DARLR: Dual-Agent Offline Reinforcement Learning for Recommender Systems with Dynamic Reward

Yi Zhang
uqyzha91@uq.edu.au
The University of Queensland
CSIRO DATA61
Brisbane, Australia

Ruihong Qiu
r.qiu@uq.edu.au
The University of Queensland
Brisbane, Australia

Xuwei Xu
xuwei.xu@uq.edu.au
The University of Queensland
Brisbane, Australia

Jiajun Liu
jiajun.liu@csiro.au
CSIRO DATA61
The University of Queensland
Brisbane, Australia

Sen Wang
sen.wang@uq.edu.au
The University of Queensland
Brisbane, Australia

Abstract

Model-based offline reinforcement learning (RL) has emerged as a promising approach for recommender systems, enabling effective policy learning by interacting with frozen world models. However, the reward functions in these world models, trained on sparse offline logs, often suffer from inaccuracies. Specifically, existing methods face two major limitations in addressing this challenge: (1) deterministic use of reward functions as static look-up tables, which propagates inaccuracies during policy learning, and (2) static uncertainty designs that fail to effectively capture decision risks and mitigate the impact of these inaccuracies. In this work, a dual-agent framework, DARLR, is proposed to dynamically update world models to enhance recommendation policies. To achieve this, a ***selector*** is introduced to identify reference users by balancing similarity and diversity so that the ***recommender*** can aggregate information from these users and iteratively refine reward estimations for dynamic reward shaping. Further, the statistical features of the selected users guide the dynamic adaptation of an uncertainty penalty to better align with evolving recommendation requirements. Extensive experiments on four benchmark datasets demonstrate the superior performance of DARLR, validating its effectiveness. The code is available at this address.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Offline Reinforcement Learning; Recommendation Systems; Multi-agent Reinforcement Learning

ACM Reference Format:
Yi Zhang, Ruihong Qiu, Xuwei Xu, Jiajun Liu, and Sen Wang. 2025. DARLR: Dual-Agent Offline Reinforcement Learning for Recommender Systems with Dynamic Reward. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3729942

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3729942

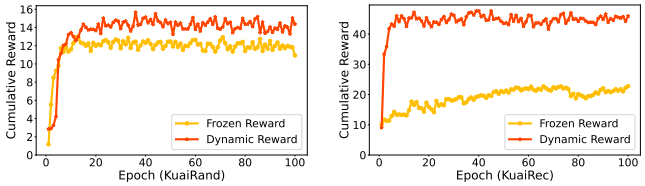

Figure 1: Recommendation policies trained from scratch with dynamic reward functions converge better than those with static reward functions on KuaiRand [19] and KuaiRec [18].

1 Introduction

Recommender systems (RecSys) achieve remarkable success in industrial scenarios such as e-commerce and social media by capturing user interests to enhance their experience [3, 4, 25]. RecSys can generally be categorized into supervised learning methods [6, 57] and reinforcement learning (RL) methods. While supervised learning treats the RecSys problem as a one-off recommendation [32, 43, 44], making it challenging to model users' long-term dynamic interests [62, 63], RL-based methods aim to optimize long-term user satisfaction through an interactive training paradigm, a direction that has recently received increased attention [10, 11].

Most RL-based RecSys methods [1, 17, 20, 66] operate in offline RL settings, where recommendation policies are learned purely from historical user interaction logs, without requiring online interactions. Given the sparsity of these offline logs, which can hardly cover all possible user-item interactions, model-based offline RL methods [24, 47, 64] adopt a strategy of training world models prior to policy learning. These world models include reward functions designed to predict user-item interaction rewards, typically trained using supervised learning, such as DeepFM [21], and are treated as *frozen look-up tables* during recommendation policy learning [20, 66]. However, these reward functions often struggle with inaccuracies, particularly for interactions that are rarely seen in the offline logs. To mitigate this, uncertainty penalties [17, 70] are

introduced to avoid inaccurately estimated item recommendations. These penalties are also *frozen* during policy learning.

Although these model-based offline RL methods for RecSys have shown promising potential in recommendation performance, their effectiveness is often hindered by: **(1) straightforward utilization** of inaccurately estimated rewards during recommendation policy learning and **(2) static uncertainty penalty** which is insufficient of capturing decision risks in RecSys. Specifically, training recommendation policies using frozen reward functions as look-up tables inevitably amplifies the impact of inaccuracies in reward functions. For instance, overestimated items are likely prioritized during policy training [16, 29], but these items often fail to meet user expectations during testing, leading to poor single-step user satisfaction and diminished long-term engagement [62]. Meanwhile, underestimated items often tend to be overlooked, further distorting the recommendation process. Static uncertainty penalties struggle to mitigate these issues, as they fail to account for the quality of reward functions and the dynamics within the policy training process in model-based offline RL for RecSys.

Since both overestimated and underestimated rewards can harm long-term user satisfaction, iteratively refining reward functions in world models becomes essential. As shown in Fig. 1, recommendation policies trained *from scratch* with refined dynamic reward functions significantly outperform those relying on static reward functions in the long-term user satisfactory on two benchmark datasets [18, 19], with dynamic reward curves demonstrating faster convergence and superior performance. This underscores the feasibility and importance of dynamic reward shaping. To achieve this, a pioneering and effective framework, *i.e.*, DARLR: **D**ual-**A**gent model-based offline **R**einforcement **L**earning for **R**eco- mmender systems, is proposed. DARLR consists of two RL agents: the ***selector*** and the ***recommender***. The selector identifies reference users based on interactions in the world models, formulating this process as an RL task with a reward model designed to balance recommendation performance and the representativeness of selected users. The recommender then aggregates reward estimations from these reference users to refine the reward functions and adaptively estimates uncertainty penalties based on both the representativeness of the selected users and the changes in reward estimations. By *continuously refining the reward estimations* of world models during the recommendation policy learning process, DARLR effectively mitigates the impact of inaccuracies in rewards. Further, as reward functions evolve, the changes of reward functions and the representativeness of selected users is dynamically updated, ensuring that the uncertainty penalty remains *dynamic* and better aligned with the offline RL for RecSys setting. The contributions are as follows:

- The limitations of inaccuracies in frozen world models within model-based offline RL for recommender systems is identified. Dynamic reward shaping is highlighted as a crucial solution to address this challenge.
- A novel dual-agent offline RL recommender, DARLR, is proposed, which consists of a selector to identify reference users and a recommender to iteratively refine reward functions and estimate uncertainty penalties.
- Extensive experiments are conducted on four recommendation benchmarks, the superiority of DARLR demonstrates the effectiveness of dynamic reward shaping.

2 Preliminaries

• **Reinforcement Learning** tasks can be formulated as Markov Decision Processes (MDPs) denoted by 5-element tuples $G =< \mathbf{S}, \mathbf{A}, T, r, \gamma >$ [51], where $\mathbf{S}$ indicates the state space and each $\mathbf{s} \in \mathbf{S}$ refers to a state in the state space. $\mathbf{A}$ is the action space of the current MDP and $\mathbf{a} \in A$ represents a specific action. T is the transition function, modeling the dynamics of the environment by $T(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}}) = P(\mathbf{s_{t+1}}|\mathbf{s} = \mathbf{s_t}, \mathbf{a} = \mathbf{a_t})$. R is the reward function, indicating the immediate reward after taking action $\mathbf{a_t}$ at state $\mathbf{s_t}$ by $r_t = R(\mathbf{s_t}, \mathbf{a_t})$. γ is the discount factor balancing the current immediate reward and future return. The objective of RL is to learn a policy π that can maximize the long-term return: $G_t = \sum_{\mathbf{s}=\mathbf{s_0}}^{s_T} \gamma^t r(\mathbf{s_t}, \mathbf{a_t})$, where $\mathbf{s_0}$ stands for the initial state, s_T is the last state and t is the timestep. In addition, the state value function and state-action value function of a given policy π are $V_\pi(\mathbf{s}) = \mathbf{E}_\pi[G_t|\mathbf{s} = \mathbf{s_t}]$ and $Q_\pi(\mathbf{s}, \mathbf{a}) = \mathbf{E}_\pi[G_t|\mathbf{s} = \mathbf{s_t}, \mathbf{a} = \mathbf{a_t}]$.

