PrefRec: Recommender Systems with Human Preferences for Reinforcing Long-term User Engagement

Wanqi Xue*
Nanyang Technological University
Singapore
wanqi001@e.ntu.edu.sg

Qingpeng Cai
Kuaishou Technology
Beijing, China
caiqingpeng@kuaishou.com

Zhenghai Xue
Nanyang Technological University
Singapore
zhenghai001@e.ntu.edu.sg

Shuo Sun
Nanyang Technological University
Singapore
shuo003@e.ntu.edu.sg

Shuchang Liu
Kuaishou Technology
Beijing, China
liushuchang@kuaishou.com

Dong Zheng
Kuaishou Technology
Beijing, China
zhengdong@kuaishou.com

Peng Jiang
Kuaishou Technology
Beijing, China
jiangpeng@kuaishou.com

Kun Gai
Unaffiliated
Beijing, China
gai.kun@qq.com

Bo An
Nanyang Technological University
Singapore
boan@ntu.edu.sg

ABSTRACT

Current advances in recommender systems have been remarkably successful in optimizing immediate engagement. However, long-term user engagement, a more desirable performance metric, remains difficult to improve. Meanwhile, recent reinforcement learning (RL) algorithms have shown their effectiveness in a variety of long-term goal optimization tasks. For this reason, RL is widely considered as a promising framework for optimizing long-term user engagement in recommendation. Though promising, the application of RL heavily relies on well-designed rewards, but designing rewards related to long-term user engagement is quite difficult. To mitigate the problem, we propose a novel paradigm, recommender systems with human preferences (or **Pre**ference-based **Rec**ommender systems), which allows RL recommender systems to learn from preferences about users' historical behaviors rather than explicitly defined rewards. Such preferences are easily accessible through techniques such as crowdsourcing, as they do not require any expert knowledge. With PrefRec, we can fully exploit the advantages of RL in optimizing long-term goals, while avoiding complex reward engineering. PrefRec uses the preferences to automatically train a reward function in an end-to-end manner. The reward function is then used to generate learning signals to train the recommendation policy. Furthermore, we design an effective optimization method for PrefRec, which uses an additional value function, expectile regression and reward model pre-training to improve the performance. We conduct experiments on a variety of long-term user engagement optimization tasks. The results show that PrefRec significantly outperforms previous state-of-the-art methods in all the tasks.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Theory of computation** → **Reinforcement learning**.

KEYWORDS

Recommender systems, long-term user engagement, reinforcement learning with human preferences

ACM Reference Format:
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 *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599473

1 INTRODUCTION

Recent recommendation systems have achieved great success in optimizing immediate engagement such as click-through rates [19, 35]. However, in real-life applications, long-term user engagement is more desirable than immediate engagement because it directly affects some important operational metrics, e.g., daily active users (DAUs) and dwell time [44]. Despite the great importance, how to effectively optimize long-term user engagement remains a significant challenge for existing recommendation algorithms. The difficulties are mainly on i) the evolving of long-term user engagement lasts for a long period; ii) factors that affect long-term user engagement are usually non-quantitative, e.g., users' satisfactory; and iii) the learning signals which are used to update our recommendation strategies are sparse, delayed, and stochastic. When trying to optimize long-term user engagement, it is very hard to relate the changes in long-term user engagement to a single recommendation [41]. Moreover, the sparsity in observing the evolution of long-term user engagement makes the problem even more difficult.

*The work was done during an internship at Kuaishou Technology.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599473

Reinforcement learning (RL) has demonstrated its effectiveness in a wide range of long-term goal optimization tasks, such as board games [33, 34], video games [11, 39], robotics [25] and algorithmic discovery [13]. Conventionally, when trying to solve a real-world problem with reinforcement learning, we need to first formulate it as a Markov Decision Process (MDP) and then learn an optimal policy that maximizes cumulative rewards or some other user-defined reinforcement signals in the defined MDP [7, 36]. Considering the characteristic that RL seeks to optimize cumulative rewards, it is rather suitable for optimizing long-term signals, such as user stickiness, in recommendation [41, 53]. As in Figure 1, recommender systems can be modeled as an agent to interact with users which serve as the environment. Each time the agent completes a recommendation request, we can record the feedback and status changes of users to calculate a reward as well as the new state for the agent. Applying RL will lead to a recommendation policy which optimizes user engagement from a long-term perspective.

Despite that RL is an emerging and promising framework to optimize long-term engagement in recommendation, it heavily relies on a delicately designed reward function to incentivize recommender systems behave properly. However, designing an appropriate reward function is very difficult especially in large-scale complex tasks like recommendation [5, 6, 10, 41]. On one hand, the reward function should be aligned with our ultimate goal as much as possible. On the other hand, rewards should be sufficiently dense and instructive to provide step-by-step guidance to the agent. For immediate engagement, we can simply use metrics such as click-through rates to generate rewards [48, 51]. Whereas for long-term engagement, the problem becomes rather difficult because attributing contributions to long-term engagement to each step is really tough. If we only assign rewards when there is a significant change in long-term engagement, learning signals could be too sparse for the agent to learn a policy. Existing RL recommender systems typically define the reward function empirically [10, 44, 53] or use short-term signals as surrogates [41], which will severely violate the aforementioned requirements of consistency and instructivity.

To mitigate the problem, we propose a new training paradigm, recommender systems with human preferences (or **Pre**ference-based **Rec**ommender systems), which allows RL recommender systems to learn from human feedback/ preferences on users' historical behaviors rather than explicitly defined rewards. We demonstrate that RL from human preferences (or preference-based RL), a framework that has led to successful applications such as ChatGPT [29], is also applicable to recommender systems. Specifically, in PrefRec, there is a (virtual) teacher giving feedback about his/her preferences on pairs of users' behaviors. We use the feedback (stored in a preference buffer) to automatically train a reward model which generates learning signals for recommender systems. Such preferences are easy to obtain because no expert knowledge is required, and we can use technologies such as crowdsourcing to easily gather a large number of labeled data. Furthermore, to overcome the problem that the reward model may not work well for some unseen actions, we introduce a separate value function, trained by expectile regression, to assist the training of the critic in PrefRec[1]. Our main contributions are threefold:

[1]PrefRec adopts the framework of actor-critic in reinforcement learning.

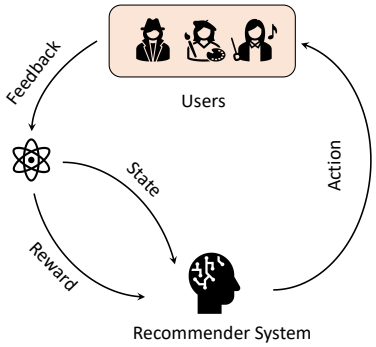

Figure 1: Reinforcement learning recommender systems.

