Implicit Reasoning for Large Language Model-based Generative Recommendation

Anonymous ACL submission

Abstract

Large Language Models (LLMs) are increasingly adopted as backbones for Generative Recommendation (GR), promising access to pretrained world knowledge. Yet reliably invoking this knowledge for GR remains poorly understood. A key obstacle is that LLM-based GR typically represents items with Semantic IDs (SIDs), disrupting LLMs' natural-language reasoning interface because these tokens are unseen by the LLM during pretraining. Existing approaches address this with expensive multi-stage pipelines that ground SIDs and elicit explicit rationales, but offer limited insight into when and why each stage is necessary. In this work, we systematically decompose explicit reasoning training pipelines for LLM-based GR, revealing three key limitations: weakened world-knowledge verbalization, misalignment between SID and natural-language token embedding spaces, and sensitivity to rationale quality—all of which hurt explicit reasoning performance. To circumvent these issues, we propose PAUSEREC, a lightweight implicit reasoning paradigm tailored for GR. PAUSEREC is exceptionally practical, avoiding costly reasoning trace acquisition and reasoning alignment training, leading to a multitude of benefits: (1) it outperforms standard explicit CoT methods by up to 6.22%, (2) it reduces training cost by up to 65% GPU hours, and (3) it speeds up inference by up to 71.3%. These results position PAUSEREC as a lightweight alternative to explicit rationale generation, enabling more effective and efficient LLM-based GR. Our project code can be found at https://anonymous.4open.science/r/PauseRec-712B.

1 Introduction

Large Language Models (LLMs) have recently been adopted as backbones for Generative Recommendation (GR), enabling LLM-based GR systems that formulate recommendation as conditional generation: an LLM reads a user history and generates the next item (Hua et al., 2023; Bao et al., 2023; Rajput et al., 2024). The appeal of LLMs for GR lies in their pretrained world knowledge (Zhao et al., 2023; Huang and Chang, 2023; Yu et al., 2024). In principle, this knowledge can help infer semantic relationships among historical items, identify a user's latent intent, and map that intent to plausible next items beyond memorized co-occurrences (Wang et al., 2025; Zhang et al., 2025). Yet the process of efficiently and effectively accessing LLMs' pretrained knowledge for GR remains poorly understood (Zhang et al., 2026).

A key obstacle to leveraging an LLM's world knowledge for GR centers on item representation. Specifically, LLM-based GR systems typically represent items with Semantic IDs (SIDs), i.e., short sequences of special tokens derived from items' semantic relations (Rajput et al., 2023). SIDs make item generation tractable given their compactness, but they are not natural-language expressions and reside outside the pretrained LLM vocabulary (Li et al., 2021). This creates a mismatch: LLMs access world knowledge through natural language, while the recommendation task is to generate a non-linguistic SID conditioned on other non-linguistic SIDs. We therefore ask: *How can pretrained LLM world knowledge be effectively leveraged to improve recommendation over SID tokens?*

Following the broader LLM literature (Yu et al., 2024; Petroni et al., 2019), one natural answer to this question is leveraging explicit Chain-of-Thought (CoT) reasoning (Wei et al., 2022; Kojima et al., 2022)[1]. Explicit CoT has been shown to improve LLM performance on a range of knowledge-intensive domains, including mathematics (Imani et al., 2023), science (Truhn et al., 2023), and coding (Jiang et al., 2026). For LLM-based GR, previous methods have pursued a similar goal through

[1] In this work, we use both "reasoning" and "rationales" to refer to LLMs' Chain-of-Thought (CoT) process, i.e., the intermediate, step-by-step traces that LLMs generate before producing a final answer (Wei et al., 2022).

multi-step training pipelines. These pipelines typically ground LLMs in SIDs via continual pretraining (CPT) on natural-language item descriptions, optimize next-item prediction with supervised fine-tuning (SFT), elicit explicit rationales through SFT over reasoning trajectories (which we refer to as CoT SFT) , and refine model responses with reinforcement learning (RL) post-training (Liu et al., 2025; Yu et al., 2025; Liang et al., 2026). Yet existing work provides limited insight into when these stages are necessary and why they help. Given each stage's high computation cost, understanding these questions is critical to both justifying the full workflow and identifying more efficient alternatives.

To address this gap, we first analyze explicit reasoning pipelines for LLM-based GR, examining each stage's contribution and necessity. We begin with CPT stage, finding that CPT-trained models can recover coarse item categories but often struggle to identify titles or fine-grained categories, indicating that grounding provides a real but incomplete semantic signal. We then test whether CoT SFT with various reasoning formats, including template-based reasoning and teacher-generated reasoning, can improve recommendation performance. Across these variants, CoT SFT consistently *underperforms* simple next-item SFT. Performance gains from explicit CoT emerge only after expensive RL post-training.

To explain this discrepancy, we identify three limitations of explicit reasoning. First, we find that CoT SFT makes pretrained world knowledge harder to verbalize under standard decoding, even though this knowledge remains recoverable from the model's logits. Second, we show that text and SID token embeddings become geometrically separated during training. Our theoretical analysis proves that this separation limits the extent to which reasoning expressed in natural-language tokens can shape the final SID prediction. Third, we demonstrate that recommendation performance is sensitive to superficial perturbations of the ground-truth rationales. Together, these findings suggest that explicit rationales are a brittle interface for exploiting LLM knowledge in LLM-based GR.

To circumvent the aforementioned challenges, we propose PAUSEREC, a lightweight *implicit* reasoning framework for SID-based GR. Instead of crafting ground-truth natural-language rationales with expensive teacher models and training the model to generate those rationales, PAUSEREC inserts a short sequence of trainable <pause> tokens before SID generation. The <pause> token is initialized and pretrained to connect language and SID representations, then optimized only through the final next-item prediction objective, giving the model latent computation steps that directly shape SID prediction. PAUSEREC addresses the three issues of explicit reasoning pipelines by (i) removing reliance on verbalizing pretrained knowledge, (ii) bridging the text-SID representation gap through a trainable <pause> token, and (iii) avoiding brittle rationale supervision. On multiple Amazon review datasets, PAUSEREC outperforms SFT and CoT-based methods by up to 6.22%, while substantially simplifying explicit reasoning pipelines; it reduces training cost by up to 65% GPU hours and speeds up inference by 71.3%, positioning implicit reasoning as a stronger and more efficient alternative for LLM-based GR. Our contributions are as follows:

- **Diagnostic analysis.** We decompose explicit reasoning pipelines for LLM-based GR and identify why they fail without RL post-training, including incomplete SID grounding, weakened world-knowledge verbalization, text–SID embeddings mismatch, and sensitivity to rationale formats.
- **Implicit reasoning framework.** We introduce a novel pipeline termed PAUSEREC, which uses trainable <pause> tokens to elicit latent reasoning without rationale supervision.
- **Empirical evaluation.** Across three Amazon review datasets, PAUSEREC improves over standard SFT and CoT-based methods by up to 6.22% while reducing training and inference overhead.

2 Preliminaries

2.1 Problem Formulation

Following the GR literature (Liu et al., 2025), we consider the sequential recommendation task. Let $\mathcal{I}$ denote the set of all items. Given a user's n chronologically ordered interaction history $H = [i_1, i_2, \ldots, i_n]$ where $i_j \in \mathcal{I}$, the task is to predict the next item i_{n+1} that the user will interact with. Following recent work (Rajput et al., 2024; Bao et al., 2023), LLM-based GR represents each item $i \in \mathcal{I}$ with a Semantic ID (SID), i.e., a sequence of tokens $s_i = [s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(L)}]$ of length L that are added to the LLM's vocabulary. Recommendation can then be framed as conditional generation:

$$p(i_{n+1}|H) = p(s_{i_{n+1}}|\text{Prompt}(H)) \qquad (1)$$

where $\text{Prompt}(H)$ converts the interaction history into a natural-language prompt listing past items

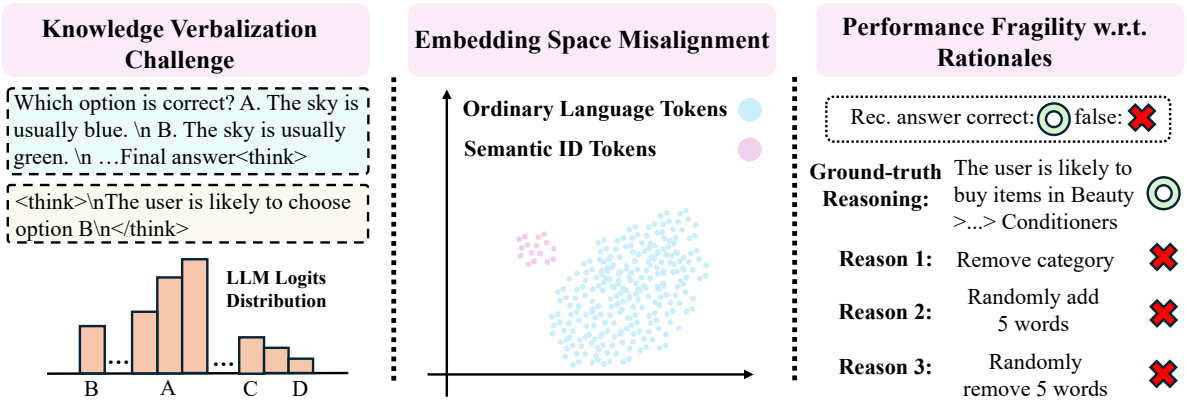

Figure 1: The three identified limitations for explicit CoT in SID-based GR. CoT SFT weakens world-knowledge verbalization (left), separates natural-language and SID embedding spaces (middle), and makes recommendation quality sensitive to rationale perturbations (right), motivating an implicit alternative to verbal rationales.