• **Offline Reinforcement Learning** is a practical solution to overcome the requirement of a large number of online interactions from general RL methods [31]. The offline dataset $D = \{(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}}, r_t)\}$ is collected by one or more behavior policies denoted by π_{B_i}. Model-based offline RL methods usually simulate the environment using offline datasets, denoted as $G' =< \mathbf{S}, \mathbf{A}, \hat{T}, \hat{R}, \gamma >$, where $\hat{T}$ and $\hat{R}$ are the estimated transition and reward function. Then, agents interact with the learned world model to sample trajectories $\tau = \{(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}}, \hat{r_t})'\}$. Recent approaches [65] incorporate uncertainty as penalties in reward functions to encourage conservatism by $r = \hat{r} - \lambda_U P_U$. This research direction exhibits substantial potential within RecSys, particularly because the inherent sparsity in offline user-item interactions significantly amplifies the risks associated with exploring extensive action spaces.

• **Formulating Recommendation as a Reinforcement Learning Problem** is often referred to as RL4RS. Recalling the 5-element tuple of MDP, each state $\mathbf{s} \in \mathbf{S}$ corresponds to a status that the RecSys needs to recommend a specific item to a user. A $\mathbf{s}$ usually comprises the user's side information, such as gender and age, and recent interaction history [22]. An action $\mathbf{a}$ corresponds to one item in the item set denoted by $\mathbf{A}$. The reward $r(s_t, a_t)$ is derived from the feedback after recommending item $\mathbf{a_t}$ in state $\mathbf{s_t}$, and it varies depending on the dataset, such as watch time on a short video platform or ratings on a product review site. As for the transition function, it serves as a state tracker, capturing the sequence of states autoregressively: $\mathbf{s}' = f(\mathbf{s}, \mathbf{a}, r)$, and is often implemented using sequential models [26, 59, 75]. Ultimately, the objective of the RecSys is to learn a policy π that maximizes the long-term user experience, expressed as $\arg\max_\pi \mathbf{E}_{\tau\sim\pi}[\sum_{(\mathbf{s},\mathbf{a})\in\tau} \gamma^t r(\mathbf{s}, \mathbf{a})]$, where τ are trajectories generated by policy π. This paper focuses on model-based offline RL, adhering to state-of-the-art approaches, such as DORL [17] and ROLeR [70], to comprehensively describe the learning process. These approaches typically involve two phases: learning the world model from offline interaction, followed by training the recommendation policy within this simulated environment.

3 Method

Generally, DARLR comprises three main modules: the world model, the selector, and the recommender, as shown in Figure 2.

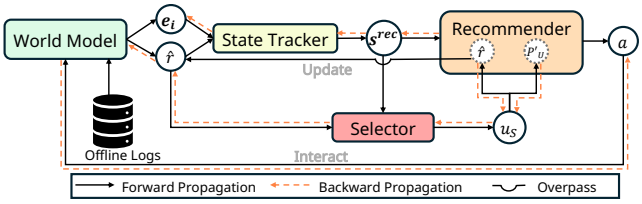

Figure 2: Overall Framework of DARLR with three main modules: a world model, a selector agent and a recommender agent. The gradient flow will influence the training of both agents, as well as the update of the world model.

3.1 World Model Learning

Intuitively, a world model is a simulated recommendation system. It is learned with offline data through supervised learning. Specifically, during world model learning, users and items are encoded from IDs and features into embeddings. The reward function in the world model is trained through a predictive model such as DeepFM [21]. In addition, some types of uncertainty penalty [20, 66] and entropy penalty [17] are also calculated in this stage.

Item Embedding, $\mathbf{e}_i \in \mathbb{R}^{d_I}$, is derived from the item ID and part of item features such as tags for music or categories for products.

$$\mathbf{e}_i = f_{\text{Item}}(i_{id}, \mathbf{F}^i_{id}), \tag{1}$$

where f_{Item} represents the item encoder, and i_{id}, $\mathbf{F^i}$ denotes the item id and item features, respectively.

User Embedding, $\mathbf{e}_u \in \mathbb{R}^{d_U}$, analogous to item embedding, utilizing user ID and auxiliary information.

$$\mathbf{e}_u = f_{\text{User}}(u_{id}, \mathbf{F}^u_{id}), \tag{2}$$

where f_{User} is the user encoder, and u_{id}, $\mathbf{F}^u$ represents the user id and the user features, respectively. $\mathbf{e}_u \in \mathbb{R}^{d_U}$ is time-invariant here.

Reward Prediction is a key output of the world model. In RecSys, user-item interaction logs can be regarded as a sparse matrix with each row corresponding to one user and each column corresponding to one item. The $(i, j)-$th element in the matrix represents user i's feedback about item j. After supervised learning, the sparse matrix is filled with predicted feedback predictions, *i.e.,* $\hat{r} \in \mathbb{R}$. Though these predictions complete the reward function, especially for the unseen user-item interactions in offline data, the influence of the accuracy of the reward predictions on the recommendation policy learning still needs exploring. DORL [17] and CIRS [20] average predictions from multiple world models (WMs) to estimate rewards by:

$$\hat{r} = \frac{1}{K}\sum_{j=1}^{K} \text{WM}_j(\mathbf{e}_i, \mathbf{e}_u), \tag{3}$$

where K is the number of world models, and WM_j represents the j-th world model.

Uncertainty Penalty is a crucial component in offline RL, often utilized to evaluate the risk associated with specific actions. Some approaches [27, 65] utilize an ensemble of world models to estimated the uncertainty. In the prior work, DORL [17], it uses Gaussian Probabilistic Model (GPM) to facilitate the building of world models, with the uncertainty of an interaction x calculated as $P_U(x) := \max_{k\in E} \sigma^2_{\theta_k}$, where k indexes the world model ensemble E and $\sigma^2_{\theta_k}$ represents the variance of the corresponding GPM. Consequently, the reward estimate in Equation (3) is modified as:

$$\bar{r}' = \hat{r} - \lambda_U P_U, \tag{4}$$

where λ_U is the uncertainty coefficient that scales the penalty. Based on the formulation of the uncertainty penalty, the quality of world models has significant influence on the estimation of uncertainty. To release the impact, a new uncertainty penalty tailored for offline RL4RS is proposed in Sec. 3.3.2.

Entropy Penalty is one of the key contributions of DORL to alleviate the Matthew effect. It is derived from the offline data, independent of the world modeling learning. The entropy penalty is calculated as the sum of a k-order entropy, which considers the recent k interactions as a pattern and evaluates the likelihood of the next item aligning with the current pattern:

$$P_E = -D_{\text{KL}}(\pi_\beta(\cdot|\mathbf{s})||\pi_u(\cdot|\mathbf{s})), \tag{5}$$

where π_β denotes the behavior policy and π_u the uniform distribution. This penalty effectively encourages policy exploration, enhancing cumulative rewards by getting rid of suboptimal. The final reward function of DORL for policy learning is:

$$r = \hat{r} - \lambda_U P_U + \lambda_E P_E, \tag{6}$$

where λ_E is the coefficient for entropy. It controls the scale of exploration is remained in this study due to its effectiveness.

3.2 Selector

The selector searches for reference uses for a given user by leveraging the intuition that users' preferences can be **dynamically represented** by that of similar users.

3.2.1 Selector Modeling. The RL formulation is detailed here.

State Representation. During recommendation policy learning, users interact with the RecSys. Within each step of user-item interaction, there are multiple selection steps. Thus, the state for the selector needs to consider information of two parts: recommendation steps and selection steps. Specifically, features from recommendation steps consist of the user embedding in Equation (2) and recent interacted items. It is captured by the state tracker of the recommender which will be introduced in Sec. 3.3.1 and it is denoted as $\mathbf{s}^{\text{rec}}_{\mathbf{t}} \in \mathbb{R}^{d_R}$. Meanwhile, features from the selection steps derive from selected users' preferences. The user-item feedbacks predicted from the world models are used here, represented as $\mathbf{p_u} \in \mathbb{R}^d$, *i.e.,* $\mathbf{p_u}$ corresponds to one row of the predicted user-item feedback matrix. So the initial selector state for user u is given by:

$$\mathbf{s}^{\text{sel}}_{\mathbf{u,0}} = \mathbf{s}^{\text{rec}}_{\mathbf{t}} \oplus \mathbf{p_u}, \tag{7}$$

where $\oplus$ indicates concatenation and $\mathbf{s}^{\text{sel}}_{\mathbf{u,0}} \in \mathbb{R}^{d_S}$. As for the transition function of the selector, it progressively incorporates the newly added users' feedbacks with a sequence model [53]:

$$\mathbf{s}^{\text{sel}}_{\mathbf{u,t+1}} = \text{Transformer}(\mathbf{s}^{\text{sel}}_{\mathbf{u,t-w^{sel}+1}}, \cdots, \mathbf{s}^{\text{sel}}_{\mathbf{u,t}}), \tag{8}$$

where w^{sel} denotes the window size of the selector's transition function. The selection terminates when K^{sel} users' preferences are collected. It is a hyperparameter related to the size and sparsity of the dataset, which will be discussed in the experiment section.

