Leveraging Unpaired Feedback for Long-Term LLM-based Recommendation Tuning

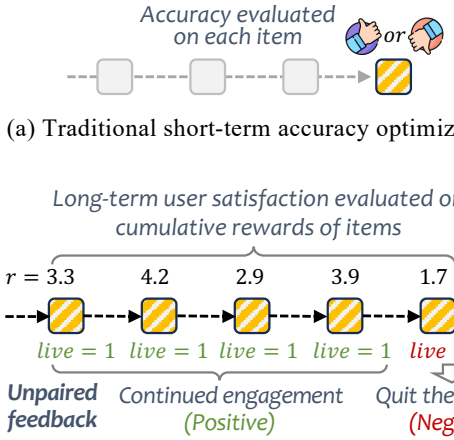
Jizhi Zhang[1*] **Chongming Gao**[1†] **Wentao Shi**[1]
Xin Chen[2] **Jingang Wang**[2] **Xunliang Cai**[2] **Fuli Feng**[1†]
[1]University of Science and Technology of China, [2]Meituan.
{cdzhangjizhi, chongminggao, shiwentao123}@mail.ustc.edu.cn,
chenxin148@meituan.com, fulifeng93@gmail.com

Abstract

Most recommender systems focus on short-term objectives such as click-through rate, often at the expense of long-term user satisfaction. This can lead to echo chambers, where users are repeatedly exposed to redundant content. While recent efforts integrate Large Language Models (LLMs) into recommendation, they typically inherit this short-sighted focus. In this work, we highlight unpaired feedback—implicit signals such as continued engagement (positive) or silent disengagement (negative) that lack explicit contrastive labels—as a key challenge for long-term recommendation. Effectively learning from such feedback is crucial for improving LLM-based recommenders in dynamic user environments. To this end, we propose **ULRec** (**U**npaired Feedback for **L**ong-Term LLM-based **Rec**ommendation Tuning), a simple framework that fine-tunes LLMs using both positive and negative unpaired feedback. ULRec leverages the KTO algorithm to incorporate these signals without requiring paired supervision. Despite its simplicity, ULRec consistently improves long-term recommendation performance, demonstrating the value of modeling unpaired user feedback.

1 Introduction

Recommendation systems play a central role in filtering information and shaping user experiences (Wu et al., 2022; Li et al., 2023a; Silveira et al., 2019). While most existing work focuses on short-term objectives such as click-through rate (CTR) (Bao et al., 2023; Guo et al., 2017; Zhou et al., 2018), this often leads to long-term issues like echo chambers, where users are repeatedly exposed to similar content (Gao et al., 2023b; Ge et al., 2021; Shi et al., 2024). Such short-sighted optimization can degrade user satisfaction over time and limit content diversity (Lin et al., 2025b).



(a) Traditional short-term accuracy optimization

(b) Long-term satisfaction modeling

Figure 1: The difference between short-term and long-term recommendations, along with examples of unpaired feedback in long-term recommendations.

Addressing long-term recommendation requires models to account for delayed rewards during training (Gao et al., 2023a). Prior efforts incorporate reinforcement learning into deep models to optimize long-term objectives, but are limited by the capacity of traditional architectures (Shi et al., 2024). Recently, Large Language Models (LLMs) have shown promise in recommendation tasks due to their strong language understanding and interaction abilities (Hou et al., 2024; Gao et al., 2023c). However, most LLM-based recommenders remain focused on short-term accuracy (Gao et al., 2025a,b; Cai et al., 2025), with little attention to long-term user retention.

In this work, we revisit the long-standing echo chamber problem in recommendation through the lens of LLMs, and argue that effectively modeling user feedback over time is essential for improving long-term user satisfaction. We distinguish two types of feedback signals: *positive feedback* from users who remain engaged, and *negative feed-*

[*] Work is done during internship at Meituan.
[†] Corresponding Author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 24507–24521
November 4-9, 2025

back from users who disengage due to content redundancy. While both types of signals are valuable, they are often unpaired—lacking explicit contrastive examples or follow-up outcomes—making them difficult to directly use in conventional supervised or reward-based training paradigms. We argue that this underexplored but practically important setting of learning from **unpaired feedback** poses a core challenge for advancing LLM-based recommendation.

To address this, we propose **ULRec** (**U**npaired Feedback for **L**ong-Term LLM-based **Rec**ommendation Tuning), a simple yet effective framework that incorporates both positive and unpaired negative feedback into LLM training. For the latter, we leverage the KTO algorithm (Ethayarajh et al., 2024), which enables the direct suppression of negatively labeled outputs without requiring paired comparisons. Despite its simplicity, ULRec consistently improves long-term recommendation quality, highlighting the value of explicitly modeling unpaired negative feedback in this setting.

Our contributions can be summarized as:

- We identify unpaired feedback—implicit positive and negative user signals without explicit contrastive labels—as a core challenge for long-term recommendation with LLMs.
- We propose ULRec, a lightweight framework that fine-tunes LLMs using both types of unpaired feedback, leveraging the KTO algorithm to handle supervision without paired data.
- We demonstrate that ULRec substantially improves long-term recommendation performance across benchmarks, validating the value of modeling unpaired feedback in LLM-based systems.

2 Preliminary

In this section, we first present the problem formulation and briefly describe the interactive environment used for simulation.

2.1 Problem Formulation

In this paper, we mainly focus on the echo chamber scenario. The classical work in this field has been conducted in the interactive recommendation scenario (Gao et al., 2023b), and it is also the setting that we follow. In this setting, at the j-th step, the goal of the recommender π is to recommend an item i (noted as making an action a_j) from the candidate set $\mathcal{I}$ to a user u, who comes from the user set $\mathcal{U}$, based on the current state s_j. s_1 denotes the initial state. After the action a_j is done, the recommender will receive a reward r_j from the user. Thus, we can denote the entire episode of interaction as:

$$\mathcal{T}^u_{1...J} = \{u, s_1, a_1, r_1, ..., s_J, a_J, r_J\}. \quad (1)$$

Meanwhile, the subsequence $\mathcal{T}^u_{1...j}$ of $\mathcal{T}^u_{1...J}$, where $1 \leq j \leq J$, can be used to represent the trajectory.

