per item), as commonly used due to the computational cost of autoregressive decoding. In contrast, FLASH employs a larger number of codebooks (e.g., 32 tokens per item) enabled by parallel prediction, allowing more expressive semantic IDs. SASRec operates on single-token item IDs without semantic tokenization.

We report the average training time per epoch for representative item ID-based and semantic ID-based models in Figure 7. All models are trained under the same hardware environment and data processing pipeline for a fair comparison. The reported training time measures the cost of training the sequential recommendation model (Stage II) after semantic IDs are constructed, and does not include the tokenization stage (Stage I). This allows us to isolate the efficiency of the generative recommendation model under different tokenization schemes.

FLASH consistently requires substantially less training time than TIGER across all datasets. This improvement is primarily due to the use of parallel prediction, which avoids the sequential dependency inherent in autoregressive decoding and leads to more efficient training. Notably, FLASH also employs a training-free tokenization scheme, eliminating the need for an additional quantizer training stage in the overall pipeline. Together with the tokenization efficiency comparison in Figure 3, these results provide a more complete view of the computational efficiency of the overall pipeline.

Compared to SASRec, FLASH introduces moderate computational overhead on most datasets, reflecting the increased modeling capacity of semantic ID-based representations.

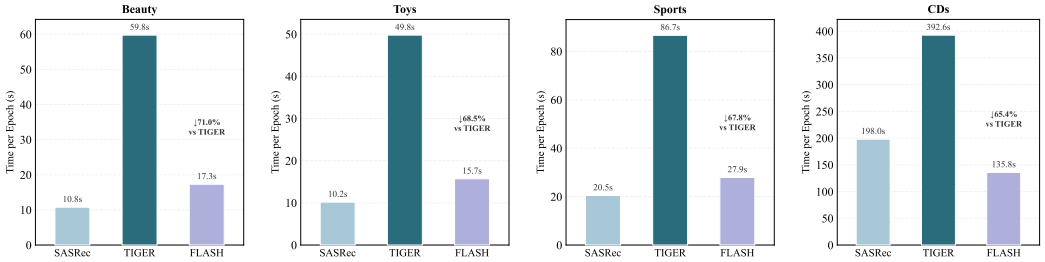


Figure 7: Average training time per epoch (seconds).

B.2 Analysis of Semantic Grounding on Additional Datasets

To complement the analysis in the main text, we report the effect of the alignment objective on the remaining datasets in Figure 8. Consistent with the observations in Figure 4, incorporating the alignment loss leads to improved performance across all evaluated datasets. This consistency suggests that the benefit of semantic grounding is robust across different data distributions, further supporting its role as a general regularization mechanism.

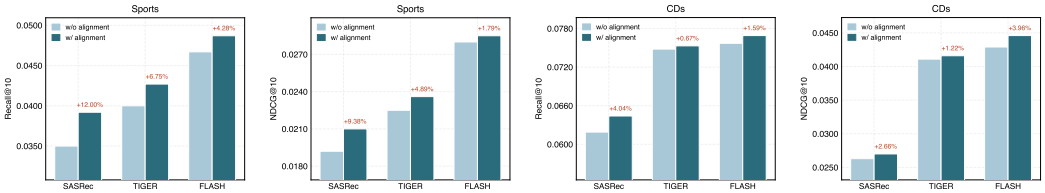


Figure 8: Effect of semantic alignment on additional datasets. "w/" and "w/o" denote training with and without the alignment loss, respectively.

B.3 Qualitative Analysis of Code Embeddings

We visualize the learned code embeddings on the unit sphere before and after training across different datasets. As shown in Figure 9, the embeddings remain broadly distributed on the unit sphere after training, while exhibiting more structured patterns. This indicates that training organizes the representation space while preserving its global geometry. These observations qualitatively support the role of semantic grounding. While SimHash provides an initial coarse-grained semantic structure through token assignments, the alignment objective further refines the embedding space, enabling the model to capture more meaningful semantic relationships.

B.4 Hyperparameter Analysis on Additional Datasets

We report additional hyperparameter analysis results on the remaining datasets in Figure 10, showing that the overall trends are consistent across datasets. Specifically, moderate values of the alignment weight λ_{align} generally yield the best performance, whereas overly large values degrade performance. Moreover, performance steadily improves as the number of codebooks increases, indicating that these datasets benefit from longer semantic IDs. For the codebook size, larger values provide modest improvements initially but exhibit diminishing returns or slight degradation as the size increases. These results further validate that the effects of key hyperparameters are stable across datasets and support the conclusions drawn in the main text.