User Selection. In each selection step, one user who is supposed to be a reference of the current user is selected:

$$u_t = \pi_{sel}(\mathbf{s}^{\text{sel}}_{\mathbf{u,t}}), \tag{9}$$

where π_{sel} refers to the selection policy.

Dynamic Selector Reward Design. In an online reinforcement learning setting, the reward design for the selector can be intuitive: it shares the same recommendation reward function from the RecSys [36]. Unfortunately, the reward design poses a substantial challenge in an offline setting since it is difficult to learn selection policy by interacting with the simulated world models. In other words, the simulated world model cannot directly *teach* the selector whether the selected users are representative enough of the current users. Therefore, an intrinsic reward function designed for the selector is urgent. Three aspects are considered in the proposed selection reward model at selection step t for current user u: as maximizing the cumulative recommendation reward is the ultimate target, single-step recommendation reward, *i.e.,* $\hat{r}$ (initially obtained from Equation (3)), is part of the intrinsic reward. Besides, based on that similar users may share similar preferences [55, 70], the similarities between the selected user' feedback vector, *i.e.,* p_u, and that of the current user are another part, termed as similarity gain:

$$r_s^{sel} = \cos(\mathbf{p_u}, \mathbf{p_{u_t}}), \tag{10}$$

where cos denotes cosine similarity and $r_s^{\text{sel}} \in \mathbb{R}$. Additionally, to ensure that the selected users are representative and informative, the selected users are expected to be diverse. Thus, the selected user's dissimilarities with the already selected users are the third part of the reward, termed as diversity gain:

$$r_d^{sel} = \frac{\sum_{u_i \in u_S} [1 - \cos(u_i, u_t)]}{||u_S||}, \tag{11}$$

where u_S is the selected user set, $||u_S||$ is the cardinal number of this set and $r_d^{\text{sel}} \in \mathbb{R}$. When $||u_S|| = 0$, $r_d^{sel} = 0$. Combining the three parts, the intrinsic reward model is:

$$r^{sel} = \hat{r} + \lambda_s r_s^{sel} + \lambda_d r_d^{sel}, \tag{12}$$

where λ_s and λ_d are two hyperparameters controlling the scale of the corresponding parts.

3.2.2 Selector Training. It is believed that the proposed selector is general so that it can be implemented by a wide range of policy gradient and actor-critic RL algorithms. For simplicity, Advantage actor-critic (A2C) [38] is chosen in DARLR. The selection happens within each recommendation step. Given a user u, the actor sequentially selects reference users for u aiming to maximize the value function of the critic until the termination of selection during the forward process. Then, the critic's value estimation is updated during loss back-propagation. It is noteworthy that employing static reward shaping, such as ROLeR [70], as an initialization step can effectively mitigate the risk of propagating uncorrected early-stage selector errors throughout subsequent dynamic reward iterations.

3.3 Recommender

The recommendation agent manages interactions between users and the RecSys. It learns a recommendation policy to enhance long-term user engagement and experience. The general formulation is described in Sec. 2. In this part, the detailed modeling of the recommender in DARLR and the connection between the selector and recommender are exhibited.

3.3.1 Recommender Modeling. For readability, the modeling introduction begins with action representation, followed by state representation and the transition function. Subsequently, the uncertainty penalty design and dynamic reward modeling are presented.

Action Representation. For each recommendation step, the current user is recommended with an item. Intuitively, the item embedding is used as the action representation [17, 20, 70]:

$$\mathbf{a_t} = \mathbf{e}_i. \tag{13}$$

State Tracker captures the dynamics of the RecSys, *i.e.,* modeling the next state representation after a user interacts with an item in the current state. Thus, it also serves as the state encoder for the recommender, which can be formulated as $\mathbf{s_{t+1}^{rec}} = P^{rec}(\mathbf{s_t^{rec}}, \mathbf{a_t})$. Following the design of DORL [17], the state tracker takes recent interacted items and received rewards into account. Further, owing to the sequential interaction nature, Transformer [53] tracker [26, 70] is preferred to seize the ordering relationship instead of the average tracker in DORL:

$$\mathbf{s_{t+1}^{rec}} = \text{Transformer}(\mathbf{s_{t-w^{rec}+1}^{rec}}, \cdots, \mathbf{s_t^{rec}}), \tag{14}$$

where w^{rec} is the window size of the recommender's transition function. Note that the state modeling of the recommender is similar to that of the selector, *i.e.,* Equation (8), except that the selector's state encoder captures information of two granularities initially.

Dynamic Recommender Reward Modeling. To decrease the impact of the world model upon recommendation policy learning, an effective offline reward shaping method is necessary. Though ROLeR [70] proposes an non-parametric clustering based method, it is a one-time reward shaping and is static during training, which does not fully utilize the offline data and discover the potential relationships between users' preferences. Therefore, the selector depicted in Sec. 3.2 is in charge of finding reference users of the current user at each recommendation step to continuously improve the reward models of the world models. Since the process description will not go into specific selection steps, t refers to the recommendation step in this part. After selection, all reference users gathers a user-dependent and step-dependent set $u_{S,t}$. Then, the dynamic reward shaping can be implemented by averaging over the set:

$$\hat{r}(u, i_t) = \frac{\sum_{u' \in u_{S,t}} \hat{r}(u', i_t)}{||u_{S,t}||}, \tag{15}$$

where i_t refers to the item selected at step t. Note that the same notation as Equation (3) is used here. It implies that the reward shaping is conducted along the whole learning process to constantly easing the impact of the inaccuracy within world models.

3.3.2 Uncertainty Design. Uncertainty penalty is a widely accepted technique used in offline RL [10, 31] to mitigate the possibilities of choosing risky actions and encouraging conservative policies. Though diverse uncertainty penalties are proposed, most of them are not tailored for RecSys. The extra sparse offline data in RecSys poses special difficulties in controlling the degree of conservatism. To address this, DARLR designs a simple yet effective uncertainty penalty that flexibly estimates the risk of the current action. This uncertainty penalty exploits the similarity gain, *i.e.,* r_s^{sel} from Equation (10), and diversity gain, *i.e.,* r_d^{sel} from Equation (11):

$$P'_U = \frac{|\hat{r} - \hat{r}_{-1}|}{r_s^{sel} + r_d^{sel}}, \tag{16}$$

where $\hat{r}_{-1}$ refers to the last reward estimation. The motivation of this design can be expressed by the representativeness of the reference user set, u_S. When reward predictions deviate significantly

Algorithm 1 Dual-Agent offline RL Recommender (DARLR).

Input:
Learned world model E (Environment); Training epoch of the recommender K; Number of trajectories in each epoch N;
Output:
Recommendation policy, π_θ^R;

Initialize the actor and critic network parameters of the recommender and the selector, θ^R, ϕ^R, θ^S and ϕ^S;
for each epoch $k = 0, 1, 2, ..., K$ **do**
  n=0
  while n < N **do**
    for each recommendation step **do**
      The actor of the selector samples a trajectory τ^S with current policy π_θ^S and stores it in T;
    end for
    The actor samples a trajectory τ^R with current policy π_θ^R by interacting with the environment E;
    Calculate advantages of the selector and the recommender with $A(\mathbf{s_t}, \mathbf{a_t}) = Q(\mathbf{s_t}, \mathbf{a_t}) - V(\mathbf{s_t})$ and the cumulative reward for each selection step in T and each recommendation step in τ^{sel}, respectively;
    Update the critics for the selector and the recommender *i.e.,* ϕ^S and ϕ^R, with loss calculated as $L_{\text{critic}} = \mathbb{E}_\tau \left[(r_t + \gamma \max_{\mathbf{a}'} Q(\mathbf{s_{t+1}}, \mathbf{a}'; \phi) - Q(\mathbf{s_t}, \mathbf{a_t}; \phi))^2 \right]$;
    Update the actor, *i.e.,* θ^R and θ^S, by ascending the policy gradient of $J_{\text{actor}} = \mathbb{E}_\tau [\log \pi_\theta(\mathbf{a_t} \mid \mathbf{s_t}) \cdot A(\mathbf{s_t}, \mathbf{a_t})]$;
    $n = n + 1$;
  end while
end for
return the policy π_θ^R;

from the previous iteration, a sharp change is acceptable if the similarity and diversity gains indicate that the selected set is sufficiently representative. However, if these two values fail to ensure representativeness, such abrupt changes introduce risks, and the selection probabilities for the corresponding actions should be constrained accordingly. The uncertainty penalty can also dynamically evolve during policy learning to ensure adaptability instead of being static in existing work [17, 20, 70].