(and optionally metadata). All methods in this paper share this generative formulation; they differ in how reasoning is inserted before SID prediction.

2.2 Existing Explicit CoT Pipelines for GR

We introduce the multiple training stages of existing explicit reasoning pipelines (Liu et al., 2025; Liang et al., 2026) for GR as follows:

Continual Pretraining (CPT). The LLM is finetuned on an interleaved corpus of SIDs and item descriptions, with only SID token embeddings trainable. This stage grounds item semantics into SID token embeddings. Given item i with description d_i, the model is trained on:

$$\mathcal{L}_{\text{CPT}} = -\mathbb{E}_{(s_i,d_i)}\left[\log p(s_i|d_i) + \log p(d_i|s_i)\right] \tag{2}$$

Next-item Supervised finetuning (SFT). The CPT model is finetuned on user-items interaction histories to predict the next item by generating its SID:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(H,i_{n+1})}\left[\log p(s_{i_{n+1}}|\text{Prompt}(H))\right] \tag{3}$$

CoT SFT. After SFT, the model is finetuned to generate natural-language rationales before the target SID. The training objective pairs each history H, rationale r, and next item i_{n+1} as

$$\mathcal{L}_{\text{Reasoning}} = -\mathbb{E}_{(H,r,i_{n+1})}\left[\log p(r, s_{i_{n+1}}|\text{Prompt}(H))\right] \tag{4}$$

Here, rationales are method-specific: some (Liu et al., 2025) use reasoning templates, while others (Liang et al., 2026) utilize a teacher LLM.

Reinforcement Learning (RL) Post-training. Existing methods further apply RL to optimize recommendation rewards directly (Liu et al., 2025; Yu et al., 2025; Liang et al., 2026), though this stage is computationally expensive.

3 Contributions of the Training Stages

Given the current gap in understanding when and why different training stages make CoT effective for GR, we analyze the role of each stage in Section 2.2. We focus on CPT and CoT SFT here; next-item SFT and RL are evaluated in Section 6.

3.1 CPT: Can LLMs Recover SID Semantics?

The primary aim of CPT is to ground SID semantics in LLMs, based on the premise that LLMs can reason over SIDs only after understanding their semantics. Before examining reasoning-related stages, we ask how much item-level semantic information an LLM recovers from SIDs after CPT.

Experimental Design. We train a Qwen3-1.7B (Team, 2025) backbone on Amazon Beauty (Ni et al., 2019) with CPT for 2 epochs, where each SID is paired with its name and category during training. After CPT, we test whether the model can generate (1) **item titles** and (2) **item categories**[2] at 1-, 2-, and 3-level granularity. We prompt with each test SID and measure exact-match accuracy; prompts and decoding are in Appendix F.3.

Results and Analysis. From Table 1, we observe that (1) *Fine-grained understanding is poor:* title recovery stays near 0% and full-category accuracy remains below 7.2% on all datasets, so item-level semantics are largely unrecovered. (2) *Coarse category signal is strong:* 1-level category accuracy reaches highest 99.6% and 2-level accuracy up to

[2]In Amazon Beauty (Ni et al., 2019), categories are three-level paths, e.g., "Beauty > Hair Care > Conditioners."

Metric	Beauty	Sports	Toys
1-level Category	98.00	97.83	99.56
2-level Category	27.80	40.49	11.85
Category (Full)	7.19	1.30	6.74
Title Recovery	0.00	0.06	0.14

Table 1: SID metadata recovery after CPT. The model recovers coarse one-level categories almost perfectly, but fails on item titles and fine-grained categories, showing that SID grounding provides partial semantic information rather than precise item-level understanding.

Method	Hit@5	NDCG@5
Next-item SFT (Baseline)	0.0533	0.0381
Template-Category	0.0492	0.0343
Template-Extended	0.0425	0.0282
Gemini 3.1 Flash-Lite Free-form	0.0520	0.0361
Gemini 3.1 Pro Free-form	0.0422	0.0276
Gemini 3.1 Flash-Lite Rejection	0.0526	0.0362
Gemini 3.1 FL Gemini Rejection	0.0524	0.0367
Gemini 3.1 Flash-Lite Restricted	0.0449	0.0307

Table 2: CoT SFT variants on Amazon Beauty. Explicit rationales fail to outperform simple next-item SFT across rationale variants, indicating that rationale supervision alone does not reliably improve GR.

40.5%, indicating that CPT captures broad categorical structure. These results show that LLMs associate SIDs with semantics from pretraining, but only at a coarse level. We next test whether CoT can convert this signal into better SID prediction.

3.2 CoT SFT: The Failure of Explicit CoT

Here, we investigate if CoT SFT improves GR.

Experimental Design. We perform CoT SFT on Qwen3-1.7B (Team, 2025) after CPT and SFT on Amazon Beauty (Ni et al., 2019), using template-based, teacher-generated, rejection-sampled, and format-restricted rationales. For template-based rationales, we use: (1) Template-Category: "The user is likely to buy items in the {target item category} category." and (2) Template-Extended: "The user demonstrates interest in {frequent categories} products. By identifying the user's preference in {characteristics}, we can predict the user's purchase of {target categories}, for example, a {item title}." For teacher-generated rationales, we use Gemini 3.1 Flash-Lite and Pro (Team et al., 2023) to produce free-form traces. For rejection sampling, Gemini 3.1 Flash-Lite generates multiple traces per sample, and we select either the trace with the highest target-SID logits (Gemini 3.1 Flash-Lite Rejection) or the trace Gemini 3.1 Pro judges to best connect the user history to the target item (Gemini 3.1 FL Gemini Rejection). Finally, format-restricted traces impose reasoning constraints, e.g., rationales must reference SIDs. Sample rationales and prompts are in Appendix F.

Results and Analysis. Table 2 shows that explicit CoT variants underperform next-item SFT. The strongest variant, Gemini rejection sampling, remains below the baseline (0.0524 vs. 0.0533 Hit@5), while weaker teacher-generated variants lose over 20% relative Hit@5, so CoT SFT alone does not reliably improve SID prediction. This pattern contrasts with CoT's success on language tasks: prior LLM-based GR work that reports gains from in-text reasoning relies on expensive RL with verifiable rewards (RLVR) after CoT SFT (Liu et al., 2025; Yu et al., 2025). While RLVR can recover performance, it requires multiple rollout trajectories per step and is substantially more expensive than next-item SFT, which raises the question: *why does CoT SFT fail for SID-based GR?*

4 Diagnosis of CoT SFT Limitations

To understand why explicit CoT SFT fails, we conduct diagnostic studies and identify three limitations of the CoT SFT stage.

4.1 Difficulty Verbalizing World Knowledge

Finding. CoT SFT does not erase LLMs' world knowledge, but makes them difficult to verbalize.

Experimental Design. We evaluate Qwen3-1.7B (Team, 2025) after CoT SFT on representative language tasks benchmarks MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and ARC-Challenge (Clark et al., 2018) in multiple-choice format. We report *text-match* accuracy (exact A/B/C/D generation) and *logit-based* accuracy (whether the correct choice has the highest logit).

Results and Analysis. Table 3 shows text-match accuracy degrades after CoT SFT, while logit-based accuracy remains close to the base model on all benchmarks. It shows that the LLM's world knowledge remains primarily in logit space and is hard to verbalize in explicit natural language text format. We next examine whether this text–SID interface mismatch is also in the token embedding space.

Dataset	Base	Text Match	Logit-based
MMLU	57.60	0.10	55.20
HellaSwag	61.00	9.00	60.40
PIQA	69.90	5.60	69.90
ARC-C	76.40	0.80	73.80

Table 3: General-language reasoning accuracy after recommendation CoT SFT. Text-match accuracy collapses while logit-based accuracy remains close to the base model, suggesting that answer information is still present in logits but is no longer reliably verbalized.

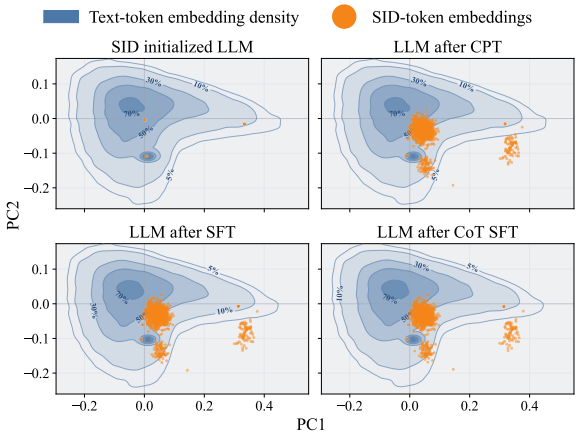

Figure 2: PCA of text and SID token embeddings across training stages. SID tokens drift away from ordinary text tokens as training progresses, indicating limited embedding space overlap between text and SIDs.