C Additional Discussions

C.1 Relation to Diffusion-based Generative Recommendation

Recent diffusion-based generative recommenders explore semantic ID construction for bidirectional and parallel generation. LLaDA-Rec [39] identifies a mismatch between residual quantization

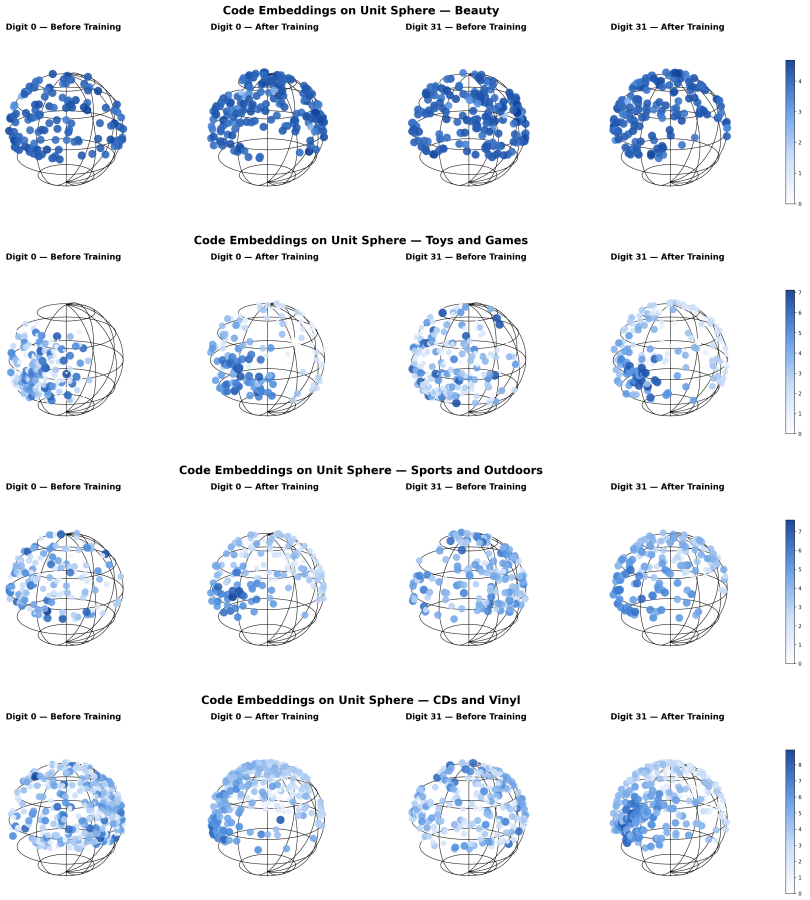


Figure 9: Visualization of learned code embeddings on the unit sphere across different datasets, before and after training. Each subfigure corresponds to a code position (labeled as "Digit" in the figure). Note that the figure adopts 0-based indexing (e.g., Digit 0 corresponds to the first code position), whereas the formulation in the main text uses 1-based indexing. The embeddings remain broadly distributed, while exhibiting more structured organization after training.

and bidirectional architectures and replaces hierarchical residual codes with parallel tokenization. Similarly, DiffGRM [31] introduces Parallel Semantic Encoding to decouple semantic ID positions for bidirectional and parallel generation, while MDGR [35] adopts a parallel codebook as the structural foundation for masked diffusion. In contrast, FLASH starts from an existing training-free tokenizer and investigates which decoding paradigm is compatible with its tokenization structure. Specifically, we revisit the previously reported poor performance of LSH-based tokenization in TIGER [37] and show that pairing its unordered codes with parallel rather than autoregressive decoding substantially improves recommendation performance.

C.2 Broader Impacts

On the positive side, FLASH removes the need for tokenizer training and enables parallel decoding, reducing tokenization and decoding costs. This may lower the barrier to adopting generative recommenders for organizations with limited resources and reduce energy consumption. Its improved cold-start generalization may also help surface previously unseen items. On the negative side, like other recommender systems, FLASH may inherit biases from interaction data and pre-trained text embeddings, including popularity bias and embedded stereotypes, potentially reducing recommendation diversity. We therefore encourage practitioners to monitor diversity and exposure fairness during deployment. These potential societal impacts are not empirically evaluated in this work.