2.2 Interactive Environment

We conduct our exploration in an interactive recommendation setting, where users are required to provide real-time feedback on recommendation results. Since collecting data directly from the real world is costly, we follow previous work by building a simulated interactive environment to facilitate training and testing using the offline recommendation dataset (Shi et al., 2024; Gao et al., 2023a,b). In this interactive environment, when the user becomes aware of an echo chamber or is recommended items they do not like, the user will give an explicit exit signal and terminate the interaction.

We can then formalize the interaction process between the environment (ENV), which simulates the user group $\mathcal{U}$, and the recommender. First, the recommender π receives user $u \in \mathcal{U}$ and the corresponding state s_j at step j, and then outputs a recommendation result a_j:

$$a_j \leftarrow \pi(s_j, u). \quad (2)$$

Next, the environment generates feedback based on user u, state s_j, and recommendation a_j:

$$F^u_j = (l_j, r_j) \leftarrow \text{ENV}(s_j, u, a_j), \quad (3)$$

where F^u_j denotes the feedback given by user u in step j for the recommendation a_j, which consists of two parts: whether the user is live l_i, and the user's reward r_j of recommendation a_j.

• **Exit Protocol.** During the interaction process, the most important information provided by the environment is the live signal l_j. Following previous work (Shi et al., 2024; Gao et al., 2023b), it is designed as a comprehensive judgment result based on both user preference and echo chamber effects. Specifically, when the following situations occur, $l_j = 0$ indicates that the user exits the interaction, otherwise $l_j = 1$ means the interaction continues: From the perspective of **user preference**, if the

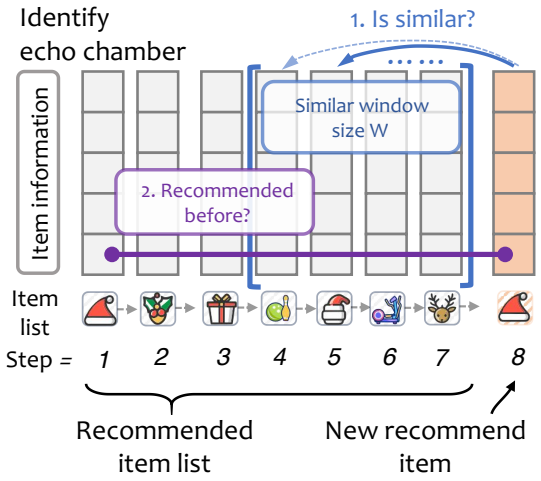


Figure 2: Protocol for identifying echo chamber in interactive environment for long-term recommendation.

returned reward r_j is less than a certain threshold τ_r, then $l_j = 0$. From the perspective of the **echo chamber**, as demonstrated in Figure 2 if among the most recent W interacted items there exists an item whose distance from a_j is less than a certain threshold (τ_d), i.e., the recommender π recommends an item that is very similar to previous ones, the user will exit ($l_j = 0$). Distinguishing from the previous simulation environment (Shi et al., 2024; Gao et al., 2023b), we further introduce an additional exit condition: if the model repeatedly recommends an item that has already been recommended in history, i.e., $a_j \in \mathcal{T}^{1...j}$, the user will also choose to exit to make the simulation environment similar to real-world scenarios and avoid potential reward hacking.

• **Evaluation Protocol.** Since we are measuring the recommender's ability to break the echo chamber in interactive recommendation, and users will exit once they perceive the echo chamber, a longer interaction with the user indicates that the recommender can better break the echo chamber while satisfying user preferences. Therefore, for a given episode $\mathcal{T}^u_{1...J}$ corresponding to user u, the metric is defined as:

$$L_u := \sum_{j=1}^{J} l_j, \quad R_u := \sum_{j=1}^{J} l_j * r_j. \tag{4}$$

L_u denotes the length of the interaction, and R_u denotes the sum of cumulative rewards over the entire trajectory. Larger values of both indicate a stronger ability of the model to break the echo chamber. Finally, the average values of L_u and R_u over the test user set $\mathcal{U}_{te}$ are used as the final evaluation metrics L and R:

$$L := \frac{1}{|\mathcal{U}_{te}|} \sum_{u \in \mathcal{U}_{te}} L_u, \quad R := \frac{1}{|\mathcal{U}_{te}|} \sum_{u \in \mathcal{U}_{te}} R_u. \tag{5}$$

3 Method

In this section, we will introduce the motivation and overview of our method, as well as the specific implementation modules.

• **Motivation**: We believe that the key to breaking echo chamber and improving the long-term recommendation performance of LLM-based recommenders lies in fine-tuning LLM models using users' unpaired positive and negative feedback. To this end, we have designed a simple yet efficient framework ULRec to leverage such feedback. Although the method is straightforward, its effectiveness demonstrates the importance of utilizing this feedback to enhance long-term recommendations.

• **Overview**: Considering the significant differences between positive and negative feedback, we have made targeted designs for each within the ULRec framework. Specifically, as demonstrated in Section 3.1 and Section 3.2, we design a two-stage training process to effectively leverage these two distinct types of feedback. As a preliminary exploration, we focus on leveraging the collected interactions between LLM and a user training set $\mathcal{U}_{tr}$, i.e., their corresponding episodes $\mathcal{T}^u_{1...J}$, for fine-tuning and thereby enhance the model's long-term recommendation capability.

3.1 Learning from Positive Feedback

• **Target**: By learning from successful cases of positive interactions within episodes, the LLM can identify which items to recommend, helping to break echo chamber.

• **Challenge**: The format of the training data does not match that of typical recommendation data; for example, it involves multiple rounds and mixes in information such as rewards. This requires us to design and construct suitable data formats for fine-tuning the LLM.

• **Method**: The issue of misaligned multi-round formats is addressed by constructing samples at the step level. Specifically, we decompose all successful actions in each episode into individual steps, as follows:

$$\{\mathcal{T}^u_{1...j-1}, a_j\}_{j=1}^{J} \leftarrow \mathcal{T}^u_{1...J}. \tag{6}$$

For each step, we hope the LLM can recommend a_j (which has been shown to elicit positive user

feedback) based on the $\mathcal{T}^u_{1\ldots j-1}$. Thus, the target for finetuning at each step is formulated as:

$$a_j \leftarrow LLM(\mathcal{T}^u_{1\ldots j-1}). \tag{7}$$

Then, we can conduct step level fine-tune for each interactive recommendation step and for each user u from training dataset $\mathcal{U}_{tr}$, thus avoiding interference from rewards or other information that does not need to be output by the LLM during supervised fine-tuning. The optimization object for each user is:

$$\min_{\theta} \sum_{j=1}^{J-1} -\log(\pi_\theta(a_j|\mathcal{T}^u_{1\ldots j-1})), \tag{8}$$

$\pi_\theta(\cdot)$ denotes the LLM, and θ represents the learnable parameters within the LLM. $J-1$ is used because the final step in each trajectory corresponds to the user's exit, which serves as a signal of negative feedback rather than a positive interaction.

