

ReflectRec: Cycle-Consistency Preference Verification for LLM-based Recommendation

Dae Kyoo Kim^{1,2} Heejung Cho^{1,3,*} Sungmoo Hong^{1,4} Eunil Park^{1,5,*}
¹Sungkyunkwan University ²NAVER Corp. ³KIAT ⁴KT Corp. ⁵Jaume I University
{kdg7447, heejungcho, isean, eunilpark}@skku.edu

Abstract

CoT-style LLM recommenders generate a natural-language preference note as a chain-of-thought prefix to ranking, but the note is trusted without verification, and the ranker never consults behaviorally similar users. We propose **ReflectRec**, organized around a preference-to-history round-trip verifier whose recall score serves as an operational signal of preference-to-history recoverability, enabling conditional regeneration for low-recoverability preferences and selection among regeneration attempts. Around this verifier, we wire a semantic Stage-1 prompt, conditional regeneration, and Forward-Look retrieval that injects future watches of behaviorally similar users as a collaborative hint. On MovieLens-100K and MovieLens-1M, the full pipeline outperforms CoT4Rec-ICL by +16.25 / +4.46 pp Hit@10, with positive gains across all ValScore buckets and three LLM backbones (Qwen3-30B, Qwen2.5-72B, and Llama-3.3-70B). We also observe a similar verifier-guided regeneration pattern on Amazon Electronics.

1 Introduction

Sequential and conversational recommendation is rapidly being reshaped by large language models (Geng et al., 2022; Hou et al., 2024; Zhang et al., 2025; Wu et al., 2024). Unlike classical collaborative filtering, which compresses interactions into latent vectors (Hwang and Park, 2021; Rendle et al., 2009), Large Language Model (LLM) based recommenders represent histories and items in natural language and can reason about *why* a user may like an item, producing human-readable rationales (Wei et al., 2022; Bao et al., 2023; Gao et al., 2023; Luo et al., 2025).

A representative instance is CoT4Rec (Yue et al., 2025), which factors ranking into two stages: a *preference analyzer* turns a raw history into

a short natural language preference note, and a *recommender* consumes the history, preference, and candidate list to predict the next item. Such pipelines inherit the interpretability of language-based recommendations while delivering strong empirical gains.

However, two weaknesses persist in Chain of Thought (CoT) style pipelines. The first is the *absence of verification for intermediate preferences*. LLM outputs are sensitive to prompt variation (Lu et al., 2022), hallucinate when forced to externalize user preferences (Huang et al., 2025), and can hurt downstream rankers when they are too abstract or factually inconsistent. However, no established mechanism measures the per-sample faithfulness of the preference or repairs low-quality preferences before they reach the ranker. The second is the *absence of explicit reference to behaviorally similar histories from other users*. At inference, the ranker attends only to one user’s history and its compressed preference; it never consults what users with similar local viewing patterns recently consumed. The local collaborative cue that classical recommenders draw heavily on is missing, and the *lost-in-the-middle* phenomenon (Liu et al., 2024) aggravates the problem for LLMs, whose attention over long histories is uneven.

We introduce **ReflectRec** (Fig. 1), a recommendation pipeline built around a single unifying signal: the *cycle-consistency verification score*. This score is computed by feeding the generated preference P_u back into the LLM alongside 50 candidates, the user’s true history items plus random negatives, and measuring the recall v_u over true history items. This single value is used for four functions: (i) an operational signal of preference-to-history recoverability, (ii) *gating* of regeneration, (iii) *selection* of the regeneration attempt with the highest recoverability score, and (iv) *diagnosis* of how downstream module effects vary across ValScore levels. Around this verifier, we wire four

*Corresponding authors.

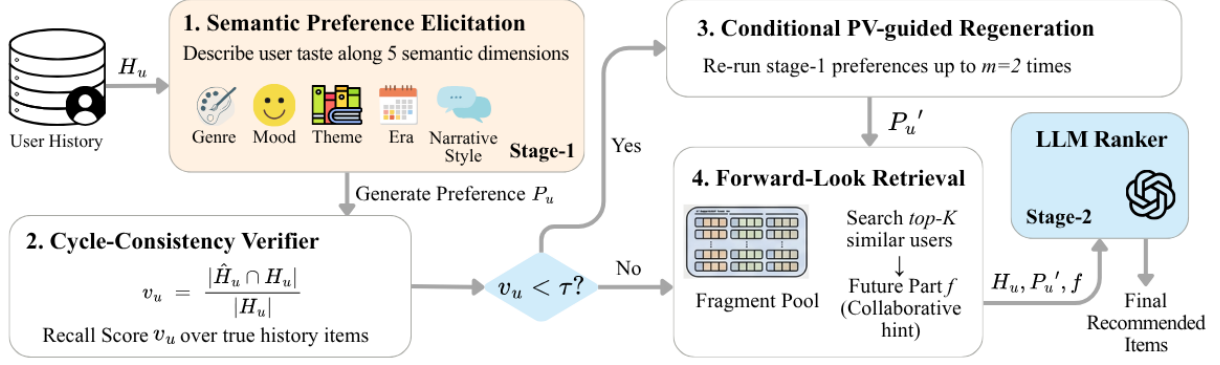


Figure 1: REFLECTREC overview. Stage-1 elicits a structured user preference P_u ; the cycle-consistency verifier yields a score of preference-to-history recoverability, which gates regeneration, selects the regeneration attempt with the highest recoverability, and is used to analyze downstream module effects. Conditional regeneration regenerates low-recoverability preferences, and Forward-Look retrieval injects the *future watches* of behaviorally similar users into the ranker prompt.

modules, which are described below.

Semantic Preference Elicitation. A single LLM call instructs the analyzer to describe user taste along five semantic dimensions, genre, mood, theme, era, and narrative style, without quoting any movie titles from the user’s history. No embedding or retrieval is involved at this stage; the resulting preference note serves as the input substrate for all three downstream modules (verifier, regeneration, and forward-looking).

Cycle-Consistency Verifier (four roles). It produces the recall v_u defined above and uses it simultaneously for measurement, gating, selection, and diagnosis.

Conditional PV-Guided Regeneration. Only samples with ValScore $< \tau=0.4$ (the LOW bucket) have their preference re-run up to $m=2$ times. The verifier scores each attempt and retains the highest score.

Forward-Look Retrieval. From a fragment pool built on the training split only, each fragment is a tuple (h, f) where h is a 5 item history window and f is the user’s *next 5 items actually watched*, we retrieve top- k fragments by history similarity and inject the *future* part $\{f\}$ into the ranker prompt as a collaborative hint. Empirically, this retrieval contributes most to the MID/HIGH/PASS buckets (§5.4).

Empirical Findings. We evaluate on MovieLens-100K ($n=843$) and MovieLens-1M ($n=1500$) with three LLM backbones (Qwen3-30B, Qwen2.5-72B, and Llama-3.3-70B). We examine whether the

cycle-consistency verifier provides a per-sample operational signal of preference-to-history recoverability; conditional regeneration improves Hit@K for low-recoverability samples; Forward-Look retrieval provides a collaborative signal complementary to regeneration; and each module contributes to overall performance with effects that vary across ValScore buckets.

