arXiv:2605.11553v1 [cs.IR] 12 May 2026

TwiSTAR:Think Fast, Think Slow, Then Act, Generative Recommendation with Adaptive Reasoning

Shiteng Cao[♠] **Kaian Jiang**[♠] **Yunlong Gong**[♠] **Zhiheng Li**[♠]
[♠]Shenzhen International Graduate School, Tsinghua University
{caost24, jka23, gongyl25}@mails.tsinghua.edu.cn, zhhli@tsinghua.edu.cn

Abstract

Generative recommendation with Semantic IDs (SIDs) has emerged as a promising paradigm, yet existing methods apply a *fixed* inference strategy, either fast direct generation or slow chain-of-thought reasoning, uniformly across all user histories. This approach creates a trade-off: fast recommendation model produces suboptimal accuracy on hard samples, while always invoking slow reasoning incurs prohibitive latency and wastes computation on easy cases. To address this, we propose **TwiSTAR**, a framework that learns to *adaptively allocate reasoning effort* per user sequence. Our system equips an LLM with three complementary tools: a fast SID-based retriever, a lightweight candidate ranker, and a slow reasoning model that generates explicit rationales before recommending. Crucially, we inject collaborative commonsense into the slow model by transforming item-to-item knowledge into natural language explanations. A planner, trained through supervised warm-up followed by agentic reinforcement learning, dynamically decides which tool to invoke. Experiments on three datasets demonstrate that our method outperforms strong baselines, achieving consistent accuracy gains while reducing inference latency compared to uniform slow reasoning.

1 Introduction

Recommender systems play a fundamental role in mitigating information overload, connecting users with relevant items in domains ranging from e-commerce to content streaming. Their effectiveness directly impacts user satisfaction and business metrics. In recent years, generative recommendation has emerged as a powerful paradigm, framing sequential recommendation as an autoregressive generation task: given a user's interaction history, the model predicts the identifier of the next item. Among identifier choices, Semantic IDs (SIDs) obtained from residual quantization (e.g., RQ-VAE) have proven particularly attractive. SIDs form a lightweight vocabulary that naturally aligns with the token space of large language models (LLMs). This approach enables seamless integration with LLMs and achieves strong performance by leveraging both collaborative signals and language priors. However, existing methods suffer from a fundamental limitation: they apply a *uniform inference strategy* to all user histories, regardless of their complexity or ambiguity.

Two predominant inference modes exist. *Fast thinking* directly generates the target SID without explicit intermediate reasoning. It is computationally efficient and well-suited for routine patterns, but often fails on long-tail scenarios [1]. *Slow reasoning* first produces a chain-of-thought (CoT) rationale, then outputs the SID, improving accuracy on hard cases and enhancing interpretability [2]. OneRec-Think [3] adopts a "Think-Ahead" architecture for industrial deployment: the computationally expensive reasoning chain and the first two itemic tokens are generated offline and cached, while an online model completes the generation under prefix constraints. This decoupling ensures real-time responsiveness but still applies the same fixed two-stage reasoning to every request, lacking adaptive control. GREAM [4] supports direct sequence recommendation and sequential reasoning recommendation, but the mode must be pre-specified per request; the system cannot dynamically

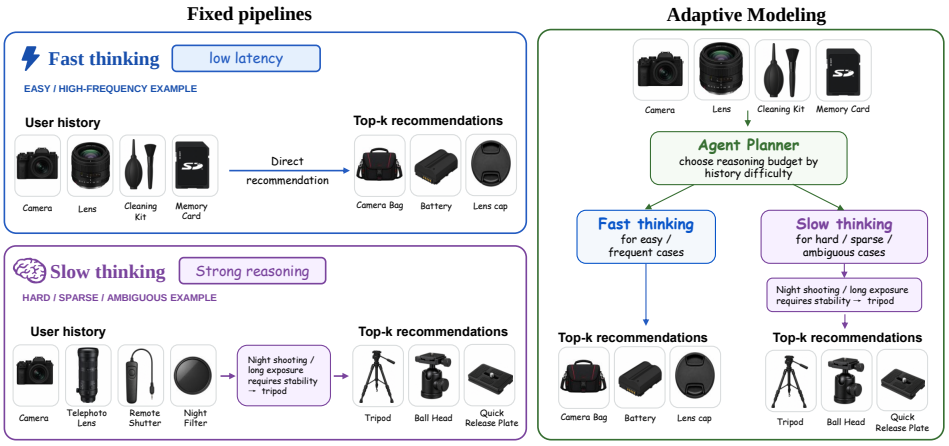

Figure 1: Illustration of fast vs. slow reasoning in generative recommendation. Left: fixed separate pipelines without adaptive selection. Right: our proposed framework with an agent planner that conditionally invokes fast retriever or slow reasoning model.

choose which mode to use based on the user's history. OxygenRec [5] introduces a fast-slow architecture, where a near-line LLM synthesizes contextual instructions and an efficient online backbone performs low-latency generation; however, the decision to invoke reasoning remains heuristic rather than learned. Consequently, an effective and efficient fusion of fast and slow reasoning, where the system adaptively allocates reasoning effort based on the difficulty of each user history, remains an open challenge. Despite recent progress in reasoning-enhanced recommenders [3–5], none adaptively allocate reasoning effort based on query difficulty.

This paper addresses a central question: *Can a generative recommender learn when to think fast and when to think slow?* Recent LLMs have begun to address a similar trade-off by moving from single-mode instruction following toward adaptive inference-time computation [6–8]. Inspired by the dual-system view of cognition, modern models increasingly support efficient responses for simple queries and deliberate reasoning for complex ones. As Figure 1 shows, we propose **TwiSTAR**, **T**wo-mode thinking, **S**low and fast **T**hinking, then **A**ct with tools to **Recommend**, a generative recommendation framework that learns to adaptively allocate reasoning effort. Our system is built upon a base model that aligns SID space with the LLM's text embedding space. This alignment grounds SID tokens in the textual contexts of their corresponding items, serving as the foundation for all subsequent components. Specifically, we develop three tools. A fast recommendation model directly predicts SIDs via efficient top-k retrieval, offering low-latency inference. A slow reasoning model first generates an explicit Chain-of-Thought rationale before outputting an SID; to instill genuine collaborative commonsense, we extract I2I relations from historical co-occurrence patterns, convert them into natural language explanation instructions, and activate the reasoning capability via RL. For cases where fast retrieval alone is insufficient but full slow reasoning is unnecessarily expensive, we introduce a ranking model, which retrieves a larger candidate set with the fast recommendation model and then applies a discriminative ranker to reorder the results. Finally, an agent planner decides, for each user sequence, which tool to invoke, enabling adaptive reasoning allocation tailored to the difficulty of the history.