- We propose a new framework, recommender systems with human preferences, to optimize long-term engagement. Our method can fully exploit the advantages of reinforcement learning in optimizing long-term goals, while avoiding the complex reward engineering in reinforcement learning.
- We design an effective optimization method for the proposed framwork, which uses an additional value function, expectile regression and reward model pre-training to improve the performance.
- We collect the first reinforcement learning from human feedback (RLHF) dataset for long-term engagement optimization problem in recommendation and propose three new tasks to evaluate performance of recommender systems. Experimental results demonstrate that PrefRec significantly outperforms previous state-of-the-art approaches on all the tasks.

2 PRELIMINARIES

2.1 Long-term User Engagement in Recommendation

Long-term user engagement is an important metric in recommendation, and typically reflects as the stickiness of users to a product. In general, given a product, we expect users to spend more time on it and/or use it as frequently as possible. In this work, we assume that users interact with the recommender systems on a session basis: when a user accesses a product, such as an App, a session begins, and it ends when the user leaves. During each session, users can launch an arbitrary number of recommendation requests as they want. Such session-based recommender systems has been widely deployed in real-life applications such as short-form videos recommendation and news recommendation [40, 45, 50, 51]. We are particularly interested in increasing

i the number of recommended items that users consume during each visit;
ii the frequency that users visit the product.

Optimizing these two indicators is nontrivial because it is difficult to relate them to a single recommendation. For example, if a user increases its visiting frequency, we are not able to know exactly which recommendation leads to the increase. To this end,

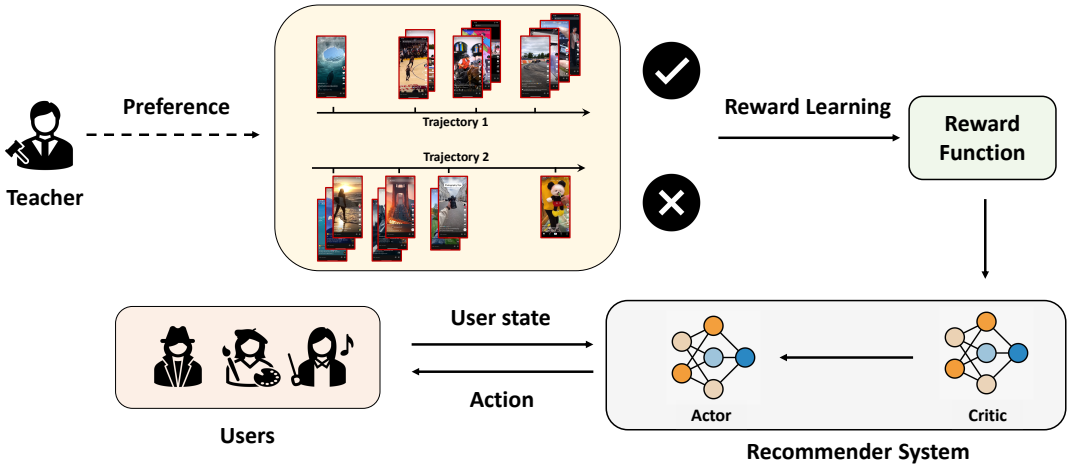

Figure 2: The framework of recommender systems with human preferences. A teacher provides feedback about his/her preferences between users' behavioral trajectories. Trajectory 1 demonstrates a trend from low-active to high-active and trajectory 2 shows an opposite tendency. Therefore, the teacher will prefer trajectory 1 to trajectory 2. With feedback of preferences, we can automatically train a reward function in an end-to-end manner. The preference-based recommender systems is then optimized by using rewards predicted by the learned reward function.

we propose to use reinforcement learning to take into account the potential impact on the future when making decisions.

2.2 Recommendation as a Markov Decision Process (MDP)

Applying RL to recommender systems requires defining recommendation as a Markov Decision Process (MDP). Recommender systems can be described as an agent to interact with users, who act as the environment. Formally, we formulate recommendation as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$:

- $\mathcal{S}$ is the continuous state space. $s \in \mathcal{S}$ indicates the state of a user which contains static information such as gender and age; and dynamic information, such as the rate of likes and retweets. A state is what the recommender systems relies on to make decisions.
- $\mathcal{A}$ is the continuous action space, where $a \in \mathcal{A}$ is an action which has the same dimension as the representation of recommendation items. We determine the item to recommend by comparing the similarity of an action and item representations [44, 48].
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition function, where $p(s_{t+1}|s_t, a_t)$ defines the probability that the next state is s_{t+1} after recommending an item a_t at the current state s_t.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. $r(s_t, a_t)$ determines how much the agent will be rewarded after recommending a_t at state s_t.
- γ is the discount factor. γ determines how much the agent cares about rewards in the distant future relative to those in the immediate future.

The recommendation policy $\pi(a|s) : \mathcal{S} \to \mathcal{A}$ is defined as a mapping from state to action. Given a policy $\pi(a|s)$, we define a state-action value function (Q-function) $Q^\pi(s, a)$ which generates the expected cumulative reward (return) of taking an action a at state s and thereafter following π:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{(s_{t'}, a_{t'}) \sim \pi} \left[r(s_t, a_t) + \sum_{t'=t+1}^{\infty} \gamma^{(t'-t)} \cdot r(s_{t'}, a_{t'}) \right]. \tag{1}$$

We seek to optimize the policy $\pi(a|s)$ so that the return obtained by the recommendation agents is maximized:

$$\max_\pi \mathcal{J}(\pi) = \mathbb{E}_{s_t \sim d_t^\pi(\cdot), a_t \sim \pi(\cdot|s_t)} \left[Q^\pi(s_t, a_t) \right], \tag{2}$$

where $d_t^\pi(\cdot)$ denotes the state visitation frequency at step t under the policy π. By optimizing the above objective, the agent can achieve the largest cumulative return in the defined MDP.