To sum up, the ultimate reward function for training the recommendation policy is described as:

$$r^{rec} = \hat{r} - \lambda_U P'_U + \lambda_E P_E. \tag{17}$$

The entropy penalty from Equation (5) is remained to encourage recommendation diversity.

3.3.3 Recommender Training. As choosing a specific RL algorithm is not the focus of this work, and to keep consistency and fair comparisons, A2C is adopted for training the policies [17, 20, 70].

3.4 Learning Framework of DARLR

To recap and enhance the readability, the overall learning of DARLR is illustrated in Alg. 1. Note that the superscript of advantage functions, actors and critics are omitted for brevity. During the forward process, DARLR's dual-agent begins sampling. Given a user, the actor of the recommender recommends an item with the aim to maximize its critic. Then, the selector's actor progressively selects reference users for the current user guided by its critic. The corresponding reward prediction and uncertainty penalty are obtained after selection. In the backward process, temporal-difference (TD) losses are calculated for the critics of the recommender and the selector. And advantage gradient losses are calculated for both actors. After the losses being back-propagated, the selector and recommender are updated.

Table 1: Dataset statistics.

Dataset	Usage	# Users	# Items	Density
KuaiRand	Train	27285	7551	0.697%
	Test	27285	7583	0.573%
KuaiRec	Train	7176	10728	16.277%
	Test	1411	3327	99.620%
Coat	Train	290	300	8.046%
	Test	290	300	5.287%
Yahoo	Train	15400	1000	2.024%
	Test	5400	1000	1.000%

4 Experiments

In this section, experiments will be conducted to evaluate the effectiveness of DARLR with the following research questions (RQs):
- (RQ1) How does DARLR perform compared with other baselines?
- (RQ2) How does the selector, including its design and components, contribute to DARLR's performance?
- (RQ3) What are the benefits of the dynamic reward functions?
- (RQ4) Is DARLR robust to the critical hyperparameters?

4.1 Setup

4.1.1 Datasets. Experiments are conducted on four benchmark datasets following [17, 20, 70] with statistics in Table 1.
- **KuaiRand** [19] contains short video browsing histories which is very sparse and collected by randomly exposing target videos to users in the standard recommendation streams.
- **KuaiRec** [18] contains short video platform browsing histories with a fully observable user-item interaction matrix where users' feedback on items is known.
- **Coat** [48] consists of shopping ratings where products can be categorized into user-self-selected and uniformly sampled items.
- **Yahoo** [37] consists of music rating records with user-selected songs for training and records of randomly picked songs for testing.

4.1.2 Baselines. Three categories of baselines are compared in the experiments. For model-based offline RL methods, DeepFM [21] is employed to learn the default world model. Thus, for each dataset, these methods share the same world model.

Naive methods:
- ***ϵ-greedy*** has the greedy policy with maximum predicted reward and a probability of $1 - \epsilon$ for random actions.
- ***UCB*** [30] estimates confidence intervals and prioritizes actions with higher bounds to balance exploitation and exploration.

Model-free RL:
- ***SQN (2020)*** [61] employs a dual-headed network with cross-entropy loss and RL objectives tailored for RecSys.
- ***BCQ (2019)*** [16] focuses on selectively policy update with high-confidence data while excluding ambiguous samples.

Table 2: The performance summary of all methods on KuaiRand and KuaiRec. GT Reward serves as the upper bound using the ground-truth (GT) reward to train the RL agent. MCD is a reference that should be controlled instead of the lower the better [17]. (Bold: best; Underline: runner-up)

Method	KuaiRand				KuaiRec			
	R_{tra} ↑	R_{each} ↑	Length ↑	MCD	R_{tra} ↑	R_{each} ↑	Length ↑	MCD
UCB	1.6510 ± 0.1515	0.3725 ± 0.0278	4.4312 ± 0.2121	0.7886 ± 0.0235	3.6059 ± 0.6092	0.8531 ± 0.1145	4.2190 ± 0.3892	0.8112 ± 0.0582
ϵ-greedy	1.7109 ± 0.1258	0.3510 ± 0.0251	4.8804 ± 0.2700	0.7735 ± 0.0239	3.5152 ± 0.7315	0.8276 ± 0.1286	4.2186 ± 0.4049	0.8226 ± 0.0482
SQN	0.9117 ± 0.9292	0.1818 ± 0.0584	4.6007 ± 3.7125	0.6208 ± 0.1865	4.6730 ± 1.2149	0.9125 ± 0.0551	5.1109 ± 1.2881	0.6860 ± 0.0931
CRR	1.4812 ± 0.1236	0.2258 ± 0.0151	6.5613 ± 0.3519	0.7326 ± 0.0187	4.1631 ± 0.2535	0.8945 ± 0.0365	4.6541 ± 0.2150	0.8648 ± 0.0168
CQL	2.0323 ± 0.1070	0.2258 ± 0.0119	9.0000 ± 0.0000	0.7778 ± 0.0000	2.5062 ± 1.7665	0.6843 ± 0.2279	3.2239 ± 1.3647	0.3858 ± 0.3853
BCQ	0.8515 ± 0.0523	0.4246 ± 0.0164	2.0050 ± 0.0707	0.9983 ± 0.0236	2.1234 ± 0.0815	0.7078 ± 0.0272	3.0000 ± 0.0000	0.6667 ± 0.0000
MBPO	10.9325 ± 0.9457	0.4307 ± 0.0210	25.3446 ± 1.8190	0.3061 ± 0.0403	12.0426 ± 1.3115	0.7701 ± 0.0290	15.6461 ± 1.6373	0.3621 ± 0.0465
IPS	3.6287 ± 0.6763	0.2163 ± 0.0141	16.8213 ± 3.1824	**0.2010 ± 0.1156**	12.8326 ± 1.3531	0.7673 ± 0.0234	16.7270 ± 1.6834	**0.2150 ± 0.0644**
MOPO	10.9344 ± 0.9634	0.4367 ± 0.0193	25.0019 ± 1.8911	0.3433 ± 0.0289	11.4269 ± 1.7500	0.8917 ± 0.0505	12.8086 ± 1.8502	0.4793 ± 0.0619
DORL	11.8500 ± 1.0361	0.4284 ± 0.0223	27.6091 ± 2.1208	0.2960 ± 0.0356	20.4942 ± 2.6707	0.7673 ± 0.0264	26.7117 ± 3.4190	0.3792 ± 0.0149
ROLeR	13.4553 ± 1.5086	0.4574 ± 0.0332	**29.2700 ± 2.3225**	0.4049 ± 0.0356	33.2457 ± 2.6403	1.2293 ± 0.0511	27.0131 ± 1.3986	0.4439 ± 0.0212
DARLR (Ours)	**13.8152 ± 1.9351**	**0.4670 ± 0.0445**	29.2028 ± 2.3869	0.4178 ± 0.0411	**35.2203 ± 2.7576**	**1.2600 ± 0.0404**	**27.3526 ± 2.1905**	0.4869 ± 0.0438
GT (Ideal)	14.3689 ± 1.9708	0.4993 ± 0.0488	28.5582 ± 2.4114	0.4109 ± 0.0397	36.7475 ± 3.4738	1.5600 ± 0.0405	23.5653 ± 2.1824	0.5594 ± 0.0267

Table 3: The performance summary of all methods on Coat and Yahoo. GT Reward serves as the upper bound using the ground-truth (GT) reward to train the RL agent. (Bold: best; Underline: runner-up). The MCD metric is not applicable for Coat and Yahoo since both datasets do not include the attribute of *most popular categories*.