Our main contributions are:

- A method to enhance collaborative reasoning in LLM-based recommenders by transforming I2I relations into natural language explanations.
- A two-stage training recipe (supervised imitation + agentic RL) that teaches the planner to selectively invoke slow reasoning, which unifies fast SID-based generation, slow reasoning, and an adaptive planner into a single agentic system. To our knowledge, this is the first work to learn when to engage reasoning in generative recommendation.
- Extensive experiments on three public datasets show that our approach consistently outperforms strong baselines in both ranking accuracy and efficiency.

2 Related Work

2.1 LLM as Recommender System

LLMs provide a natural interface for recommendation by representing users, items, and preferences in text. A straightforward direction encodes items with structured templates, thereby formulating recommendation within the native vocabulary space of LLMs. Prior text-based LLM recommenders instantiate this idea through product-description generation [9], instruction-style prompting [10], hierarchical attribute modeling [11], preference-optimized generation [12, 13], multimodal item summarization [14], and language-processing formulations that combine textual features with behavioral signals [15, 16, 1]. Despite their flexibility and interpretability, purely text-based representations are inefficient for large-scale recommendation and may fail to ground generated text to a unique catalog item. Semantic-ID-based generative recommendation addresses this limitation by representing each item as a compact sequence of discrete tokens and casting recommendation as catalog-grounded sequence generation. LC-Rec [17] highlights the gap between language semantics in LLMs and collaborative semantics. PLUM [18] adapts pretrained LLMs to industrial-scale generative recommendation through continued pre-training and task-specific fine-tuning. TS-Rec [19] investigates token-level SID semantics by initializing SID token embeddings with semantic-aware signals. These studies show that SID tokens should be semantically and collaboratively aligned rather than treated as opaque symbols. Recent work further introduces explicit reasoning into generative recommendation. OneRec-Think [3] and SIDReasoner [20] extend generative recommendation to reasoning through itemic alignment and reasoning activation. Other industrial systems [21–24] demonstrate that LLM-based recommenders are advancing toward reasoning-capable and production-scale deployment. Despite this progress, existing models lack adaptive reasoning allocation, applying the same inference strategy regardless of query difficulty. Consequently, they cannot balance efficiency and effectiveness across user histories of varying complexity. More severely, the abundance of easy samples that fast recommendation model can already handle tends to degrade the performance of slow reasoning models, leading to undesirable collapse.

2.2 Agent for Recommender System

Recent agentic recommender systems explore different aspects of process-level decision making. AMEM4Rec [25] introduces evolving memory for agentic LLM recommenders, capturing collaborative signals through cross-user memory evolution. RecGPT-V2 [26] develops a hierarchical multi-agent framework for industrial user-intent reasoning and multi-objective optimization. RecBot [27] and TalkPlay [28] further investigate interactive recommendation agents that translate natural language requests into actionable strategies or tool calls. Together, these works show that LLM recommenders are moving beyond passive item generation toward agents with adaptive reasoning, memory, interaction, and tool-use capabilities. However, existing agentic recommenders typically optimize one aspect of the recommendation process. They do not fully address how a SID-based generative recommender should jointly allocate inference behaviors under heterogeneous user histories. We formulate generative recommendation as an agentic inference problem: the model should decide not only *what* to recommend, but also *how* to recommend, whether to think fast, invoke a ranking tool, or think slow with collaborative reasoning.

3 Preliminaries

We first introduce Semantic IDs (SIDs) and formulate sequential recommendation as autoregressive generation over SIDs.

Given an item $i \in \mathcal{I}$ with textual description t_i, we encode it into a continuous embedding:

$$\mathbf{e}_i = \mathrm{Encoder}(t_i). \tag{1}$$

We then apply residual k-means to quantize $\mathbf{e}_i$ into a sequence of discrete codes. Let L denote the number of quantization layers and K the codebook size per layer. Starting from $\mathbf{r}_{i,0} = \mathbf{e}_i$, the quantization at layer j is given by

$$c_{i,j} = \arg\min_{k \in \{1,\dots,K\}} \|\mathbf{r}_{i,j-1} - \mathbf{v}_{j,k}\|_2^2, \qquad \mathbf{r}_{i,j} = \mathbf{r}_{i,j-1} - \mathbf{v}_{j,c_{i,j}}, \tag{2}$$

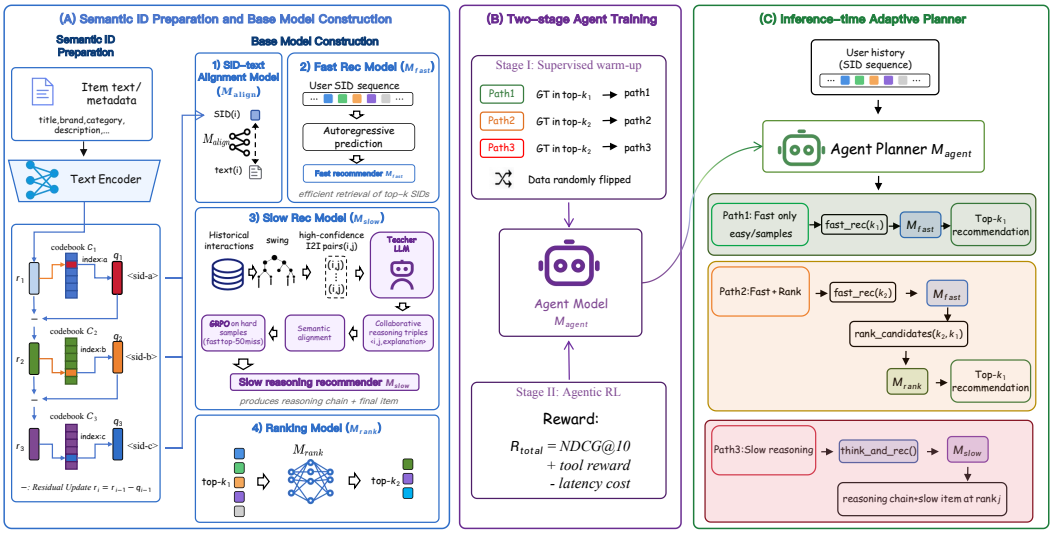

Figure 2: Our framework first extracts SIDs from item metadata and aligns them with text embeddings to ground semantic meaning. It then trains a fast SID-based retrieval model. Next, it injects collaborative commonsense into the slow recommendation model and activates reasoning via reinforcement learning. In the second stage, the planner is trained through supervised warm-up followed by reinforcement learning to balance accuracy and latency. This agentic design ensures that reasoning is applied only when beneficial, avoiding the inefficiency of uniform slow reasoning.

where $\mathbf{v}_{j,k}$ is the k-th codeword in the j-th codebook. The resulting Semantic ID of item i is

$$\mathrm{SID}(i) = [c_{i,1}, c_{i,2}, \ldots, c_{i,L}]. \tag{3}$$

For a user u, let $\mathcal{H}_u = [i_1, i_2, \ldots, i_{T-1}]$ denote the historical interaction sequence, and let i_T be the next item to predict. Replacing each item with its SID, sequential recommendation is formulated as autoregressive SID prediction:

$$p\big(\mathrm{SID}(i_T) \mid \mathrm{SID}(i_{<T})\big) = \prod_{j=1}^{L} p\big(c_{i_T,j} \mid c_{i_T,<j}, \mathrm{SID}(i_{<T})\big), \tag{4}$$

where $c_{i_T,<j} = [c_{i_T,1}, \ldots, c_{i_T,j-1}]$. This formulation allows a pretrained language model to perform recommendation via constrained token generation over valid SIDs.

