Reinforcing Long-Term Performance in Recommender Systems with User-Oriented Exploration Policy

Changshuo Zhang*
Gaoling School of AI,
Renmin University of China
Beijing, China
lyingcs@ruc.edu.cn

Sirui Chen*
University of Illinois at
Urbana-Champaign
Illinois, USA
chensr16@gmail.com

Xiao Zhang†
Gaoling School of AI,
Renmin University of China
Beijing, China
zhangx89@ruc.edu.cn

Sunhao Dai
Gaoling School of AI,
Renmin University of China
Beijing, China
sunhaodai@ruc.edu.cn

Weijie Yu
School of Information Technology
and Management, University of
International Business and Economics
Beijing, China
yu@uibe.edu.cn

Jun Xu
Gaoling School of AI,
Renmin University of China
Beijing, China
junxu@ruc.edu.cn

ABSTRACT

Reinforcement learning (RL) has gained popularity in recommender systems for improving long-term performance by effectively exploring users' interests. However, modern recommender systems face the challenge of different user behavioral patterns among millions of items, making exploration more difficult. For example, users with varying activity levels require different exploration intensities. Unfortunately, previous studies often overlook this aspect and apply a uniform exploration strategy to all users, which ultimately hampers long-term user experiences. To tackle these challenges, we propose User-Oriented Exploration Policy (UOEP), a novel approach that enables fine-grained exploration among user groups. We first construct a distributional critic that allows policy optimization based on varying quantile levels of cumulative reward feedback from users, representing user groups with different activity levels. Using this critic as a guide, we design a population of distinct actors dedicated to effective and fine-grained exploration within their respective user groups. To simultaneously enhance diversity and stability during the exploration process, we also introduce a population-level diversity regularization term and a supervision module. Experimental results on public recommendation datasets validate the effectiveness of our approach, as it outperforms all other baselines in terms of long-term performance. Moreover, further analyses reveal the benefits of our approach, including improved performance for low-activity users and increased fairness among users.

*Both authors contributed equally to this research.
†Xiao Zhang is the corresponding author. Work partially done at Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education.

CCS CONCEPTS

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

KEYWORDS

Recommender Systems, Reinforcement Learning

ACM Reference Format:
Changshuo Zhang, Sirui Chen, Xiao Zhang, Sunhao Dai, Weijie Yu, and Jun Xu. 2024. Reinforcing Long-Term Performance in Recommender Systems with User-Oriented Exploration Policy. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657714

1 INTRODUCTION

Recommender Systems (RS) are integral components embedded within an array of web services, encapsulating e-commerce [18, 36], social media [51, 52], streaming music [10, 57], and news feeding [49, 50], among others. In recent years, there has been a notable surge in the interest in Reinforcement Learning (RL) given its distinctive ability to explore user interests and elevate long-term performance within RS. In contrast to traditional supervised methods [32, 53] that optimize immediate user feedback, RL-based RS [5, 7, 40, 58, 59] treats the streaming recommendation problem as an interactive process, focusing on maximizing the cumulative rewards of users in the long term. RL-based methods provide a deeper understanding of user preferences, enabling recommender systems to generate personalized recommendations that align with long-term user satisfaction.

Unlike other RL-based tasks, applying RL to RS in practical scenarios suffers from large action space and sparse user feedback [13, 31], which contribute to intricate user behavioral patterns. For instance, most users and items exhibit only a limited number of interactions, whereas a minority of highly active users or popular items contribute to a small fraction of the total. This difference in user behavior results in a long-tailed return distribution, which in turn adds difficulties to the task of learning user preferences. Therefore, an effective RL-based recommender is expected to learn

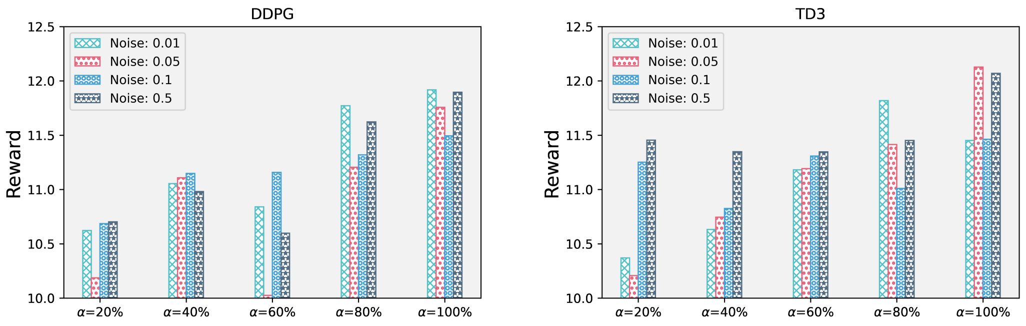

Figure 1: Illustration Experiment. We resorted users based on their activity levels (total number of clicks) and selected the five bottom α-quantile ($\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$) of user groups. We then trained two RL algorithms, DDPG and TD3, on these five groups under four varying noise levels. We conducted all experiments with four different seeds and reported average results.

diverse policies that exploit more for the active users while exploring more for the inactive ones.

To better illustrate the differences among users, we conduct experiments with the quantile-based grouping of users on a public short video dataset named KuaiRand. Specifically, we arrange users by their total number of clicks. Subsequently, we select users from various bottom quantiles ($0 \sim \alpha$) of this sorted outcome, which represents users with the bottom α activity levels. We then evaluate the performance of two typical RL algorithms under different levels of noise on these user groups, where the larger noise represents more intense exploration. Figure 1 demonstrates that as the quantile decreases, indicating users with lower activity levels, there is a tendency for them to exhibit a preference for higher levels of noise. The observation suggests that the optimal exploration intensity varies significantly across different user groups, emphasizing the importance of implementing more effective and fine-grained exploration in RL-based recommender systems.

Despite its critical significance, the effective exploration strategies at the user level remain largely uninvestigated. Traditional reinforcement learning exploration strategies, such as noise perturbation and ϵ-greedy, introduce randomness mainly during the selection of actions [56]. Recent studies have made strides in this area by employing decomposition techniques that make RL more manageable with recommendation slates, or by using alignment methods to balance the trade-off between exploration and exploitation in the latent action representation space [13, 19, 31]. However, these methods have primarily concentrated on the regularization of action space, overlooking return distribution variations among users and thus not effectively enabling user-oriented exploration.

Based on the above observations, in this paper, we propose a novel approach called UOEP for reinforcing user-oriented exploration in recommender systems. UOEP employs the return distribution under different quantiles to explicitly characterize the activity level of users and group them according to the quantile. In this way, multiple actors can be learned and each actor corresponds to a specific user group with a predefined level of activity. Specifically, we first introduce a distributional critic to learn the return distribution instead of an expected return. Given the return distribution, multiple actors are learned where each actor subsequently optimizes towards different target quantiles within the return distribution. This approach enables us to customize the exploration intensity for different groups. Then, a population diversity regularization term is introduced to encourage diversity among the different actors, promoting effective exploration in the action space. Considering that introducing multiple actors at once may cause the instability of the training, we also incorporate a supervision module to ensure the stability during the actor learning process. We evaluate the performance of our approach and compare it with state-of-the-art recommendation approaches through comprehensive experiments on various public and industrial recommendation datasets. The experimental results demonstrate its advantages in terms of long-term performance. Further analyses demonstrate our model's ability to improve the experience for low-active users and enhance overall fairness, highlighting the additional advantages of our approach.

