SAPO: STEP-ALIGNED POLICY OPTIMIZATION FOR REASONING-BASED GENERATIVE RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation treats next-item prediction as autoregressive item-identifier generation. Specifically, items are encoded as semantic identifiers (SIDs), which are short coarse-to-fine token sequences whose early tokens capture broad semantics and later tokens refine them. Recent work augments this paradigm with reasoning traces and optimizes them via reinforcement learning with verifiable rewards, typically an outcome-reward algorithm scored on the generated SID. However, in realistic recommendation, such reward design becomes a single scalar per rollout; when a generated SID mismatches, it cannot identify which SID-token prediction caused the mismatch and may penalize matched SID-token positions together with the mismatched position. We identify that the natural unit of credit assignment in this setting is a single reasoning step (one thinking block paired with one SID token). We instantiate this idea in SAPO (Step-Aligned Policy Optimization): rather than broadcasting one advantage to the whole response, SAPO computes a separate group-relative advantage for each reasoning step and applies it only to the corresponding thinking block and SID token. Across three real-world recommendation datasets, SAPO stabilizes RL training and achieves competitive or improved performance, with the largest gains where fine-grained credit assignment matters most. Our results suggest that reinforcement-learning objectives for structured generation should mirror the decoder's own decomposition of the output. Our code is available at `https://anonymous.4open.science/r/SAPO`.

1 INTRODUCTION

The item recommendation task requires a model to identify items a user is likely to interact with from a large catalog of closely related candidates. Traditional systems typically decouple candidate retrieval from ranking, whereas generative recommendation offers an alternative by casting recommendation as autoregressive item generation. Building on advances in language modeling (Minaee et al., 2024), this formulation has become increasingly attractive, with prior work demonstrating strong empirical performance (Rajput et al., 2023; Deng et al., 2025). Because assigning a unique token to every item would require a prohibitively large output vocabulary, items in this paradigm are represented as semantic identifiers (SIDs), which are short token sequences produced by a hierarchical vector quantizer. The language model is then trained to autoregressively generate the SID of the item a user will interact with next.

A complementary line of work augments generative recommenders with explicit reasoning traces, typically by prompting the model to produce a single reasoning trace before SID decoding (Lin et al., 2025; You et al., 2025; Liu et al., 2025a; He et al., 2026). These traces are intended to help the model interpret user histories and surface relevant preference signals prior to generating an item identifier. Recent systems increasingly adopt a post-training recipe that first aligns the backbone to the recommendation vocabulary via supervised fine-tuning (SFT), then activates structured reasoning, and finally applies reinforcement learning (RL) from verifiable rewards (RLVR) (Lambert et al., 2024) when reliable outcome checks are available. This recipe bridges SFT-only reasoning recommenders and newer SFT-then-RL variants, including GRPO-style approaches for recommendation (Zhu et al., 2025). Once a response contains both a reasoning trace and a multi-token SID, the central training question becomes one of credit assignment under the match feedback of SID: *which SID-token prediction caused the final mismatch?*

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.

For hierarchical SID decoding, credit assignment should respect the positional structure of the SID hierarchy. While final-item verification checks only the whole SID tuple, each SID-token position reveals whether the model has captured fine-grained item information or has failed to identify the broad intent. Outcome-reward RL discards this structure by broadcasting a scalar reward across the reasoning tokens and all SID-token predictions. This mismatch produces two failure modes: matched SID-token positions can be penalized together with the mismatched position, and scalar advantage broadcast cannot localize the mismatched SID-token prediction among unrelated reasoning steps. The underlying issue is action granularity: the rollout is too coarse, yet individual tokens are too fine-grained to serve as the credit-assignment unit for structured SID decoding.

To address this mismatch, we propose **SAPO**, a reinforcement learning method for the RL stage of reasoning-based generative recommendation. SAPO exploits the structure already present in SID decoding: each generated SID tuple exposes multiple verifiable token predictions, giving every rollout a natural decomposition into reasoning steps. We therefore define a reasoning step as one thinking block paired with its corresponding SID token and treat this pair as the action unit for reinforcement learning. Once the action unit is fixed, the rest of the update follows naturally: SAPO assigns a verifiable match reward to each reasoning step from SID-token correctness, estimates a group-relative advantage at the reasoning-step level, and optimizes a step-aligned surrogate with step-normalized token aggregation so that each reasoning step receives its own update signal. The resulting credit assignment remains consistent with the original exact-match objective while providing denser supervision when exact SID-tuple matches are sparse. Our main contributions are as follows:

- We identify the mismatch between exact-match outcome feedback and reasoning-based SID decoding as a problem of action granularity: rollout-level rewards cannot identify the specific mismatch steps, while rollout-level advantages can penalize matched step positions (Section 3).
- We propose **SAPO**, which treats each reasoning step, pairing one thinking block with one SID token, as the RL action unit and derives from this choice a verifiable per-step match reward, a step-level group-relative advantage, and a step-aligned surrogate with step-normalized token aggregation, and introduces no learned reward model (Section 4).
- Experiments on three real-world recommendation datasets show that SAPO produces more stable RL dynamics and yields strong overall Recall and NDCG results, especially in settings where fine-grained credit assignment matters (Section 5).

2 PRELIMINARIES

This section introduces the notation and setup used throughout the paper; Appendix A provides a consolidated summary of symbols and notations used in the paper.

2.1 GENERATIVE RECOMMENDATION WITH HIERARCHICAL SEMANTIC IDENTIFIERS

We formulate generative recommendation over hierarchical semantic identifiers (SIDs). Let $\mathcal{C}$ denote the item catalog. Following recent work on generative recommendation with SIDs (Rajput et al., 2023; Deng et al., 2025), each item $v \in \mathcal{C}$, where v denotes an item, is mapped to a tuple of K SIDs. The tuple can be obtained with a hierarchical vector-quantization scheme, such as a residual-quantization variational autoencoder (RQ-VAE) adapted from residual-quantized image generation (Lee et al., 2022). Let $\mathcal{V}^{(k)}$ denote the vocabulary of the k-th codebook for $k = 1, \ldots, K$. The quantizer defines an injection $\phi\colon \mathcal{C} \to \mathcal{V}^{(1)} \times \mathcal{V}^{(2)} \times \cdots \times \mathcal{V}^{(K)}$. Let $\Phi := \phi(\mathcal{C})$ denote the set of valid catalog SID tuples. Each item v is then uniquely represented as $\phi(v) = (s^{(1)}, s^{(2)}, \ldots, s^{(K)})$, where $s^{(k)} \in \mathcal{V}^{(k)}$ captures progressively finer-grained item semantics, with larger k corresponding to finer semantic levels. We set K=3, which suffices to uniquely identify every item in our catalogs.

Given a chronologically ordered user interaction history $x = (v_1, \ldots, v_{T_x})$, we study the sequential recommendation task of predicting the next item $v^\star = v_{T_x+1}$. The history is encoded as a textual prompt that contains past interactions and a task instruction, and we learn a policy $\pi_\theta(y|x)$ parameterized by a language model with parameters θ. Its response y terminates in a valid SID tuple $\phi(\hat{v}) \in \Phi$, where $\hat{v}$ denotes the predicted next item. In the simplest non-reasoning setting, the policy directly decodes the K SID tokens. We defer the reasoning-augmented response layout and the SAPO update to Section 4.

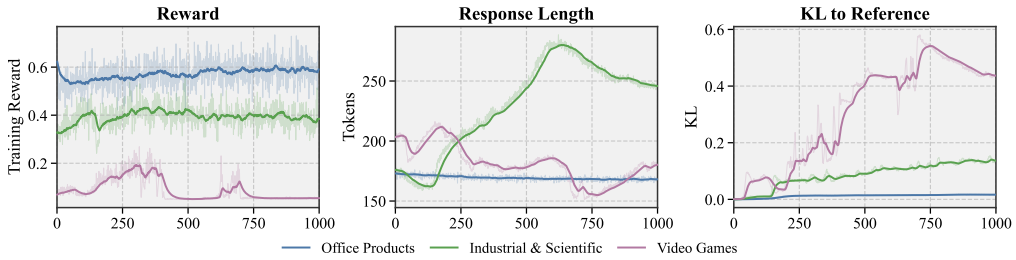

Figure 1: Motivating failure mode of outcome-reward GRPO across three recommendation datasets. GRPO typically improves reward early but later plateaus or becomes unstable as response length and KL to the reference grow. These dynamics reflect the difficulty of assigning credit from a single rollout-level scalar.

2.2 Three-Stage Training Setup

Motivated by recent reasoning-based generative recommenders such as OneRec-Think (Liu et al., 2025a) and SIDReasoner (He et al., 2026), we adopt a three-stage setup for reasoning-based SID decoding. **Stage 1** aligns the language model to the SID vocabulary and the recommendation prompt format, so the model can produce valid SID tokens for the next-item prediction task. **Stage 2** trains the policy to generate K thinking blocks followed by the K SID tokens of the predicted item, giving the response one reasoning block for each SID-token position. **Stage 3** applies reinforcement learning (RL) from final exact-SID feedback, while the SID hierarchy exposes intermediate SID-token match signals for credit assignment. The next section analyzes why the standard outcome-reward RL objective is misaligned with this per-step decoding structure.

3 The Action-Granularity Mismatch in Outcome-Reward GRPO

Outcome-reward GRPO. We focus on the RL stage of Section 2.2, where the policy already generates reasoning traces and SID tokens. For each prompt $x \in \mathcal{D}_{\mathrm{RL}}$, where $\mathcal{D}_{\mathrm{RL}}$ denotes the set of reinforcement-learning training prompts, outcome-reward GRPO samples G rollouts $\{y_i\}_{i=1}^{G}$ (with $[G] = \{1, \ldots, G\}$) and assigns each rollout a scalar reward $r(x, y_i)$, typically an exact-match indicator for the ground-truth SID tuple. The group-relative advantage is

$$\hat{A}_i = \frac{r(x, y_i) - \mathrm{mean}_{j \in [G]}\, r(x, y_j)}{\mathrm{std}_{j \in [G]}\, r(x, y_j)}. \tag{1}$$

It then optimizes the token-level clipped policy-gradient surrogate:

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{x, \{y_i\}} \left[\frac{1}{\sum_{\ell=1}^{G} |y_\ell|} \sum_{i=1}^{G} \sum_{t \in y_i} \min\Big(w_{i,t}(\theta) \hat{A}_i,\ \mathrm{clip}(w_{i,t}(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_i \Big) \right], \tag{2}$$

where $w_{i,t}(\theta) = \pi_\theta(y_{i,t} \mid x, y_{i,<t}) / \pi_{\theta_{\mathrm{old}}}(y_{i,t} \mid x, y_{i,<t})$ is the per-token importance ratio, θ_{old} denotes the policy parameters before the update, and ϵ is the clipping radius. A Kullback–Leibler (KL) penalty to the Stage 2 reference is used in practice but omitted here because it is shared by all Stage 3 variants.

Where granularity fails. Eq. (2) assigns one scalar advantage $\hat{A}_i$ to the entire response. For large-catalog SID decoding, this scalar is too coarse: final feedback reports the mismatch, but not which SID-token prediction caused it. Consider $K = 3$ with ground-truth tuple $(s^{\mathrm{gt},(1)}, s^{\mathrm{gt},(2)}, s^{\mathrm{gt},(3)})$. The near-miss prediction with one last mismatch token $(s^{\mathrm{gt},(1)}, s^{\mathrm{gt},(2)}, \tilde{s}^{(3)})$ and the coarse-error prediction $(\tilde{s}^{(1)}, \tilde{s}^{(2)}, \tilde{s}^{(3)})$ both receive zero exact-match reward. Outcome-reward GRPO therefore assigns them the same rollout-level advantage and broadcasts it to every token; in the near miss, matched SID-token predictions are updated together with the mismatched prediction.

Why reward shaping is insufficient. Figure 1 illustrates this failure mode across all three recommendation datasets: reward dynamics plateau or become irregular as response length and KL to the reference grow. Reward sparsity amplifies this problem because exact SID-tuple matches are rare, making group-relative advantages weak or noisy, especially on harder recommendation datasets.

