

LUPE: Long-History User Profile Evolution for LLM-based Recommendation

Yifei Sun¹, Lu Wang², Fangkai Yang², Pu Zhao², Liqun Li²,
Mingzheng Li², Jianjin Zhang², Jimeng Sun², Zhiwei Dai², Nan Hu²,
Zhengxin Zeng², Yuefeng Zhan², Weihao Han², Qingwei Lin²,
Saravan Rajmohan², Weiwei Deng², Feng Sun², Qi Zhang², Dongmei Zhang²

¹Peking University ²Microsoft

Correspondence: yifeisun0710@gmail.com

Abstract

Textual user profiling is an effective paradigm for LLM-based recommendation, but existing methods usually treat profile generation as a one-shot summarization problem. This limitation becomes more severe in long-history recommendation, where early preference signals may be overcompressed and recent behaviors may be overemphasized. We propose **LUPE**, a Long-history User Profile Evolution framework for LLM-based recommendation. Instead of generating a static profile from the full history, LUPE maintains a persistent textual profile and updates it progressively over temporally ordered interaction chunks. This allows the model to preserve long-term preferences while incorporating new behavioral evidence. To learn how profiles should evolve, LUPE optimizes a profile evolution policy via reinforcement learning using feedback from a frozen downstream LLM-based recommender. This allows the model to directly align profile updates with recommendation utility rather than relying on heuristic or supervised profile targets. Experiments on Yelp, Amazon CDs and Vinyl, and Amazon Books show that LUPE consistently outperforms strong baselines on both rating prediction and ranking tasks. Further analyses show that explicit profile evolution is especially helpful for users with long and heterogeneous interaction histories. The code repository: https://github.com/LongHistoryProfile/long_history_profile.

1 Introduction

Large language models (LLMs) have recently become a promising paradigm for recommender systems due to their strong semantic understanding and reasoning ability (Wang and Lim, 2023; Gao et al., 2023; Bao et al., 2023; Xi et al., 2024; Ren et al., 2024; Jiang et al., 2025). Compared with traditional ID-based methods, LLM-based recommendation represents user preferences in natural

language, providing a more expressive and interpretable basis for personalization. Prior work has explored this direction through instruction tuning, knowledge augmentation, representation learning, and reinforcement learning-based alignment (Bao et al., 2023; Jiang et al., 2025; Xi et al., 2024; Ren et al., 2024; Gao et al., 2025; Lin et al., 2025; Fang et al., 2025). Among these approaches, textual user profiling has emerged as an effective way to connect LLMs with personalized recommendation. Instead of directly feeding full interaction histories into the model, user profiling summarizes preference signals into a compact textual representation, improving both efficiency and interpretability (Zhang, 2024; Wang et al., 2025). Prior studies show that such profiles can substantially improve personalization quality (Zhang, 2024; Wang et al., 2025).

However, long-history user modeling remains a key challenge for LLM-based recommendation. In real-world systems, active users often accumulate hundreds of interactions over extended periods, making it difficult to encode the entire history within a limited textual context. Long contexts introduce redundancy and noise, weaken important signals, and make it harder to identify which past behaviors remain relevant (Xia et al., 2025; Zhu et al., 2024; Lin et al., 2024). Even with extended context windows, full-history prompting becomes increasingly costly and difficult to scale across users.

This issue is evident in real-world datasets. For example, in Amazon Books 2023 (Hou et al., 2024), 8,080 users have histories longer than 32k tokens and 1,333 users exceed 128k tokens, making full-history prompting impractical for large-scale deployment. These statistics suggest that full-history prompting can quickly become costly, making budget-efficient serving and broad deployment increasingly difficult as user histories grow longer.

This challenge is further compounded by the

temporal nature of user preferences. User interests are not static: some preferences remain stable over time, while others gradually weaken or newly emerge (Zhou et al., 2018, 2019; Pi et al., 2020). Therefore, long-history recommendation is not only a context-length problem, but also a temporal modeling problem. In this setting, it is insufficient to merely compress the full history once; instead, the model should preserve enduring preference signals while continuously incorporating new behavioral evidence.

Existing methods only partially address this challenge. LLM-based recommendation methods mainly improve task alignment or profile generation (Bao et al., 2023; Jiang et al., 2025; Gao et al., 2025; Lin et al., 2025; Yang et al., 2023; Zhang, 2024; Wang et al., 2025; Fang et al., 2025), while lifelong behavior modeling methods focus on handling long sequential histories through hierarchical search, retrieval, or incremental inference (Xia et al., 2025; Zhu et al., 2024; Lin et al., 2024). Although some prior work leverages downstream recommendation signals to improve profile construction, most existing approaches still treat user representation as a static or one-shot summarization process. As a result, they do not explicitly model how a user profile should be progressively updated as new behaviors arrive over long histories, nor how such updates should be optimized for downstream recommendation utility.

In this work, we formulate long-history user modeling for LLM-based recommendation as a *profile evolution* problem. Rather than viewing a user profile as a one-shot summary, we treat it as a persistent state that continuously evolves by integrating new behavioral evidence while preserving stable long-term preferences. This perspective shifts user modeling from static summarization to a sequential decision process over evolving representations.

Based on this idea, we propose **LUPE**, a Long-history User Profile Evolution framework for LLM-based recommendation. LUPE processes temporally ordered behavior segments and incrementally updates the user profile in a stage-wise manner. Instead of relying on predefined update rules or supervised profile targets, LUPE learns a *profile evolution policy* directly from downstream recommendation feedback via reinforcement learning. This enables the model to automatically balance historical consistency with responsiveness to recent user behavior, and to optimize profiles for

decision-relevant downstream utility. The resulting evolved profile serves as a compact yet temporally-aware representation that better preserves preference signals relevant for recommendation. Experimental results show that LUPE achieves strong and consistent performance across multiple interaction-based recommendation benchmarks, with particularly clear gains in ranking and accuracy metrics. Further analyses demonstrate that explicitly modeling profile evolution and learning it from downstream feedback is especially beneficial for users with long and heterogeneous interaction histories. Our main contributions are summarized as follows:

- We introduce *profile evolution* as a principled formulation for long-history LLM-based recommendation, reframing user modeling as a sequential state update process rather than static profile summarization.
- We propose **LUPE**, a reinforcement learning framework that learns a profile evolution policy to update textual user representations from downstream recommendation feedback, enabling adaptive and utility-driven user modeling.
- We empirically demonstrate that LUPE consistently achieves strong performance on long-history recommendation benchmarks, and provide analysis showing its effectiveness for modeling complex and heterogeneous user behavior.

2 Related Work

2.1 Long-History User Modeling for Recommendation

Modeling long user histories is a central challenge in recommendation. Early sequential methods such as DIN (Zhou et al., 2018), DIEN (Zhou et al., 2019), and SIM (Pi et al., 2020) model target-aware interests and temporal dynamics, but mainly operate on ID-based behavior sequences. More recent LLM-based methods, including Rella (Lin et al., 2024), LIBER (Zhu et al., 2024), and HiTLBM (Xia et al., 2025), introduce retrieval, summarization, or hierarchical chunking to improve long-sequence behavior understanding. PersonaX (Shi et al., 2025), PURE (Bang and Song, 2025), and Locate-Forget-Update (Liu et al., 2025a) further consider segmented histories, continuous profile maintenance, or representation adaptation under preference shifts. However, these methods mainly

focus on behavior understanding, profile maintenance, or representation updating, rather than explicitly modeling how a textual user profile should be progressively updated over long interaction histories.

2.2 Profiles in Recommendation

User profiles are a classical tool for personalized recommendation. Early methods relied on structured or hand-crafted profile attributes, such as CRESUP (Chen et al., 2007) and UPC-Sim (Widiyaningtyas et al., 2021). Recent LLM-based approaches move toward natural-language profile generation, including PALR (Yang et al., 2023), KAR (Xi et al., 2024), Guided Profile Generation (Zhang, 2024), and End-to-end Training for Recommendation with Language-based User Profiles (Gao et al., 2024). These methods highlight the value of semantic user representations, but mainly focus on profile construction or optimization rather than profile updating under long and evolving histories.