4.2 Text–SID Embedding Misalignment

Finding. SID and natural-language tokens become geometrically separated in the token embedding space, causing difficulty for LLM to unify text and SIDs under a coherent rationale (see results and analysis for specific reasons).

Experimental Design. We visualize token embeddings after SID initialization, CPT, SFT, and CoT SFT using PCA (Jolliffe, 2025), comparing ordinary text tokens with SID tokens.

Results and Analysis. Figure 2 shows text and SID embeddings diverging across stages, and the gap is already pronounced after CPT and continue to slightly expand during SFT and CoT SFT. This token embedding discrepancy suggests difficulty in unifying language and SIDs in one coherent representation. Specifically, Appendix D proves that when text- and SID-induced hidden-state directions are weakly coupled, updates driven by natural-language rationales can only weakly shift the logits over SID tokens, so explicit CoT has limited leverage on the final recommendation.

Reasoning Variant	Hit@5	NDCG@5
Ground-truth	0.1165	0.0836
Remove category	0.0540	0.0376
Drop 5 words	0.0950	0.0834
Add 5 words	0.1145	0.0682

Table 4: Rationale perturbation sensitivity on Amazon Beauty (Ni et al., 2019). Removing the target category more than halves performance, and even small word-level perturbations affect accuracy, showing that explicit CoT relies on brittle rationale cues.

4.3 Performance Fragility w.r.t. Rationales

Finding. After CoT SFT, recommendation performance is highly sensitive to the rationale text at inference, even when generated reasoning only slightly deviates from the ground-truth rationale.

Experimental Design. We test CoT SFT models with ground-truth rationales and controlled perturbations—removing the target item category, randomly dropping five words, or randomly adding five noise words—and measure Hit@5 and NDCG@5 under each setting.

Results and Analysis. Table 4 shows that performance is highly sensitive to rationale content. Removing the target category more than halves Hit@5 (0.1165 to 0.0540) and NDCG@5 (0.0836 to 0.0376). Surface perturbations also matter: dropping five words reduces Hit@5 by 18.5%, while adding five noise words reduces NDCG@5 by 18.4%. Explicit CoT thus depends on brittle rationale cues, especially whether the text preserves semantics needed for the target SID.

4.4 Summary of Findings

Our diagnostics expose three CoT failures: **weakened verbalization** leaves answer signals in logits but weakens decoding; **text–SID embedding misalignment** limits rationale effects on SID logits; and **fragile rationales** make metrics sensitive to small edits. This motivates *implicit reasoning in latent space*: learned <pause> tokens bridge language to SIDs without decoding brittle intermediate natural language reasoning text.

5 Methodology: PauseRec

Motivated by Section 4, we propose PauseRec, an implicit reasoning method for LLM-based GR that keeps the CPT and next-item SFT stages of explicit pipelines but replaces CoT SFT and RL with pause-based latent computation.

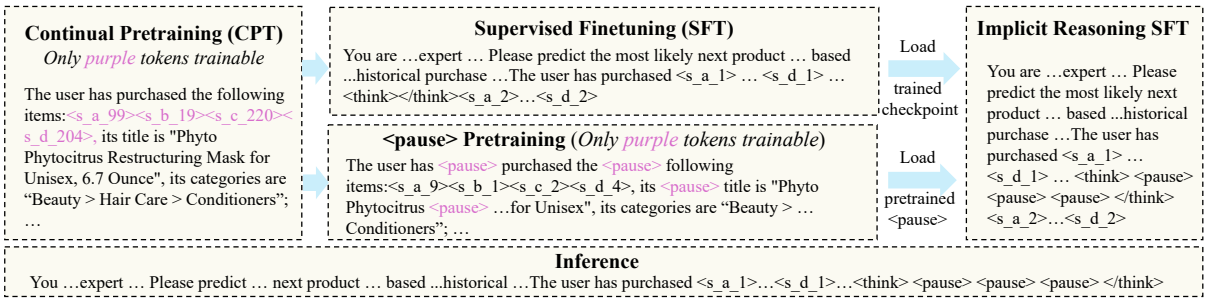

Figure 3: Overview of PAUSEREC. Instead of generating explicit rationales and applying RL post-training, PAUSEREC pretrains a <pause> token to bridge text and SID representations, then inserts pause tokens before SID generation and trains them only through the final next-item prediction loss.

5.1 Overview of PAUSEREC

As illustrated in Fig. 3, we perform CPT (Section 2.2), then conduct next-item SFT and <pause> pretraining in parallel on the same CPT checkpoint. The SFT branch follows Section 2.2; the pause branch finetunes on the CPT corpus with <pause> tokens injected at random text positions so the token learns semantic transitions between language and SID tokens. We then load the pretrained <pause> embedding into the SFT checkpoint and run implicit reasoning SFT on next-item data with k pauses inserted between user history and target SID, optimizing only SID positions.

PAUSEREC addresses the three CoT failures above: (1) **Computation without verbalization** via latent <pause> steps that need not be decoded as natural language; (2) **Bridging embedding spaces** via CPT-grounded pause pretraining (Appendix E visualizes the trained <pause> token positioned between embedding spaces); and (3) **Avoiding rationale supervision** by masking loss on pause positions and optimizing only target SID.

5.2 <pause> Token Initialization

We add <pause> to the vocabulary and initialize its embedding at the mean of all token embeddings after CPT, with variance set to 10^{-9} times the embedding variance (equivalent to a near-deterministic start at the vocabulary center): $\mathbf{e}^{(0)}_{\texttt{<pause>}} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \mathbf{e}_v$, where $\mathcal{V}$ is the full vocabulary. This center initialization gives <pause> a neutral starting point between text and SIDs.

5.3 Two-Stage Training

Stage 1: <pause> Token Pretraining. Starting from the CPT checkpoint, we finetune on the CPT corpus with <pause> inserted at random positions covering 10% of each sequence (Fig. 3). Only $\mathbf{e}_{\texttt{<pause>}}$ is trainable; all other parameters remain frozen. This concentrates updates on the bridge token while preserving grounded SID embeddings and the pretrained language backbone.

Stage 2: Implicit Reasoning SFT. We load the pretrained $\mathbf{e}_{\texttt{<pause>}}$ into the SFT checkpoint and append k pauses between user history and target:

$$x' = \text{Prompt}(H) \| \underbrace{\texttt{<pause>}, \ldots, \texttt{<pause>}}_{k \text{ times}} \quad (5)$$

The LLM is finetuned with loss masked at <pause> positions; only target SID tokens are optimized:

$$\mathcal{L}_{\text{implicit}} = -\sum_{l=1}^{L} \log p_\theta \left(s_{n+1}^{(l)} \mid x', s_{n+1}^{(1:l-1)} \right) \quad (6)$$

By not imposing loss on pause positions, we avoid imitating a fixed teacher rationale distribution and instead let the model use pauses only when they improve SID prediction. In practice, pause slots act as task-specific latent scratch space between the textual history and discrete SID outputs. See Appendix F.1 for sample prompts and formatted training text for each stage of PAUSEREC.

5.4 Inference

At test time, we use the same prompt template as implicit-reasoning SFT (Appendix F.1), insert k literal <pause> tokens between the <think> and </think> tags before the SID output, and autoregressively decode the next SID. No rationale text is generated at inference, which removes the token overhead of explicit CoT while preserving a dedicated computation window before SID prediction.

6 Experiments

6.1 Experimental Setup

Datasets. We evaluate on three Amazon review datasets (Ni et al., 2019): Beauty, Sports and Outdoors, and Toys and Games. Following (Liu et al.,

Method	Beauty				Sports				Toys			
	H@5	H@10	N@5	N@10	H@5	H@10	N@5	N@10	H@5	H@10	N@5	N@10
GRU4Rec	0.0395	0.0584	0.0265	0.0326	0.0190	0.0312	0.0122	0.0161	0.0330	0.0490	0.0228	0.0279
SASRec	0.0402	0.0607	0.0254	0.0320	0.0199	0.0301	0.0106	0.0141	0.0448	0.0626	0.0300	0.0358
BERT4Rec	0.0232	0.0396	0.0146	0.0199	0.0102	0.0175	0.0065	0.0088	0.0215	0.0332	0.0131	0.0168
HGN	0.0319	0.0536	0.0196	0.0266	0.0183	0.0313	0.0109	0.0150	0.0326	0.0517	0.0192	0.0254
HSTU	0.0424	0.0652	0.0280	0.0353	0.0268	0.0343	0.0173	0.0226	0.0366	0.0566	0.0245	0.0309
TIGER	0.0405	0.0623	0.0267	0.0337	0.0215	0.0347	0.0137	0.0179	0.0337	0.0547	0.0209	0.0276
ReaRec	0.0450	0.0704	0.0262	0.0344	0.0214	0.0332	0.0116	0.0154	0.0523	0.0764	0.0298	0.0376
Next-item SFT	0.0533	0.0733	0.0381	0.0445	0.0287	0.0409	0.0198	0.0237	0.0565	0.0781	0.0402	0.0471
OneRec-Think	0.0563	**0.0791**	0.0398	**0.0471**	0.0288	0.0412	0.0199	0.0239	0.0579	0.0797	0.0412	0.0482
PAUSEREC	**0.0568**	0.0746	**0.0401**	0.0467	**0.0294**	**0.0422**	**0.0203**	**0.0245**	**0.0615**	**0.0838**	**0.0434**	**0.0509**

Table 5: Main recommendation results on three Amazon datasets. PAUSEREC consistently improves over next-item SFT and exceeds the baselines on most metrics; boldface and underlining mark the best and second-best results.