4 Methodology

To overcome the limitations of uniform inference in generative recommendation, as illustrated in Figure 2, we propose a two-stage framework that learns to adaptively allocate reasoning effort per user sequence. The core idea is to equip a semantically aligned language model with three specialized tools: a fast SID-based retriever for routine patterns, a ranking model for candidate refinement, and a slow chain-of-thought model for hard cases that require explicit reasoning. A learned agent planner then dynamically decides which tool to invoke.

4.1 Aligned Base Model

The core challenge of using SIDs in a generative recommender is that initial SID tokens, which are obtained via residual k-means (Section 3), are discrete codes without inherent linguistic meaning. A pretrained language model cannot directly interpret these tokens, which hinders effective next-item prediction. To bridge this gap, we align SID token embeddings with the natural language semantics of their corresponding items, while keeping the main parameters of the LLM fixed.

Specifically, for each item i, we construct an item-level alignment sequence by pairing its SID with its textual metadata, such as its title, category, and other available metadata. We feed these item-level sequences into a pretrained causal language model and optimize the standard language modeling objective:

$$\mathcal{L}_{\text{align}} = -\sum_{i \in \mathcal{I}} \sum_{m=1}^{|x_i|} \log p_\theta(x_{i,m} \mid x_{i,<m}). \tag{5}$$

After alignment, SID tokens are grounded in the textual contexts of their corresponding items. As a result, when an SID appears in later recommendation sequences, the LLM can process it as a semantically meaningful item representation rather than an opaque discrete code. We denote the resulting model as $\mathcal{M}_{\text{align}}$, which serves as the foundation for the subsequent fast, slow, and agentic components.

4.2 Fast Rec Model and Ranking Model

After alignment, we train $\mathcal{M}_{\text{fast}}$ to perform sequential recommendation. Given a user's interaction history represented by their SIDs, the model autoregressively predicts the next item's SID:

$$\mathcal{L}_{\text{rec}} = -\sum_{u} \sum_{t} \sum_{\ell=1}^{L} \log p_{\mathcal{M}_{\text{fast}}}(c_{t,\ell} \mid c_{t,<\ell}, \text{SID}(i_{<t})). \tag{6}$$

Training data is constructed offline by truncating each user's chronological sequence into input–target pairs, where the input contains only the SID tokens of historical items. During inference, we constrain decoding to valid SIDs via a prefix trie and retrieve top-K candidates with beam search. This yields a fast recommendation model that retrieves candidate items efficiently.

The ranking model $\mathcal{M}_{\text{rank}}$ is built upon the Deep Interest Network [29], which captures the relevance between a user's historical behavior sequence and a candidate item through a local activation unit. Training data is constructed as follows: for each user, we retrieve K_2 candidates using $\mathcal{M}_{\text{fast}}$. If the ground truth is not among them, we add it as a positive sample and randomly sample $K_1 - 1$ negatives from the retrieved set. We monitor AUC on a validation set for early stopping.

4.3 Slow Rec Model: Collaborative Reasoning as Language Injection

The core idea is to transform implicit collaborative signals into explicit natural language reasoning chains, thereby teaching the model *what* co-occurrence patterns are and *why* they exist. Directly eliciting explanations is noisy and prone to hallucination: the true next item is only one of many plausible continuations, and the model often overfits to sequential id correlations without capturing robust statistical regularity. In contrast, I2I relations aggregate over many user sequences, providing denser, more reliable statistical signals. Learning from I2I explanations forces the model to internalize stable co-occurrence patterns, rather than memorizing fragile sequential shortcuts. For each related pair (i, j), we generate a reasoning instruction: "In collaborative filtering, item i and item j are highly correlated. Please explain why users who purchase i also tend to purchase j?"

A teacher LLM (e.g., Qwen3.5-397B-A17B [30]) produces a diverse set of explanations, forming triples $\langle i, j, \text{explanation} \rangle$. These triples are then mixed with semantic alignment data to fine-tune the aligned recommendation model. Formally, let $\mathcal{D}_{\text{collab}}$ be the set of (instruction, explanation) pairs derived from I2I relations. The training loss is a standard language modeling loss on $\mathcal{D}_{\text{collab}}$. After this stage, the model acquires *collaborative commonsense*: it does not memorize the I2I table but learns to verbalize statistical co-occurrence logic, enabling zero-shot reasoning on unseen pairs. We denote this semantically aligned and reasoning-enhanced model as $\mathcal{M}_{\text{base}}$, with all parameters fixed in the subsequent routing learning phase.

To further elicit reasoning before recommendation, we train a *think* model using GRPO. Only those user sequences where the fast recommendation model fails to hit the ground truth within its HitRate@50 predictions are selected as training data, this focuses the reasoning ability on long-tail and difficult samples. The detailed specifications of reward function are provided in Appendix A. The resulting model $\mathcal{M}_{\text{slow}}$ can produce explicit chain-of-thought reasoning before outputting a recommendation.

4.4 Planner Agent: Two-Stage Agent Training

We freeze $\mathcal{M}_{\text{fast}}$ and $\mathcal{M}_{\text{slow}}$ and optimize an agentic model $\mathcal{M}_{\text{agent}}$ using a two-stage procedure. The agentic model is initialized from the aligned base model; it takes a user's sequence as input and outputs tool calls. Tools include:

- `fast_rec(k)`: calls $\mathcal{M}_{\text{fast}}$ to beam search k SIDs.
- `rank_candidates(m, n)`: invokes the ranking model $\mathcal{M}_{\text{rank}}$, which takes m items as input and outputs the top-n items after reordering.
- `think_and_rec(j)`: invokes $\mathcal{M}_{\text{slow}}$ to produce a reasoning chain and then beam search j SIDs as the final recommendation.

We generate pseudo-labels for the agent by evaluating $\mathcal{M}_{\text{fast}}$ on a held-out set:

- Path 1 (fast only): If the ground truth is already retrieved by the fast recommendation model with a small candidate set size K_1, the agent invokes the fast recommendation model with that K_1.
- Path 2 (fast + ranking): For each user, we retrieve K_2 candidates using $\mathcal{M}_{\text{fast}}$. If the ground-truth item is included in the retrieved set, it is treated as the positive item. If the ground truth is absent, we insert it as the positive item and sample negatives from the retrieved candidates.
- Path 3 (slow reasoning): $\mathcal{M}_{\text{slow}}$ generates K_1 candidate SIDs through constrained beam search, and these candidates are used as the final top-K_1 list.