3.2 Learning from Negative Feedback

• **Target**: By learning from failed cases in episodes, we aim to help the LLM recognize which items may cause echo chamber, thereby optimizing recommendation strategies to enhance long-term benefits.
• **Challenge**: Negative feedback is often unpaired, as users who disengage due to echo chamber effects provide no subsequent positive interactions for direct comparison, which makes it difficult to construct effective training signals for fine-tuning the LLM.
• **Method**: Considering the unpaired nature of user interactions and our goal of highlighting the importance of incorporating negative feedback when tuning LLMs for long-term recommendation, we directly adopt the KTO algorithm (Ethayarajh et al., 2024), with minor modifications for training stability (see Appendix A.1 for details). Specifically, based on the model fine-tuned with positive feedback, we further employ the KTO algorithm to enable the LLM to recognize which items may lead to echo chamber effects and thus should not be recommended. Our optimization objective is then formulated as:

$$\begin{aligned}
&\min_{\theta} L_{\mathrm{KTO}}\left(\pi_\theta, \pi_{\mathrm{ref}}\right) = \mathbb{E}_{\mathcal{T},a,e\sim D_{tr}}\left[\lambda_l - v(\mathcal{T}, a)\right], \\
&v(\mathcal{T}, a) = \begin{cases} \lambda_1 \sigma\left(\beta\left(r_\theta(\mathcal{T}, a) - z_0\right)\right), \text{if } l = 1 \\ \lambda_0 \sigma\left(\beta\left(z_0 - r_\theta(\mathcal{T}, a)\right)\right), \text{if } l = 0 \end{cases}, \\
&r_\theta(\mathcal{T}, a) = \log\left(\frac{\pi_\theta(a|\mathcal{T})}{\pi_{\mathrm{ref}}(a|\mathcal{T})}\right),
\end{aligned} \tag{9}$$

λ_1, and λ_0 are hyperparameters representing the weights of positive feedback and negative feedback, which can be set to 1 by default, π_{ref} is the model before KTO alignment, β is a hyperparameter to control the difference from the reference model, z_0 denotes the KL-divergence between π_{ref} and π_θ, θ denotes the learnable parameters in the LLM, $\sigma(\cdot)$ denotes the sigmoid function. Note that when $l = 0$, KTO reduces the corresponding $\pi_\theta(a|\mathcal{T})$, thereby enabling the LLM to learn how to avoid echo chambers from negative feedback. During training, we found that using only the KTO loss could lead to a large number of output format errors. Therefore, we added a supervised finetune loss L_s, and used a hyperparameter α to control its strength on positive samples after KTO to constrain the LLM, the final optimization object is formulated as:

$$\min_{\theta} L_{KTO} + \alpha L_s. \tag{10}$$

3.3 Grounding

During recommendation, the LLM may generate results outside the item candidate set $\mathcal{I}$. Following previous work (Bao et al., 2025; Shi et al., 2024), we use a dedicated grounding LLM for grounding. Specifically, we first use the grounding LLM to extract embeddings e_i for all items in the candidate set $\mathcal{I}$, where $i \in \mathcal{I}$. Then, once recommendation result not in the item canidate set i.e.,$a'_j \notin \mathcal{I}$, we use the grounding LLM to extract the embedding $e_{a'_j}$ for a'_j (the recommendation result at step j). The grounded recommendation result is then generated by finding the nearest item in the item candidate set $\mathcal{I}$ using L2 distance:

$$a_j \leftarrow \arg\min_{i\in\mathcal{I}} ||e_{a'_j} - e_i||_2, \tag{11}$$

where $||.||_2$ denotes L2 distance, a_j denotes the final recommendation result. Following Shi et al. (2024), Llama2-7b (Touvron et al., 2023) is used for grounding.

4 Experiment

We conducted detailed experiments to address the following three Research Questions (RQs):

- **RQ1**: By tuning with unpaired feedback, can our proposed ULRec demonstrate a strong ability to break the echo chamber?
- **RQ2**: Is it meaningful to model users' positive and negative feedback?

- **RQ3**: Is ULRec effective with different backbones and hyper-parameters?

4.1 Experiment Setup

In this section, we introduce the setup of the simulated environment, the training data construction method used by ULRec, and some implementation details of the baselines and our proposed ULRec.

4.1.1 Simulated Environment

Following previous work (Shi et al., 2024; Gao et al., 2023a), since using real users to evaluate recommendation results is very costly, we use recommendation datasets to build simulated environments for training and evaluation. Following Shi et al. (2024), we use two datasets from different recommendation scenarios, games and books, to build simulated environments and validate the effectiveness of our proposed method. The statistical information can be found at Appendix A.3.

- **Steam** (Kang and McAuley, 2018) is a dataset from a gaming scenario. We filter out users and items with fewer than five interactions. User ratings are set to 5 if the gameplay time exceeds 3 hours; otherwise, the rating is set to 2.

- **Amazon Book** (Ni et al., 2019) is a book recommendation dataset, specifically the "book" subset from Amazon Review dataset[1]. It contains book review records on Amazon from 1996 to 2018, with ratings ranging from 1 to 5. Users and items with fewer than 90 interactions are filtered out.

• **Simulated Environment Details**. The simulated environment needs to provide two important real-time feedback signals: reward and item similarity. Following the setup in BiLLP (Shi et al., 2024), we first split the dataset into training and testing parts by time, i.e., $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$, to ensure a distinction between the training and testing environments and to better reflect real-world scenarios. For both $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$, we use DeepFM (Guo et al., 2017) to fit the data and obtain the corresponding user and item embeddings, $e_u^{\mathcal{D}}$ and $e_i^{\mathcal{D}}$, where $u \in \mathcal{U}$, $i \in \mathcal{I}$. With these embeddings, we can: 1) use DeepFM to compute online rewards between any user and item, and 2) calculate the similarity between two items using the L2 distance between their embeddings. In this way, all necessary components for environment simulation are constructed.

