MTRec: Learning to Align with User Preferences via Mental Reward Models

Mengchen Zhao[1] **Yifan Gao**[2] **Yaqing Hou**[2*] **Xiangyang Li**[3] **Pengjie Gu**[4]
Zhenhua Dong[3] **Ruiming Tang**[3] **Yi Cai**[1]
[1]School of Software Engineering, South China University of Technology
[2]School of Computer Science and Technology, Dalian University of Technology
[3]Huawei Noah's Ark Lab
[4]Nanyang Technological University
`{zzmc, ycai}@scut.edu.cn`
`{otz, houyq}@mail.dlut.edu.cn`
`{lixiangyang34, dongzhenhua, tangruiming}@huawei.com`

Abstract

Recommendation models are predominantly trained using implicit user feedback, since explicit feedback is often costly to obtain. However, implicit feedback, such as clicks, does not always reflect users' real preferences. For example, a user might click on a news article because of its attractive headline, but end up feeling uncomfortable after reading the content. In the absence of explicit feedback, such erroneous implicit signals may severely mislead recommender systems. In this paper, we propose MTRec, a novel sequential recommendation framework designed to align with real user preferences by uncovering their internal satisfaction on recommended items. Specifically, we introduce a mental reward model to quantify user satisfaction and propose a distributional inverse reinforcement learning approach to learn it. The learned mental reward model is then used to guide recommendation models to better align with users' real preferences. Our experiments show that MTRec brings significant improvements to a variety of recommendation models. We also deploy MTRec on an industrial short video platform and observe a 7% increase in average user viewing time.

1 Introduction

In interactive recommender systems, explicit feedback (e.g., ratings) is inherently sparse. Consequently, recommendation models predominantly rely on implicit signals (e.g., clicks) for training. However, such signals frequently fail to capture users' real preferences. For instance, clicking on a video may not indicate satisfaction with its content, while skipping a video could stem from prior exposure to similar content on other platforms rather than genuine dislike. These observations highlight a fundamental misalignment between recommendation models and users' real preferences.

To mitigate such misalignment caused by erroneous feedback signals, a natural approach is to incentivize users to provide explicit feedback. However, in real-world scenarios, users exhibit low propensity to offer such feedback due to cognitive burdens and interface constraints. Prior studies treat erroneous feedback signals as noisy labels and applying denoising techniques to address them, yet their effectiveness remains limited because erroneous feedback is not random noise by its nature Wang et al. [2021]. Some alternative methods attempt to mitigate erroneous feedback via multi-feedback fusion, yet they often struggle when confronted with conflicting feedback Chen et al. [2021a]. Overall,

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

existing works focus on data mining approaches, lacking a deep understanding of the mismatch between users' implicit feedback and their real preferences.

In this work, we aim to quantify and uncover users' internal satisfaction with recommendations, thereby bridging the gap between the recommendation model and users' real preferences. In fact, each time user takes an action (e.g., consume an item), a private feeling will be generated in her mind, telling how she is satisfied by taking the action. We summarize such private feeling as *mental reward*. We have following two observations on the mental reward. O1: The mental reward will influence user's short-term interests and her subsequent behaviors. For example, if an user clicked on a news but felt uncomfortable with the content, she would lose interests on that topic and probably not click on similar news again. O2: Users are maximizing their accumulated mental rewards. This is reasonable because users naturally pursue good experiences during interaction with the recommender system. The above observations indicate that mental reward plays an important role in user's sequential decision making. If we can directly optimize user's mental rewards, the recommendation model would be better aligned with users' real preferences.