To encourage exploration during supervised warm-up, we randomly relabel a small fraction of Path-1 samples as higher-cost paths. The agentic model is trained with a standard cross-entropy loss on the sequence of tool calls and arguments.

After supervised warm-up, we further refine $\mathcal{M}_{\text{agent}}$ using reinforcement learning (e.g., GRPO or PPO). The reward is designed to balance final recommendation quality and process efficiency:

$$R_{\text{total}} = \underbrace{\text{NDCG@10(final ranked list)}}_{\text{outcome reward}} + \underbrace{\eta \cdot \mathbb{I}[\text{valid tool sequence}]}_{\text{process reward}} - \beta \cdot (\text{latency cost}). \quad (7)$$

5 Experiment

Datasets. We conduct experiments on three real-world recommendation datasets from the Amazon review benchmark [31]: BEAUTY, TOYS AND GAMES and SPORTS AND OUTDOORS. For evaluation, we adopt the leave one out strategy: the last interaction of each user is held out for testing, the second-last for validation, and all preceding interactions for training. We evaluate over the full item catalog rather than sampled negatives.

Baselines. We compare our proposed method against a diverse set of competitive baselines, covering traditional sequential recommenders, generative recommendation models, and reasoning-enhanced approaches. All baseline results are reproduced under the same experimental setting with consistent evaluation protocols. A detailed description of each baseline is provided in Appendix C.

Implementation Details. We implement our framework using Qwen3.5-4B as the base LLM. Semantic IDs are generated with $L = 3$ quantization layers and codebook size $K = 256$ per layer, yielding a vocabulary of $3 \times 256 = 768$ SID tokens. For the fast recommendation model, we use beam search with width $k = 50$. Due to space constraints, we report detailed implementations in Appendix E.1.

Evaluation Metrics. We adopt two standard ranking metrics: Recall@K and Normalized Discounted Cumulative Gain@K (NDCG@K). For each user, the model outputs a top-K recommendation list $\mathcal{R}_u(K)$. Recall@K measures whether the ground-truth item i_u appears in this list. NDCG@K further accounts for the ranking position.

6 Discussion

In this section, we address our five research questions: RQ1 confirms the overall superiority of our framework across diverse datasets, RQ2 quantifies the individual contributions of tools, RQ3 validates

that explicit I2I explanation injections instill genuine collaborative commonsense, RQ4 demonstrates that hard-sample training improves the usefulness of slow reasoning on difficult cases, and RQ5 reveals the planner's decisions substantially align with oracle routing patterns.

6.1 Overall Performance (RQ1)

Table 1: Overall performance comparison of our method and baselines on three datasets. The best results are in **bold** and the second-best results are underlined.

Dataset	Metric	HGN	GRU4Rec	SASRec	TIGER	HSTU	OneRec-Think	Ours
Beauty	R@5	0.0325	0.0392	0.0397	0.0409	0.0418	0.0557	**0.0609**
	R@10	0.0531	0.0585	0.0606	0.0622	0.0648	0.0770	**0.0880**
	N@5	0.0197	0.0263	0.0258	0.0267	0.0280	0.0390	**0.0415**
	N@10	0.0267	0.0325	0.0320	0.0337	0.0352	0.0461	**0.0504**
Sports	R@5	0.0188	0.0190	0.0199	0.0219	0.0263	0.0281	**0.0324**
	R@10	0.0316	0.0312	0.0306	0.0342	0.0348	0.0401	**0.0476**
	N@5	0.0114	0.0122	0.0107	0.0137	0.0168	0.0188	**0.0214**
	N@10	0.0155	0.0157	0.0146	0.0179	0.0221	0.0228	**0.0257**
Toys	R@5	0.0327	0.0330	0.0447	0.0338	0.0365	0.0553	**0.0594**
	R@10	0.0522	0.0491	0.0621	0.0546	0.0561	0.0774	**0.0828**
	N@5	0.0193	0.0223	0.0300	0.0210	0.0244	0.0389	**0.0425**
	N@10	0.0255	0.0279	0.0357	0.0277	0.0308	0.0461	**0.0505**

Table 1 reports the overall performance comparison between various baseline models and our proposed method on three Amazon review datasets. Across all datasets and metrics, our method consistently outperforms all baselines. Among the baselines, OneRec-Think shows the strongest overall performance and consistently ranks second, although it consumes more computational resources during both training and inference. These results demonstrate the effectiveness of our proposed approach in capturing user preferences and generating accurate recommendations across diverse product categories.

6.2 Ablation Study (RQ2)

To quantify the contribution of each component, we compare the full model against five variants on the Beauty dataset: (1) **w/o Ranking** (ranker removed); (2) **w/o Slow Reasoning** (slow model removed); (3) **w/o Slow Reasoning and Ranking** (both removed); (4) **w/o Alignment, Slow Reasoning and Ranking** (plus random SID initialization); (5) **Slow Reasoning Only w/o Planner** (slow model applied to every sequence).

Removing the ranking model (w/o Ranking) degrades R@10 from 0.0880 to 0.0820 and N@10 from 0.0504 to 0.0479, while only slightly reducing cost (0.94×). This indicates that the ranking tool provides a non-trivial accuracy boost by refining the fast recommendation model's candidate set, yet its own overhead is modest. Removing slow reasoning (w/o slow reasoning) causes a smaller drop, but its cost reduces dramatically. This shows that the slow model is expensive but brings substantial gains when invoked selectively. When both slow reasoning and ranking are removed (w/o slow reasoning and Ranking), performance falls further, confirming that each tool contributes uniquely to the final accuracy. Aligning SID tokens with natural language semantics is the bedrock of our framework. Without alignment, even the fast recommendation model's retrieval becomes severely impaired, and the reasoning capability cannot be activated. The slow reasoning Only variant, which applies the slow reasoning model uniformly without the planner, achieves lower performance than the full model while incurring more cost. (All reported cost values are normalized to the full model's total inference latency, which already includes the planner's own decision-making overhead.) This does not imply that slow reasoning is universally superior. Since $\mathcal{M}_{\text{slow}}$ is intentionally specialized on hard samples, applying it to easy histories may introduce unnecessary generation noise and degrade direct collaborative matching. Therefore, the full model with the planner delivers better accuracy with much lower average latency than always using slow reasoning, even after accounting for the planner's extra computation. Absolute latency and throughput numbers (in milliseconds and queries