[1] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

4.1.2 Training dataset construction

Since our goal is to explore whether user feedback can help the LLM learn information related to echo chamber and thus enhance its ability to break them, we use offline data generated from interactions between the LLM with very limited environment priors and the environment when constructing training samples. This allows us to verify whether the collected feedback from interactions can help the model learn to break echo chamber without knowing the specifics of the environment. Specifically, we use LLaMA-3-70B[2] (Grattafiori et al., 2024) to interact with 2,000 users in the training environments of two simulated environments using the ReAct (Yao et al., 2023) framework, and construct two training datasets based on these interactions. Details of the prompt used by LLaMA-3-70B to construct the training dataset, as well as some data processing details and examples of training samples, can be found in Appendix A.5. We then filtered out episodes with successful interactions less than or equal to 1 to ensure the quality of the data.

4.1.3 Baselines

We mainly compare two types of baselines. The first type consists of traditional reinforcement learning methods based on RL, including classic approaches such as BCQ (Fujimoto et al., 2019), CQL (Kumar et al., 2020), CRR (Wang et al., 2020), DQN (Mnih et al., 2013), SQN (Xin et al., 2020), and A2C (Mnih et al., 2016), as well as DORL (Gao et al., 2023a), a State-Of-The-Art (SOTA) method specifically designed to break echo chambers in traditional recommender systems. We also compare with BiLLP (Shi et al., 2024), a SOTA method for breaking the echo chamber in long-term LLM-based recommenders. "Base" refers to using the backbone LLM without fine-tuning, making recommendations in the same manner as ULRec.

4.1.4 Evaluation

We evaluate the recommendation results from these methods in the test environment. The interaction is terminated if any of the following exit conditions are triggered: the user rating falls below the threshold τ_r, the distance between any of the last W recommended items is below the threshold τ_d, or the model recommends an item which has already

[2] https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct

Table 1: The performance comparison between our method, traditional methods, and LLM-based methods is presented. "Len" denotes the interaction length, and "Traj Reward" represents the cumulative reward for the entire trajectory. Both metrics are better when higher. We use **bold** to indicate the best-performing results.

	BCQ	CQL	CRR	DQN	SQN	A2C	DORL	BiLLP	Base	ULRec
Steam Len ↑	3.15	2.08	11.23	5.04	3.30	10.13	8.02	9.26	2.38	**18.09**
Steam Traj Reward ↑	13.24	10.03	46.14	22.32	14.66	40.49	32.85	41.62	10.39	**83.66**
Amazon Len ↑	3.97	6.89	3.65	3.26	2.41	5.74	5.22	6.18	1.66	**10.42**
Amazon Traj Reward ↑	17.75	31.55	16.65	14.66	11.12	26.30	23.75	26.90	7.12	**46.10**

Table 2: Ablation study. **Bold** means the best result. "ULRec-P" denotes a variant of ULRec that learns positive feedback directly through SFT only.

	Steam		Amazon	
Mehtod	Len ↑	Traj Reward ↑	Len ↑	Traj Reward ↑
Base	2.38	10.39	1.66	7.12
ULRec-P	11.87	54.99	8.65	38.58
w/o SFT	0.27	0.99	4.09	18.17
w/o KTO	13.30	61.33	8.83	39.75
ULRec	**18.09**	**83.66**	**10.42**	**46.10**

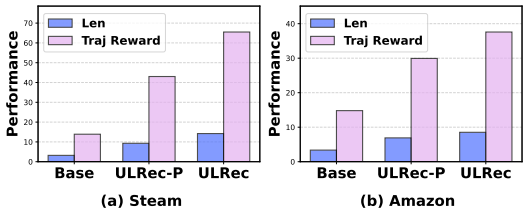


Figure 3: Results using the different backbone. We replace the backbone with Qwen2.5-7B and report the results of the "Len" and "Traj Reward" metrics on two datasets. Higher values are better for both metrics.

been recommended before. Echo chamber performance is measured by "Len" (interaction length before exit) and "Traj Reward" (cumulative reward), with higher values indicating better results. See Section 2.2 for details. We randomly selected 100 users from the test environment for evaluation. For the simulated environment, we set $W = 4$, rating threshold $\tau_r = 2$ for all environments, distance threshold $\tau_d = 50$ for Steam, and $\tau_d = 15$ for Amazon following (Shi et al., 2024).

4.1.5 Implementation Details

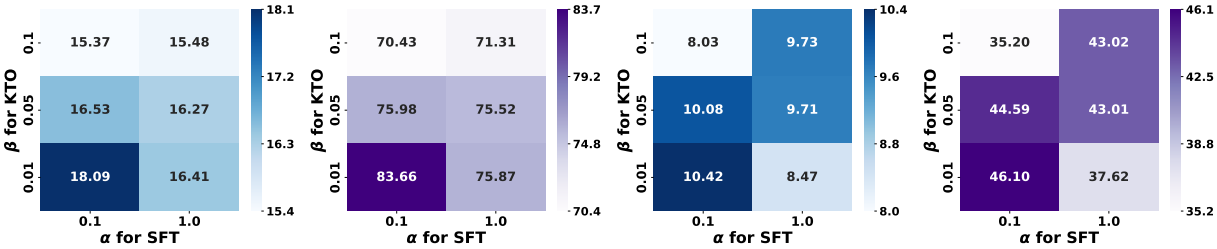
For the main results, we report the average of three experimental runs to reduce the impact of randomness. In implementing ULRec, we choose Llama-3-8B[3] (Grattafiori et al., 2024) as the default backbone LLM. We used the Transformers (Wolf et al., 2020) for SFT, the TRL (von Werra et al., 2020) for the KTO, and the vLLM (Kwon et al., 2023) for LLM inference. Both SFT and KTO were trained for 3 epochs. The learning rate was set to 5×10^{-6}, and β was tuned over $\{0.01, 0.05, 0.1\}$ as recommended in the original KTO paper (Ethayarajh et al., 2024). The SFT loss weight α was tuned over $\{0.1, 1\}$. Full-parameter fine-tuning was conducted on 4 A100 80G GPUs, and ULRec process on one dataset consumed approximately 10 GPU hours. The temperature of all LLM-based methods is set to 0.5 during inference. Traditional reinforcement learning methods are implemented using the EasyRL4Rec (Yu et al., 2024) library. Due to page limitations, more details can be found in Appendix.A.2. The code of ULRec can be found at https://github.com/jizhi-zhang/ULRec.