To this end, we propose MTRec, a novel sequential recommendation framework which uses a learned Mental Reward Model to guide the recommendation model to align with users' real preferences. First of all, we model the user's decision making as a Markov Decision Process (MDP). With the assumption that the user always maximizes her accumulated mental rewards, we use Inverse Reinforcement Learning (IRL) to infer a mental reward function from users' behavioral data. However, plain IRL recovers a deterministic mental reward function, which fails to capture the random nature of the mental rewards. To address this, we propose a Quantile Regression Inverse Q-Learning (QR-IQL) approach to learn a distributional mental reward function, which maps a state-action pair to a distribution of mental rewards. Hence, we use the rewards predicted by the mental reward model as complementary supervision signals to guide the training of recommendation model. In such a way, the misalignment between recommendation model and user's real preferences can be greatly reduced. Experiments on two public datasets show that MTRec significantly improves the performance of several popular recommendation models, in terms of Area Under Curve (AUC) and Normalised Capped Importance Sampling (NCIS). We also test MTRec in Virtual Taobao to demonstrate its effectiveness on reinforcement learning based recommendation models. Moreover, we deployed MTRec in a real-world industrial short video recommendation platform and observed a 7% increase in average user viewing time over a 7-day period during the online A/B test.

Our main contributions are summarized as follows.

- We identify the misalignment problem in sequential recommendation, where erroneous user feedback could severely deviate recommendation model from users' real preferences.
- We introduce MTRec, a novel sequential recommendation framework that aims to bridge the gap between the recommendation model and users' real preferences by a learned mental reward model, which uncovers users' internal satisfaction with recommendations.
- To capture the random nature of the mental rewards, we develop a distributional variant of IRL called QR-IQL to learn the mental reward model. We show how to use the learned mental reward model to guide the optimization of sequential recommendation models.
- We conducted extensive offline and online experiments to demonstrate the improvements brought by MTRec. Additionally, we deployed MTRec in a real-world industrial short video recommendation platform and observed a significant increase in user engagement.

2 Related Works

Implicit user feedback in recommendation. Since explicit feedback is very sparse, industrial recommender systems rely on implicit feedback (e.g., click, video watching) to train recommendation models Covington et al. [2016], Zhou et al. [2018], Dai et al. [2021], Yang et al. [2021]. However, these models ignore the fact that implicit feedback may not reflect users' real preferences. Some works treat such erroneous feedback as random noise and try to eliminate its influence via denoising, yet these methods have limited accuracy because it is hard to distinguish it from genuine responses Yu and Qin. [2020], Wang et al. [2021]. Another line of works address various types of biases in user feedback, including position bias, selection bias, popularity bias and exposure bias woong Lee et al. [2021], Yi et al. [2023]. However, errors in implicit user feedback differ from the aforementioned biases. In some sense, such errors can be regarded as a certain type of inductive bias Jiawei et al. [2023], because implicit feedback is wrongly assumed to reflect users' real preferences.

AI alignment through reward model. Emerging from Natural Language Processing, alignment algorithms have proven effective due to their ability to guide Large Language Models (LLMs) in matching human values. Many popular alignment methods employ a reward model to provide fine-tuning signals. For example, Reinforcement Learning from Human Feedback (RLHF) learns human preferences through a reward model trained with human-rated outputs Ouyang et al. [2022]. Reward rAnked FineTuning (RAFT) uses a reward model to select the best set of training samples based on model outputs Dong et al. [2023]. Inspired by these works, we aim to learn a reward model to guide recommendation models in aligning with users' real preferences. Unlike prior approaches that rely on human annotators to provide reward model labels, our reward model is directly learned from existing user behavioral data, making it more industry-friendly.

RL and IRL for sequential recommendation. Sequential recommendation is typically modeled as interactions between users and recommender systems. Prior works have explored using reinforcement learning (RL) to optimize recommendation policy, where the recommender system is modeled as an agent and the users are treated as key components of environment Chen et al. [2019], Zheng et al. [2018]. RL-based methods have great potential to maximize recommender systems' long-term revenue, but they often suffer from the bias of user simulators. Inverse RL (IRL) aims to recover the agent's reward function from expert trajectories. Some works apply IRL to infer the reward function for the recommender system, assuming expert recommendation policies are available Chen et al. [2021b], Liu et al. [2023a]. However, these works ignore that the users are also active agents, and understanding user behaviors is essential for improving recommendation policies. In our work, we model users as agents and use IRL to infer the optimal reward model from their behaviors. Although similarly employing IRL techniques, our work differs fundamentally from existing works by inferring a user-centric reward model, as opposed to system-centric reward modeling.