3 CHALLENGES OF DESIGNING THE REWARD FUNCTION

Despite that RL is a promising approach to optimize long-term user engagement, the application of RL heavily relies on a well-designed reward function. The reward function must be able to reflect the changes in long-term engagement at each time step. In the meantime, it should be able to provide instructive guidance to the agent for optimizing the policy. Practically, quantifying rewards properly is very challenging because it is really difficult to relate changes in long-term engagement to a single recommendation [41]. For example, when recommending a video to a user, we have no way of knowing how many videos the user will continue to watch on the platform before exiting the current session, and obviously, how this amount will be affected by the recommendation is even harder to know. For this reason, it is challenging to give a reward regarding the impact of recommended videos on the average video consumption of users. Similarly, recommending a video cannot be used to predict when the user will revisit the platform after

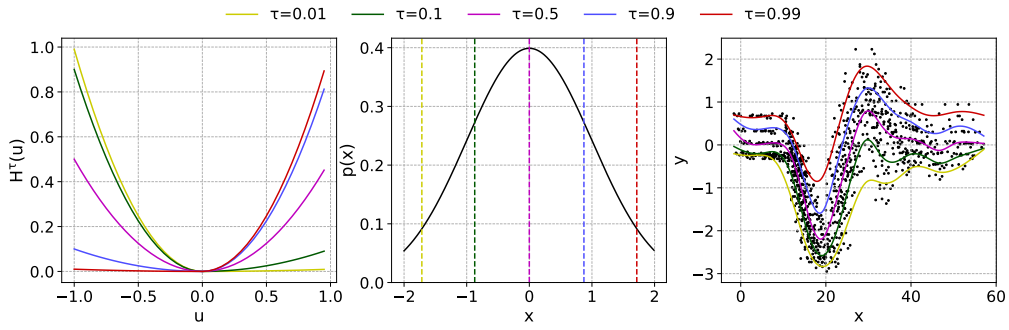

Figure 3: Left: Illustration of the expectile loss function $H^\tau(u)$. $\tau = 0.5$ corresponds to the MSE loss, and $\tau > 0.5$ upweights positives differences u. Center: Expectiles of a Gaussian distribution. Right: An example of expectile regression on a two-dimensional distribution.

exiting the current session. Designing rewards related to the visiting frequency of users is also difficult. As a compromise solution, one could only assign rewards at the beginning or the end of a session. However, this kind of reinforcement signals will be too sparse for the recommender systems to learn a reasonable policy, especially when the session length is large [44]. Existing methods either design the reward function highly empirically [10, 44, 53] or use immediate engagement signals as surrogates [41], which will cause deviation between the optimization objective and the real long-term engagement. For this reason, it is urgent to propose a framework to address the difficulties in reward designing when using RL to optimize long-term user engagement.

4 RECOMMENDER SYSTEMS WITH HUMAN PREFERENCES

To resolve the difficulties in designing the reward function, we propose a novel paradigm, **Pef**erence-based **Rec**ommender systems (PrefRec), which allows an RL recommender systems to learn from preferences on users' historical behaviors rather than explicitly defined rewards. By this way, we can overcome the problems in designing the reward function when optimizing long-term user engagement. In this section, we first introduce how to utilize preferences to generate reinforcement signals for learning a recommender systems. Then, we discuss how to optimize the performance of PrefRec by using expectile regression to better estimate the value function. Next, we propose to pre-train the reward function to stabilize the learning process. Last, we summarize the algorithm.

4.1 Reinforcing from Preferences

While the reward for a recommendation request is hard to obtain, preferences between two trajectories of users' behaviors are easy to determine. For example, if one trajectory shows a transition from low-active to high-active and the other shows an opposite trend or an insignificant change, we can easily indicate the preference between them. Labeling preferences does not require any expert knowledge, so we can easily use techniques such as crowdsourcing to obtain a large amount of feedback on preferences. In PrefRec, we assume that there is a human teacher providing preferences between user's behaviors and the recommender systems uses this feedback to perform the task. There are mainly two advantages in using preferences: i) labeling the preference between a pair of trajectories is quite simple compared to designing rewards for every step; and ii) the recommender systems is incentivized to learn the preferred behavior directly because reinforcement signals come from preferences.

We provide the framework of recommender systems with human preferences in Figure 2. As we can find, there is a teacher providing preferences between a pair of users' behavioral trajectories[2]. The teacher could be humans or or even a program with labeling criteria. For trajectory 1, the user increases its visiting frequency gradually and consumes more and more items in each session, which indicates that it is satisfied with the current recommendation policy. On the contrary, trajectory 2 shows the user is becoming less and less active, suggesting that the current recommender system should be improved to better serve the user. The teacher will obviously prefer trajectory 1 to trajectory 2. After generating such preference data, we use them to automatically train a reward function $\hat{r}(s, a; \psi)$, parameterized by ψ, in an end-to-end manner. The preference-based recommender systems uses the predicted reward $\hat{r}(s, a; \psi)$ rather than hand-crafted rewards to update its policy.

Formally, a trajectory σ is a sequence of observations and actions $\{(s_1, a_1), \ldots, (s_T, a_T)\}$. Given a pair of trajectories (σ^0, σ^1), a teacher provides a feedback indicating which trajectory is preferred, i.e., $y = (0, 1)$, $(1, 0)$ or $(0.5, 0.5)$, where $(0, 1)$ indicates trajectory σ^1 is preferred to trajectory σ^0, i.e., $\sigma^1 \succ \sigma^0$; $(1, 0)$ indicates $\sigma^0 \succ \sigma^1$; and $(0.5, 0.5)$ implies an equally preferable case. Each feedback is triple (σ^0, σ^1, y) which is stored in a preference buffer $\mathcal{D}_p = \{((\sigma^0, \sigma^1, y))_i\}_{i=1}^N$. We use a deep neural network with parameters ψ to predict the reward $\hat{r}(s, a; \psi)$ at a specific step (s, a). By following the Bradley-Terry model [4, 30, 43], we assume that the teacher's probability of preferring a trajectory depends exponentially on the accumulated sum of the reward over the trajectory. Then the probability that trajectory σ^1 is preferred to trajectory σ^0 can be written as a function of $\hat{r}(s, a; \psi)$:

$$P_\psi\left[\sigma^1 \succ \sigma^0\right] = \frac{\exp\left(\sum_t \hat{r}\left(\mathbf{s}_t^1, \mathbf{a}_t^1; \psi\right)\right)}{\sum_{i\in\{0,1\}} \exp\left(\sum_t \hat{r}\left(\mathbf{s}_t^i, \mathbf{a}_t^i; \psi\right)\right)}. \tag{3}$$

[2]For a pair of trajectories, they may come from different users or from different periods of the same user.

Algorithm 1: Recommender Systems with Human Preferences

Input: Preference buffer $\mathcal{D}_p = \{(\sigma^0, \sigma^1, y)_i\}_{i=1}^N$, replay buffer $\mathcal{D}_r = \{(s, a, s')_i\}_{i=1}^M$, pre-train episodes K