2.3 Reinforcement Learning for LLM-based Recommendation

Recent studies also explore reinforcement learning to align LLM-based recommendation with downstream recommendation objectives. LettinGo (Wang et al., 2025) improves user profile generation with downstream feedback, while ReCLM (Jiang et al., 2025), Rec-R1 (Lin et al., 2025), and SPRec (Gao et al., 2025) apply reinforcement learning to recommendation instruction tuning, closed-loop policy learning, and self-play-based debiasing, respectively. DUET (Chen et al., 2026) further incorporates reinforcement learning into joint user-item profile optimization. These methods demonstrate the effectiveness of downstream RL supervision, but focus on optimizing profile generation or recommendation behavior rather than the progressive evolution of user profiles over long histories. In contrast, our method treats profile updating as a sequential evolution process, progressively refining a textual user profile over history chunks and optimizing this process with GRPO.

3 Method

3.1 Problem Setup

For each user u , let the chronological interaction history be

$$\mathcal{H}_u = \{r_1, r_2, \dots, r_T\},$$

where r_t denotes the t -th review interaction containing review text and rating. For notational simplicity, we omit the user index u , and all variables are defined with respect to an individual user.

To model long histories progressively, the sequence is partitioned into chronological interaction chunks, each corresponding to one stage of profile updating:

$$R_s = \{r_{(s-1)K+1}, \dots, r_{sK}\},$$

where K is the chunk size. At stage s , the model maintains a textual profile state P_{s-1} and updates it as

$$P_s = f_\theta(P_{s-1}, R_s).$$

To evaluate profile quality, each chunk is associated with the next interaction. If R_s ends at step sK , the target item i_s and rating y_s correspond to interaction r_{sK+1} . The evolved profile P_s together with the target item is used by a downstream predictor to estimate $\hat{y}_s$, which provides the reward signal for training. The goal is to learn a profile evolution policy that progressively updates user profiles to better support downstream recommendation.

3.2 Long-history User Profile Evolution

Figure 1 illustrates the overall framework of LUPE. We formulate long-history user modeling as a *textual profile evolution problem*. Instead of generating a static profile from the entire interaction history, LUPE maintains a persistent textual profile that evolves as new behavioral evidence arrives. This recursive formulation allows the model to capture long interaction histories while maintaining a compact representation of user preferences.

At stage s , the model receives the existing profile state P_{s-1} together with the newly observed interaction chunk R_s and generates an updated profile $P_s = f_\theta(P_{s-1}, R_s)$, where f_θ denotes the profile evolution model. At each stage, the model updates the existing profile by integrating newly observed interactions, allowing the profile to progressively accumulate long-term preference signals while incorporating recent behavioral evidence. Conceptually, the profile functions as a persistent preference state whose evolution is modeled in natural language space. Figure 2 shows an example of a single profile evolution process.

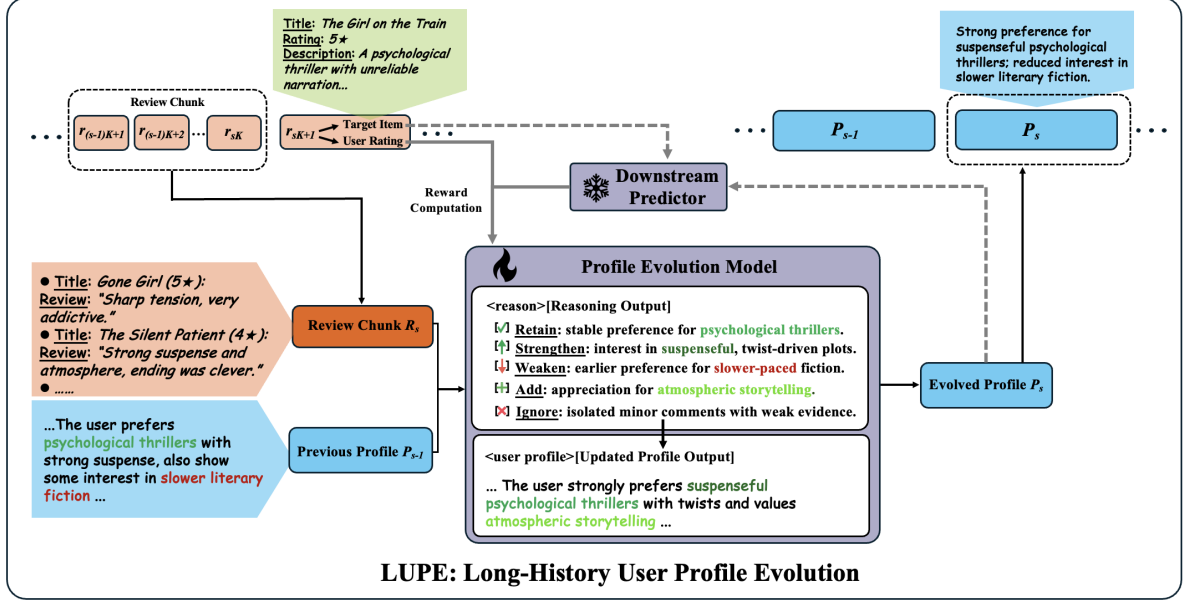


Figure 1: Overview of LUPE. Given a long interaction history, LUPE partitions the sequence into chronological interaction chunks. The user profile is treated as a persistent textual state that evolves sequentially over these chunks. At each stage, the profile evolution model generates candidate profile updates, which are evaluated by a frozen downstream predictor to produce reward signals for reinforcement learning.

A key challenge is determining how the profile should evolve when new behavioral evidence arrives. Since there is no ground-truth target profile and surface-level textual similarity does not necessarily reflect recommendation utility, LUPE learns the profile evolution policy directly from downstream recommendation feedback via reinforcement learning.

To provide such feedback, we employ a frozen LLM-based recommendation model, referred to as the *downstream predictor*, instantiated using Qwen3-8B in an instruction-following setting. Given a user profile and a target item description, the model predicts the rating that the user would assign to the item and outputs it in a structured `<rating>` format. The predictor operates in a zero-shot manner without task-specific fine-tuning, and item descriptions are constructed from available metadata such as titles and textual attributes. To ensure numerical stability, we constrain the output format and enforce two-decimal precision. We empirically observe that the predictor produces stable and consistent outputs under a fixed prompt template, enabling reliable rewards during training. Details of the predictor are provided in Appendix A.2.

Specifically, for each training instance, the model samples a group of candidate updated profiles

$$\{\tilde{P}_s^{(g)}\}_{g=1}^G \sim f_\theta(P_{s-1}, R_s),$$

where G denotes the group size. Each candidate profile is evaluated by the downstream predictor

$$\hat{y}_s^{(g)} = g_\phi(\tilde{P}_s^{(g)}, i_s),$$

which estimates the rating that the user would assign to the target item.

We define the reward based on rating prediction accuracy

$$r_s^{(g)} = 1 - \frac{|y_s - \hat{y}_s^{(g)}|}{M},$$

where y_s is the ground-truth rating and M denotes the maximum rating gap (e.g., $M = 4$ for a 1–5 rating scale). This formulation yields rewards in the range $[0, 1]$, where higher values correspond to more accurate predictions.

Although the same downstream predictor is used for both training and evaluation, it remains strictly frozen throughout the entire process. The evolution model only observes scalar reward signals and has no access to the predictor’s internal states or gradients, preventing direct overfitting to the evaluator. Instead, it must improve the informativeness of the generated profiles so that they better support downstream recommendation. Moreover, since the predictor operates solely on the textual profile and item description, useful preference signals must be explicitly encoded in the profile rather than relying on prompt-specific artifacts.

Prompted State-Update Input

Task

First, explain what is retained, strengthened, weakened, added, and ignored. Then update the user’s preference profile by integrating recent reviews with the existing long-term profile.

