SPRINT: SINGLE-STEP GENERATIVE RECOMMENDATION VIA AVERAGE PROBABILITY VELOCITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Semantic ID (SID) based generative recommendation represents each item as a sequence of discrete tokens, and recommends by generating the SID of the item a user would like to interact with. Both dominant paradigms in this domain generally pay for generation token by token: *autoregressive models* decode the tokens left-to-right, while *non-autoregressive models* decode in parallel yet still need multiple rounds of refinement to stay competitive. Therefore, both generally spend multiple forward passes per item, a cost that is prohibitive in latency-sensitive recommender systems. We ask whether an item can be generated in a single forward pass, and answer it through a new perspective which we call *average probability velocity*. We view SID generation as a flow of token generation probabilities and characterize it by its average velocity over the whole generation process. We prove that this average velocity is fully determined by the average generation probability of each token. Therefore, we directly parameterize and learn the probabilities of all tokens in a single forward pass with a bidirectional Transformer. As these probabilities are generated independently across positions and the coherence among tokens is lost, we further design a dual-level flow contrastive objective to restore the coherence among an item's tokens. It contrasts the target SID against negative SIDs at both the token and SID levels. The token level ranks the generation probabilities of the target tokens above those of negative SIDs, while the SID level scores the tokens of each SID as a whole item for capturing token coherence of each item. Extensive experiments on 8 real-world datasets show that our model not only generates recommendations far more efficiently ($8.39-10.04\times$ speedup over the second-fastest AR/NAR method) but also attains superior recommendation accuracy (7.77% average improvement over the second-best). Our code is available at `https://anonymous.4open.science/r/SPRINT-EBEF/`.

1 INTRODUCTION

Generative recommendation (GR) with semantic IDs (SIDs) tokenizes each item into a short sequence of discrete tokens (*i.e.*, an SID), such as $[\texttt{a_2}, \texttt{b_24}, \texttt{c_182}, \texttt{d_78}]$, and recommends by generating the SID of the next item a user is interested in (Rajput et al., 2023; Wang et al., 2024a). Since an SID spans L tokens (typically 3–4), the cost of a single recommendation is determined by the number of forward passes required to produce these L tokens. Autoregressive (AR) GR decodes them left-to-right, one token per pass, and therefore requires L passes (Rajput et al., 2023; Hou et al., 2025b). Non-autoregressive (NAR) GR, most prominently masked diffusion-based (Shi et al., 2025; Liu et al., 2026b; Mu et al., 2026; Shah et al., 2025), removes the left-to-right constraint and can, in principle, generate all L tokens in a single pass. However, in pactice, these methods generally still need to generate the tokens one by one to build the conditioning among tokens (Shi et al., 2025).

Either way, generating a single item costs L forward passes, further multiplied by beam width for a ranked list. This forms an efficiency bottleneck in latency-sensitive recommender systems, which must serve requests within a tight latency budget: added delay degrades user engagement and inflates serving cost at scale (Guo et al., 2026b). A natural question is whether we can pay just one forward pass to derive an SID consisting of L tokens. Simply running an NAR model for a single step does not work. As Table 1 shows, collapsing state-of-the-art masked-diffusion GR to one step erases up to 96.31% of its accuracy (consistent with the results in (Shi et al., 2025)), because tokens decoded

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

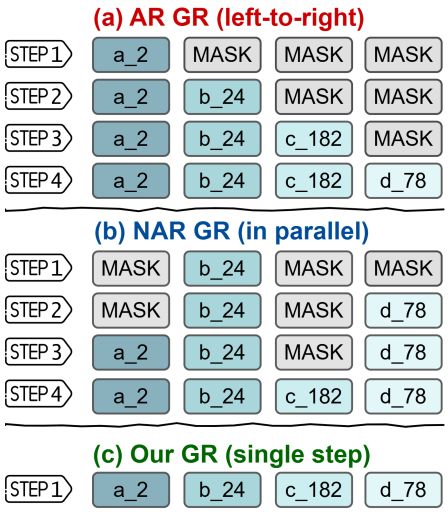

Figure 1: Comparison of AR, NAR, and our generative recommendation (GR) models. [a_2 , b_24 , c_182 , d_78] is the token sequence (*i.e.*, SID) of an item.

Table 1: Single-step vs. four-step generation performance of existing masked diffusion-based NAR GR methods. LLaDA-Rec (Shi et al., 2025) and DiffGRM (Liu et al., 2026b) are two representative and state-of-the-art masked diffusion-based methods. In these methods, each item is indexed by 4 discrete tokens. "R" and "N" are the abbreviations of Recall and NDCG.

Methods	Generation steps	Scientific		Instruments	
		R@10	N@10	R@10	N@10
LLaDA-Rec	4-step	0.0474	0.0256	0.0623	0.0337
	1-step	0.0033	0.0019	0.0023	0.0013
	Drop (%)	93.04%	92.58%	96.31%	96.14%
DiffGRM	4-step	0.0605	0.0337	0.0659	0.0356
	1-step	0.0066	0.0042	0.0152	0.0069
	Drop (%)	89.09%	87.54%	76.93%	80.62%

independently lose the conditioning that keeps an SID coherent. Existing single-step designs avoid this collapse only at a price. RPG (Hou et al., 2025a) predicts all positions at once using a multi-token prediction objective, but it must inflate the SID to 16–64 tokens for expressiveness, enlarging the embedding table to $4-16\times$ that of the 4-token SIDs used by most NAR models (Shi et al., 2025; Liu et al., 2026b). OneGR (Wang et al., 2026b) produces position-wise scores in one pass, but it then runs an A* search (Hart et al., 1968) whose look-ahead expansion is itself time-consuming.

A potential remedy is flow matching (Lipman et al., 2023), which reduces the many sampling steps of diffusion while preserving generation quality, and whose recent one-step variants (Geng et al., 2025; 2026) attain single-step generation by modeling a flow's average velocity over a finite interval. However, directly applying it to build a single-step GR is nontrivial due to two challenges (CHs). **CH1: How to attain single-step generation over the discrete token space of SIDs?** Existing single-step flow matching models (Geng et al., 2025; 2026) operate in continuous space and recover the average velocity through an identity that differentiates the velocity with respect to a continuous state, which is undefined over the categorical space of SIDs. Meanwhile, existing discrete flow matching (Gat et al., 2024; Campbell et al., 2024) generally requires multi-step iterative refinement. **CH2: How to maintain the inter-token coherence of an SID under the single-step paradigm?** When all L tokens are decoded independently in one step, the inter-token coherence that separates the matched SID from other SIDs and arbitrary token combinations is discarded, which lowers per-position accuracy and yields inaccurate recommendation generation.

To address these challenges, in this paper, we present an effective and efficient **S**ingle-ste**P** gene**R**at**I**ve recomme**N**da**T**ion model, **SPRINT**. **To address CH1**, we build SPRINT from a new perspective called *average probability velocity*. We view SID generation as a flow of token generation probabilities. At each position, the generation probability flow moves from the mask token to the target token. The rate of this movement is probability velocity. Multi-step models follow its instantaneous value and thus generate the SID token by token. We instead characterize the flow by its average velocity over the whole generation process. We theoretically prove that this average velocity is fully determined by the average generation probability of each token at each position. SPRINT therefore directly parameterizes and learns this probability with a bidirectional Transformer, which generates the probabilities of all tokens in an SID from the user history in a single forward pass. **To address CH2**, we design a dual-level flow contrastive objective, which contrasts the target SID against the SIDs of non-interacted items at two levels. At the token level, it raises the generation probabilities of the target tokens over those of negative SIDs, sharpening the prediction at each position. At the SID level, it scores each SID as a whole item rather than as separate tokens, and ranks the target SID above negative ones. This teaches the model which token combinations form the target item, restoring the inter-token coherence within a single step.

Beyond being single-step, SPRINT is concise by design, which brings two further advantages in efficiency. First, it keeps a **compact 4-token SID**, in contrast to the 16–64 tokens required by some other single-step methods (Hou et al., 2025a; Xia et al., 2026), which enlarge the embedding table by 4–$16\times$. Second, its **architecture is lightweight** (a bidirectional Transformer with a 1-layer encoder and a 4-layer decoder) with no auxiliary mask-scheduling (*e.g.*, history-aware mask position allocation (Mu et al., 2026)) or data-augmentation machinery (*e.g.*, on-policy coherent noising (Liu et al., 2026b)). Extensive experiments on 8 real-world datasets show that SPRINT decodes 8.39–$10.04\times$ faster than the second-fastest AR/NAR method while improving accuracy by 7.77% on average across all datasets and metrics over the second-best. **Our main contributions are summarized as: (1)** We are the first to model SID generation from the perspective of average probability velocity, casting the generation of an item's SID as the average velocity of the probability flow over token generation, generating SID in a single step; **(2)** We propose a novel dual-level flow contrastive objective to restore inter-token coherence within a single step; **(3)** We provide theoretical analysis and extensive experiments on 8 datasets, which verify both the efficiency and the accuracy of SPRINT.

2 Related Work

We briefly review related work here; please refer to Appendix A for more related work.

Autoregressive GR. Autoregressive (AR) GR decodes the tokens of an SID from left to right, imposing a strict sequential dependency. TIGER (Rajput et al., 2023) pioneered this paradigm by quantizing item embeddings into semantic IDs (SIDs) via residual quantization (Lee et al., 2022) and decoding them token-by-token. Subsequent work extends it through semantic-collaborative alignment (Zheng et al., 2024; Deng et al., 2025), industrial-scale transducers (Zhai et al., 2024), and contextual tokenization (Hou et al., 2025b). Whatever their differences, all of these methods inherit the cost of left-to-right decoding: an SID of L tokens requires L forward passes.

Non-autoregressive GR. Non-autoregressive (NAR) GR generates the tokens of an item's SID in parallel, relaxing the left-to-right sequential dependency of AR decoding (Ren et al., 2024). Masked diffusion-based GR (Shi et al., 2025; Liu et al., 2026b; Mu et al., 2026; Shah et al., 2025) is the most widely adopted NAR paradigm. Although NAR GR can decode from arbitrary positions, it still requires token-by-token refinement to capture token dependency to competitive performance (Shi et al., 2025), thus still require L forward passes. RPG (Hou et al., 2025a) and OneGR (Wang et al., 2026b) instead predict all tokens simultaneously, but they either greatly enlarge the embedding table or rely on an A* search (Hart et al., 1968) to select tokens sequentially at the generation stage. In contrast, we propose a theoretically principled, effective, and efficient single-step GR model.

Flow matching. Flow matching (Lipman et al., 2023; Liu et al., 2023) learns a time-dependent velocity field transporting a prior to the data distribution. Although it reduces the number of steps required by diffusion, it models the instantaneous velocity and therefore still needs to solve the generation trajectory over multiple steps. To remove this cost, single-step flow matching models (Geng et al., 2025; 2026; Kornilov et al., 2024) instead learn the average velocity over a finite interval. However, they are all defined only in continuous space. Discrete flow matching (Campbell et al., 2024; Gat et al., 2024) does extend the framework to discrete data, but remains built on instantaneous transition rates and thus still entails iterative sampling. Flow matching has also been introduced to recommender systems (Shi et al., 2026; Ye et al., 2026), yet none of them operates in the discrete SID-based generative recommendation domain. In this paper, we bridge this gap by developing a single-step GR model from the perspective of average probability velocity.

3 SPRINT: Single-step Generative Recommendation

3.1 Problem Formulation and Overview

Sequential recommendation. Existing generative recommendation (GR) generally follows the sequential recommendation paradigm (Rajput et al., 2023). Specifically, given a user's chronologically ordered interaction history $\mathcal{H} = [i_1, i_2, \cdots, i_n]$, the goal is to predict the next item i_{n+1} the user would like to interact with, ultimately producing a ranked list of K candidates.

Items as discrete token sequences (SID). Following the GR paradigm, each item is not an atomic index but a short tuple of L discrete tokens $[c_i^1, c_i^2, \cdots, c_i^L]$, *i.e.*, a semantic ID (SID). A user's

interaction history is then represented as $\mathcal{H} = [c_1^1, \cdots, c_1^L, \cdots, c_n^1, \cdots, c_n^L]$. This discretization can be realized by various vector quantization techniques, such as RQ-VAE (Lee et al., 2022; Rajput et al., 2023) and optimized product quantization (OPQ) (Ge et al., 2013; Jegou et al., 2010). We adopt OPQ following previous work (Hou et al., 2025a; Liu et al., 2026b; Wang et al., 2026b), because single-step generation does not decode a token conditioned on preceding ones, which is precisely what residual tokenizers assume. We empirically verify this choice in Section 4.3, where replacing OPQ with RQ-VAE or RQ-Kmeans consistently degrades accuracy.

Generative recommendation as next SID generation. Recommending an item then means generating its SID $[c_{n+1}^1, c_{n+1}^2, \cdots, c_{n+1}^L]$ conditioned on $\mathcal{H}$. Dominant paradigms decode this sequence either autoregressively, one token at a time from left to right and thus in L forward passes (Rajput et al., 2023), or, more recently, in parallel or in arbitrary order via masked diffusion (Liu et al., 2026b; Shi et al., 2025; Mu et al., 2026). However, as shown in Table 1, the latter still requires L forward passes to attain competitive recommendation accuracy.

Our goal: single-step generation. We aim to produce the full target SID in a single forward pass. To this end, we cast decoding as a discrete flow from an all-mask source to the target SID, and rather than integrating that flow with many small steps, we learn an *average probability velocity* that jumps from source to target in one shot. The remainder of this section develops (i) Setup and notations (§3.2); (ii) Average probability velocity for single-step SID generation (§3.3); (iii) Dual-level flow contrastive training and inference (§3.4).

3.2 Setup and Notations

Intuition. Generating a recommendation starts from an all-mask SID $[\mathbb{m}, \cdots, \mathbb{m}]$ in which no position has been decided, and ends at the SID of the item a user would like to interact with. This can be viewed as a process indexed by a time $t \in [0, 1]$ along which positions gradually commit to their target tokens. Existing GR realizes this process step by step, whereas we ask how far the whole process moves on average over the interval, so that it can be completed in one jump.