2 PROBLEM FORMULATION

We focus on the task of session-based recommendation in this paper. Formally, given an item candidate pool $\mathcal{I}$, the recommender system aims to select a list of items $x_t^u = \{i_1, \ldots, i_n\}$ for the current user u at each time step t. Here, $i_k \in \mathcal{I}$ for $1 \leq k \leq n$, and n represents the list size, which denotes the number of items provided to the user during each interaction within the session. The goal of the learning is to encourage users to engage in more interactions within a session (*depth*) and maximize their cumulative rewards, such as clicks or purchases over the sessions (*return*). We formulate this problem as the Markov Decision Process (MDP), which consists of 5-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$:

- $\mathcal{S}$ is the *continuous state space*, where $s \in \mathcal{S}$ indicates the state of a user including static features such as gender and dynamic features such as historical interactions.
- $\mathcal{A}$ is the *action space*, where $a \in \mathcal{A}$ is possible recommendation lists in a single interaction within a session, i.e., $\mathcal{A} = \mathcal{I}^n$.
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the *state transition function*, where $p(s'|s, a)$ specifies the probability of transitioning from the current state s to a new state s' after taking action a.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the *reward function* that maps a user state s and an action a to an immediate reward $r(s, a)$, which is related to the user feedback, such as clicks or likes.
- γ is the *discount factor* for weighting future rewards relative to immediate rewards.

The policy function $\pi(a|s) : \mathcal{S} \to \mathcal{A}$ represents the probability of selecting action a given state s. We define $\mathcal{Z}^{\pi}(s, a)$ to represent

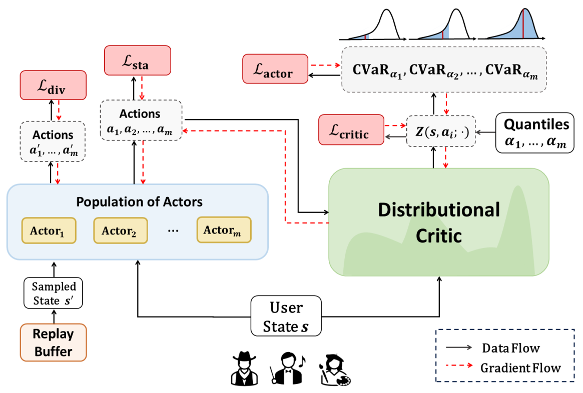
$\mathcal{L}_{\text{div}}$
$\mathcal{L}_{\text{sta}}$
$\mathcal{L}_{\text{actor}}$
$\text{CVaR}_{\alpha_1}, \text{CVaR}_{\alpha_2}, \dots, \text{CVaR}_{\alpha_m}$
Actions $a'_1, \dots, a'_m$
Actions $a_1, a_2, \dots, a_m$
$\mathcal{L}_{\text{critic}}$
$Z(s, a_i; \cdot)$
Quantiles $\alpha_1, \dots, \alpha_m$
Population of Actors
Actor$_1$
Actor$_2$
Actor$_m$
Distributional Critic
Sampled State s'
Replay Buffer
User State s
Data Flow
Gradient Flow

Figure 2: The proposed approach UOEP. It includes a population of m actors, where m is the population size and each actor$_i$ outputs an action a_i based on the current user state s. The action a_i along with state s is fed into the distributional critic. Utilizing both its quantile value α_i and the critic's output $Z(s, a_i; \cdot)$, actor$_i$ calculates the conditional value at risk (CVaR) measure to derive its policy gradients.

the return distribution under π given state s and action a, equaling in distribution $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$. The objective of standard RL is to learn an optimal policy $\pi^*(a|s)$ that maximizes the expectation of $\mathcal{Z}^\pi$, denoted as $\max_{\pi^*} \mathcal{J}_{\mathbb{E}}(\pi^*) := \mathbb{E}[\mathcal{Z}^\pi]$. However, as previously stated, due to the complex distribution of user feedbacks in RS, solely maximizing expectations results in ineffective and coarse-grained exploration. In this paper, we turn to facilitate the exploration by improving the distortion of the return distribution, as opposed to focusing solely on the expectation. This can be represented as $\max_{\pi^*} \mathcal{J}_{\mathbb{D}}(\pi^*) := \mathbb{D}[\mathcal{Z}^\pi]$. where $\mathbb{D}$ is a distortion operator that maps the reward distribution to real numbers. Since our goal is to improve the effectiveness of exploration under different user groups, a better choice of the distortion operator $\mathbb{D}$ is CVaR$_\alpha$, which captures the expected return when experiencing a given bottom α-quantile of the possible outcomes [43]. Mathematically, it can be represented as follows:

$$\text{CVaR}_\alpha(\mathcal{Z}^\pi) = \frac{1}{\alpha} \int_0^\alpha F^{-1}(\mathcal{Z}^\pi; \tau) \mathrm{d}\tau, \tag{1}$$

where $\alpha \in [0, 1]$ and the *quantile function*, denoted as $F^{-1}(\mathcal{Z}^\pi; \cdot)$, is the inverse function of the cumulative distribution function (CDF) of the return distribution $\mathcal{Z}^\pi$. It maps a quantile value $\tau \in [0, 1]$ to the corresponding value within the distribution. When $\alpha = 1$, the CVaR value is equal to the expectation of the entire distribution.

3 UOEP: THE PROPOSED APPROACH

In this section, we introduce our approach: User-Oriented Exploration Policy (UOEP). As depicted in Figure 2, UOEP primarily comprises a population of actors and a distributional critic. Each actor within this population operates independently to offer distinct recommendations to users. Additionally, each actor is assigned a unique α value, which is utilized for optimization targeting distinct bottom α-quantiles of the return distribution, denoted as CVaR$_\alpha$. This approach ensures that each actor learns within user groups characterized by different activity levels. To enable effective exploration among actors in the population and ensure stability during the learning process, we introduce two regularization terms. We first incorporate a population diversity regularization loss $\mathcal{L}_{\text{div}}$ to facilitate effective exploration through diversified actors. Then, we introduce a supervision module $\mathcal{L}_{\text{sta}}$ to stabilize the learning process. We delve into the learning process for the distributional critic in Section 3.1. Next, in Section 3.2, we define the actor loss, which relies on a more effective distortion operator, CVaR, applied to the acquired return distribution and optimize it using a gradient-based approach. Then, we introduce the two regularization terms, $\mathcal{L}_{\text{div}}$ and $\mathcal{L}_{\text{sta}}$ in Section 3.3. Finally, We summarize the training and inference procedure in Section 3.4.