Method	Coat			Yahoo		
	R_{tra} ↑	R_{each} ↑	Length ↑	R_{tra} ↑	R_{each} ↑	Length ↑
UCB	73.6713 ± 1.8105	2.4557 ± 0.0604	30.0000 ± 0.0000	66.7578 ± 1.2539	2.2253 ± 0.0418	30.0000 ± 0.0000
ϵ-greedy	72.0042 ± 1.6054	2.4001 ± 0.0535	30.0000 ± 0.0000	64.3439 ± 1.2911	2.1448 ± 0.0430	30.0000 ± 0.0000
SQN	72.6142 ± 2.0690	2.4205 ± 0.0690	30.0000 ± 0.0000	57.7270 ± 5.7506	1.9242 ± 0.1917	30.0000 ± 0.0000
CRR	67.3830 ± 1.6274	2.2461 ± 0.0542	30.0000 ± 0.0000	57.9941 ± 1.6752	1.9331 ± 0.0558	30.0000 ± 0.0000
CQL	68.9835 ± 1.8659	2.2995 ± 0.0622	30.0000 ± 0.0000	62.2909 ± 3.3466	2.0764 ± 0.1116	30.0000 ± 0.0000
BCQ	68.8012 ± 1.7627	2.2934 ± 0.0588	30.0000 ± 0.0000	61.7388 ± 1.7808	2.0580 ± 0.0594	30.0000 ± 0.0000
MBPO	71.1930 ± 2.0943	2.3731 ± 0.0698	30.0000 ± 0.0000	64.5500 ± 2.1567	2.1517 ± 0.0719	30.0000 ± 0.0000
IPS	73.8872 ± 1.8417	2.4629 ± 0.0614	30.0000 ± 0.0000	57.8499 ± 1.7955	1.9283 ± 0.0599	30.0000 ± 0.0000
MOPO	71.1805 ± 2.0560	2.3727 ± 0.0685	30.0000 ± 0.0000	65.5098 ± 2.0996	2.1837 ± 0.0700	30.0000 ± 0.0000
DORL	71.3992 ± 2.0640	2.3800 ± 0.0688	30.0000 ± 0.0000	66.3509 ± 2.2237	2.2117 ± 0.0741	30.0000 ± 0.0000
ROLeR	76.1603 ± 2.1200	2.5387 ± 0.0707	30.0000 ± 0.0000	68.3637 ± 1.8550	2.2788 ± 0.0618	30.0000 ± 0.0000
DARLR (Ours)	**78.0429 ± 2.1462**	**2.6014 ± 0.0715**	30.0000 ± 0.0000	**68.5418 ± 1.9014**	**2.2847 ± 0.0634**	30.0000 ± 0.0000
GT (Ideal)	80.0895 ± 2.4545	2.6696 ± 0.0818	30.0000 ± 0.0000	68.8791 ± 3.2867	2.2960 ± 0.1096	30.0000 ± 0.0000

• ***CQL (2020)*** [29] employs a conservative strategy for out-of-distribution data influence when updating state-action values.

• ***CRR (2020)*** [58] refines policies by aligning updates with the deviation observed in behavior policies, emphasizing stability.

Model-based RL:

• ***IPS (2015)*** [52] applies statistical sample re-weighting to enable actor-critic policies to learn from the re-weighted offline dataset.

• ***MBPO (2019)*** [24] optimizes an actor-critic policy through iterative training on a learned world model.

• ***MOPO (2020)*** [65] introduces an uncertainty penalty via an ensemble of world models to improve policy robustness.

• ***DORL (2023)*** [17] has an uncertainty penalty with entropy to foster diverse recommendations to deal with the Matthew Effect.

• ***ROLeR (2024)*** [70] leverages non-parametric soft-label clustering to reshape the reward obtained from the world models.

Ideal situation:

• ***GT (Ideal)*** uses the ground-truth reward as an ideal baseline and upper-bound for training recommendation policies with A2C.

4.1.3 Evaluation and Metric. To effectively evaluate the performance of DARLR, the following evaluation protocol and metrics are employed to keep consistency with previous work [17, 20, 70].

• R_{tra} is the mean cumulative reward in testing episodes. It is a direct reflection of the long-term user satisfactory and user engagement.

• R_{each} represents the mean single-step reward during testing. It shows the overall single-step satisfactory of users.

• Length is the mean interaction length of the testing trajectories.

• Majority category domination (MCD) is an indication of the recommendation diversity based on the tags of items proposed by [17]. Note that (1) MCD is not an absolutely comparable metric for recommendation performance. Thus, when MCD of a method is in a reasonable range, the recommendation diversity is considered reasonable. (2) MCD is only applicable for KuaiRec and KuaiRand but not for Coat and Yahoo due to the availability of tags.

There are two termination conditions: (1) M items of the same category are recommended to a user in its recent N transitions. For a fair comparison with [17, 70], $M = 0, N = 4$ is retained in the

experiments, which means for every four transitions, the items' categories should differ; (2) the maximum interaction length: 30.

4.1.4 Implementation. The sampling steps for the recommender is 100000 and the number of reference users of the selector is chosen from $[10, 20, 30, 40]$ for KuaiRec and Coat, $[50, 100, 150, 200]$ for KuaiRand, $[25, 50, 75, 100]$ for Yahoo. Both the similarity gain, *i.e.,* r_s^{sel}, and the diversity gain, *i.e.,* r_d^{sel} range in $[0, 1]$, and the former is expected to have greater influence on the intrinsic reward. Thus, λ_s and λ_d are evaluated in $[0.5, 1, 2, 5]$ and $[0.01, 0.05, 0.1, 0.5]$, respectively across all datasets. Then, λ_U and λ_E are tuned in $[0.01, 0.05, 0.1, 0.5, 1]$ across all datasets. The state transition functions for the selector and recommender are Transformer [53] whose number of encoder layers and attention heads are chosen from $[1, 2, 3]$. The window sizes for both transition functions are chosen from $[3, 5, 10]$. Adam is used as the optimizer for the actors and critics of the selector and the recommender, as well as the state transition functions. The default learning rate is set to 0.001.

Note that while the dual-agent learning paradigm introduces an additional RL loop, the forward sampling efficiency of the selector can be enhanced by constraining the number of reference users and performing user selection within a smaller user subset, typically obtained via clustering techniques. Meanwhile, the backward computation of the dual-agent framework can be executed in parallel, ensuring that the overall computational complexity of DARLR remains comparable to existing approaches like DORL and ROLeR. Empirically, all experiments were conducted using an NVIDIA RTX™ A6000 GPU with 48 GB GDDR6 memory, with each trial completed within 10 GPU hours.

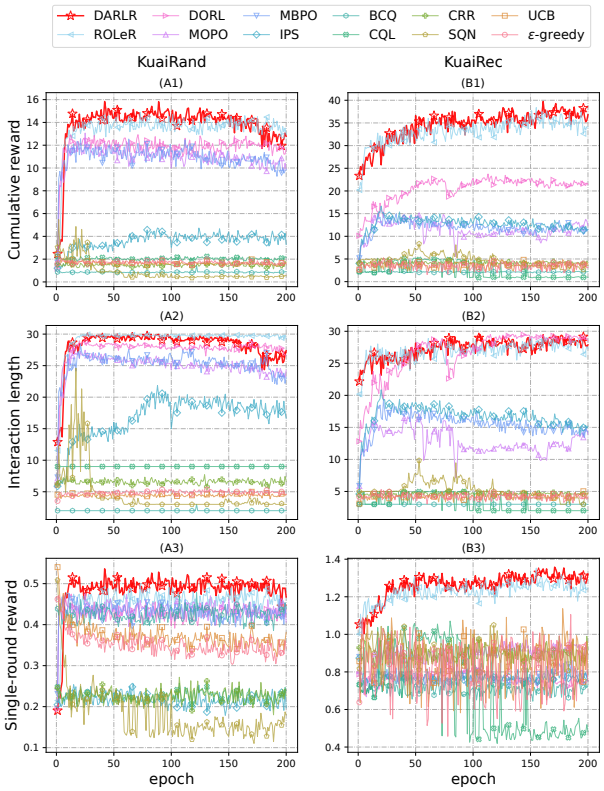

Figure 3: The overall performance on KuaiRand and KuaiRec.