3 Preliminaries

Sequential recommendation. Typically, a sequential recommendation model takes a sequence of user-item interactions as input and predicts the next items that mostly attract the users. Due to the lack of explicit feedback, sequential recommendation tasks are usually formulated as predicting the next item that is most likely to induce target user behaviors such as clicks:

$$i_t = \arg\max_{i \in I} p_\phi(\hat{a}|i, h_{t-1}), \tag{1}$$

where $\hat{a}$ represents user's behavior to be predicted, $I = \{i_1, ..., i_N\}$ is the set of candidate items. The parameterized function p_ϕ measures the probability of the target behavior $\hat{a}$. The interaction history h_t consists of a list of tuples up to time step t, where each tuple $< u, i, a >_t$ consists of a user u, an item i and the user's action $a \in A$. In a simplified case, user's action space A can be restricted to $\{click, skip\}$, hence the recommendation task is reduced to predicting the user's click-trough rate.

Inverse reinforcement learning. Conventionally, maximum entropy IRL aims to learn a deterministic reward function $r : S \times A \to \mathbb{R}$ by solving Problem 2 Brian et al. [2008].

$$\max_{r \in \mathcal{R}} \min_{\pi \in \Pi} \mathbb{E}_{\pi_E}[r(s, a)] - \mathbb{E}_\pi[r(s, a)] - \mathcal{H}(\pi), \tag{2}$$

where $\mathcal{H}(\pi) \triangleq \mathbb{E}_\pi[-\log \pi(a|s)]$ denotes the entropy of policy π. Intuitively, this formulation learns a reward function that assigns high reward to the expert policy π_E and a low reward to other policies, while searching for the optimal policy under the reward function in the inner loop. The expectation over policies can be replaced by the occupancy measure $\rho_\pi(s, a) = \pi(a|s) \sum_t \gamma^t P(s_t = s|\pi)$, which specifies a probability distribution over (s, a) pairs. Then, Problem 2 can rewritten as:

$$\max_{r \in \mathcal{R}} \min_{\pi \in \Pi} \mathbb{E}_{\rho_E}[r(s, a)] - \mathbb{E}_\rho[r(s, a)] - \mathcal{H}(\pi) - \phi(r), \tag{3}$$

where ϕ is a convex regularizer on r Jonathan and Ermon [2016]. If we define the regularizer as $\psi(x) = x - \phi(x)$, then $\mathbb{E}_{\rho_E}[r(s,a)] - \phi(r)$ can be compactly represented by $\mathbb{E}_{\rho_E}[\psi(r(s,a))]$. Moreover, the reward function r can be represented by the soft Q-function as $r(s,a) = Q(s,a) - \gamma\mathbb{E}_{s'\sim P(s,a)}V(s')$, where $V(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q(s,a) - \log\pi(a|s)]$. Also the optimal policy π^* can be represented by the soft Q-function as $\pi^*(a|s) = \frac{1}{\Delta(s)}\exp(Q(s,a))$, where $\Delta(s) = \sum_{a'}\exp(Q(s,a'))$ is the normalization factor. Then, the inner problem becomes trivial so that Problem 3 can be reformulated as follows Brian et al. [2008], Haarnoja et al. [2018].