3.1 Learning Distributional Critic

To learn a distributional critic, we first introduce the *distributional Bellman equation*:

$$\mathcal{Z}^\pi(s, a) \overset{\mathcal{D}}{=} r(s, a) + \gamma \mathcal{Z}^\pi(s', a'), \tag{2}$$

where random variables s', a' are drawn according to $s' \sim p(\cdot|s, a)$ and $a' \sim \pi(\cdot|s')$, and $A \overset{\mathcal{D}}{=} B$ denotes the equality in probability distribution between the random variables A and B [3]. In practice, we choose to represent the return distribution by learning its implicit quantile function using an Implicit Quantile Network (IQN) [8, 45]. This approach offers computational advantages for efficiently calculating CVaR. Specifically, we learn the distributional critic $Z_\theta(s, a; \tau)$ parameterized by θ, to approximate the *quantile function* $F^{-1}(\mathcal{Z}^\pi(s, a); \cdot)$ at a given quantile $\tau \in [0, 1]$. Formally, for each sampled transition (s, a, r, s'), the temporal difference (TD) error can be computed as follows:

$$\delta_{\tau, \tau'} = r + \gamma Z'_{\theta'}(s', a'; \tau') - Z_\theta(s, a; \tau), \tag{3}$$

where $a' \sim \pi(\cdot|s')$, τ, τ' are independently sampled from the uniform distribution, i.e., $\tau, \tau' \sim \mathbb{U}(0, 1)$ and Z' is a target network whose parameters are soft-updated to match the corresponding models [14]. To accurately learn the relationships between quantiles, we utilize the τ-quantile Huber Loss as proposed by [9]. The loss function is defined as:

$$\mathcal{L}_\kappa(\delta; \tau) = \underbrace{\left|\tau - \mathbb{1}_{\{\delta<0\}}\right|}_{\text{quantile loss}} \cdot \underbrace{\begin{cases} \delta^2/2\kappa & \text{if } |\delta| \le \kappa, \\ |\delta| - \kappa/2 & \text{otherwise}, \end{cases}}_{\text{Huber loss}} \tag{4}$$

where $\kappa > 0$ is a hyperparameter that controls the growth rate of the loss, and it is typically set to 1. We approximate the critic loss for all levels τ by sampling N independent quantiles τ and N' independent target quantiles τ', resulting in the following expression:

$$\mathcal{L}_{\text{critic}}(\theta) = \mathbb{E}_{(s,a,r,s',a')\sim\pi} \left[\frac{1}{N \cdot N'} \sum_{i=1}^{N} \sum_{j=1}^{N'} \mathcal{L}_\kappa \left(\delta_{\tau_i, \tau'_j}; \tau_i \right) \right]. \tag{5}$$

3.2 Optimizing Population of Policies Towards CVaR

Now, we can start optimizing the population of actors. Building upon our previous observation, we aim to have each actor learn towards different τ-quantile return distributions, which can be achieved by computing CVaR_α. Given a reliable distributional critic, the CVaR can be efficiently approximated using a sampling-based scheme from its quantile representation. Similar as Eq. (1), for each state-action pair (s, a), we can use the critic's output $Z_\theta(s, a; \tau)$ to compute CVaR_α by Monte Carlo sampling:

$$\begin{aligned}\mathrm{CVaR}_\alpha\left(\mathcal{Z}^\pi(s,a)\right) &= \frac{1}{\alpha}\int_0^\alpha Z_\theta(s,a;\tau)\mathrm{d}\tau \\ &\approx \frac{1}{K}\sum_{k=1}^{K} Z_\theta\left(s,a;\tau_k\right),\ \tau_k \sim \mathbb{U}(0,\alpha),\end{aligned} \tag{6}$$

where K is the sampling times. Hence, considering an actor $\pi_{\phi_i}(a|s)$ parameterized by ϕ_i along with its assigned quantile α_i, the actor loss can be written as:

$$\mathcal{L}_{\text{actor}}(\phi_i) = -\mathbb{E}_{(s,a)\sim\pi_{\phi_i}}\left[\mathrm{CVaR}_{\alpha_i}\left(\mathcal{Z}^\pi(s,a)\right)\right],\ i\in\{1,\ldots,m\}, \tag{7}$$

where m is the population size. Under the guidance of Eq. 7, each actor within the population learns for a specific quantile value, ensuring fine-grained exploration of user groups with varying activity levels. Furthermore, during the training process, we observed that directly setting a low value for α makes it difficult for the trained actors to perform well. This phenomenon is known as the *blindness to success* problem [15], which refers to the tendency to overlook successful cases and get stuck in local optima. To mitigate this issue, we have incorporated a soft mechanism into our actors. Instead of using a fixed level, the actor will optimize the α' value gradually decreasing from 1 to α during training. With this setting, we can compute α_t during training using:

$$\alpha_t = \max\left\{\alpha, 1-\beta\cdot(1-\alpha)\cdot t\right\}, \tag{8}$$

where β is the quantile decay ratio and t is the current time. Intuitively, it becomes clear that all actors undergo initial optimization across the complete return distribution, after which they progressively fine-tune their optimization within their respective designated quantiles. Furthermore, it's worth noting that all actors share a common replay buffer. This buffer stores transitions (s, a, r, s') from interactions between actors and the environment, which are used for training the distributional critic.

3.3 Facilitating Diversity and Stability for Population

A population of actors learning across different quantiles of the return distribution simplifies the exploration process. This is because each actor focuses on a smaller subset of users, rendering the exploration task more effective. However, this approach brings forth new challenges. One challenge emerges when each actor within the population learns individually. This could lead to various members of the population sharing overlapping portions of the explored action space or continually switching between different behavioral patterns [20, 35]. Such scenarios could potentially undermine the efficiency of the exploration process. Another challenge is the potential instability introduced by multiple actors learning simultaneously. Therefore, it becomes crucial to ensure that the actor population maintains both **Diversity** and **Stability**.

Diversity. Following [35], we measure the diversity of the population of actors using individual *behavior embeddings* to compute the volume of the kernel matrix between actors. By utilizing it as a regularization term, we prevent actors from becoming too similar, thereby promoting diversity and enabling a broader exploration of potential actions.

The behavior embedding, denoted as $\Phi(\pi) = \{\pi(\cdot|s)\}_{s\in\mathcal{S}}$, serves as a vectorized representation of an actor's behavior. This representation is embedded within the behavior space, facilitating the measurement of similarity between different actors. However, in the context of recommender systems, the state space is exceptionally vast and complex. To manage this challenge, a subset of states is sampled, with the number of samples significantly smaller than the size of the overall state space. Utilizing the subsequent formula, we approximate the behavior embedding through an expectation:

$$\widehat{\Phi}(\pi) = \mathbb{E}_{s\sim\mathcal{S}}\left[\{\pi(\cdot|s)\}\right]. \tag{9}$$