4.2 Overall Performance (RQ1)

The results of the main experiments on four benchmark datasets are exhibited in Table 2 and 3. For more intuitive observation, the learning curves on KuaiRand and KuaiRec are shown in Figure 3. From these results, it can be found that the proposed DARLR achieves a better or comparable performance against current state-of-the-art methods with respect to the cumulative reward on all datasets. Meanwhile, DARLR yields the closest performance to the ideal scenarios where the recommendation policies are trained in the testing environments. Considering the termination conditions of the user-RecSys interaction, higher cumulative rewards, *i.e.,* R_{tra}, comes from a better balance between the exploitation and exploration, which is extra difficult in the offline settings. Purely pursuing a high average single-step reward, *i.e.,* R_{each}, often leads to premature termination, thereby constraining long-term user engagement. This phenomenon is evident in the performance of MBPO and MOPO on KuaiRand, as well as SQN and CRR on KuaiRec. While these methods achieve relatively high R_{each} values, their overall returns are limited by the constrained interaction length *i.e., Length*. Moreover, prioritizing the maximization of *Length* alone is insufficient to optimize user satisfaction. On Coat and Yahoo, while all methods achieve the maximum interaction length, the single-step reward becomes the dominant factor of the cumulative reward.

Follow the above analyse and break down the cumulative reward into R_{each} and *Length*. DARLR achieves the highest single-step reward across the four datasets. Specifically, the primary advantages of both ROLeR and DARLR stem from high R_{each}. Both methods emphasize the importance of reward model accuracy. While they do not show significant difference on Yahoo as their performance is very close to that of the ideal scenario, DARLR's superior performance on KuaiRec and Coat compared to ROLeR relies on higher single-step rewards. This advantage is due to DARLR's dynamic updating of reward models, enabling more accurate estimations of real-world environments. In addition, while DARLR and ROLeR achieve similar R_{each} and interaction length on KuaiRec, DARLR outperforms ROLeR significantly in cumulative reward. This result highlights that DARLR is sophisticated in effectively balancing exploration and exploitation.

Apart from the above analysis, there are some valuable findings. As introduced before, MCD is an associated metric which should be controlled in a certain range instead of being minimized. It can be evident by the MCD of IPS on KuaiRand and KuaiRec. Though IPS has the lowest MCD on the two datasets, the corresponding performance is inferior to strong baselines. By analyzing the MCD and *Length* of IPS, it is apparent that the relatively low performance on the two datasets is due to an excessive bias toward exploration. Similarly, examining the MCD and R_{each} of BCQ on KuaiRand and SQN on KuaiRec reveals that an overemphasis on exploitation results in early termination, ultimately constraining cumulative rewards. In general, model-based offline RL algorithms such as MOPO, DORL, ROLeR, and DARLR outperform model-free methods like BCQ and CQL, as well as bandit-based approaches, on KuaiRand and KuaiRec. This superiority may be attributed to the higher category density in these datasets, which challenges model-free methods in achieving

Table 4: Ablation study on DARLR's components and design. Evaluation metric: cumulative reward (R_{tra}).

Methods	KuaiRec	KuaiRand	Coat	Yahoo
ROLeR	33.2457 ± 2.6403	13.4553 ± 1.5086	76.1603 ± 2.1200	68.3637 ± 1.8550
DARLR w. r_{static}	32.4758 ± 1.9175	12.9016 ± 1.6503	74.7500 ± 2.0629	66.0434 ± 2.0149
DARLR w. $P_{U,\text{static}}$	32.8984 ± 1.7779	11.5230 ± 1.9026	73.7688 ± 2.0995	66.7788 ± 3.7383
DARLR w. $\hat{r}$	13.4651 ± 1.8317	8.6143 ± 1.6487	69.4249 ± 1.9675	56.2085 ± 2.9618
DARLR w. $\hat{r} + r_s^{\text{sel}}$	34.1514 ± 2.3736	11.7861 ± 1.4230	75.0405 ± 2.0078	67.6826 ± 2.2255
DARLR w. $\hat{r} + r_d^{\text{sel}}$	15.7642 ± 1.7882	6.7861 ± 1.4230	67.8568 ± 2.4669	58.5037 ± 2.3929
DARLR	35.2203 ± 2.7576	13.8152 ± 1.9351	78.0429 ± 2.1462	68.5418 ± 1.9014

long interaction lengths. Conversely, on Coat and Yahoo, where reaching interaction limits is less challenging for the baselines, model-free RL algorithms demonstrate competitive performance. However, their effectiveness is still hindered by the inaccuracies and utilization of the reward models.

4.3 Ablation Study (RQ2)

The effectiveness of the key designs and the proposed components are verified in this part. Recalling the method of DARLR, the selector enables dynamic reward shaping and uncertainty estimation. Thus, the static one-time reward shaping method, *i.e.,* ROLeR, is used as a baseline. Moreover, DARLR w. r_{static} and DARLR w. $P_{U,\text{static}}$ represent variants that change the reward function and uncertainty penalty to that of the ROLeR, respectively. Looking into the selector, the intrinsic reward design plays a critical role. To test the components of the intrinsic reward, DARLR w. $\hat{r}$ refers to the variant that the selector shares the same reward with the recommender. Further, DARLR w. $\hat{r}+r_s^{sel}$ and DARLR w. $\hat{r}+r_d^{sel}$ denote the variants that the intrinsic rewards are calculated as $\hat{r} + r_s^{sel}$ and $\hat{r} + r_d^{sel}$, respectively. The results are listed in Table 4.

The complete DARLR outperforms its variants and the ROLeR baseline, confirming the advantages of the selector, namely the benefits of its dynamic reward shaping and uncertainty penalties. Looking into the second and third rows in Table 4, both DARLR w. $\hat{r} + r_s^{sel}$ and DARLR w. $\hat{r} + r_d^{sel}$ cannot outperform ROLeR and DARLR, which suggests that the static reward shaping does not cope well with the dynamic uncertainty estimation and vice veasa. While DARLR w. $\hat{r}$ aligns with the selector's intended online reward model (3.2.1), its offline performance suffers due to the inability to accurately evaluate reference user selection, inducing biases when sharing the recommender's reward. Similarly, the ineffectiveness of DARLR w. $\hat{r} + r_d^{\text{sel}}$ underscores the limitations of adding diversity-driven intrinsic rewards when reference user evaluation is misaligned. Interestingly, DARLR w. $\hat{r} + r_s^{\text{sel}}$ achieves better outcomes among variants, though still suboptimal compared to ROLeR. This highlights the dual importance of leveraging user similarity for reference selection and incorporating diversity gains in intrinsic reward design, which are critical for robust performance.

4.4 Effectiveness of Dynamic Reward (RQ3)

This section analyses the impact of dynamic reward shaping during policy learning. The mean reward differences per training epoch ("Error"), defined as the deviation between ground truth and estimated rewards, are evaluated on KuaiRec and Coat datasets. DORL and ROLeR, with static reward shaping, serve as baselines.

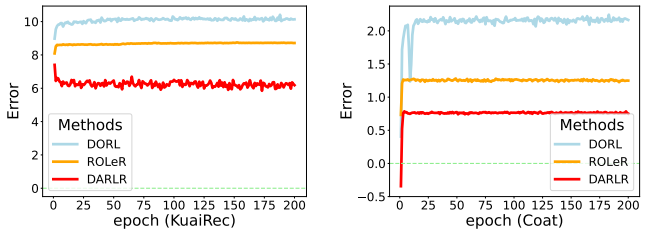

Figure 4: The mean reward differences (Error) during training of DORL, ROLeR and DARLR. The reward differences of the dynamic reward shaping method (DARLR) are consistently smaller than those of the static reward shaping method (ROLeR and DORL).

As shown in Figure 4, DARLR, leveraging dynamic reward shaping, consistently achieves lower reward differences compared to DORL and ROLeR across all training epochs in both environments. These findings align with the recommendation performance reported in Tables 2 and 3, further highlighting the efficacy of dynamically updating the reward model. In contrast, the relatively flat reward difference levels of the other methods underscore the limitations of static reward shaping, which struggles to capture the evolving nature of recommendation tasks. Notably, the reward difference curves on Coat (right plot in Figure 4) exhibit sudden increases during the initial epochs, potentially caused by sampling on accurately estimated interactions as all three curves share the same phenomenon with the same seed. Despite this, DARLR maintains competitive reward differences, ranging between $(0.5, 1)$, demonstrating its robustness across all epochs.