$$\max_{Q\in\Omega}\mathbb{E}_{s,a\sim\rho_E}[\psi(Q(s,a) - \gamma\mathbb{E}_{s'\sim P(s,a)}V^*(s'))] - (1-\gamma)\mathbb{E}_{s_0\sim\rho_0}[V^*(s_0)], \tag{4}$$

where $V^*(s) = \log\sum_a \exp Q(s,a)$. Note that the objective of Problem 4 depends only on Q, which allows us to solve the problem by directly optimizing a Q-network. Solving Problem 4 with the expert data D_E results in the optimal Q-function, and the deterministic rewards can be recovered by:

$$r(s,a) = Q(s,a) - \gamma\mathbb{E}_{\rho_E}[\log\sum_a \exp Q(s',a)], \tag{5}$$

The misalignment problem in recommendation. In the context of LLM, the alignment problem is defined as optimizing the outputs of LLMs towards matching human values Ouyang et al. [2022]. For recommender systems, the general goal is to maximize users' satisfaction by selecting appropriate items for them. This is naturally in accord with the alignment problem in LLMs, in the sense that aligning with human values is similar to aligning with user preferences. However, directly maximizing user satisfaction is very challenging for recommender systems. Therefore, most existing works focus on optimizing some surrogate objectives such as click-through rate and conversion rate Wang et al. [2017], Zhou et al. [2019]. Although the surrogate objectives make it convenient to optimize recommendation models, they suffer from inductive biases since users' implicit feedback may not reflect their real preferences. As a consequence, *the optimization goal of recommendation models may deviate from users' real preferences.* Compared with one-shot recommendation, the misalignment problem in sequential recommendation is even worse due to the accumulation of errors.

Inspired by recent advances in LLM alignment, we aim to develop a reward model that helps to align recommendation models with users' real preferences. The challenges are two-fold. First, we do not have an off-the-shelf reward model that reflects users' real preferences, under the lack of their explicit feedback. To address this, we propose the Mental Reward model learned from rich user behavioral data to approximate their real preferences. Second, unlike static human values, users' preferences could be highly stochastic, potentially due to unpredictable environmental factors and preference shifts. To resolve this, we propose a distributional Inverse RL approach to capture the randomness of the mental reward model, which will be illustrated in Section 4.

4 Method

Motivation. User behaviors have been extensively studied in the literature of recommender systems. However, existing works treat user behaviors as labels for recommendation models. In fact, during the interaction with recommender systems, users are active agents rather than static label providers. Moreover, users are implicitly maximizing their own reward functions by taking actions in recommender systems. Based on the above insights, we believe that uncovering the user's reward function would significantly benefit recommendation models in aligning with users' real preferences.

Overview. We propose a MenTal reward based Recommendation framework MTRec, which consists of three main parts. First, we introduce a novel User-Centric Markov Decision Process, where uses are modeled as active agents during their interaction with recommender systems. Second, we develop a distributional IRL method called Quantile Regression Inverse Q-learning (QR-IQL) for learning the mental reward model. Last, we show how to use the learned mental reward model to guide existing recommendation models to align with users' real preferences.

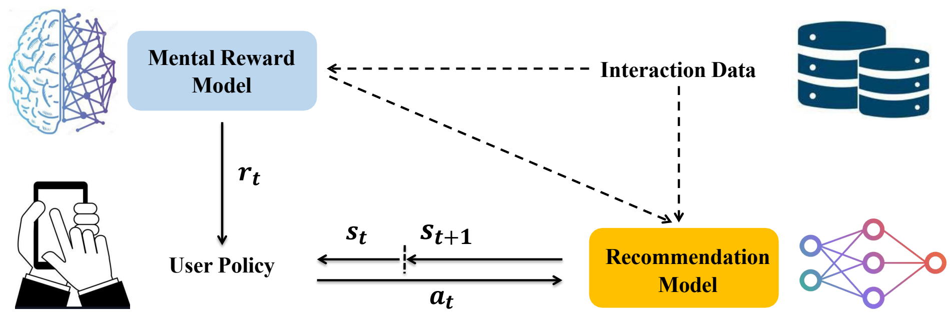

Figure 1: The overall framework of MTRec. The solid lines represent the interaction process. The dashed lines represent the information flow between data and models. Our goal is to recover the mental reward model and use it to improve the recommendation model.