Initialize the reward model $r(s, a; \psi)$, the Q-function $Q(s, a; \theta)$, the V-function $V(s; \eta)$, the recommendation policy $\pi(\cdot|s; \mu)$
Set soft-update rate λ and initialize the target Q-function $Q(s, a; \hat{\theta})$
for *pre-train episodes* $k = 1$ *to* K **do**
  Sample a mini-batch of preferences (σ^0, σ^1, y) from $\mathcal{D}_p$
  Update the reward model $r(s, a; \psi)$: $\psi \leftarrow \psi - \nabla_\psi \mathcal{L}_\psi$ (Eq. 4)
end
while *not end training* **do**
  Sample a mini-batch of transitions (s, a, s') for $\mathcal{D}_r$
  Label the sampled transitions by the reward model to obtain $(s, a, \hat{r}, s')$
  Update the V-function $V(s; \eta)$: $\eta \leftarrow \eta - \nabla_\eta \mathcal{L}_\eta^V$ (Eq. 9)
  Update the Q-function $Q(s, a; \theta)$: $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}_\theta^Q$ (Eq. 6)
  Update the recommendation policy $\pi(\cdot|s; \mu)$: $\mu \leftarrow \mu - \nabla_\mu \mathcal{L}_\mu^P$ (Eq. 10)
  Soft-update the target Q-function $Q(s, a; \hat{\theta})$: $\hat{\theta} \leftarrow \lambda\theta + (1 - \lambda)\hat{\theta}$
  if *fine-tune reward model* **then**
    Sample a mini-batch of preferences (σ^0, σ^1, y) from $\mathcal{D}_p$
    Update the reward model $r(s, a; \psi)$: $\psi \leftarrow \psi - \nabla_\psi \mathcal{L}_\psi$ (Eq. 4)
  end
end

Since the preference buffer $\mathcal{D}_p = \{((\sigma^0, \sigma^1, y))_i\}_{i=1}^N$ contains true labels of preferences, we can train the reward model $\hat{r}(s, a; \psi)$ through supervised learning, updating it by minimizing the cross-entropy loss:

$$\mathcal{L}_\psi = - \mathop{\mathbb{E}}_{(\sigma^0, \sigma^1, y) \sim \mathcal{D}_p} \left[y(0) \log P_\psi \left[\sigma^0 \succ \sigma^1\right] + y(1) \log P_\psi \left[\sigma^1 \succ \sigma^0\right] \right], \tag{4}$$

where $y(0)$ and $y(1)$ are the first and second element of y, respectively. After learning the reward model $\hat{r}(s, a; \psi)$, we can use it to generate learning signals for training the recommendation policy.

4.2 Optimizing the Recommendation Policy

When learning the recommendation policy, there is a replay buffer $\mathcal{D}_r$ storing training data. Unlike conventional RL recommender systems, we store (s, a, s') rather than (s, a, r, s') in the buffer $\mathcal{D}_r$ because we cannot obtain explicit rewards from the environment[3]. We utilize the learned reward model $\hat{r}(s, a; \psi)$ to label the reward for a tuple (s, a). After labelling, an intuitive approach is to learn the Q-function by minimizing the following temporal difference (TD) error:

$$\mathcal{L}_\theta^Q = \mathop{\mathbb{E}}_{(s,a,s') \sim \mathcal{D}_r} \left[\left(\hat{r}(s, a; \psi) + \gamma Q(s', \pi(\cdot|s'); \hat{\theta}) - Q(s, a; \theta) \right)^2 \right], \tag{5}$$

where $\mathcal{D}_r$ is the replay buffer, $Q(s, a; \theta)$ is a parameterized Q-function, $Q(s, a; \hat{\theta})$ is a target network (e.g., with soft updating of parameters defined via Polyak averaging), and $\pi(\cdot|s)$ is the recommendation policy. However, in practice, we find that this method does not perform well. It may be because the recommendation policy $\pi(\cdot|s)$ will choose significantly different actions from the stored data, making the reward function and the Q-function unable to predict the corresponding values. To resolve this, we introduce a separate value function (V-function) that predicts how good or bad a state is. By doing so, we can eliminate the uncertainty that comes with the recommended policy. Instead of minimizing the TD error in Eq. 5, we turn to minimize the following loss to learn the Q-function:

$$\mathcal{L}_\theta^Q = \mathop{\mathbb{E}}_{(s,a,s') \sim \mathcal{D}_r} \left[\left(\hat{r}(s, a; \psi) + \gamma V(s'; \eta) - Q(s, a; \theta) \right)^2 \right], \tag{6}$$

where $V(s'; \eta)$ is the V-function with parameters η. Given the replay buffer $\mathcal{D}_r$, the relationship between Q-function and V-function is

$$V(s) = \mathbb{E}_a[Q(s, a)], (s, a) \in \mathcal{D}_r. \tag{7}$$

Conventionally, we can optimize the parameters of the V-function by minimizing the following Mean Squared Error (MSE) loss:

$$\mathcal{L}_\eta^V = \mathop{\mathbb{E}}_{(s,a) \sim \mathcal{D}_r} \left[\left(Q(s, a; \hat{\theta}) - V(s; \eta) \right)^2 \right]. \tag{8}$$

However, such V-function corresponds to the behavior policy which collects the replay buffer $\mathcal{D}_r$. We want to achieve improvement upon the behavior policy. Inspired by expectile regression [23, 24], we let the V-function to regress the τ expectile ($\tau \geq 0.5$) of $Q(s, a)$ rather than the mean statistics as in Eq. 7. Then the loss for V-function becomes:

$$\mathcal{L}_\eta^V = \mathop{\mathbb{E}}_{(s,a) \sim \mathcal{D}_r} \left[H^\tau (Q(s, a; \hat{\theta}) - V(s; \eta)) \right], \tag{9}$$

where $H^\tau(u) = |\tau - \mathbb{I}(u < 0)|u^2$, $\mathbb{I}(\cdot)$ is the indicator function. Specially, if $\tau = 0.5$, Eq. 8 and Eq. 9 are identical. For $\tau > 0.5$, this asymmetric loss (Eq. 9) downweights the contributions of $Q(s, a; \hat{\theta})$ smaller than $V(s; \eta)$ while giving more weights to larger values (as in Fig. 3, left). Fig. 3 (right) illustrates expectile regression on a two-dimensional distribution: increasing τ leads to more data points below the regression curve. Back to the learning of the V-function, the purpose is to let $V(s; \eta)$ to regress an above-average value. The Q-function is jointly trained with the V-function, while it also serves as the critic to guide the update of the recommendation policy, i.e., the actor. The recommendation policy is optimized by minimizing the loss:

$$\mathcal{L}_\mu^P = - \mathop{\mathbb{E}}_{(s,a) \sim \mathcal{D}_r} \left[Q(s, \pi(\cdot|s; \mu); \theta) \right], \tag{10}$$

where $\pi(\cdot, s; \mu)$ is the recommendation policy with parameters μ.

4.3 Pre-training the Reward Function

The reward function is used to automatically generate reinforcement signals to train the recommendation policy. However, A reward model that is not well-trained will cause the collapse of the

[3] With a slight abuse of notation, we use s' to denote the next state.

recommendation policy. Thus, to stabilize the training, we propose to pre-train the reward function before starting the updating of the recommendation policy. Specifically, we first prepare a preference buffer that stores a pool of preference feedback between interaction histories. Then we initialize a deep neural network with the structure of multi-layer perceptron to learn from the preference buffer. The neural network is updated by minimizing the loss in Eq. 4. After training for several episodes, we start the training of the recommendation policy. At this phase, the reward model is updated simultaneously with the recommender systems. By doing so, the reward model can handle potential shift in human preferences and can therefore generate more accurate learning signals.