In practice, we will sample a batch of states from the shared replay buffer of the population. We then execute all actors on these sampled states to approximate their respective behavioral embeddings. Once these approximations are obtained, we use a specific metric to quantify the diversity within the actor population. To compare two different actors, we employ a kernel function $\mathcal{K}$ to map their behavioral embeddings into a higher-dimensional feature space. In this space, we choose the squared exponential (SE) kernel, which is defined as follows:

$$\mathcal{K}_{\text{SE}}\left(x_1,x_2\right) = \exp\left(-\frac{\|x_1-x_2\|^2}{2l^2}\right), \tag{10}$$

where l is a hyperparameter satisfying $l > 0$. Then we compute a kernel matrix $\mathbf{K}$ among the actors within the population, defined as $\mathbf{K} = \mathcal{K}_{\text{SE}}\left(\widehat{\Phi}\left(\pi_{\phi_i}\right), \widehat{\Phi}(\pi_{\phi_j})\right)_{i,j=1}^{m}$. Geometrically, the determinant of this matrix, $\det(\mathbf{K})$, corresponds to the volume of a parallelepiped defined by the feature maps of the chosen kernel function. A larger volume indicates a higher perceived degree of diversity within the actor population. We optimize population diversity as a regularization term in the population loss function. Specifically, this loss is defined as:

$$\begin{aligned}\mathcal{L}_{\text{div}}(\phi_1,\ldots,\phi_m) &= -\log\det(\mathbf{K}) \\ &= -\log\det\left(\mathcal{K}_{\text{SE}}\left(\widehat{\Phi}\left(\pi_{\phi_i}\right), \widehat{\Phi}(\pi_{\phi_j})\right)_{i,j=1}^{m}\right).\end{aligned} \tag{11}$$

Stability. To ensure stability in the learning process, we introduce a supervision module $\mathcal{L}_{\text{sta}}$. Its objective is to align the outputs of the actors with the user response. With the help of the supervision module, each actor can more effectively utilize the detailed user feedback on every item. One simple approach to achieve this is to combine the cross-entropy loss of each actor, defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{sta}}(\phi_i) = \mathbb{E}_{(s,a)\sim\pi_{\phi_i}}\Big[& y\log\pi_{\phi_i}(a|s) \\ &+ (1-y)\log\left(1-\pi_{\phi_i}(a|s)\right)\Big],\ i\in\{1,\ldots,m\},\end{aligned} \tag{12}$$

where y is the supervision signal such as user clicks.

Table 1: Statistics of datasets after preprocessing.

Dataset	Users	Items	# of record	List size n
KuaiRand	986	11,643	96,532	10
MovieLens-1M	6041	3953	97,382	10
RL4RS	-	283	781,367	9

Balancing Stability and Diversity. With the diversity loss and stability loss calculated, the total population loss can be written as:

$$\mathcal{L}_{\text{total}}(\phi_1, \ldots, \phi_m) = \sum_{i=1}^{m} \big(\mathcal{L}_{\text{actor}}(\phi_i) + \lambda_1 \mathcal{L}_{\text{sta}}(\phi_i)\big) + \lambda_2 \mathcal{L}_{\text{div}}(\phi_1, \ldots, \phi_m), \tag{13}$$

where the coefficients λ_1 and λ_2 are coefficients dynamically tuned through a two-armed bandit framework. In this context, we consider the two losses, $\mathcal{L}_{\text{div}}$ and $\mathcal{L}_{\text{sta}}$, as the two arms of the bandit. During training iterations, we alternate between optimizing the model based on the diversity loss and the stability loss. These loss functions have distinct training objectives. If we notice declining performance, indicating reduced model effectiveness, we prioritize the stability loss by setting λ_2 to 0, aiming to minimize the impact of excessive exploration and improve stability. Conversely, if we observe performance improvements, suggesting enhanced model performance, we prioritize the diversity loss by setting λ_1 to 0, encouraging the model to explore more extensively and search for potential optimization directions.

3.4 Algorithm Implementation

We train an online environment simulator and choose the DDPG algorithm as the backbone of our UOEP algorithm.

Training Procedure. The training procedure of UOEP is outlined in Algorithm 1. Particularly, we assign a target network to both the critic and each actor. In each epoch, the critic loss and total population loss are calculated, and gradient descent and soft update operations are performed.

Inference Procedure. Inference procedure of UOEP is show in Algorithm 2. Only the trained distributional critic and actor population are used during execution. Each time the recommender system receives the user's status, we use a *critic-trusted* method to select the optimal action. Specifically, all actors provide action a to the critic, which then selects the one with the largest return expectation to execute. The calculation of return expectation and the selection of the optimal action are shown in Eq. (14) and Eq. (15):

$$Q(s, a) = \mathbb{E}\left[\mathcal{Z}(s, a)\right] \approx \frac{1}{K} \sum_{k=1}^{K} Z\left(s, a; \tau_k\right), \tau_k \sim \mathbb{U}(0, 1), \tag{14}$$

$$a^* = \arg\max_{a_i} Q\left(s, a_i\right), i \in \{i, \ldots, m\}. \tag{15}$$

4 EXPERIMENTS

In this section, we conduct experiments on public and industrial datasets to verify the effectiveness of the UOEP. We mainly focus on the following questions: *Q1. Can UOEP consistently outperform previous state-of-the-art methods (Section 4.2)? Q2. How does UOEP work and how do each of its components contribute (Section 4.3)? Q3. What potential does UOEP have (Section 4.5)?* The source code of experiments is shared at https://github.com/lyingCS/UOEP.

Algorithm 1 Training procedure of UOEP

```
Input: Shared replay buffer $\mathcal{B}$; Critic $Z_\theta$ and critic-target $Z_{\theta'}$;
    Population $\mathcal{P}$ includes $m$ actors $\pi_{\phi_1}, \ldots, \pi_{\phi_m}$, actor-targets
    $\pi_{\phi'_1}, \ldots, \pi_{\phi'_m}$ and corresponding quantile levels $\alpha_1, \ldots, \alpha_m$; On-
    line simulator $\mathcal{S}$.
Output: Optimal population $\mathcal{P}^*$ includes $m$ actors learned from
    their each quantile level $\alpha_i$.
 1: for $t = 1, \ldots$ do
 2:     for $i = 1, \ldots, m$ do
 3:         Execute action according to $\pi_{\phi_i}$ and current state.
 4:         Get reward and new state from $\mathcal{S}$ then store the transition
            in $\mathcal{B}$.
 5:     end for
 6:     Sample a random minibatch of $B$ transitions from $\mathcal{B}$.
 7:     Compute critic loss $\mathcal{L}_{\text{critic}}(\theta)$ by Eq. (5).
 8:     Gradient step $\theta$ with $\mathcal{L}_{\text{critic}}(\theta)$.
 9:     Compute total population loss $\mathcal{L}_{\text{total}}(\phi_1, \ldots, \phi_m)$ by Eq. (13).
10:     Gradient step $\phi_1, \ldots, \phi_m$ with $\mathcal{L}_{\text{total}}(\phi_1, \ldots, \phi_m)$.
11:     Perform soft-update on $\theta'$ and $\pi'_{\phi_1}, \ldots, \pi'_{\phi_m}$.
12: end for
13: return $\mathcal{P}^*$.
```

Algorithm 2 Inference procedure of UOEP

```
Input: Critic $Z_\theta$; Population $\mathcal{P}$ includes actors $\pi_{\phi_1}, \ldots, \pi_{\phi_m}$; Cur-
    rent state $s$.
Output: Optimal action $a^*$
 1: for $t = 1, \ldots,$ m do
 2:     Select action $a_i = \pi_{\phi_i}(s)$ according to $\pi_{\phi_i}$ and $s$.
 3:     Calculate $Q(s, a_i)$ through Eq. (14).
 4: end for
 5: Choose action through Eq. (15)
 6: return $a^*$.
```

4.1 Experimental Settings

Datasets. To facilitate the testing of RL-based methods' long-term performance, we select three public recommendation datasets: *KuaiRand1K*[1] is a recent dataset for sequential short-video recommendation, and we use the 1K version with irrelevant videos removed. *ML1M*[2] is a subset of the MovieLens dataset, which consists of 1 million user ratings of movies. *RL4RS*[3] is a session-based dataset that was introduced in the BigData Cup 2021 to promote recommendation research in RL. We preprocess them into the sequential recommendation format and split the data based on the recorded timestamps, with the first 75% used for training and the last 25% for evaluation. The preprocessing methods are described in Appendix A.1 and the statistics are provided in Table 1. Note that the RL4RS dataset provides user profile features instead of user IDs, so it does not have a count of unique users in the dataset.