However, sparsity is not the whole issue. A denser rollout-level reward could replace the exact-match indicator; for example, one could use

$$r_{\text{shape}}(x, y_i) = \sum_{k=1}^{K} \mathbb{1}\left[s_i^{(k)} = s^{\text{gt},(k)}\right], \tag{3}$$

but Eq. (2) would still convert this score into a single rollout-level advantage and assign it to every token. Such shaping reports that a rollout is partially correct, but it still cannot isolate the mismatched SID-token prediction. **This is what we refer to as the action-granularity mismatch.** The rollout is too coarse because it hides which of the K verifiable SID-token predictions succeeded or failed. Individual response tokens are too fine because arbitrary reasoning tokens do not have verifiable correctness labels, whereas SID-token positions do. The natural credit-assignment unit lies in between: the reasoning step that pairs a thinking block with its SID token.

This mismatch causes two concrete problems. First, *matched SID-token credit is discarded*: matched SID-token predictions receive no positive signal when another SID token is wrong, so plausible near misses are treated as complete failures. Second, *reasoning-step advantage localization is lost*: an error at one SID-token position assigns the same scalar weight to token gradients from unrelated thinking blocks and SID-token positions. In long reasoning responses, this diffuse broadcast can reinforce or penalize many reasoning tokens together even though the verifier only checks the SID-token predictions, contributing to length drift, format degradation, and weak group-relative contrast. These observations motivate aligning reward, advantage, and token aggregation at the reasoning-step granularity in the next section.

4 METHODOLOGY

Motivated by the analysis above, SAPO addresses the RL-stage credit-assignment problem in reasoning-based SID decoding. Its central design choice is to use each reasoning step, defined as a SID-token prediction together with its associated thinking block, as the credit-assignment (action) unit. Once this reasoning step is fixed, reward placement, advantage estimation, and token aggregation should all be defined at the same granularity. We now define the reasoning-step decomposition used by SAPO and present the reasoning-step match reward and the step-aligned surrogate. The overall pipeline is illustrated in Figure 2.

4.1 SAPO OVERVIEW AND REASONING-STEP DECOMPOSITION

Reasoning-augmented generative recommenders ask the policy to generate structured reasoning before producing SID tokens. In our implementation, the model first generates all K thinking blocks and then the K SID tokens as a contiguous block:

$$y = \big[\underbrace{\langle\texttt{think}\rangle\, \tau^{(1)}\, \langle/\texttt{think}\rangle \oplus \cdots \oplus \langle\texttt{think}\rangle\, \tau^{(K)}\, \langle/\texttt{think}\rangle}_{\text{reasoning trace with } K \text{ thinking blocks}} \oplus \underbrace{s^{(1)}\, s^{(2)}\, \cdots\, s^{(K)}}_{\text{all } K \text{ SID tokens}}\big]. \tag{4}$$

This layout enables prefix-tree (trie)-based constrained decoding with key-value (KV) cache reuse across SID-token positions. For credit assignment, SAPO pairs the k-th thinking block with the k-th SID token and treats that pair as *reasoning step* k. Formally, under Eq. (4) we define reasoning step k as $(\tau^{(k)}, s^{(k)})$, write $|\tau^{(k)}|$ for the number of tokens in the k-th thinking block, and define $|y^{(k)}| = |\tau^{(k)}| + 1$ as the size of the paired unit including its SID token. This hierarchy-indexed pairing uses the shared ordinal index of the coarse-to-fine SID levels and the ordered thinking blocks, while preserving the contiguous SID suffix that enables efficient constrained decoding. This pairing is used only to assign credit under the blocked decoding layout; it does not change the autoregressive factorization or causal decoding order. Appendix F details the implementation layout and explains why SAPO's reward and advantage remain well defined under this pairing.

4.2 REASONING-STEP MATCH REWARD

With the reasoning step fixed, reward should be placed at SID-token positions rather than only at the end of the rollout. For a rollout y_i with generated SID sequence $(s_i^{(1)}, \ldots, s_i^{(K)})$ and ground truth

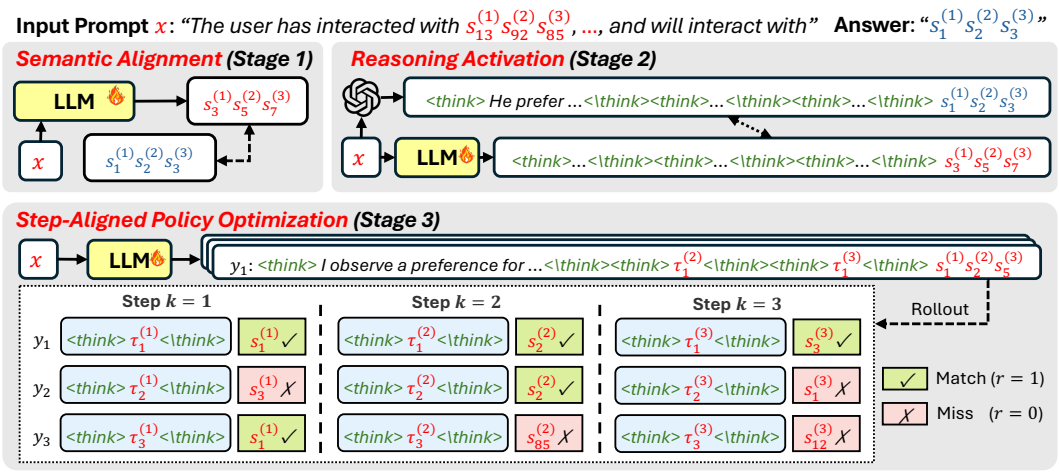

Figure 2: Overview of **SAPO**. Stage 1 aligns the language model to the SID vocabulary, Stage 2 activates level-aware reasoning, and Stage 3 applies step-aligned reinforcement learning. Under the released decoding layout, the policy generates K thinking blocks followed by the K SID tokens of the predicted item. SAPO assigns a reasoning-step match reward using SID-token correctness, attaches a small format bonus at the last SID-token position, computes a reasoning-step group-relative advantage, and optimizes a surrogate with step-normalized token aggregation.

$(s^{\mathrm{gt},(1)}, \ldots, s^{\mathrm{gt},(K)})$, SAPO uses a reasoning-step match reward term plus a small final format bonus:

$$r_{i,k} \;=\; \alpha \cdot \mathbb{1}\Big[s_i^{(k)} = s^{\mathrm{gt},(k)}\Big] \;+\; \beta \cdot \mathbb{1}[k = K] \cdot b_i, \tag{5}$$

where $\mathbb{1}[\cdot]$ is the indicator function, $b_i = 1$ when y_i contains exactly K extracted thinking blocks and ends in a valid SID tuple, and $b_i = 0$ otherwise. The first term preserves credit for matched SID-token positions when another SID token mismatches, while the second term attaches a small bonus to the last SID-token position when the rollout is structurally well formed. The following objective-consistency discussion focuses on the reasoning-step match reward term.

Objective consistency. Let $m_{i,k} := \alpha\, \mathbb{1}[s_i^{(k)} = s^{\mathrm{gt},(k)}]$ denote the match component of Eq. (5). Although SAPO decomposes exact-match feedback into reasoning-step rewards, it does not change the exact-match optimum under the standard realizability condition:

$$\sum_{k=1}^{K} m_{i,k} = \alpha \left|\{k \in [K] : s_i^{(k)} = s^{\mathrm{gt},(k)}\}\right|. \tag{6}$$

Let $\mathcal{J}_{\mathrm{out}}$ be the expected exact-match reward and $\mathcal{J}_{\mathrm{match}}$ be the expected cumulative match reward.

Proposition 1 (Reasoning-step match reward and outcome consistency)**.** *If the policy class* Π *contains an exact-match policy, then*

$$\arg\max_{\pi \in \Pi} \mathcal{J}_{\mathrm{out}}(\pi) = \arg\max_{\pi \in \Pi} \mathcal{J}_{\mathrm{match}}(\pi) \tag{7}$$

Thus the reasoning-step match reward supplies denser supervision while preserving the exact-match optimum; Appendix E.1 gives the proof and further discussion.

4.3 Reasoning-Step Advantage and SAPO Surrogate

Once reward is placed at SID-token positions, both the advantage and the surrogate loss should be defined at the same granularity. SAPO therefore defines advantages over reasoning steps $(\tau^{(k)}, s^{(k)})$, rather than over the whole sequence (too coarse) or each individual token (too fine-grained). Given the return $r_{i,k}$ at SID-token position k in Eq. (5), we compute a group-relative advantage and its step-normalized form independently for each reasoning step:

$$\hat{A}_{i,k} = \frac{r_{i,k} - \mathrm{mean}_{i' \in [G]}(r_{i',k})}{\mathrm{std}_{i' \in [G]}(r_{i',k})}, \quad \tilde{A}_{i,k} = \frac{\hat{A}_{i,k}}{|y_i^{(k)}|}. \tag{8}$$

Here "step-normalized" means that the reasoning-step advantage is divided by the paired unit length $|y_i^{(k)}|$, which includes the k-th thinking block and its paired SID token. Using the same per-token importance ratio $w_{i,t}(\theta)$ as in Eq. (2), SAPO optimizes the resulting step-aligned surrogate with step-normalized token aggregation,

$$\mathcal{J}_{\text{SAPO}}(\theta) = \mathbb{E}_{x,\{y_i\}}\left[\frac{1}{\sum_{\ell=1}^{G}|y_\ell|}\sum_{i=1}^{G}\sum_{k=1}^{K}\sum_{t\in y_i^{(k)}}\min\Big(w_{i,t}(\theta)\tilde{A}_{i,k},\ \text{clip}(w_{i,t}(\theta),1-\epsilon,1+\epsilon)\tilde{A}_{i,k}\Big)\right]. \tag{9}$$

Gradient analysis. Ignoring clipping for exposition, Eq. (9) gives the gradient view:

$$\begin{aligned}\nabla_\theta\mathcal{J}_{\text{SAPO}}(\theta) &\propto \mathbb{E}\left[\frac{1}{\sum_{\ell=1}^{G}|y_\ell|}\sum_{i=1}^{G}\sum_{k=1}^{K}\sum_{t\in y_i^{(k)}} g_{i,k,t}(\theta)\right],\\ g_{i,k,t}(\theta) &= \underbrace{w_{i,t}(\theta)}_{\text{importance}}\cdot\underbrace{\hat{A}_{i,k}}_{\text{reasoning-step advantage}}\cdot\underbrace{\frac{1}{|y_i^{(k)}|}\nabla_\theta\log\pi_\theta(y_{i,t}\mid x,y_{i,<t})}_{\text{step-normalized token gradient}}.\end{aligned} \tag{10}$$

This decomposition highlights the two differences from rollout-level GRPO. First, SAPO replaces the rollout-wide scalar advantage with the reasoning-step signal $\hat{A}_{i,k}$, so rollouts that differ only at SID-token position k produce a contrastive update at that step without assigning the same advantage to unrelated SID-token positions. Second, the factor $1/|y_i^{(k)}|$ averages the token gradients within each reasoning step, ensuring that the update magnitude depends on the quality of the reasoning step rather than the number of tokens it contains. Together, these two changes explain why SAPO mitigates the excessive reasoning verbosity and length drift observed under rollout-level advantage broadcast in Section 5.3.

5 Experiments

We address three research questions (RQs). **(RQ1)** Does SAPO improve end-task recommendation quality against generative recommendation baselines? **(RQ2)** Does reasoning-step credit assignment stabilize reinforcement learning dynamics for recommendation with sparse exact-match feedback? **(RQ3)** How do the reasoning-step match reward and reasoning-step advantage contribute to SAPO?

5.1 Experimental Setup

Datasets. We evaluate on three categories from the Amazon Reviews dataset (Ni et al., 2019), including `Office-Products`, `Video-Games`, and `Industrial-and-Scientific`, following the widely adopted sequential recommendation protocol of (Kang & McAuley, 2018). For each category, we sort user interactions chronologically, keep users with at least five interactions, and construct (history, next-item) pairs under a leave-one-out split: the last interaction forms the test record, the second-to-last forms the validation record, and the remaining prefix forms the training set. For every item, we learn a K=3-level RQ-VAE codebook over the item text embeddings (title, category, brand, and description) and use the resulting SID tuple as the target. Detailed statistics and the SID tokenization procedure are deferred to Appendix H; concrete record and prompt examples are provided in Appendix I. Implementation details, backbone, hyperparameters, and reproducibility information are deferred to Appendix J. To keep the main text concise, single-category figures and diagnostics use `Industrial-and-Scientific` as the representative dataset.