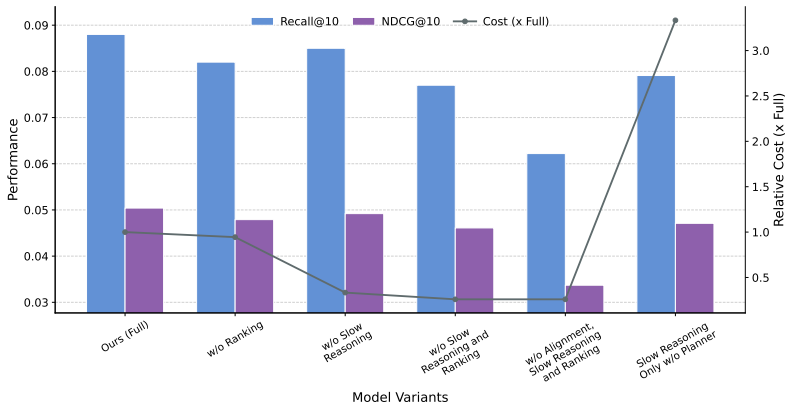

Figure 3: Recall@10, NDCG@10 and relative inference cost (normalized to the full model) of different ablation variants on the Beauty dataset. The full model achieves the best performance while maintaining moderate cost. Removing any component degrades accuracy, and applying slow reasoning uniformly (without planner) incurs more time cost with lower accuracy.

per second) along with a detailed breakdown of the planner's overhead are reported in Appendix E. In summary, the ablation study validates that all components are essential. Moreover, the adaptive planner is critical for translating the raw capability of slow reasoning into an efficient and effective system.

6.3 Does Explicit I2I Explanation Work? (RQ3)

A core claim of our method is that transforming I2I relations into natural language explanations and fine-tuning the LLM on these explanations injects genuine collaborative commonsense, rather than memorizing co-occurrence tables. To test this claim, we design a discriminative evaluation task that directly measures whether the model has internalized the underlying collaborative similarity structure. We first extract all unique I2I pairs from the Beauty dataset, resulting in 25,438 pairs (i, j) where item j is frequently co-purchased after item i. The first-level SID codes of i and j are identical for 26.25% of the I2I pairs. When both the first and second codes match, this occurs for only 3.65% of the pairs. For each such pair, we randomly sample 9 distractor items from the item catalog and construct a candidate set of 10 items: the true associated item j plus 9 negatives. The model is then presented with item i (described by its SID and optionally its textual metadata) and the list of 10 candidate SIDs. It must select the candidate that is most collaboratively similar to item i. We report the accuracy of selecting the ground-truth j.

We evaluate two model variants: the aligned model $\mathcal{M}_{\text{align}}$ (without explicit I2I explanation training) and the slow recommendation model $\mathcal{M}_{\text{slow}}$ (after fine-tuning on I2I explanation instructions). The base model achieves only 49% accuracy, indicating that alignment alone does not capture fine-grained collaborative relationships. After injecting I2I explanations, accuracy jumps to 84%. These results suggest that I2I explanation tuning helps the model better capture collaborative associations beyond the aligned SID representation.

6.4 Why Does Thinking Need Hard Samples? (RQ4)

We first train the slow reasoning model on the *entire* training set. Under this setting, the think model becomes *underperforming*: it fails to handle ambiguous requests better than the fast non-think model. Specifically, we define *easy samples* as those whose history contains less than two second-level category (about 30% of the entire dataset), and *hard samples* as those with three or more distinct second-level categories. As shown in Table 2, on the hard test set, the think model trained on full data is even slightly *lower* than the fast recommendation model. This confirms that mixing easy and hard samples during training leads the model to fail to develop genuine reasoning ability for difficult cases.

In contrast, selective hard-sample training yields a modest but consistent improvement on hard cases, suggesting that reasoning supervision is more useful when concentrated on samples where direct generation fails. A concrete case study is provided in Appendix D to illustrate this phenomenon.

Table 2: Think model trained on full data or selective samples (Beauty dataset).

Model	Hard samples (R@10)	Easy Samples (R@10)
Non-think (fast)	0.0706	0.0760
Think (trained on full data)	0.0692	**0.0782**
Think (trained on hard samples)	**0.0723**	0.0735

6.5 Planner Decision Analysis (RQ5)

We analyze the decisions made by the learned planner $\mathcal{M}_{\text{agent}}$ on the Beauty test set. The planner is given a user history and must choose among three tools: (1) fast retrieval only (`fast_rec`), (2) fast retrieval plus ranking (`fast_rec+rank`), and (3) slow reasoning (`think_and_rec`). The planner invokes `fast_rec` only for 62.3% of sequences, `fast_rec+rank` for 27.1%, and `think_and_rec` for 10.6%. To understand whether these decisions align with difficulty, we compare the planner's choices with an oracle that knows the performance of each tool. For each sequence, we compute the optimal tool as the one that yields the highest NDCG@10 (ties broken by lowest latency). The planner achieves 78.3% agreement with the oracle. The most common mismatch is choosing `fast_rec` when the oracle would prefer `fast_rec+rank` (13.4% of all sequences), i.e., regret due to underestimating the need for ranking. Conversely, the planner rarely chooses slow reasoning when fast only would suffice (1.7%). The planner successfully identifies difficult sequences: among the 10.6% where slow reasoning is invoked, the ground truth is not in the fast recommendation model's top-50 for 86.2% of them, confirming that the planner targets genuine hard cases. Compared to a heuristic baseline that triggers slow reasoning when the number of distinct categories in the user's history exceeds a tuned threshold, our learned planner achieves higher accuracy (NDCG@10: 0.0504 vs. 0.0489), confirming the value of learning the routing policy.

7 Conclusion and Future Work

We presented TwiSTAR, a generative recommendation framework that learns to adaptively allocate reasoning effort. At its core, we unify a fast SID-based retriever, a ranking model, and a slow reasoning model that generates explicit rationales. By transforming I2I patterns into language explanations, the slow recommendation model achieves reasoning. A two-stage training recipe, combining supervised imitation with reinforcement learning, teaches the planner when to invoke slow reasoning, when to rank, and when fast retrieval suffices. Extensive experiments on three Amazon review datasets demonstrate that our method consistently outperforms strong baselines, while ablation studies confirm the contribution of each component. Importantly, we show that the hard-sample training improves the usefulness of slow reasoning on difficult cases and that the planner's decisions reduce average inference latency compared to uniform slow reasoning. Beyond these empirical gains, our system offers a conceptual advance: it treats both the fast recommendation model (traditional recommendation systems) and the slow-thinking model (incorporates world knowledge and reasoning) as tools that an agent can execute. This design ensures seamless compatibility with existing recommendation approaches and opens up a novel paradigm for future recommendation systems, where heterogeneous reasoning modes are orchestrated by a learnable planner.

Despite these promising results, several limitations remain that point to directions for future work. First, our current implementation still incurs non-negligible overhead for the slow reasoning model, even though it is invoked sparingly; future work could explore early termination of reasoning chains. Second, the planner's decisions are learned offline; extending to online adaptation with bandit feedback could enable continuous improvement. Finally, while we focused on sequential recommendation, the idea of adaptive reasoning allocation may generalize to other generative tasks such as conversational recommendation or query rewriting, which we leave for future investigation.