SID generation probability path. We treat SID generation as transporting probability mass from a source distribution p to a target distribution q over discrete token sequences of length L. Let $p_0(x) = p$ and $p_1(x) = q$, and let a time-indexed path $p_t(x)$, $t \in (0, 1)$, interpolate between them. In our setting the source is the fully masked sequence $[\mathbb{m}, \cdots, \mathbb{m}]$ of length L and the target is the SID $[c_{n+1}^1, \ldots, c_{n+1}^L]$ of the target item i_{n+1}. Given a coupling $\pi(x_0, x_1)$ of source and target endpoints, which in our case couples the all-mask start $x_0 = [\mathbb{m}, \cdots, \mathbb{m}]$ with the ground-truth next item's SID $x_1 = [c_{n+1}^1, \ldots, c_{n+1}^L]$, the marginal path factorizes over positions through a conditional path:

$$p_t(x) = \sum_{x_0, x_1} p_t(x|x_0, x_1)\pi(x_0, x_1), \qquad p_t(x|x_0, x_1) = \prod_{l=1}^{L} p_t(x^l|x_0, x_1). \tag{1}$$

The endpoints are pinned by delta functions (Gat et al., 2024), $p_0(x|x_0, x_1){=}\delta_{x_0}(x)$ and $p_1(x|x_0, x_1){=}\delta_{x_1}(x)$, where $\delta_y(x) = \prod_l \delta_{y^l}(x^l)$ and $\delta_{y^l}(x^l) = \mathbb{1}[x^l = y^l]$.

Convex-interpolant path. A simple and effective per-token path linearly mixes the two endpoints under a scalar scheduler κ_t (*e.g.*, $\kappa_t = t$):

$$p_t(x^l|x_0, x_1) = (1 - \kappa_t)\delta_{x_0}(x^l) + \kappa_t\delta_{x_1}(x^l), \qquad \kappa_0 = 0, \kappa_1 = 1, \dot{\kappa}_t \geq 0, \tag{2}$$

where κ_t measures how much of the target item's SID has been decided: at $\kappa_t = 0$ every SID position is still masked and nothing is committed, and at $\kappa_t = 1$ all positions have committed to the tokens of the target SID. Here $\dot{\kappa}_t := \frac{d\kappa_t}{dt}$ is the instantaneous rate at which positions switch from mask to their target token.

Instantaneous probability velocity. As t advances from 0 to 1, each SID position occasionally jumps from its current value to its target value, so that the target item is progressively revealed and running the process to $t = 1$ yields the complete SID. What governs these jumps is the probability velocity field v_t: $v_t^l(x^l, z)$ is the rate at which position l, currently holding value z^l, jumps to value x^l. Concretely, over an infinitesimal step $h > 0$, a token at $X_t^l = z^l$ has next-state distribution $X_{t+h}^l \sim \delta_{z^l}(\cdot) + hv_t^l(\cdot, z)$. The marginal velocity v_t cannot be computed directly, so we construct it as the marginalization of a conditional velocity, for which we introduce the following interpolant:

$$v_\tau^l(x^l, z|x_0, x_1) = v_\tau^l(x^l, z|x_1) = \frac{\dot{\kappa}_\tau}{1 - \kappa_\tau}\big[\delta_{x_1}(x^l) - \delta_z(x^l)\big]. \tag{3}$$

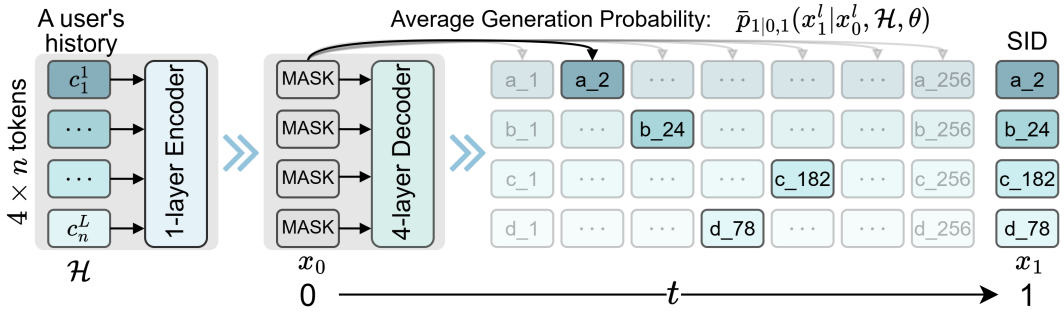

Figure 2: Illustration of SPRINT's single-step generation via average probability velocity. We here only explicitly present the generation principle of the first token of an item. In fact, all tokens are generated simultaneously under the same principle.

The bracket $[\delta_{x_1} - \delta_z]$ adds probability mass onto the target x_1 and removes it from the current state z, that is, it pushes each position away from where it currently sits and toward the token of the item being generated, while the scalar $\dot{\kappa}_\tau/(1-\kappa_\tau)$ sets the pace toward x_1. Since x_0 is always the all-mask sequence, it can be omitted from the conditioning. The derivation of this interpolant is provided in Appendix C.1, and we prove its validity as a CTMC generator (Gat et al., 2024; Campbell et al., 2024) in Appendix C.2. Standard discrete flow matching would learn v_τ and integrate it with many small jump steps; we next show how to collapse it into a single step.

3.3 Average Probability Velocity for Single-Step GR

Average probability velocity. We learn the average probability velocity over a finite interval $[r, t]$, defined as the time-averaged instantaneous probability velocity:

$$\bar{v}^l_{r,t}(x^l, z) = \frac{1}{t-r}\int_r^t v^l_\tau(x^l, z)\,\mathrm{d}\tau. \tag{4}$$

Writing this in conditional form and substituting Eq. 3, we obtain

$$\bar{v}^l_{r,t}(x^l, z|x_1) = \frac{1}{t-r}\left[\int_r^t \frac{\dot{\kappa}_\tau}{1-\kappa_\tau}\mathrm{d}\tau\right]\left[\delta_{x_1}(x^l) - \delta_z(x^l)\right] = \lambda_{r,t}\left[\delta_{x_1}(x^l) - \delta_z(x^l)\right], \tag{5}$$

with $\lambda_{r,t} = \frac{1}{t-r}\ln\frac{1-\kappa_r}{1-\kappa_t}$ [1] (complete derivation in Appendix C.3). This is precisely where our formulation departs from continuous mean-velocity methods: because the direction of transport $[\delta_{x_1} - \delta_z]$ does not vary with τ, averaging the velocity reduces to averaging a scalar rate, so the discrete convex interpolant makes the average probability velocity integrable in closed form and $\lambda_{r,t}$ is obtained without the Jacobian–vector-product identity used in the continuous case.

Marginal average probability velocity. Since the target x_1 is unknown at inference, we construct the marginal counterpart:

$$\bar{v}^l_{r,t}(x^l, z) = \frac{1}{t-r}\int_r^t \frac{\dot{\kappa}_\tau}{1-\kappa_\tau}\left[p_{1|\tau}(x^l|z) - \delta_z(x^l)\right]\mathrm{d}\tau = \lambda_{r,t}\left[\bar{p}_{1|r,t}(x^l|z) - \delta_z(x^l)\right], \tag{6}$$

where the average (denoising) posterior is $\frac{\dot{\kappa}_\tau}{1-\kappa_\tau}$-weighted time average of instantaneous posterior:

$$\bar{p}_{1|r,t}(x^l|z) := \frac{\int_r^t \frac{\dot{\kappa}_\tau}{1-\kappa_\tau} p_{1|\tau}(x^l|z)\,\mathrm{d}\tau}{\int_r^t \frac{\dot{\kappa}_\tau}{1-\kappa_\tau}\,\mathrm{d}\tau}. \tag{7}$$

This makes explicit what the network must learn: not the posterior at any single time, but the interval-averaged one. $\bar{p}_{1|r,t}(x^l|z)$ is the model's belief about which item to recommend, read out position by position given a partially unmasked SID z. It predicts the token that position l of the

[1] Note that $\lambda_{r,t} \to \infty$ as $t \to 1$, which is the expected behaviour of a CTMC rate when the transition must complete by the terminal time (Campbell et al., 2024). It serves only as an intermediate quantity in the derivation and is never used during optimization or generation.

target item should take. This is therefore the recommendation predictor itself, and single-step generation reads the recommendation straight out of it. Refer to Appendix C.4 for a complete derivation.

Parameterization. We approximate $\bar{p}_{1|r,t}(x^l|z)$ by a network with parameters θ. Since recommendation must be conditioned on the user's historical interactions, the network takes both $\mathcal{H}$ and the starting state (at r) as input, and we write $\bar{p}_{1|r,t}(x^l|z, \mathcal{H}, \theta)$. The backbone is a lightweight bidirectional Transformer: a 1-layer encoder embeds the user history $\mathcal{H}$, and a 4-layer decoder attends to the encoder output from L mask queries, one per SID position, and emits for each position l a distribution over the M tokens of the corresponding codebook $\mathcal{C}^l$.

Time sampling. To align training with single-step inference while still learning a genuine interval-indexed average velocity, we fix the terminal time $t = 1$ for all training samples, since one-step generation always targets the clean SID at $t = 1$. We then sample the source time r in a mixed manner: with probability ρ we set $r = 0$, i.e., the fully masked source $[\mathbb{m}, \cdots, \mathbb{m}]$ that exactly matches the inference condition; with probability $1 - \rho$ we draw $r \sim \mathcal{U}(0, 1)$, so that the source x_r is a partially committed SID in which each position independently holds its target token with probability κ_r and the mask token otherwise. The ρ-fraction anchors the precise single-step map used at inference, while the remaining fraction makes $\bar{p}_{1|r,1}$ a non-degenerate function of r rather than a single fixed denoiser, and additionally supplies conditional supervision in which the model predicts the remaining positions of an item given those already committed.

Training should align the marginal average probability velocity in Eq. 6 with its conditional counterpart in Eq. 5, which amounts to aligning $\lambda_{r,t}\bar{p}_{1|r,t}(x^l|z, \mathcal{H}, \theta)$ with $\lambda_{r,t}\delta_{x_1}(x^l)$. To achieve this, a natural choice is per-token cross-entropy loss:

$$-\sum\nolimits_{l=1}^{L} \log \bar{p}_{1|r,t}(x_1^l|x_r, \mathcal{H}, \theta), \tag{8}$$

where $z = x_r$ is the state of the generated SID at time r, x_1 is target item's SID, and x_1^l is its l-th token. Following (Gat et al., 2024), we omit $\lambda_{r,t}$ during training since it is shared by two terms.

From per-token likelihood to contrastive supervision. Eq. 8 draws supervision only from the tokens of target item's SID $x_1 = [c_{n+1}^1, \ldots, c_{n+1}^L]$. In recommendation, however, non-target items are informative signals in their own right, since what a user does not interact with shapes the ranking as much as what they do. We therefore replace Eq. 8 with a flow contrastive loss:

$$-\log \frac{\exp\left(\sum_{l=1}^{L} \log \bar{p}_{1|r,t}(x_1^l|x_r, \mathcal{H}, \theta)\right)}{\sum_{c\in\mathcal{S}} \exp\left(\sum_{l=1}^{L} \log \bar{p}_{1|r,t}(c^l|x_r, \mathcal{H}, \theta)\right)}, \tag{9}$$

where c is the SID of an item and $\mathcal{S}$ is a sampled SID set (a target SID and N randomly sampled SIDs of non-interacted items).

3.4 Dual-level Flow Contrastive Training and Inference

The objective in Eq. 9 calculates the generation probability flow of SIDs as a product of per-token beliefs. Under single-step generation, all L tokens are emitted simultaneously and therefore independently, so this factorized score cannot express how the tokens of an SID constrain one another: positions that are each individually plausible may combine into a tuple that indexes inaccurate items. Modeling this inter-token dependency is thus a second requirement of single-step GR, and we meet it by extending Eq. 9 into a dual-level flow contrastive loss:

$$\mathcal{L} = -\sum_{(\mathcal{H},x_1)\in\mathcal{D}} \left[\log \frac{\exp\left(\sum_{l=1}^{L} \log \bar{p}_{1|r,t}(x_1^l \mid x_r, \mathcal{H}, \theta)\right)}{\sum_{c\in\mathcal{S}} \exp\left(\sum_{l=1}^{L} \log \bar{p}_{1|r,t}(c^l \mid x_r, \mathcal{H}, \theta)\right)} + \log \frac{\exp(\langle \mathbf{u}, \mathbf{v}_{x_1}\rangle)}{\sum_{c\in\mathcal{S}} \exp\left(\langle \mathbf{u}, \mathbf{v}_c\rangle\right)}\right], \tag{10}$$

where $\langle\cdot,\cdot\rangle$ denotes cosine similarity. $\mathbf{u} = \mathrm{MLP}\big(\mathrm{Cat}(\mathbf{h}^1, \ldots, \mathbf{h}^L)\big)$ serves as the representation of the user/sequence. The L decoder hidden states $\{\mathbf{h}^l\}_{l=1}^{L}$ are produced by L mask queries attending to the encoded history $\mathcal{H}$, so each $\mathbf{h}^l$ already carries the model's belief, conditioned on the user's history, about what token position l of the target item should take. Concatenating them and passing the result through an MLP therefore yields a single vector that summarizes the target item the user is predicted to interact with, rather than L separate per-position beliefs. The item-side representation of an SID c is $\mathbf{v}_c = \mathrm{MLP}\big(\mathrm{Cat}(\mathbf{e}_{c^1}, \ldots, \mathbf{e}_{c^L})\big)$, where $\mathbf{e}_{c^l} \in \mathbb{R}^{d_m}$ is the embedding of token c^l.

The two levels play complementary roles. The token-level term optimizes the marginal prediction at each position and is what makes individual tokens accurate. The SID-level term scores the L tokens jointly as one object and pulls this joint assignment toward the item the user actually interacts with, which is exactly the dependency that independent decoding discards. Because $\mathbf{v}_c$ depends only on an item's SID and not on the user, the full matrix $\{\mathbf{v}_c\}_{c\in\mathcal{T}}$ is computed once and reused, so the SID-level term costs a single matrix product at inference time.