Baselines. We compare SAPO against representative methods from three families. **(i) Traditional sequential recommenders**: GRU4Rec (Hidasi et al., 2015), SASRec (Kang & McAuley, 2018), and Caser (Tang & Wang, 2018). **(ii) Generative recommenders without reasoning**: TIGER (Rajput et al., 2023), HSTU (Zhai et al., 2024), LCRec (Zheng et al., 2024), and LETTER (Wang et al., 2024). **(iii) Reasoning-based generative recommenders**: SIDReasoner (He et al., 2026), OneRec-Think (Liu et al., 2025a), and ReaRec (Tang et al., 2025). Brief descriptions of all baselines are provided in Appendix G.

Table 1: Main results on three Amazon-review datasets. R@k and N@k denote Recall@k and NDCG@k, respectively. The best result in each column is **bolded** and the second best is underlined.

	Office-Products				Video-Games				Industrial-and-Scientific			
Method	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
(i) Traditional sequential recommenders												
GRU4Rec	0.0240	0.0135	0.0386	0.0181	0.0207	0.0139	0.0357	0.0187	0.0457	0.0408	0.0532	0.0432
SASRec	0.1019	0.0824	0.1167	0.0871	0.0501	0.0345	0.0723	0.0416	0.0807	0.0647	0.0964	0.0697
Caser	0.0232	0.0139	0.0347	0.0176	0.0247	0.0161	0.0417	0.0214	0.0459	0.0377	0.0578	0.0415
(ii) Generative recommenders without reasoning												
TIGER	0.0569	0.0415	0.0729	0.0467	0.0475	0.0328	0.0692	0.0397	0.0979	0.0792	0.1291	0.0885
HSTU	0.1204	0.1069	0.1323	0.1107	0.0539	0.0396	0.0746	0.0462	0.1008	0.0898	0.1138	0.0940
LCRec	0.0193	0.0135	0.0370	0.0191	0.0158	0.0115	0.0177	0.0121	0.0552	0.0512	0.0600	0.0527
LETTER	0.1315	0.1074	0.1520	0.1139	0.0445	0.0294	0.0709	0.0378	0.1080	0.0850	0.1389	0.0950
(iii) Reasoning-based generative recommenders												
ReaRec	0.1173	0.0988	0.1385	0.1057	0.0568	0.0381	0.0843	0.0470	0.0973	0.0796	0.1205	0.0870
OneRec-Think	0.1353	0.1107	0.1506	0.1159	0.0413	0.0296	0.0575	0.0349	0.0464	0.0398	0.0520	0.0416
SIDReasoner	**0.1381**	0.1116	0.1629	0.1197	0.0578	0.0385	0.0897	0.0487	0.1109	0.0905	0.1438	0.1010
SAPO	0.1371	**0.1145**	**0.1649**	**0.1241**	**0.0620**	**0.0413**	**0.0986**	**0.0529**	**0.1181**	**0.0935**	**0.1444**	**0.1018**

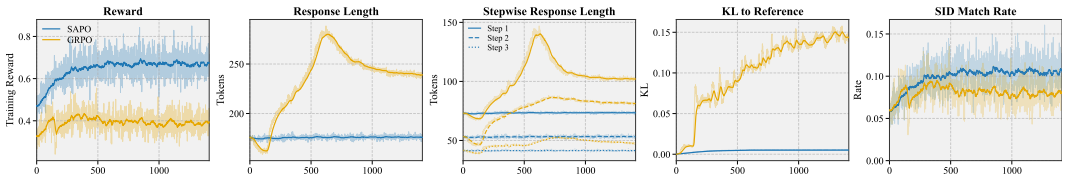

Figure 3: RL training dynamics on the Industrial-and-Scientific dataset. We compare SAPO and GRPO from the same Stage 2 checkpoint using training reward, response length, stepwise response length, KL to the reference, and SID match rate; blue denotes SAPO and orange denotes GRPO.

5.2 Comparison with Baselines (RQ1)

To address **RQ1**, we evaluate SAPO against the baselines on the held-out test set of each category. Following common practice (Kang & McAuley, 2018), we report Recall@k (R@k) and normalized discounted cumulative gain (NDCG@k, denoted N@k) at $k \in \{5, 10\}$.

Table 1 shows that SAPO improves end-task ranking quality, while the pattern is more informative than uniform dominance. The most direct comparison is SIDReasoner, which shares the same broad reasoning-based setup but uses a rollout-level outcome reward in Stage 3: SAPO improves every NDCG and R@10 result over SIDReasoner and trails it only on `Office-Products` Recall@5. This pattern suggests that reasoning activation alone is not sufficient; once reasoning is introduced, the RL credit signal must also respect the hierarchical SID-token structure. Across the full table, SAPO ranks first in 11 of the 12 metric-by-dataset settings and second in the remaining one. It achieves the best NDCG in all six NDCG columns; among the six Recall columns, it ranks first in five and second in one rather than uniformly dominating every Recall setting. SAPO is consistently stronger than non-reasoning generative baselines, supporting the broader conclusion that reasoning benefits most when RL credit assignment follows the reasoning-step structure. We report an extended performance comparison with SIDReasoner and outcome-reward GRPO in Appendix N.

5.3 Training Dynamics (RQ2)

We next address **RQ2** by comparing the training dynamics of GRPO and SAPO from the same Stage 2 checkpoint. Figure 3 reports the representative `Industrial-and-Scientific` run using five signals: training reward, response length, stepwise response length, KL to the reference, and SID match rate. SAPO shows a steadier reward trajectory and stronger SID-match dynamics, while keeping both overall reasoning-trace length and per-reasoning-step thinking-block lengths more controlled. The per-reasoning-step length curves further suggest that this control does not come from shortening only one thinking block, but from stabilizing the reasoning budget across SID-token positions. GRPO exhibits noisier and less stable reward dynamics under sparse exact-match feedback, together with larger length and KL drift; Appendix M provides the corresponding diagnostics for the other two datasets. The SID match-rate panel links these optimization dynamics to the target hierarchy, showing whether training actually improves SID-token predictions rather

Table 2: Component ablation of SAPO on three Amazon-review datasets. The per-step labels correspond to the reasoning-step advantage and reasoning-step match reward defined at SID-token positions. We additionally report the reasoning activation performance on stage 2 for reference. The best result in each column is **bolded** and the second best is underlined.

	Office-Products				Video-Games				Industrial-and-Scientific			
Method	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
Reasoning Activation	0.1143	0.0933	0.1410	0.1020	0.0577	0.0375	0.0926	0.0466	0.1032	0.0812	0.1299	0.0899
Full SAPO	**0.1371**	**0.1145**	**0.1649**	**0.1241**	0.0620	**0.0413**	**0.0986**	**0.0529**	**0.1181**	**0.0935**	**0.1444**	**0.1018**
w/o per-step advantage	0.1229	0.1008	0.1494	0.1093	**0.0625**	0.0411	0.0979	0.0528	0.1171	0.0916	0.1401	0.0990
w/o per-step match reward	0.1268	0.1029	0.1535	0.1115	0.0591	0.0385	0.0941	0.0498	0.1149	0.0913	0.1389	0.0990
w/o both *(pure GRPO)*	0.1235	0.1001	0.1490	0.1083	0.0290	0.0209	0.0427	0.0256	0.1032	0.0846	0.1207	0.0903

than only increasing rollout-level reward. Taken together, these dynamics are consistent with the mechanism in Section 4.3: SAPO ties updates to reasoning-step advantages and step-normalized token aggregation, rather than broadcasting one scalar advantage over the whole rollout.

5.4 Ablation Study (RQ3)

To address **RQ3**, we run component ablations on the three categories and report all completed entries in Table 2; single-category training diagnostics use `Industrial-and-Scientific` as the representative setting. Starting from full SAPO, we retrain Stage 3 after removing one credit-assignment ingredient at a time. Specifically, we consider the following three variants: ***(i) w/o per-step advantage***, removes reasoning-step advantage localization by replacing $\hat{A}_{i,k}$ with a single rollout-level advantage while keeping the per-step match reward and step-normalized token aggregation. ***(ii) w/o per-step match reward***, removes dense SID-token feedback by replacing the per-step match reward with the final SID-tuple exact-match reward shared by all steps. ***(iii) w/o both (pure GRPO)***, removes both ingredients and broadcasts one rollout-level advantage to all response tokens.

Table 2 shows that SAPO's two ingredients are complementary rather than interchangeable. Across the completed ablations, full SAPO achieves the best result in 11 of 12 metrics. The only exception is `Video-Games` R@5, where removing the per-step advantage is narrowly higher (0.0625 vs. 0.0620). The two single-component variants trail full SAPO in 23 of their 24 metric comparisons, while pure GRPO is substantially weaker overall, indicating that dense SID-token feedback and reasoning-step advantage localization address different parts of the credit-assignment problem.

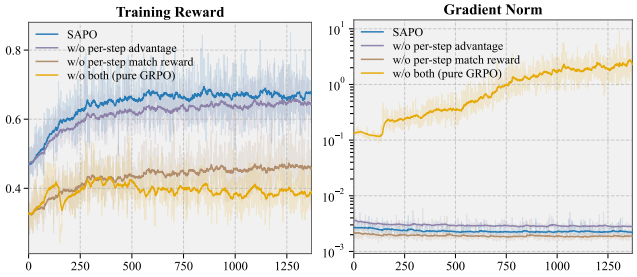

Figure 4: Ablation diagnostics: reward and gradient norm ($\log_{10}$ scale) on the Industrial & Scientific dataset.

Beyond final ranking metrics, Figure 4 clarifies why this complementarity matters during RL. The single-component ablations do not exhibit the gradient-norm explosion of pure GRPO, suggesting that either retaining the reasoning-step match reward or retaining the step-aligned surrogate is enough to avoid the worst rollout-level update amplification. However, their reward trajectories remain weaker, and full SAPO leads the ablation variants on 11 of 12 final ranking metrics, which suggests that stability alone is not sufficient: the strongest overall performance requires both a denser SID-token match signal and reasoning-step advantage assignment. Appendix P reports a controlled permutation ablation that keeps SID-token credit fixed and reassigns only the SID-level advantages applied to thinking blocks; the resulting degradation shows that the ordered block-to-level mapping matters for ranking quality.

5.5 Case Study

To complement the quantitative results, we provide qualitative examples showing how SAPO shapes reasoning behavior. Figure 5 compares outcome-reward training and SAPO on the same `Industrial-and-Scientific` test prompt. GRPO produces a plausible near miss: it captures

Shared prompt history. <a_15><b_155><c_49> (Industrial & Scientific category); <a_91><b_29><c_123> (Kamo 3D-printer extruder brass nozzle); <a_73><b_244><c_79> (pin-header and Dupont connector kit); <a_158><b_206><c_3> (black PLA 3D-printer filament); <a_159><b_179><c_39> (NEMA17 flexible coupling for CNC); <a_30><b_192><c_4> (NEMA17 stepper motor for DIY CNC).
Target. <a_158><b_89><c_233>: DRV8825 StepStick stepper motor drivers for 3D-printer electronics, CNC machines, or robotics.

Outcome GRPO

Thinking block 1. I notice the user has a strong interest in 3D printing and related hardware, as seen in their engagement with <a_91><b_29><c_123>, <a_158><b_206><c_3>, and <a_159><b_179><c_39>. They also explored electronics components like <a_73><b_244><c_79> and <a_30><b_192><c_4>, indicating a hands-on approach to building and customizing devices. The inclusion of <a_15><b_155><c_49> suggests an appreciation for precision and quality in their projects.

Thinking block 2. Given the variety of items–ranging from filament and extruder parts to electronic connectors and stepper motors–the user appears to enjoy assembling, upgrading, and maintaining 3D printers and related equipment. Their pattern of selecting both consumables and mechanical components points to a methodical, project-oriented mindset, possibly for DIY or small-scale manufacturing. I would expect the user to continue exploring compatible accessories, upgrade parts, or additional 3D printing materials that support their technical interests.

Thinking block 3. Overall, the user may enjoy further items related to 3D printer hardware, electronics, or precision tools that enable more advanced or varied project builds. I will recommend the next item as <a_91><b_105><c_93>, which aligns with their ongoing interest in high-quality, compatible components for 3D printing.