[1] https://kuairand.com/
[2] https://grouplens.org/datasets/movielens/1m/
[3] https://github.com/fuxiAIlab/RL4RS

Table 2: Overall performance of the proposed UOEP and all the baselines on three datasets. The best performance is shown in bold, second best performance is underlined. Experiments are repeated 5 times with different random seeds, and the average and standard deviation are reported.

Algorithms	KuaiRand		ML1M		RL4RS	
	Total Reward	Depth	Total Reward	Depth	Total Reward	Depth
SL	<u>14.16±0.05</u>	<u>14.78±0.04</u>	14.85±0.46	15.40±0.41	7.76±0.25	9.06±0.23
A2C	9.48±0.18	10.60±0.17	13.00±0.46	13.72±0.41	<u>7.83±0.12</u>	<u>9.16±0.11</u>
DDPG	10.66±2.98	11.66±2.66	15.34±0.68	15.83±0.61	7.77±0.95	9.07±0.88
TD3	11.46±1.37	12.38±1.23	15.36±0.45	15.85±0.41	7.62±0.46	8.99±0.38
Wolpertinger	12.44±0.81	13.25±0.72	15.84±0.46	16.27±0.41	7.77±0.43	9.08±0.43
HAC	13.49±0.52	14.18±0.46	<u>15.96±0.32</u>	<u>16.38±0.28</u>	7.66±0.74	9.00±0.64
UOEP	**14.39±0.10**	**14.98±0.09**	**16.59±0.11**	**16.95±0.10**	**8.48±0.53**	**9.60±0.50**

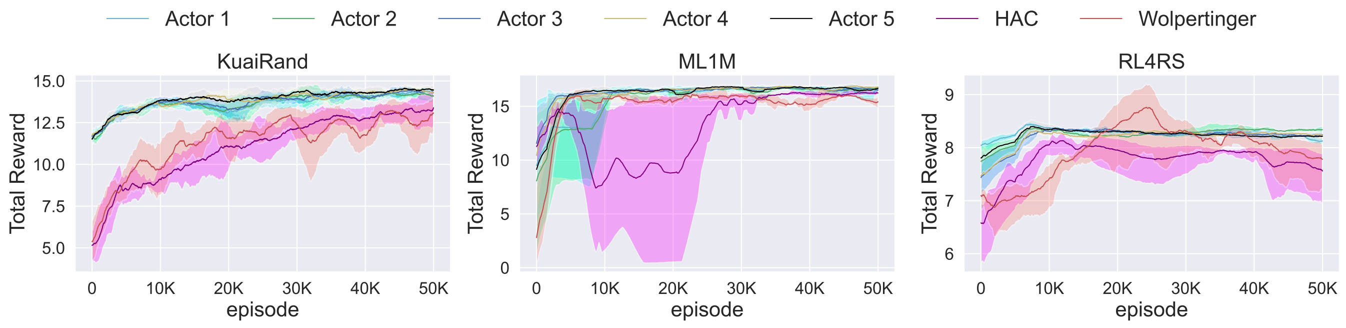

Figure 3: Learning curves for 5 actors of UOEP, HAC, and Wolpertinger on three datasets.

Online Environment Simulator. Following [31], we construct online simulators based on the datasets to capture the reward signals obtained from interactions with users in each round. Specifically, we train a user response model $\Psi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^n$ for each dataset. The user state is derived from static user features and dynamic historical interactions. Ψ outputs the probabilities that the user will provide positive feedback for each item in the recommended list a_t. The final user response, represented by a binary vector $\mathbf{y}_t \in 0, 1^n$ (e.g., click or no click), is uniformly sampled from these probabilities. Notably, our focus here leans towards online evaluation rather than offline settings [11].

Evaluation Metrics and Baselines. We evaluate long-term performance using two metrics: *Total Reward* (sum of rewards in a user session) and *Depth* (number of interactions in a session). These metrics are obtained by simulating user sessions in an online environment with the learned policy. Higher values indicate better performance in both metrics. Appendix A.2 provides detailed information on reward and session designs. Our method is compared with various baselines, including supervised learning method, classic reinforcement learning methods (A2C, DDPG, TD3), Wolpertinger method for large discrete action space, and exploration-based reinforcement learning method HAC:

- **Supervised Learning** model is optimized using observed exposure and user feedback via binary cross-entropy loss.
- **A2C** [33] combines policy gradients and value-based methods using an actor-critic architecture.
- **DDPG** [29] handles continuous action spaces using deterministic policy gradients.
- **TD3** [14] enhances DDPG with twin critics and delayed updates to improve stability.
- **Wolpertinger** [13] embeds discrete actions into a continuous space for efficient reinforcement learning.
- **HAC** [31] decomposes item list generation into hyper-action inference and effect-action selection steps.

4.2 Overall Performance

We train UOEP with $m = 5$ actors and assign their quantile values $\alpha = 0.2, 0.4, 0.6, 0.8, 1.0$, respectively. For each model, we conduct a grid search on the hyperparameters to pick the setting with the best results and perform experiments with 5 random seeds, reporting the mean performance in Table 2. We can observe that our UOEP framework consistently achieves the best performance across all datasets. Compared to the best baselines, it improves performance by 1.6%, 3.9%, and 8.3% on three datasets, indicating the effectiveness of our proposed algorithm. Notably, on the KuaiRand dataset with the largest action space (11643 items), only our algorithm overperforms the supervised learning algorithm, demonstrating the stronger exploration capability in large action space.

For RL-based baselines, A2C demonstrates excellent performance in RL4RS. However, it performs poorly on datasets with larger action space, such as KuaiRand and ML1M. On the other hand, DDPG improves the performance on these datasets by utilizing a deterministic policy. TD3 exhibits similar behavior to DDPG and slightly enhances the quality of recommendations. Wolpertinger, designed for large discrete action space, outperforms other classical reinforcement learning methods in KuaiRand and ML1M datasets. Moreover,

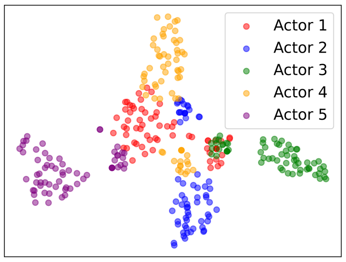

Figure 4: The t-SNE visualization of the population.