4.4 Overall Algorithm

We provide the overall algorithm of PrefRec in Algorithm 1. A preference buffer $\mathcal{D}_p$ and a replay buffer $\mathcal{D}_r$ is required for training PrefRec. From lines 1 to 2, we do the initialization for the deep neural networks and set hyper-parameters such as soft-update rate. From lines 3 to 5, we pre-train the reward model to confirm that it is able to provide reasonable learning signals when updating the recommendation policy. Lines 7 to 18 describe the training process of the recommendation policy. We first sample a mini-batch of transitions from the replay buffer $\mathcal{D}_r$. Then we lable the samples data with the reward model. Following that, we update the parameters of the V-function, the Q-function and the recommendation policy accordingly. Finally, if fine-tuning the reward model, we will train the reward model to keep it up-to-date.

4.5 Discussions

PrefRec differs from both inverse reinforcement learning (IRL) [28] and model-based RL [54]. The key differences between PrefRec and IRL include that IRL requires costly expert demonstrations while PrefRec only requires simple label-like feedback. The reward function in PrefRec is learned by aligning with human preferences, whereas in IRL it is inferred from expert demonstrations. On the other hand, model-based RL relies on environmental rewards and aims to simplify learning by approximating the reward and transition functions. In contrast, PrefRec does not require environmental rewards and is designed to be learned without them.

5 EXPERIMENTAL RESULTS

We conduct extensive experiments to evaluate our algorithm. In particular, we will answer the following research questions (RQs):

- **RQ1**: Can the framework of PrefRec lead to improvement in long-term user engagement?
- **RQ2**: Whether PrefRec is able to outperform existing state-of-the-art methods?
- **RQ3**: Whether the learned reward signals is able to reflect the true underlying rewards?
- **RQ4**: How do the components in PrefRec contribute to the performance?

5.1 Preparing the dataset

Since PrefRec is a new framework in recommendation, there is no available dataset for evaluation. To prepare the dataset, we track

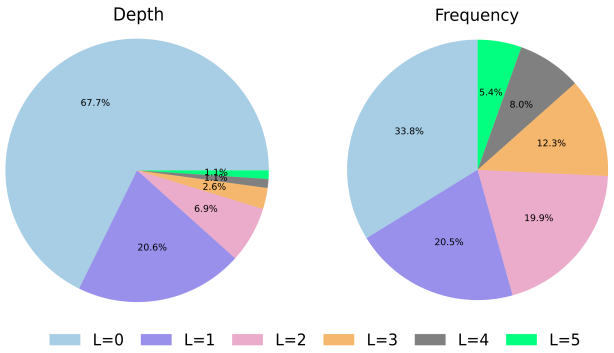

Figure 4: The proportion of the levels in session depth and visiting frequency.

complete interaction histories of around 100, 000 users from a leading short-form videos platform for months, during which over 25 millions recommendation services are provided[4]. For each user, we record its interaction history in a session-request form: the interaction history consists of several sessions with each session containing a number of recommendation requests (see Sec. 2.1). The recommender system provides service at each recommendation request where we record the state of a user and the action of the recommender system. Each state is a 245-dimensional vector containing the user's state information, such as gender, age and historical like rate. We applied scaling, normalization and clipping to each dimension of states in order to stabilize the input of models. An action is a 8-dimensional vector which is the representation of recommended item at a request. We record the timestamp of each time a user starts and exits a session. With the timestamp, we can calculate the duration that a user revisits the platform and thus can infer the visiting frequency. We measure changes in long-term user engagement at the end of each session for each user. Specifically, we first calculate the average session depth δ^u_{avg} and the average revisiting time ϵ^u_{avg} for each user u during the time span. For each session, δ^u_i denotes the number of requests in the i-th session of the user u. If δ^u_i is larger than the average session depth δ^u_{avg}, we consider that there is an improvement in session depth. Similarly, we can calculate the revisiting time ϵ^u_i for session i and if it is less than the average revisiting time ϵ^u_{avg}, we consider that there is an improvement in visiting frequency. We quantify the changes in session depth and visiting frequency into six levels (level 0 to level 5; the more higher level, the more positive change) where the levels are determined by the following equations, respectively:

$$\begin{aligned} \mathbf{L}^u_i(depth) &= \left(\lfloor \frac{\delta^u_i}{\delta^u_{avg}} \rfloor \right).clip(0,5) \\ \mathbf{L}^u_i(frequency) &= \left(\lfloor \frac{\epsilon^u_{avg}}{\epsilon^u_i} \rfloor \right).clip(0,5) \end{aligned} \tag{11}$$

We provide the proportion of the calculated levels of all sessions in Fig. 4. For session depth, most of sessions stay in level 0 and only

[4]Data samples and codes are at https://www.dropbox.com/sh/hgsqg5fabnvmp26/AABF-2dvarI_bdyygYEt5aw7a?dl=0.

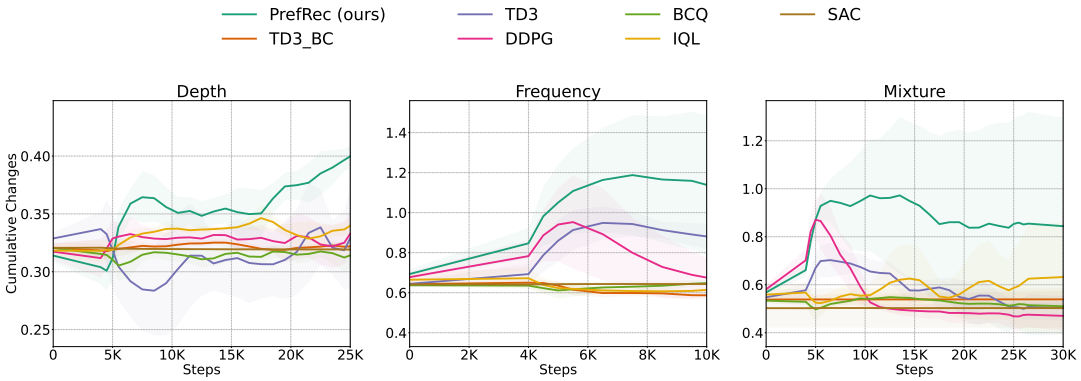

Figure 5: Learning curves of RL recommender systems under the framework of PrefRec, averaged over 5 runs.

very few of them are located in level 4 and 5. The visiting frequency demonstrates a more even distribution.