The full pipeline reaches Hit@10 = 31.67% on ML-100K and 39.93% on ML-1M. Both gains are statistically significant ($p=0.014$, $p=0.036$, paired bootstrap). In the random-future control, similarity-based Forward-Look yields +3.32 / +3.14 pp higher Hit@10 than arbitrary future hints. In the unified pool ($n=2343$), all four ValScore buckets exhibit positive $\Delta\text{Hit@10}$.

Contributions. We introduce a task-grounded *cycle-consistency verifier* that checks whether an LLM-generated preference note can recover the user’s history items from a candidate pool, with a single recall score simultaneously serving four operational roles: measure, gate, select, and diagnose. Around this verifier, we build **ReflectRec**, a four-module pipeline that yields complementary gains across ValScore buckets, supported by a random-future control, paired bootstrap CIs, and consistent results across three LLM backbones.

2 Related Work

LLM as a recommender. Recent advancements in language modeling shifted personalized recommendations from traditional numerical prediction (Hwang and Park, 2021; Rendle et al., 2009) to linguistic reasoning by reformulating them as

a text-to-text generation task (Geng et al., 2022). Based on this generative paradigm, contemporary approaches have incorporated diverse strategies, such as zero-shot inference (Hou et al., 2024), collaborative embedding fusion (Zhang et al., 2025; Liao et al., 2024), data augmentation (Wang et al., 2024), and external knowledge grounding (Xi et al., 2024).

Chain-of-thought recommendation. Yue et al. (2025) introduce the two-stage analyzer/ranker pipeline that we extend. Zheng et al. (2024) learn LLM-aligned representations; all assume that intermediate rationales help downstream recommendations, but none explicitly measure the rationale’s faithfulness to the user history.

Retrieval-augmented recommendation. RAG is the standard for knowledge-intensive NLP (Lewis et al., 2020). Luo et al. (2025) combine RAG with LLM recommendations by retrieving similar items. A natural extension is to retrieve behavior fragments (short interaction segments of other users); within that space, we make a specific design choice: the retrieval target should be aligned with the next-item prediction task rather than with the preference itself. Therefore, we do not score fragments by similarity to the preference; instead, we score them by similarity of their history part to the test user’s history and feed their future part (the next five items actually watched after the history window) into the ranker as a collaborative hint.

Quality and self-verification of LLM intermediates. Prompt sensitivity (Lu et al., 2022), hallucinated preferences (Huang et al., 2025), and self-correction via backtracking (Cai et al., 2025) all motivate the explicit verification of LLM intermediates. Liu et al. (2024) show that LLM attention degrades in the middle of the long contexts. Round-trip/cycle-consistency has a long history in NLP and vision, back-translation for translation quality, QAGS and FactCC for summarization fact-checking, and self-consistency CoT for reasoning. An adjacent example is Kim et al. (2025), who use reverse reconstruction (recovering the original problem from a reasoning trajectory and answer) as a mutual-verification signal to improve the logical-semantic consistency of small language models on mathematical reasoning. We apply the same round-trip idea to preference verification in LLM-based recommendations, where a single verification score simultaneously drives the measure, gate, and select

operations and diagnoses the subpopulation effects of downstream modules.

3 Method

3.1 Problem Formulation

Let $\mathcal{U}$ denote the set of users and $\mathcal{I}$ the set of items. Each user $u \in \mathcal{U}$ has a chronologically ordered history $H_u = [i_1, \dots, i_T]$ with $i_t \in \mathcal{I}$. Given a candidate set $C_u \subset \mathcal{I}$, the recommender predicts the next item.

$$\begin{aligned} P_u^{(0)} &= f_{\text{pref}}(H_u), \\ v_u &= f_{\text{val}}(H_u, P_u^{(0)}), \\ P'_u &= f_{\text{regen}}(H_u, P_u^{(0)}, v_u), \\ R_u &= f_{\text{ret}}(H_u; \mathcal{F}_{\text{pool}}), \\ \hat{y}_u &= f_{\text{rec}}(H_u, P'_u, R_u, C_u) \end{aligned} \quad (1)$$

f_{pref} produces the semantic preference, f_{val} the verification score, f_{regen} returns the regenerated preference when $v_u < \tau$ and the original otherwise, f_{ret} retrieves Forward-Look fragments, and f_{rec} is the final ranker. All LLM-based stages use the same backbone.

3.2 Stage 1: Semantic Preference Prompt

Design Principle. The ranker prompt already contains raw history H_u . For the preference P_u to contribute information the ranker cannot infer elsewhere, P_u must be *complementary* to H_u rather than redundant with it: P_u should encode descriptive content (genre, sub-genre, mood and emotional tone, thematic content, era or stylistic period, narrative style) rather than restating items.

Prompt. The Stage-1 prompt asks the LLM for a 60–100 word descriptive preference along the five axes above. It discourages verbatim title quotations; any title that still appears is detected by exact case-insensitive substring matching and is used as a tie-breaker when selecting among regeneration attempts (§3.4).

3.3 Cycle-Consistency Preference Verification

Principle: preference $\rightarrow$ history round-trip. Directly evaluating how well a preference note P_u reflects a user’s preference is difficult. “How well does this text describe the user’s taste?” has no direct ground truth, and LLM-judge-style self-rating is prone to prompt sensitivity and sycophancy (Lu et al., 2022). We measure how many items from the history H_u that generated P_u can be recovered

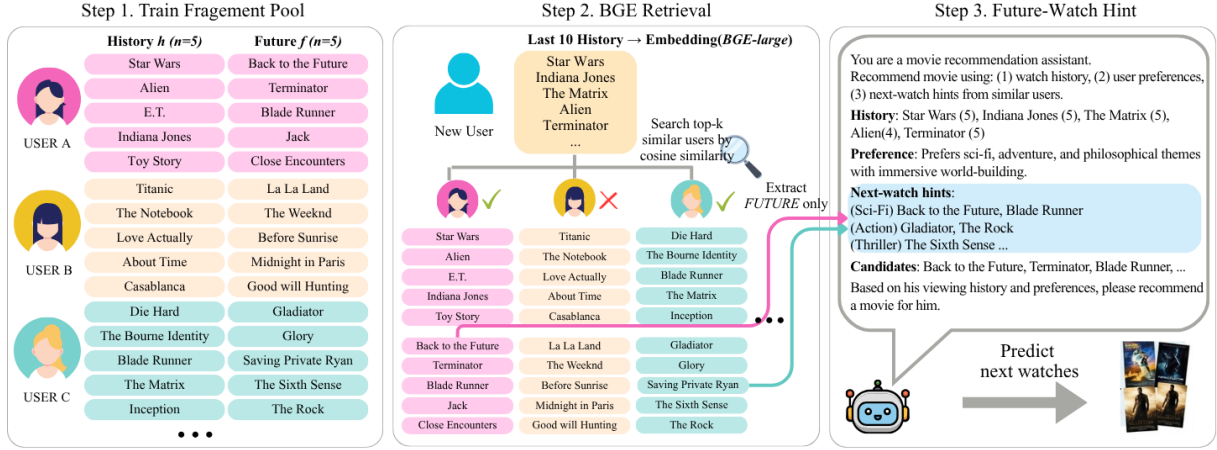


Figure 2: Training users’ histories are split into history h and future f fragments to build a fragment pool (Step 1). A new user’s last 10 interactions are embedded via BGE-large and matched to the top- K similar users by cosine similarity (Step 2). Only the future fragments of matched users are extracted and injected as next-watch hints into the LLM ranker prompt alongside the user’s history and preferences (Step 3).

from P_u . We use this reconstruction recall as an operational signal of preference-to-history recoverability.