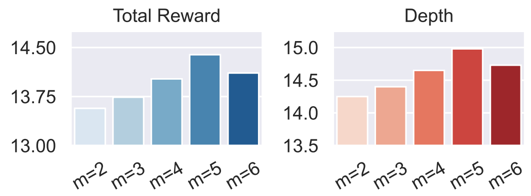

Figure 5: Ablations for the number of actors (denoted by m) in UOEP on KuaiRand.

compared with other baselines, HAC further enhances performance on datasets with large amount of items such as KuaiRand and ML1M, by emphasizing its exploration design. We also provide the learning curve of UOEP, HAC, and Wolpertinger in Figure 3. It can be observed that the learning of all five actors in UOEP is faster and more stable across all three datasets compared to the other two baseline methods. The learning processes of HAC and Wolpertinger exhibit significant fluctuations in ML1M and RL4RS, respectively, further demonstrating the effectiveness and stability of UOEP.

4.3 Further Analysis and Ablation Studies

How does UOEP work? To understand the working process of UOEP, we plot t-SNE embeddings [46] of the five actors learned by UOEP for generating actions. During the testing phase, we randomly sample a subset of users and serve them with the five learned actors. The generated actions, displayed in different colors in Figure 4, form distinct clusters for each actor. This clustering behavior demonstrates that UOEP effectively explores a broad action space customized for different users.

Effect of the Distributional Critic. In our method, we employ a distributional critic that allows actors to optimize at different quantile levels of cumulative rewards. To explore the effects of the distributional critic and the role of optimizing at different quantile levels of cumulative rewards, we disabled the distributional critic in UOEP and replaced it with a deterministic critic. In other words, all our actors optimized the expectation of the cumulative return distribution. We provide the recommendation quality of the distributional and deterministic critic in Table 3. Notably, using the deterministic critic results in a significant decrease in recommendation quality, emphasizing the importance of exploration in different user groups.

Number of Actors in UOEP. The number of actors in UOEP significantly impacts performance. To explore this effect, we conduct experiments to find the optimal number that achieves the best performance. Specifically, we choose number of actors $m \in \{2, 3, 4, 5, 6\}$ and set the quantiles to $\{\frac{1}{m}, \frac{2}{m}, \ldots, 1\}$ accordingly. The results are

Table 3: Ablations for the distributional critic in UOEP on KuaiRand.

Critic	Total Reward	Depth
Distributional	**14.39 ± 0.10**	**14.98 ± 0.09**
Deterministic	13.01 ± 0.42	13.76 ± 0.37

Table 4: Ablations for the regularization losses in UOEP on KuaiRand.

	Total Reward	Depth
UOEP	**14.39 ± 0.10**	**14.98 ± 0.09**
w/o $\mathcal{L}_{\text{div}}$	14.03 ± 0.20	14.66 ± 0.18
w/o $\mathcal{L}_{\text{sta}}$	13.11 ± 0.62	13.85 ± 0.56
w/o $\mathcal{L}_{\text{div}}$, $\mathcal{L}_{\text{sta}}$	11.50 ± 0.90	12.41 ± 0.81

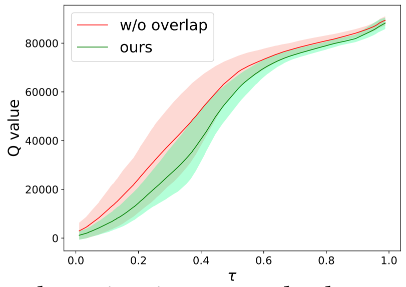

Figure 6: Q-value estimation. We randomly sample an initial state, take all items (11643 in total) as actions, and then sample 100 tau values between 0 and 1 into the distributional critic of the trained UOEP (w/o overlap) and UOEP (original), and then take the average over all actions. We have tested for all five random seeds. The horizontal axis is τ, and the vertical axis is the q value estimated by the critic.

shown in Figure 5. It can be seen that with the increase in the number of actors, the quality of recommendation basically shows an upward trend, indicating that a finer grouping of users eases the exploration process and improves long-term performance. Notably, the performance achieves its peak when $m = 5$ and then declines, probably due to the increased difficulty of learning UOEP as the population size grows. Furthermore, since the training of each actor is independent, parallelizing UOEP is not difficult to implement, allowing us to easily accelerate the training speed.

Effect of Regularization Loss in UOEP. To explore the effects of the designed regularization losses, we disable $\mathcal{L}_{\text{div}}$, $\mathcal{L}_{\text{sta}}$ and both in UOEP, respectively. As shown in Table 4, removing any regularization loss leads to a significant decrease in performance, with removing both having an even larger negative impact. This indicates that improving both diversity and stability in exploration is beneficial for the performance of our algorithm, resulting in higher recommendation quality. Furthermore, removing the stability loss resulted in a marked increase in variance, highlighting its role in minimizing instability during the learning process.

4.4 UOEP Grouping Strategy: Balancing User Activity Levels and Q-Value Accuracy

UOEP's strategy of dividing users into groups based on bottom α values, where $\alpha \in [0.2, 0.4, 0.6, 0.8, 1.0]$, results in a nested structure

Table 5: Overall performance of a non-overlapping version of UOEP against our original (nested) version of UOEP. Experiments are repeated 5 times with different random seeds, and the average and standard deviation are reported.

Algorithms	KuaiRand		ML1M		RL4RS	
	Total Reward	Depth	Total Reward	Depth	Total Reward	Depth
UOEP (w/o overlap)	14.37±0.26	14.97±0.24	16.60±0.06	16.96±0.52	8.24±0.31	9.43±0.28
UOEP (ours)	14.39±0.10	14.98±0.09	16.59±0.11	16.95±0.10	8.48±0.53	9.60±0.50

Table 6: Low-Activity users and fairness performance.

Algorithms	KuaiRand			ML1M			RL4RS		
	$\text{CVaR}_{0.3}$	$\text{CVaR}_{0.4}$	Gini(%)	$\text{CVaR}_{0.3}$	$\text{CVaR}_{0.4}$	Gini(%)	$\text{CVaR}_{0.3}$	$\text{CVaR}_{0.4}$	Gini(%)
SL	1.72	5.40	26.81	5.64	8.08	21.28	2.56	4.62	19.04
A2C	-0.15	0.08	45.22	4.61	6.50	24.24	1.95	3.61	24.17
DDPG	0.06	1.43	36.56	6.57	8.98	19.69	2.44	4.20	20.61
TD3	0.19	1.57	37.56	6.87	9.24	19.09	2.87	4.20	21.27
Wolpertinger	0.19	2.05	34.32	7.67	10.00	17.66	2.69	4.25	20.88
HAC	0.26	2.30	33.82	7.89	10.23	17.25	1.82	3.39	25.16
UOEP (OURS)	**2.48**	**6.01**	**25.67**	**9.51**	**11.59**	**14.64**	**5.94**	**6.76**	**10.54**

among these groups. For instance, the group with $\alpha = 0.4$ encompasses users from the $\alpha = 0.2$ group. This intentional overlap in grouping is underpinned by two fundamental reasons:

Focus on Low-Activity Users. As discussed in the introduction, we can directly exploit the well-established preferences of high-activity users more effectively due to the richer data available. Conversely, low-activity users stand to gain more from high-intensity exploration, as it helps uncover their latent interests. This observation is validated by our validation experiments in the introduction section. In our method, we specifically target the lower activity users by grouping them based on the bottom quantiles of the return distribution. This approach ensures that our system consistently prioritizes the exploration of interests among these users across different actors in the model. As a result, no matter which actor is served during the inference phase, the interests of low-activity users are always taken into account. By targeting the bottom quantiles of the return distribution, we ensure a concentrated focus on low-activity users, thereby capturing their interests more effectively.