Existing User Profile (long-term)

{...The user prefers psychological thrillers with strong suspense and tension. They also show some interest in slower literary fiction. Their ratings are often high for immersive and emotionally engaging books...}

Recent User Reviews (most recent last)

```
{  
  Gone Girl (5★): “Sharp tension, very addictive.”  
  The Silent Patient (4★): “Strong suspense and atmosphere,  
  ending was clever.”  
  The Girl on the Train (5★): “Twisty, tense, and hard to put  
  down.”  
  .....  
}
```

↓ LUPE Profile Evolution Model ↓

Structured State-Update Output

<reason>

Retain: stable preference for psychological thrillers.
Strengthen: interest in suspenseful, twist-driven plots.
Weaken: earlier tendency toward slower literary fiction.
Add: appreciation for atmospheric storytelling.
Ignore: isolated weak details with little cross-review support.
</reason>

<user_profile>

The user strongly prefers psychological thrillers with suspenseful, twist-driven plots. They value tension, immersive pacing, and atmospheric storytelling. Interest in slower literary fiction appears less prominent in recent behavior.
</user_profile>

Figure 2: A single profile evolution step in LUPE. The model updates the existing textual profile using a newly observed review chunk through explicit reasoning.

Following GRPO, rewards within each candidate group are normalized to compute advantages, and the evolution policy is updated accordingly. This continuous reward signal enables the model to capture incremental improvements in profile quality and supports stable learning of the profile evolution policy. See Appendix A.1 for the full training algorithm.

4 Experiment

4.1 Experimental Settings

Datasets. We conduct experiments on three review-based recommendation datasets: *Amazon Books*, *Amazon CDs and Vinyl*, and *Yelp*. For Amazon, we use the Amazon Reviews 2023¹ (Hou et al.,

Dataset	Split	#User	#Review	Len Range	Avg Len
Yelp	Train (ID)	3,798	656,346	[100, 499]	172.81
	Test (ID)	422	72,734	[100, 490]	172.36
	Test (OOD)	145	111,068	[502, 3048]	765.99
Amazon CDs and Vinyl	Train (ID)	920	153,818	[100, 497]	167.19
	Test (ID)	102	16,736	[102, 401]	164.08
	Test (OOD)	34	24,387	[513, 1912]	717.26
Amazon Books	Train (ID)	5,283	928,799	[100, 499]	175.81
	Test (ID)	587	105,036	[100, 493]	178.94
	Test (OOD)	238	179,781	[500, 3134]	755.38

Table 1: Dataset statistics for the in-distribution (ID) and out-of-distribution (OOD) settings.

2024) 5-core Books and CDs and Vinyl subsets. For Yelp, we use the Yelp Open Dataset.² To study long-history recommendation, we retain users with at least 100 interactions. Users with history lengths in [100, 500) form the in-distribution (ID) setting, where users are split into training and test sets with a 9:1 ratio at the user level. We further construct an out-of-distribution (OOD) test set using users with at least 500 interactions, enabling evaluation on much longer histories. Detailed dataset statistics are reported in Table 1.

Evaluation Protocol and Metrics. We evaluate all methods under a rolling user-history protocol. For each test user, evaluation instances are constructed every 10 interactions in chronological order. At each step, only interactions before the target are used as history. Based on the observed history, each method builds a user profile, which is fed into the downstream predictor to estimate the target rating; the predicted score is then used for ranking.

For rating prediction, we report three widely adopted metrics (Wang et al., 2025; Fang et al., 2025): *MAE*, *RMSE*, and *Accuracy*. Since the predictor outputs continuous ratings with two-decimal precision, we define Accuracy with a tolerance threshold: a prediction is correct if its absolute error is no greater than 0.5.

For ranking evaluation, each target interaction is paired with a candidate set containing the ground-truth item and 9 negative items, following recent LLM-based recommendation studies such as AgentCF (Zhang et al., 2024) and AgentCF++ (Liu et al., 2025b). The negatives are sampled from a hard-negative pool constructed by EASE (Steck, 2019). Candidate items are ranked by predicted scores, and we report *NDCG@1/5/10*. This protocol serves as a controlled ranking evaluation

¹<https://amazon-reviews-2023.github.io/>

²<https://business.yelp.com/data/resources/open-dataset/>

for comparing profile construction methods under the same candidate set, while Appendix B.2 further reports an industrial-style top-100 retrieval-then-reranking evaluation using Swing (Yang et al., 2020).

Baselines. We compare with representative baselines covering direct history modeling, LLM-based representation learning, profile generation, and explicit reasoning. *All History Rating* uses the full interaction history without explicit profiling. *KAR* (Xi et al., 2024) incorporates LLM-extracted user/item knowledge, while *RLMRec* (Ren et al., 2024) learns semantic representations from reviews and aligns them with collaborative signals. *PALR* (Yang et al., 2023) formulates recommendation as language-based ranking; *LettinGo* (Wang et al., 2025) optimizes profiles generated from recent interactions; and *Reason4Rec* (Fang et al., 2025) performs multi-step reasoning over user history. All methods use the same rolling protocol with their original history settings: full history for *All History Rating*, *KAR*, *RLMRec*, and *PALR*, recent 70 interactions for *LettinGo*, and recent 10 interactions for *Reason4Rec*.

4.2 Main Results

Table 2 presents the main results of our method and the baselines on in-distribution datasets. We use Qwen3-8B (Team, 2025) as the backbone model for profile generation and downstream prediction. Detailed GRPO training settings are summarized in Appendix C.1.

Strong overall results. LUPE demonstrates strong performance across all datasets, with particularly consistent gains in Accuracy and ranking metrics. On the Book dataset, LUPE achieves 55.98% Accuracy, surpassing LettinGo (Wang et al., 2025) by 5.60 points. On Yelp, it further improves Accuracy to 40.08%, again outperforming the strongest baselines by clear margins. On Amazon CDs and Vinyl, although LettinGo achieves slightly better MAE and RMSE, LUPE attains the highest Accuracy and competitive ranking performance, indicating improved utility for recommendation scenarios.

Better than structured and reasoning-based baselines. Compared with representative baselines such as *KAR* (Xi et al., 2024) and *Reason4Rec* (Fang et al., 2025), our method consistently provides stronger prediction results. On Book, for example, our method reduces MAE to 0.5525, compared with 0.7752 for *KAR* and 0.7036 for *Reason4Rec*, while also achieving the best Ac-

curacy. This suggests that our method captures user preference signals more effectively than approaches based on predefined knowledge structures or explicit multi-step reasoning.

Clear advantages in ranking. The improvement is even more evident in downstream ranking. Our method achieves the best NDCG@1/5/10 on all three datasets, with the largest gains appearing on Book. For instance, NDCG@1 improves from 0.3566 with LettinGo to 0.5962, and NDCG@10 rises from 0.6122 to 0.7704. These results indicate that the learned profiles are more effective at distinguishing the true target item from hard negatives, leading to better ranking quality in downstream recommendation. Appendix B.2 further shows that LUPE remains effective in an industrial-style top-100 retrieval-then-reranking setting using Swing (Yang et al., 2020).

4.3 Ablation Study

Effect of reinforcement learning. We compare the full model with w/o RL, which removes reward-based optimization. As shown in Table 3, reinforcement learning consistently improves downstream performance on all three datasets, especially in Accuracy and ranking metrics. The gain is particularly clear on Amazon CDs and Vinyl and Amazon Books, where Accuracy improves by more than 10 points, while Yelp also shows a notable increase from 35.25% to 40.08%. Similar trends are observed on ranking performance.

Effect of Backbone Scale. We further analyze the effect of model scale on the CD dataset by applying our method to Qwen3-4B, Qwen3-8B, and Qwen3-14B. Table 4 shows that backbone scale affects downstream performance, with larger models generally providing stronger profile evolution and reasoning ability. The improvements, however, are not strictly linear, suggesting diminishing returns at higher scales.

Effect of history chunk size. We evaluate chunk sizes of 5, 10, and 20 on Amazon CDs and Vinyl, with each model trained under its corresponding chunk size and tested under the same protocol. Table 5 shows that chunk size 5 achieves the best Accuracy and NDCG@1/5/10, indicating that more frequent updates better preserve decision-relevant preference signals. In contrast, chunk size 20 performs worse, suggesting that overly coarse updates weaken profile evolution.