Verification score. Given P_u , the verifier LLM receives a mixed pool of 50 candidates (the user’s true history items plus random negatives) and is asked to select the ones that the user has most likely watched. The recall over true history items is:

$$\text{ValScore}(P_u) = \frac{|\hat{H}_u \cap H_u|}{|\hat{H}_u|}. \quad (2)$$

A higher ValScore indicates that more of the user’s observed history items can be recovered from the generated preference. We therefore use ValScore as a score of preference-to-history recoverability.

Round-trip versus LLM-judge baseline. An alternative design would ask the LLM to rate the preference text on a 1–5 scale. Such self-ratings tend to (i) give consistently high scores to abstract, well-written preferences (sycophancy) and (ii) be sensitive to minor wording changes in the rating prompt. In contrast, round-trip verification is grounded in *concrete, observable output* (the recall over history items) and therefore naturally penalizes hallucinated abstract preferences, as they cannot supply enough cues to distinguish the true history items from random negatives.

Candidate pool of size 50. The pool of 50 contains the user’s true history items and random negatives drawn from $\mathcal{I} \setminus H_u$. With an average $|H_u| = 10$, the positive-to-negative ratio is 1:4.

We use this 1:4 construction as a descriptive reference for interpreting ValScore. The regeneration operating threshold $\tau = 0.4$ is selected empirically based on downstream regeneration performance.

Four roles of the verifier. The single score v_u plays four roles in the pipeline. First, it serves as a per-sample recoverability signal (**measure**). Second, $v_u < \tau=0.4$ defines the LOW bucket, where conditional regeneration fires (**gate**). Third, when regeneration produces $m=2$ attempts, the verifier scores each attempt and retains the highest-scoring one (**select**). Finally, ValScore is used to analyze how downstream module effects vary across score levels (**diagnose**). The verifier itself does not directly change Hit@K; it is used for regeneration gating and selection and for analyzing downstream module effects.

3.4 Conditional PV-guided Regeneration

For samples with $\text{ValScore} < \tau$ we re-run Stage-1 at a sampling temperature 0.5 up to $m=2$ times. Each attempt is re-verified, and the verifier selects the highest-scoring attempt as P'_u . Samples with $\text{ValScore} \geq \tau$ keep $P'_u = P_u^{(0)}$.

A natural concern is that the aggressive regeneration of all weak samples could over-sharpen the preference and hurt the downstream Hit@K. We confirm empirically that regeneration helps only on the LOW bucket: across 8 independent regeneration seeds at $T=0.5$ on the unified $n=116$ LOW pool, the mean $\Delta\text{Hit@10}$ is +2.48 pp (95% CI [+1.01, +3.94], $t=4.0$, $p \approx 0.005$; significant after

Bonferroni correction for the six tested temperatures). The threshold $\tau=0.4$ is selected from a sweep $\tau \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$ over the unified pool (Table 10).

3.5 Forward-Look Retrieval

Design principle. As illustrated in Fig. 2, the forward-look retrieval module operates across three steps: fragment pool construction, history-based retrieval, and future-fragment injection into the ranker prompt. A preference-similar retrieval signal (“other users whose preferences resemble the current user’s”) is structurally aligned with P'_u . When P'_u is already a strong signal, such retrieval is redundant as it can shift the ranker’s attention without supplying new evidence about the *next item*. Therefore, we align the retrieval target with the prediction task by retrieving fragments that tell the ranker what users with similar histories *actually watched next*.

Fragment pool construction. The fragment pool $\mathcal{F}_{\text{pool}}$ is built only from the training split. Each fragment is a tuple (h, f) in which h is a 5-item history window from a training user’s chronological sequence and f is the next 5 items watched right after h . We use a 300-fragment pool for ML-100K and a 3000-fragment pool for ML-1M (subsampled from $\sim 137\text{k}$ candidates).

Embedding and retrieval. Each fragment’s history part h and the test user’s history string H_u are encoded with BGE-large-en-v1.5 (Xiao et al., 2024) using the instruction prefix “*Represent this movie list for retrieving items with similar genre*”. At inference, we retrieve the top- $k=10$ fragments by cosine similarity to H_u . We use a single aspect rather than the four aspects of prior work; multi-aspect retrieval is empirically less robust across datasets (§5.6). The future parts $\{f_j\}$ of the retrieved fragments are concatenated into a hint block as follows:

```
Users with similar viewing patterns next
watched:
  (genre) <future titles 1-5>
  . . .
```

This block is inserted between P'_u and C_u in the Stage-2 prompt.

Prompt guidance matters. A *collaborative-filtering-aware* system prompt adds +0.83 pp $\Delta\text{Hit}@10$ on ML-100K; without it the LLM treats the hint block as distractors.

Component	Setting
LLM (analyzer/ranker/verifier)	Qwen3-30B-A3B-Instruct
Retrieval embedder	BAAI/bge-large-en-v1.5
Fragment size w	5
Future window $ f $	5
Forward-look top- k	10 (single aspect)
Verifier candidates	50 (true + random neg.)
Verification threshold τ	0.4 (sweep: 0.3–0.7)
Regeneration attempts m	2
Regeneration temperature	0.5 (main MovieLens run)
ICL few-shot examples	3

Table 1: Implementation settings. The threshold sweep uses $T=0.7$ in a separate single-seed analysis, while Electronics regeneration uses $T=1.0$.

3.6 Stage 2: Ranker

The Stage-2 input concatenates the user history, (possibly regenerated) preference, forward-look hint, and candidate set:

$$X_u = [H_u; P'_u; R_u; C_u]. \quad (3)$$

The ranker uses the same Qwen3-30B backbone as Stage-1 and the verifier. Full implementation settings are summarized in Table 1.

4 Experimental Setup

4.1 Datasets

We use two MovieLens datasets: **MovieLens-100K** (ML-100K) (Harper and Konstan, 2015) and **MovieLens-1M** (ML-1M) (Harper and Konstan, 2015). For cross-domain validation, we additionally use **Amazon Electronics** (ELEC) from the McAuley Amazon Reviews dataset (He and McAuley, 2016), a product domain with sparser interaction patterns and titles dominated by technical specifications. The dataset statistics are presented in Table 2. For each user in all three datasets, the last 30 ratings form the evaluation window: the first 10 items are the history H_u , and the next 20 are the ground-truth target list (Hit@ k checks whether the predicted title appears within the first k targets). The candidate set C_u is a shuffled 100-item set containing targets and random negatives.