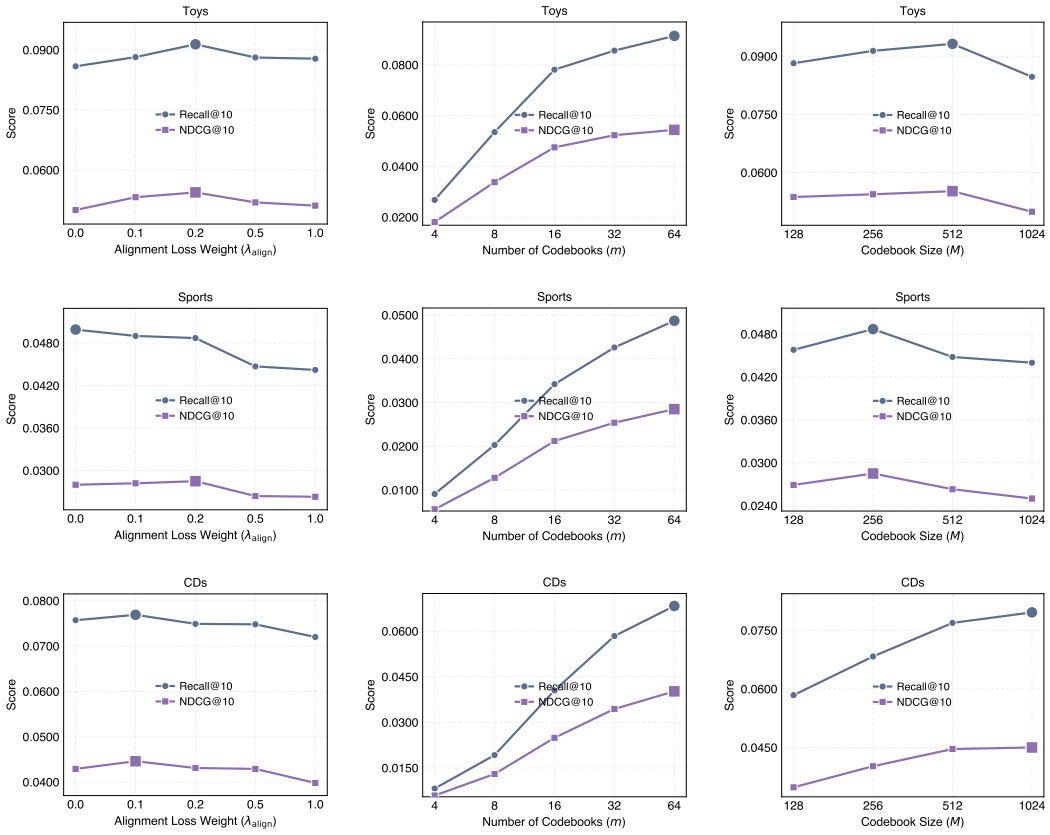


Figure 10: Performance of FLASH over different hyperparameters on the remaining datasets.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main contributions and claims, which are consistent with the experimental results and the scope of evaluation in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations in § 5, including reliance on pre-computed text embeddings, potential hash collisions, the lack of industrial-scale validation, and the current focus on textual signals.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides detailed descriptions of the model architecture, data preprocessing, training procedure, and evaluation protocol, with additional implementation details in § 3.1 and Appendix A.3. Code is also released to further facilitate reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide publicly available code and detailed instructions for data preprocessing, training, and evaluation in our GitHub repository.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide detailed experimental settings, including data splits, hyperparameters, training procedures, and evaluation protocols in Section 3.1 and Appendices A.1 and A.3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report statistical significance using paired t-tests ($p < 0.05$) for the main results in Table 1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide compute resources details in Appendix A.3. Tokenization time is reported in Table 2 and Figure 3, and training time per epoch is reported in Appendix B.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research follows the NeurIPS Code of Ethics and uses publicly available recommendation datasets without collecting private user information or conducting human-subject experiments.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: We discuss potential positive impacts (e.g., reduced computational cost and improved exposure of cold-start items) and negative impacts (e.g., inherited biases and reduced recommendation diversity), along with mitigation considerations in Appendix C.2.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not involve high-risk models or datasets that require special safeguards. The datasets used are publicly available and commonly used in recommendation research.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets, models, and prior methods used in this work are properly cited, and their usage follows the corresponding licenses and terms of use.

    Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide documentation for the released code, including instructions for running experiments and the associated license in the repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper describes the use of pre-trained LLM text embeddings as a core component of semantic ID construction (§ 2.1) and semantic alignment (§ 2.2.3).

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.