References

[1] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. LLaRA: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 1785–1795, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657690. URL https://doi.org/10.1145/3626772.3657690.

[2] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

[3] Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, Yifei Hu, Qigen Hu, Xinchen Luo, Lejian Ren, Zixing Zhang, Qianqian Wang, Kuo Cai, Yunfan Wu, Hongtao Cheng, Zexuan Cheng, Lu Ren, Huanjie Wang, Yi Su, Ruiming Tang, Kun Gai, and Guorui Zhou. OneRec-Think: In-text reasoning for generative recommendation, 2025. URL https://arxiv.org/abs/2510.11639.

[4] Minjie Hong, Zetong Zhou, Zirun Guo, Ziang Zhang, Ruofan Hu, Weinan Gan, Jieming Zhu, and Zhou Zhao. Generative reasoning recommendation via LLMs, 2025. URL https://arxiv.org/abs/2510.20815.

[5] Xuegang Hao, Ming Zhang, Alex Li, Xiangyu Qian, Zhi Ma, Yanlong Zang, Shijie Yang, Zhongxuan Han, Xiaolong Ma, Jinguang Liu, Zhen Li, Zhida Jiang, Shusheng Wang, Ning Tang, Yanchen Qiao, Chenxiang Yang, Chen Sun, Jincheng Yuan, Chunhua Peng, Heng Hu, Peijun Yang, Baopeng Yuan, Caiyun Qiu, Zhaolong Xing, Haofei Yuan, Haipeng Zhang, Yuzhang Guo, Weijie Ding, Jiahua Gao, Hao Huang, Zhen Chen, Tongxuan Liu, and Pinghua Gong. OxygenREC: An instruction-following generative framework for e-commerce recommendation. *arXiv preprint arXiv:2512.22386*, 2025.

[6] An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[7] Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, Piero Kauffmann, Yash Lara, Caio César Teodoro Mendes, Arindam Mitra, Besmira Nushi, Dimitris Papailiopoulos, Olli Saarikivi, Shital Shah, Vaishnavi Shrivastava, Vibhav Vineet, Yue Wu, Safoora Yousefi, and Guoqing Zheng. Phi-4-reasoning technical report. *arXiv preprint arXiv:2504.21318*, 2025.

[8] Anthropic. Claude 3.7 Sonnet and Claude Code. https://www.anthropic.com/news/claude-3-7-sonnet, 2025. Accessed: 2026-04-29.

[9] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. M6-Rec: Generative pretrained language models are open-ended recommender systems, 2022. URL https://arxiv.org/abs/2205.08084.

[10] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. TALLRec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 1007–1014, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702419. doi: 10.1145/3604915.3608857. URL https://doi.org/10.1145/3604915.3608857.

[11] Wenlin Zhang, Chuhan Wu, Xiangyang Li, Yuhao Wang, Kuicai Dong, Yichao Wang, Xinyi Dai, Xiangyu Zhao, Huifeng Guo, and Ruiming Tang. LLMTreeRec: Unleashing the power of large language models for cold-start recommendations, 2024. URL https://arxiv.org/abs/2404.00702.

[12] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. A bi-step grounding paradigm for large language models in recommendation systems. *ACM Trans. Recomm. Syst.*, 3(4), April 2025. doi: 10.1145/3716393. URL https://doi.org/10.1145/3716393.

[13] Yuxin Chen, Junfei Tan, An Zhang, Zhengyi Yang, Leheng Sheng, Enzhi Zhang, Xiang Wang, and Tat-Seng Chua. On softmax direct preference optimization for recommendation. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

[14] Saketh Reddy Karra and Theja Tulabandhula. InteraRec: Interactive recommendations using multimodal large language models. In *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2024 Workshops, RAFDA and IWTA, Taipei, Taiwan, May 7–10, 2024, Proceedings*, page 32–43, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-981-97-2649-3. doi: 10.1007/978-981-97-2650-9_3. URL https://doi.org/10.1007/978-981-97-2650-9_3.

[15] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5), 2023. URL https://arxiv.org/abs/2203.13366.

[16] Zhixuan Chu, Hongyan Hao, Xin Ouyang, Simeng Wang, Yan Wang, Yue Shen, Jinjie Gu, Qing Cui, Longfei Li, Siqiao Xue, James Y Zhang, and Sheng Li. Leveraging large language models for pre-trained recommender systems, 2023. URL https://arxiv.org/abs/2308.10837.

[17] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1435–1448, 2024. doi: 10.1109/ICDE60146.2024.00118.

[18] Ruining He, Lukasz Heldt, Lichan Hong, Raghunandan Keshavan, Shifan Mao, Nikhil Mehta, Zhengyang Su, Alicia Tsai, Yueqi Wang, Shao-Chuan Wang, Xinyang Yi, Lexi Baugher, Baykal Cakici, Ed Chi, Cristos Goodrow, Ningren Han, He Ma, Romer Rosales, Abby Van Soest, Devansh Tandon, Su-Lin Wu, Weilong Yang, and Yilin Zheng. PLUM: Adapting pre-trained language models for industrial-scale generative recommendations. In *Proceedings of the ACM Web Conference 2026*, WWW '26, page 8093–8104, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400723070. doi: 10.1145/3774904.3792802. URL https://doi.org/10.1145/3774904.3792802.

[19] Jiawei Feng, Xiaoyu Kong, Leheng Sheng, Bin Wu, Chao Yi, Feifang Yang, Xiang-Rong Sheng, Han Zhu, Xiang Wang, Jiancan Wu, and Xiangnan He. Fine-grained semantics integration for large language model-based recommendation, 2026. URL https://arxiv.org/abs/2602.22632.

[20] 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, 2026. URL https://arxiv.org/abs/2603.23183.

[21] Fengxin Li, Yi Li, Yue Liu, Chao Zhou, Yuan Wang, Xiaoxiang Deng, Wei Xue, Dapeng Liu, Lei Xiao, Haijie Gu, Jie Jiang, Hongyan Liu, Biao Qin, and Jun He. Leadre: Multi-faceted knowledge enhanced llm empowered display advertisement recommender system, 2025. URL https://arxiv.org/abs/2411.13789.

[22] Edoardo D'Amico, Marco De Nadai, Praveen Chandar, Divita Vohra, Shawn Lin, Max Lefarov, Paul Gigioli, Gustavo Penha, Ilya Kopysitsky, Ivo Joel Senese, Darren Mei, Francesco Fabbri, Oguz Semerci, Yu Zhao, Vincent Tang, Brian St. Thomas, Alexandra Ranieri, Matthew N. K. Smith, Aaron Bernkopf, Bryan Leung, Ghazal Fazelnia, Mark VanMiddlesworth, Timothy Christopher Heath, Petter Pehrson Skiden, Alice Y. Wang, Doug J. Cole, Andreas Damianou, Maya Hristakeva, Reid Wilbur, Tarun Chillara, Vladan Radosavljevic, Pooja Chitkara, Sainath Adapa, Juan Elenter, Bernd Huber, Jacqueline Wood, Saaketh Vedantam, Jan Stypka, Sandeep Ghael, Martin D. Gould, David Murgatroyd, Yves Raimond, Mounia Lalmas, and Paul N. Bennett. Deploying semantic ID-based generative retrieval for large-scale podcast discovery at spotify, 2026. URL https://arxiv.org/abs/2603.17540.