Inference. At inference, the start point is an all-mask sequence $[\mathbb{m}, \cdots, \mathbb{m}]$. A single forward pass of the encoder-decoder then produces both the $L \times M$ table of average generation probabilities $\bar{p}_{1|0,1}(\cdot|x_0, \mathcal{H}, \theta)$ and the user-side representation $\mathbf{u}$. To ensure that the generated SID corresponds to a valid item, we perform trie-constrained generation over $\mathcal{T}$, *i.e.*, the set of valid trajectories in the SID prefix tree (Hokamp & Liu, 2017; Hu et al., 2026c), following many existing GR works (Yang et al., 2025b; Hu et al., 2026c;a; Jin et al., 2025a; Wang et al., 2026b; Guo et al., 2026b; Hou et al., 2025a). Formally, the generation score of SID c aligns with the training process:

$$\mathrm{softmax}\Big(\sum\nolimits_{l=1}^{L} \log \bar{p}_{1|0,1}(c^l|x_0, \mathcal{H}, \theta)\Big) + \mathrm{softmax}\big(\langle \mathbf{u}, \mathbf{v}_c\rangle\big), \tag{11}$$

where softmax is taken over $\mathcal{T}$, mirroring the two levels of Eq. 10. The K valid SIDs with the highest scores are returned as the recommendation list.

4 Experiments

4.1 Experimental Setup

Datasets and Evaluation. We evaluate all methods on eight datasets spanning multiple categories: three from Amazon Reviews 2014 (McAuley et al., 2015) (Sports, Beauty, Toys), four from Amazon Reviews 2023 (Hou et al., 2026b) (Instruments, Scientific, Games, Arts), and Yelp (Wang et al., 2024a). Dataset statistics are provided in Table 6. We report Recall@K (R@K) and NDCG@K (N@K) over the entire item catalog with $K \in \{5, 10\}$, following common practice (Rajput et al., 2023; Hou et al., 2025a). To evaluate model efficiency, we report the per-epoch training time and the total inference time on the whole test set.

Baselines. We compare SPRINT against representative and state-of-the-art methods from three families: **(1) Item ID-based:** GRU4Rec (Hidasi et al., 2016), SASRec (Kang & McAuley, 2018), BERT4Rec (Sun et al., 2019), PreferDiff (Liu et al., 2025c), FAVE (Shi et al., 2026); **(2) Semantic ID-based (autoregressive):** TIGER (Rajput et al., 2023), ActionPiece (Hou et al., 2025b), COBRA (Yang et al., 2025b), SID-MLP (Guo et al., 2026b); **(3) Semantic ID-based (non-autoregressive):** RPG (Hou et al., 2025a), LLaDA-Rec (Shi et al., 2025), DiffGRM (Liu et al., 2026b), OneGR (Wang et al., 2026b). Detailed descriptions of these methods are provided in Appendix D.2.

Implementation Details. Each item is tokenized into L=4 tokens and the maximum length of user history is fixed at 20. The backbone is a bidirectional Transformer following prior work (Liu et al., 2026b) with a 1-layer encoder and a 4-layer decoder. ρ is searched over {0, 0.25, 0.5, 0.75, 1}. The number of negative SIDs sampled in Eq. 10 is set to 100 on all datasets except Arts, where it is set to 1000. All results are averaged over five runs with different random seeds. Additional implementation details are provided in Appendix D.3.

4.2 Main results

Recommendation accuracy. As shown in Table 2, SPRINT outperforms all baselines on every dataset, improving N@5 by up to 10.26% and R@5 by up to 11.50% over the second-best method. Two factors account for this. First, the dual-level flow contrastive objective addresses three requirements at once: per-token supervision, negative preference modeling, and inter-token coherence modeling, none of which a purely per-token objective can satisfy on its own. Second, single-step generation scores an SID as a whole rather than committing to tokens one at a time, which avoids the error accumulation inherent to token-by-token decoding, where an early mistaken token constrains every subsequent one (Wang et al., 2026b).

Generation efficiency. We compare inference-time generation cost of the whole test dataset under identical settings (batch size, beam width where applicable, and sequence length; details are provided in Appendix D.3) between SPRINT and accuracy-oriented multi-step GR methods (compari-

Table 2: Performance comparison between baselines and our proposed method. The best performance is in **bold** and the second-best performance is underlined. "*" denotes the improvement over the second-best performance is statistically significant ($p < 0.05$) according to a paired t-test. Due to space limitations, the results of N@10 and R@10 are provided in Table 8. Results on Yelp and Arts datasets are provided in Table 12.

Method	Sports		Beauty		Toys		Scientific		Instruments		Games	
	R@5	N@5	R@5	N@5	R@5	N@5	R@5	N@5	R@5	N@5	R@5	N@5
Item ID-based												
GRU4Rec	0.0129	0.0086	0.0164	0.0099	0.0097	0.0059	0.0184	0.0128	0.0297	0.0196	0.0461	0.0307
SASRec	0.0233	0.0154	0.0387	0.0249	0.0463	0.0306	0.0240	0.0152	0.0331	0.0211	0.0516	0.0323
BERT4Rec	0.0115	0.0075	0.0203	0.0124	0.0116	0.0071	0.0157	0.0100	0.0255	0.0160	0.0315	0.0199
PreferDiff	0.0275	0.0190	0.0455	0.0317	0.0603	0.0403	0.0161	0.0114	0.0211	0.0139	0.0356	0.0241
FAVE	0.0302	0.0194	0.0585	0.0404	0.0616	0.0444	0.0227	0.0146	0.0327	0.0209	0.0545	0.0351
Semantic ID-based (autoregressive)												
TIGER	0.0264	0.0181	0.0454	0.0321	0.0521	0.0371	0.0282	0.0183	0.0359	0.0233	0.0529	0.0348
ActionPiece	0.0316	0.0205	0.0511	0.0340	0.0477	0.0315	0.0272	0.0174	0.0388	0.0247	0.0585	0.0382
COBRA	0.0305	0.0215	0.0537	0.0395	0.0619	0.0462	0.0272	0.0179	0.0344	0.0231	0.0477	0.0307
SID-MLP	0.0265	0.0176	0.0440	0.0302	0.0375	0.0242	0.0295	0.0192	0.0395	0.0257	0.0578	0.0378
Semantic ID-based (non-autoregressive)												
RPG	0.0314	0.0216	0.0550	0.0381	0.0592	0.0401	0.0257	0.0174	0.0362	0.0241	0.0579	0.0397
LLaDA-Rec	0.0281	0.0184	0.0506	0.0352	0.0506	0.0344	0.0310	0.0203	0.0406	0.0268	0.0623	0.0415
DiffGRM	0.0363	0.0245	0.0603	0.0414	0.0618	0.0455	0.0405	0.0273	0.0426	0.0281	0.0617	0.0408
OneGR	0.0377	0.0257	0.0628	0.0426	0.0635	0.0464	0.0357	0.0245	0.0368	0.0249	0.0544	0.0369
SPRINT	**0.0411***	**0.0270***	**0.0675***	**0.0451***	**0.0708***	**0.0501***	**0.0444***	**0.0301***	**0.0462***	**0.0305***	**0.0636***	**0.0427***
Improv.	9.02%	5.06%	7.48%	5.87%	11.50%	7.97%	9.63%	10.26%	8.45%	8.54%	2.09%	2.89%

Table 3: The inference (*i.e.*, generation) time cost (s) comparison and the speedup of our method SPRINT over baselines. A comparison of training time cost is provided in Table 9. Besides, the comparison between SPRINT and other efficiency-oriented methods is provided in Table 10.

Methods	Sports		Beauty		Toys		Scientific		Instruments		Games	
	Cost	Speedup	Cost	Speedup	Cost	Speedup	Cost	Speedup	Cost	Speedup	Cost	Speedup
TIGER	80.91	30.53×	39.69	26.11×	32.45	25.35×	150.82	36.97×	166.92	36.13×	280.00	37.09×
ActionPiece	126.36	47.68×	86.16	56.68×	71.46	55.83×	185.67	45.51×	234.12	50.67×	385.05	51.00×
RPG	24.70	9.32×	14.79	9.73×	12.85	10.04×	34.25	8.39×	40.18	8.70×	64.43	8.53×
LLaDA-Rec	286.60	108.15×	183.70	120.86×	161.20	125.94×	413.70	101.40×	468.40	101.39×	756.80	100.24×
DiffGRM	37.97	14.33×	23.60	15.53×	20.12	15.72×	53.23	13.05×	59.84	12.95×	96.23	12.75×
OneGR	35.51	13.40×	22.05	14.51×	19.26	15.05×	50.22	12.31×	57.65	12.48×	140.94	18.67×
SPRINT	**2.65**	-	**1.52**	-	**1.28**	-	**4.08**	-	**4.62**	-	**7.55**	-

son with efficiency-oriented methods is provided in Table 10). As shown in Table 3, SPRINT is the fastest method on all six datasets. Against RPG, the second-fastest baseline, it delivers speedups of $8.39\times$ to $10.04\times$; against AR methods such as TIGER and ActionPiece the gap widens to $25.35\times$–$56.68\times$, and against the multi-step masked-diffusion method LLaDA-Rec it reaches up to $125.94\times$. This advantage follows directly from single-step generation: SPRINT performs one forward pass per user regardless of SID length L, whereas AR and NAR baselines generally pay L passes, further multiplied by the beam width when a ranked list is required. SPRINT also outperforms other single-step methods (RPG and OneGR) due to our lightweight and concise design.

4.3 Ablation Study

To assess the contribution of each component in SPRINT, we conduct ablation studies along two axes in Table 4. Overall, the full model achieves the best performance across all six datasets and both metrics, confirming that every component contributes positively.

We first replace OPQ tokenizer with RQ-VAE and RQ-Kmeans. Both alternatives lead to a substantial performance drop across all datasets. This indicates that the residual, sequentially dependent codes produced by RQ-based tokenizers are ill-suited to our single-step formulation. The parallel, mutually independent subspaces of OPQ align naturally with single-step generation, where all tokens of an SID are predicted simultaneously rather than conditioned on one another. This is why we adopt OPQ following other NAR models (Liu et al., 2026b; Hou et al., 2025a; Wang et al., 2026b).

We then ablate three key ingredients of our model. Removing the token-level flow (*i.e.*, the first term in Eq. 10 and Eq. 11) consistently degrades accuracy, as the model loses the fine-grained

Table 4: Ablation study on six datasets.

Methods	Sports		Beauty		Toys		Scientific		Instruments		Games	
	R@5	N@5	R@5	N@5	R@5	N@5	R@5	N@5	R@5	N@5	R@5	N@5
SPRINT (full)	**0.0411**	**0.0270**	**0.0675**	**0.0451**	**0.0708**	**0.0501**	**0.0444**	**0.0301**	**0.0462**	**0.0305**	**0.0636**	**0.0427**
OPQ $\rightarrow$ RQ-VAE	0.0284	0.0186	0.0505	0.0348	0.0452	0.0307	0.0277	0.0176	0.0383	0.0250	0.0585	0.0385
OPQ $\rightarrow$ RQ-Kmeans	0.0296	0.0195	0.0518	0.0352	0.0546	0.0355	0.0308	0.0202	0.0401	0.0262	0.0593	0.0390
w/o Token-level flow	0.0387	0.0256	0.0639	0.0417	0.0679	0.0486	0.0428	0.0294	0.0454	0.0294	0.0624	0.0403
w/o SID-level flow	0.0297	0.0197	0.0540	0.0366	0.0650	0.0470	0.0313	0.0214	0.0317	0.0210	0.0452	0.0297
w/o Contrast	0.0250	0.0176	0.0533	0.0375	0.0613	0.0453	0.0258	0.0190	0.0255	0.0179	0.0399	0.0282

supervision that sharpens the prediction at each position. Removing the SID-level flow (*i.e.*, the second term in Eq. 10 and Eq. 11) causes a much sharper drop: without it, an item is scored as a product of independent per-position beliefs, so positions that are each individually plausible may combine into a tuple that indexes an inaccurate SID. This is exactly the reason why we design the SID-level term to restore token coherence. The gap between the two variants is itself informative. The SID-level term scores the L tokens jointly as a single item, a signal that no amount of per-position supervision can supply, whereas the per-position resolution it lacks is partially recovered by the joint score itself, since $\mathbf{v}_c$ is assembled from the token embeddings of all positions. Further replacing the final objective in Eq. 10 with Eq. 8 (*w/o contrast*) causes an even larger drop, since this variant discards two signals at once: the inter-token dependency, as above, and the contrastive normalization over non-target items. The objective is then driven purely by the likelihood of the ground-truth SID and never learns what separates the target item from competing candidates, which is precisely what ranking quality depends on.

4.4 Hyperparameter Analysis

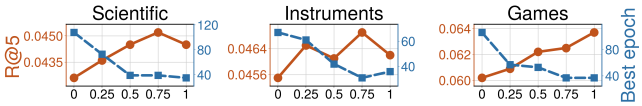

Figure 3: Effect of ρ on model performance and the epoch to achieve the best performance.

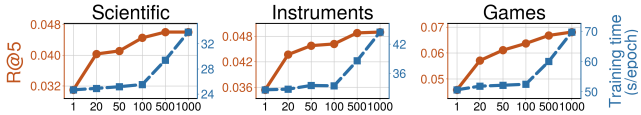

Figure 4: Effect of N on model performance and the per-epoch training time cost.

Effect of ρ. The ratio ρ controls the fraction of training samples anchored at the all-mask source $r = 0$, while the rest start from partially committed sources. As shown in Figure 3, accuracy generally improves as ρ grows and peaks at $\rho = 0.75$ or 1. Convergence also accelerates as ρ grows. Samples with $r > 0$ may teach the model to complete a partly decided SID, which differs from the inference condition. Raising ρ redirects this gradient signal to the all-mask source that the model actually decodes from, which speeds up convergence.

Effect of N. N is the number of negative SIDs sampled in Eq. 10. As shown in Figure 4, accuracy rises steadily as N grows to 100 and then improves only slowly up to 1000. In contrast, training time stays nearly flat up to $N = 100$ but grows sharply afterwards. More negatives help the model separate the target from competing SIDs, but beyond 100 they bring little new signal at a much higher cost. We therefore set $N = 100$ by default. In practice, N can be adjusted to balance accuracy and training cost according to the needs of a specific system.