4.1 User-Centric Markov Decision Process

Reinforcement learning has been applied to model the interaction between users and recommender systems. Existing works usually model the recommender system as agent, who recommends items and receives user feedback as reward Zheng et al. [2018], Chen et al. [2019]. However, they often ignore the strategic behavior of the users. In this work, we focus on studying users' behaviors, whose decision making can be modeled as a Markov Decision Process $\mathcal{M} = \langle S, A, P, R, \pi \rangle$:

- S denotes the state space. A state $s_t = (h_t, i_t)$ includes the interaction history up to time t. At each time step, the interaction is recorded by a tuple $< u, i, a >_t$, consisting of an user u, displayed item i and the user's action a.
- A denotes user's action space. An action $a \in \mathcal{A}$ represents user's response to the item displayed to her. User's responses could be of various types, including explicit feedback such as news clicks and video watchings. In this work, we consider a general case where users' responses are either positive or negative, leading to a simplified action space.
- $P : S \times A \to S$ denotes the transition function. Following existing works on sequential recommendation Zhou et al. [2018, 2019], Wang-Cheng and McAuley [2018], after an user takes an action, a new state $s_{t+1} = s_t \cup \{h_{t+1}, i_{t+1}\}$ will be generated.
- $R : S \times A \to \theta(r)$ denotes the mental reward function which maps a state and an action to a distribution of mental reward $r \in \mathbb{R}$. Note that we model the mental reward r as a random variable instead of a deterministic value, since r could be influenced by unpredictable environmental factors and preference shifts. Such a modeling allows us to capture the intrinsic randomness and potentially richer information on user's mental rewards.
- $\pi : \mathcal{S} \to \mu(a)$ denotes the user's behavioral policy, which maps a state to a distribution over actions. A stochastic user policy facilitates reasoning her reward function using IRL.

Figure 1 illustrates the interaction process between user and recommendation model. At each round, the user receives a recommended item and select an action to respond. After that, a mental reward that summarizes user's satisfaction about the current item is generated. Note that the mental reward is highly correlated with user's real preference, but is unknown to the recommendation model. However, the mental reward will influence the user's subsequent behaviors as she seeks to maximize the accumulated mental rewards. We will focus on estimating the mental rewards in the following parts.

4.2 Uncovering the Mental Reward Model

Inverse reinforcement learning (IRL) aims to find a reward model that explains behaviors of the expert policy Saurabh and Doshi [2021]. Since the user naturally maximizes her mental rewards, her policy can be regarded as the expert policy. Therefore, IRL could be used to recover the mental reward model using the interaction data. *However, existing IRL methods focus on deterministic reward models, which fail to capture the intrinsic randomness of the user's mental rewards.* To this end, we will develop a distributional version of IRL algorithm for uncovering the mental reward model.

4.2.1 A distributional perspective on IRL

Conventional IRL methods such as MaxEntIRL iteratively optimize the reward function and the policy until convergence Brian et al. [2008]. Bayesian IRL methods assume that there are multiple reward functions and focus on estimating their posterior distribution using Bayes' rule Deepak and Amir [2007], Jaedeug and Kim [2012]. Although Bayesian IRL methods introduce various distributions over reward functions, they are still restricted to deterministic reward functions. By contrast, we aim to learn a distributional reward function where the reward can be stochastic.

Since we model the user's mental reward r as a random variable, Equation 5 can be rephrased using a distributional operator as $\mathcal{T}^\pi r(s,a) :\overset{D}{=} Q(s,a) - \gamma \mathbb{E}_{\rho_E}[\log \sum_a \exp Q(s',a)]$, where $X :\overset{D}{=} U$ denotes equality of probability laws, that is, the random variable X is distributed according to the same law as U. As each $Q(s,a)$ function uniquely determines a distribution of $r(s,a)$, learning a distributional reward function is reduced to learning a distributional Q-function.