[23] Marco De Nadai, Edoardo D'Amico, Max Lefarov, Alexandre Tamborrino, Divita Vohra, Mark VanMiddlesworth, Shawn Lin, Jacqueline Wood, Jan Stypka, Eliza Klyce, Keshi Dai, Timothy Christopher Heath, Martin D. Gould, Yves Raimond, Sandeep Ghael, Tony Jebara,

Andreas Damianou, Vladan Radosavljevic, Paul N. Bennett, Mounia Lalmas, and Praveen Chandar. A unified language model for large scale search, recommendation, and reasoning. *arXiv preprint arXiv:2603.17533*, 2026.

[24] Mingfu Liang, Yufei Li, Jay Xu, Kavosh Asadi, Xi Liu, Shuo Gu, Kaushik Rangadurai, Frank Shyu, Shuaiwen Wang, Song Yang, Zhijing Li, Jiang Liu, Mengying Sun, Fei Tian, Xiaohan Wei, Chonglin Sun, Jacob Tao, Shike Mei, Wenlin Chen, Santanu Kolay, Sandeep Pandey, Hamed Firooz, and Luke Simon. Generative reasoning re-ranker, 2026. URL https://arxiv.org/abs/2602.07774.

[25] Minh-Duc Nguyen, Hai-Dang Kieu, and Dung D. Le. AMEM4Rec: Leveraging cross-user similarity for memory evolution in agentic LLM recommenders. *arXiv preprint arXiv:2602.08837*, 2026.

[26] Chao Yi, Dian Chen, Gaoyang Guo, Jiakai Tang, Jian Wu, Jing Yu, Mao Zhang, Wen Chen, Wenjun Yang, Yujie Luo, Yuning Jiang, Zhujin Gao, Bo Zheng, Binbin Cao, Changfa Wu, Dixuan Wang, Han Wu, Haoyi Hu, Kewei Zhu, Lang Tian, Lin Yang, Qiqi Huang, Siqi Yang, Wenbo Su, Xiaoxiao He, Xin Tong, Xu Chen, Xunke Xi, Xiaowei Huang, Yaxuan Wu, Yeqiu Yang, Yi Hu, Yujin Yuan, Yuliang Yan, and Zile Zhou. RecGPT-V2 technical report. *arXiv preprint arXiv:2512.14503*, 2025.

[27] Jiakai Tang, Yujie Luo, Xunke Xi, Fei Sun, Xueyang Feng, Sunhao Dai, Chao Yi, Dian Chen, Zhujin Gao, Yang Li, Xu Chen, Wen Chen, Jian Wu, Yuning Jiang, and Bo Zheng. Interactive recommendation agent with active user commands. *arXiv preprint arXiv:2509.21317*, 2025.

[28] Seungheon Doh, Keunwoo Choi, and Juhan Nam. TalkPlay-Tools: Conversational music recommendation with LLM tool calling. *arXiv preprint arXiv:2510.01698*, 2025.

[29] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, pages 1059–1068, New York, NY, USA, 2018. Association for Computing Machinery. doi: 10.1145/3219819.3219823. URL https://doi.org/10.1145/3219819.3219823.

[30] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[31] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes, 2015. URL https://arxiv.org/abs/1506.04757.

[32] Chen Ma, Peng Kang, and Xue Liu. Hierarchical gating networks for sequential recommendation, 2019. URL https://arxiv.org/abs/1906.09217.

[33] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016. URL https://arxiv.org/abs/1511.06939.

[34] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation, 2018. URL https://arxiv.org/abs/1808.09781.

[35] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval, 2023. URL https://arxiv.org/abs/2305.05065.

[36] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations, 2024. URL https://arxiv.org/abs/2402.17152.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

A Detailed Reward Design for GRPO Training

We describe the reward function used to train the slow reasoning model $\mathcal{M}_{\text{slow}}$.

$$R_{\text{slow}} = \lambda_{\text{think}} r_{\text{think}} + \lambda_{\text{sid}} r_{\text{sid}} + \lambda_{\text{hit}} r_{\text{hit}}. \tag{8}$$

A.1 Reasoning Presence (r_{think})

The model must output a non-empty reasoning block delimited by `<think>` and `</think>` tags, with total character length at least 20.

$$r_{\text{think}} = \begin{cases} +1 & \text{if } \texttt{<think>...</think>} \text{ exists and length} \geq 20, \\ -1 & \text{otherwise.} \end{cases} \tag{9}$$

A.2 SID Format Correctness (r_{sid})

The expected strict format is: `<|sid_begin|><s_a_#><s_b_#><s_c_#><|sid_end|>`, where each $\#$ is a code integer. A soft match accepts any pattern that resembles a valid SID.

$$r_{\text{sid}} = \begin{cases} +1 & \text{strict format matched,} \\ +0.2 & \text{soft SID pattern matched,} \\ -1 & \text{no valid SID found.} \end{cases} \tag{10}$$

A.3 Hierarchical Hit Reward (r_{hit})

We parse both the ground-truth and predicted SIDs into three levels (a, b, c). The reward is based on the longest matching prefix:

$$r_{\text{hit}} = \begin{cases} 0 & \text{if no prefix matches,} \\ 1.0 & \text{if only } a \text{ matches,} \\ 2.0 & \text{if } a, b \text{ match,} \\ 5.0 & \text{if } a, b, c \text{ match.} \end{cases} \tag{11}$$

This hierarchical shaping provides dense learning signals even when perfect prediction is not yet achieved, while still strongly incentivizing completely correct SID outputs.

B Pseudocode for Planner Agent Training

Algorithm 1 Planner Agent Training

Input: User sequences $\mathcal{U}$, ground truth items, base models $\mathcal{M}_{\text{fast}}, \mathcal{M}_{\text{slow}}$, ranking model $\mathcal{M}_{\text{rank}}$.
Stage 1: supervised warm-up
for each user $u \in \mathcal{U}$ **do**
  $candidates_{50} \leftarrow \mathcal{M}_{\text{fast}}(u, k = 50)$
  Determine path (1/2/3) based on hit position of ground truth.
  With 20% probability, flip path 1 to 2 or 3.
  Record the required tool call sequence as label.
end for
Train $\mathcal{M}_{\text{agent}}$ via cross-entropy on tool call sequences.
Stage 2: Agentic RL
Freeze $\mathcal{M}_{\text{fast}}, \mathcal{M}_{\text{slow}}, \mathcal{M}_{\text{rank}}$.
Optimize $\mathcal{M}_{\text{agent}}$ with GRPO using reward R_{total}.
return Trained agent $\mathcal{M}_{\text{agent}}$.