5 Conclusion

Existing GR models generally generate an item token by token, which causes high inference latency that hinders their deployment in real-world RSs. In this paper, we proposed SPRINT, a single-step generative recommendation model built from a new perspective of average probability velocity. We view SID generation as a flow of token generation probabilities from an all-mask sequence to the target SID. Instead of following the instantaneous velocity of this flow step by step, we characterize it by its average velocity over the whole generation process. We prove that this average velocity is fully determined by the average generation probability of each token at each position. Single-step generation thus reduces to generating the probabilities of all tokens simultaneously in one forward pass. To restore the inter-token coherence lost in this process, we design a dual-level flow contrastive objective. It contrasts the target SID against negative SIDs at both the token and SID levels. Experiments on 8 real-world datasets show that SPRINT significantly improves both generation efficiency and recommendation accuracy. The limitations of our method are discussed in Appendix F.

AI USE STATEMENT

In this work, we used LLMs to polish the language of the manuscript, *e.g.*, correcting grammar, improving readability, and refining wording. We additionally employed LLM agents to automate parts of our experimental workflow, *e.g.*, orchestrating batched hyperparameter runs and collecting the resulting metrics. LLMs were not involved in any substantive aspect of the research: the research idea, methodology and model design. All AI-assisted text and results were reviewed and verified by the authors, who take full responsibility for the content of this paper.

ETHICS STATEMENT

This paper aims to develop a single-step generative recommendation model to improve the generation efficiency of GR models. We confirm that we do not anticipate any negative impacts and our work does not violate the ICLR code of ethics.

REPRODUCIBILITY STATEMENT

All results reported in our paper are fully reproducible. The pseudocode of our proposed method is provided in Algorithms 1 and 2. The implementation details are provided in Appendix D.3 and the hyperparameter settings of our method are provided in Table 7. The code of our method is available at `https://anonymous.4open.science/r/SPRINT-EBEF/`.

REFERENCES

Yimeng Bai, Chang Liu, Yang Zhang, Dingxian Wang, Frank Yang, Andrew Rabinovich, Wenge Rong, and Fuli Feng. Bi-level optimization for generative recommendation: Bridging tokenization and generation. In *SIGIR*, pp. 53–64, 2026.

Zhuo Cai, Shoujin Wang, Victor W Chu, Usman Naseem, Yang Wang, and Fang Chen. Unleashing the potential of diffusion models towards diversified sequential recommendations. In *SIGIR*, pp. 1476–1486, 2025.

Zhuo Cai, Shoujin Wang, Jin Li, Peilin Zhou, Victor W Chu, Fang Chen, Tianqing Zhu, and Charu Aggarwal. Steering diffusion models towards credible content recommendation. In *ICLR*, pp. 42132–42163, 2026.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi S. Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *ICML*, pp. 5453–5512, 2024.

Yiwen Chen, Fuwei Zhang, Zehao Chen, Hehan Li, Peizhi Xu, Hanmeng Liu, Shuanglong Li, Xin Pei, Fuzhen Zhuang, and Zhao Zhang. Lasar: Latent adaptive semantic aligned reasoning for generative recommendation. *arXiv preprint arXiv:2605.10207*, 2026.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

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

Dengzhao Fang, Jingtong Gao, Yu Li, Xiangyu Zhao, and Yi Chang. Prism: Purified representation and integrated semantic modeling for generative sequential recommendation. *arXiv preprint arXiv:2601.16556*, 2026.

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

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *NeurIPS*, volume 37, pp. 133345–133385, 2024.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2013.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *NeurIPS*, volume 38, pp. 75460–75482, 2025.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *CVPR*, pp. 30467–30476, 2026.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. In *NeurIPS*, volume 36, pp. 16693–16715, 2023.

Zihao Guo, Jian Wang, Ruxin Zhou, Youhua Liu, Jiawei Guo, Jun Zhao, Xiaoxiao Xu, Yongqi Liu, and Kaiqiao Zhan. S2gr: Stepwise semantic-guided reasoning in latent space for generative recommendation. In *KDD*, pp. 1498–1507, 2026a.

Zitian Guo, Yupeng Hou, Clark Mingxuan Ju, Neil Shah, and Julian McAuley. Mlps are efficient distilled generative recommenders. *arXiv preprint arXiv:2605.12617*, 2026b.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Yingzhi He, Yan Sun, Junfei Tan, Yuxin Chen, Xiaoyu Kong, Chunxu Shen, Xiang Wang, An Zhang, and Tat-Seng Chua. Reasoning over semantic ids enhances generative recommendation. In *KDD*, pp. 1638–1649, 2026.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, volume 33, pp. 6840–6851, 2020.

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *ACL*, pp. 1535–1546, 2017.

Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. In *WWW*, pp. 1162–1171, 2023.

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 *KDD*, pp. 956–966, 2025a.

Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H. Chi, Julian J. McAuley, and Derek Zhiyuan Cheng. Actionpiece: Contextually tokenizing action sequences for generative recommendation. In *ICML*, volume 267, 2025b.

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

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking llms as semantic encoders. In *ACL*, pp. 3251–3265, 2026b.

Peiyu Hu, Wayne Lu, and Jia Wang. From ids to semantics: A generative framework for cross-domain recommendation with adaptive semantic tokenization. In *AAAI*, volume 40, pp. 14874–14882, 2026a.

Yang Hu, Jiayi Guo, Jingui Ma, Ning Li, Jiangling Qin, Yanming Li, Yang Deng, Xiaoshuang Chen, and Kaiqiao Zhan. Once generated, ranked: End-to-end generative slate recommendation with unified semantic-collaborative ids. *arXiv preprint arXiv:2608.17613*, 2026b.

Yuxuan Hu, Yuhao Wang, Tianbo Huang, Chao Zhang, Ziwei Liu, Lihua Zhang, and Xiangyu Zhao. Empowering cross-domain sequential recommendation with hybrid tokenization and serial-parallel decoding. In *RecSys*, 2026c.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2010.

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

Jinqiu Jin, Yang Zhang, Fuli Feng, and Xiangnan He. Generative multi-target cross-domain recommendation. *arXiv preprint arXiv:2507.12871*, 2025a.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *ICLR*, pp. 23378–23402, 2025b.

Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvesh Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. Generative recommendation with semantic ids: a practitioner's handbook. In *CIKM*, pp. 6420–6425, 2025.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *ICDM*, pp. 197–206, 2018.

Xiaoyu Kong, Leheng Sheng, Junfei Tan, Yuxin Chen, Jiancan Wu, An Zhang, Xiang Wang, and Xiangnan He. Minionerec: An open-source framework for scaling generative recommendation. *arXiv preprint arXiv:2510.24431*, 2025.

Nikita Kornilov, Petr Mokrov, Alexander Gasnikov, and Alexander Korotin. Optimal flow matching: Learning straight trajectories in just one step. In *NeurIPS*, volume 37, pp. 104180–104204, 2024.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, pp. 11513–11522, 2022.

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

Yongqi Li, Xinyu Lin, Wenjie Wang, Fuli Feng, Liang Pang, Wenjie Li, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. A survey of generative search and recommendation in the era of large language models. *arXiv preprint arXiv:2404.16924*, 2024.

Zihao Li, Aixin Sun, and Chenliang Li. Diffurec: A diffusion model for sequential recommendation. *ACM Transactions on Information Systems*, 42(3):1–28, 2023.

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

Chengyi Liu, Yongqi Zhou, Junwei Pan, Zhixiang Feng, Chengguo Yin, Haijie Gu, Jie Jiang, Yinghao Liu, Yujuan Ding, Qing Li, et al. Diffusion language model for recommendation. *arXiv preprint arXiv:2607.21519*, 2026a.

Enze Liu, Bowen Zheng, Cheng Ling, Lantao Hu, Han Li, and Wayne Xin Zhao. Generative recommender with end-to-end learnable item tokenization. In *SIGIR*, pp. 729–739, 2025a.

Feng Liu, Lixin Zou, Xiangyu Zhao, Min Tang, Liming Dong, Dan Luo, Xiangyang Luo, and Chenliang Li. Flow matching based sequential recommender model. In *IJCAI*, pp. 3108–3116, 2025b.

Shuo Liu, An Zhang, Guoqing Hu, Hong Qian, and Tat-seng Chua. Preference diffusion for recommendation. In *ICLR*, pp. 79844–79881, 2025c.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, et al. Onerec-think: In-text reasoning for generative recommendation. *arXiv preprint arXiv:2510.11639*, 2025d.

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 *WWW*, pp. 5853–5864, 2026b.

Ziwei Liu, Yejing Wang, Wanyu Wang, Wang Zejian, Qidong Liu, Zijian Zhang, Wei Huang, Chong Chen, and Xiangyu Zhao. The best of both worlds: Harmonizing semantic and hash ids for sequential recommendation. In *KDD*, pp. 7703–7714, 2026c.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Tianze Luo, Haotian Yuan, and Zhuang Liu. Soflow: Solution flow models for one-step generative modeling. In *ICLR*, pp. 101623–101641, 2026.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pp. 43–52, 2015.

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.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *ACL Findings*, pp. 1864–1874, 2022.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *NeurIPS*, volume 38, pp. 50608–50646, 2025.

James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998.

Gustavo Penha, Edoardo D'Amico, Marco De Nadai, Enrico Palumbo, Alexandre Tamborrino, Ali Vardasbi, Max Lefarov, Shawn Lin, Timothy Heath, Francesco Fabbri, et al. Semantic ids for joint generative search and recommendation. In *RecSys*, pp. 1296–1301, 2025.

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

Haohao Qu, Shanru Lin, Yujuan Ding, Yiqi Wang, and Wenqi Fan. Diffusion generative recommendation with continuous tokens. In *WWW*, pp. 7259–7270, 2026.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. In *NeurIPS*, volume 36, pp. 10299–10315, 2023.

Yuxin Ren, Qiya Yang, Yichun Wu, Wei Xu, Yalong Wang, and Zhiqiang Zhang. Non-autoregressive generative models for reranking recommendation. In *KDD*, pp. 5625–5634, 2024.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *NeurIPS*, volume 37, pp. 130136–130184, 2024.

Kulin Shah, Bhuvesh Kumar, Neil Shah, and Liam Collins. Masked diffusion for generative recommendation. *arXiv preprint arXiv:2511.23021*, 2025.

Ke Shi, Yao Zhang, Feng Guo, Jinyuan Zhang, JunShuo Zhang, Shen Gao, and Shuo Shang. FAVE: flow-based average velocity establishment for sequential recommendation. In *SIGIR*, pp. 1633–1643, 2026.

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.

Zhengyang Su, Isay Katsman, Yueqi Wang, Ruining He, Lukasz Heldt, Raghunandan Keshavan, Shao-Chuan Wang, Xinyang Yi, Mingyan Gao, Onkar Dalal, et al. Vectorizing the trie: Efficient constrained decoding for llm-based generative retrieval on accelerators. In *KDD*, pp. 8009–8020, 2026.

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 *CIKM*, pp. 1441–1450, 2019.

Jiakai Tang, Sunhao Dai, Teng Shi, Jun Xu, Xu Chen, Wen Chen, Jian Wu, and Yuning Jiang. Think before recommend: Unleashing the latent reasoning power for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2026.

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

Puji Wang, Yingchen Zhang, Ruqing Zhang, Jiafeng Guo, and Xueqi Cheng. One-pass decoding for generative recommendation with wfst-constrained a* search. In *SIGIR*, pp. 4273–4278, 2026b.

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 *CIKM*, pp. 2400–2409, 2024a.

Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, et al. Eager: Two-stream generative recommender with behavior-semantic collaboration. In *KDD*, pp. 3245–3254, 2024b.

Yejing Wang, Shengyu Zhou, Jinyu Lu, Qidong Liu, Xinhang Li, Wenlin Zhang, Feng Li, Pengjie Wang, Chuan Yu, Jian Xu, Bo Zheng, and Xiangyu Zhao. Gflowgr: Fine-tuning generative recommendation frameworks with generative flow networks. In *SIGIR*, pp. 1925–1936, 2026c.

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

Yuhao Wang, Junwei Pan, Xinhang Li, Maolin Wang, Yuan Wang, Yue Liu, Dapeng Liu, Jie Jiang, and Xiangyu Zhao. Empowering large language model for sequential recommendation via multimodal embeddings and semantic ids. In *CIKM*, pp. 3209–3219, 2025.

Ming Xia, Zhiqin Zhou, Guoxin Ma, and Dongmin Huang. Unleash the potential of long semantic ids for generative recommendation. *arXiv preprint arXiv:2602.13573*, 2026.

Chaoqun Yang, Xinyu Lin, Wenjie Wang, Yongqi Li, Teng Sun, Xianjing Han, and Tat-Seng Chua. Earn: Efficient inference acceleration for llm-based generative recommendation by register tokens. In *KDD*, pp. 3483–3494, 2025a.

Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, LIU LIN, et al. Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations. In *NeurIPS*, volume 38, pp. 93746–93770, 2025b.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. In *NeurIPS*, volume 36, pp. 24247–24261, 2023.

Wencai Ye, Mingjie Sun, Shaoyun Shi, Peng Wang, Wenjin Wu, and Peng Jiang. Das: Dual-aligned semantic ids empowered industrial recommender system. In *CIKM*, pp. 6217–6224, 2025.

Xiaoxin Ye, Chengkai Huang, Hongtao Huang, and Lina Yao. Gaussian mixture flow matching with domain alignment for multi-domain sequential recommendation. In *WWW*, pp. 6159–6170, 2026.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *ICML*, pp. 58484–58509, 2024.

Lei Zhang, Zihao Chen, Wuji Zhang, Hongke Zhao, and Likang Wu. Gdiffmae: Guided diffusion enhanced mask graph autoencoder for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

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 *ICDE*, pp. 1435–1448, 2024.

Bangguo Zhu, Peng Huo, Yuanbo Zhao, Zhicheng Du, Jun Yin, and Senzhang Wang. Time-aware diffusion based on preference disentanglement for generative recommendation. *arXiv preprint arXiv:2606.01670*, 2026.

Appendix of SPRINT

CONTENTS

A More Related Work **17**

A.1 Generative Recommendation 17

A.2 Flow matching 18

B Notations **19**

C Theoretical Derivations **19**

C.1 Derivation of the interpolant in Equation 3 19

C.2 Validity of the Conditional Velocity in Proposition 1 20

C.3 Conditional Average Probability Velocity 21

C.4 Marginal Average Probability Velocity 21

D More Experiments **22**

D.1 Datasets Statistics 22

D.2 Baseline Description 23

D.3 More Implementation Details 24

D.4 More Comparison Results ($K = 10$) 25

D.5 Training Efficiency Analysis 25

D.6 Comparison between SPRINT and Other Efficiency-oriented Methods 26

D.7 Parameter Memory 27

D.8 Experiments on More Diverse Datasets 27

D.9 Performance of Decoder-only Architecture 28

D.10 More Hyperparameter Analysis 28

D.11 Performance Comparison Under Larger K 29

D.12 Performance of TIGER using OPQ as the tokenizer 29

E Algorithms **29**

F Limitations **30**

A More Related Work

A.1 Generative Recommendation

A.1.1 Autoregressive Generative Recommendation.