[3] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

4.2 Overall Performance (RQ1)

Table 1 presents the comparison results between our method and various baselines. We have the following main observations:

- Compared to traditional methods, our approach consistently achieves higher performance. This indicates that the proposed ULRec is highly effective at leveraging feedback to break echo chamber.
- BiLLP performs relatively weakly in some cases. This is because, in our environment simulation, we introduced a realistic new rule where users become bored if items recommended by previous models are recommended again, which reduces the model's ability to hack the environment by repeatedly outputting the same items recommended several turns before. Although BiLLP's metrics decrease, it still outperforms almost all traditional methods and shows relatively stable performance on both datasets.
- Some traditional RL-based methods achieve good performance but lack stability. For ex-



(a) Steam Len. (b) Steam Traj Reward. (c) Amazon Len. (d) Amazon Traj Reward.

Figure 4: The impact of different hyperparameters on the experimental results of ULRec using KTO to model unpaired negative feedback across two datasets. Deeper colors indicate better performance, with the y-axis representing the KTO hyperparameter β and the x-axis representing the weight of the SFT loss α.

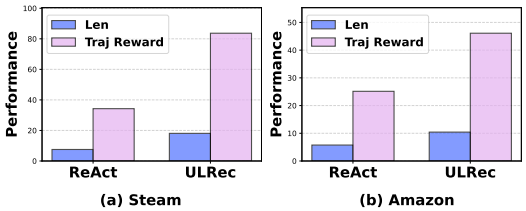


Figure 5: Comparison of the performance of models used for data generation. ReAct represents the results of interactions using LLaMA-3-70B in the ReAct manner. We measured the performance of our method on two datasets using two metrics.

ample, CRR and CQL perform well on one dataset but poorly on the other. In contrast, LLM-based methods show more consistent performance across both datasets, demonstrating the strong generalization ability and robustness of LLMs.

4.3 Ablation Study (RQ2)

In this section, we conduct ablation studies on ULRec to demonstrate the effectiveness of its different components as demonstrated in Table.2. "Base" refers to the results of the model without any tuning, "ULRec-P" represents the results after fine-tuning using positive feedback, "w/o SFT" indicates the results when the SFT loss is not included during the second step of aligning with negative feedback, and "w/o KTO" shows the results of the second step with only SFT and no KTO loss. "ULRec" is our final proposed solution. We have the following observations:

- Using positive feedback enhances the LLM's ability to break out of echo chamber. It can be observed that "ULRec-P" consistently outperforms Base on both datasets, indicating that our proposed ULRec can effectively leverage information from positive feedback and demonstrating the importance of modeling positive feedback.
- Introducing negative feedback is meaningful. We observe that ULRec consistently outperforms models trained only with SFT and those with additional SFT after SFT. This demonstrates that modeling negative feedback through KTO is effective and highlights the necessity of incorporating negative feedback in long-term LLM-based recommendation tuning.
- When using negative feedback, if only KTO is applied without SFT constraints, the model shows a significant performance drop. This is because KTO disrupts the output format, preventing correct item recommendations. This demonstrates the necessity of incorporating SFT during KTO optimization to stabilize the training process.

4.4 In-depth Analysis (RQ3)

In this section, we analyze the effectiveness of ULRec from two perspectives: different backbones and parameter sensitivity.

4.4.1 Effect of Backbones

We present the experimental results of using different the backbone, i.e. Qwen2.5-7B[4] (Yang et al., 2024), for ULRec in Figure 3. The following observations can be made:

- After changing the backbone, ULRec still demonstrates strong performance compared to the base (non-finetuned) model. This validates the broad applicability of our proposed ULRec.
- ULRec still shows significant improvement over SFT. This further demonstrates the importance of using unpaired feedback to help break echo chamber across different backbones.

[4] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

Since we used the pure ReAct (Yao et al., 2023) format of LLaMA-3-70B[5] (Grattafiori et al., 2024) to construct the training data, we further compare the performance of our proposed ULRec with the ReAct model used for data generation, both directly evaluated on the test set. This demonstrates that by learning from users' unpaired feedback, the 8B model can achieve results comparable to or even better than the 70B model, as shown in Figure 5 , further demonstrating that the benefit comes from the feedback signal itself rather than distillation from a larger model, highlighting the importance of learning from unpaired user feedback while breaking the echo chamber effect.

4.4.2 Hyper-Parameters Analysis

We focus on two hyperparameters in ULRec: β for KTO and α for the supervised finetuning loss reweighting. Due to resource constraints, we conducted experiments on both datasets with $\beta \in \{0.01, 0.05, 0.1\}$ and $\alpha \in \{0.1, 1\}$. The results are demonstrated in Figure 4, with following main observations:

- When the SFT weight α is small, decreasing β tends to improve performance. This is reasonable because, during KTO optimization, a smaller β means the model focuses more on negative samples in the optimization process. For example, when the reward of a negative feedback is close to 0, a smaller β effectively increases its weight during optimization, making it more likely for the model to further reduce its reward. However, a smaller β means focusing more on negative samples with near-zero rewards, increasing sensitivity to noise in negative feedback. Given that Figure 4 shows strong β sensitivity, showing that β may need to be carefully tuned.

- As the SFT weight α decreases, ULRec's performance tends to improve. This further highlights the importance of modeling both types of feedback. If the SFT weight α is too large, the model focuses more on optimizing positive feedback, which restricts the optimization space for negative feedback. However, removing the SFT weight entirely risks causing output format collapse after training, as shown in Table 2. The trade-off between negative feedback modeling and format collapse also makes α an important hyperparameter.