	Train (GPU hours)	Inference (s)
PauseRec	107.34	0.0586 ± 0.0093
OneRec-Think	305.62	0.2043 ± 0.0255

Table 6: Efficiency comparison on Qwen3-1.7B with Amazon Beauty. By avoiding RL post-training and natural-language rationale generation, PAUSEREC uses about 65% fewer training GPU hours and is roughly 3.5× faster per inference sample than OneRec-Think.

2025), we filter users and items with fewer than five interactions and use a leave-last-out split: the final item is held out for testing, the second-to-last for validation, and the third-to-last as the training target with all earlier interactions as input.

Baselines. We compare (1) *traditional sequential recommenders*: GRU4Rec (Hidasi et al., 2016), SASRec (Kang and McAuley, 2018), BERT4Rec (Sun et al., 2019), and HGN (Ma et al., 2019); (2) *generative retrieval models*: HSTU (Zhai et al., 2024) and TIGER (Rajput et al., 2023); (3) *LLM-based models*: next-item SFT (our reproduction) and OneRec-Think (Liu et al., 2025) (explicit CoT with RLVR); and (4) *implicit reasoning*: ReaRec (Lin et al., 2024).

Metrics and Implementation. We report Hit@5, Hit@10, NDCG@5, and NDCG@10 (Rajput et al., 2023). The backbone is Qwen3-1.7B (Team, 2025). CPT runs for 3 epochs (lr 10^{-4}), pause pretraining for 2 epochs (lr 10^{-3}), and implicit SFT for 5 epochs (lr 5×10^{-5}) with AdamW (wd 0.01). Main results use k=5 pauses. Training and evaluation prompts match the templates in Appendix F.1.

6.2 Effectiveness & Efficiency

We evaluate the effectiveness and efficiency of PAUSEREC. From Table 5, we observe that (1) **Consistent gains over next-item SFT:** PAUSEREC improves every metric over the next-item SFT baseline, with relative gains up to 8.85% on Toys Hit@5. (2) **Competitive with or better than RL-based CoT:** PAUSEREC outperforms OneRec-Think on 10 of 12 metrics, including all Sports and Toys metrics, with up to 6.22% relative improvement on Toys Hit@5; OneRec-Think remains higher on Beauty Hit@10 and NDCG@10. (3) **Substantial gains over non-LLM recommenders:** PAUSEREC consistently outperforms all baselines, highlighting the value of LLM knowledge through a SID-compatible interface.

From Table 6, PAUSEREC reduces training GPU hours by 65% and inference latency by roughly 3.5× on Beauty by avoiding RL post-training and rationale generation. OneRec-Think uses relatively short template rationales here; inference savings grow with generated tokens (Appendix G.1).

6.3 Ablation Studies

We compare our CPT-grounded pause pretraining with alternative initializations before implicit SFT: the mean of text embeddings only, the mean of SID embeddings only, and the default special-token initialization. Table 7 shows that the pretrained <pause> token performs best, with modest but consistent gains over text-only, SID-only, and default initializations. This supports pause pretraining for bridging text and SID embedding spaces.

6.4 Parameter Analysis

We analyze the effect of pause count k. All settings share the same pause pretraining; we append k pauses during implicit SFT and use the same k at inference. Figure 4 shows moderate values of k

Initialization	Hit@5	NDCG@5
Center of text tokens only	0.0559	0.0395
Center of SID tokens only	0.0548	0.0387
Default Initialization	0.0560	0.0394
Pretrained (Ours)	**0.0568**	**0.0401**

Table 7: <pause> initialization ablation on Beauty. Pretraining the pause token to bridge text and SID contexts gives the best Hit@5 and NDCG@5, outperforming text-only, SID-only, and default initializations.

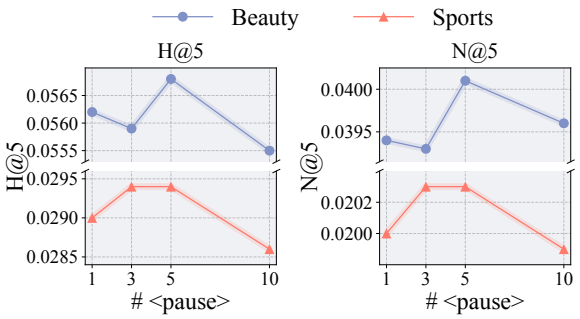

Figure 4: Effect of the number of <pause> tokens. Moderate latent computation works best (k=5), while further increasing k provides no performance improvement.

work best (k=5); increasing to k=10 does not consistently help (Table 10), suggesting useful latent computation saturates after some pause steps.

6.5 Qualitative Analysis

To understand how PAUSEREC uses latent pause computation during inference, we analyze where each <pause> token in the reasoning block attends in the surrounding context. For each pause token, we average its outgoing attention to context tokens across all layers and heads, and visualize how that distribution changes relative to the preceding pause token (Fig. 5). From Fig. 5 (see full prompt in Appendix F.2), we observe a multi-stage process. (1) **Context orientation:** early pauses attend broadly to the instruction and history boundary, establishing that the next SID should be inferred from purchase history. (2) **Preference aggregation:** middle and later pauses shift toward historical SIDs, identifying purchases relevant to user intent. With pause position proceeds, the LLM focuses on a small salient subset of SIDs, locating items similar to target item. This staged transition explains why latent pause computation improves GR.

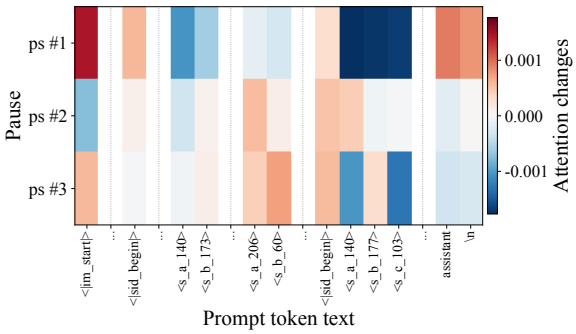

Figure 5: Attention changes across <pause> positions for a representative recommendation. Early pause tokens attend broadly to the prompt and history boundary, while later pause tokens focus on a smaller set of historical SID tokens; red and blue indicate increased and decreased attention after each pause.

7 Related Work

LLM-based GR. Recent work uses LLMs as rankers or feature extractors (Hou et al., 2023) and as generative recommenders that output SIDs (Rajput et al., 2024; Hua et al., 2023). These pipelines typically use CPT on item-text corpora to ground SIDs (Bao et al., 2023), then apply next-item SFT. Our work builds on this foundation and asks when pretrained world knowledge improves SID prediction beyond standard training.

Reasoning in LLMs. Chain-of-Thought prompting improves reasoning on language tasks such as math (Wei et al., 2022) and science (Lewkowycz et al., 2022), but SID-based GR involves non-linguistic outputs. Recent GR systems add CoT SFT and RL on top of CPT (Liu et al., 2025; Yu et al., 2025; Liang et al., 2026); our stage-wise analysis clarifies when those additions help. Implicit reasoning via latent tokens appears in quiet CoT (Zelikman et al., 2024) and ReaRec (Lin et al., 2024); to our knowledge, we provide the first systematic explicit-vs-implicit comparison for LLM-based GR with an analysis of CoT SFT failure.

8 Conclusion

This paper shows that explicit rationales are a poor interface for SID-based generative recommendation: LLMs retain useful signals, but weakened verbalization, text–SID embedding mismatch, and rationale sensitivity limit CoT SFT. PAUSEREC replaces rationales with trainable <pause> tokens, enabling latent reasoning that bridges language and SIDs. Extensive experiments show that PAUSEREC is effective and efficient.

References

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. TALLRec: An effective and efficient tuning framework to align large language model with recommendation. *ArXiv*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. PIQA: Reasoning about physical commonsense in natural language. *AAAI*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *ArXiv*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *ArXiv*.

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

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. In *ICML*.

Wenyue Hua, Yingqiang Xu, Yunchao Ge, Yao Zhang, Shuyuan Xu, Jianchao Tan, and Yongfeng Dong. 2023. UP5: Unbiased foundation model for fairness-aware recommendation. *ArXiv*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of ACL*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. In *ACL*.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2026. A survey on large language models for code generation. *ACM Trans. Softw. Eng. Methodol.*

Ian Jolliffe. 2025. Principal component analysis. In *Int. Encycl. Stat. Sci.* Springer.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *NeurIPS*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. In *NeurIPS*.

Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized transformer for explainable recommendation. In *ACL-IJCNLP*.

Mingfu Liang, Yufei Li, Jay Xu, Kavosh Asadi, Xi Liu, Shuo Gu, Kaushik Rangadurai, Frank Shyu, Shuaiwen Wang, Song Yang, and 1 others. 2026. Generative reasoning re-ranker. *ArXiv*.