Prediction. <a_91><b_105><c_93>: extruder nozzle and tube kit. ***Wrong; the target is not in top-10.***

SAPO

Thinking block 1. Examining the user's interaction history, I see a clear pattern of engagement with components and accessories central to 3D printing and CNC projects. The user consistently selects items such as <a_91><b_29><c_123>, <a_158><b_206><c_3>, and <a_159><b_179><c_39>, which are directly related to filament, extrusion, and mechanical motion–key elements in building or maintaining 3D printers.

Thinking block 2. The inclusion of <a_73><b_244><c_79> and <a_30><b_192><c_4> suggests an interest in electronics and stepper motor control, indicating a hands-on approach to hardware assembly and customization. The user's choices reflect a preference for reliable, compatible, and high-quality parts, often in bulk or multi-pack formats, which points to ongoing or large-scale projects.

Thinking block 3. Based on these patterns, the user may enjoy exploring further 3D printer upgrades, motion control components, or specialized accessories that enhance printer performance and expand functionality.

Prediction. <a_158><b_89><c_233>: DRV8825 StepStick stepper motor drivers. ***Correct.***

Figure 5: Case study on `Industrial-and-Scientific`. Both methods receive the same prompt, with SID tokens highlighted in red. Outcome GRPO follows the general interest but misses the target, whereas SAPO identifies more discriminative evidence and predicts the correct SID.

the broad 3D-printing domain but treats the history as generic printer parts, leading to another accessory. The key missing cue is the transition from filament and extrusion components to NEMA17 coupling and stepper motor, which indicates motion control. SAPO preserves this finer-grained cue by reasoning about electronics and stepper motor control, and ranks the motor-driver SID first.

Because both models receive the same prompt and start from the same Stage 2 reasoning-activation checkpoint, the contrast primarily reflects the effect of the Stage 3 credit-assignment objective. The outcome reward only distinguishes complete SID-tuple success from failure, so this broadly relevant but wrong accessory receives the same scalar penalty as a completely unrelated item. SAPO instead assigns feedback to reasoning steps using SID-token correctness, making the update sensitive to which SID-token prediction is correct and which refinement fails. The case study therefore complements the training-dynamics results: reasoning-step credit assignment improves stability while also encouraging reasoning that preserves the evidence needed for fine-grained SID selection. We provide additional case studies, including examples from `Video-Games` and `Office-Products`, in Appendix R.

6 Conclusion

We introduced **SAPO**, a step-aligned reinforcement-learning method for reasoning-based generative recommenders that decode items over a K-level SID codebook. The central observation is that outcome-reward RL uses the wrong action granularity for this setting: it obscures which SID-token prediction caused the mismatch and can penalize matched SID-token predictions together with erroneous refinements. SAPO resolves this mismatch by treating each reasoning step $(\tau^{(k)}, s^{(k)})$ as the action unit and deriving from that choice a per-step match reward, a per-step group-relative advantage, and a step-aligned surrogate with step-normalized token aggregation. Across the evaluated settings, SAPO materially stabilizes training dynamics and yields competitive-to-improved Recall and NDCG relative to outcome-reward baselines, with the clearest gains where fine-grained credit matters most. Beyond SIDs, our analysis suggests that whenever a generation task admits a natural hierarchical decomposition, such as a codebook, a tool-call schema, or a structured answer format, the hierarchy itself can serve as a fundamental source of per-step supervision for reinforcement learning, without any learned reward model.

AI USE STATEMENT

Generative AI tools were used in two roles in this work. First, as a component of the method under study: the Stage 2 reasoning-activation training set consists of synthetic reasoning traces generated by large language models, and Appendix Q reports how the choice of generator affects the final results. Second, as author-side support: we used these tools to implement parts of the training, evaluation, and plotting code, to edit the manuscript for readability, to help identify relevant literature, and to produce figures from computed results. We did not use generative AI to develop the theoretical framework, formulate the mathematical claims, write or supply critical ingredients for the proofs, propose or refine the hypotheses, design the experimental methodology, clean or reformat the datasets, or interpret the experimental results; translation and qualitative data analysis are not applicable here. We reviewed all AI-assisted output and we take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies credit assignment for reinforcement learning on generative recommenders and uses only publicly available Amazon Reviews data; it involves no human subjects, no personally identifying information, and no new data collection. We discuss the societal considerations that follow from improving recommendation accuracy, including filter-bubble reinforcement and preference manipulation, in Appendix D, and we discuss the scope and limitations of our claims in Appendix C.

REPRODUCIBILITY STATEMENT

Our anonymized implementation is available at the link given in the abstract. The problem setting and notation are fixed in Section 2 and Appendix A; the reward, advantage, and surrogate objective are stated in Section 4, with the full assumptions and proofs of the consistency claim in Appendix E and the decoding layout that defines a reasoning step in Appendix F. Dataset construction and per-category statistics are described in Appendix H, with concrete records and prompt templates in Appendix I. Backbones, hyperparameters, and the three-stage training schedule are given in Appendix J and Appendix K, and the baselines we compare against are specified in Appendix G.

REFERENCES

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.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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. *arXiv preprint arXiv:2603.23183*, 2026.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11523–11532, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Jiacheng Lin, Tian Wang, and Kun Qian. Rec-r1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning. *Transactions on Machine Learning Research*, 2025.

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*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Cristina Menghini, Peter Ney, Hamza Kwisaba, Zifan, Wang, Miles Turpin, Felix Binder, Jean-Christophe Testud, Aidan Boyd, Nathaniel Li, Ivan Evtimov, Klaudia Krawiecka, Arman Zharmagambetov, Jeremy Kritz, Alexander R. Fabbri, Daniel Song, Jinpeng Miao, Joonas Hjelt, Meghna Ramani, Leona Lan, Reza Aghajani, Joanna Bitton, Mahesh Pasupuleti, Devin Norder, Khalid El-Arini, Paridhi Singh, Vítor Albiero, Sahana CB, Rashnil Chaturvedi, Elahe Dabir, Edoardo Debenedetti, Jim Gust, Ziwen Han, Kat He, Sean Hendryx, Lifeng Jin, Polina Kirichenko, Sandra Lefdal, Kenneth Li, Asad Liaqat, Inna Lin, Despoina Magka, Neal Mangaokar, Ishita Mediratta, Zach Miller, Smitha Milli, Niloofar Mireshghallah, Saba Nazir, Hung Nguyen, Maximilian Nickel, Kelvin Niu, Kerem Oktar, Bhargavi Paranjape, Parth Pathak, Maya Pavlova, Emmanuel Ramirez, David Renardy, Candace Ross, Yasha Sheynin, Claudia Shi, Shivam Singhal, Evangelia Spiliopoulou, Rakshith Sharma Srinivasa, Jamelle Watson-Daniels, Spencer Whitman, Adina Williams, Chen Xing, Andy Zou, Tommy Ma, Siqi Deng, James Beldock, Prashant Ratanchandani, Kate Plawiak, Taesung Lee, Ryan Victory, Lindsay Hundley, Rachad Alao, Himaghna Bhattacharjee, Jianfeng Chi, Gary Frost, Pegah Ghahremani, Niki Howe, Yuheng Huang, Saeed Jahed, Hannah Korevaar, Trang Le, Zhe Liu, Jinghong Luo, Qin Lyu, Nina Mehrabi, Abraham Montilla, Chirag Nagpal, Cyrus Nikolaidis, Rajvardhan Oak, Manoj Ravi, Vidya Sarma, Aman Shankar, Alana Shine, Eric Michael Smith, Mariana Tandon, Michael Tontchev, Caoyu Wang, Zihan Wang, Corinne Wong, Zheng Wu, Hongyuan Zhan, Justin Zhao, Zexuan Zhong, Chengxu Zhuang, Tristan Goodman, Ayaz Minhas, Harrison Rudolph, Victoria Jeffries, Ingrid Dickinson, Alex Vaughan, Lauren Deason, Kamalika Chaudhuri, Julian Michael, Shengjia Zhao, and Summer Yue. Muse spark safety & preparedness report, 2026. URL `https://arxiv.org/abs/2606.12429`.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 188–197, 2019.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike

Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the 20th European Conference on Computer Systems*, pp. 1279–1297, 2025.

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. *arXiv preprint arXiv:2503.22675*, 2025.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 565–573, 2018.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 2400–2409, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Shuo Yang, Jiangxia Cao, Haipeng Li, Yuqi Mao, and Shuchao Pang. Reccot: Enhancing recommendation via chain-of-thought. *arXiv preprint arXiv:2506.21032*, 2025b.

Runyang You, Yongqi Li, Xinyu Lin, Xin Zhang, Wenjie Wang, Wenjie Li, and Liqiang Nie. R^2ec: Towards large recommender models with reasoning. In *The 39th Annual Conference on Neural Information Processing Systems*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *International Conference on Machine Learning*, pp. 58484–58509. PMLR, 2024.

Luankang Zhang, Yonghao Huang, Hang Lv, Xuyang Zhi, Mingjia Yin, Yuyang Ye, Wei Guo, Hao Wang, and Enhong Chen. Why thinking hurts: Diagnosing and rectifying linguistic inertia in large language models for recommendation. *arXiv preprint arXiv:2602.16587*, 2026.

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)*, pp. 1435–1448. IEEE, 2024.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Yaochen Zhu, Harald Steck, Dawen Liang, Yinhan He, Vito Ostuni, Jundong Li, and Nathan Kallus. Rank-grpo: Training llm-based conversational recommender systems with reinforcement learning. *arXiv preprint arXiv:2510.20150*, 2025.

APPENDIX

A Notation and Symbols **15**

B Related Work **15**

B.1 Generative Recommendation with Semantic Identifiers 15

B.2 Reasoning and Reinforcement Learning for Large-Language-Model (LLM)-based Recommendation 15

C Limitations **16**

D Broader Impacts. **16**

E Theoretical Proofs and Discussions **16**

E.1 Formal Statement and Proof of the Consistency Claim 17

F Implementation Layout and Reasoning-Step Pairing **19**

G Baseline Descriptions **19**

H Dataset Details **19**

I Dataset and LLM Input Examples **20**

I.1 Per-Category Dataset Examples. 20

I.2 Prompt Template and Instantiated Example 21

J Implementation Details **22**

K Training Details **22**

L Training Efficiency **23**

M Extended Training Dynamics **23**

N Extended Performance Comparison **24**

O Scaling to Larger Language Models **24**

P Reasoning-Step Alignment Ablation **25**

Q Sensitivity to the Reasoning-Trace Generator. **25**

R Extended Case Studies **25**

A Notation and Symbols

We summarize the mathematical notations used throughout the paper in Table 3.

Table 3: Summary of frequently used notation.

Symbol	Domain	Meaning
$\mathcal{C}$	catalog	Item catalog.
$v, v^\star, \hat{v}$	item	An item, the ground-truth next item, and the predicted item.
$x = (v_1, \ldots, v_{T_x})$	sequence	Chronologically ordered user interaction history used as the sequential-recommendation prompt.
K	integer	Number of SID codebook levels; we use $K{=}3$ in all experiments.
$\mathcal{V}^{(k)}$	set	Vocabulary of the k-th SID codebook.
$\phi(v)$	mapping	Hierarchical SID tuple assigned to item v.
$s^{(k)}$	SID token	SID token at codebook level k.
$s^{\mathrm{gt},(k)}$	SID token	Ground-truth SID token at codebook level k.
y_i	rollout	The i-th sampled rollout for a prompt in a GRPO group.
$\tau^{(k)}$	token span	Thinking block paired with the level-k SID token.
$y_i^{(k)}$	step unit	Reasoning step consisting of $\tau_i^{(k)}$ and $s_i^{(k)}$.
G	integer	Number of rollouts sampled per prompt for group-relative normalization.
$r(x, y_i)$	scalar	Outcome reward used by standard GRPO.
$r_{i,k}$	scalar	SAPO reward assigned to rollout i at SID level k.
$\hat{A}_i, \hat{A}_{i,k}$	scalar	Rollout-level GRPO advantage and reasoning-step SAPO advantage.
$\tilde{A}_{i,k}$	scalar	Step-normalized advantage applied to the tokens of reasoning step k.
$w_{i,t}(\theta)$	ratio	Importance ratio for token t in rollout i.
ϵ	scalar	Clipping radius.
α, β	scalar	Coefficients for the per-step match reward and final-step format bonus.