4.2.2 Quantile Regression Inverse Q-learning (QR-IQL)

Problem 4 aims to learn a deterministic Q-function, while we aim to learn a distributional Q-function. Following the QR-DQN Will et al. [2018], the distribution of Q can be characterized by a quantile distribution. We denote by Z the variable associated with the distribution of Q, that is, $Q(s,a) = \mathbb{E}[Z(s,a)]$. Let $\lambda : S \times A \to \mathbb{R}^N$ be a parametric model, where N is the number of quantiles. Then a quantile distribution Z_λ maps a state-action pair (s,a) to a uniformly probability distribution supported on $\{\lambda_i(s,a)\}_{i=1}^N$. Instead of learning a scalar value $Q(s,a)$, our model will estimate the positions of supports $\{\lambda_i(s,a)\}_{i=1}^N$ and calculate $Q_\lambda(s,a) = \frac{1}{N}\sum_{i=1}^N \lambda_i(s,a)$.

Recall that in Problem 4, we optimize a Q-network that outputs a scalar Q-value. In order to learn the distributional Q-function, we change the output layer of the Q-network to be of size $|A| \times N$. To derive the objective for the quantile regression inverse Q-learning, we made two modifications on Problem 4. First, since our mental reward model is learned from offline data, the second term in Problem 4 (i.e., $(1-\gamma)\mathbb{E}_{s_0\sim\rho_0}[V(s_0)]$) can be replaced by $\mathbb{E}_{(s,a)\sim\rho_E}[V(s) - \gamma V(s')]$. In other words, $\mathbb{E}_{s_0\sim\rho_0}[V(s_0)]$ is irrelevant with the initial state distribution ρ_0. Second, as is suggested in Garg et al. [2021], we choose the regularizer as $\psi(x) = x - \frac{1}{4\alpha}x^2$ for the ease of optimization while bounding the rewards. Finally, we formulate the objective of the quantile regression inverse Q-learning as Problem 6. The complete derivation is provided in Appendix A.1.

$$\max_{Q_\lambda} \mathbb{E}_{\rho_E}[Q_\lambda(s,a) - \log \sum_a \exp Q_\lambda(s,a)] - \frac{1}{4\alpha}\mathbb{E}_{\rho_E}[(Q_\lambda(s,a) - \log \sum_a \exp Q_\lambda(s',a))^2], \quad (6)$$

We apply the Pinball loss to get the optimal quantile distribution supports $\{\lambda_i(s,a)\}_{i=1}^N$ and the optimal Q_λ^*, which we can use to calculate the mental rewards as:

$$r^*(s,a) = Q_\lambda^*(s,a) - \gamma \mathbb{E}_{\rho_E}[\log \sum_a \exp Q_\lambda^*(s',a)]. \quad (7)$$

Detailed optimization algorithm is provided in Appendix A.2. While building upon ideas from QR-DQN for RL, our QR-IQL is the first distributional IRL algorithm that uncovers the underlying distribution of rewards, which is key to capture the randomness of users' mental rewards.

4.3 Applications of the Mental Reward model

Generally, a sequential recommendation model takes user features, candidate item features, interaction histories and some contextual features as input, and output a score for the candidate item used for ranking. For the sake of brevity, we represent the recommendation model as $F_\zeta(i_t|h_t)$, which is parameterized by ζ. Although the mental reward model $r(s,a)$ indicates users' preferences to some extent, it lacks sufficient feature-level modeling and thus cannot be directly used for recommendation. Instead, we use $r(s,a)$ to provide additional learning signals for recommendation models, However, combining $r(s,a)$ with existing recommendation models is non-trivial due to the variety of learning objectives. Fortunately, these objectives fall into several categories. We will use the following two typical examples to illustrate how to use the mental reward model in practice.