Xinfeng Lin, Fuqiang Zhang, Meng Wang, Weiwen Zhou, Xin Huang, Kai He, Shengsheng Chen, and Lizhen Liu. 2024. ReaRec: Reasoning for sequential recommendation. In *WWW*.

Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, and 1 others. 2025. Onerec-think: In-text reasoning for generative recommendation. *ArXiv*.

Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *KDD*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, and 1 others. 2023. Recommender systems with generative retrieval. *NeurIPS*.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, and 1 others. 2024. Recommender systems with generative retrieval. In *NeurIPS*.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *ArXiv*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Daniel Truhn, Jorge S Reis-Filho, and Jakob Nikolas Kather. 2023. Large language models should be used as scientific reasoning engines, not knowledge databases. *Nat. Med.*

Xinfeng Wang, Jin Cui, Fumiyo Fukumoto, and Yoshimi Suzuki. 2025. AGRec: Adapting autoregressive decoders with graph reasoning for LLM-based sequential recommendation. In *Findings of ACL*.

wangshy31. 2025. OneRec-Think. https://github.com/wangshy31/OneRec-Think. GitHub repository.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*.

Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, and 1 others. 2024. Kola: Carefully benchmarking world knowledge of large language models. In *ICLR*.

Qinyong Yu, Ke Fu, Shuai Zhang, Zihan Lv, Fan Wu, and Feng Wu. 2025. ThinkRec: Thinking-based recommendation via LLM. *ArXiv*.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-STAR: Language models can teach themselves to think before speaking. *ArXiv*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *ACL*.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, and 1 others. 2024. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *ICML*.

Haochen Zhang, Tianyi Zhang, Junze Yin, Oren Gal, Anshumali Shrivastava, and Vladimir Braverman. 2025. CoVE: Compressed vocabulary expansion makes better LLM-based recommender systems. In *Findings of ACL*.

Luankang Zhang, Yonghao Huang, Hang Lv, Mingjia Yin, Liangyue Li, Zulong Chen, Hao Wang, and Enhong Chen. 2026. Why thinking hurts? diagnosing and rectifying the reasoning shift in foundation recommender models. *ArXiv*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *ArXiv*.

A Limitations and Potential Risks

PAUSEREC leaves several natural extensions for future work. First, our study uses a compact pause-token design and reports sensitivity to pause length, but does not exhaustively tune all possible pause placements, initialization schedules, or decoding variants. Second, our evaluation follows the standard offline next-item prediction protocol; complementary user-facing studies could further examine how latent reasoning affects perceived usefulness, diversity, and recommendation presentation. Finally, because implicit <pause> tokens are not natural-language rationales, their intermediate computation is less directly readable by users, motivating additional probing and visualization tools for analyzing how pause tokens support SID prediction.

Like other recommender systems, PAUSEREC may amplify popularity bias, reinforce historical user preferences too strongly, or inherit biases present in the interaction data and pretrained LLM. Deployments should therefore include privacy-preserving data handling, bias and fairness auditing, and safeguards against over-personalization or harmful recommendations.

B AI Usage

AI writing assistance was used only to polish grammar, clarity, and sentence flow. All research ideas, experimental designs, analyses, results, and final claims were developed, checked, and approved by the authors.

C Artifacts

Our artifact release includes the code and scripts for constructing SID-based training data, running the four PAUSEREC training stages, evaluating constrained SID decoding, and generating the tables and figures reported in the paper. The implementation is based on the open-source OneRec-Think repository (wangshy31, 2025); we extend it with pause-token pretraining, implicit-reasoning finetuning, evaluation utilities, and experiment orchestration for PAUSEREC. OneRec-Think is released under the Apache License 2.0, which permits reuse and modification. Our code is released under the MIT License.

D Theoretical Analysis of Text–SID Separation

We formalize why geometric separation between natural-language and SID representations can weaken explicit CoT. Consider the final step after a rationale, where the model must generate a SID token. Let v_s denote the output embedding of SID token s, and let the SID logit be

$$z_s(h) = v_s^\top h,$$

where h is the hidden state before SID generation. Let $\mathcal{U}_{\text{text}}$ be the subspace in which hidden states move when the model generates or is optimized on natural-language rationale tokens, and let

$$\mathcal{U}_{\text{SID}} = \text{span}\{v_y - v_s : y, s \in \mathcal{S}\}$$

be the subspace that controls relative SID logits. Define the text–SID coupling coefficient

$$\rho = \|P_{\mathcal{U}_{\text{SID}}} P_{\mathcal{U}_{\text{text}}}\|_2,$$

where $P_{\mathcal{U}}$ is the orthogonal projection onto subspace $\mathcal{U}$. Smaller ρ means stronger separation between text-induced hidden-state movement and SID-discriminative directions.

Theorem 1 (Text–SID separation bounds the effect of verbal rationales)**.** *Suppose adding a natural-language rationale changes the hidden state before SID generation from h to $h + \Delta$, where*

$$\Delta = \Delta_{text} + r, \quad \Delta_{text} \in \mathcal{U}_{text}, \quad \|r\| \le \epsilon.$$

Assume $\|v_y - v_s\| \le B$ for all valid SID tokens $y, s \in \mathcal{S}$. Let

$$\rho = \|P_{\mathcal{U}_{\text{SID}}} P_{\mathcal{U}_{text}}\|_2.$$

Then for any target SID token y and competing SID token s,

$$\big|\big(z_y(h+\Delta) - z_s(h+\Delta)\big) - \big(z_y(h) - z_s(h)\big)\big| \le B(\rho\|\Delta_{text}\| + \epsilon). \tag{7}$$

Consequently, if the target SID initially trails some competitor by margin γ,

$$z_y(h) - z_s(h) \le -\gamma,$$

and

$$\gamma > B(\rho\|\Delta_{text}\| + \epsilon),$$

then the rationale cannot make y outrank s.

Proof. Let the SID margin between target token y and competing token s be

$$m(h) = z_y(h) - z_s(h).$$

The change in this margin after adding the rationale is

$$m(h+\Delta) - m(h) = \big(z_y(h+\Delta) - z_s(h+\Delta)\big) - \big(z_y(h) - z_s(h)\big). \tag{8}$$

Since $z_s(h) = v_s^\top h$, we have

$$m(h+\Delta) - m(h) = (v_y - v_s)^\top \Delta.$$

Substituting $\Delta = \Delta_{\text{text}} + r$ gives

$$|(v_y - v_s)^\top \Delta| \le |(v_y - v_s)^\top \Delta_{\text{text}}| + |(v_y - v_s)^\top r|.$$

Because only the projection of Δ_{text} onto the SID-discriminative subspace can affect relative SID logits,

$$|(v_y - v_s)^\top \Delta_{\text{text}}| \le \|v_y - v_s\| \cdot \|P_{\mathcal{U}_{\text{SID}}} \Delta_{\text{text}}\|.$$

Since $\Delta_{\text{text}} \in \mathcal{U}_{\text{text}}$,

$$P_{\mathcal{U}_{\text{SID}}} \Delta_{\text{text}} = P_{\mathcal{U}_{\text{SID}}} P_{\mathcal{U}_{\text{text}}} \Delta_{\text{text}}.$$

By the definition of ρ,

$$\|P_{\mathcal{U}_{\text{SID}}} P_{\mathcal{U}_{\text{text}}} \Delta_{\text{text}}\| \le \rho \|\Delta_{\text{text}}\|.$$

Therefore,

$$|(v_y - v_s)^\top \Delta_{\text{text}}| \le B\rho \|\Delta_{\text{text}}\|.$$

For the residual term,

$$|(v_y - v_s)^\top r| \le \|v_y - v_s\| \|r\| \le B\epsilon.$$

Combining the two bounds yields

$$|m(h+\Delta) - m(h)| \le B(\rho \|\Delta_{\text{text}}\| + \epsilon).$$

It remains to show the ranking consequence. Define

$$M = B(\rho \|\Delta_{\text{text}}\| + \epsilon).$$

The bound above implies that the rationale can change the SID margin by at most M. If the target SID initially trails competitor s by margin γ, then

$$m(h) = z_y(h) - z_s(h) \le -\gamma.$$

After adding the rationale,

$$m(h+\Delta) = m(h) + \big(m(h+\Delta) - m(h)\big) \le m(h) + M.$$

Since $m(h) \le -\gamma$, we obtain

$$m(h+\Delta) \le -\gamma + M.$$

If $\gamma > M$, then

$$m(h+\Delta) < 0.$$

Equivalently,

$$z_y(h+\Delta) < z_s(h+\Delta).$$

Thus, even after adding the rationale, the target SID token y still receives a lower logit than the competing SID token s, so y cannot outrank s. □

E PAUSEREC Embedding Visualization

Figure 6 visualizes token embeddings after each stage of the PAUSEREC pipeline, including CPT, pause-token pretraining, next-item SFT, and implicit-reasoning SFT. Across stages, the <pause> token stays at the boundary between the natural-language token cluster and the SID token cluster rather than collapsing into either group. This boundary placement provides empirical evidence that the pause token connects semantics across the two embedding spaces, helping route natural-language knowledge toward SID generation and thereby improving recommendation performance.