B Related Work

B.1 Generative Recommendation with Semantic Identifiers

Generative recommendation has emerged as an alternative to discriminative matching: each item is represented as a short sequence of discrete tokens, and the recommender is trained as a language model over these tokens (Rajput et al., 2023; Deng et al., 2025; Lee et al., 2022). A central design choice is the tokenization scheme: residual-quantization VAEs produce hierarchical *semantic identifiers* (SIDs) that capture coarse-to-fine item semantics, which both improves generalization to unseen items and enables compact generation (Rajput et al., 2023; Lee et al., 2022). Our work inherits this representation and shows how the K-level structure of SIDs provides a natural scaffold for step-level credit assignment, an observation that, to the best of our knowledge, has not been leveraged by prior work on generative recommenders, including recent reasoning-based systems such as SIDReasoner (He et al., 2026) that still rely on a scalar outcome reward.

B.2 Reasoning and Reinforcement Learning for Large-Language-Model (LLM)-based Recommendation

Following the success of reinforcement learning from verifiable rewards on math and code (Guo et al., 2025; Lambert et al., 2024), several recent systems apply group-relative policy-optimization algorithms (Shao et al., 2024; Yu et al., 2025; Liu et al., 2025b; Zheng et al., 2025) to LLM-based recommenders. The systems most closely related to ours include SIDReasoner (He et al., 2026), OneRec-Think, R^{2}ec, ReaRec, and RecCoT (Liu et al., 2025a; You et al., 2025; Tang et al., 2025; Yang et al., 2025b), all of which augment generative recommenders with chain-of-thought reasoning and train them with a scalar outcome reward fed into GRPO or a close variant. SAPO directly addresses this misalignment by assigning rewards at the per-step granularity; it can be viewed as a form of process-reward RL (Lightman et al., 2024; Uesato et al., 2022) that obtains per-step supervision directly from the codebook hierarchy, without any human-annotated or learned reward

model. Concurrently, Rank-GRPO (Zhu et al., 2025) proposes rank-level credit assignment for conversational recommenders; our work is complementary in that it targets the *codebook hierarchy* of SID decoding rather than the *positional hierarchy* of a recommendation list. A complementary diagnosis is provided by (Zhang et al., 2026), who identify an *inference-time* failure mode of reasoning-based generative recommenders: long chain-of-thought traces induce a textual-inertia bias in which a generic-text subspace dominates inference and dilutes the SID evidence needed to decode the correct item. Our analysis targets a different, *training-time* failure mode: rollout-level credit diffused over reasoning tokens and SID-token predictions. The two diagnoses are nevertheless consistent. An outcome-reward gradient budget spread indiscriminately over reasoning tokens reinforces the same reasoning-token distribution whose dominance (Zhang et al., 2026) observe at inference time, whereas SAPO concentrates the budget on the verifiable SID tokens.

C Limitations

We highlight several directions for extending SAPO beyond the present study. *Evaluation scale and diversity.* Our experiments focus on three English Amazon Reviews categories with a fixed $K = 3$ SID hierarchy. Future work can evaluate SAPO on larger catalogs, deeper or less balanced SID hierarchies, cross-lingual item descriptions, and domains with different interaction dynamics. It would also be useful to study how step-aligned credit assignment behaves under different initialization regimes, including weaker Stage 2 checkpoints. *Richer feedback settings.* This paper uses ground-truth SID tuples to obtain clean per-level verifiable feedback. A natural next step is to extend SAPO to settings with multiple acceptable next items, noisy implicit feedback, or production-oriented click and conversion signals. These settings may require combining SID-level match rewards with calibrated relevance, diversity, or user-satisfaction objectives. *System and decoding variants.* Our implementation uses a blocked layout in which the policy generates all K thinking blocks before decoding the contiguous SID block. Future work can explore alternative response layouts, decoding constraints, and reward schedules, including interleaved reasoning–SID generation, adaptive step weighting, and deeper integration with trie-constrained decoding. These extensions may further improve efficiency, controllability, and deployment flexibility.

D Broader Impacts

SAPO improves the training of generative recommenders that decode items as hierarchical SIDs. On the positive side, per-step credit assignment reduces the tendency of outcome-reward training to inflate reasoning length and collapse reward, which translates into more sample-efficient RL fine-tuning and more interpretable (coarse-to-fine) reasoning traces that practitioners can audit. On the negative side, any improvement to the accuracy of personalized recommenders carries the standard risks of the broader generative-recommendation setting: filter-bubble reinforcement, preference manipulation by adversarial content producers, and information asymmetry between platforms and users. Our contribution targets the RL credit-assignment layer and does not introduce new data sources or deployment pathways beyond those already common in the literature, so we do not identify a novel direct path to harm; we nonetheless encourage downstream practitioners who build on SAPO to pair it with diversity-aware reward shaping and with the auditing tools proposed in the fairness and filter-bubble literature.

E Theoretical Proofs and Discussions

This appendix formalizes the objective-consistency claim used in Section 4.2. Appendix E.1 proves that the cumulative match component of SAPO's per-step reward equals α times the number of matched SID positions, preserves the exact-match optimum under realizability, and becomes an aligned marginal refinement when exact matching is not realizable. Appendix F formalizes the implementation decoding order in Eq. (4) and explains why the paired unit $(\tau^{(k)}, s^{(k)})$ remains the relevant credit-assignment unit. These results do not modify the SAPO method of Section 4; they only formalize the consistency claim and the implementation details behind step-aligned credit assignment.

E.1 Formal Statement and Proof of the Consistency Claim

For a rollout y_i, let

$$\phi(\hat{v}_i) = (s_i^{(1)}, \ldots, s_i^{(K)}), \qquad \phi(v_i^\star) = (s^{\mathrm{gt},(1)}, \ldots, s^{\mathrm{gt},(K)}) \tag{11}$$

denote the generated and ground-truth SID tuples. Define the number of matched SID positions as

$$\#\mathrm{Match}\big(\phi(\hat{v}_i), \phi(v_i^\star)\big) := \sum_{k=1}^{K} \mathbb{1}\left[s_i^{(k)} = s^{\mathrm{gt},(k)}\right]. \tag{12}$$

Recall that SAPO's per-step reward in Eq. (5) contains a match component

$$m_{i,k} := \alpha\, \mathbb{1}\left[s_i^{(k)} = s^{\mathrm{gt},(k)}\right]. \tag{13}$$

We analyze this match component separately from the final-step format bonus.

Proposition 2 (Per-step match rewards and outcome optima)**.** *Let Π be a policy class over rollouts, and define*

$$\mathcal{J}_{\mathrm{out}}(\pi) := \mathbb{E}_{x,y\sim\pi(\cdot|x)}\left[\mathbb{1}(\phi(\hat{v}) = \phi(v^\star))\right], \tag{14}$$

and

$$\mathcal{J}_{\mathrm{match}}(\pi) := \mathbb{E}_{x,y\sim\pi(\cdot|x)}\left[\sum_{k=1}^{K} m_k\right]. \tag{15}$$

Then the following statements hold.

1. ***Identity.*** *For every rollout y_i,*

$$\sum_{k=1}^{K} m_{i,k} = \alpha\, \#\mathrm{Match}\big(\phi(\hat{v}_i), \phi(v_i^\star)\big). \tag{16}$$

2. ***Optimum preservation under realizability.*** *Suppose there exists a policy $\pi^\star \in \Pi$ such that*

$$\Pr_{y\sim\pi^\star(\cdot|x)}\left[\phi(\hat{v}) = \phi(v^\star)\right] = 1 \qquad \textit{for every } x \in \mathcal{D}_{\mathrm{RL}}. \tag{17}$$

   Then

$$\arg\max_{\pi\in\Pi} \mathcal{J}_{\mathrm{out}}(\pi) = \arg\max_{\pi\in\Pi} \mathcal{J}_{\mathrm{match}}(\pi). \tag{18}$$

3. ***Aligned refinement under non-realizability.*** *Without the realizability assumption,*

$$\mathcal{J}_{\mathrm{match}}(\pi) = \alpha \sum_{k=1}^{K} R_k(\pi), \tag{19}$$

   where

$$R_k(\pi) := \Pr_{x,y\sim\pi(\cdot|x)}\left[s^{(k)} = s^{\mathrm{gt},(k)}\right] \tag{20}$$

   is the level-k marginal match probability. By contrast,

$$\mathcal{J}_{\mathrm{out}}(\pi) = \Pr_{x,y\sim\pi(\cdot|x)}\left[\bigwedge_{k=1}^{K} s^{(k)} = s^{\mathrm{gt},(k)}\right], \tag{21}$$

   the joint probability that all SID positions match. Thus maximizing $\mathcal{J}_{\mathrm{match}}$ maximizes the uniform sum of level-wise marginal match probabilities, providing an aligned refinement of the single joint exact-match objective.

Proof. **Part 1.** By definition,

$$\frac{m_{i,k}}{\alpha} = \mathbb{1}\left[s_i^{(k)} = s^{\mathrm{gt},(k)}\right]. \tag{22}$$

Summing over k gives the number of SID positions on which the generated and ground-truth tuples agree:

$$\sum_{k=1}^{K} \frac{m_{i,k}}{\alpha} = \sum_{k=1}^{K} \mathbb{1}\left[s_i^{(k)} = s^{\mathrm{gt},(k)}\right] = \#\mathrm{Match}\big(\phi(\hat{v}_i), \phi(v_i^\star)\big). \tag{23}$$

Multiplying both sides by α proves the identity.

Part 2. For any policy π, the outcome objective satisfies

$$0 \le \mathcal{J}_{\mathrm{out}}(\pi) \le 1. \tag{24}$$

Under the realizability assumption, the upper bound is attainable. Therefore, a policy maximizes $\mathcal{J}_{\mathrm{out}}$ if and only if it generates the ground-truth SID tuple almost surely for every prompt.

For the cumulative match reward, Part 1 implies

$$0 \le \sum_{k=1}^{K} m_k \le K\alpha. \tag{25}$$

The upper bound $K\alpha$ is attained for a rollout if and only if

$$s^{(k)} = s^{\mathrm{gt},(k)} \qquad \text{for all } k \in [K], \tag{26}$$

which is equivalent to

$$\phi(\hat{v}) = \phi(v^\star). \tag{27}$$

Since the quantizer ϕ is injective on the catalog, this is also equivalent to exact item recovery. Thus a policy maximizes $\mathcal{J}_{\mathrm{match}}$ if and only if it generates the ground-truth SID tuple almost surely for every prompt. The maximizers of $\mathcal{J}_{\mathrm{out}}$ and $\mathcal{J}_{\mathrm{match}}$ are therefore identical over Π.

Part 3. By Part 1 and linearity of expectation,

$$\mathcal{J}_{\mathrm{match}}(\pi) = \mathbb{E}_{x,y\sim\pi(\cdot|x)}\left[\sum_{k=1}^{K} m_k\right] = \alpha \sum_{k=1}^{K} \Pr_{x,y\sim\pi(\cdot|x)}\left[s^{(k)} = s^{\mathrm{gt},(k)}\right] = \alpha \sum_{k=1}^{K} R_k(\pi). \tag{28}$$

In contrast,

$$\mathcal{J}_{\mathrm{out}}(\pi) = \Pr_{x,y\sim\pi(\cdot|x)}\left[\bigwedge_{k=1}^{K} s^{(k)} = s^{\mathrm{gt},(k)}\right]. \tag{29}$$

This joint exact-match probability is bounded above by each marginal:

$$\mathcal{J}_{\mathrm{out}}(\pi) \le R_k(\pi), \qquad \forall k \in [K], \tag{30}$$

and hence

$$\mathcal{J}_{\mathrm{out}}(\pi) \le \min_{k\in[K]} R_k(\pi). \tag{31}$$

Therefore, two policies can have the same exact-match probability while differing in their level-wise marginals. The cumulative match reward distinguishes such policies by assigning credit to matched SID positions, and is therefore an aligned marginal refinement of the outcome objective. □

Remark (informative rollout groups). The same distinction appears in the group-relative advantages used during RL. Outcome-reward GRPO produces nonzero group-relative contrast only when the exact-match indicators differ across rollouts in the same group. Equivalently, the group must contain at least two rollouts with different values of

$$\mathbb{1}[\phi(\hat{v}) = \phi(v^\star)]\,. \tag{32}$$

SAPO instead computes reasoning-step advantages from the level-wise indicators

$$\mathbb{1}\left[s^{(k)} = s^{\mathrm{gt},(k)}\right], \qquad k \in [K]. \tag{33}$$

Thus, for a fixed level k, SAPO has nonzero step-level contrast whenever these level-k indicators differ across rollouts. This condition is strictly weaker than requiring different exact-match indicators. For example, if all rollouts miss the final item but some match the first SID level while others do not, the outcome advantage vanishes for the whole group, whereas SAPO still obtains a nonzero advantage at level 1. More generally, whenever sampled rollouts disagree on any SID position, SAPO can provide a step-level training signal even if every rollout is an exact-match failure. Therefore, step-aligned advantages are informative on a strictly larger set of rollout groups than outcome-level advantages.