Cross-evaluator robustness. We evaluate LUPE with a different frozen predictor, Llama-3 8B, on

Method	Yelp						Amazon CDs and Vinyl						Amazon Books					
	MAE	RMSE	Acc (%)	N@1	N@5	N@10	MAE	RMSE	Acc (%)	N@1	N@5	N@10	MAE	RMSE	Acc (%)	N@1	N@5	N@10
All History Rating	1.1650	1.4758	30.36	0.1223	0.2379	0.4324	0.9835	1.3666	27.90	0.1542	0.3509	0.4975	1.0263	1.3132	20.79	0.1530	0.3507	0.4927
KAR	0.8593	1.1032	34.33	0.1983	0.3396	0.4638	0.7456	1.0740	37.91	0.1244	0.3117	0.4680	0.7373	0.9531	40.24	0.1786	0.4065	0.5356
RLMRec	0.9536	1.2214	30.04	0.1536	0.2964	0.4392	0.8065	1.1547	39.89	0.1805	0.3263	0.5187	0.7708	0.9945	36.94	0.2257	0.4674	0.5723
PALR	1.0890	1.3765	25.22	0.1655	0.3066	0.4749	0.8530	1.1798	35.63	0.2355	0.3695	0.5242	0.8244	1.0774	34.53	0.1928	0.4309	0.5371
LG	0.9134	1.2583	35.06	0.2167	0.3691	0.5089	0.6037	0.9697	50.16	0.2512	0.3898	0.5371	0.6041	0.9006	50.38	0.3566	0.4987	0.6122
R4Rec	0.9093	1.2672	36.15	0.1812	0.3274	0.4624	0.6980	1.0446	47.55	0.2218	0.3359	0.5529	0.7036	1.0409	48.43	0.2607	0.4274	0.5468
Ours	0.8287	1.1373	40.08	0.2278	0.4579	0.5478	0.6144	1.0325	55.14	0.2967	0.4843	0.5919	0.5525	0.9233	55.98	0.5962	0.7168	0.7704

Table 2: Main results on Yelp, Amazon CDs and Vinyl, and Amazon Books. Baselines include KAR (Xi et al., 2024), RLMRec (Ren et al., 2024), PALR (Yang et al., 2023), LG (Wang et al., 2025), and R4Rec (Fang et al., 2025). For rating prediction, lower MAE/RMSE and higher Acc indicate better performance. For ranking evaluation, higher N@1/5/10 indicates better performance, where N@K denotes NDCG@K. Best results are shown in bold.

Method	Yelp						Amazon CDs and Vinyl						Amazon Books					
	MAE	RMSE	Acc (%)	N@1	N@5	N@10	MAE	RMSE	Acc (%)	N@1	N@5	N@10	MAE	RMSE	Acc (%)	N@1	N@5	N@10
w/o RL	0.8438	1.0954	35.25	0.2084	0.3519	0.5023	0.7084	1.0131	43.98	0.1605	0.3536	0.4979	0.7285	0.9734	41.74	0.2324	0.4231	0.5469
Ours	0.8287	1.1373	40.08	0.2278	0.4579	0.5478	0.6144	1.0325	55.14	0.2967	0.4843	0.5919	0.5525	0.9233	55.98	0.5962	0.7168	0.7704

Table 3: Effect of reinforcement learning on Yelp, Amazon CDs and Vinyl, and Amazon Books. “w/o RL” removes reward-based optimization from the profile controller. N@K denotes NDCG@K. Best results are shown in bold.

Backbone	MAE	RMSE	Acc (%)	N@1	N@5	N@10
Qwen3-4B	0.6159	1.0551	52.89	0.2490	0.4909	0.5778
Qwen3-8B	0.6144	1.0325	55.14	0.2967	0.4843	0.5919
Qwen3-14B	0.6299	0.9961	56.30	0.2895	0.5261	0.6171

Table 4: Effect of backbone scale on the Amazon CDs and Vinyl dataset. N@K denotes NDCG@K.

History chunk size	MAE	RMSE	Acc (%)	N@1	N@5	N@10
5	0.6139	1.1329	61.68	0.3074	0.5649	0.6213
10	0.6144	1.0325	55.14	0.2967	0.4843	0.5919
20	0.6775	1.0820	53.52	0.2506	0.4154	0.5557

Table 5: Effect of history chunk size on the Amazon CDs and Vinyl dataset. N@K denotes NDCG@K.

Method	MAE	RMSE	Acc (%)	N@1	N@5	N@10
All History Rating	0.8192	1.2924	26.77	0.1431	0.3397	0.4483
KAR	0.7456	1.0082	36.65	0.1934	0.3898	0.4462
RLMRec	0.7665	1.0568	38.87	0.1710	0.3705	0.5441
PALR	0.8372	1.1452	36.73	0.2538	0.3834	0.5506
LettinGo	0.6532	0.9432	48.39	0.2704	0.4192	0.5272
Reason4Rec	0.7754	1.0736	42.98	0.2392	0.3946	0.5474
Ours	0.6838	0.9867	52.96	0.3294	0.4934	0.5748

Table 6: Cross-evaluator robustness on the Amazon CDs and Vinyl dataset using a different frozen downstream predictor, Llama-3 8B. N@K denotes NDCG@K. Best results are shown in bold.

Amazon CDs and Vinyl, while keeping the generated profiles and rolling protocol unchanged. As shown in Table 6, LUPE remains the best-performing method, improving over the strongest baseline by 4.57 Accuracy points and 0.0590 / 0.0742 / 0.0242 in NDCG@1/5/10. This indicates that LUPE’s gains are not tied to the original Qwen3-based evaluator and transfer across frozen predictors.

4.4 Out-of-Distribution Analysis on History-Length Dynamics

We further analyze how rating prediction accuracy varies with user history length on Yelp, Amazon CDs and Vinyl, and Amazon Books. This analysis is conducted on the out-of-distribution (OOD) test split, whose users all have at least 500 historical interactions. For each user, we also evaluate accuracy at every 10-th interaction step up to the 500-th interaction. The full OOD results are reported in Appendix Table 7. For each history length, we average accuracy over all corresponding test instances. To improve readability, we smooth the raw curves using a centered running average with window size 10.

Figure 3 shows that our method maintains the strongest overall trend across most history lengths on all three datasets. This advantage is especially clear on Amazon Books, where our method stays above the baselines for nearly the entire trajectory. On Amazon CDs and Vinyl, it also achieves the best overall curve, while LettinGo and Reason4Rec are only competitive in limited local segments. On Yelp, although all methods fluctuate more, our method remains at or near the top for most history lengths. We also observe that Reason4Rec shows larger oscillations, especially at longer histories, consistent with its reliance on only the most recent 10 interactions. Similarly, LettinGo tends to decline in the later part of the trajectory, likely because it only uses the most recent 70 interactions and thus cannot fully preserve longer-range preference signals. We note that the OOD trends are

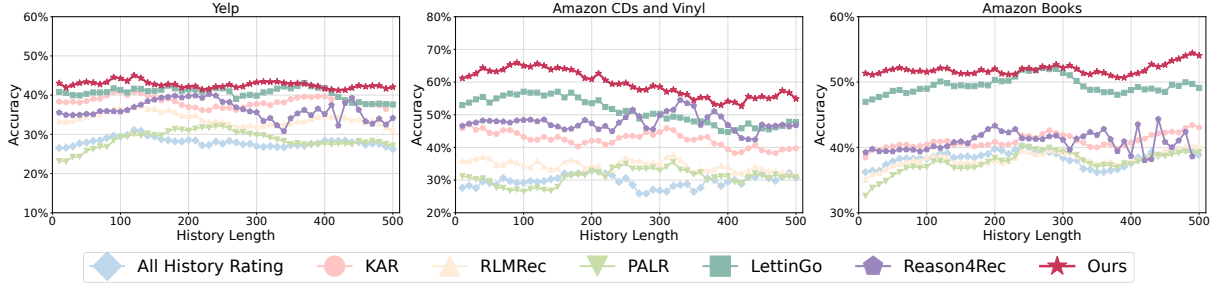


Figure 3: Accuracy across different user history lengths under the rolling evaluation protocol.