After processing the data, we can prepare the two buffers, i.e., the preference buffer $\mathcal{D}_p$ and the replay buffer $\mathcal{D}_r$, which are used in PrefRec. For the preference buffer $\mathcal{D}_p$, we uniformly sample 20,000 pairs of users who have launched for more than 200 recommendation requests in the platform. We set the length of trajectory σ as 100 and randomly sample a segment of trajectories with this length from the interaction histories of the selected users. To generate the preferences, we write a scripted teacher who provides its feedback by comparing cumulative levels of changes on the trajectory segments. For the replay buffer $\mathcal{D}_r$, we randomly sample 80% of users as the training set and split their interaction histories into transitions $(s, a, s')$to fill up the replay buffer $\mathcal{D}_r$.

5.2 Baselines

We compare PrefRec with a variety of baselines, including reinforcement learning methods for off-policy continues control (DDPG, TD3, SAC), offline reinforcement learning algorithms (TD3_BC, BCQ, IQL), and imitation learning:

- **DDPG** [26]: An off-policy reinforcement learning algorithm which concurrently learns a Q-function and a deterministic policy. The update of the policy is guided by the Q-function.
- **TD3** [15]: A reinforcement learning algorithm which is designed upon DDPG. It applies techniques such as clipped double-Q learning, delayed policy updates, and target policy smoothing.
- **SAC** [18]: An off-policy reinforcement learning algorithm trained to maximize a trade-off between expected return and entropy, a measure of randomness in the policy. It also incorporates tricks such as the clipped double-Q learning.
- **TD3_BC** [14]: An offline reinforcement learning algorithm designed based on TD3. It applies a behavior cloning (BC) term to regularize the updating of the policy.
- **BCQ** [16]: An offline algorithm which restricts the action space in order to force the agent towards behaving similarly to the behavior policy.
- **IQL** [24]: An offline reinforcement learning method which uses expectile regression to estimate the value of the best action in a state.
- **IL**: Imitation learning treats the behaviors in the replay buffer as expert knowledge and learns a mapping from observations to actions by using expert knowledge as supervisory signals.

Since our work focuses on addressing complex reward engineering when reinforcing long-term engagement and how to convey human intentions to RL-based recommender systems, we mainly make comparisons with classical RL algorithms. Works like FeedRec [53] emphasizes on designing DNN architecture and is orthogonal to PrefRec which focuses on the policy optimization process.

5.3 Evaluation Metric

Among the 10,0000 users, we sample 80% of them as the training set and the remaining 20% users constitute the test set. For the test users, we store their complete interaction histories separately, in the session-request format. We adopt Normalised Capped Importance Sampling (NCIS) [37], a widely used standard offline evaluation method [12, 17], to evaluate the performance. Formally, the score of a policy π is calculated by

$$\tilde{J}^{NCIS}(\pi) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \left[\frac{\sum_{i=0}^{|\mathcal{T}_u|} \tilde{\rho}_i(\pi, \mathcal{T}_u) \mathbf{L}_i^u}{\sum_{i=0}^{|\mathcal{T}_u|} \tilde{\rho}_i(\pi, \mathcal{T}_u)} \right], \tag{12}$$

where $\mathcal{U}$ is the set of test users, $\mathcal{T}_u$ is the set of sessions of the user u, $\tilde{\rho}_i(\pi, \mathcal{T}_u)$ is the probability that the policy π follows the request trajectory of the i-th session in $\mathcal{T}_u$, and $\mathbf{L}_i^u$ is the level of change for the i-the session (as defined in Eq. 11). Intuitively, $\tilde{J}^{NCIS}(\pi)$ awards a policy with a high score if the policy has large probability to follow a good trajectory.

5.4 Implementation Details

To ensure fairness in comparison across all methods and experiments, a consistent network architecture is utilized. This architecture consists of a 3-layer Multi-Layer Perceptron (MLP) with 256 neurons in each hidden layer. The hyper-parameters for the PrefRec method are listed in Table 1. All methods were implemented using the PyTorch framework.

5.5 Overall Results

We conduct experiments to verify if the framework of PrefRec can improve long-term user engagement in terms of i) session depth;

Table 1: Hyper-parameters of PrefRec.

Hyper-parameter	Value
Optimizer	Adam [22]
Actor Learning Rate	5×10^{-6}
Critic Learning Rate	5×10^{-5}
State Dimensions	245
Action Dimensions	8
Transitions Batch Size	4096
Preferences Batch Size	256
Normalized Observations	Ture
Gradient Clipping	False
Fine-Tuning	True
Discount Factor	0.9
Expectile Rate	0.7
Soft-update Rate	0.999
Segment Length	100
Preference Buffer Size	2×10^4
Replay Buffer Size	3×10^6
Number of Pre-train Epoch	3
Number of Train Epoch	5

ii) visiting frequency; and iii) a mixture of the both. We randomly sample 500 users from the test set and use them to plot the learning curves of our method and the baselines. As in Fig. 5, PrefRec achieves a significant and consistent increase in cumulative long-term engagement changes in all the tasks, although it does not receive any explicit reinforcement signal. Similar phenomenon can also be observed in some of those generic reinforcement learning algorithms, such as DDPG. They demonstrates growth in specific tasks, though not that stable. The learning curves indicate that the learned reward function is able to provide reasonable reinforcement signals. and the framework of PrefRec provides an effective training paradigm to achieve reward-free recommendation policy learning. Next, we save the models with the best performance in training and test their performance on the whole test set. As in Table 2, those generic reinforcement learning algorithms poorly without explicit rewards. PrefRec is able to outperform all the baselines by a wide margin in all the three tasks, showing the effectiveness of the proposed optimization methods. Despite utilizing only a single dataset, optimizing session depth and visiting frequency are distinct challenges. The results of the experiments demonstrate the generalization ability of PrefRec as it consistently delivers improvements across both tasks. The optimization of session depth and visiting frequency are not necessarily interdependent; the algorithm that produces a deep session may not result in high visiting frequency, and similarly, high visiting frequency does not guarantee a deep session (as seen in Fig. 5).

5.6 Training Process of the Reward Function

To ensure the validity of the learning signals generated by the reward function, it must accurately predict the preferences that are utilized in the training phase. To evaluate the performance of the reward function, we have plotted its prediction accuracy in Fig. 6. The results show a noticeable improvement in accuracy as the training process advances. Additionally, we also plot the learning loss

Table 2: Overall performance comparisons on various long-term user engagement optimization tasks. The "±" indicates 95% confidence intervals.