Trained baselines (GRU4Rec, SASRec, and LLM-SRec) are first reported under standard sampled evaluation (leave-one-out, GT + 99 negatives) to provide cross-method context. Because these leave-one-out results are not directly comparable to the ICL conditions, SASRec and GRU4Rec are also evaluated under the same 100-candidate protocol as ReflectRec. The same candidate pool and rank-based Hit@K computation are used for the

Dataset	#Users	#Items	#Inter.	Sparsity	Deg.Cent.
ML-100K	943	1,682	100,000	0.9370	106.04
ML-1M	6,040	3,416	999,611	0.9516	165.50
Elec.	1,500	55,588	45,000	0.9995	30.00

Table 2: Dataset statistics. ML-100K and ML-1M are GroupLens originals; Elec. is a 1,500-user subset of Amazon Electronics (McAuley) sampled from users with ≥ 30 ratings. #Items is the item universe spanned by the test pool; #Inter is the evaluation-window interaction count ($1,500 \times (10 + 20)$).

protocol-matched trained baselines and ICL conditions. The primary protocol follows user-wise chronological order for comparison with CoT4Rec. In the timestamp-safe setting, a fragment is eligible for retrieval only if its complete five-item future window ends before the query cutoff.

4.2 Compared Conditions

Trained baselines include **GRU4Rec** (Hidasi et al., 2015), **SASRec** (Kang and McAuley, 2018), and **LLM-SRec** (Kim et al., 2025). ICL baselines are **LLMSRec-Syn** (Wang and Lim, 2024) and **CoT4Rec-ICL** (Yue et al., 2025). Our three ReflectRec conditions add modules cumulatively: **Stage-1 only** (semantic preference, no verifier), **Stage-1+Regen** (adds verifier-gated regeneration), and **ReflectRec (Full)** (all four modules).

4.3 Metrics

We report Hit@1, Hit@5, and Hit@10 across all conditions, including the baselines.

5 Results

5.1 Main Results

Table 3 reports the cross-method comparison on MovieLens-100K, MovieLens-1M, and Amazon Electronics. Alongside trained-recommender results under the standard leave-one-out protocol, we report SASRec and GRU4Rec rerun under the same 100-candidate protocol as ReflectRec.

Leave-one-out and 100-candidate results use different evaluation protocols and should not be directly compared. Under the same 100-candidate protocol, the gap between trained baselines and ICL methods becomes smaller than under the leave-one-out comparison. For example, on ML-1M, the Hit@10 gap between SASRec and ReflectRec decreases from 42.21 pp to 22.47 pp, while ReflectRec (39.93) exceeds GRU4Rec (31.40). The 100-candidate protocol uses only the first 10 user inter-

actions as history, whereas conventional leave-one-out evaluation uses the full preceding interaction history. Table 4 reports the apples-to-apples ICL comparisons. The full ReflectRec pipeline reaches Hit@10 31.67% on ML-100K and 39.93% on ML-1M, +16.25 pp / +4.46 pp over CoT4Rec-ICL and +3.91 pp / +3.40 pp over Stage-1.

Retrieval decomposition. A retrieval-only decoder using the same Forward-Look evidence without the LLM ranker achieves Hit@10 of 42.70 / 45.40 on ML-100K / ML-1M. This shows that the retrieved collaborative evidence itself provides a strong recommendation signal.

5.2 Cumulative Module Ablation

Table 4 serves as a cumulative ablation; each row adds a single module on top of the previous one: **Semantic Stage-1** adds +12.34 pp Hit@10 on ML-100K (15.42→27.76), the largest single gain, confirming that descriptive preference complementarity is the dominant lever. **Verifier + Regen** yields small overall gains (+0.59 / −0.13 pp) but concentrates in the LOW bucket: the 8-seed mean $\Delta\text{Hit@10} = +2.48$ pp at $T=0.5$ on the unified $n=116$ LOW pool (95% CI [+1.01, +3.94], $p \approx 0.005$). **Forward-Look** adds +3.32 / +3.53 pp (bootstrap CIs exclude zero; $p=0.014$, $p=0.036$).

5.3 Cross-domain Validation on Amazon Electronics

To test whether the verifier and regeneration mechanism transfer beyond the movie domain, we evaluate them on Amazon Electronics with a Stage-1 prompt adapted to product semantics. Instead of the thematic axes (genre, mood, era, narrative style) used for movies, Electronics Stage-1 elicits five concrete product axes: technical specifications, industry standards/compatibility, form factor, brand family, and quality tier (see Appendix K for the full template). The verifier, regeneration mechanism, ranker, and $\tau=0.4$ threshold are identical to those in the movie setup; we set the regeneration temperature to 1.0 and $m=2$ attempts. The main Electronics evaluation focuses on transfer of the verifier and regeneration mechanism, while Forward-Look transfer is analyzed separately.

Table 5 reports the results for $n=1,500$ test users. Stage-1 alone reaches Hit@10 = 29.67%; adding conditional regeneration yields 29.87% ($\Delta_{\text{overall}} = +0.20$ pp). The module’s effect concentrates in the LOW bucket, as on the movie datasets:

Method	ML-100K			ML-1M			Electronics		
	Hit@1	Hit@5	Hit@10	Hit@1	Hit@5	Hit@10	Hit@1	Hit@5	Hit@10
<i>Trained baselines — leave-one-out</i>									
GRU4Rec	10.29	29.16	46.98	13.74	38.81	53.46	16.84	37.96	49.92
SASRec	20.89	56.10	72.75	37.81	71.85	82.14	22.28	48.01	59.21
LLM-SRec	9.86	31.28	48.04	17.38	45.89	62.30	22.18	49.70	62.79
<i>Trained baselines rerun — 100-candidate protocol</i>									
GRU4Rec	5.69	21.00	39.38	4.60	18.33	31.40	4.53	20.20	35.13
SASRec	9.61	40.09	64.06	11.87	42.73	62.40	7.53	34.33	61.73
<i>ICL — 100-candidate protocol</i>									
LLMSRec-Syn	3.80	13.40	24.44	4.20	17.67	33.40	3.20	14.93	23.40
CoT4Rec-ICL	1.19	6.76	15.42	5.20	20.67	35.47	4.20	14.20	26.80
ReflectRec (ours)	3.08	16.61	31.67	6.60	25.00	39.93	4.47	16.20	29.87

Table 3: Cross-method comparison across the three datasets. Leave-one-out and 100-candidate results use different evaluation protocols and should not be directly compared. Under the same 100-candidate protocol, the gap between trained baselines and ICL methods is smaller than suggested by the leave-one-out results.