Autoregressive (AR) GR decodes the tokens of an SID from left to right, imposing a strict sequential dependency (Rajput et al., 2023; Hou et al., 2025b; Yang et al., 2025b; Fang et al., 2026; Qiao et al., 2026; Ju et al., 2025; He et al., 2026; Wang et al., 2026c; Hou et al., 2026a). TIGER (Rajput et al., 2023) pioneered this paradigm by quantizing item embeddings into semantic IDs (SIDs) via residual quantization (Lee et al., 2022; Kong et al., 2025; Penha et al., 2025; Hou et al., 2023) and decoding them token-by-token.

These methods derives its SIDs purely from content embeddings; however, the resulting token space reflects semantic similarity alone: two items that users frequently co-consume need not be close in it, so the identifiers themselves carry no collaborative signal. To overcome this issue, some works incorporate collaborative signals pre-trained recommendation models into item tokenization stage (Wang et al., 2024a; Liu et al., 2026c; Wang et al., 2025; Ye et al., 2025; Li et al., 2026). LETTER (Wang et al., 2024a) resolves this by extending the RQ-VAE tokenizer with collaborative alignment and a diversity regularizer that additionally mitigates code assignment imbalance, EAGER (Wang et al., 2024b) by keeping behavioral and semantic information in two separate streams so that neither is collapsed into the other, and LC-Rec (Zheng et al., 2024) by integrating collaborative semantics into an LLM-based recommender.

These methods enrich what the tokenizer encodes, but they all learn it in a stage that precedes and is separate from the recommender, so the identifiers are frozen before the recommender ever sees them and can never adapt to the recommendation objective. To address this limitation, a plenty of works explore to develop end-to-end GR (Liu et al., 2025a; Fu et al., 2026; Li et al., 2026; Wang et al., 2026a; Jiang et al., 2026; Bai et al., 2026; Hu et al., 2026b). ETEGRec (Liu et al., 2025a) eliminates this decoupling by optimizing tokenizer and recommender end to end, using recommendation-oriented alignment together with an alternating optimization scheme to keep joint training stable. A residual form of the same rigidity is that an item receives the same SID regardless of the actions surrounding it; ActionPiece (Hou et al., 2025b) addresses this by tokenizing action sequences contextually, so that an item's tokens depend on the context in which it occurs.

Inspired by the success of large reasoning models (Tang et al., 2026), another line of work equips GR with reasoning ability. One branch performs explicit reasoning in natural language. OneRec-Think (Liu et al., 2025d) aligns itemic tokens with the textual space of an LLM. It then activates reasoning through reasoning scaffolding and enhances it with a recommendation-specific reward. SIDReasoner (He et al., 2026) first strengthens the alignment between SIDs and language. It then optimizes reasoning trajectories via outcome-driven reinforcement learning, without explicit reasoning annotations. The other branch reasons implicitly in latent space. S^2GR (Guo et al., 2026a) performs stepwise latent reasoning guided by the semantics of hierarchical SID codes. LASAR (Chen et al., 2026) adopts recurrent hidden-state feedback and adapts the reasoning depth to each sample.

Even with well-constructed identifiers, generation itself remains error-prone: an SID is a tuple drawn from L codebooks, and an unconstrained decoder can emit combinations that correspond to no item in the corpus. Trie-constrained decoding answers this by building a prefix tree over all valid SIDs and masking, at every step, any token that would leave a valid root-to-leaf path (Su et al., 2026; Wang et al., 2026b; Li et al., 2024; Hu et al., 2026a).

What none of these advances changes is the decoding paradigm itself. Tokens are still emitted strictly left to right, so an SID of L tokens costs L forward passes, further multiplied by the beam width when a ranked list is required, and this cost is inherent to autoregression rather than to any particular instantiation of it. Relaxing the left-to-right constraint is precisely what the non-autoregressive methods reviewed next set out to do.

A.1.2 Non-autoregressive Generative Recommendation

The L sequential forward passes that autoregression imposes are a consequence of its factorization, not of the recommendation task, which motivates non-autoregressive (NAR) decoding: tokens can

be emitted in parallel from any position without left-to-right constraint (Ren et al., 2024). Removing the causal factorization, however, removes with it the conditioning that kept the tokens of an SID coherent, and the NAR literature is largely an account of how that lost dependency is put back.

The dominant answer is to restore it iteratively. Following the rapid progress of diffusion models (Ho et al., 2020; Yang et al., 2023; Cai et al., 2025; 2026; Qu et al., 2026) and diffusion language models (Nie et al., 2025; Gulrajani & Hashimoto, 2023; Sahoo et al., 2024), diffusion-based GR (Qu et al., 2026; Shi et al., 2025; Liu et al., 2026a; Zhang et al., 2025; Zhu et al., 2026) reveals the positions of an SID over several rounds, each round conditioning on the tokens already committed. LLaDA-Rec (Shi et al., 2025) and MaskGR (Shah et al., 2025) adopt discrete diffusion to decode in arbitrary rather than left-to-right order, and subsequent work refines which positions to reveal and when, for instance through history-aware mask position allocation (Mu et al., 2026) or on-policy coherent noising (Liu et al., 2026b). Order is thus no longer fixed, but the number of network evaluations is not reduced: competitive accuracy still requires token-by-token refinement (Shi et al., 2025), so L forward passes remain, and collapsing the schedule to a single step causes accuracy to collapse with it (Table 1).

A second answer avoids iteration by making a single parallel prediction expressive enough to stand on its own. RPG (Hou et al., 2025a; Xia et al., 2026) predicts all positions simultaneously under a multi-token objective and excludes incoherent outputs through graph-constrained decoding. The dependency, however, is enforced at decoding time rather than learned, and expressiveness is bought by lengthening the SID to 16–64 tokens, which enlarges the embedding table to roughly $4\times$–$16\times$ that of the 4-token SIDs used by most NAR models (Shi et al., 2025; Liu et al., 2026b; Mu et al., 2026). OneGR (Wang et al., 2026b) selects tokens by an A* search (Hart et al., 1968) that weighs each prefix against a look-ahead bound over the remaining positions, so decoding is not strictly single-pass in scoring but remains sequential and multiple steps.

These methods either still require multiple steps, substantially enlarge the embedding table, or rely on time-consuming search without a principled single-step guarantee. To bridge these gaps, in this paper, we propose a theoretically principled, effective and efficient single-step generative recommendation model.

A.2 Flow matching

Flow matching (Lipman et al., 2023; Liu et al., 2023; Jin et al., 2025b; Dao et al., 2023) was introduced to shorten the long sampling trajectories of diffusion models, learning a time-dependent velocity field that transports a prior to the data distribution along substantially straighter paths. What it learns, however, is the *instantaneous* velocity, which describes an infinitesimal move and must therefore be integrated: sampling still solves the generation trajectory over many network evaluations, so the cost is reduced rather than removed.

Single-step flow matching addresses this residual cost by changing the object of learning (Luo et al., 2026; Geng et al., 2025; 2026). Geng et al. (Geng et al., 2025) model the *average* velocity over a finite interval, which spans the trajectory in one evaluation, and recovers it through an identity that relates the average field to the instantaneous one by differentiating with respect to the state; later work improves its training stability (Geng et al., 2026) or attains one-step transport by learning straight trajectories directly (Kornilov et al., 2024). This resolves the step count, but the enabling identity is defined only where the state is continuous and differentiable, and has no counterpart over the categorical token space in which SIDs live.

Discrete flow matching (Campbell et al., 2024; Gat et al., 2024) supplies the missing half of the picture, extending the framework to categorical data by formulating generation as a continuous-time Markov chain over discrete states. Yet it remains built on instantaneous transition rates, and therefore inherits exactly the limitation that motivated the average-velocity view in the first place: sampling is again iterative. The two lines are thus complementary but disjoint, one achieving single-step generation only in continuous space, the other achieving discreteness only at multi-step cost.

Within recommendation, flow matching has so far been used to model preference transport rather than to accelerate generation, whether for sequential recommendation (Shi et al., 2026; Liu et al., 2025b), and multi-domain settings (Ye et al., 2026), but none of these works operates in discrete

Table 5: Notations and explanations.

Notation	Explanation
$\mathcal{H}$	Discrete token sequence of the user's interacted items
n	Maximum length of items in the user history
K	Number of items in the returned recommendation list
L	Number of positions in an SID
M	Size of each codebook
$\mathcal{C}^l$	Codebook of position l, from which the l-th token of an SID is drawn
$c = [c^1, \ldots, c^L]$	Semantic ID (SID) of an item; c^l is its token at position l
$\mathcal{T}$	Set of valid trajectories in the SID prefix tree
$\mathbb{m}$	Mask token, denoting a position that has not yet been decided
x_0	Source of the flow, *i.e.*, the all-mask sequence $[\mathbb{m}, \ldots, \mathbb{m}]$
x_1	Target of the flow, *i.e.*, the SID of the target item i_{n+1}
x_r	State of the SID being generated at time r
z	Generic current state of a position
r, t	Start and terminal times of the interval over which the velocity is averaged
τ	Integration variable ranging over $[r, t]$
$p_t(x)$	Time-indexed probability path over SIDs
$\pi(x_0, x_1)$	Coupling of source and target endpoints
κ_t	Scheduler measuring how much of the target SID has been decided
$\dot{\kappa}_t$	Time derivative of κ_t, *i.e.*, the instantaneous switching rate
$\delta_y(\cdot)$	Indicator that a position (or an SID) equals y
$v^l_\tau(x^l, z)$	Instantaneous probability velocity: rate at which position l jumps from z^l to x^l
$\bar{v}^l_{r,t}(x^l, z)$	Average probability velocity of position l over the interval $[r, t]$
$\lambda_{r,t}$	Interval-level scalar rate, $\frac{1}{t-r} \ln \frac{1-\kappa_r}{1-\kappa_t}$
$p_{1\|\tau}(x^l\|z)$	Instantaneous target posterior at position l
$\bar{p}_{1\|r,t}(x^l\|z, \mathcal{H}, \theta)$	Average generation probability learned by the network; the token predictor
θ	Parameters of the bidirectional Transformer (1-layer encoder, 4-layer decoder)
$\mathbf{h}^l$	Last-layer decoder hidden state at SID position l
$\mathbf{e}_{c^l}$	Embedding of token c^l in codebook $\mathcal{C}^l$
$\mathbf{u}$	User/Sequence-side joint representation, obtained from $[\mathbf{h}^1, \ldots, \mathbf{h}^L]$
$\mathbf{v}_c$	SID-side joint representation of a candidate SID c
$\langle \cdot, \cdot \rangle$	Cosine similarity
N	Number of negative SID samples
$\mathcal{D}$	Training set of (history, target SID) pairs
$\mathcal{L}$	Dual-level flow contrastive loss

SID generation domain or generates an item in a single step. We close the gap between the two lines by deriving the average probability velocity directly in discrete token space.

B Notations

The descriptions of the notations used in this paper are presented in Table 5.

C Theoretical Derivations

C.1 Derivation of the interpolant in Equation 3

The marginal velocity $v^l_\tau(x^l, z)$ is intractable, so we first derive the velocity conditioned on a fixed target x_1 (the source x_0 can be omitted since it is an all-mask sequence), for which the path is a simple interpolant. We use the convex-interpolant conditional path, whose τ-derivative follows since only κ_τ depends on τ:

$$p_\tau(x^l|x_1) = (1 - \kappa_\tau)\delta_{\mathbb{m}}(x^l) + \kappa_\tau \delta_{x_1}(x^l), \tag{C.12}$$

$$\dot{p}_\tau(x^l|x_1) = \dot{\kappa}_\tau[\delta_{x_1}(x^l) - \delta_{\mathbb{m}}(x^l)]. \tag{C.13}$$

A velocity is admissible only if it generates this path through the Kolmogorov equation (Campbell et al., 2024; Gat et al., 2024), *i.e.*, the rate of change of the probability at x^l equals the net probability flowing into x^l from every current state z^l:

$$\dot{p}_\tau(x^l|x_1) = \sum_{z^l} v_\tau^l(x^l, z^l|x_1) p_\tau(z^l|x_1). \tag{C.14}$$

Proposition 1. *The conditional velocity generating Eq. C.12 in denoiser mode is:*

$$v_\tau^l(x^l, z|x_1) = \frac{\dot{\kappa}_\tau}{1-\kappa_\tau}[\delta_{x_1}(x^l) - \delta_z(x^l)]. \tag{C.15}$$

Proof. In denoiser mode (predicting the clean target, as in x-prediction diffusion) the source coefficient is zero, leaving a two term ansatz supported on the target and the current state:

$$v_\tau^l(x^l, z|x_1) = a_\tau^1 \delta_{x_1}(x^l) + a_\tau^2 \delta_z(x^l). \tag{C.16}$$

Substitute Eq. C.16 into the right-hand side of Eq. C.14, we have:

$$\begin{aligned}
\sum_{z^l} v_\tau^l(x^l, z^l \mid x_1)\, p_\tau(z^l \mid x_1) &= a_\tau^1\, \delta_{x_1}(x^l) \underbrace{\sum_{z^l} p_\tau(z^l \mid x_1)}_{=1} + a_\tau^2 \underbrace{\sum_{z^l} \delta_{z^l}(x^l)\, p_\tau(z^l \mid x_1)}_{=p_\tau(x^l|x_1)} \\
&= a_\tau^1\, \delta_{x_1}(x^l) + a_\tau^2\, p_\tau(x^l \mid x_1) \\
&= a_\tau^1\, \delta_{x_1}(x^l) + a_\tau^2 \big[(1-\kappa_\tau)\delta_{\mathbb{m}}(x^l) + \kappa_\tau \delta_{x_1}(x^l)\big] \\
&= (a_\tau^1 + a_\tau^2 \kappa_\tau)\, \delta_{x_1}(x^l) + a_\tau^2 (1-\kappa_\tau)\, \delta_{\mathbb{m}}(x^l).
\end{aligned} \tag{C.17}$$

Setting equation C.17 equal to equation C.13 and matching the coefficients of the linearly independent indicators δ_{x_1} and $\delta_{\mathbb{m}}$ (distinct since $x_1 \neq \mathbb{m}$):

$$a_\tau^1 + a_\tau^2 \kappa_\tau = \dot{\kappa}_\tau, \qquad a_\tau^2 (1-\kappa_\tau) = -\dot{\kappa}_\tau. \tag{C.18}$$

Solving the second for a_τ^2 and back-substituting:

$$a_\tau^2 = -\frac{\dot{\kappa}_\tau}{1-\kappa_\tau}, \qquad a_\tau^1 = \dot{\kappa}_\tau - a_\tau^2 \kappa_\tau = \frac{\dot{\kappa}_\tau(1-\kappa_\tau) + \dot{\kappa}_\tau \kappa_\tau}{1-\kappa_\tau} = \frac{\dot{\kappa}_\tau}{1-\kappa_\tau}. \tag{C.19}$$

Substituting equation C.19 into equation C.16 gives equation C.15. Proof completed.