F Implementation Layout and Reasoning-Step Pairing

The policy we actually train and release follows the implementation layout of Eq. (4), in which all K thinking blocks are generated first and the K SID tokens are decoded as a contiguous block. Under this layout, SAPO defines reasoning step k by pairing the k-th thinking block $\tau^{(k)}$ with the k-th SID token $s^{(k)}$. This pairing is used only to assign credit; it is not an alternative autoregressive factorization of the model.

The distinction matters because under Eq. (4) every $s^{(k)}$ is conditioned on the full reasoning trace, i.e., all K thinking blocks, and the previously generated SID tokens, not merely on a local step fragment. We adopt the blocked layout because (a) the contiguous SID block enables trie-based constrained decoding and beam search directly over the K-level codebook while reusing the KV cache across SID positions, and (b) later thinking blocks can refine earlier ones before any SID is committed.

SAPO therefore never relies on an alternative decode order. Per-step match rewards are placed on SID-token positions (Section 4.2), and per-step advantages are computed from per-step group statistics (Section 4.3); both depend only on the extracted reasoning steps $(\tau^{(k)}, s^{(k)})$ and the corresponding level-k match indicators. All definitions and derivations in Sections 3–4.3 and in Appendix E therefore remain valid under the implementation layout, with $|y^{(k)}| = |\tau^{(k)}| + 1$ denoting the size of the reasoning step.

G Baseline Descriptions

Traditional sequential recommenders. GRU4Rec (Hidasi et al., 2015) models user sessions with gated recurrent units (GRUs) and ranks items by session-parallel mini-batch training. **SASRec** (Kang & McAuley, 2018) applies a left-to-right Transformer to the interaction sequence, predicting the next item with a causal self-attention mechanism. **Caser** (Tang & Wang, 2018) treats the interaction sequence as a matrix and applies horizontal and vertical convolutional filters to capture point-level and union-level sequential patterns.

Generative recommenders without reasoning. TIGER (Rajput et al., 2023) represents items as sequences of SIDs produced by an RQ-VAE and trains a Transformer to autoregressively generate SID tuples for the next item. **HSTU** (Zhai et al., 2024) is a large-scale generative retrieval model that encodes heterogeneous user actions with a hierarchical sequence Transformer. **LCRec** (Zheng et al., 2024) aligns a pre-trained language model with collaborative item IDs via a lightweight adapter, enabling generative recommendation without full fine-tuning. **LETTER** (Wang et al., 2024) learns item tokenization end-to-end with the generative recommender, jointly optimizing the codebook and the decoder.

Reasoning-based generative recommenders. SIDReasoner (He et al., 2026) augments a generative recommender with chain-of-thought reasoning and trains Stage 3 with outcome-reward RL; its Stage 1 and Stage 2 setup matches the one used in this work, which makes it the closest point of comparison. **OneRec-Think** (Liu et al., 2025a) extends the OneRec framework with a reasoning module and outcome-reward GRPO training. **ReaRec** (Tang et al., 2025) uses a think-then-recommend paradigm with reasoning traces distilled from a teacher model, followed by outcome-reward RL fine-tuning.

H Dataset Details

Source and preprocessing. We use the Amazon Reviews corpus (Ni et al., 2019) and follow the 5-core sequential recommendation protocol of (Kang & McAuley, 2018): users and items with fewer than five interactions are filtered out, and the remaining interactions are sorted chronologically per user. We restrict the timespan to October 2016–November 2018 so that all three categories share a comparable activity window. Each user's interaction sequence is split leave-one-out: the last item is held out for test, the second-to-last for validation, and the remaining prefix for training (with all valid (history, target) sliding-window pairs used as training samples).

Per-category statistics. Table 4 reports the final split sizes used throughout the paper. The three categories span a range of catalog sizes (3.5k–3.9k items) and activity densities (avg. sequence length 5.96–6.45), providing a consistent but non-trivial test bed for generative recommendation.

Table 4: Dataset statistics for the three Amazon Reviews categories after 5-core filtering and leave-one-out splitting. '#Train/#Valid/#Test' count (history, target) pairs; 'Avg. seq. len.' is the average number of observed interactions per test user (including the test target).

Category	#Users	#Items	#Train	#Valid	#Test	Avg. seq. len.
Office-Products	4,866	3,459	38,924	4,866	4,866	5.97
Video-Games	6,142	3,858	49,133	6,142	6,142	6.45
Industrial-and-Scientific	4,533	3,686	36,259	4,532	4,533	5.96

SID tokenization (RQ-VAE). For each category we learn a residual-quantization variational autoencoder (RQ-VAE) over item text embeddings concatenating title, category, brand, and description. The quantizer has $K=3$ codebook levels with 256 entries per level, yielding a 3-tuple SID per item of the form $\langle a_{i_1}\rangle\langle b_{i_2}\rangle\langle c_{i_3}\rangle$ with $i_k \in \{0,\ldots,255\}$. The RQ-VAE is trained per category with mean squared error (MSE) reconstruction loss and commitment loss until all item embeddings receive a unique SID tuple; because $K=3$ levels at 256^3 codes far exceed the catalog size, a single pass suffices.

Codebook utilization. The effective codebook utilization (fraction of the 256 entries at each level that appear in at least one item's SID) is reported in Table 5. Level 1 is sparsely used (19–34%), reflecting the small number of broad categorical clusters in each dataset, whereas levels 2 and 3 are fully utilized. This pattern motivates SAPO's per-step credit assignment: since the prediction difficulty per step is highly asymmetric, a single scalar advantage cannot reflect which SID-token position the policy gets right or wrong.

Table 5: Codebook utilization per level across the three categories (# unique codes observed among 256 entries).

Category	Level 1 (a)	Level 2 (b)	Level 3 (c)
Office-Products	88	256	256
Video-Games	60	256	256
Industrial-and-Scientific	48	256	256

I Dataset and LLM Input Examples

This appendix complements Appendix H by providing concrete examples of the raw item/user/interaction records, the learned $K=3$ SID tuples, and the natural-language prompt template used by the LLM at training and inference time.

I.1 Per-Category Dataset Examples

Below we show representative records for each of the three Amazon Reviews categories. All values are drawn from the filtered and RQ-VAE-tokenized corpus described in Appendix H.

Office-Products.

- *Item example*: item_id=B000GHJL2C, title="Sanford Uni-Ball Vision Needle Micro Roller-ball Pen", category="Office Products → Pens & Writing → Pens", brand="Uni-ball", description="Smooth, ultra-fine 0.5 mm tip; archival-quality pigment ink".
- *SID encoding*: $(s^{(1)}, s^{(2)}, s^{(3)}) = (\langle a_{17}\rangle, \langle b_{93}\rangle, \langle c_{41}\rangle)$ (3-level RQ-VAE over text embeddings of title + category + brand + description).
- *User example*: user_id=A1P...XQ, #interactions= 8 (after 5-core filtering).
- *Interaction example (test record)*: user_id=A1P...XQ, item_sid=<a_17><b_93><c_41>, history_item_sid=[<a_3><b_12><c_201>, <a_17><b_88><c_14>, <a_5><b_220><c_77>], timestamp=2018-09-24.

Video-Games.

Prompt Template (Stage 3 RL rollout).

System: *Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Can you recommend the next item for the user based on their interaction history?*

User: The user has sequentially interacted with items ⟨history_sid_list⟩. Can you recommend the next item for him? Let's think step by step before making recommendation. Directly output the item SID after thinking.

Expected assistant response:
<think> ⟨reasoning trace 1⟩ </think> <think> ⟨reasoning trace 2⟩ </think> <think> ⟨reasoning trace 3⟩ </think> ⟨SID level 1⟩⟨SID level 2⟩⟨SID level 3⟩

Figure 6: Prompt template used for Stage 3 RL training. Placeholders in $\langle\cdot\rangle$ are filled with per-example content: ⟨history_sid_list⟩ is a comma-separated list of SID tuples drawn from the user's interaction history, and the expected response is $K{=}3$ <think>...</think> thinking blocks (one per codebook level, matching the decoding layout of Eq. (4)) followed by the $K{=}3$ SID tokens for the predicted next item. Constrained decoding over the learned SID trie guarantees that every sampled SID is a valid item under the per-category codebook.

- *Item example*: item_id=B00E46CV88, title="Nintendo Selects: Super Mario Galaxy", category="Video Games → Legacy Systems → Nintendo Systems → Wii", brand="Nintendo", description="Platforming adventure featuring spherical worlds and gravity-based puzzles".
- *SID encoding*: $(s^{(1)}, s^{(2)}, s^{(3)}) = (\langle a_6\rangle, \langle b_{145}\rangle, \langle c_{182}\rangle)$.
- *User example*: user_id=A2K...WB, #interactions= 11.
- *Interaction example (test record)*: user_id=A2K...WB, item_sid=<a_6><b_145><c_182>, history_item_sid=[<a_6><b_210><c_35>, <a_2><b_102><c_19>, <a_6><b_145><c_248>, <a_31><b_7><c_60>], timestamp=2017-12-02.

Industrial-and-Scientific.

- *Item example*: item_id=B003U6XQ12, title="Fluke 117 Electricians True RMS Multimeter", category="Industrial & Scientific → Test & Measurement → Multimeters", brand="Fluke", description="True-RMS voltage and current readings with non-contact voltage detection".
- *SID encoding*: $(s^{(1)}, s^{(2)}, s^{(3)}) = (\langle a_{22}\rangle, \langle b_4\rangle, \langle c_{118}\rangle)$.
- *User example*: user_id=A9R...LN, #interactions= 6.
- *Interaction example (test record)*: user_id=A9R...LN, item_sid=<a_22><b_4><c_118>, history_item_sid=[<a_22><b_4><c_31>, <a_13><b_198><c_9>, <a_22><b_55><c_240>], timestamp=2018-04-15.

The history and target are both expressed as SID tuples so that the entire retrieval/ranking objective reduces to next-item SID prediction; the item text fields are only used during SID tokenization and for human inspection, never shown to the LLM.

I.2 Prompt Template and Instantiated Example

The LLM receives a two-message chat prompt: a system instruction that frames the task and a user message containing the SID-valued interaction history. Figure 6 shows the template with placeholders, and Figure 7 instantiates it with a concrete Office-Products record together with a representative SAPO rollout following the implementation layout of Eq. (4): $K{=}3$ contiguous thinking blocks, one per codebook level, followed by the $K{=}3$ SID tokens under trie-based constrained decoding.

Prompt Example (`Office-Products`).

System: *Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Can you recommend the next item for the user based on their interaction history?*

User: The user has sequentially interacted with items `<a_3><b_12><c_201>`, `<a_17><b_88><c_14>`, `<a_5><b_220><c_77>`. Can you recommend the next item for him? Let's think step by step before making recommendation. Directly output the item SID after thinking.

Ground-truth next item: `<a_17><b_93><c_41>` (Sanford Uni-Ball Vision Needle Micro Rollerball Pen).

Example SAPO rollout (full-match case):

`<think>` The user's recent purchases cluster around general office writing supplies, with a strong signal from `<a_17>` (an office-supplies coarse code that appears in the second history item). The fine-grained codes span pens and similar stationery, so the next item likely sits under the same level-1 cluster. `</think>` `<think>` Narrowing to level 2, the user has interacted with sub-code `<b_88>` (ballpoint writing); a nearby sub-code around `<b_93>` covers rollerball/gel-ink pens, which is a natural next purchase given the existing ballpoint history. `</think>` `<think>` Within that level-2 cluster, level-3 variants around `<c_14>` to `<c_45>` correspond to fine-point rollerball variants; `<c_41>` specifically matches a smooth ultra-fine needle-tip rollerball, consistent with the item description style of the user's prior purchases. `</think>` `<a_17><b_93><c_41>`

Figure 7: A concrete prompt instance from `Office-Products`. The user's history is represented entirely as SID tuples, and the rollout generates $K{=}3$ `<think>...</think>` thinking blocks (one per codebook level, matching the decoding layout of Eq. (4)) followed by the $K{=}3$ SID tokens. In this example the rollout recovers the ground-truth target exactly (all three levels correct); see Section 5.5 and Appendix R for side-by-side comparisons with outcome-reward rollouts under the same prompt. The per-step match reward of Eq. (5) is non-zero at every codebook level for this rollout (all three per-step match indicators fire), which is exactly the per-step match signal SAPO exploits.