F Implementation Details

F.1 Sample Prompts for PAUSEREC Training and Inference

We provide concrete prompt text for each stage of PAUSEREC on Amazon Beauty, using the same leave-last-out training split as in Section 6. The example user has two items in the prediction history; the held-out target item is *Raw African Black Soap from Ghana 1 Lb* (semantic ID shown in the implicit-reasoning blocks below). Chat-based stages use the fixed system instruction and chat turn delimiters shown in the full-sequence examples. During pause pretraining, <pause> tokens are inserted at random word boundaries. During implicit-reasoning SFT and inference, k=5 pause tokens are placed between the <think> and </think> tags before the target SID; loss is masked on those pause positions during SFT.

Continual pretraining (CPT).

> The user has purchased the following items: <|sid_begin|><s_a_99><s_b_19><s_c_220><s_d_204><|sid_end|>, its title is "Phyto Phytocitrus Restructuring Mask for Unisex,

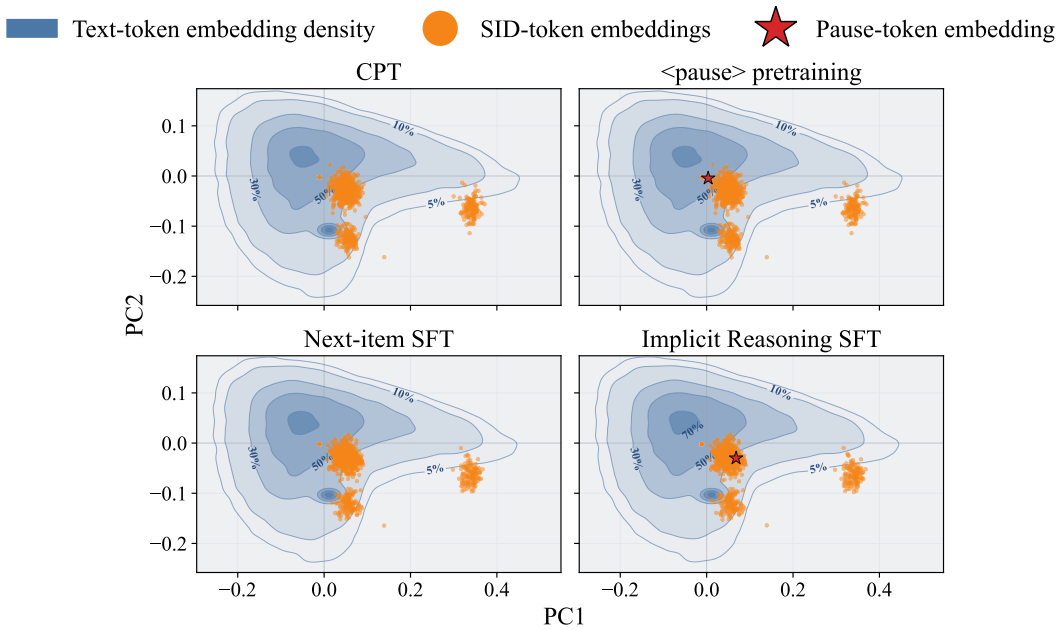

Figure 6: Token embeddings after each stage of the PAUSEREC pipeline. The learned <pause> token lies near the boundary between natural-language tokens and SID tokens, indicating that pause pretraining positions it as a bridge between the two embedding spaces. This supports the role of <pause> tokens in connecting semantic information from natural language to SID prediction.

6.7 Ounce", its categories are "Beauty > Hair Care > Conditioners"; <|sid_begin|><s_a_238><s_b_74><s_c_13><s_d_122><|sid_end|>, its title is "Matrix Biolage Colorcaretherapie Color Care Shampoo and Conditioner Set 33.8oz 1 Liter", its categories are "Beauty > Hair Care > Shampoo & Conditioner Sets"; <|sid_begin|><s_a_226><s_b_110><s_c_129><s_d_207><|sid_end|>, its title is "Raw African Black Soap from Ghana 1 Lb", its categories are "Beauty > Bath & Body > Cleansers > Soaps";

Pause-token pretraining (10% random <pause> insertion; seed 42).

The user has <pause> purchased the following items: <|sid_begin|><s_a_99><s_b_19><s_c_220><s_d_204><|sid_end|>, its title is "Phyto <pause> <pause> Phytocitrus Restructuring <pause> Mask for Unisex, 6.7 Ounce", its categories are "Beauty > <pause> Hair Care <pause> > Conditioners"; <|sid_begin|><s_a_238><s_b_74><s_c_13><s_d_122><|sid_end|>, <pause> its title is "Matrix Biolage Colorcaretherapie Color Care Shampoo and Conditioner Set 33.8oz 1 Liter", its categories are "Beauty > Hair Care > Shampoo & Conditioner Sets"; <|sid_begin|><s_a_226><s_b_110><s_c_129><s_d_207><|sid_end|>, its title is "Raw African Black Soap from Ghana 1 Lb", its categories are "Beauty > Bath <pause> & Body > Cleansers > Soaps";

Next-item supervised finetuning: user prompt.

The user has purchased the following items: <|sid_begin|><s_a_99><s_b_19><s_c_220><s_d_204><|sid_end|>; <|sid_begin|><s_a_238><s_b_74><s_c_13><s_d_122><|sid_end|>;

Next-item supervised finetuning: full sequence (empty thinking block).

<|im_start|>system You are a professional recommendation expert who needs to recommend the next possible purchase for users based on their purchase history. Please predict the most likely next product that the user will purchase based on the user's historical purchase information.<|im_end|> <|im_start|>user The user has purchased the following items: <|sid_begin|><s_a_99><s_b_19><s_c_220><s_d_204><|sid_end|>; <|sid_begin|><s_a_238><s_b_74><s_c_13><s_d_122><|sid_end|>;<|im_end|> <|im_start|>assistant <think></think> <|sid_begin|><s_a_226><s_b_110><s_c_129><s_d_207><|sid_end|><|im_end|>

Implicit-reasoning finetuning: user prompt and target SID.

The user has purchased the following items: <|sid_begin|><s_a_99><s_b_19><s_c_220><s_d_204><|sid_end|>, its title is "Phyto Phytocitrus Restructuring Mask for Unisex, 6.7 Ounce", its categories are "Beauty > Hair Care > Conditioners"; <|sid_begin|><s_a_238><s_b_74><s_c_13><s_d_122><|sid_end|>, its title is "Matrix Biolage Colorcaretherapie Color Care Shampoo and Conditioner Set 33.8oz 1 Liter", its categories are "Beauty > Hair Care > Shampoo & Conditioner Sets"; Target SID: <|sid_begin|><s_a_226><s_b_110><s_c_129><s_d_207><|sid_end|>

Implicit-reasoning finetuning: full sequence (k=5 pause tokens).

<|im_start|>system You are a professional recommendation expert who needs to recommend the next possible purchase for users based on their purchase

history. Please predict the most likely next product that the user will purchase based on the user's historical purchase information.<|lim_end|> <|lim_start|>user The user has purchased the following items: <|sid_begin|><s_a_99><s_b_19><s_c_220><s_d_204><|sid_end|>, its title is "Phyto Phytocitrus Restructuring Mask for Unisex, 6.7 Ounce", its categories are "Beauty > Hair Care > Conditioners"; <|sid_begin|><s_a_238><s_b_74><s_c_13><s_d_122><|sid_end|>, its title is "Matrix Biolage Colorcaretherapie Color Care Shampoo and Conditioner Set 33.8oz 1 Liter", its categories are "Beauty > Hair Care > Shampoo & Conditioner Sets";<|lim_end|> <|lim_start|>assistant <think> <pause><pause><pause><pause><pause> </think> <|sid_begin|><s_a_226><s_b_110><s_c_129><s_d_207><|sid_end|><|lim_end|>

Inference: prompt prefix before constrained SID decoding.

<|lim_start|>system You are a professional recommendation expert who needs to recommend the next possible purchase for users based on their purchase history. Please predict the most likely next product that the user will purchase based on the user's historical purchase information.<|lim_end|> <|lim_start|>user The user has purchased the following items: <|sid_begin|><s_a_99><s_b_19><s_c_220><s_d_204><|sid_end|>, its title is "Phyto Phytocitrus Restructuring Mask for Unisex, 6.7 Ounce", its categories are "Beauty > Hair Care > Conditioners"; <|sid_begin|><s_a_238><s_b_74><s_c_13><s_d_122><|sid_end|>, its title is "Matrix Biolage Colorcaretherapie Color Care Shampoo and Conditioner Set 33.8oz 1 Liter", its categories are "Beauty > Hair Care > Shampoo & Conditioner Sets";<|lim_end|> <|lim_start|>assistant <think> <pause><pause><pause><pause><pause> </think>

F.2 Prompt for Qualitative Attention Analysis

Figure 5 visualizes the pause-token attention pattern for the following example. The target item is a Dove hair styling spray, and the history contains multiple hair-care and styling products. In the later pause steps, attention to the history item whose SID begins with <s_a_206><s_b_60>, *Natures Bounty Optimal Solutions Hair, Skin and Nails Gummies*, increases. This item is related to the target through hair-care intent, so the increase supports the main-paper analysis that later pauses retrieve and aggregate target-relevant historical evidence before SID generation.