Method	Hit@1	Hit@5	Hit@10
ML-100K ($n=843$)			
CoT4Rec-ICL	1.19	6.76	15.42
Stage-1 (semantic)	3.32 (+2.13)	14.23 (+7.47)	27.76 (+12.34)
w/ Regen	3.44 (+0.12)	14.23 (—)	28.35 (+0.59)
Full (w/ FL)	3.08 (-0.36)	16.61 (+2.38)	31.67 (+3.32)
ML-1M ($n=1500$)			
CoT4Rec-ICL	5.20	20.67	35.47
Stage-1 (semantic)	6.07 (+0.87)	22.40 (+1.73)	36.53 (+1.06)
w/ Regen	6.33 (+0.26)	22.47 (+0.07)	36.40 (-0.13)
Full (w/ FL)	6.60 (+0.27)	25.00 (+2.53)	39.93 (+3.53)

Table 4: Ablation results under the 100-candidate protocol. Forward-Look retrieval is abbreviated as FL throughout. The full pipeline (Stage-1 + Regen + FL) improves Hit@10 over CoT4Rec-ICL by +16.25 pp on ML-100K and +4.46 pp on ML-1M. The ML-1M CoT4Rec-ICL baseline is our own measurement using the same Qwen3-30B backbone, as the original CoT4Rec paper did not report ICL results on ML-1M.

Condition	Hit@1	Hit@5	Hit@10
<i>Full pool, $n=1,500$</i>			
Stage-1 only	4.47	16.07	29.67
w/ Regen	4.47 (+0.00)	16.20 (+0.13)	29.87 (+0.20)
<i>LOW bucket $v_u < 0.4$, $n=105$</i>			
Stage-1 only	3.81	13.33	22.86
w/ Regen	2.86 (-0.95)	15.24 (+1.90)	26.67 (+3.81)

Table 5: Amazon Electronics evaluation (Qwen3-30B, same 100-candidate ICL protocol) with a spec-centric Stage-1 prompt. The LOW-bucket Hit@10 gain follows the same direction as in the movie datasets.

on the $n=105$ LOW sub-pool, $\Delta\text{Hit@10} = +3.81$ pp (one-sided paired bootstrap $p_1=0.157$), the same direction as the movies unified LOW pool (+2.48 pp, 8-seed mean at $T=0.5$).

Bucket	n	ML-100K	ML-1M
LOW [0, 0.4)	29 / 87	+0.00	+1.15
MID [0.4, 0.5)	84 / 145	+2.38	+4.14
HIGH [0.5, 0.7)	301 / 525	+3.32	+4.76
PASS [0.7, 1.0]	429 / 743	+3.73	+2.83
Weighted (unified)	843 / 1500	+3.32	+3.53

Table 6: Per-bucket forward-look contribution to $\Delta\text{Hit@10}$ (%), holding conditional regeneration constant. The unified LOW bucket ($n=116$) is +0.86 pp, completing the all-buckets-positive property of the full pipeline on the unified pool ($n=2343$).

For Forward-Look transfer on Electronics, direct raw-title Forward-Look with top- $k=10$ achieves Hit@10 of 28.33. Candidate-aligned semantic tie-break achieves 34.47 / 41.20 / 43.33 at $k=10/50/100$, while retrieval-only using the same retrieval evidence achieves 34.40 / 39.87 / 42.40. These results suggest that candidate-aligned retrieval may be more suitable for Electronics than direct raw-title integration.

5.4 Verification-Score Bucket Analysis

To characterize the contribution of each module, we stratify Hit@10 by val_score bucket {LOW, MID, HIGH, PASS}. Table 6 reports the per-bucket $\Delta\text{Hit@10}$ of the full pipeline against the Stage-1 + Regen variant (i.e., the forward-looking contribution alone, holding regeneration constant).

The bucket structure allows the verification score to serve as a *descriptive lens* for where each module’s gains concentrate: conditional regeneration’s gains are concentrated in the LOW bucket; forward-look’s gains are concentrated in MID and above. The two modules are tied by the same val_score

Variant	ML-100K Δ Hit@10	ML-1M Δ Hit@10
$k=1$, 1 aspect	+2.37	+0.60
$k=5$, 1 aspect	+3.08	+1.80
$k=10$, 1 aspect (selected)	+3.32	+3.53
$k=20$, 1 aspect	+4.74	+1.20
$k=1$, 4 aspects (multi)	+3.80	+2.53
$k=5$, 4 aspects (multi)	+4.74	+2.80
Preference query ($k=5$)	+1.66	–
Plain prompt ($k=10$)	+0.59	+1.53

Table 7: Forward-look variant ablation (selected rows out of 23). The chosen configuration is $k=10$, single-aspect retrieval, and the collaborative-filtering-aware system prompt.

signal but produce their effects on disjoint sub-populations.

5.5 Threshold Sweep

A sweep over $\tau \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$ confirms $\tau=0.4$ as the optimal threshold; tighter values over-restrict unrecoverable samples and looser values dilute the LOW signal (Appendix A).

5.6 Forward-Look Design Ablation

Table 7 summarizes a sweep of 23 forward-looking variants across the two datasets (the full leaderboard is in the supplementary material). Increasing k from 1 to 10 removes the ML-100K LOW bucket damage that small- k retrieval induces (-10.34 pp at $k=1 \rightarrow 0.00$ pp at $k=10$), suggesting that the LLM benefits from a more diverse set of collaborative cues. Going beyond $k=10$ saturates or reverses gains on ML-1M owing to oversampling, making $k=10$ the cross-dataset sweet spot. Single-aspect retrieval is more robust than 4-aspect multiview retrieval across datasets and simplifies the retrieval module to a single BGE embedding pass. Using the chosen preference rather than the user history as the retrieval query yields an over-sharpened candidate set that substantially reduces Hit@10 (for example, -17.24 pp on ML-100K LOW). Finally, without a collaborative-filtering-aware system prompt, the LLM treats the hint block as distractor content; a single line of descriptive guidance in the system prompt recovers most of this loss.

5.7 Backbone Robustness

To confirm that ReflectRec is not tied to one LLM, we rerun the full pipeline on ML-100K with two additional backbones: qwen/qwen2.5-72b-instruct (same family, larger scale, and earlier generation) and

Backbone	Stage-1	+Regen	Full
Qwen3-30B (ours)	27.76	28.35	31.67
Qwen2.5-72B	28.00	28.59	39.03
Llama-3.3-70B	31.32	30.96	36.65
Δ vs Stage-1			
Qwen3-30B	–	+0.59	+3.91
Qwen2.5-72B	–	+0.59	+11.03
Llama-3.3-70B	–	–0.36	+5.34

Table 8: Backbone robustness on ML-100K. All three backbones show positive gains for the full pipeline over the Stage-1-only baseline. The larger Qwen2.5-72B extracts the largest gain (+11.03 pp), suggesting that the collaborative signal in forward-look retrieval scales with model capacity.