J Implementation Details

Default backbone. We use Qwen3-1.7B (Yang et al., 2025a) as the primary backbone and share the Stage 1 and Stage 2 checkpoints across all RL variants so that every comparison isolates the effect of the Stage 3 algorithm. Stages 1 and 2 are trained with AdamW, a cosine schedule, and a 5% warm-up on the enriched SFT and reasoning-activation datasets described in Appendix K. In Stage 3 we build on the VERL (Sheng et al., 2025) training framework with vLLM-based rollouts. For each prompt we sample $G = 16$ rollouts with temperature and top-p both set to 1.0, a maximum response length of 1024 tokens, and a mini-batch of 256 prompts per update. The actor learning rate is 1×10^{-6} with a 2% warm-up, the KL coefficient against the Stage 2 reference is set to 10^{-3} (low-variance KL), the entropy coefficient is 0, and the clip range ϵ is 0.2 for both SAPO and the outcome-reward baselines. The released SAPO implementation assigns the per-step match reward of Eq. (5) at SID-token positions, attaches the format bonus only to the last SID position, and requires exactly K extracted thinking blocks before any per-step reward is credited.

Reproducibility. All experiments run on 4 NVIDIA GPUs. Full hyperparameters and compute budgets are provided in Appendix K.

K Training Details

Stages 1–2 use the HuggingFace `Trainer`, and Stage 3 builds on VERL (Sheng et al., 2025) with vLLM-based rollouts. Table 6 summarizes the key hyperparameters per stage for the default Qwen3-1.7B backbone.

Compute budget. On 4×NVIDIA H100 (80 GB) GPUs, end-to-end training of the default Qwen3-1.7B pipeline takes approximately 20 hours for Stage 1, 1 hour for Stage 2, and 48 hours for Stage 3 per category (wall-clock). Stage 3 dominates total cost and scales roughly linearly with the number of rollouts per prompt.

Table 6: Hyperparameters per training stage for the Qwen3-1.7B backbone. All stages use AdamW on 4 GPUs, with the per-stage precision listed below. Early stopping on validation loss (patience 3) is active in Stages 1–2. Stage 3 follows the group-relative surrogate of Eq. (9) with the rollout parameters shown.

Hyperparameter	**Stage 1 (SFT)**	**Stage 2 (reasoning)**	**Stage 3 (RL)**
Initial checkpoint	Qwen3-1.7B (Yang et al., 2025a)	Stage 1 final	Stage 2 final
Optimizer	AdamW	AdamW	AdamW
Learning rate	3×10^{-4}	1×10^{-5}	1×10^{-6}
LR schedule	cosine	cosine	cosine
Warm-up	20 steps	10 steps	2% of total
Weight decay	0.0	0.0	0.0
Global batch size	1024 samples	1024 samples	256 prompts
Per-GPU micro-batch	1	8	8
Max epochs	10	5	10
Max sequence length	1024 tokens	1024 tokens	1024+1024
Precision	BF16	BF16	FP32
Stage 3 RL-specific.			
Rollouts per prompt G	16 (vLLM, temperature 1.0, top-$p=1.0$, GPU-mem util. 0.8)		
KL coefficient λ	10^{-3} (low-variance KL to the Stage 2 reference)		
Entropy coefficient	0		
Clip range ϵ	0.2 (SAPO and outcome-reward baselines)		
Reward coefficients	$\alpha=1.0$ (match), $\beta=0.2$ (format)		
Decoding constraint	trie-based constrained decoding over the K-level SID vocabulary		

L Training Efficiency

We report the computational overhead of SAPO relative to outcome-reward training in Table 7. The per-step reward computation and advantage normalization add minimal overhead because they operate on the $K=3$ SID positions per rollout rather than on all tokens. We measure wall-clock time over a five-step profiling run and report averages over steps 2–5 to exclude initialization and first-step warm-up effects.

Table 7: Training efficiency comparison on `Office-Products` (Qwen3-1.7B, 4× GPU). Peak memory denotes maximum allocated GPU memory.

Method	Wall-clock / step (s)	Peak GPU mem. (GB)	Reward+adv. / step (s)
Outcome	85.15	74.83	0.59
SAPO	85.03	74.89	1.13

SAPO introduces no measurable wall-clock slowdown in this short profiling run: its average step time is within 0.15% of the outcome-reward variant. The peak allocated GPU memory is also essentially unchanged (74.89 GB vs. 74.83 GB). The reward and advantage computation time increases from 0.59 s to 1.13 s per step, but the additional 0.54 s accounts for only 0.63% of a full update step, so it is dominated by rollout generation and actor updates. We therefore report the directly measured reward/advantage overhead rather than steps to convergence, since convergence in stochastic GRPO training is not a well-defined quantity for a short profiling run.

M Extended Training Dynamics

Figures 8 and 9 extend the main-text diagnostic from `Industrial-and-Scientific` to the remaining two categories. On `Video-Games`, GRPO exhibits a stronger instability pattern, with reward collapse or oscillation, response-length drift, rapid KL growth, and an unstable SID match rate, whereas SAPO keeps reward and SID-match dynamics more controlled. On `Office-Products`, the difference is milder but follows the same direction: SAPO improves reward and SID match rate more steadily, keeps response length comparable, and shows slower KL growth than GRPO.

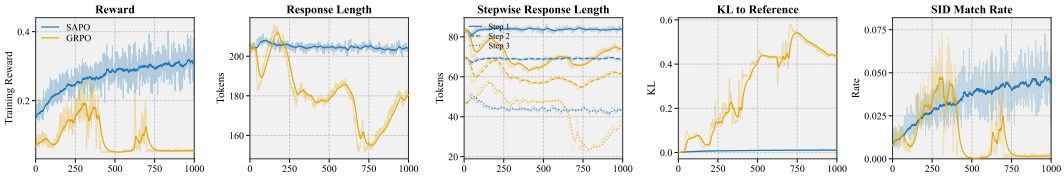

Figure 8: RL training dynamics on the `Video-Games` dataset. This figure follows the same diagnostic layout as Figure 3, comparing SAPO and GRPO from the same Stage 2 checkpoint using training reward, response length, stepwise response length, KL to the reference, and SID match rate; blue denotes SAPO and orange denotes GRPO.

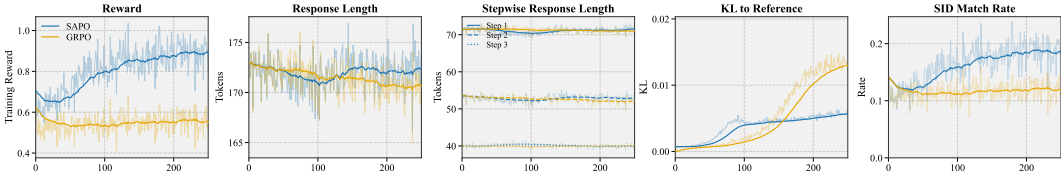

Figure 9: RL training dynamics on the `Office-Products` dataset.

Together with Figure 3, these per-category diagnostics are consistent with the RQ2 conclusion that reasoning-step credit assignment stabilizes sparse exact-match RL across datasets rather than only in the representative main-text run.

N Extended Performance Comparison

To study the performance of SAPO's gain, we rerun outcome-GRPO, SIDReasoner, and SAPO from the same Stage 2 checkpoint on `Industrial-and-Scientific` across three independent Stage 3 seeds. Bars report the mean over the three runs, and error bars denote ± 1 standard deviation. Figure 10 shows that SAPO improves all four metrics over outcome-GRPO while exhibiting a smaller standard deviation in every case, supporting the intuition that per-step advantage normalization improves training robustness. Relative to SIDReasoner, SAPO also achieves a higher mean on all four metrics, which indicates an advantage of step-aligned credit assignment over rollout-level reward shaping.

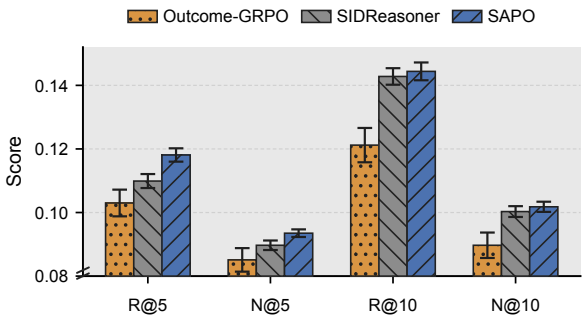

Figure 10: Performance comparison on the `Industrial-and-Scientific` dataset.

O Scaling to Larger Language Models

To test whether SAPO remains effective beyond the default Qwen3-1.7B backbone, we extend the comparison to Qwen3-4B and Qwen3-8B on `Industrial-and-Scientific`. All three model sizes use the same data split and evaluation protocol. For each size, we compare SAPO with SIDReasoner using Recall@k (R@k) and NDCG@k (N@k) at $k \in \{5, 10\}$.

Figure 11 shows that both methods improve on every metric as the backbone grows from 1.7B to 8B. SAPO also outperforms SIDReasoner in all 12 backbone-size and metric comparisons. The largest gains at 8B occur on N@5 and N@10, where SAPO improves the absolute scores by 0.0074 and 0.0055, respectively. These results show that SAPO's benefit is not specific to the default 1.7B backbone and persists as model capacity increases.

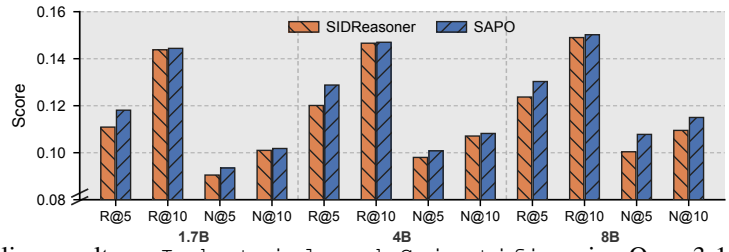

Figure 11: Scaling results on `Industrial_and_Scientific` using Qwen3-1.7B, Qwen3-4B, and Qwen3-8B. SAPO and SIDReasoner are evaluated under the same data split and evaluation protocol. Each backbone block reports R@5, R@10, N@5, and N@10. Paired orange and blue bars denote SIDReasoner and SAPO, respectively.

P Reasoning-Step Alignment Ablation

The ordered pairing in SAPO is defined by the SID hierarchy rather than by adjacency in the output sequence: successive SID levels refine the item representation from coarse to fine, and Stage 2 provides the same number of ordered thinking blocks, so SAPO uses their shared ordinal index as a structured prior for credit assignment. We retain a contiguous SID suffix because it enables trie-constrained beam search with KV-cache reuse, whereas interleaving free-form reasoning between SID levels would require additional generation along each search branch. We test whether this hierarchy-indexed correspondence affects optimization. Let $\hat{A}_{i,k}$ denote the advantage for SID step k. *Ordered (SAPO)* assigns $\hat{A}_{i,k}$ to $\tau_i^{(k)}$. *Reversed* assigns $\hat{A}_{i,K+1-k}$ to $\tau_i^{(k)}$, giving the mapping $1\leftarrow3$, $2\leftarrow2$, and $3\leftarrow1$ for $K=3$. *Random* samples a fresh permutation σ independently for each training example and assigns $\hat{A}_{i,\sigma(k)}$ to $\tau_i^{(k)}$. All variants use the same per-level rewards and keep SID-token credit unchanged, so $s_i^{(k)}$ receives $\hat{A}_{i,k}$; only the mapping from SID-level advantages to thinking blocks changes.

Figure 12 shows that preserving step-level signals is insufficient when their mapping to thinking blocks is permuted. Relative to ordered SAPO, the two misaligned variants reduce NDCG by 11.0–12.6% and Recall by 11.7–15.4% across the two cutoffs. Both the fixed reversal and the per-example random permutations degrade all four metrics, showing that the ordered block-to-level mapping is functionally important for credit assignment. Since Reversed is deterministic, the effect cannot be attributed solely to the additional assignment noise introduced by Random.