Target Item Title: Dove Hair Styling Oxygen Moisture Root Lift Spray, 3.3 Ounce SID: <|sid_begin|><s_a_140><s_b_39><s_c_151><s_d_68><|sid_end|> Categories: Beauty > Hair Care > Styling Products > Hair Sprays

Full prompt text.

<|lim_start|>system You are a professional recommendation expert who needs to recommend the next possible purchase for users based on their purchase history. Please predict the most likely next product that the user will purchase based on the user's historical purchase information.<|lim_end|> <|lim_start|>user The user has purchased the following items: <|sid_begin|><s_a_6><s_b_192><s_c_205><s_d_33><|sid_end|>, its title is "Axe Primed Just Clean Shampoo, 12-Ounce Bottle (Pack of 3)", its categories are "Beauty > Hair Care > Shampoos"; <|sid_begin|><s_a_248><s_b_8><s_c_99><s_d_150><|sid_end|>, its title is "Olay Regenerist Micro-Sculpting Serum 1.7 Fl Oz", its categories are "Beauty > Skin Care > Face"; <|sid_begin|><s_a_113><s_b_56><s_c_77><s_d_2><|sid_end|>, its title is "Clearasil Ultra Acne Treatment Daily Face Wash, 6.78 Ounce (Pack of 3)", its categories are "Beauty > Skin Care > Face > Cleansers > Washes"; <|sid_begin|><s_a_6><s_b_6><s_c_17><s_d_210><|sid_end|>, its title is "Pantene Pro-V Expert Collection Agedefy Conditioner 8.4 Fl Oz", its categories are "Beauty > Hair Care > Conditioners"; <|sid_begin|><s_a_6><s_b_222><s_c_222><s_d_71><|sid_end|>, its title is "Pantene Pro-V Expert Collection Agedefy Shampoo 10.1 Fl Oz", its categories are "Beauty > Hair Care > Shampoos"; <|sid_begin|><s_a_255><s_b_71><s_c_242><s_d_99><|sid_end|>, its title is "Burt's Bees Lip Gloss, Autumn Haze, 0.2 Fluid Ounces", its categories are "Beauty > Makeup > Lips > Lip Glosses"; <|sid_begin|><s_a_155><s_b_51><s_c_96><s_d_246><|sid_end|>, its title is "Nexxus Youth Renewal Rejuvenating Shampoo, 13.5 Ounce", its categories are "Beauty > Hair Care > Shampoos"; <|sid_begin|><s_a_248><s_b_86><s_c_54><s_d_216><|sid_end|>, its title is "Simple Protecting Light Moisturizer Spf 15, 4.2 Ounce", its categories are "Beauty > Skin Care > Face > Creams & Moisturizers > Fluids & Lotions > Lotions"; <|sid_begin|><s_a_113><s_b_255><s_c_31><s_d_115><|sid_end|>, its title is "Dove go fresh, Burst Body Wash, 24 Ounce (Pack of 2)", its categories are "Beauty > Bath & Body > Cleansers > Body Washes"; <|sid_begin|><s_a_140><s_b_254><s_c_162><s_d_130><|sid_end|>, its title is "Nexxus Youth Renewal Plump and Lift Blow Dry Spray, 7.5 Ounce", its categories are "Beauty > Hair Care > Styling Products > Hair Sprays"; <|sid_begin|><s_a_140><s_b_205><s_c_36><s_d_36><|sid_end|>, its title is "Nexxus Youth Renewal Rejuvenating Elixir, 0.94 Ounce", its categories are "Beauty > Hair Care > Hair & Scalp Treatments"; <|sid_begin|><s_a_155><s_b_190><s_c_158><s_d_81><|sid_end|>, its title is "CLEAR MEN SCALP THERAPY 2 in 1 Anti-Dandruff Shampoo and Conditioner, Dry Scalp Hydration, 12.9oz", its categories are "Beauty > Hair Care > Shampoos"; <|sid_begin|><s_a_113><s_b_67><s_c_140><s_d_115><|sid_end|>, its title is "Schick Hydro Silk Disposable Razor, 3 Count", its categories are "Beauty > Skin Care > Body > Moisturizers > Oils"; <|sid_begin|><s_a_21><s_b_45><s_c_114><s_d_49><|sid_end|>, its title is "Own Products Refining Moisture Night Cream", its categories are "Beauty > Skin Care > Face > Creams & Moisturizers > Night Creams"; <|sid_begin|><s_a_113><s_b_204><s_c_185><s_d_233><|sid_end|>, its title is "Nivea Q10 Skin Firming Body Lotion , 13.5 fl oz (Pack of 2)", its categories are "Beauty > Skin Care > Body > Moisturizers > Lotions"; <|sid_begin|><s_a_135><s_b_169><s_c_250><s_d_60><|sid_end|>, its title is "L'Oreal Paris Age Perfect Hydra-Nutrition Moisturizer, 1.7-Fluid

Ounce", its categories are "Beauty > Skin Care > Face > Creams & Moisturizers > Fluids & Lotions > Fluids"; <|sid_begin|><s_a_238><s_b_3><s_c_5><s_d_18><|sid_end|>, its title is "Cristophe Professional Glossing Shampoo, 10 Ounce", its categories are "Beauty > Hair Care > Shampoos"; <|sid_begin|><s_a_140><s_b_173><s_c_52><s_d_91><|sid_end|>, its title is "Tresemme Keratin Smooth Smoothing Creme Serum, 3.5 Ounce", its categories are "Beauty > Hair Care > Styling Products > Creams, Gels & Lotions"; <|sid_begin|><s_a_206><s_b_60><s_c_224><s_d_48><|sid_end|>, its title is "Natures Bounty Optimal Solutions Hair, Skin and Nails Gummies, 80 Count", its categories are "Beauty > Skin Care"; <|sid_begin|><s_a_140><s_b_177><s_c_103><s_d_200><|sid_end|>, its title is "Dove Hair Styling Oxygen Moisture Leave In Foam, 5.1 Ounce", its categories are "Beauty > Hair Care > Styling Products > Mousses & Foams";<|im_end|> <|im_start|>assistant <think> <pause><pause><pause> </think>

F.3 SID Metadata Decoding Prompt

For the metadata recovery experiment in Table 1, we use the first two items in each dataset's pretraining file as in-context examples and then query the remaining items. No judge model is used; predictions are evaluated by exact string match after parsing the generated Title: and Category: fields. The prompt template is shown below, with the final user turn line-wrapped for readability:

<|im_start|>system Please generate the title and category of the product based on its semantic ID.<|im_end|> <|im_start|>user {shot_1_sid}<|im_end| <|im_start|>assistant Title: "{shot_1_title}", Category: "{shot_1_categories}" <|im_end|> <|im_start|>user {shot_2_sid}<|im_end| <|im_start|>assistant Title: "{shot_2_title}", Category: "{shot_2_categories}" <|im_end|> <|im_start|>user Generate the Title and Category of this product {query_sid}. Please only generate with NO thinking! <|im_end|> <|im_start|>assistant

Generation uses greedy decoding (do_sample=False) with max_new_tokens=256.

F.4 Sample Rationales Used in CoT SFT

Below we provide the reasoning field from the first sample of each CoT SFT variant.

Template-Category.

The user is likely to buy items in the Beauty > Tools & Accessories > Mirrors > Makeup Mirrors category.

Template-Extended.

The user demonstrates interest in hair care and skin care products. By identifying the user's preference in natural, high-quality beauty and personal care items, we can predict that, to complement their previous purchases and enhance their daily grooming routine, the user will purchase beauty tools, for example, a Home Travel 9X/1X Folding Lighted Cosmetic Mirror.

Gemini 3.1 Flash-Lite Free-form.

The user consistently purchases hair care products and beauty tools, suggesting a strong interest in hair maintenance and styling. Given the previous acquisition of hair care treatments and a cosmetic mirror, the model can infer a logical progression toward purchasing functional hair styling accessories like clips to complement their existing routine.

Gemini 3.1 Pro Free-form.

The user's purchase history demonstrates a strong focus on hair care and beauty maintenance, moving from washing and conditioning treatments to grooming tools like a cosmetic mirror. An LLM would logically infer that after acquiring products to clean and treat their hair, the user's next step in their routine would be purchasing styling accessories like hair clips to manage and style it.

Gemini 3.1 Flash-Lite Rejection.

The user consistently purchases hair care products and beauty tools, indicating a strong focus on hair maintenance and grooming. By identifying the pattern of hair-related purchases, the model can infer a logical progression toward styling accessories like hair clips to complement the existing hair care routine.

Gemini 3.1 FL Gemini Rejection.

The user consistently invests in high-quality, natural, and specialized personal grooming products, suggesting a transition from purchasing consumable maintenance items to acquiring tools that facilitate their beauty and self-care routine. A lighted cosmetic mirror serves as a logical functional upgrade to complement their established regimen of premium hair and skin care products.

Gemini 3.1 Flash-Lite Restricted.

Step 1: The user's purchase history shows a consistent focus on beauty and personal care products. Step 2: This suggests the user is expanding from consumables to grooming tools. Step 3: A lighted cosmetic mirror is the most logical next purchase because it supports the user's existing routine.