[5] https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct

5 Related Work

5.1 LLMs for Recommendation

LLMs' possess powerful capabilities, such as open world knowledge and the ability to quickly learn new tasks, has inspired numerous applications in the field of recommender systems (Wu et al., 2024b; Lin et al., 2025a; Zhao et al., 2024; Zhang et al., 2025), which can mainly be divided into two major approaches. The first approach uses LLMs directly in recommendation systems without fine-tuning (Zhang et al., 2023a; Hou et al., 2024), using their abilities such as in-context learning (Gao et al., 2023c; Liu et al., 2023) to do recommendation. However, LLMs are not specially designed for recommendation, which limits their recommendation performance when used directly without fine-tuning (Bao et al., 2023). The second approach fine-tunes LLMs for recommendation tasks, enabling them to generate recommendations directly. This has achieved notable short-term gains in scenarios like CTR, reranking, and all-ranking (Bao et al., 2023; Wu et al., 2024a; Liao et al., 2024; Zhang et al., 2023b; Bao et al., 2025; Wang et al., 2025a; Fan et al., 2025), but pays less attention to long-term benefits. A closely related work, SERAL (Xi et al., 2025), fine-tunes LLMs for serendipity recommendation. Unlike SERAL, we focus on using unpaired feedback from users' multi-turn interactions in interactive scenarios to fine-tune the LLM to break echo chambers, and evaluate through an interactive recommendation manner.

5.2 Interactive Recommendation

To enhance the online interaction capability of recommendation systems and users, researchers have modeled the interactive recommendation as a Markov decision process (MDP) and employed RL algorithms to maximize long-term user engagement benefits (Dulac-Arnold et al., 2015; Ie et al., 2019; Zhao et al., 2018; Wang et al., 2025b). CIRS (Gao et al., 2023b) acquires a causal user model from historical data, thereby enhancing the strategic planning of RL policies. DORL (Gao et al., 2023a) mitigates the Matthew Effect in offline RL to optimize sustained user interaction. These traditional RL algorithms require training from scratch, which is inefficient and suffers from poor generalization. In contrast, BiLLP (Shi et al., 2024) leverages the general capabilities of LLMs and employs a bi-level learnable planner framework to enhance online user interaction. However, optimizing prompts alone

does not sufficiently enable LLMs to learn recommendation tasks effectively, which consequently limits its performance. In contrast, our method focuses on directly using users' unpaired feedback to fine-tune LLMs, thereby efficiently enhancing their long-term recommendation ability. Moreover, unlike the implicit feedback setting in classical recommender systems which requires artificially constructing negative samples (e.g., through negative sampling (Zhang et al., 2013; Rendle and Freudenthaler, 2014)), our unpaired negative feedback directly originates from user behavior (simulated by the environment) and is capable of capturing the echo chamber effect.

6 Conclusion

In this work, we revisit the long-standing echo chamber problem in recommendation systems from the perspective of LLMs, emphasizing the importance of fine-tuning by effectively modeling users' time-varying feedback to enhance long-term user satisfaction. We identify an under-explored yet critical challenge: how to learn from unpaired feedback, especially negative signals from users' non-engagement behaviors. To address this, we propose ULRec, a lightweight framework that integrates unpaired feedback into the LLM fine-tuning process, which utilizes the KTO algorithm to handle implicit negative signals without relying on paired preference data. Our extensive experiments demonstrate that ULRec significantly improves the long-term recommendation quality of recommendation systems, proving the value of explicitly modeling unpaired feedback. This work not only deepens our understanding of long-term feedback mechanisms in LLM-based recommender systems but also suggests new directions for future research on long-term recommendation strategies. Moreover, although we have optimized the simulation environment based on previous work (Shi et al., 2024; Gao et al., 2023b) to better adapt to real-world scenarios, there still inevitably exists a gap between the simulation and the real environment. How to further reduce the discrepancy between the real world and the simulated environment remains a question worthy of exploration.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (62402470, 62272437), Postdoctoral Fellowship Program of CPSF (GZC20241643) and Anhui Postdoctoral Scientific Research Program Foundation (No.2025B1063).

Limitations

This paper mainly focuses on fine-tuning LLMs with user's unpaired feedback to enhance their long-term recommendation ability thereby breaking the echo chamber. However, our work has the following limitations: 1. Due to the high cost of real-world online experiments, we use simulation to demonstrate the effectiveness of our method. In the future, leveraging unpaired feedback in real-world LLM-based recommendation systems to break the echo chamber would further validate our approach. 2. Our experiments are mainly conducted on models with around 8B parameters. Extending the study to models of different sizes to verify the effectiveness of unpaired feedback in breaking echo chamber would provide stronger support for our method.

Ethical Considerations

In this paper, we propose the ULRec framework, which enhances the LLM-based recommender's long-term recommendation ability in breaking echo chamber by modeling users' unpaired feedback, without introducing new ethical issues. Regarding data, we use publicly available datasets that do not contain sensitive user information, or where such information has been highly anonymized to ensure user privacy is not compromised. ULRec itself does not introduce additional bias; however, when deploying ULRec, attention should be paid to potential risks arising from inherent biases in LLMs. Therefore, if ULRec is to be deployed in real-world scenarios, we recommend conducting thorough risk assessments and being mindful of potential risks.

References

Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. 2025. A bi-step grounding paradigm for large language models in recommendation systems. ACM Transactions on Recommender Systems, 3(4):1–27.

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.

Shihao Cai, Jizhi Zhang, Keqin Bao, Chongming Gao, Qifan Wang, Fuli Feng, and Xiangnan He. 2025. Agentic feedback loop modeling improves recommendation and user simulation. In Proceedings of the 48th International ACM SIGIR conference on Research and Development in Information Retrieval, pages 2235–2244.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. arXiv preprint arXiv:2401.08281.

Gabriel Dulac-Arnold, Richard Evans, Peter Sunehag, and Ben Coppin. 2015. Reinforcement learning in large discrete action spaces. CoRR, abs/1512.07679.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. arXiv preprint arXiv:2402.01306.

Chenxiao Fan, Chongming Gao, Wentao Shi, Yaxin Gong, Zihao Zhao, and Fuli Feng. 2025. Fine-grained list-wise alignment for generative medication recommendation. Advances in Neural Information Processing Systems.

Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In International conference on machine learning, pages 2052–2062. PMLR.

Chongming Gao, Ruijun Chen, Shuai Yuan, Kexin Huang, Yuanqing Yu, and Xiangnan He. 2025a. Sprec: Self-play to debias llm-based recommendation. In Proceedings of the ACM on Web Conference 2025, pages 5075–5084.

Chongming Gao, Mengyao Gao, Chenxiao Fan, Shuai Yuan, Wentao Shi, and Xiangnan He. 2025b. Process-supervised llm recommenders via flow-guided tuning. In Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 1934–1943.

Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023a. Alleviating matthew effect of offline reinforcement learning in interactive recommendation. In Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval, pages 238–248.

Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2023b. Cirs: Bursting filter bubbles by counterfactual interactive recommender system. ACM Transactions on Information Systems, 42(1):1–27.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023c. Chat-rec: Towards interactive and explainable llms-augmented recommender system. arXiv preprint arXiv:2303.14524.

Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, et al. 2021. Towards long-term fairness in recommendation. In Proceedings of the 14th ACM international conference on web search and data mining, pages 445–453.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. arXiv preprint arXiv:1703.04247.

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 European Conference on Information Retrieval, pages 364–381. Springer.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. In IJCAI, pages 2592–2599. ijcai.org.

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

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. Advances in neural information processing systems, 33:1179–1191.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles, pages 611–626.

Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, Juntao Tan, Shuchang Liu, and Yongfeng Zhang. 2023a. Fairness in recommendation: Foundations, methods, and applications. ACM Transactions on Intelligent Systems and Technology, 14(5):1–48.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023b. Towards general text embeddings with multi-stage contrastive learning. arXiv preprint arXiv:2308.03281.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 1785–1795.

Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, et al. 2025a. How can recommender systems benefit from large language models: A survey. ACM Transactions on Information Systems, 43(2):1–47.

Siyi Lin, Chongming Gao, Jiawei Chen, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. 2025b. How do recommendation models amplify popularity bias? an analysis from the spectral perspective. In Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, WSDM '25, page 659–668, New York, NY, USA. Association for Computing Machinery.

Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. arXiv preprint arXiv:2304.10149.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pages 1928–1937. PmLR.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

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

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In Proceedings of the 7th ACM international conference on Web search and data mining, pages 273–282.

Wentao Shi, Xiangnan He, Yang Zhang, Chongming Gao, Xinyue Li, Jizhi Zhang, Qifan Wang, and Fuli Feng. 2024. Large language models are learnable planners for long-term recommendation. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 1893–1903.

Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. 2019. How good your recommender system is? a survey on evaluations in recommendation. International Journal of Machine Learning and Cybernetics, 10:813–831.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Bohao Wang, Feng Liu, Jiawei Chen, Xingyu Lou, Changwang Zhang, Jun Wang, Yuegang Sun, Yan Feng, Chun Chen, and Can Wang. 2025a. Msl: Not all tokens are what you need for tuning llm as a recommender. Preprint, arXiv:2504.04178.

Jie Wang, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2025b. Large language model driven policy exploration for recommender systems. In Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, pages 107–116.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. 2020. Critic regularized regression. Advances in Neural Information Processing Systems, 33:7768–7778.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2022. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. IEEE Transactions on Knowledge and Data Engineering, 35(5):4425–4445.

Likang Wu, Zhaopeng Qiu, Zhi Zheng, Hengshu Zhu, and Enhong Chen. 2024a. Exploring large language

model for graph data understanding in online job recommendations. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 9178–9186.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024b. A survey on large language models for recommendation. World Wide Web, 27(5):60.

Yunjia Xi, Muyan Weng, Wen Chen, Chao Yi, Dian Chen, Gaoyang Guo, Mao Zhang, Jian Wu, Yuning Jiang, Qingwen Liu, et al. 2025. Bursting filter bubble: Enhancing serendipity recommendations with aligned large language models. arXiv preprint arXiv:2502.13539.

Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-supervised reinforcement learning for recommender systems. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, pages 931–940.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. arXiv e-prints, pages arXiv–2412.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR).

Yuanqing Yu, Chongming Gao, Jiawei Chen, Heng Tang, Yuefeng Sun, Qian Chen, Weizhi Ma, and Min Zhang. 2024. Easyrl4rec: An easy-to-use library for reinforcement learning based recommender systems. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 977–987.

Jizhi Zhang, Keqin Bao, Wenjie Wang, Yang Zhang, Wentao Shi, Wanhong Xu, Fuli Feng, and Tat-Seng Chua. 2025. Envisioning recommendations on an llm-based agent platform. Communications of the ACM, 68(5):48–57.

Jizhi Zhang, Keqin Bao, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023a. Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation. In Proceedings of the 17th ACM Conference on Recommender Systems, pages 993–999.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023b. Recommendation as instruction following: A large language model empowered recommendation approach. ACM Transactions on Information Systems.

Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pages 785–788.

Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In RecSys, pages 95–103. ACM.

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, et al. 2024. Recommender systems in the era of large language models (llms). IEEE Transactions on Knowledge and Data Engineering.

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.

A Appendix

A.1 Stablize KTO Training

We adopted several strategies to ensure the stability of KTO during the training process. The main aspects are as follows:

- 1. Differentiating between format tokens and non-format tokens. Note that during each interaction, if the LLM is to make a recommendation, it needs to output a "recommend[·]" format, where the item to be recommended is placed inside the brackets []. If KTO is directly applied for optimization, the tokens "recommend[ ]" may appear with excessively high frequency. This can lead to over-focusing on these format tokens when optimizing negative samples, causing two issues: first, potential format collapse making the output unrecognizable; second, the model neglecting the valuable parts within [] that should be optimized. Therefore, during the KTO optimization process, we masked out the format tokens and retained only the tokens corresponding to the items (non-format tokens). For SFT loss, since we need to ensure format consistency, we optimize both the format tokens and non-format tokens together.

- 2. We slightly adjusted the output format to "recommend[item]". If the item directly follows the opening bracket "[" during tokenization, it could easily cause the "[" to stick to the subsequent token, leading to confusion between format and non-format tokens and thus destabilizing the output format. To address this, we prevent the issue by adding spaces before and after the item.

Table 3: Dataset statistics. "Int." is short for "interactions".

Datasets	User	Item	Train Int.	Test Int.
Steam	6,012	190,365	1,654,303	958,452
Amazon	3,109	13,864	339,701	137,948

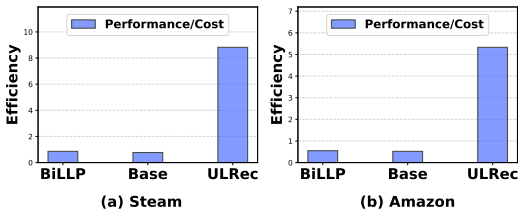


Figure 6: Efficiency Superiority Comparison. The efficiency advantage is evaluated by the ratio of long-term reward performance metric ("Traj Reward") to the cost per recommendation (i.e., the number of tokens required for a single interactive recommendation), where a higher ratio indicates better efficiency.