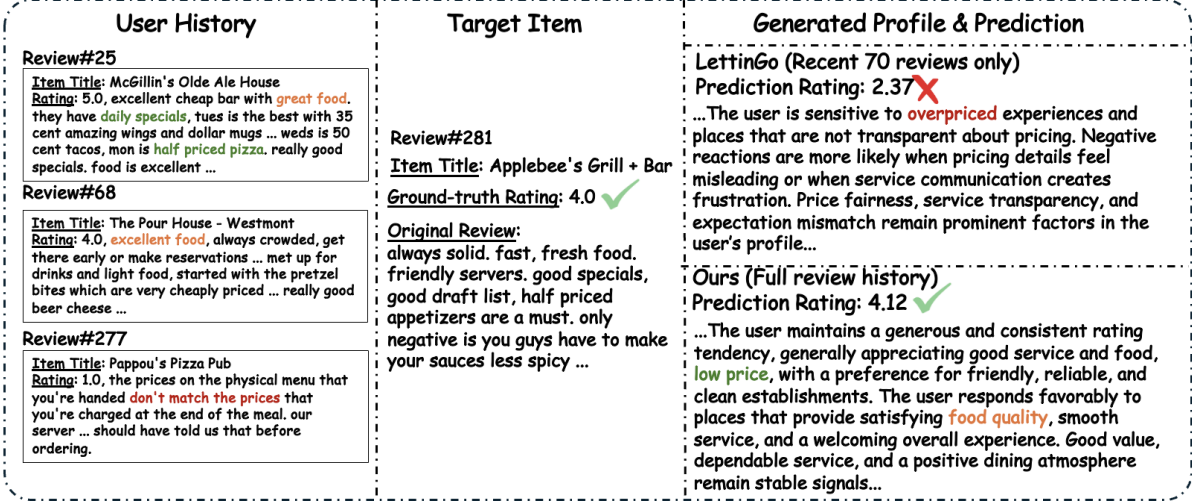


Figure 4: Case study from Yelp.

based on smoothed curves and primarily serve as qualitative evidence.

4.5 Case Study

We present representative case studies from all three datasets.

Yelp. Figure 4 presents a late-stage target item, review #281 (*Applebee's Grill + Bar*), whose key preference signals occur much earlier. Reviews #25 and #68 reveal stable preferences for food quality, drinks, specials, and value, while recent review #277 adds a negative short-window signal about pricing and service transparency. LettinGo is biased by this recent negative evidence and underestimates the target item, whereas our method preserves the long-term preference signals and predicts closer to the ground truth.

Additional representative case studies are provided in Appendix B.3.

4.6 Training Dynamics and Token Efficiency

Training Dynamics. We train LUPE separately on each dataset using the same procedure. Taking Amazon CDs and Vinyl as an example, the training

split contains 920 users and 153,818 interactions, with history lengths ranging from 100 to 497 (average 167.19). Training is conducted segment by segment, where each segment corresponds to one pass over the training users. We use Qwen3-8B as the policy model and train with $4 \times A100$ 80GB GPUs. Each segment takes about 1 hour, and training becomes largely stable after around the fifth segment. Figure 5 shows the training curves on Amazon CDs and Vinyl: the reward rises quickly in the early segments and then stabilizes at a high level, while the KL loss first increases and then gradually settles into a smoother range, indicating stable RL optimization under KL regularization.

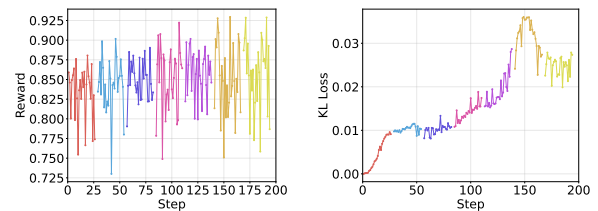


Figure 5: Training curves on Amazon CDs and Vinyl. Left: reward. Right: KL loss.

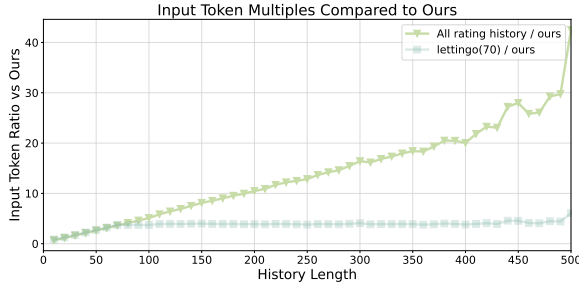


Figure 6: Input token multiples compared to LUPE as history length increases.

Input Token Efficiency. We compare input token costs under the rolling evaluation protocol. As shown in Figure 6, full-history prompting grows nearly linearly and exceeds $40\times$ the cost of LUPE at longer horizons, while LettinGo is still about $4\text{--}6\times$ more expensive despite truncation. By processing only one interaction chunk plus a compact profile state, LUPE keeps input size nearly constant and requires only 1298.61 tokens on average, showing better scalability for long-history recommendation.

5 Conclusion

In this paper, we presented LUPE, a long-history user profile evolution framework for LLM-based recommendation. Instead of treating user profiling as a one-shot summary over the full interaction history, LUPE models user preference as a persistent textual profile that is progressively updated over chronological interaction chunks. This provides a scalable way to handle long and diverse histories while preserving temporal coherence.

A key aspect of LUPE is that profile updating is directly optimized with downstream recommendation performance. At each stage, the model updates the previous profile state using the newly observed interaction chunk, and reinforcement learning encourages profile transitions that lead to better recommendation outcomes. As a result, the learned profile can preserve stable long-term preferences while adapting to newly emerged interests.

Experiments on multiple real-world interaction-based recommendation benchmarks show that LUPE consistently achieves strong performance on both rating prediction and ranking tasks. The results suggest that explicit profile evolution is more effective than static profile generation for long-history recommendation, especially when user histories are long and heterogeneous.

Limitations

Our study has several limitations. First, LUPE introduces higher training complexity than one-shot profiling methods, since it requires staged profile updates and reinforcement learning optimization. Second, our experiments are conducted on interaction-based recommendation datasets, so its generalization to other settings, such as sparse implicit-feedback or multi-domain scenarios, still needs further validation. Third, the profile evolution model is optimized with a frozen downstream recommender, which helps isolate the effect of profile quality but may limit adaptability to other recommendation architectures. Finally, the current method uses a fixed history segmentation strategy, which may not always align with real preference shifts. Exploring lighter training schemes, broader evaluation settings, and more adaptive update mechanisms is an important direction for future work.

References

- Seunghwan Bang and Hwanjun Song. 2025. Llm-based user profile management for recommender system. *arXiv preprint arXiv:2502.14541*.
- 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.
- Ting Chen, Wei-Li Han, Hai-Dong Wang, Yi-Xun Zhou, Bin Xu, and Bin-Yu Zang. 2007. Content recommendation system based on private dynamic user profile. In *2007 International conference on machine learning and cybernetics*, volume 4, pages 2112–2118. IEEE.
- Yue Chen, Yifei Sun, Lu Wang, Fangkai Yang, Pu Zhao, Minjie Hong, Yifei Dong, Minghua He, Nan Hu, Jianjin Zhang, Zhiwei Dai, Yuefeng Zhan, Weihao Han, Hao Sun, Qingwei Lin, Weiwei Deng, Feng Sun, Qi Zhang, Saravan Rajmohan, and Dongmei Zhang. 2026. DUET: Joint exploration of user-item profiles in recommendation system. In *Findings of the Association for Computational Linguistics: ACL 2026*.
- Yi Fang, Wenjie Wang, Yang Zhang, Fengbin Zhu, Qifan Wang, Fuli Feng, and Xiangnan He. 2025. [Reason4rec: Large language models for recommendation with deliberative user preference alignment](#). *Preprint*, arXiv:2502.02061.
- Chongming Gao, Ruijun Chen, Shuai Yuan, Kexin Huang, Yuanqing Yu, and Xiangnan He. 2025. Sprec:

- Self-play to debias llm-based recommendation. In *Proceedings of the ACM on Web Conference 2025*, pages 5075–5084.
- Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*.
- Zhaolin Gao, Joyce Zhou, Yijia Dai, and Thorsten Joachims. 2024. End-to-end training for recommendation with language-based user profiles. *arXiv preprint arXiv:2410.18870*.
- Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*.
- Yangqin Jiang, Yuhao Yang, Lianghao Xia, Da Luo, Kangyi Lin, and Chao Huang. 2025. Reclm: Recommendation instruction tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15443–15459.
- Jiacheng Lin, Tian Wang, and Kun Qian. 2025. Rec-r1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning. *arXiv preprint arXiv:2503.24289*.
- Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. In *Proceedings of the ACM Web Conference 2024*, pages 3497–3508.
- Hao Liu, Le Wu, Min Hou, Han Wu, Kun Zhang, Xin Li, and Si Wei. 2025a. An efficient llm-based evolutionary recommendation with locate-forget-update paradigm. *arXiv preprint arXiv:2511.16414*.
- Jiahao Liu, Shengkang Gu, Dongsheng Li, Guangping Zhang, Mingzhe Han, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025b. [Agentcf++: Memory-enhanced llm-based agents for popularity-aware cross-domain recommendations](#). In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 2566–2571, New York, NY, USA. Association for Computing Machinery.
- Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2685–2692.
- Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*, pages 3464–3475.
- Yunxiao Shi, Wujiang Xu, Zhang Zeqi, Xing Zi, Qiang Wu, and Min Xu. 2025. Personax: A recommendation agent-oriented user modeling framework for long behavior sequence. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5764–5787.
- Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867.
- Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, pages 3251–3257.
- Qwen Team. 2025. [Qwen3 technical report](#). Preprint, arXiv:2505.09388.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).
- Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153*.
- Lu Wang, Di Zhang, Fangkai Yang, Pu Zhao, Jianfeng Liu, Yuefeng Zhan, Hao Sun, Qingwei Lin, Weiwei Deng, Dongmei Zhang, Feng Sun, and Qi Zhang. 2025. [Lettingo: Explore user profile generation for recommendation system](#). In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2, KDD '25*, New York, NY, USA. Association for Computing Machinery.
- Triyanna Widiyaningtyas, Indriana Hidayah, and Teguh B Adji. 2021. User profile correlation-based similarity (upcsim) algorithm in movie recommendation system. *Journal of Big Data*, 8(1):52.
- 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.
- Yu Xia, Rui Zhong, Hao Gu, Wei Yang, Chi Lu, Peng Jiang, and Kun Gai. 2025. Hierarchical tree search-based user lifelong behavior modeling on large language model. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1758–1767.
- Fan Yang, Zheng Chen, Ziyang Jiang, Eunah Cho, Xiaojian Huang, and Yanbin Lu. 2023. Palr: Personalization aware llms for recommendation. *arXiv preprint arXiv:2305.07622*.

Xiaoyong Yang, Yadong Zhu, Yi Zhang, Xiaobo Wang, and Quan Yuan. 2020. Large scale product graph construction for recommendation in e-commerce. *arXiv preprint arXiv:2010.05525*.

Jiarui Zhang. 2024. Guided profile generation improves personalization with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4005–4016.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024. *Agentcf: Collaborative learning with autonomous language agents for recommender systems*. In *Proceedings of the ACM Web Conference 2024, WWW '24*, page 3679–3689, New York, NY, USA. Association for Computing Machinery.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5941–5948.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1059–1068.

Chenxu Zhu, Shigang Quan, Bo Chen, Jianghao Lin, Xiaoling Cai, Hong Zhu, Xiangyang Li, Yunjia Xi, Weinan Zhang, and Ruiming Tang. 2024. Liber: Life-long user behavior modeling based on large language models. *arXiv preprint arXiv:2411.14713*.

A Supplementary Method Details

A.1 Training Algorithm

Here we provide the full training procedure of LUPE, described in Algorithm 1.

A.2 Frozen Downstream Rating Predictor

The downstream predictor is a frozen LLM-based rating estimator instantiated with Qwen3-8B. It takes as input the evolved textual user profile together with the target item information, including the item title and description, and predicts the rating that the user would assign to the target item.

We implement the predictor using prompted generation. The model is instructed to output exactly one `<rating>` tag containing a numeric score between 1.00 and 5.00. To support fine-grained reward comparison in reinforcement learning, we require the predictor to return ratings with **two-decimal precision**. This continuous-valued output allows the reward to distinguish incremental differences in prediction error, rather than collapsing them into coarse integer-level matches.

Algorithm 1 LUPE Training

Require: Training user set $\mathcal{U}_{\text{train}}$; profile evolution model f_θ ; frozen downstream predictor g_ϕ ; chunk size K ; group size G ; batch size B

Ensure: Trained profile evolution model f_θ

- 1: Initialize all user profiles in $\mathcal{U}_{\text{train}}$ as empty
- 2: $s \leftarrow 1$
- 3: **while** $\mathcal{U}_{\text{train}} \neq \emptyset$ **do**
- 4: **for** each mini-batch $\mathcal{B} \subset \mathcal{U}_{\text{train}}$ with $|\mathcal{B}| = B$ **do**
- 5: $\mathbf{R}_s^\mathcal{B} \leftarrow$ the s -th history chunks of users in $\mathcal{B}$
- 6: $\mathbf{i}_s^\mathcal{B} \leftarrow$ the next target items after $\mathbf{R}_s^\mathcal{B}$
- 7: $\mathbf{y}_s^\mathcal{B} \leftarrow$ the corresponding ground-truth ratings
- 8: $\tilde{\mathbf{P}}_s^{\mathcal{B},1:G} \leftarrow f_\theta(\mathbf{P}_{s-1}^\mathcal{B}, \mathbf{R}_s^\mathcal{B})$
- 9: $\hat{\mathbf{y}}_s^{\mathcal{B},1:G} \leftarrow g_\phi(\tilde{\mathbf{P}}_s^{\mathcal{B},1:G}, \mathbf{i}_s^\mathcal{B})$
- 10: $\text{reward}_s^{\mathcal{B},1:G} \leftarrow 1 - \frac{|\mathbf{y}_s^\mathcal{B} - \hat{\mathbf{y}}_s^{\mathcal{B},1:G}|}{M}$
- 11: $\theta \leftarrow \text{GRPO}(\theta, \mathbf{r}_s^{\mathcal{B},1:G})$
- 12: **end for**
- 13: $\mathbf{P}_s \leftarrow f_\theta(\mathbf{P}_{s-1}, \mathbf{R}_s)$
- 14: Remove users that do not have a valid $(s+1)$ -th target interaction from $\mathcal{U}_{\text{train}}$
- 15: $s \leftarrow s + 1$
- 16: **end while**
- 17: **return** f_θ

The predictor remains fixed throughout both training and inference. As a result, improvements in downstream prediction performance can be attributed to better user profile evolution rather than changes in the evaluator itself.

Figure 7 illustrates the simplified prompting structure used by the frozen rating predictor.

B Additional Analysis

B.1 Results on the Out-of-Distribution Split

We report the full results on the **out-of-distribution (OOD)** split in Table 7.

B.2 Industrial-Style Retrieval-then-Reranking Evaluation

To complement the controlled hard-negative ranking protocol used in the main experiments, we further evaluate LUPE in an industrial-style two-stage retrieval-then-reranking setting. Specifically, we first use Swing (Yang et al., 2020), an item-to-item similarity-based retrieval method used in large-scale e-commerce recommendation systems,

Method	Yelp			Amazon CDs and Vinyl			Amazon Books		
	MAE	RMSE	Acc (%)	MAE	RMSE	Acc (%)	MAE	RMSE	Acc (%)
All History Rating	1.0671	1.5009	28.53	0.5798	1.2439	29.47	0.7671	1.0086	38.53
KAR (Xi et al., 2024)	0.8083	1.0317	38.34	0.7151	0.9694	42.53	0.7296	0.9259	41.04
RLMRec (Ren et al., 2024)	0.9138	1.1799	34.05	0.7887	1.0431	34.44	0.7594	0.9637	38.29
PALR (Yang et al., 2023)	1.0021	1.2801	28.64	0.7938	1.0423	31.33	0.7435	0.9407	37.80
LettinGo (Wang et al., 2025)	0.7482	0.9630	42.09	0.6345	0.9888	51.43	0.6681	0.9804	49.02
Reason4Rec (Fang et al., 2025)	0.8093	1.0672	38.15	0.6980	1.0090	46.88	0.7457	0.9409	42.34
Ours	0.7482	0.9981	43.86	0.6030	0.9570	58.99	0.6332	0.8617	51.92

Table 7: Main results on the out-of-distribution (OOD) split. The OOD split contains users with substantially longer histories than those in the training set. Lower MAE/RMSE and higher Acc indicate better performance. Best results are shown in bold.