Hint source	ML-100K	ML-1M
None (Stage-1 + Regen)	0.00	0.00
Random-future control	+0.12	–1.07
w/ Forward-look retrieval	+3.44	+2.07
Net retrieval contribution	+3.32	+3.14

Table 9: Random-future control compares similarity-based Forward-Look with arbitrary future hints. All cells are Δ Hit@10 (%) relative to the no-hint baseline.

meta-llama/llama-3.3-70b-instruct (different family). All LLM stages (Stage-1, verifier, conditional regeneration, and ranker) are regenerated under the new backbone; BGE retrieval embeddings are backbone-independent and are reused.

Table 8 shows that (i) the full ReflectRec pipeline outperforms Stage-1-only on every tested backbone, with Δ ranging from +3.91 to +11.03 pp; (ii) larger and newer backbones extract larger gains from the forward-look retrieval module; and (iii) the bucket-level non-negativity property holds under both same-family scale-up and cross-family transfer (Appendix D).

5.8 Random-Future Control

To examine the effect of history-similarity-based fragment retrieval in Forward-Look, we compare it with a random-future condition that replaces similarity retrieval with five fragments sampled uniformly from the fragment pool. Table 9 shows that the random hint yields +0.12 pp on ML-100K and -1.07 pp on ML-1M, while Forward-Look retrieval yields +3.44 pp / +2.07 pp. In the random-future control, similarity-based Forward-Look yields +3.32 / +3.14 pp higher Hit@10 than arbitrary future hints on ML-100K / ML-1M.

6 Discussion

The dominant gain is Stage-1 preference complementarity (+12.34 pp on ML-100K): descriptive abstractions supply information the ranker cannot infer from H_u alone.

History-aligned retrieval provides collaborative evidence complementary to P'_u . The retrieval-only results show that the retrieved collaborative evidence itself is a strong recommendation signal, suggesting that effective LLM-based integration of this evidence remains an important direction.

Bucket-level results show that module effects vary across ValScore levels. Regeneration gains concentrate in LOW, whereas Forward-Look gains are larger in MID and above.

7 Conclusion

We presented **ReflectRec**, an LLM recommendation pipeline organized around a cycle-consistency verifier. The verifier uses preference-to-history recoverability as an operational signal, gates regeneration, selects among regeneration attempts, and is used to analyze downstream module effects.

Preference recoverability and collaborative signals are complementary: low-recoverability preferences benefit from regeneration, while history-aligned retrieval provides collaborative cues that are difficult to obtain from the preference alone.

Limitations

Backbone-specific threshold. The operating threshold $\tau=0.4$ is calibrated for Qwen3-30B-A3B-Instruct. Table 8 confirms that the qualitative pattern (Full > Stage-1) holds for two additional backbones (Qwen2.5-72B, Llama-3.3-70B), but the optimal τ has not been retuned per backbone. The threshold also depends on verifier candidate-pool size and the positive-to-negative ratio, so it should be recalibrated when the verifier configuration changes.

Temporal validity. The primary evaluation follows user-wise chronological order but does not impose a shared production-time cutoff. In the strict timestamp-safe analysis, the Forward-Look gain on ML-1M is not statistically significant, motivating further evaluation of temporal robustness.

Forward-Look generalization. The main Forward-Look results are established on the movie datasets. On Amazon Electronics, direct raw-title

Forward-Look decreases performance, whereas candidate-aligned retrieval shows potential improvements. Further validation is therefore needed for cross-domain generalization of Forward-Look.

Stage-1 axis selection. The five-axis Stage-1 prompts used in this study (thematic for movies, spec-centric for electronics) were hand-designed per domain. An automatic axis-selection procedure that maps a domain’s item-side text to discriminative semantic axes would broaden its applicability to further text-heavy item domains (books, papers).

Constrained decoding. Adding prefix-constrained decoding to the candidate set is likely to close part of the gap to the trained baselines.

Ethical Considerations

We use only public datasets (MovieLens-100K, MovieLens-1M, and a public release of Amazon Reviews); no personally identifying information is processed. LLM inference is performed within the policy bounds of the used API endpoints. General recommendation: System risks (filter bubbles, bias amplification) are not directly addressed; however, the interpretable verification score is a step toward auditable intermediate signals.

Acknowledgments

This research was supported by the “Advanced GPU Utilization Support Program” funded by the Government of the Republic of Korea (Ministry of Science and ICT), and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2026-25616928, RS-2024-00436936, RS-2024-00439139). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (RS-2026-25617165 (MSIT), RS-2026-25501632 (MSIT and MOE)).

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. In *Proceedings of the 17th ACM conference on recommender systems*, pages 1007–1014.
- Hongyi James Cai, Junlin Wang, Xiaoyin Chen, and Bhuwan Dhingra. 2025. How Much Backtracking is

- Enough? Exploring the Interplay of SFT and RL in Enhancing LLM Reasoning. <https://arxiv.org/abs/2505.24273>.
- Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. <https://arxiv.org/abs/2303.14524>.
- Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*, pages 299–315.
- F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):1–19.
- Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th international conference on world wide web*, pages 507–517.
- Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. <https://arxiv.org/abs/1511.06939>.
- Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *Proceedings of the 46th European Conference on Information Retrieval*, pages 364–381. Springer.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.
- Syjung Hwang and Eunil Park. 2021. Movie recommendation systems using actor-based matrix computations in south korea. *IEEE Transactions on Computational Social Systems*, 9(5):1387–1393.
- Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE.
- Sein Kim, Hongseok Kang, Kibum Kim, Jiwan Kim, Donghyun Kim, Minchul Yang, Kwangjin Oh, Julian McAuley, and Chanyoung Park. 2025. Lost in sequence: Do large language models understand sequential recommendation? In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 1160–1171.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 9459–9474.
- Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llra: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1785–1795.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.
- Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.
- Sichun Luo, Jian Xu, Xiaojie Zhang, Linrong Wang, Sicong Liu, Hanxu Hou, and Linqi Song. 2025. Rall-rec+: Retrieval augmented large language model recommendation with reasoning. *Expert Systems with Applications*, 297:129508.
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461.
- Jianling Wang, Haokai Lu, James Caverlee, Ed H Chi, and Minmin Chen. 2024. Large language models as data augmenters for cold-start item recommendation. In *Companion Proceedings of the ACM Web Conference 2024*, pages 726–729.
- Lei Wang and Ee-Peng Lim. 2024. The whole is better than the sum: Using aggregated demonstrations in in-context learning for sequential recommendation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 876–895.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 24824–24837.
- Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, and 1 others. 2024. A survey on large language models for recommendation. *World Wide Web*, 27(5):60.

- Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 12–22.
- Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muenighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.
- Weiqi Yue, Yuyu Yin, Xin Zhang, Binbin Shi, Tingting Liang, and Jian Wan. 2025. Cot4rec: Revealing user preferences through chain of thought for recommender systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 13142–13151.
- Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2329–2340.
- Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *Proceedings of the 2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1435–1448. IEEE.

A Appendix: Threshold Sweep

We select the conditional-regeneration operating threshold $\tau=0.4$ using a single-seed sweep under the fixed verifier design. This threshold sweep uses $T=0.7$ and is separate from the 8-seed regeneration analysis in Section 3.4. Per-dataset results are reported in Table 11.