	Depth	Frequency	Mixture
DDPG [26]	0.3211 ±0.0060	1.1408 ±0.0163	1.0642 ±0.0119
TD3 [15]	0.3495 ±0.0038	0.9714 ±0.0141	0.6423 ±0.0097
SAC [18]	0.3190 ±0.0031	0.6416 ±0.0058	0.5348 ±0.0047
TD3_BC [14]	0.3246 ±0.0032	0.6781 ±0.0060	0.5326 ±0.0047
BCQ [16]	0.3142 ±0.0033	0.6580 ±0.0060	0.5528 ±0.0052
IQL [24]	0.3311 ±0.0054	1.0354 ±0.0141	0.8230 ±0.0099
IL	0.3202 ±0.0031	0.6406 ±0.0058	0.5348 ±0.0047
PrefRec (ours)	**0.4229 ±0.0077**	**1.7706 ±0.0206**	**1.3788 ±0.0133**
% Improv.	21.00%	55.20%	29.56%

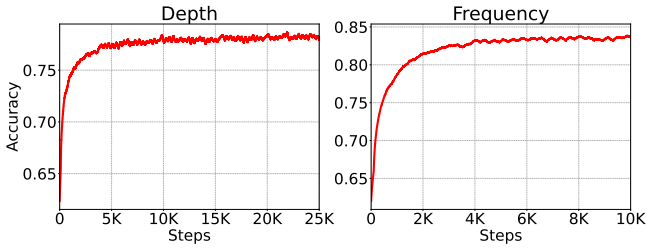

Figure 6: Prediction accuracy of the reward function.

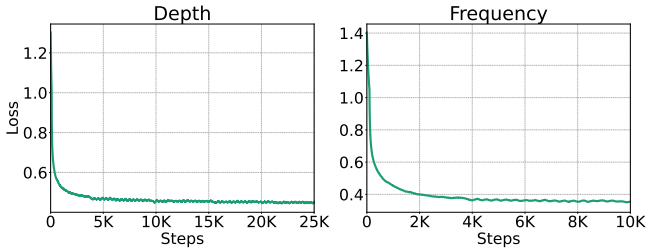

Figure 7: Learning loss of the reward function.

of the reward function in Fig. 7. The results indicate a substantial decrease in the loss, reducing it to a much lower level compared to its initial value. The improvement in prediction accuracy and the reduction in loss demonstrate that the reward function is becoming increasingly effective at accurately predicting the preferences used in the training phase. This is crucial for ensuring that the learning signals generated by the reward function are reliable and accurate, enabling the model to learn from the preferences effectively.

5.7 Ablations

To understand how the components in PrefRec affect the performance, we perform ablation studies on the expectile regression factor and the pre-training/fine-tuning of the reward function. As can be found in Table 8, when $\tau = 0.5$ where the expectile regression becomes identical to the regression to mean, there is an obvious drop in performance. The phenomenon indicates that the expectile

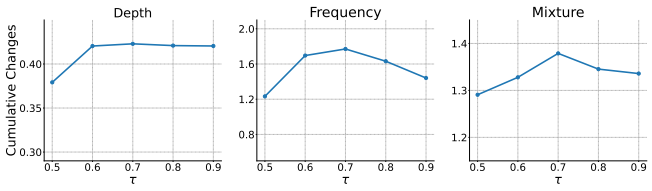

Figure 8: Ablations on the expectile regression factor τ.

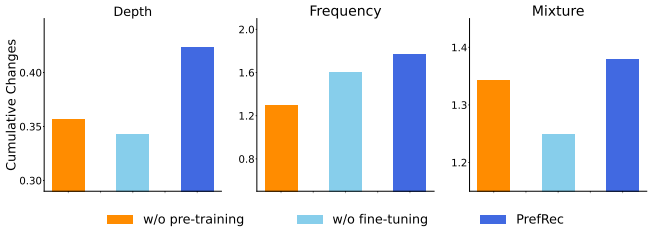

Figure 9: Ablations on reward function pre-training and reward function fine-tuning.

regression with a proper τ contributes to the improvement in performance. What's more, if compared with the results of DDPG in Table 2, we can find that introducing a separate V-function also benefits the performance since DDPG can be considered an algorithm whose $\tau = 0.5$ and without a V-function. Next, we study whether the pre-training and fine-tuning of the reward function is useful. As in Table 9, if we directly update the recommendation policy without pre-training the reward function, the performance decreases. Similarly, training the recommendation policy with a frozen reward function also degrades the performance. The results suggests that the pre-training and fine-tuning of the reward function are important to the performance.

6 RELATED WORK

6.1 Optimizing Long-term User Engagement in Recommendation

Long-term user engagement has recently attracted increasing attention due to its close connection to user stickiness. One of the major difficulties in promoting long-term user engagement is the lack of consistent and informative reward signals. Reward signals can be sparse and change over time, making it difficult to accurately predict and respond to user needs. Zhang et al. [47] augments the sparse rewards by re-weighting the original feedbacks with counterfactual importance sampling. Wu et al. [42] consider user click as immediate reward and user's return time as a hint on long-term engagement. It also assumes a stationary distribution of candidates for recommendation. In contrast, Zhao et al. [49] empirically identify the problem of non-stationary users' taste distribution and propose distribution-aware recommendation methods. Other researches focus on maximizing the diversity of recommendations as an indirect approach to optimize long-term engagement. For example, Adomavicius et al. [1] propose a graph-based approach to maximize diversities based on maximum flow. Ashton et al. [2] propose that high recommendation diversities is related to long-term user engagement. Apart from diversities, Cai et al. [5] conduct large-scale empirical studies and propose several surrogate criteria for optimizing long-term user engagements, including high-quality consumption, repeated consumption, etc. In this paper we avoid directly learning from the sparse and non-stationary signals inferred from the environment. Instead, we propose to learn a parameterized reward function to optimize the long-term user engagement.

6.2 Reinforcement Learning for Recommender Systems

Reinforcement learning allows an autonomous agent to interact with environment and optimizes long-term goals from experiences, which is particularly suitable for tasks in recommender systems [3, 9, 27, 31, 46, 53]. Shani et al. [31] first proposed to employ Markov Decision Process (MDP) to model the recommendation behavior. The subsequent researches focus on standard RL problems, e.g., exploration and exploitation trade-off in the context of recommendation systems [8, 38], together with model-based or simulator-based approaches to improve sample efficiency [3, 32]. Recently, there has been works on designing proper reward functions for efficient training of recommender systems. For example, Zou et al. [53] use a Q-network with hierachical LSTM to model both the instant and long-term reward. Ji et al. [21] model the recommendation process as a Partially Observable MDP and estimate the lifetime values of the recommended items. Zheng et al. [51] explicitly model the future returns by introducing the user activeness score. Ie et al. [20] decompose the Q-function for tractable and more efficient optimization. There are also researches focusing on behavior diversity [41, 52] as a surrogate for the reward function. Instead of relying on the aforementioned handcrafted reward signals, in this paper we propose to automatically train a reward function based on preferences between users' behavioral trajectories, which avoids the difficulties in reward engineering.