Method	Yelp				Amazon CDs and Vinyl				Amazon Books			
	N@1	N@5	N@10	MRR	N@1	N@5	N@10	MRR	N@1	N@5	N@10	MRR
All History Rating	0.0365	0.0587	0.0914	0.0631	0.0359	0.0690	0.0845	0.0810	0.0594	0.0827	0.1208	0.1051
KAR (Xi et al., 2024)	0.0308	0.0766	0.1072	0.0863	0.0362	0.0854	0.1176	0.1090	0.0713	0.1121	0.1805	0.1371
RLMRec (Ren et al., 2024)	0.0513	0.1217	0.1577	0.1372	0.0380	0.1190	0.1627	0.1367	0.0680	0.1226	0.1960	0.1598
PALR (Yang et al., 2023)	0.0959	0.1447	0.2035	0.1846	0.0806	0.1458	0.1949	0.1576	0.1598	0.1870	0.2268	0.1968
LettinGo (Wang et al., 2025)	0.1219	0.1996	0.2409	0.2051	0.1436	0.1919	0.2193	0.2070	0.1825	0.2249	0.2757	0.2564
Reason4Rec (Fang et al., 2025)	0.0830	0.1185	0.1702	0.1477	0.1031	0.1289	0.1990	0.1406	0.1230	0.1494	0.2056	0.1661
LUPE	0.1309	0.2035	0.2528	0.2171	0.1542	0.2118	0.2335	0.2203	0.1881	0.2342	0.2942	0.2746

Table 8: Industrial-style retrieval-then-reranking evaluation on Yelp, Amazon CDs and Vinyl, and Amazon Books. N@K denotes NDCG@K. Best results are shown in bold.

Prompted Rating Prediction Input

Task

Predict the rating (1.00–5.00) that the user would give to the target item based strictly on the user profile and item information. Output exactly one `<rating>` tag with two-decimal precision.

User Profile

{...The user strongly prefers psychological thrillers with suspenseful, twist-driven plots and immersive pacing...}

Item

Title: The Silent Patient

Description: A psychological thriller featuring suspense, tension, and a strong twist ending.

↓ FROZEN RATING PREDICTOR ↓

Structured Rating Output

`<rating>4.27</rating>`

Figure 7: Prompted downstream rating prediction in LUPE. The frozen predictor receives the evolved textual user profile together with the target item information and outputs one continuous rating in a constrained `<rating>` format.

to retrieve a top-100 candidate set for each target interaction. Then, each method reranks the retrieved candidates using its constructed user profile or history representation and the same frozen downstream predictor. Since this remains a single-positive evaluation setting, we additionally report MRR to directly measure the rank position of the

ground-truth item. Results are shown in Table 8.

Table 8 shows that LUPE remains effective under the industrial-style retrieval-then-reranking setting. Across all three datasets, LUPE consistently achieves the best NDCG@1, NDCG@5, NDCG@10, and MRR. The gains on NDCG@1 and MRR are particularly important because the evaluation follows a single-positive ranking setup, where the rank position of the ground-truth item directly reflects top-rank recommendation quality. These results indicate that the evolved user profile is not only useful in the controlled hard-negative protocol, but also helps place the ground-truth item near the top of a retrieved top-100 candidate list. Overall, this suggests that LUPE can naturally serve as a profile-aware reranking component in candidate-generation-plus-reranking recommendation systems.

B.3 Case Study

Amazon CDs and Vinyl. Figure 8 shows a late-stage target item (#181, *Homecoming: Live In Orlando*) whose key signal is not simply whether the user is generally positive, but whether a release is judged as truly exceptional or merely strong. Earlier reviews already reflect this distinction. In review #21, the user emphasizes polished production and memorable large-scale sound, while review #78 explicitly states that an album is enjoyable

User History	Target Item	Generated Profile & Prediction
Review#21 <u>Item Title:</u> The New Earth <u>Rating:</u> 5.0, this album captures a very uplifting and expansive sound, and it feels ambitious in both concept and execution, the arrangements are rich, the production is polished , and the music has a sense of scale that makes it memorable ...	Review#181 <u>Item Title:</u> Homecoming: Live In Orlando <u>Ground-truth Rating:</u> 4.0 ✓	LettinGo (Recent 70 reviews only) <u>Prediction Rating:</u> 5.00 ✗ ...The user demonstrates a very generous and enthusiastic rating tendency, with frequent top scores and a strong positive orientation toward live recordings , polished production, and emotionally engaging performances. The profile emphasizes broad appreciation for uplifting music, strong atmosphere , and memorable listening experiences, but places less weight on the distinction between excellent works and merely good ones...
Review#78 <u>Item Title:</u> A Christmas Celebration <u>Rating:</u> 4.0, this is enjoyable and well produced, but it does not quite reach the same level as the strongest albums in the collection. the performance is solid and appealing, though not especially exceptional in the long run ...	Original Review: this live album is enjoyable and well executed, with strong musical energy and a polished overall presentation. however, while it is clearly a good release, it does not feel as distinctive or outstanding as the very best entries in the discography ...	Ours (Full review history) <u>Prediction Rating:</u> 4.31 ✓ ...The user shows a stable tendency to assign 5.0 ratings to exceptional works and 4.0 ratings to albums that are strong but do not fully reach the highest standard . Reviews consistently focus on performance quality , production polish , emotional impact, and the broader artistic value of the recording. Across the history, the user distinguishes between memorable, standout releases and works that are enjoyable, well-produced, and solid, yet not truly exceptional...
Review#176 <u>Item Title:</u> Worship <u>Rating:</u> 5.0, this is a strong live recording with energetic performance and emotional presence. the album works especially well in capturing atmosphere and audience connection, and it stands out as one of the more memorable recent releases from this group ...		

Figure 8: Case study from Amazon CDs and Vinyl.

and well produced but “does not quite reach the same level” as the strongest works. A recent review (#176) further highlights appreciation for live recording quality, atmosphere, and performance energy. LettinGo captures the user’s broad enthusiasm for live recordings and polished production, but fails to preserve the finer-grained calibration between exceptional works and albums that are good yet not top-tier, and therefore overestimates the target (5.00). In contrast, our method retains this long-term evaluative pattern and correctly places the target closer to the ground-truth rating of 4.0.

Amazon Books. Figure 9 shows a user whose target item appears very late in the sequence (review #251). Earlier reviews already reveal stable fine-grained preferences: review #27 emphasizes courtroom drama, plot, and mystery, while review #66 again highlights courtroom drama and well-developed characters. At the same time, a recent review (#241) introduces a strong negative signal centered on filler, boredom, and explicit language. LettinGo appears to overreact to this recent local evidence and predicts a much lower score, while our method preserves the earlier positive preference structure together with a more faithful rating calibration. As a result, our prediction matches the ground-truth rating of 5.0 much more closely than LettinGo.

B.4 Feature visualization of generated profiles

We further analyze the semantic structure of generated user profiles through t-SNE (Van der Maaten and Hinton, 2008) visualization. For each dataset,

we randomly sample **10 users** and collect all generated profiles associated with these users from the rollout data. We then encode the profile texts using **all-mpnet-base-v2** (Song et al., 2020) and project the resulting embeddings into two dimensions using t-SNE. Profiles from the same user are shown with the same color.

Figure 10 compares the profile distributions produced by **LettinGo** and our method. A clear pattern is that the profiles generated by our method form much more compact and better-separated user-specific clusters across all three datasets, whereas LettinGo often produces more mixed and diffuse distributions. This difference is particularly evident on **Amazon Books** and **Amazon CDs and Vinyl**, where the clusters under our method are visually distinct, while LettinGo exhibits substantial overlap among different users. On **Yelp**, although the distributions are inherently noisier, our method still shows noticeably improved cluster compactness and inter-user separation. These results suggest that our structured profile update mechanism produces more consistent and discriminative user representations, which helps explain the stronger downstream recommendation performance observed in the main results.