τ	$n_{<\tau}$	$\Delta\text{Hit@10}$	95% CI
0.3	32	-9.38	[-21.88, +0.00]
0.4	116	+6.90*	[+0.00, +13.79]
0.5	345	+1.45	[-2.32, +5.22]
0.6	684	+0.29	[-2.19, +2.78]
0.7	1171	+0.09	[-1.96, +2.13]

Table 10: τ sweep for LOW-bucket ($v_u < \tau$) conditional regeneration on the unified pool (ML-100K + ML-1M), single-seed at $T=0.7$ ($B=10,000$ paired bootstrap). The starred entry marks the operating threshold $\tau=0.4$. Per-dataset breakdown is in Appendix Table 11.

B Appendix: Per-dataset τ Sweep Detail

Table 11 decomposes the unified-pool sweep in Table 10 into ML-100K and ML-1M components. At $\tau=0.4$, the unified +6.90 pp is the weighted average of ML-100K +13.79 pp ($n=29$) and ML-1M +4.60 pp ($n=87$); both per-dataset point estimates are positive while their bootstrap CIs individually include zero, so the significance on the unified pool reflects a consistent effect direction.

Dataset	τ	$n_{<\tau}$	$\Delta\text{Hit@10}$	95% CI
ML-100K	0.3	4	+0.00	[+0.00, +0.00]
	0.4	29	+13.79	[+0.00, +31.03]
	0.5	113	+0.88	[-5.31, +7.08]
	0.6	239	+0.42	[-3.35, +4.60]
	0.7	414	+0.24	[-3.14, +3.62]
ML-1M	0.3	28	-10.71	[-25.00, +0.00]
	0.4	87	+4.60	[-2.30, +11.49]
	0.5	232	+1.72	[-3.02, +6.47]
	0.6	445	+0.22	[-2.92, +3.37]
	0.7	757	+0.00	[-2.65, +2.64]

Table 11: Per-dataset LOW-bucket τ sweep. $B=10000$ paired bootstrap. Unified-pool result is in Table 10.

C Appendix: Forward-Look Per-Bucket Bootstrap

Table 12 reports bootstrap confidence intervals for the per bucket forward-look contribution ($\Delta\text{Hit@10}$, Full vs. Stage-1+Regen).

Bucket	Dataset	n	$\Delta\text{Hit@10}$	95% CI
Overall	ML-100K	843	+3.32*	[+0.36, +6.52]
	ML-1M	1500	+3.53**	[+1.13, +4.33]
LOW	ML-100K	29	+0.00	[-13.79, +13.79]
	ML-1M	87	+1.15	[-9.20, +11.49]
MID	ML-100K	84	+2.38	[-8.33, +13.10]
	ML-1M	145	+4.14	[-2.76, +11.03]
HIGH	ML-100K	301	+3.32	[-1.99, +8.64]
	ML-1M	525	+4.76*	[+0.57, +8.95]
PASS	ML-100K	429	+3.73	[-1.17, +8.39]
	ML-1M	743	+2.83	[-0.67, +6.33]

Table 12: Per-dataset and per-bucket bootstrap detail of forward-look $\Delta\text{Hit@10}$ ($B=10000$). * $p < 0.05$, ** $p < 0.01$. On the unified pool ($n=2343$), the overall $\Delta = +3.54$ pp, CI [+1.62, +5.46], $p = 0.0001$.

D Appendix: Per-Bucket Backbone Robustness

Table 13 reports the per-bucket $\Delta\text{Hit@10}$ of the full pipeline versus the Stage-1 baseline across the three backbones.

Backbone	LOW	MID	HIGH	PASS
Qwen3-30B	+13.79	+2.38	+3.32	+3.73
Qwen2.5-72B	+8.41	+13.95	+11.46	+10.21
Llama-3.3-70B	+7.45	+7.14	+7.07	+0.00

Table 13: Per-bucket $\Delta\text{Hit@10}$ (full pipeline vs. Stage-1 baseline) across the three backbones. All cells are non-negative; only the Llama-3.3-70B PASS bucket equals the Stage-1 baseline exactly (saturation at the high baseline of 38.03%). The full pipeline contributes non-trivially across all ValScore buckets for all three backbones.

E Appendix: Negative Construction and Pool Sensitivity

The original verifier uses 10 history items and 40 catalog-wide unobserved distractors, without separately restricting distractor availability at query time. To examine sensitivity to negative construction, we compare two structured 50-item conditions. Both exclude history and evaluation targets; Arm A samples distractors from the remaining catalog, while Arm B restricts distractors to items available by the query cutoff. Availability is determined using release date for ML-100K and title year for ML-1M.

Mean ValScore remains similar across the two datasets, while LOW routing membership shows some sensitivity to negative construction on ML-100K. When Arm-B scores are used to reroute re-

Data	n	Mean A/B	B-A [95% CI]	r	LOW J.
ML-100K	668	.4260/.4331	.0070 [-.0034, .0174]	.5944	.4214
ML-1M	1,500	.5185/.5187	.0003 [-.0027, .0033]	.9399	.7941

Table 14: Verifier sensitivity between catalog-wide Arm A and time-available Arm B. r is Pearson correlation and LOW J. is the Jaccard overlap of $V < .4$ membership.

generation, Hit@10 relative to Stage-1 changes by $+0.30 / +0.40$ pp on ML-100K / ML-1M, with 95% CIs of $[-0.90, +1.50]$ and $[-0.47, +1.27]$.

Data	30 items	50 items	100 items
ML-100K	.5390	.4310	.3110
ML-1M	.6220	.5343	.4133

Table 15: Mean Arm-B ValScore by candidate-pool size.

Candidate-pool size sensitivity. Mean ValScore decreases as the candidate pool grows. Thus, $\tau = 0.4$ is an operating threshold that depends on the candidate-pool construction and should be recalibrated when the verifier configuration changes.

F Appendix: Classical kNN Sensitivity

User-kNN and Item-kNN use cosine similarity over the explicit 1–5 rating matrix. Each evaluation user’s row ends at the 10-item query point, excluding evaluation targets and later interactions. We report all tested k settings for sensitivity analysis.

Data	Method	k	H@1	H@5	H@10
ML-100K	User-kNN	20	4.86	20.28	38.91
	User-kNN	40	3.32	14.71	29.42
	User-kNN	80	1.54	10.79	22.89
	Item-kNN	5	1.30	4.98	9.25
	Item-kNN	10	.83	4.51	8.42
ML-1M	User-kNN	20	1.73	10.60	21.40
	User-kNN	40	1.20	7.60	14.93
	User-kNN	80	.67	4.00	8.13
	Item-kNN	5	1.33	4.60	7.93
	Item-kNN	10	.33	.73	1.13

Table 16: Full-pool same-candidate classical collaborative baselines (%).

Classical kNN performance varies by dataset and k : User-kNN is relatively strong on ML-100K, whereas its effect is more limited on ML-1M.