4.5 Hyperparameter Sensitivity (RQ4)

For DARLR, there are five key hyperparameters: (1) K^{sel}, the termination of the selection process described in Sec.3.2.1; (2) λ_s, the coefficient of the similarity gain in the intrinsic reward in Eq. (12) and (3) λ_d, the coefficient of the diversity gain; (4) λ_U, the coefficient of the uncertainty penalty in Eq. (17) and (5) λ_E, the coefficient of the entropy penalty. The testing ranges of hyperparameters have been described in Sec. 4.1.4. The corresponding results are listed in Fig.5. The cumulative reward is used as the evaluation metric and the dash line in each subplot represents the performance of DORL. Since the subplots in each column share the same legends, part of them are omitted for brevity.

Observing the first row in Fig.5, K^{sel} significantly influences performance, with dataset-specific optimal values. Smaller K^{sel} benefits denser datasets such as KuaiRec, while larger K^{sel} enhances performance on sparser datasets like KuaiRand. The second row shows the influence of the similarity gain's coefficient (λ_s). Obvious peaks can be found in the subplots, similar to the observation in the third row–the subplots for the coefficient of the diversity gain (λ_d). Combing the results of the four subplots, both λ_s and λ_d have a significant impact on the cumulative reward, and the scale of λ_s is supposed to be larger. In addition, according to the last two rows in this figure, both the coeffients of the uncertainty penalty and entropy penalty yield the highest cumulative reward at moderate values, emphasizing the importance of the balance of reward maximization, uncertainty handling and exploration robustness.

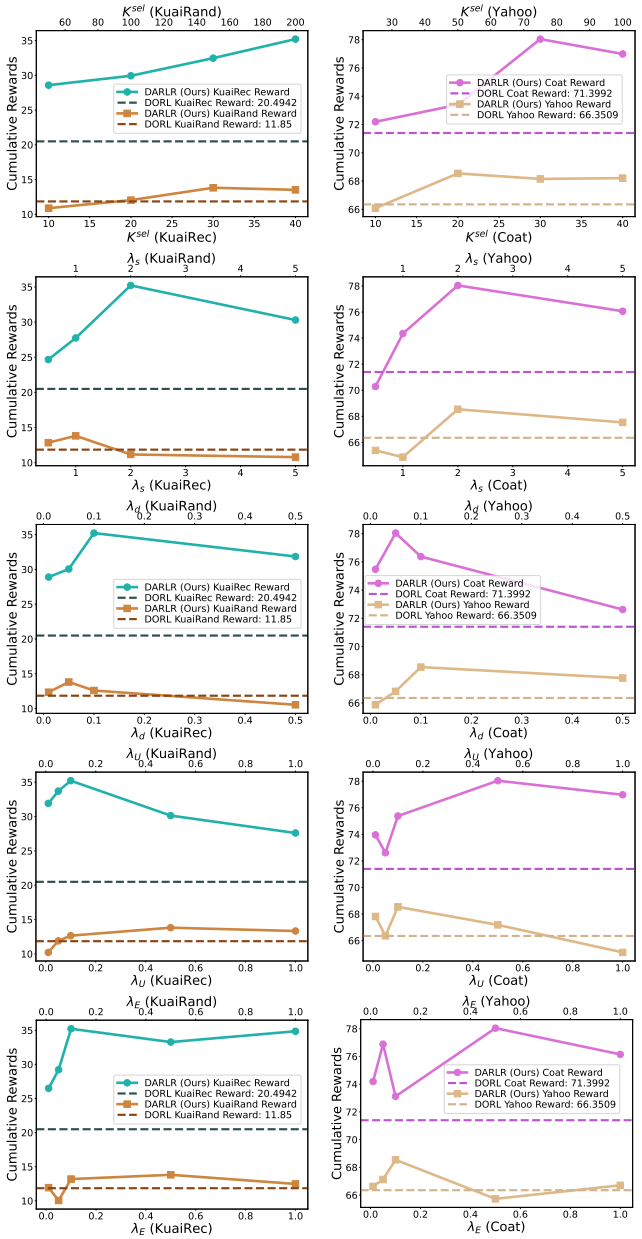

Figure 5: Hyperparameter sensitivity with different $K^{\mathbf{sel}}$, λ_s, λ_d, λ_U, and λ_E on four datasets.

5 Related Work

• **Reinforcement Learning in Recommender System.** Given the interactive nature and sparsity of RecSys [41, 42], RL methods draw increasing attention [11, 69]. With the MDP formalization, some researches explore to derive informative state representations from user features and historical interactions [22, 26]. SlateQ [23] proposes to decompose the large action space in Q-learning based methods. PrefRec [62] utilizes RLHF [50] to learn a reward model from sparse datasets. For the practicability, [8] adapts REINFORCE [60] to a large action space on the orders of millions. [9] augments the policy learning to improve sample efficiency, targeting the sparse signal issue in RecSys. Some methods explore offline RL in RecSys. CIRS [20] uses causal graphs to capture the user preferences. DORL [17] introduces entropy penalty to encourge the exploration of offline policies. ROLeR [70] proposes a static reward shaping method to ease the impact of inaccurate world models. DARLR operates within the same offline model-based RL setting, yet the significant difference lies in the evolving world model in contrast to the fixed models applied in these related works.

• **Multi-agent Reinforcement Learning in Recommender System.** A multi-agent system (MAS) contains more than one intelligent agents [5] with wide applications in robotic control [39], traffic flow management [67], and video games [34, 46]. In some scenarios where one agent is incapable of handling the user interactions [71], user recommendation customization [73], large hierarchical action spaces [7], etc., MARL exhibits its potentials [11]. MA-RDPG [14] uses two agents' communication to facilitate multi-scenario recommendations. MAHRL [72] uses hierarchical RL to decompose tasks into sub-tasks. RAM [74] designs two agents for advertising and recommendation. Nevertheless, the effectiveness of MARL remains underexplored in model-based offline RL for RecSys.

• **Offline Reinforcement Learning.** Offline RL can train a policy purely from offline data [2, 28, 31, 40, 56]. This learning paradigm is promising in domains where the offline data is significantly less expensive than online interactions such as robotics [35, 49], autonomous driving [12, 13], and recommender systems [1, 10]. Existing model-free offline RL algorithms [16, 29, 45, 58] introduces conservatism during training to deal with value overestimation issues, such as BCQ [16], CQL [29], and PRDC [45]. Offline model-based RL algorithms [24, 27, 33, 47, 65] are more sample-efficient due to the explicit modeling of transition functions and reward functions. MOPO [65] and MOReL [27] learn an ensemble of world models to estimate the state-action uncertainty. RAMBO [47] trains an adversarial environment model for RL learning. While these methods demonstrate improved performance on RL benchmarks [15], their effective adaptation to RecSys requires further investigation [10].

6 Conclusion and Future Work

In this paper, the limitations of frozen reward shaping and uncertainty estimation derived from world models in model-based offline recommender systems are identified. To address these issues, a dual-agent method, DARLR, is proposed to dynamically improve the reward function and uncertainty estimation. Specifically, a selector is employed to select reference users which considers both the similarity and diversity among users. Then, a recommender aggregates the information of reference users to improve the reward functions as well as estimating the uncertainty penalty, facilitating effective offline recommendation policy learning. Empirically,The proposed method is validated through extensive experiments on four challenging datasets, outperforming all baselines, including state-of-the-arts. In future work, the development of dynamic reward shaping methods tailored for large-scale datasets will be investigated. Further, to achieve more efficient hyperparameter tuning and reduce manual effort, the adoption of large language models (LLMs) [68] and meta-learning [54] in DARLR will be explored.

7 Acknowledegments

This work is supported by projects DE200101610, DE250100919, CE200100025 funded by Australian Research Council, and CSIRO's Science Leader Project R-91559.

References

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* (2022).

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An optimistic perspective on offline reinforcement learning. In *ICML*.

[3] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *KBS* (2013).

[4] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* (2002).

[5] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *TSMC* (2008).

[6] Jianxin Chang, Chenbin Zhang, Zhiyi Fu, Xiaoxue Zang, Lin Guan, Jing Lu, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, et al. 2023. TWIN: TWo-stage interest network for lifelong user behavior modeling in CTR prediction at kuaishou. In *SIGKDD*.

[7] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-scale interactive recommendation with tree-structured policy gradient. In *AAAI*.

[8] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *WSDM*.