C.2 Validity of the Conditional Velocity in Proposition 1

For equation C.15 to be a legitimate CTMC generator (Norris, 1998; Campbell et al., 2024), two conditions must hold.

Condition 1 (columns sum to zero — probability conservation). Mass leaving z must reappear elsewhere, so summing over all destinations x^l must give zero:

$$\sum_{x^l} v_\tau^l(x^l, z \mid x_1) = \frac{\dot{\kappa}_\tau}{1-\kappa_\tau}\Big[\sum_{x^l} \delta_{x_1}(x^l) - \sum_{x^l} \delta_z(x^l)\Big] = \frac{\dot{\kappa}_\tau}{1-\kappa_\tau}\,[1-1] = 0. \tag{C.20}$$

Condition 2 (non-negative off-diagonal — valid jump rates). For $x^l \neq z$ we have $\delta_z(x^l) = 0$, so we have,

$$v_\tau^l(x^l, z \mid x_1) = \frac{\dot{\kappa}_\tau}{1-\kappa_\tau}\, \delta_{x_1}(x^l) \geq 0, \tag{C.21}$$

using $\dot{\kappa}_\tau \geq 0$ (monotone scheduler) and $1-\kappa_\tau \geq 0$. Both hold, so equation C.15 is a valid velocity generating equation C.12.

C.3 Conditional Average Probability Velocity

To jump over a whole interval $[r, t]$ in one evaluation instead of many small steps, we use the time-averaged velocity:

$$\bar{v}^l_{r,t}(x^l, z) = \frac{1}{t-r}\int_r^t v^l_\tau(x^l, z)\, d\tau. \tag{C.22}$$

Proposition 2. *The conditional average velocity retains the same "target minus current" structure as the instantaneous one, with the time-varying rate $\frac{\dot\kappa_\tau}{1-\kappa_\tau}$ collapsed into a single interval-level scalar $\lambda_{r,t}$:*

$$\bar{v}^l_{r,t}(x^l, z \mid x_1) = \lambda_{r,t}\big[\delta_{x_1}(x^l) - \delta_z(x^l)\big], \qquad \lambda_{r,t} := \frac{1}{t-r}\ln\frac{1-\kappa_r}{1-\kappa_t}, \tag{C.23}$$

where coefficient $\lambda_{r,t}$ is the time-averaged switching rate over $[r, t]$: it is the total rate accumulated across the interval, $\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}d\tau = \ln\frac{1-\kappa_r}{1-\kappa_t}$, divided by the interval length $t-r$. Because the direction of transport $[\delta_{x_1} - \delta_z]$ does not change with τ, averaging the velocity reduces entirely to averaging this scalar rate—which is exactly what makes a single interval-spanning step well-defined.

Proof. Substitute C.15 into C.22, we have:

$$\begin{aligned}
\bar{v}^l_{r,t}(x^l, z|x_1) &= \frac{1}{t-r}\int_r^t v^l_\tau(x^l, z|x_1)d\tau \\
&= \frac{1}{t-r}\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}[\delta_{x_1}(x^l) - \delta_z(x^l)]d\tau \\
&= \frac{1}{t-r}[\delta_{x_1}(x^l) - \delta_z(x^l)]\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}d\tau \\
&= \frac{1}{t-r}[\delta_{x_1}(x^l) - \delta_z(x^l)]\,[-\ln(1-\kappa_\tau)]_r^t \\
&= \frac{1}{t-r}\ln\frac{1-\kappa_r}{1-\kappa_t}[\delta_{x_1}(x^l) - \delta_z(x^l)] \\
&= \lambda_{r,t}[\delta_{x_1}(x^l) - \delta_z(x^l)],
\end{aligned} \tag{C.24}$$

where $\lambda_{r,t} = \frac{1}{t-r}\ln\frac{1-\kappa_r}{1-\kappa_t}$. Proof completed.

C.4 Marginal Average Probability Velocity

We need the marginal velocity, since the target x_1 is unknown at generation time. By the standard flow-matching marginalization, the marginal velocity that generates p_τ is the average of the conditional velocities weighted by the target posterior $p_\tau(x_1 \mid z)$—the model's belief, given the current partial state z, about which item is being generated:

$$v^l_\tau(x^l, z) = \sum_{x_1} v^l_\tau(x^l, z \mid x_1)\, p_\tau(x_1 \mid z). \tag{C.25}$$

The interval-average of this quantity is subtle for one reason: the posterior $p_\tau(x_1 \mid z)$ depends on τ and lives inside the time integral $[r, t]$. Consequently the marginal average velocity is not obtained by taking the conditional result C.23 and replacing x_1 by a single posterior mean; it requires a specifically weighted average posterior, defined next.

Proposition 3. *Let $p_{1|\tau}(x^l \mid z)$ be the instantaneous target posterior and $\bar{p}_{1|r,t}(x^l \mid z)$ is $\frac{\dot\kappa_\tau}{1-\kappa_\tau}$-weighted time average over $[r, t]$:*

$$p_{1|\tau}(x^l \mid z) := \sum_{x_1}\delta_{x_1}(x^l)\, p_\tau(x_1 \mid z), \qquad \bar{p}_{1|r,t}(x^l \mid z) := \frac{\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, p_{1|\tau}(x^l \mid z)\, d\tau}{\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, d\tau}. \tag{C.26}$$

Then the marginal average velocity keeps the same "posterior minus current" form as C.23, with the weighted-average posterior in place of the target indicator:

$$\bar{v}^l_{r,t}(x^l, z) = \lambda_{r,t}\big[\bar{p}_{1|r,t}(x^l \mid z) - \delta_z(x^l)\big]. \tag{C.27}$$

Proof. Substitute the conditional velocity C.15 into the marginalization C.25, we have:

$$\begin{aligned} v_\tau^l(x^l, z) &= \sum_{x_1} v_\tau^l(x^l, z \mid x_1)\, p_\tau(x_1 \mid z) \\ &= \frac{\dot\kappa_\tau}{1-\kappa_\tau} \sum_{x_1} \big[\delta_{x_1}(x^l) - \delta_z(x^l)\big]\, p_\tau(x_1 \mid z) \\ &= \frac{\dot\kappa_\tau}{1-\kappa_\tau} \Big[\underbrace{\sum_{x_1} \delta_{x_1}(x^l)\, p_\tau(x_1 \mid z)}_{= p_{1|\tau}(x^l|z)} - \delta_z(x^l) \underbrace{\sum_{x_1} p_\tau(x_1 \mid z)}_{=1} \Big] \\ &= \frac{\dot\kappa_\tau}{1-\kappa_\tau} \big[p_{1|\tau}(x^l \mid z) - \delta_z(x^l)\big]. \end{aligned} \tag{C.28}$$

The first sum is the posterior by definition in C.26 following (Gat et al., 2024). $\sum_{x_1} p_\tau(x_1|z) = 1$ by the law of total probability.

Insert Eq. C.28 into the Eq. C.22 of the average velocity and separate the posterior term from the current-state term:

$$\bar v_{r,t}^l(x^l, z) = \underbrace{\frac{1}{t-r}\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, p_{1|\tau}(x^l \mid z)\, d\tau}_{\text{(I)}} \; - \; \delta_z(x^l) \; \underbrace{\frac{1}{t-r}\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, d\tau}_{\text{(II)}}. \tag{C.29}$$

Term (II) is exactly $\lambda_{r,t}$ by Eq. C.23. For term (I), multiply and divide by the normalizer $\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau} d\tau = (t-r)\lambda_{r,t}$, which turns the integral into the weighted-average posterior in C.26:

$$\begin{aligned} \text{(I)} &= \frac{1}{t-r}\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, p_{1|\tau}(x^l \mid z)\, d\tau \\ &= \frac{1}{t-r}\left(\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, d\tau\right) \cdot \frac{\displaystyle\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, p_{1|\tau}(x^l \mid z)\, d\tau}{\displaystyle\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, d\tau} \\ &= \frac{1}{t-r}\left(\int_r^t \frac{\dot\kappa_\tau}{1-\kappa_\tau}\, d\tau\right) \bar p_{1|r,t}(x^l \mid z) \\ &= \frac{1}{t-r}\,(t-r)\,\lambda_{r,t}\,\bar p_{1|r,t}(x^l \mid z) \\ &= \lambda_{r,t}\,\bar p_{1|r,t}(x^l \mid z). \end{aligned} \tag{C.30}$$

Substituting (I) and (II) back,

$$\bar v_{r,t}^l(x^l, z) = \lambda_{r,t}\,\bar p_{1|r,t}(x^l \mid z) - \lambda_{r,t}\,\delta_z(x^l) = \lambda_{r,t}\big[\bar p_{1|r,t}(x^l \mid z) - \delta_z(x^l)\big],$$

which is C.27. Proof completed.

D More Experiments

D.1 Datasets Statistics

We evaluate all methods on eight datasets spanning multiple categories: three from Amazon Reviews 2014 (McAuley et al., 2015) (Sports, Beauty, Toys)[2], four from Amazon Reviews 2023 (Hou et al., 2026b) (Instruments, Scientific, Games, Arts)[3], and Yelp[4] (Wang et al., 2024a). The statistics of our

[2]https://cseweb.ucsd.edu/ jmcauley/datasets/amazon/links.html

[3]https://huggingface.co/datasets/McAuley-Lab/Amazon-Reviews-2023

[4]https://business.yelp.com/data/resources/open-dataset/

Table 6: Statistics of the datasets used in this paper.

Dataset	#Users	#Items	#Interactions	Avg. Sequence Length
Sports	35,598	18,357	296,337	8.32
Beauty	22,363	12,101	198,502	8.87
Toys	19,412	11,924	167,597	8.63
Instruments	57,439	24,587	511,836	8.91
Scientific	50,985	25,848	412,947	8.10
Games	94,762	25,612	814,586	8.60
Yelp	30,431	20,033	316,354	10.40
Arts	197,287	89,959	1,786,437	9.06

used datasets are reported in Table 6. Each user's review history is arranged into a single sequence ordered by timestamp. For all methods, the commonly used leave-one-out strategy is adopted for data splitting, following previous works (Kang & McAuley, 2018; Rajput et al., 2023): in each sequence, the most recent interaction is used for testing, the second most recent for validation, and the remainder for training.

D.2 Baseline Description

We compare SPRINT against representative and SOTA baselines spanning three families.

Item ID-based Methods.

- **GRU4Rec** (Hidasi et al., 2016) pioneers RNN-based session recommendation, employing GRU units to encode the sequence of intra-session interactions for next-item prediction.
- **SASRec** (Kang & McAuley, 2018) adopts self-attention encoder that adaptively weighs previously interacted items to capture sequential dynamics. This attains the long-range modeling capacity of RNNs while retaining the parallel training efficiency of Transformers.
- **BERT4Rec** (Sun et al., 2019) replaces left-to-right decoding with a deep bidirectional Transformer trained under a Cloze (masked-item) objective.
- **PreferDiff** (Liu et al., 2025c) is a diffusion-based recommender that recasts the BPR objective into a log-likelihood ranking loss over multiple negative samples, optimized via a variational upper bound with a cosine (rather than MSE) reconstruction error.
- **FAVE** (Shi et al., 2026) is a flow matching-based recommendation model that learns a direct user-to-item trajectory via an average-velocity formulation, thereby avoiding the prior mismatch and iterative-solver redundancy of the noise-to-data paradigm.

Semantic ID-based Methods (Autoregressive).

- **TIGER** (Rajput et al., 2023) assigns each item a semantic ID produced by an RQ-VAE over content embeddings, and trains a Transformer to autoregressively generate the next item's semantic ID.
- **ActionPiece** (Hou et al., 2025b) performs context-aware tokenization of action sequences, merging co-occurring feature patterns in the spirit of BPE so that identical actions receive different tokens depending on their surrounding context. The resulting context-sensitive vocabulary improves autoregressive generation over context-agnostic SID tokenizers.
- **COBRA** (Yang et al., 2025b) unifies sparse and dense retrieval through cascaded generation: it first produces a sparse semantic ID and then, conditioned on it, a dense vector, both refined end-to-end in a coarse-to-fine manner. At inference, a BeamFusion strategy combines beam search with nearest-neighbor scores to balance accuracy and diversity.
- **SID-MLP** (Guo et al., 2026b) proposes an MLP-centric distillation framework for efficient GR inference. It retains one-layer attention module to obtain the first token and then uses cascaded MLPs to generate the remaining tokens sequentially.

Table 7: Hyperparameter settings across datasets.

Hyperparameter	Sports	Beauty	Toys	Scientific	Instruments	Games	Yelp	Arts
learning rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
warmup steps	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
dropout rate	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2
weight decay	0	0	0	0	0	0	0	0.05
d_m	256	256	256	256	256	256	512	512
d_{ff}	1024	1024	1024	1024	1024	1024	2048	1024
num_heads	4	4	4	4	4	4	4	8
L	4	4	4	4	4	4	4	4
M	256	256	256	256	256	256	256	256
encoder layers	1	1	1	1	1	1	1	1
decoder layers	4	4	4	4	4	4	4	4
ρ	1	1	1	0.75	0.75	1	1	1
num_neg_samples	100	100	100	100	100	100	100	1000
max_seq_length	20	20	20	20	20	20	20	20
max_epochs	150	150	150	150	150	150	150	150
early stop patience	10	10	10	10	10	10	10	10

Semantic ID-based Methods (Non-autoregressive).