G Appendix: Logged Qualitative Cases

The following examples are selected from actual evaluation logs and include sample indices. They

illustrate the behavior of the verifier and regeneration and are not intended to represent average performance.

High-recoverability case — ML-100K, sample 70. The history includes *Searching for Bobby Fischer*, *Casablanca*, *Field of Dreams*, *Fargo*, and *The Shawshank Redemption*. The generated preference note captures emotionally resonant and character-driven narratives, moral depth, redemption, and restrained performances, with ValScore 0.9. Stage-1 and Forward-Look both predict *The Remains of the Day*, and Hit@10 remains $1 \rightarrow 1$.

Successful regeneration — ML-100K, sample 204. The history includes *The English Patient*, *Tomorrow Never Dies*, *Contact*, *Air Force One*, and *Liar Liar*. The initial preference has ValScore 0.3. After regeneration, characteristics such as a drama–suspense mixture, high-stakes personal crisis, and interpersonal tension become more specific, increasing ValScore to 0.6. The prediction changes from *Sophie’s Choice* to *As Good as It Gets*, and Hit@10 improves from $0 \rightarrow 1$.

Counterexample — ML-100K, sample 529. The history includes *Scream*, *Evita*, *Lost Highway*, *Crash*, *187*, and *Volcano*. After regeneration, features such as psychological thriller, moral ambiguity, and atmospheric tone become more specific, increasing ValScore from $0.3 \rightarrow 0.4$. The prediction changes from *Dead Man* to *Chinatown*, while Hit@10 decreases from $1 \rightarrow 0$.

H Appendix: ValScore and Recommendation Correctness

To examine the association between ValScore and Stage-2 recommendation correctness, we report Hit@10 by initial ValScore bucket and the sample-level Pearson correlation.

ValScore bucket	ML-100K Hit@10	ML-1M Hit@10
LOW [0,.4)	20.69 ($n = 29$)	24.14 ($n = 87$)
MID [.4,.5)	26.19 ($n = 84$)	25.52 ($n = 145$)
HIGH [.5,.7)	24.58 ($n = 301$)	35.43 ($n = 525$)
PASS [.7,1]	30.77 ($n = 429$)	40.92 ($n = 743$)
Pearson r	.044 [-.022, .110]	.115 [.065, .165]

The sample-level Pearson correlation between ValScore and Stage-2 correctness is $r = .044$ on ML-100K (95% CI $[-.022, .110]$) and $r = .115$ on ML-1M (95% CI $[.065, .165]$).

Decoder	ML-100K	ML-1M
Global popularity	32.38	30.80
Query-time-safe popularity	37.13	30.13
Forward-Look retrieval-only	42.70	45.40
Timestamp-safe Forward-Look retrieval-only	40.69	42.87

Table 17: Hit@10 (%) for popularity and Forward-Look retrieval-only baselines.

I Appendix: Timestamp-Safe Diagnostic

In the strict timestamp-safe subset, Forward-Look improves Hit@10 over Stage-1+Regeneration by +5.14 pp on ML-100K ($n = 564$, 95% CI [+1.06, +9.22]). On ML-1M, the gain is +0.67 pp ($n = 1,487$, 95% CI [-1.75, +3.09]).

J Appendix: Popularity-Only Baseline Analysis

Forward-Look retrieval-only achieves higher Hit@10 than both global popularity and query-time-safe popularity baselines on both datasets.

K Appendix: Prompt Templates

This appendix provides the full system / user prompts used at every LLM call in the pipeline. {...} denotes runtime-filled placeholders.

Stage-1 semantic preference generation (§3.2).

system: You are a helpful assistant that analyzes a user’s movie preferences based on their watching history. Write a concise preference description in about 60–100 words.

CRITICAL CONSTRAINT – do NOT name or quote any specific movie title from the user’s history. Describe taste only in terms of: genre / sub-genre, mood and emotional tone, thematic content, era or stylistic period, narrative style. Mentioning a specific title from the history will be considered a failure.

user: The user’s previous order of watching movies and their ratings were: {history_str}. Please analyze his preference characteristics based on the previous behavior.

Stage-1 semantic preference — Electronics adaptation (§5.3). The Electronics prompt replaces the five thematic axes with five spec-centric axes; the preamble and constraint structure are otherwise identical.

system: You are a helpful assistant that analyzes a user’s product preferences based on their purchase history. Write a

concise preference description in about 60–100 words.

CRITICAL CONSTRAINT – do NOT name or quote any specific product title from the user’s history. Describe the user’s preference along five concrete technical axes: (1) technical specifications (resolution / capacity / speed / wattage / connectivity), (2) industry standards and compatibility (USB-C, HDMI, IEEE, etc.), (3) form factor (size, weight, portability), (4) brand family or product line, and (5) quality tier. Be precise and concrete about the technical characteristics that unify their purchases. Avoid generic adjectives.

user: The user’s previous order of purchasing products and their ratings were: {history_str}. Please analyze his preference characteristics based on the previous behavior.

Validator: cycle-consistency verification (§3.3).

system: You are a movie analyst. Given a user preference description, identify which movies from a list the user has likely watched.

user: A user has the following preference characteristics: {preference}

From the movies listed below, select the ones this user has most likely already watched (based on the preference description). Respond with ONLY a comma-separated list of movie titles, nothing else.

Movies:

- {candidate_1}
- {candidate_2}
- ...
- {candidate_50}

The candidate set comprises the user’s true history items ($|H_u| = 10$) plus 40 random negatives, shuffled. More precisely, the original sampler uses 40 distinct catalog-wide unobserved distractors, does not restrict query-time availability, and does not explicitly remove later evaluation targets. Sensitivity to this construction is analyzed in Appendix E.

Stage-2 Ranker (with Forward-Look hint, §3.5).

system: You are a movie recommendation assistant. Use (1) the user’s history, (2) their preference description, AND (3) ‘next-watch hints’ showing what other users with similar viewing patterns watched NEXT – these are candidates from a collaborative-filtering signal. Pay attention to them. Respond with ONLY the title of one movie from the candidate list.

user: The user’s previous order of watching movies and their ratings were:

```

{history_str}
His Personal preferences: {preference}
Users with similar viewing patterns next
watched:
  (genre) {future_titles_fragment_1}
  (genre) {future_titles_fragment_2}
  . . .
  (genre) {future_titles_fragment_k}
The movies available for selection
include: {candidate_1} ; {candidate_2}
; . . . ; {candidate_100} .
Based on his viewing history and
preferences, please recommend a movie
for him.

```

Stage-2 Ranker (without Forward-Look hint — the A/C ablation conditions). The hint block is removed and only the system prompt is replaced by:

```

system: You are a movie recommendation
assistant. Given a user's history, their
preference description, and a candidate
list, respond with ONLY the title of the
single best movie to recommend from the
candidate list.

```

Conditional Regen (§3.4). For regeneration, the same Stage-1 prompt is re-called at sampling temperature $T=0.5$ up to $m=2$ times. Each attempt is scored by the Validator prompt above, and the highest-scoring attempt becomes P'_u .