A.2 Additional Implementation Details

Here, we provide more implementation details. The warm-up ratio was set to 0.1. For BiLLP, we use the same backbone as ULRec, namely Llama-3-8B[6] (Grattafiori et al., 2024). Due to resource constraints, we did not use remote APIs, but instead chose the high-performance gte-Qwen2-1.5B-instruct[7] (Li et al., 2023b) for getting embedding for retrieval, and performed vector retrieval with FAISS (Douze et al., 2024) and SentenceTransformers (Reimers and Gurevych, 2020). For traditional methods, following previous work, we train 100,000 episodes in the training simulation environment to ensure sufficient training for these models. Following BiLLP (Shi et al., 2024), when constructing the simulation environment, we filter out users with fewer than 90 interactions in the Amazon dataset to ensure the accuracy of user preference modeling in the simulated environment. For recommendation, we provide the recommender with the last 15 interactions of each user, a setting that is also consistent with BiLLP (Shi et al., 2024).

A.3 Dataset statistics

We present the statistics of the datasets used to build the simulated environment in Table 3; this information is the same as Shi et al. (2024).

A.4 Addtional Results

We design experiments to demonstrate that ULRec achieves high efficiency. As shown in Figure 6, considering that autoregressive token generation incurs substantial computational costs in practical LLM deployments, we evaluate ULRec's efficiency by measuring the long-term reward attainable per unit interaction cost (expressed as the total number of tokens involved per recommendation). It can be observed that ULRec exhibits a significant advantage over both Base and BiLLP on two datasets.

A.5 Training dataset construction.

A.5.1 Traning data example

We construct training samples using interaction data generated by ReAct, removing the thought part and retaining only the action part. We also assume that after fine-tuning, the model is likely to output recommendable items, so we replace the pre-grounded items with grounded items to complete the construction of the training data. An example of a training sample can be found in Figure 7. Here, "Input" refers to the part fed into the model, while "Target" is the expected output during fine-tuning. Negative feedback can be processed similarly and then organized into the TRL's KTO format for input to KTO optimization.

A.5.2 Prompt for constructing training dataset.

Table 8 shows the ReAct-style prompt we used to obtain offline training data by interacting with the training environment. In the prompt, we provide the LLM with minimal information about the environment, and the case is included only to demonstrate the required output format. We expect the model to discover the answer through its own exploration and output the result in the correct format.

[6] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[7] https://huggingface.co/Alibaba-NLP/gte-Qwen2-1.5B-instruct

Training data example.

Input: Solve a recommendation task with interleaving Action, Observation steps.
Action can be the following types:
(1) recommend[item], which recommend an item to user based on user's interest. Your goal is to meet the user's interest as much as possible and make recommendations to users as many times as possible. Note that if the user is not satisfied with your recommendations, he will quit and not accept new recommendations

You may take as many steps as necessary.
Here are some examples:

Question: The user's viewing history is ['Pretty in Pink', Öne Flew Over the Cuckoo's Nest, 'Ransom', 'Saving Private Ryan', 'X-Men', 'Coyote Ugly', 'The Patriot', 'Me, Myself and Irene', 'Gone in 60 Seconds', 'The Perfect Storm', 'Titanic', 'The Haunting', 'Bedknobs and Broomsticks', 'Clerks', 'The Matrix', 'The Shawshank Redemption', 'Vacation', 'Father of the Bride', 'Wallace & Gromit: The Best of Aardman Animation', 'Back to the Future', 'Fight Club'], please recommend item for this user
Action 1: recommend[Forrest Gump]
Observation 1: Episode continue, reward=0.30697035862193367
Action 2: recommend[The Godfather]
Observation 2: Episode finished, reward=0.49717313011547304

(END OF EXAMPLES)
Question: The user's viewing history is ['Torchlight', T̈om Clancy's Rainbow Six® Vegas 2, 'F.E.A.R.', 'Torchlight II', 'Just Cause 2', 'Just Cause 2', B̈aldur's Gate: Enhanced Edition, B̈aldur's Gate: Enhanced Edition, 'LEGO® Marvel™ Super Heroes', 'Rise of Nations: Extended Edition', 'Warhammer® 40,000: Dawn of War® - Soulstorm', 'Age of Empires II HD', 'Age of Empires II HD', T̈om Clancy's Ghost Recon®, 'Dungeon Siege'], please recommend item for this user
Action 1: recommend[Dungeon Siege]
Observation 1: Episode continue, reward=1.856873869895935
Action 2:

Target: recommend[Borderlands]

Figure 7: Training data example.

Prompt for constructing training dataset.

Prompt:

Solve a recommendation task with interleaving Thought, Action, Observation steps. Thought can reason about the current situation and current user interest, and Action can be the following types: (1) recommend[item], which recommend an item to user based on user's interest. Your goal is to meet the user's interest as much as possible and make recommendations to users as many times as possible. Note that if the user is not satisfied with your recommendations, he will quit and not accept new recommendations. You may take as many steps as necessary.

Here are some examples:

Question: The user's viewing history is ['Pretty in Pink', Öne Flew Over the Cuckoo's Nest, 'Ransom', 'Saving Private Ryan', 'X-Men', 'Coyote Ugly', 'The Patriot', 'Me, Myself and Irene', 'Gone in 60 Seconds', 'The Perfect Storm', 'Titanic', 'The Haunting', 'Bedknobs and Broomsticks', 'Clerks', 'The Matrix', 'The Shawshank Redemption', 'Vacation', 'Father of the Bride', 'Wallace & Gromit: The Best of Aardman Animation', 'Back to the Future', 'Fight Club'], please recommend item for this user

Thought: The user seems to enjoy a mix of drama, action, and comedy. They also seem to appreciate classic films. I should recommend a movie that fits these categories.

Action: recommend[Forrest Gump]

Observation: Episode continue, reward=0.30697035862193367

Thought:The user seems to have responded positively to the previous recommendation. They seem to enjoy movies with a strong narrative and compelling characters. I should recommend another movie in a similar vein.

Action: recommend[The Godfather]

Observation: Episode finished, reward=0.49717313011547304

(END OF EXAMPLES)

Question:

Figure 8: Prompt for constructing training dataset.