C Baselines

C.1 Traditional Sequential Recommenders.

- **HGN** [32]: A hierarchical graph neural network that captures user–item interactions at multiple granularities via a gated meta-path mechanism.
- **GRU4Rec** [33]: A classic RNN-based model that uses gated recurrent units to model user session sequences for next-item prediction.
- **SASRec** [34]: A self-attention based sequential model that adaptively attends to previous items and has become a widely used strong baseline.

C.2 Generative Recommendation Models.

- **TIGER** [35]: A pioneering generative recommender that uses Semantic IDs derived from RQ-VAE and autoregressively decodes the next item's SID.
- **HSTU** [36]: A hierarchical sequential transduction architecture for large-scale recommendation; we use a reproducible configuration adapted to the Amazon benchmark.

C.3 Reasoning-Enhanced Recommendation Models.

- **OneRec-Think** [3]: The most relevant competitor to our work. It introduces itemic alignment via multi-task pre-training, reasoning activation through supervised fine-tuning. However, it applies the same reasoning strategy uniformly to all user histories.

D A Case Study On Slow Rec Model

We present a concrete example to illustrate why the thinking model requires hard samples. Consider a user whose historical interactions are dominated by nail art stamping equipment: she has purchased Konad stamping polishes (white, black, clear top coat), a stamp and scraper set, 25 pattern plates from Bundle Monster, along with various OPI polishes, base coats, and cuticle removers. Her next actual purchase is a *40-pack nail art stamping plate bundle*. When the fast recommendation model or the slow recommendation model trained on full samples encounters this sequence, both collapse to popular, high-frequency predictions that share surface-level lexical cues with the history. The fast recommendation model recommends metallic nail polish sets and holographic polishes (e.g., China Glaze Hologram), due to the training distribution. The model fails to recognize that the user has already moved beyond general nail polish into the specialized sub-task of stamping, where the bottleneck is not color polish but pattern plates. In contrast, the slow recommendation model (trained exclusively on hard samples) successfully generates a correct recommendation with an interpretable reasoning process. Its internal chain-of-thought unfolds as: *The user has purchased stamping polishes in multiple colors, a stamp and scraper, and a set of 25 plates. In the stamping workflow, after acquiring basic plates, users typically expand their design library by buying additional plate bundles. The next logical item is therefore a larger collection of plates.*

This reasoning leverages knowledge (the stamping process) and the understanding of the user. Such reasoning is unlocked only when the model is forced to generalize from challenging examples where shallow co-occurrence fails.

E Latency and Throughput Analysis

We report absolute latency and throughput on a single A100 GPU (batch size 1, Beauty test set). The fast recommendation model baseline achieves 0.39 s/sample and 2.56 samples/s. When the planner invokes slow reasoning, latency increases to 2.15 s/sample, adding 1.76 s absolute overhead per sample. The overhead is dominated by chain-of-thought generation (1.60 s) rather than SID decoding (0.16 s). For the full test set, the adaptive planner invokes slow reasoning only on 14.8% of sequences, keeping average latency near 0.65 s/sample and total inference time under 39 minutes, versus 128.7 minutes for uniform slow reasoning. Uniform fast reasoning would take 23.4 minutes, which is 0.6× the adaptive time, while uniform slow reasoning is 3.3× the adaptive time.

E.1 Implementation Details

We implement our framework using Qwen3.5-4B[30] as the base LLM. All models are trained and evaluated on 8 NVIDIA A100 (80GB) GPUs. Below we detail the configurations for each component.

For each dataset, we follow the leave-one-out strategy: the last interaction for each user is used for testing, the second-last for validation, and all preceding ones for training. We filter out users and items with fewer than 3 interactions to ensure sufficient sequence length [34]. Item metadata (title and category) are concatenated as the textual description t_i. We generate Semantic IDs with $L = 3$ quantization layers and a codebook size $K = 256$ per layer, resulting in a vocabulary of $3 \times 256 = 768$ SID tokens. The residual k-means model is trained on the item embeddings from text encoder [37].

E.1.1 Aligned Base Model ($\mathcal{M}_{\text{align}}$)

The base LLM (Qwen3.5-4B) is fine-tuned for 3 epochs with a learning rate of $2\text{e}{-5}$ and a cosine decay scheduler. We freeze all parameters except the embedding matrix of the 768 SID tokens. The batch size is set to 64 sequences, each truncated or padded to a maximum length of 512 tokens. The training objective is the standard cross-entropy loss for next-token prediction.

E.1.2 Fast Rec Model ($\mathcal{M}_{\text{fast}}$) and Ranker

Starting from $\mathcal{M}_{\text{align}}$, we train $\mathcal{M}_{\text{fast}}$ for 5 epochs on the sequential recommendation task. The learning rate is reduced to $1\text{e}{-5}$ and the batch size is 128. During inference, we use beam search with a width of $k = 50$ and a length penalty of 1.0. The prefix trie for valid SIDs is constructed from the training set's item catalog.

E.1.3 Slow Rec Model ($\mathcal{M}_{\text{slow}}$) with Collaborative Reasoning

We first generate I2I pairs based on co-occurrence in the same session or sequence. For each pair, we use Qwen3.5-397B-A17B via API to generate a reasoning explanation, resulting in 20,000 final triples after filtering low-quality responses (e.g., length < 30 tokens). The teacher model is prompted with the instruction: "In collaborative filtering, item i and item j are highly correlated. Please explain why users who purchase i also tend to purchase j?" These triples are mixed with the alignment data at a 1:1 ratio to fine-tune $\mathcal{M}_{\text{align}}$ for 2 epochs with a learning rate of $1\text{e}{-6}$, producing the base model with collaborative commonsense.

To further train $\mathcal{M}_{\text{slow}}$, we select training sequences where $\mathcal{M}_{\text{fast}}$ fails to retrieve the ground truth within top-50 (hard samples). The model is then optimized using GRPO [38]. We use a sampling temperature of 0.7, and the policy is updated with a clipping parameter $\epsilon = 0.2$. The reward function is detailed in Appendix A, with weights set to $\lambda_{\text{hit}} = 5.0$ and $\lambda_{\text{format}} = 1.0$. The maximum generation length for the thinking chain is 256 tokens.

E.1.4 Planner Agent ($\mathcal{M}_{\text{agent}}$) Training

The agent model is initialized from $\mathcal{M}_{\text{align}}$. For supervised warm-up, we generate pseudo-labels on 100,000 user sequences from the training set, using the path selection logic in Section 4.4 (with probe sizes $K_1 = 10$ and $K_2 = 50$). We train $\mathcal{M}_{\text{agent}}$ for 3 epochs with a learning rate of $1\text{e}{-5}$ and a batch size of 32. The tool call is formatted as a structured JSON object. For RL fine-tuning, we employ GRPO for another 2 epochs.