- **RPG** (Hou et al., 2025a) generates all tokens of a long, unordered semantic ID in parallel within a single decoding step, decoupling the number of decoding passes from ID length.
- **LLaDA-Rec** (Shi et al., 2025) casts SID generation as discrete masked diffusion, denoising SID tokens in parallel and in a flexible order rather than strictly left-to-right. This bidirectional, any-order formulation better respects the joint structure of the tokens that collectively identify an item.
- **DiffGRM** (Liu et al., 2026b) replaces the autoregressive decoder with a masked discrete diffusion model, enabling bidirectional context and any-order parallel generation of SID tokens. It further introduces parallel semantic encoding, on-policy coherent noising that concentrates supervision on hard-to-predict tokens, and confidence-guided parallel denoising for diverse top-K candidate generation.
- **OneGR** (Wang et al., 2026b) produces position-wise scores in one pass, and then adopt A* search over valid SIDs to retrieval top-K valid SIDs.

D.3 More Implementation Details

We employ $L = 4$ codebooks, each containing $M = 256$ codewords (*i.e.*, tokens) following existing non-autoregressive GR models (Shi et al., 2025; Liu et al., 2026b; Wang et al., 2026b). This means each item is discretized into $L = 4$ tokens. The tokens are derived using OPQ following previous work (Liu et al., 2026b; Wang et al., 2026b). In the bidirectional Transformer, the token embedding dimension is $d_m = 256$ following previous work (Shi et al., 2025; Li et al., 2023). The numbers of encoder layer and decoder layer are 1 and 4, respectively. The feed-forward layer dimension d_{ff} is 1024 and the number of attention heads is 4. The MLP to derive $\mathbf{u}$ and $\mathbf{v}_c$ is a two layer architecture (*i.e.*, $1024 \to 256 \to$ ReLU $\to 256$). Following (Liu et al., 2026b), all 4 tokens of an item are mapped into a 256-dimension embedding before being fed into the encoder. Training/Validation/Test batch sizes are set to 1024/256/256. The maximum length of user history is fixed at $n = 20$. For fair comparison, we use `sentence-t5-base` (Ni et al., 2022) as the semantic encoder to derive items' content embeddings in the tokenization stage following prior work (Liu et al., 2026b; Wang et al., 2026b; Hou et al., 2025a), so that no method benefits from a stronger semantic encoder. Model parameters are optimized using AdamW (Loshchilov & Hutter, 2017), with the learning rate searched over $\{1\text{e}{-4}, 5\text{e}{-4}, 1\text{e}{-3}, 5\text{e}{-3}, 1\text{e}{-2}\}$. The maximum number of training epochs is set to 150. Following previous work (Liu et al., 2026b), we use early stopping with patience 10 to stop training if the validation score $0.8 \times \text{N@10} + 0.2 \times \text{R@10}$ does not improve for 10 consecutive epochs. The hyperparameter values used in our implementation are reported in Table 7.

Settings for efficiency comparison. For fair comparison of model efficiency, we set all methods' training/val/test batch sizes to 1024/256/256, maximum sequence length to 20, beam width to 10

Table 8: Performance comparison between baselines and our proposed method. The best performance is in **bold** and the second-best performance is underlined. "*" denotes the improvement over the second-best performance is statistically significant ($p < 0.05$) according to a paired t-test.

Method	Sports		Beauty		Toys		Scientific		Instruments		Games	
	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
Item ID-based												
GRU4Rec	0.0204	0.0110	0.0283	0.0137	0.0176	0.0084	0.0272	0.0156	0.0453	0.0246	0.0712	0.0387
SASRec	0.0350	0.0192	0.0605	0.0318	0.0675	0.0374	0.0379	0.0197	0.0525	0.0273	0.0823	0.0421
BERT4Rec	0.0191	0.0099	0.0347	0.0170	0.0203	0.0099	0.0264	0.0134	0.0412	0.0211	0.0530	0.0267
PreferDiff	0.0405	0.0218	0.0660	0.0388	0.0851	0.0483	0.0216	0.0132	0.0294	0.0166	0.0501	0.0287
FAVE	0.0455	0.0244	0.0847	0.0489	0.0855	0.0520	0.0359	0.0188	0.0518	0.0271	0.0858	0.0451
Semantic ID-based (autoregressive)												
TIGER	0.0400	0.0225	0.0648	0.0384	0.0712	0.0432	0.0446	0.0236	0.0566	0.0300	0.0823	0.0442
ActionPiece	0.0500	0.0264	0.0775	0.0424	0.0730	0.0396	0.0437	0.0227	0.0609	0.0318	0.0910	0.0486
COBRA	0.0434	0.0257	0.0725	0.0456	0.0781	0.0515	0.0424	0.0228	0.0509	0.0285	0.0750	0.0395
SID-MLP	0.0417	0.0225	0.0715	0.0391	0.0600	0.0314	0.0459	0.0244	0.0612	0.0327	0.0916	0.0486
Semantic ID-based (non-autoregressive)												
RPG	0.0463	0.0263	0.0809	0.0464	0.0869	0.0490	0.0395	0.0218	0.0545	0.0300	0.0853	0.0485
LLaDA-Rec	0.0432	0.0233	0.0747	0.0429	0.0752	0.0423	0.0474	0.0256	0.0623	0.0337	0.0942	0.0517
DiffGRM	0.0550	0.0305	0.0876	0.0502	0.0834	0.0524	0.0605	0.0337	0.0659	0.0356	0.0950	0.0515
OneGR	0.0574	0.0332	0.0914	0.0519	0.0874	0.0538	0.0507	0.0294	0.0526	0.0299	0.0799	0.0451
SPRINT	**0.0621***	**0.0347***	**0.0987***	**0.0552***	**0.0998***	**0.0594***	**0.0656***	**0.0369***	**0.0713***	**0.0386***	**0.1001***	**0.0534***
Improv.	8.19%	4.52%	7.99%	6.36%	14.19%	10.41%	8.43%	9.50%	8.19%	8.43%	5.37%	3.29%

Table 9: Training time per epoch (s) comparison and the speedup of our method over baselines.

Methods	Sports		Beauty		Toys		Scientific		Instruments		Games	
	Cost	Speedup	Cost	Speedup	Cost	Speedup	Cost	Speedup	Cost	Speedup	Cost	Speedup
TIGER	122.01	6.67×	86.54	6.92×	70.58	6.73×	170.04	6.63×	220.70	6.55×	351.27	6.69×
ActionPiece	122.59	6.70×	98.45	7.87×	80.57	7.68×	170.16	6.64×	241.50	7.17×	403.22	7.68×
RPG	21.39	1.17×	16.20	1.30×	13.76	1.31×	29.27	1.14×	37.43	1.11×	62.83	1.20×
LLaDA-Rec	202.50	11.07×	143.70	11.49×	118.80	11.33×	276.00	10.77×	368.90	10.95×	573.40	10.92×
DiffGRM	69.33	3.79×	47.29	3.78×	39.03	3.72×	91.36	3.56×	122.52	3.64×	195.26	3.72×
OneGR	102.22	5.59×	70.27	5.62×	57.91	5.52×	139.15	5.43×	189.26	5.62×	284.66	5.42×
SPRINT	**18.29**	-	**12.51**	-	**10.49**	-	**25.63**	-	**33.68**	-	**52.49**	-

if applicable, token embedding size to 256. Other settings are set according to the best setting specific to datasets in their original paper. These settings are only used for efficiency comparison. In accuracy comparison, these hyperparameters follow the original papers and are tuned per dataset.

D.4 More Comparison Results ($K = 10$)

We report the performance comparison of R@10 and N@10 in Table 8. Similar to the overall comparison reported in Table 2, our proposed model SPRINT also consistently outperform baseline methods in terms of R@10 and N@10.

D.5 Training Efficiency Analysis

As shown in Table 9, we have the following observation:

- SPRINT's training time cost is also lower than that of existing AR and NAR GR models. It can be attributed to our single-step generation formulation and the lightweight model architecture, 1-layer encoder and 4-layer decoder, with no auxiliary mask-scheduling (*e.g.*, history-aware mask position allocation (Mu et al., 2026)) or data-augmentation machinery (*e.g.*, on-policy coherent noising (Liu et al., 2026b)).
- Besides, we also compare the convergence curves of SPRINT and the fastest baseline method RPG. As shown in Figure 5, SPRINT converges much faster than RPG, generally requiring 20-40 epochs. Hence, this further indicates that our proposed method SPRINT can be trained efficiently.

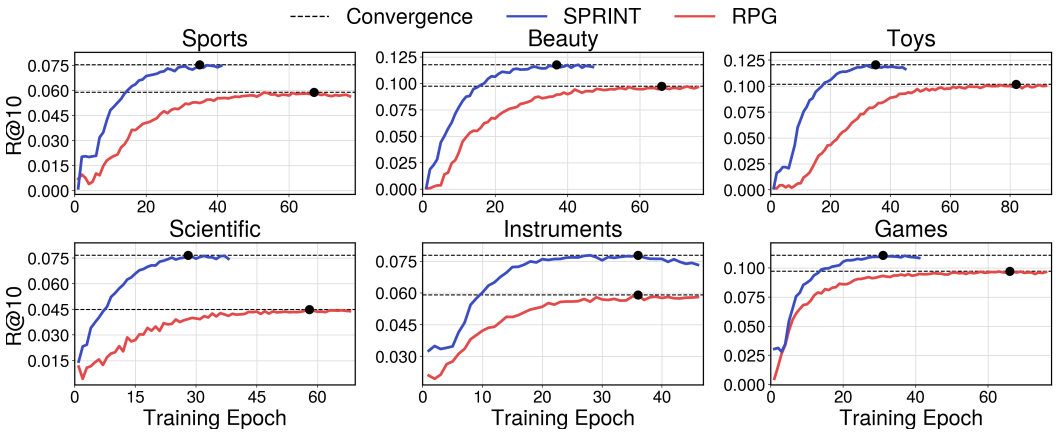

Figure 5: Training convergence comparison with RPG on six datasets.

D.6 Comparison between SPRINT and Other Efficiency-oriented Methods

In addition to efficiency comparison with existing accuracy-oriented AR and NAR GR models, we also compare our model with efficiency-oriented GR models. To achieve this, we select the following method for comparison:

- **AtSpeed** (Lin et al., 2025) accelerates LLM-based generative recommendation via speculative decoding. Since a ranked list requires every top-K beam sequence of the target LLM to be drafted correctly, it aligns the draft model with the target model's top-K sequences (AtSpeed-S), and further relaxes verification so that high-probability non-top-K drafts can also be accepted (AtSpeed-R).
- **EARN** (Yang et al., 2025a) reduces the computation and KV-cache cost of LLM-based generative recommendation. Observing that early layers exhibit dense, informative attention while later layers are largely redundant, it uses the first few layers to compress the interaction history into register tokens placed at the sequence boundaries, and processes only these register tokens in the remaining layers.
- **NEZHA** (Wang et al., 2026d) is a speculative decoding architecture for industrial generative recommendation that dispenses with a separate draft model and model-based verifier. It attaches a lightweight autoregressive draft head to the main model for self-drafting, and verifies drafted candidates by a hash lookup against the set of valid SIDs rather than by calling the target model.
- **SID-MLP** (Guo et al., 2026b) distills an autoregressive SID-based recommender (*e.g.*, TIGER) into lightweight MLPs. Motivated by the observation that SID prediction becomes much easier after the first token, it freezes the Transformer encoder and replaces the attention-based decoder with prefix-conditioned MLP heads.

Table 10: Inference time cost (s) comparison across six datasets. The fastest is in **bold** and the second-fastest is underlined.

Methods	Sports	Beauty	Toys	Scientific	Instruments	Games
AtSpeed	64.10	41.88	34.97	91.83	103.74	170.44
EARN	76.07	47.62	41.91	109.58	124.79	204.45
NEZHA	17.57	11.03	9.58	25.17	28.37	46.85
SID-MLP	<u>5.18</u>	<u>3.18</u>	<u>2.86</u>	<u>7.11</u>	<u>7.67</u>	<u>12.75</u>
SPRINT	**2.65**	**1.52**	**1.28**	**4.08**	**4.62**	**7.55**

For all competitive methods, we use TIGER as their backbone model. As shown Table 10, SPRINT achieves the lowest inference cost across all six datasets, requiring only 1.28–7.55s to generate recommendations. AtSpeed and NEZHA accelerate autoregressive generative recommenders via

speculative decoding, while EARN reduces computation by compressing the interaction history into register tokens; nevertheless, all three retain a fundamentally sequential decoding process and remain far slower than SPRINT, which is about 24–29$\times$ faster than AtSpeed and EARN (*e.g.*, 2.65s vs. 64.10s and 76.07s on Sports) by predicting all tokens of an SID in a single step. Compared with NEZHA, which mitigates but does not eliminate autoregressive decoding, SPRINT is still about 6$\times$ faster. The closest competitor is SID-MLP, which distills an autoregressive recommender into lightweight MLP heads; although it likewise avoids attention-based sequential decoding, SPRINT remains consistently faster (*e.g.*, 1.28s vs. 2.86s on Toys) while delivering substantially higher recommendation accuracy (Table 2). These results demonstrate that the efficiency of SPRINT stems from its single-step generative formulation rather than merely from a compact architecture, making it well-suited for latency-sensitive, real-world recommendation scenarios.

D.7 Parameter Memory

Table 11 compares the parameter memory of representative generative recommenders across the autoregressive (AR) and non-autoregressive (NAR) paradigms. The vocabulary size of ActionPiece is set to 40,000 following default setting of original paper. When the SID length of RPG is set to 16, its memory is 8.53 M. However, in many datasets, RPG requires an SID length of 64, significantly increasing the memory to 23.69 M. Within these paradigms, SPRINT is the most lightweight, requiring only 5.59 M parameters, on par with the most compact baseline, DiffGRM (5.60 M), and roughly 2.5$\times$ smaller than TIGER. Crucially, SPRINT attains this minimal parameter footprint while delivering the best recommendation accuracy (Table 2), indicating that its single-step generative formulation improves effectiveness without incurring any additional parameter cost.