Overestimation of Q-Values. In RL, Q-learning algorithms often suffer from Q-value overestimation [12, 23, 24, 28, 38, 47] - the tendency to overestimate the value of actions. Environments with high levels of noise and uncertainty like recommender systems [21] are particularly susceptible to this issue. In our case, when users are grouped into multiple non-overlapping groups, this issue intensifies due to the challenge of accurately evaluating Q-values in more narrowly defined user groups. Such overestimation leads to incorrect assessments of the Q-value, which can impede convergence, destabilize learning, and reduce the efficiency of exploration. To empirically validate our method, we compared the performance of our nested grouping model with a version using non-overlapping groups. The results, detailed in Table 5, indicate a slight performance advantage for the nested model. Furthermore, Figure 6 illustrates the Q-value distributions for both models, clearly showing the heightened overestimation in the non-overlapping version.

In conclusion, while a non-overlapping group approach might seem more straightforward and less redundant, our nested grouping strategy is deliberately chosen. It not only places a targeted emphasis on low-activity users but also effectively counters the challenges of Q-value overestimation. This approach ultimately leads to more effective and stable exploration policies.

4.5 Exploring UOEP's Potential: Low-Activity User Experience and Fairness

Our actor training leverages the CVaR values from the distributional critic, which focuses on the tail-end of the return distribution. Inspired by recent work that integrates CVaR with collaborative filtering to enhance the experiences of low-activity users [44], we hypothesize that UOEP has similar capabilities for long-term performance, which we validate through empirical analysis. Furthermore, we explore UOEP's potential to improve the fairness of the long-term performance within the recommender system. We conduct experiments across 3 datasets with 5 random seeds. To analyze the model's performance on low active users, we choose $\text{CVaR}_{0.3}$ and $\text{CVaR}_{0.4}$ as metrics, representing the average value of the lower 30% and 40% of the Total Reward distribution over the test set, respectively. For assessing fairness, we utilize the Gini coefficient [48] as our metric, with lower values indicating greater fairness.

As summarized in Table 6, UOEP outperforms on both the $\text{CVaR}_{0.3}$ and $\text{CVaR}_{0.4}$ metrics across all three datasets. Additionally, the model achieves significant reductions in the Gini coefficient. These results solidly establish UOEP's effectiveness in serving low-activity users and promoting long-term fairness within the system.

5 RELATED WORK

Reinforcement Learning in Recommender Systems. Deep reinforcement learning, combining deep neural networks with reinforcement learning, has gained attention in recommender systems research. Early works like [39] formulated recommendation as an MDP and experimented with model-based RL. Zheng et al. [60] first applied DQN for news recommendation. Dulac-Arnold et al. [13] enabled RL for large discrete action space. Liu et al. [30] tested actor-critic methods on recommendation datasets. Recently, RL has shown success in real-world applications. Chen et al. [5] scaled batch RL to billions of users. Hu et al. [17] extended DDPG for learning-to-rank. Liu et al. [31] proposed aligned hyper actor-critic learning in large action space. Chen et al. [7] enabled adaptive re-ranking with multi-objective RL without retraining. Overall, deep RL has become an important technique for building recommender systems.

Exploration in Reinforcement Learning. Balancing exploitation and exploration is a key challenge in reinforcement learning. Contextual Bandit algorithms [26, 27, 54, 61] typically involve calculating the upper confidence bound of each arm or sampling actions based on the posterior distribution of observed context. In Deep RL, classic strategies for exploration include epsilon-greedy [42], parameter space noise [37], upper confidence bound algorithms [1], intrinsic motivation techniques like count-based bonuses [2] and prediction error [4]. Novelty search rewards reaching new states regardless of external reward [25]. Recent methods optimize exploration by adapting behavioral diversity [35] or information-theoretic bonuses [16, 34, 41]. However, the above works primarily rely solely on context or state and employ reward-irrelevant exploration strategies. These approaches risk overlooking the interests of users with fewer interactions in the context of recommender systems. Therefore, many RL-based recommender systems prioritize exploration, emphasizing it to enhance long-term user engagement [6, 31, 55, 62]. Notably, large-scale experiments by Google [6] have shown that exploration can significantly enhance user retention and activity levels. These results underscore the importance of integrating RL's exploration capabilities in recommender systems to improve long-term user experiences.

6 CONCLUSION

In this work, we propose UOEP to reinforce user-oriented exploration in streaming recommender systems. UOEP works by first characterizing the activity level of users based on the return distribution under different quantiles. It then learns multiple actors where each actor corresponds to a specific user group with a predefined level of activity. Moreover, a population diversity regularization along with a supervision term is designed to ensure diversity and stability during the actor learning process. We conduct extensive experiments on various public and industrial recommendation datasets. Experimental results demonstrate the advantages of UOEP over previous state-of-the-art algorithms in long-term performance. Further analyses indicate enhanced experience for low-activity users and improved fairness.

ACKNOWLEDGMENTS

This work was funded by the National Science and Technology Major Project (2022ZD0114802), the National Natural Science Foundation of China (No. 62376275, 62377044), Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. Supported by fund for building world-class universities (disciplines) of Renmin University of China. Supported by Public Computing Cloud, Renmin University of China. Supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (23XNKJ13).

A EXPERIMENT SETUP

A.1 Dataset Preprocessing

The three datasets are preprocessed into a unified format, arranging each record in chronological order to include user features, user history, exposed items, user feedback, and timestamps. Then we split them into the first 75% for training and the last 25% for evaluation according to record timestamps.

Similar to the dataset preprocessing in [31], for the ML1M dataset, we consider movies rated 4 and 5 by users as positive samples, and other movies as negative samples. For the KuaiRand dataset, we first remove videos with less than 50 occurrences and then consider videos with a watch time ratio greater than 0.8 as positive samples, and others as negative samples.

For both the ML1M and KuaiRand datasets, we split each user session into sequences of item lists with a length of 10, in chronological order. Only the positive samples before each segmented list are considered historical behavior. The formatted data follows the same structure as the RL4RS dataset.

A.2 Reward and Session Designs

In each round of the session, the recommender system provides the user with a list of items, and the user provides feedback. Our reward function, $r(s_t, a_t)$, is designed as the average reward obtained from all items in the list. Specifically, clicked items receive a reward of 1, while non-clicked items receive a reward of -0.2.

Additionally, each user is assigned an initial temper value at the beginning of the session. In each interaction, the temper value is reduced to varying degrees based on the quality of the recommendation. If the recommendation quality is poor, the user's temperament value decreases rapidly until it reaches 0 or lower, which is considered the end of the user session. Furthermore, there is a maximum depth limit of 20 rounds for each session.

A.3 Model Architectures

Actor. RL methods use SASRec [22] as the actor backbone. User history is encoded into a 32-dimensional vector with positional embeddings, processed by a 2-layer Transformer encoder with 4 heads and a 0.1 dropout rate. The resulting action vector is dot-producted with item embeddings to select top-k items for the user.