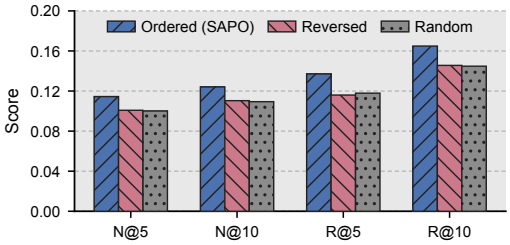

Figure 12: Reasoning-step alignment on `Office_Products`. Ordered denotes SAPO; Reversed and Random change only reasoning-block credit while preserving SID-token credit. N denotes NDCG and R denotes Recall.

Q Sensitivity to the Reasoning-Trace Generator

We examine whether recommendation quality depends on the language model used to generate the reasoning-activation traces for Stage 2. Table 8 reports results on `Office_Products` for three trace generators, with the rest of the training and evaluation pipeline held fixed.

At Stage 2 the generator choice spans 5.0% of relative R@10 and 4.4% of relative N@10 across the three trace generators. After Stage 3 these spreads narrow to 0.9% and 0.6%. Thus, in this setting, final recommendation quality is relatively insensitive to the reasoning-trace generator, and reinforcement learning further attenuates differences inherited from reasoning-activation.

R Extended Case Studies

This appendix provides additional qualitative examples complementing Figure 5. Each example follows the same layout as Figure 5: a shared history and target appear first, followed by side-by-side

Table 8: Sensitivity to the model used to generate Stage 2 reasoning-activation traces on `Office-Products`. Stage 3 uses SAPO. R@10 and N@10 denote Recall@10 and NDCG@10, respectively. The best result in each column is **bolded**.

	Stage 2 (Reasoning-activation)		Stage 3 (Reinforcement learning)	
Trace generator	R@10	N@10	R@10	N@10
GPT-4o-mini (OpenAI et al., 2024)	0.1410	0.1020	**0.1649**	0.1241
MuseSpark-1.3 (Menghini et al., 2026)	**0.1458**	**0.1049**	0.1643	**0.1248**
Qwen3.8-27B (Qwen Team, 2026)	0.1389	0.1005	0.1635	0.1240

outcome-GRPO and SAPO reasoning traces. SID tokens are shown in red. The thinking blocks below are copied from the raw model outputs; malformed outcome-GRPO outputs are shown as raw malformed thinking blocks rather than rewritten summaries.

Malformed Reasoning and Lost Series Signal (`Video-Games`).

Shared prompt history. <a_250><b_165><c_76> (Super Mario World); <a_219><b_191><c_170> (SNES-compatible console); <a_141><b_253><c_131> (Star Wars: Rogue Squadron); <a_250><b_14><c_196> (Super Mario 64); <a_205><b_82><c_228> (Cruis'n USA); <a_175><b_113><c_236> (Donkey Kong 64).
Target. <a_250><b_183><c_190>: Yoshi's Story.

Outcome GRPO

Thinking block 1. 16-bit Entertainment System - Playstation 2, identified as <a_219><b_191><c_170>, is a classic video game console that revolutionized the gaming industry with its extensive library of games and multimedia capabilities. This console is particularly appealing to users who appreciate retro gaming and nostalgia for the iconic PlayStation 2 era.
Thinking block 2. 16-bit Entertainment System - Playstation 2, identified as <a_219><b_191><c_170>, is a classic video game console that revolutionized the gaming industry with its extensive library of games and multimedia capabilities. This console is particularly appealing to users who appreciate retro gaming and nostalgia for the iconic PlayStation 2 era.
Thinking block 3. 28-in-1 Blue Game Card Case Holder Cartridge Box for Nintendo 3DS Video Games by NYKKOLA, identified as <a_219><b_116><c_229>, is a practical accessory for gamers who want to keep their game cards organized and easily accessible. This product is particularly appealing to users who own multiple Nintendo 3DS games and are looking for a way to store and manage their game cartridges efficiently.
Prediction. <a_219><b_116><c_229>: 28-in-1 Nintendo 3DS game-card case. ***Wrong; the target is not in top-10.***

SAPO

Thinking block 1. Analyzing the user's interaction history, I notice a strong inclination towards classic Nintendo games, particularly those that feature platforming and adventure elements. The user has engaged with multiple titles from the Super Mario franchise, such as <a_250><b_165><c_76> and <a_250><b_14><c_196>, indicating a preference for nostalgic gameplay experiences that emphasize exploration and character-driven narratives.
Thinking block 2. Additionally, the user has shown interest in action-adventure games like <a_141><b_253><c_131>, which suggests a broader appreciation for immersive storytelling and dynamic gameplay mechanics. The presence of <a_205><b_82><c_228> further indicates a willingness to explore different genres, yet the predominant focus remains on Nintendo's iconic titles.
Thinking block 3. Given this pattern, it's reasonable to predict that the user would be drawn to another classic platforming experience that aligns with their established interests. So the predicted next item is: <a_250><b_183><c_190>
Prediction. <a_250><b_183><c_190>: Yoshi's Story. ***Correct.***

Takeaway. Outcome-reward training produces malformed/repetitive blocks, whereas SAPO preserves the game-series signal.

Final-SID Refinement (`Industrial-and-Scientific`).

Shared prompt history. <a_79><b_24><c_40> (thermal paste); <a_36><b_103><c_152> (digital audio converter); <a_15><b_10><c_53> (Industrial & Scientific category); <a_94><b_242><c_43> (1000mL filtering flask).
Target. <a_94><b_93><c_57>: Buchner-style filter funnel.

Outcome GRPO

Thinking block 1. I notice the user's initial interest in <a_79><b_24><c_40> suggests a focus on technical components, possibly for computing or electronics, indicating a preference for precision and performance. The subsequent engagement with <a_36><b_103><c_152> shows an openness to practical, utility-driven devices that facilitate home or workspace improvements. Moving forward, the user's selection of <a_15><b_10><c_53> and <a_94><b_242><c_43> points to a growing interest in laboratory equipment.
Thinking block 2. I also observe a tendency toward items that offer versatility and compatibility with various setups, as seen in the choice of <a_36><b_103><c_152>. The user may enjoy exploring further laboratory accessories, specialized glassware, or other scientific tools that support hands-on experimentation.
Thinking block 3. Overall, the user appears motivated by products that enable precise work, facilitate experimentation, and improve their workspace or laboratory setup. I will recommend the next item as <a_94><b_93><c_171>, which aligns with their demonstrated interest in reliable, versatile laboratory equipment.
Prediction. <a_94><b_93><c_171>: generic glass funnel. ***Wrong final SID step.***

SAPO

Thinking block 1. I observe that the user has consistently engaged with items related to laboratory and technical equipment, such as <a_79><b_24><c_40>, <a_36><b_103><c_152>, <a_15><b_10><c_53>, and <a_94><b_242><c_43>. These selections suggest a strong interest in scientific apparatus, particularly those supporting experimental setups, precise measurements, and chemical handling.
Thinking block 2. The user appears to value durability, compatibility with lab environments, and items that facilitate hands-on scientific work. Their choices indicate a preference for tools and accessories that enhance laboratory efficiency and reliability.
Thinking block 3. Given this pattern, the user may enjoy further exploring laboratory glassware, measurement instruments, or other specialized equipment that supports scientific experimentation and technical tasks.
Prediction. <a_94><b_93><c_57>: Buchner-style filter funnel. ***Correct.***

Takeaway. Both methods keep the labware prefix, but SAPO corrects the final SID step using the filtering-flask cue.

Over-Continuity under a Sharp Product-Family Shift (`Office-Products`).

Shared prompt history. <a_102><b_62><c_137> (Sharpie yellow highlighters); <a_102><b_10><c_219> (Sharpie orange highlighters).
Target.
<a_187><b_147><c_193>
(Pendaflex file folders).

Outcome GRPO

Thinking block 1. I notice that the user has interacted with two items, both of which are high-quantity, functional office supplies designed for marking and highlighting text. Both items share the attribute <a_102>, indicating a strong preference for a particular brand or product line.
Thinking block 2. The user's choices suggest a focus on practicality and efficiency, with an emphasis on tools that facilitate organization and document management. The repeated selection of items with attributes <b_62> and <b_10> points to a tendency toward products that enhance productivity and support workflow.
Thinking block 3. Given this pattern, the user may enjoy other office supplies that offer organizational benefits, such as labeling, sorting, or document management solutions.
Prediction. <a_187><b_147><c_193>: Pendaflex file folders. ***Correct.***

SAPO

Thinking block 1. I notice the user has engaged with two items, both featuring the <a_102> attribute, which points to a consistent preference for a particular brand or product line. The first interaction involved <a_102><b_62><c_137>, a highlighter with a chisel tip, while the second was <a_102><b_10><c_219>, also a highlighter but with a different color and a larger pack size.
Thinking block 2. Both items emphasize practicality–bright color, smear-resistant ink, and versatility for highlighting text. The user appears motivated by utility and efficiency, possibly for tasks like studying, organizing, or office work.
Thinking block 3. Given this pattern, the user may enjoy other stationery items that offer similar functional benefits, such as multi-pack highlighters, markers, or organizational tools with ergonomic features or enhanced usability.
Prediction. <a_102><b_220><c_138>: Sharpie fluorescent pink highlighters. ***Wrong.***

Takeaway. SAPO does not eliminate all errors: when the next item breaks sharply from the observed family, it can overuse local continuity.

Recovering a Recency-Sensitive Product-Family Cue (`Office-Products`).

Shared prompt history. <a_41><b_50><c_26> (Brother ink); <a_41><b_41><c_111> (Canon toner); <a_170><b_39><c_252> (Avery foam stamp pad); <a_18><b_81><c_41> (Avery 3-ring binder); <a_130><b_228><c_138> (Staples invisible tape); <a_183><b_150><c_4> and <a_183><b_150><c_127> (LINKYO Canon toner); <a_255><b_117><c_30> (BIC black ballpoint pens); <a_31><b_53><c_79> (ACCO mini binder clips); <a_255><b_117><c_124> (BIC blue ballpoint pens).
Target. <a_255><b_117><c_102>: BIC Round Stic red ballpoint pens.

Outcome GRPO

Thinking block 1. Examining the user's interaction history, I notice a strong focus on essential office supplies, particularly those related to printing and document management. The repeated selection of items like <a_41><b_50><c_26>, <a_41><b_41><c_111>, <a_183><b_150><c_4>, and <a_183><b_150><c_127> suggests a preference for reliable, high-yield consumables, likely to support frequent printing tasks.
Thinking block 2. The inclusion of <a_170><b_39><c_252> and <a_130><b_228><c_138> points to an interest in versatile, everyday office tools, while <a_18><b_81><c_41> and <a_31><b_53><c_79> indicate a need for organization and document handling. The user also shows a tendency to purchase in bulk, as seen with <a_255><b_117><c_30> and <a_255><b_117><c_124>, which may reflect a preference for value and long-term supply.
Thinking block 3. Given these patterns, the user may enjoy exploring additional high-capacity, cost-effective printing supplies, organizational accessories, or bulk office consumables that support efficient document processing and management.
Prediction. <a_183><b_150><c_15>: LINKYO HP toner cartridge. ***Wrong; the target is not in top-10.***

SAPO

Thinking block 1. Examining the user's interaction history, I notice a strong focus on essential office and productivity supplies, particularly those related to printing and document management. The repeated selection of items like <a_41><b_50><c_26>, <a_41><b_41><c_111>, <a_183><b_150><c_4>, and <a_183><b_150><c_127> suggests a preference for reliable, high-yield consumables, likely to support frequent printing needs.
Thinking block 2. The inclusion of <a_170><b_39><c_252> and <a_130><b_228><c_138> points to an interest in versatile, everyday office tools, while <a_18><b_81><c_41> and <a_31><b_53><c_79> indicate a practical approach to organizing and managing paperwork. The user also shows a tendency to purchase in bulk, as seen with <a_255><b_117><c_30> and <a_255><b_117><c_124>, which may reflect a need for cost-effective, high-volume writing instruments.
Thinking block 3. Overall, this user tends to prefer functional, value-oriented supplies that support efficient document handling and organization, and may enjoy exploring additional bulk packs, multi-color options, or other practical office essentials.
Prediction. <a_255><b_117><c_102>: BIC Round Stic red ballpoint pens. ***Correct.***

Takeaway. SAPO preserves a recency-sensitive product-family cue that outcome GRPO washes out under a broader office-supplies summary.