7 CONCLUSIONS

In this paper, we propose PrefRec, a novel paradigm of recommender systems, to improve long-term user engagement. PrefRec allows RL recommender systems to learn from preferences between users' historical behaviors rather than explicitly defined rewards. By this way, we can fully exploit the advantages of RL in optimizing long-term goals, while avoiding complex reward engineering. PrefRec uses the preferences to automatically learn a reward function. Then the reward function is applied to generate reinforcement signals for training the recommendation policy. We design an effective optimization method for PrefRec, which utilizes an additional value function, expectile regress and reward function pre-training to enhance the performance. Experiments demonstrate that PrefRec significantly and consistently outperforms the current state-of-the-art on various of long-term user engagement optimization tasks.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. 2011. Maximizing aggregate recommendation diversity: A graph-theoretic approach. In *Proc. of the 1st International Workshop on Novelty and Diversity in Recommender Systems*. 3–10.

[2] Ashton Anderson, Lucas Maystre, Ian Anderson, Rishabh Mehrotra, and Mounia Lalmas. 2020. Algorithmic effects on the diversity of consumption on spotify. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, 2155–2165.

[3] Xueying Bai, Jian Guan, and Hongning Wang. 2019. A model-based reinforcement learning with adversarial training for online recommendation. In *NeurIPS*. 10734–10745.

[4] Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* 39 (1952), 324–345.

[5] Qingpeng Cai, Shuchang Liu, Xueliang Wang, Tianyou Zuo, Wentao Xie, Bin Yang, Dong Zheng, Peng Jiang, and Kun Gai. 2023. Reinforcing User Retention in a Billion Scale Short Video Recommender System. In *Companion Proceedings of the ACM Web Conference 2023*. 421–426.

[6] Qingpeng Cai, Zhenghai Xue, Chi Zhang, Wanqi Xue, Shuchang Liu, Ruohan Zhan, Xueliang Wang, Tianyou Zuo, Wentao Xie, Dong Zheng, Peng Jiang, and Kun Gai. 2023. Two-Stage Constrained Actor-Critic for Short Video Recommendation. In *Proceedings of the ACM Web Conference 2023*. 865–875.

[7] Minmin Chen, Bo Chang, Can Xu, and Ed H Chi. 2021. User response models to improve a reinforce recommender system. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 121–129.

[8] 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.

[9] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1187–1196.

[10] Konstantina Christakopoulou, Can Xu, Sai Zhang, Sriraj Badam, Trevor Potter, Daniel Li, Hao Wan, Xinyang Yi, Ya Le, Chris Berg, et al. 2022. Reward shaping for user satisfaction in a REINFORCE recommender. *arXiv preprint arXiv:2209.15166* (2022).

[11] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2021. First return, then explore. *Nature* 590, 7847 (2021), 580–586.

[12] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. 2018. More robust doubly robust off-policy evaluation. In *ICML*. 1447–1456.

[13] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. 2022. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* 610, 7930 (2022), 47–53.

[14] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *NeurIPS* 34 (2021).

[15] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *ICML*. 1587–1596.

[16] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *ICML*. 2052–2062.

[17] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline a/b testing for recommender systems. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 198–206.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[19] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

[20] 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*, Sarit Kraus (Ed.). 2592–2599.

[21] Luo Ji, Qi Qin, Bingqing Han, and Hongxia Yang. 2021. Reinforcement learning to optimize lifetime value in cold-start recommendation. In *CIKM*. ACM, 782–791.

[22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[23] Roger Koenker and Kevin F Hallock. 2001. Quantile regression. *Journal of economic perspectives* 15, 4 (2001), 143–156.

[24] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2022. Offline reinforcement learning with in-sample Q-Learning. In *ICLR*.

[25] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[26] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. *ICLR* (2016).

[27] Bogdan Mazoure, Paul Mineiro, Pavithra Srinath, Reza Sharifi Sedeh, Doina Precup, and Adith Swaminathan. 2021. Improving long-term metrics in recommendation systems using short-horizon offline RL. *arXiv preprint arXiv:2106.00589* (2021).

[28] Andrew Y. Ng and Stuart J. Russell. 2000. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*. 663–670.

[29] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155* (2022).

[30] Jongjin Park, Younggyo Seo, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. 2022. SURF: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning. *ICLR*.

[31] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *J. Mach. Learn. Res.* 6 (2005), 1265–1295.

[32] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and Anxiang Zeng. 2019. Virtual-Taobao: Virtualizing real-world online retail environment for Reinforcement Learning. In *AAAI*. 4902–4909.

[33] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[34] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (2017), 354–359.

[35] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.

[36] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

[37] Adith Swaminathan and Thorsten Joachims. 2015. The self-normalized estimator for counterfactual learning. *NeurIPS* 28 (2015).

[38] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2017. Exploration: A study of count-based exploration for deep reinforcement learning. In *NeurIPS*. 2753–2762.

[39] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[40] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: Challenges, progress and prospects. In *IJCAI*. 6332–6338.

[41] Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed H. Chi, and Minmin Chen. 2022. Surrogate for long-term user experience in recommender systems. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 4100–4109.

[42] Qingyun Wu, Hongning Wang, Liangjie Hong, and Yue Shi. 2017. Returning is believing: Optimizing long-term user engagement in recommender systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1927–1936.

[43] Wanqi Xue, Bo An, Shuicheng Yan, and Zhongwen Xu. 2023. Reinforcement Learning from Diverse Human Preferences. arXiv:2301.11774

[44] 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 *The Eleventh International Conference on Learning Representations*.

[45] Ruohan Zhan, Changhua Pei, Qiang Su, Jianfeng Wen, Xueliang Wang, Guanyu Mu, Dong Zheng, Peng Jiang, and Kun Gai. 2022. Deconfounding duration bias in watch-time prediction for video recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4472–4481.

[46] Qihua Zhang, Junning Liu, Yuzhuo Dai, Yiyan Qi, Yifan Yuan, Kunlun Zheng, Fan Huang, and Xianfeng Tan. 2022. Multi-Task Fusion via Reinforcement Learning for Long-Term User Satisfaction in Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4510–4520.

[47] 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 *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 41–50.

[48] 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 *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 871–880.

[49] Xing Zhao, Ziwei Zhu, and James Caverlee. 2021. Rabbit Holes and Taste Distortion: Distribution-Aware Recommendation with Evolving Interests. In *WWW '21: The Web Conference 2021*. 888–899.

[50] Yifei Zhao, Yu-Hang Zhou, Mingdong Ou, Huan Xu, and Nan Li. 2020. Maximizing cumulative user engagement in sequential recommendation: An online optimization perspective. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2784–2792.

[51] 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 *WWW*. 167–176.

[52] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. 2010. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences* 107, 10 (2010), 4511–4515.

[53] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.

[54] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. 2020. Model Based Reinforcement Learning for Atari. In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1xCPJHtDB