Deterministic Critic. For DDPG and TD3, the critic is an MLP with two layers (256 and 64 units), taking encoded user state and action vectors as inputs to output a Q-value.

Value Critic. For A2C, the critic is an MLP with two layers (256 and 64 units), taking the encoded user state vector as input to output a Q-value.

Implicit Quantile Network. For UOEP, the critic processes user state, action vectors, and quantile values through two layers (256 and 64 units) to produce a 16-dimensional vector. This vector is combined with quantile outputs and passed through a 32-unit layer to output quantile values.

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47 (2002), 235–256.

[2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

[3] Marc G Bellemare, Will Dabney, and Rémi Munos. 2017. A distributional perspective on reinforcement learning. In *International conference on machine learning*. PMLR, 449–458.

[4] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. 2018. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355* (2018).

[5] 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 *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[6] Minmin Chen, Yuyan Wang, Can Xu, Ya Le, Mohit Sharma, Lee Richardson, Su-Lin Wu, and Ed Chi. 2021. Values of user exploration in recommender systems. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 85–95.

[7] Sirui Chen, Yuan Wang, Zijing Wen, Zhiyu Li, Changshuo Zhang, Xiao Zhang, Quan Lin, Cheng Zhu, and Jun Xu. 2023. Controllable Multi-Objective Re-ranking with Policy Hypernetworks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3855–3864.

[8] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. 2018. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*. PMLR, 1096–1105.

[9] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. 2018. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[10] Sunhao Dai, Ninglu Shao, Jieming Zhu, Xiao Zhang, Zhenhua Dong, Jun Xu, Quanyu Dai, and Ji-Rong Wen. 2024. Modeling User Attention in Music Recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*.

[11] Romain Deffayet, Thibaut Thonet, Jean-Michel Renders, and Maarten De Rijke. 2023. Offline evaluation for reinforcement learning-based recommendation: a critical issue and some alternatives. In *ACM SIGIR Forum*, Vol. 56. ACM New York, NY, USA, 1–14.

[12] Jingliang Duan, Yang Guan, Shengbo Eben Li, Yangang Ren, Qi Sun, and Bo Cheng. 2021. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE transactions on neural networks and learning systems* 33, 11 (2021), 6584–6598.

[13] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679* (2015).

[14] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[15] Ido Greenberg, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. 2022. Efficient risk-averse reinforcement learning. *Advances in Neural Information Processing Systems* 35 (2022), 32639–32652.

[16] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. Vime: Variational information maximizing exploration. *Advances in neural information processing systems* 29 (2016).

[17] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 368–377.

[18] Guangda Huzhang, Zhenjia Pang, Yongqing Gao, Yawen Liu, Weijie Shen, Wen-Ji Zhou, Qing Da, Anxiang Zeng, Han Yu, Yang Yu, et al. 2021. AliExpress Learning-To-Rank: Maximizing online model performance without going online. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[19] 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. (2019).

[20] Ethan C. Jackson and Mark Daley. 2019. Novelty Search for Deep Reinforcement Learning Policy Network Weights by Action Sequence Edit Metric Distance. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery.

[21] Olivier Jeunen and Bart Goethals. 2021. Pessimistic reward models for off-policy learning in recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 63–74.

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

[23] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[24] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. 2020. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*. PMLR, 5556–5566.

[25] Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.

[26] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

[27] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 297–306.

[28] Zhunan Li and Xinwen Hou. 2019. Mixing update q-value for deep reinforcement learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–6.

[29] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[30] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027* (2018).

[31] Shuchang Liu, Qingpeng Cai, Bowen Sun, Yuhao Wang, Ji Jiang, Dong Zheng, Peng Jiang, Kun Gai, Xiangyu Zhao, and Yongfeng Zhang. 2023. Exploration and Regularization of the Latent Action Space in Recommendation. In *Proceedings of the ACM Web Conference 2023*. 833–844.

[32] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[33] 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*. PMLR, 1928–1937.

[34] Shakir Mohamed and Danilo Jimenez Rezende. 2015. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems* 28 (2015).

[35] Jack Parker-Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts. 2020. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 18050–18062.

[36] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.

[37] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. 2017. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905* (2017).

[38] Prakruthi Prabhakar, Yiping Yuan, Guangyu Yang, Wensheng Sun, and Ajith Muralidharan. 2022. Multi-objective Optimization of Notifications Using Offline Reinforcement Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3752–3760.

[39] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).

[40] Chenglei Shen, Xiao Zhang, Wei Wei, and Jun Xu. 2023. Hyperbandit: Contextual bandit with hypernewtork for time-varying user preferences in streaming recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2239–2248.

[41] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. 2019. Model-based active exploration. In *International conference on machine learning*. PMLR, 5779–5788.

[42] Richard S Sutton and Andrew G Barto. 1999. Reinforcement learning: An introduction. *Robotica* 17, 2 (1999), 229–235.

[43] Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. 2019. Worst cases policy gradients. *arXiv preprint arXiv:1911.03618* (2019).

[44] Riku Togashi, Tatsushi Oka, Naoto Ohsaka, and Tetsuro Morimura. 2023. Safe Collaborative Filtering. *arXiv preprint arXiv:2306.05292* (2023).

[45] Núria Armengol Urpí, Sebastian Curi, and Andreas Krause. 2021. Risk-averse offline reinforcement learning. *arXiv preprint arXiv:2102.05371* (2021).

[46] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[47] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[48] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. 2023. A survey on the fairness of recommender systems. *ACM Transactions on Information Systems* 41, 3 (2023), 1–43.

[49] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1652–1656.

[50] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3597–3606.

[51] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2020), 4753–4766.

[52] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *The world wide web conference*. 2091–2102.

[53] Jun Xu, Xiangnan He, and Hang Li. 2018. Deep learning for matching in search and recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1365–1368.

[54] Xiao Xu, Fang Dong, Yanghua Li, Shaojian He, and Xin Li. 2020. Contextual-bandit based personalized recommendation with time-varying user interests. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 6518–6525.

[55] Surong Yan, Chenglong Shi, Haosen Wang, Lei Chen, Ling Jiang, Ruilin Guo, and Kwei-Jay Lin. 2023. Teach and Explore: A Multiplex Information-guided Effective and Efficient Reinforcement Learning for Sequential Recommendation. *ACM Transactions on Information Systems* (2023).

[56] Yang Yu. 2018. Towards Sample Efficient Reinforcement Learning.. In *IJCAI*. 5739–5743.

[57] Xiao Zhang, Sunhao Dai, Jun Xu, Zhenhua Dong, Quanyu Dai, and Ji-Rong Wen. 2022. Counteracting user attention bias in music streaming recommendation via reward modification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2504–2514.

[58] Xiao Zhang, Haonan Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. 2021. Counterfactual reward modification for streaming recommendation with delayed feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 41–50.

[59] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM conference on recommender systems*. 95–103.

[60] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*. 167–176.

[61] Li Zhou and Emma Brunskill. 2016. Latent contextual bandits and their application to personalized recommendations for new users. *arXiv preprint arXiv:1604.06743* (2016).

[62] Zheqing Zhu and Benjamin Van Roy. 2023. Deep exploration for recommendation systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 963–970.