[9] Minmin Chen, Bo Chang, Can Xu, and Ed H Chi. 2021. User response models to improve a reinforce recommender system. In *WSDM*.

[10] Xiaocong Chen, Siyu Wang, Julian McAuley, Dietmar Jannach, and Lina Yao. 2023. On the opportunities and challenges of offline reinforcement learning for recommender systems. *TOIS* (2023).

[11] Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang. 2023. Deep reinforcement learning in recommender systems: A survey and new perspectives. *KBS* (2023).

[12] Christopher Diehl, Timo Sebastian Sievernich, Martin Krüger, Frank Hoffmann, and Torsten Bertram. 2023. Uncertainty-aware model-based offline reinforcement learning for automated driving. *IEEE RA-L* (2023).

[13] Xing Fang, Qichao Zhang, Yinfeng Gao, and Dongbin Zhao. 2022. Offline reinforcement learning for autonomous driving with real world driving data. In *ITSC*.

[14] Jun Feng, Heng Li, Minlie Huang, Shichen Liu, Wenwu Ou, Zhirong Wang, and Xiaoyan Zhu. 2018. Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning. In *WWW*.

[15] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *CoRR* abs/2004.07219 (2020).

[16] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *ICML*.

[17] Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023. Alleviating matthew effect of offline reinforcement learning in interactive recommendation. In *SIGIR*.

[18] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. 2022. KuaiRec: A fully-observed dataset and insights for evaluating recommender systems. In *CIKM*.

[19] Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. 2022. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos. In *CIKM*.

[20] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2023. CIRS: Bursting filter bubbles by counterfactual interactive recommender system. *TOIS* (2023).

[21] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*.

[22] Jin Huang, Harrie Oosterhuis, Bunyamin Cetinkaya, Thijs Rood, and Maarten de Rijke. 2022. State encoders in reinforcement learning for recommendation: A reproducibility study. In *SIGIR*.

[23] 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*.

[24] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. 2019. When to trust your model: Model-based policy optimization. In *NeurIPS*.

[25] Wei Jiang, Xinyi Gao, Guandong Xu, Tong Chen, and Hongzhi Yin. 2024. Challenging Low Homophily in Social Recommendation. In *WWW*.

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

[27] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. 2020. Morel: Model-based offline reinforcement learning. In *NeurIPS*.

[28] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NIPS*.

[29] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. In *NeurIPS*.

[30] Tze Leung Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* (1985).

[31] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *CoRR* abs/2005.01643 (2020).

[32] Yang Li, Tong Chen, Yadan Luo, Hongzhi Yin, and Zi Huang. 2021. Discovering Collaborative Signals for Next POI Recommendation with Iterative Seq2Graph Augmentation. In *IJCAI*.

[33] Xiao-Yin Liu, Xiao-Hu Zhou, Guo-Tao Li, Hao Li, Mei-Jiang Gui, Tian-Yu Xiang, De-Xing Huang, and Zeng-Guang Hou. 2024. MICRO: Model-Based Offline Reinforcement Learning with a Conservative Bellman Operator. In *IJCAI*.

[34] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*.

[35] Jianlan Luo, Perry Dong, Jeffrey Wu, Aviral Kumar, Xinyang Geng, and Sergey Levine. 2023. Action-quantized offline reinforcement learning for robotic skill learning. In *CoRL*. 1348–1361.

[36] Yadan Luo, Zi Huang, Zheng Zhang, Ziwei Wang, Jingjing Li, and Yang Yang. 2019. Curiosity-driven reinforcement learning for diverse visual paragraph generation. In *MM*.

[37] Benjamin M Marlin and Richard S Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *RecSys*.

[38] 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 *ICML*.

[39] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. Facmac: Factored multi-agent centralised policy gradients. In *NeurIPS*.

[40] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. 2023. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *TNNLS* (2023).

[41] Ruihong Qiu, Zi Huang, and Hongzhi Yin. 2021. Memory augmented multi-instance contrastive predictive coding for sequential recommendation. In *ICDM*.

[42] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive learning for representation degeneration problem in sequential recommendation. In *WSDM*.

[43] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the item order in session-based recommendation with graph neural networks. In *CIKM*.

[44] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. Gag: Global attributed graph neural network for streaming session-based recommendation. In *SIGIR*.

[45] Yuhang Ran, Yi-Chen Li, Fuxiang Zhang, Zongzhang Zhang, and Yang Yu. 2023. Policy regularization with dataset constraint for offline reinforcement learning. In *ICML*.

[46] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *JMLR* (2020).

[47] Marc Rigter, Bruno Lacerda, and Nick Hawes. 2022. RAMBO-RL: Robust Adversarial Model-Based Offline Reinforcement Learning. In *NeurIPS*.

[48] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *ICML*.

[49] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. 2022. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *CoRL*. 907–917.

[50] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *NeurIPS* (2020).

[51] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[52] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual risk minimization: Learning from logged bandit feedback. In *ICML*.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

[54] Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Joaquin Vanschoren, Thorsteinn Rögnvaldsson, and KC Santosh. 2024. Advances and challenges in meta-learning: A technical review. *TPAMI* (2024).

[55] Hengliang Wang and Kedian Mu. 2020. Aspect-Level Attributed Network Embedding via Variational Graph Neural Networks. In *DASFAA*.

[56] Ruosong Wang, Dean P. Foster, and Sham M. Kakade. 2021. What are the Statistical Limits of Offline RL with Linear Function Approximation?. In *ICLR*.

[57] Xu Wang, Jiangxia Cao, Zhiyi Fu, Kun Gai, and Guorui Zhou. 2025. HoME: Hierarchy of Multi-Gate Experts for Multi-Task Learning at Kuaishou. In *SIGKDD*.

[58] 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. In *NeurIPS*.

[59] Shaowei Wei, Zhengwei Wu, Xin Li, Qintong Wu, Zhiqiang Zhang, Jun Zhou, Lihong Gu, and Jinjie Gu. 2024. Leave No One Behind: Online Self-Supervised

Self-Distillation for Sequential Recommendation. In *WWW*.
[60] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* (1992).
[61] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-supervised reinforcement learning for recommender systems. In *SIGIR*.
[62] Wanqi Xue, Qingpeng Cai, Zhenghai Xue, Shuo Sun, Shuchang Liu, Dong Zheng, Peng Jiang, Kun Gai, and Bo An. 2023. PrefRec: Recommender Systems with Human Preferences for Reinforcing Long-term User Engagement. In *SIGKDD*.
[63] Wanqi Xue, Qingpeng Cai, Ruohan Zhan, Dong Zheng, Peng Jiang, Kun Gai, and Bo An. 2023. ResAct: Reinforcing Long-term Engagement in Sequential Recommendation with Residual Actor. In *ICLR*.
[64] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. 2021. Combo: Conservative offline model-based policy optimization. In *NeurIPS*.
[65] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. Mopo: Model-based offline policy optimization. In *NeurIPS*.
[66] 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 *SIGIR*.
[67] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *WWW*.
[68] Michael R Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. 2023. Using large language models for hyperparameter optimization. *arXiv preprint arXiv:2312.04528* (2023).
[69] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys* (2019).
[70] Yi Zhang, Ruihong Qiu, Jiajun Liu, and Sen Wang. 2024. ROLeR: Effective Reward Shaping in Offline Reinforcement Learning for Recommender Systems. In *CIKM*.
[71] Yang Zhang, Chenwei Zhang, and Xiaozhong Liu. 2017. Dynamic scholarly collaborator recommendation via competitive multi-agent reinforcement learning. In *RecSys*.
[72] Dongyang Zhao, Liang Zhang, Bo Zhang, Lizhou Zheng, Yongjun Bao, and Weipeng Yan. 2020. Mahrl: Multi-goals abstraction based deep hierarchical reinforcement learning for recommendations. In *SIGIR*.
[73] Xiangyu Zhao, Long Xia, Lixin Zou, Hui Liu, Dawei Yin, and Jiliang Tang. 2020. Whole-chain recommendations. In *CIKM*.
[74] Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. 2020. Jointly learning to recommend and advertise. In *SIGKDD*.
[75] Peilin Zhou, You-Liang Huang, Yueqi Xie, Jingqi Gao, Shoujin Wang, Jae Boum Kim, and Sunghun Kim. 2024. Is Contrastive Learning Necessary? A Study of Data Augmentation vs Contrastive Learning in Sequential Recommendation. In *WWW*.