C Experiment Setup

C.1 Training Details

We train the policy model using GRPO implemented in the verl framework. Table 9 summarizes the main training hyperparameters.

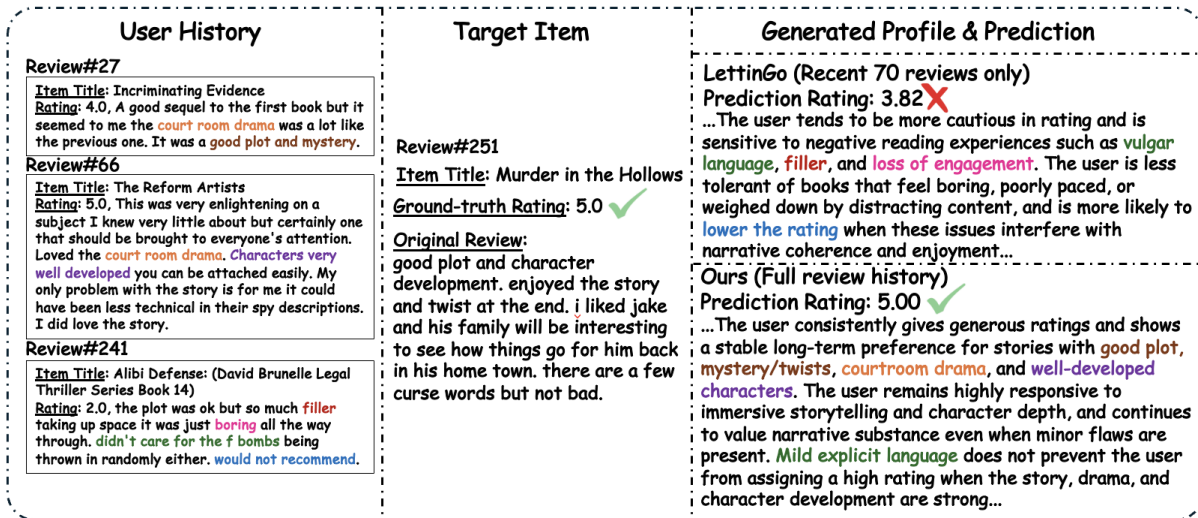


Figure 9: Case study from Amazon Books.

Hyperparameter	Value
Rollout backend	vLLM
Rollout samples per prompt	8
Rollout temperature	1.0
Training batch size	32
Learning rate	1×10^{-6}
Max prompt length	8192
Max response length	1024
KL loss coefficient	0.001
KL loss type	low_var_kl
KL in reward	Disabled
Entropy coefficient	0
Gradient checkpointing	Enabled
Actor dynamic batch size	True
Reference log-prob dynamic batch size	True
Rollout log-prob dynamic batch size	True

Table 9: Main training hyperparameters used in RL optimization.

C.2 Baseline Prompts

KAR Prompt

Task: Analyze user preferences based on business reviewing history

Input: {user_history} - User's business reviewing history with sentiments over time

Instructions:

1. Analyze the user's preferences considering business names and categories
2. Take into account sentiment patterns over time
3. Provide clear explanations based on reviewing history details
4. Consider other pertinent factors that may influence preferences

PALR Prompt

Task: Summarize user preferences using keywords.

Input: {user_history} - historical businesses with user sentiments.

Output Format: An itemized list ranked by importance.

Template:

- KEY_WORD_1: "HISTORY_BUSINESS_1", "HISTORY_BUSINESS_2"
- KEY_WORD_2: "HISTORY_BUSINESS_3"

Instructions:

1. Extract key preference indicators from user interaction history.
2. Rank keywords by importance.

RLMRec Prompt

Role: Business recommendation assistant

Task: Determine business types a user is likely to enjoy

Input Format:

- Title: Business name
- Categories: Business categories
- Sentiment: User sentiment toward business

Output Requirements:

1. JSON format only

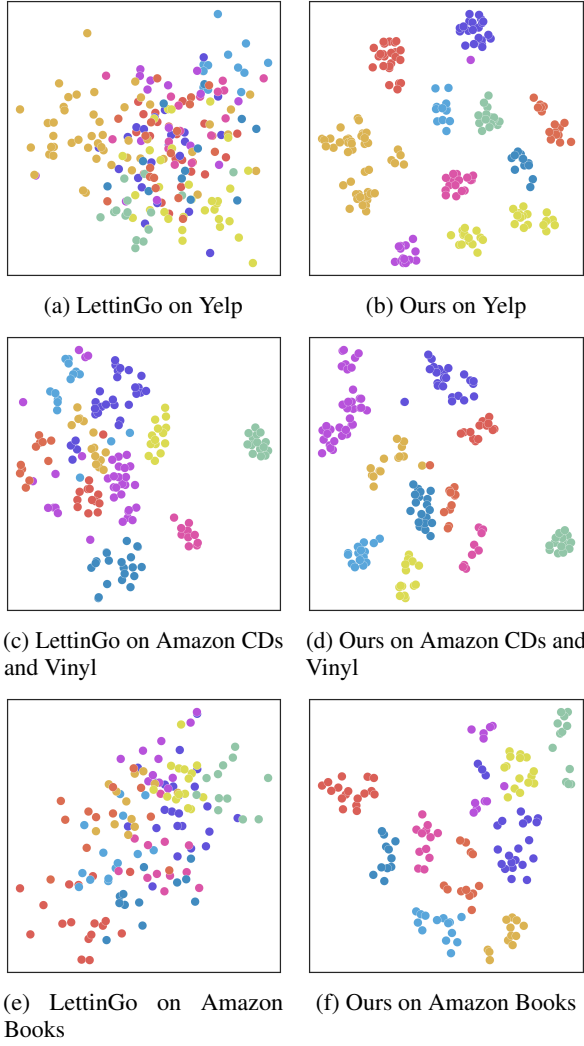


Figure 10: t-SNE visualization of generated user profiles on Yelp, Amazon CDs and Vinyl, and Amazon Books. For each dataset, we randomly sample 10 users and plot the embeddings of all generated profiles from these users. Profiles with the same color belong to the same user. Compared with LettinGo, our method produces more compact and better-separated user-specific clusters.

2. Structure:

```
{
  "summarization": "Types of businesses user
likely enjoys" ( $\leq 100$  words),
  "reasoning": "Brief explanation for summarization" (no word limit)
}
```

3. No additional text outside JSON

Input: INTERACTION ITEMS:
{user_history}

LG Prompt

You will serve as an assistant to help me generate a user profile based on this user's sentiments history to better understand this users' interest and thus predict his/her sentiment about a target item. I will provide you with some behavior history of the user in this format: [item attributes and sentiment]. The user profile you generate should contain as much useful content as possible to help predict the user's sentiment towards a new business.

USER HISTORY: {user_history}.

PROFILE YOU GENERATE:

R4Rec Prompt (Reasoner)

User Review History

$\langle H_u \rangle$ organized as below)

1. Title of Item 1

Positive Aspects: [Aspect 1], [Aspect 2], ...

Negative Aspects: [Aspect 1], [Aspect 2], ...

User Preference Elements: [Preference 1], [Preference 2], ...

2. Title of Item 2

Positive Aspects: [Aspect 1], [Aspect 2], ...

Negative Aspects: [Aspect 1], [Aspect 2], ...

User Preference Elements: [Preference 1], [Preference 2], ...

...

Item Review History by Other Users

$\langle H_i \rangle$ organized in the same format as above)

Task: Analyze whether the user will like the new Music i based on the user's preferences and the item's features. Provide your rationale in one concise paragraph.

D The Use of Large Language Models

We used a Large Language Model (LLM) solely for editorial support, such as improving grammar, refining phrasing, and enhancing readability. All substantive aspects of this work, including the core ideas, method development, experimental setup, implementation, and result analysis, were carried out by the authors independently. The LLM was not involved in forming the research questions, designing the methodology, conducting the experiments, or drawing conclusions from the findings.