Table 11: Parameter memory comparison across paradigms.

Paradigms	Methods	#Parameters
AR	TIGER	13.97 M
	ActionPiece	9.58 M
NAR	RPG	8.53 M – 23.69 M
	LLaDA-Rec	6.85 M
	DiffGRM	5.60 M
	SPRINT	5.59 M

D.8 Experiments on More Diverse Datasets

Apart from the six datasets from Amazon used in the main text, we also evaluate the effectiveness of our method on more diverse datasets, including Yelp (Wang et al., 2024a) and Arts (Hou et al., 2026b). From the results reported in Table 12, we can find that SPRINT also consistently achieves the best performance on Yelp and Arts. This verifies the effectiveness and generalization of our method.

Table 12: Performance comparison between baselines and our proposed method. The best performance is in **bold** and the second-best performance is underlined. "*" denotes the improvement over the second-best performance is statistically significant ($p < 0.05$) according to a paired t-test.

Methods	Yelp				Arts			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
GRU4Rec	0.0216	0.0138	0.0377	0.0190	0.0169	0.0106	0.0281	0.0142
SASRec	0.0240	0.0167	0.0370	0.0209	0.0263	0.0143	0.0416	0.0192
TIGER	0.0235	0.0156	0.0376	0.0202	0.0264	0.0174	0.0409	0.0220
ActionPiece	0.0285	0.0182	0.0481	0.0251	0.0245	0.0165	0.0356	0.0200
RPG	0.0216	0.0147	0.0319	0.0180	0.0246	0.0168	0.0364	0.0206
LLaDA-Rec	0.0322	0.0218	0.0502	0.0276	0.0267	0.0173	0.0422	0.0223
DiffGRM	0.0272	0.0177	0.0460	0.0237	0.0414	0.0291	0.0522	0.0326
OneGR	0.0189	0.0124	0.0314	0.0164	0.0275	0.0197	0.0379	0.0231
SPRINT	**0.0350***	**0.0235***	**0.0548***	**0.0298***	**0.0427***	**0.0298***	**0.0613***	**0.0354***
Improv.	*8.70%*	*7.80%*	*9.16%*	*7.97%*	*3.14%*	*2.41%*	*17.43%*	*8.59%*

Table 13: Comparison between encoder-decoder and decoder-only architectures for SPRINT across six datasets.

Methods	Sports				Beauty				Toys			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
SPRINT (encoder-decoder)	**0.0411**	**0.0270**	**0.0621**	**0.0347**	**0.0675**	**0.0451**	**0.0987**	**0.0552**	0.0708	0.0501	0.0998	0.0594
SPRINT (decoder-only)	0.0393	0.0259	0.0611	0.0329	0.0660	0.0437	0.0971	0.0537	**0.0739**	**0.0531**	**0.1013**	**0.0618**
Methods	Scientific				Instruments				Games			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
SPRINT (encoder-decoder)	**0.0444**	**0.0301**	**0.0656**	**0.0369**	0.0462	0.0305	0.0713	0.0386	0.0636	**0.0427**	**0.1001**	**0.0534**
SPRINT (decoder-only)	0.0434	0.0294	0.0649	0.0364	**0.0482**	**0.0320**	**0.0730**	**0.0400**	**0.0639**	0.0418	0.0998	0.0533

D.9 Performance of Decoder-only Architecture

To evaluate whether SPRINT can perform well on other architectures, we additionally implement it on a decoder-only backbone (4-layer decoder) and compare it against the default encoder-decoder variant. As shown in Table 13, SPRINT delivers strong performance under both architectures. The decoder-only architecture only performs slightly worse performance than the encoder-decoder architecture on Sports, Beauty, Scientific, and Games datasets, and even achieves better performance on Toys and Instruments dataset. This demonstrates that the effectiveness of SPRINT stems from our specific model designs rather than from any particular backbone, and that the method is readily transferable across architectural choices.

D.10 More Hyperparameter Analysis

Effect of ρ. We further probe the sensitivity to ρ on Sports, Beauty, and Toys datasets, reporting both R@5 and the best epoch (the epoch attaining peak validation performance) in Fig. 6. Two consistent trends emerge. First, recommendation accuracy steadily improves as ρ grows and peaks at $\rho = 1$ on all three datasets. This indicates that anchoring training at the all-mask source, which matches the inference condition, directly benefits single-step generation. Second, and more strikingly, ρ substantially accelerates convergence. The best epoch drops sharply as ρ grows, so a larger ρ reaches the optimum with far fewer training epochs. Taken together, ρ improves effectiveness and efficiency simultaneously. It yields better accuracy while greatly reducing the training cost, and $\rho = 1$ offers the best trade-off on these three datasets.

Effect of N. We further examine the effect of N on Sports, Beauty, and Toys datasets, reporting both R@5 and the training time per epoch in Fig. 7. Two consistent trends emerge. First, recommendation accuracy rises steadily as N grows to 100 and then gains more slowly. Beyond 100, R@5 keeps inching up on Sports, while it peaks at $N = 500$ and slightly declines at $N = 1000$ on Beauty and Toys. This suggests that a moderate number of negatives already provides most of the contrastive signal, whereas excessive negatives bring little new information and may even introduce noise. Second, the training time stays nearly flat up to $N = 100$ but grows sharply afterwards, since the cost of scoring negative SIDs becomes dominant. Taken together, $N = 100$ captures most of the accuracy gain at negligible extra cost, which is consistent with the observations on other datasets.

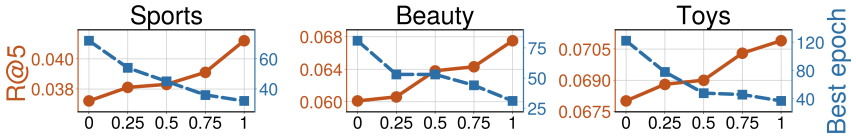

Figure 6: Effects of ρ on Sports, Beauty, and Toys datasets.

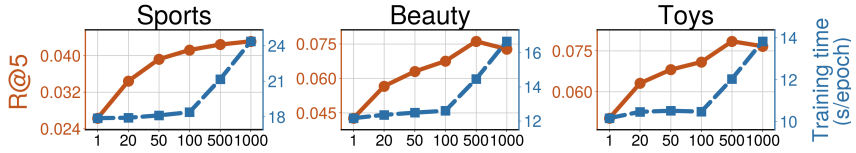

Figure 7: Effects of N on Sports, Beauty, and Toys datasets.

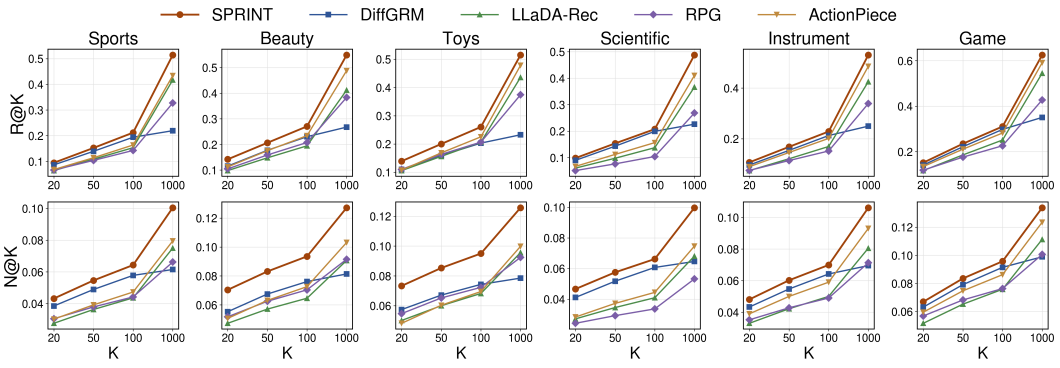

Figure 8: Performance comparison between SPRINT and other generative recommendation methods under different K values in range $\{20, 50, 100, 1000\}$.

Table 14: Comparison of TIGER based on RQ-VAE and OPQ.

Methods	Sports				Beauty				Toys			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
TIGER (RQ-VAE)	**0.0264**	**0.0181**	**0.0400**	**0.0225**	**0.0454**	**0.0321**	**0.0648**	**0.0384**	**0.0521**	**0.0371**	**0.0712**	**0.0432**
TIGER (OPQ)	0.0220	0.0140	0.0375	0.0190	0.0390	0.0262	0.0603	0.0330	0.0346	0.0223	0.0584	0.0299
Methods	Scientific				Instruments				Games			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
TIGER (RQ-VAE)	**0.0282**	**0.0183**	**0.0446**	**0.0236**	**0.0359**	**0.0233**	**0.0566**	**0.0300**	**0.0529**	**0.0348**	**0.0823**	**0.0442**
TIGER (OPQ)	0.0257	0.0166	0.0412	0.0216	0.0339	0.0219	0.0526	0.0278	0.0496	0.0321	0.0794	0.0416

D.11 Performance Comparison Under Larger K

Small cutoffs ($K \in \{5, 10\}$) probe only the head of the ranking. To assess whether SPRINT also orders items well deeper into the list, Figure 8 compares it against DiffGRM, LLaDA-Rec, RPG, and ActionPiece on R@K and N@K for $K \in \{20, 50, 100, 1000\}$ across all six datasets. SPRINT remains the best method at every evaluated K, on every dataset, and under both metrics, and its margin over the strongest baseline does not erode as K grows—indeed widening in most settings. We attribute this to SPRINT scoring each candidate SID jointly rather than committing to tokens one position at a time: joint scoring keeps the entire candidate pool well-ordered far from the top, whereas the error accumulation of token-by-token or iterative decoding degrades the tail of the ranking. This robustness at large K is valuable for systems that must surface many items per user.

D.12 Performance of TIGER using OPQ as the tokenizer

We replace the RQ-VAE used in TIGER (Rajput et al., 2023) with an OPQ tokenizer and report the results in Table 14. We observe that OPQ consistently degrades TIGER's performance across all six datasets and all metrics. This is because the left-to-right generation of TIGER inherently requires the SIDs to exhibit sequential dependencies: RQ-VAE quantizes an item embedding in a coarse-to-fine residual manner, so that each codeword is conditioned on the residual left by its predecessors, yielding a hierarchical token order that aligns naturally with the autoregressive decoding of TIGER. OPQ, in contrast, partitions the embedding into mutually independent subspaces and quantizes each in parallel; the resulting codewords carry no inherent ordering and are effectively permutation-invariant. As a result, the autoregressive decoder cannot exploit any conditional structure among the generated tokens, which explains the observed performance drop.

E Algorithms

The algorithms of SPRINT's training and inference stage are present in Algorithms 1 and 2.

Algorithm 1 Training of SPRINT

Require: Training set $\mathcal{D} = \{(\mathcal{H}, x_1)\}$; sampled SID set $\mathcal{S}$; codebooks $\{\mathcal{C}^l\}_{l=1}^{L}$
Ensure: Network parameters θ
Tokenize all items into L-token SIDs with OPQ ▷ Tokenization
while not converged **do**
  Sample a mini-batch $\{(\mathcal{H}, x_1)\} \subset \mathcal{D}$
  Obtain partially masked sequence x_r according to r
  Encode $\mathcal{H}$ and decode from L mask queries to obtain $\bar{p}_{1|r,1}(\cdot \mid x_r, \mathcal{H}, \theta) \in \mathbb{R}^{L \times M}$ and $\{\mathbf{h}^l\}_{l=1}^{L}$
  $\mathbf{u} \leftarrow \mathrm{MLP}\big(\mathrm{Cat}(\mathbf{h}^1, \ldots, \mathbf{h}^L)\big)$ ▷ User/Sequence-side representation
  $\mathbf{v}_c \leftarrow \mathrm{MLP}\big(\mathrm{Cat}(\mathbf{e}_{c^1}, \ldots, \mathbf{e}_{c^L})\big)$ for all $c \in \mathcal{S}$ ▷ SID-side representations
  Compute token-level score $\sum_{l=1}^{L} \log \bar{p}_{1|r,1}(c^l \mid x_r, \mathcal{H}, \theta)$ ▷ Single-step token generation
  Compute SID-level score $\langle \mathbf{u}, \mathbf{v}_c \rangle$ for all $c \in \mathcal{S}$
  Compute the dual-level flow contrastive loss $\mathcal{L}$ by Eq. 10
  Update θ with AdamW
end while
return θ

Algorithm 2 Inference of SPRINT.

Require: User history $\mathcal{H}$; trained parameters θ; the set of valid trajectories in the SID prefix tree $\mathcal{T}$; recommendation list length K
Ensure: Ranked list of K items
Offline: precompute $\mathbf{v}_c$ for all $c \in \mathcal{T}$ ▷ User/Sequence-independent, computed once
$x_0 \leftarrow [\mathbb{m}, \ldots, \mathbb{m}]$ ▷ Generate from all-mask sequence
Run one forward pass of the encoder-decoder on $(\mathcal{H}, x_0)$ to obtain $\bar{p}_{1|0,1}(\cdot \mid x_0, \mathcal{H}, \theta) \in \mathbb{R}^{L \times M}$ and $\mathbf{u}$ ▷ Single-step generation
Compute the generation score of an SID using Eq. 11
return the K SIDs with the highest generation probability

F LIMITATIONS

This work aims to develop a single-step SID generation paradigm from a new perspective of average probability velocity. It still has several limitations. First, although the average-velocity formulation is defined over a general interval $[r, t]$, SPRINT fixes the terminal time $t = 1$ and anchors most training sources at $r = 0$. As our experiments show, the best performance is mostly obtained with $\rho \in \{0.75, 1\}$, *i.e.*, when the large majority of sources match the strict single-step inference condition, while a small fraction of partially committed sources can still be beneficial. This suggests that intermediate intervals carry useful signal, but we only exploit them as auxiliary supervision for the single map from $r = 0$ to $t = 1$, and do not explore their behavior at inference or a tunable few-step variant, which we leave to future work. Second, since SPRINT decodes over all valid trajectories in SID prefix tree, its inference cost grows with the size of the item corpus. It is thus best suited to the ranking stage, where a candidate set has already been retrieved and the goal is to produce a fine-grained recommendation list, rather than to fully end-to-end settings that retrieve directly from billions of items. This limitation is also shared by many existing GR works (Wang et al., 2026b; Hu et al., 2026c; Yang et al., 2025b). Scaling single-step generation to such settings is left to future work.