F.5 Teacher Model Prompts for Reasoning Generation in CoT SFT

F.5.1 Free-form Reasoning Prompt

Given a user's purchase history, explain step-by-step how an LLM might reason to predict the next item they would purchase.
User's purchase history: {description}
The ground truth next item is: - Title: "{groundtruth_title}" - Categories: "{groundtruth_categories}"
Provide concise reasoning (1-2 sentences) explaining how an LLM could logically infer this next purchase from the user's prior items. Focus on patterns in categories, titles, or user preferences that would lead to this prediction. Do not include any preamble or labels—output only the reasoning text.

F.5.2 Format-Restricted Reasoning Prompt

P1: System Role & Task Definition You are an expert at analyzing e-commerce purchase patterns and predicting user preferences. You are helping create reasoning traces for recommendation training data. You will receive a user's purchase history, a list of candidate items, and one known positive target item. Your goal is to produce a predictive rationale: first infer the user's most likely next need from the purchase history, then verify whether the known target item is the best available match.
P2: Collaborative Context Presentation === USER PURCHASE HISTORY === {history_block}
=== CANDIDATE ITEMS === {candidate_block}
=== KNOWN TARGET ITEM === Candidate {target_candidate_id}: Title: {target_record["title"]}; Categories: {target_record["categories"]}; SID: {target_record["sid"]}
P3: Reasoning Procedure Work in two phases. Phase A: Before relying on the known target item, identify the purchase-history items with the strongest evidence. Cite the smallest number of SIDs needed to support the inferred pattern, usually 1-4. Based on those items, infer the user's likely next need, replenishment signal, complement need, routine continuation, tool-versus-consumable transition, or category progression. Phase B: Then evaluate whether the known target item matches that inferred need better than the strongest one or two alternatives in the candidate list. Prefer concrete signals such as repetition, recency, category progression, complementarity, brand continuity, and tool-versus-consumable transitions when they are supported by the evidence.
P4: Critical Guidelines as Output Constraint CRITICAL GUIDELINES: 1. When referring to purchase-history items, cite them directly using their SID. 2. In Step 1 and Step 2, do not mention the target title, target SID, or candidate numbers. 3. Use only evidence visible in the provided titles and categories. 4. Do not hallucinate hidden preferences or unsupported product attributes. 5. In Step 3, compare against at most two real alternative candidates; do not discuss the full candidate list. 6. If the target is not strongly supported, say "the fit is weaker than ideal but still the best available match" instead of inventing certainty. 7. Keep the rationale concise, specific, and information-dense. Avoid generic filler. 8. Do not output markdown fences.
P5: Structured Multi-Step Reasoning Format Return strict JSON with a single key "reasoning". The value must be one string in this format: Step 1: History-only evidence summary citing the most relevant SIDs and the dominant recurring pattern. Step 2: Based only on the history, infer the user's most likely next need or next category, and state confidence as high, medium, or low. Step 3: Now evaluate why the target item "{target_record["title"]}" is the best available match to that inferred need relative to the strongest one or two alternatives. Summary: One short concluding sentence, under 20 words.

F.6 General Language Benchmark Prompts and Generated Reasoning

For the diagnostic experiment in Table 3, we evaluated the target model on MMLU, HellaSwag, PIQA, and ARC-Challenge using a general-language benchmark run.

Text-generation prompt. For the target model's text generation, we used a chat-style prompt that explicitly opens a thinking block:

<lim_start|>user Answer the following multiple-choice question. You may explain briefly if useful, but finish with 'Final answer: <letter>'. {multiple-choice block} <lim_end|> <lim_start|>assistant <think>

The multiple-choice block lists the question followed by lettered options and ends with Final answer:. For MMLU only, the block contains five in-context examples sampled from the MMLU development split before the test question; the other benchmarks are zero-shot. HellaSwag questions are prefixed with Complete the sentence:. PIQA uses the two candidate solutions as options A/B. ARC-Challenge uses the answer choices from the dataset after normalizing them to A/B/C/D labels.

Logit prompt. For logit-based accuracy, we did not ask the model to generate a rationale. Instead, we used the following prompt and compared the next-token logits assigned to the option letters:

Answer the following multiple-choice question with the option letter only. {multiple-choice block ending with Answer:}

MMLU again uses five in-context examples, each ending with Answer: {gold letter}, followed by the test question ending with Answer:. HellaSwag, PIQA, and ARC-Challenge use the same zero-shot question blocks as above but end with Answer:.

What the model actually generated. Text generations used greedy decoding (do_sample=False) with max_new_tokens=128. The extracted reasoning_text field was non-empty for all 27,094 target examples. However, the generated text was usually not a multi-step rationale. It was most often a short answer-likelihood statement inside the `<think>` block, followed after `</think>` by SID tokens. For example, the first ARC-Challenge row has raw generation:

The user is likely to answer C </think> <|sid_begin|> <s_a_91><s_b_84> <s_c_156><s_d_20><|sid_end|> <lim_end|>

Thus, the content inside `<think>...</think>` for that example is exactly:

The user is likely to answer C.

Representative extracted reasoning_text entries from the artifact are:

- MMLU: The user is likely to answer D

- HellaSwag: The user is likely to answer D
- PIQA: The user is likely to answer B
- ARC-Challenge: The user is likely to answer C

Across all target examples, the most frequent extracted reasoning strings were The user is likely to answer C (11,891 examples), The user is likely to answer B (5,938), The user is likely to answer D (5,118), and The user is likely to answer A (1,732). There were 215 unique extracted reasoning strings in total. In 27,074 of 27,094 examples, the text after `</think>` began with SID tokens rather than a natural-language final answer. These artifacts show that the model learned to fill the thinking block with a shallow answer-prediction phrase, while the explicitly verbalized answer remained unreliable.

F.7 Dataset Statistics

Dataset	# Users	# Items	# Interactions
Beauty	22,363	12,101	198,502
Sports	35,598	18,357	296,337
Toys	19,412	11,924	167,597

Table 8: Dataset statistics after preprocessing. The three Amazon benchmarks span 11.9K–18.4K items and 167K–296K interactions, providing recommendation tasks of different scales.

G Complementary Experiment Results

G.1 Inference Speed of CoT SFT Variants

Table 9 shows that all CoT SFT variants incur substantially higher inference latency than PAUSEREC. Even the shortest template rationale is about 3.5× slower than PAUSEREC, while the longer template and teacher-generated variants are roughly 5.5–7.1× slower. This gap comes from the need to autoregressively generate rationale tokens before decoding the SID. In contrast, PAUSEREC inserts a fixed number of `<pause>` tokens and immediately proceeds to constrained SID decoding, so its latency is largely independent of rationale length.

We ran the benchmark on a single NVIDIA A100-SXM4-80GB GPU. We did not use vLLM. The benchmark used Hugging Face AutoModelForCausalLM.generate with PyTorch, fp16 on CUDA, batch size 16, greedy decoding (do_sample=False), max reasoning tokens 128, and max SID tokens 20.

Model	Seconds/Sample
Template-Category	0.2043 ± 0.0255
Template-Extended	0.4211 ± 0.0459
Gemini 3.1 Flash-Lite Free-form	0.3224 ± 0.0423
Gemini 3.1 FL Gemini Rejection	0.3711 ± 0.0466
Gemini 3.1 Flash-Lite Rejection	0.3379 ± 0.0431
PAUSEREC	$\mathbf{0.0591 \pm 0.0093}$

Table 9: Inference latency for CoT SFT variants and PAUSEREC on 500 Beauty samples. PAUSEREC is fastest because it uses fixed `<pause>` tokens instead of generating natural-language rationales.

Dataset	# ps	H@5	H@10	N@5	N@10
Beauty	1	0.0562	**0.0760**	0.0394	0.0466
	3	0.0559	0.0758	0.0393	0.0465
	5	**0.0568**	0.0746	**0.0401**	**0.0467**
	10	0.0555	0.0742	0.0396	0.0465
Sports	1	0.0290	0.0416	0.0200	0.0240
	3	**0.0294**	0.0407	**0.0203**	0.0240
	5	**0.0294**	**0.0422**	**0.0203**	**0.0245**
	10	0.0286	0.0415	0.0199	0.0241
Toys	1	0.0614	0.0835	**0.0440**	**0.0513**
	3	0.0610	0.0831	0.0434	0.0506
	5	**0.0615**	**0.0838**	0.0434	0.0509
	10	0.0612	0.0826	0.0437	0.0509

Table 10: Extended results for different numbers of `<pause>` tokens. No single value dominates every dataset and metric, but $k = 5$ gives the best overall tradeoff and is used in the main experiments; boldface and underlining mark the best and second-best results.

G.2 Parameter Analysis

In this subsection, we provide supplementary experiment results on parameter analysis for our proposed PAUSEREC pipeline. Specifically, as shown in Table 10, we provide the full pause-count sweep behind Fig. 4. We observe that, the number of `<pause>` tokens yielding best performance is not identical for every dataset and metric, but k=5 is the most robust setting: number of `<pause>` tokens being 5 is best or tied for best on 9 of 12 metrics and remains close to the best result on the remaining metrics. Using only one or three pauses is already competitive, suggesting that a small latent computation window is useful. Increasing to ten pauses does not provide consistent additional gains and sometimes slightly hurts performance, indicating that the benefit of pause-based reasoning saturates after a moderate number of latent steps.