Classification-based models. Many sequential recommendation tasks are formulated as binary classification problems with the following Cross Entropy loss:

$$\mathcal{L}_{CE}(\zeta) = -\mathbb{E}_{D_E}[a^P \log(F_\zeta(i|h)) + a^N \log(1 - F_\zeta(i|h))], \tag{8}$$

where $a^P = 1$ and $a^N = 0$ indicate user's positive and negative responses, and $F_\zeta(i|h)$ represents the estimated probability of clicking on item i based on history h. We want the recommendation models to also maximize the expectation of user's mental rewards, leading to the following alignment loss:

$$\mathcal{L}_{Align}(\zeta) = -\mathbb{E}_{D_E}[r^*(s, a^P) \cdot F_\zeta(i|h) + r^*(s, a^N) \cdot (1 - F_\zeta(i|h))], \tag{9}$$

Then, the final loss for training the recommendation model can be written as a weighted combination of the two losses:

$$\mathcal{L}_{Final}(\zeta) = \mathcal{L}_{CE}(\zeta) + \kappa \cdot \mathcal{L}_{Align}(\zeta). \tag{10}$$

RL-based models. In a typical setting, the recommender system is modeled as an agent, who maximizes the accumulated system rewards. We denote by $\hat{r}(h, i, a)$ the system reward after recommending item i and receiving user feedback a given interaction history h. In this context, $F_\zeta(i|h)$ represents the RL-based recommendation policy, whose goal is to maximize the expectation of accumulated rewards $\hat{r}$:

$$\mathcal{L}_{RL}(\zeta) = -\mathbb{E}_{i_t \sim F_\zeta}[\sum_t \hat{r}(h_{t-1}, i_t, a_t)] \tag{11}$$

Since we want the recommendation model to also maximize the mental rewards of the user. We simply add the mental reward r^* to $\hat{r}$ and obtain the following objective.

$$\mathcal{L}_{Final}(\zeta) = -\mathbb{E}_{i_t \sim F_\zeta}[\sum_t \hat{r}(h_{t-1}, i_t, a_t) + \kappa \cdot r^*(h_{t-1}, i_t, a_t)] \tag{12}$$

See Appendix A.3 for more implementation details.

5 Experiments

In this section, we report the performance of MTRec in both offline and online settings, with focuses on answering the following research questions (RQs).

- (RQ1:) How does MTRec improve classification-based recommendation models?
- (RQ2:) How does MTRec improve RL-based recommendation models?
- (RQ3:) Does the learned mental reward model provide useful information?
- (RQ4:) How does MTRec perform in online A/B test?

5.1 Experiments on Public Datasets (RQ1)

Datasets. The Amazon dataset McAuley et al. [2015] collects user review data from Amazon e-commerce platform. We use two subsets of the Amazon dataset: Books and Electronics in our offline experiments. More details on processing the datasets are provided in Appendix A.4.

Baselines. We use eight widely used recommendation models as baselines and combine each of them with MTRec to test the improvements brought by MTRec. **Wide&Deep** Cheng et al. [2016] is a hybrid recommendation model combining a wide linear model and deep neural network for collaborative filtering. **PNN** Qu et al. [2016] is a neural network architecture designed for CTR prediction in recommender systems. **DeepFM** Guo et al. [2017] is a hybrid recommendation model combining factorization machines and deep neural networks. **SASRec** Wang-Cheng and McAuley [2018] uses the self-attention mechanism to model sequential patterns for recommendation systems. **DIN** Zhou et al. [2018] is an attention-based neural model for sequential recommendation, where the attention mechanism aims to distinguish the interest of a user's historical behaviors. **DIEN** Zhou et al. [2019] designs a sequential architecture to model interest evolution for recommendation, which uses an auxiliary loss to capture temporal interests. **LinRec** Liu et al. [2023b] is a lightweight linear recommendation model designed for efficient computation and scalability with large datasets. **SIGMA** Liu et al. [2024] is a sequential recommendation model that uses a selective gating mechanism to focus on the most relevant user behaviors for improved performance.

Evaluation metrics. We use the following two metrics: Area Under Curve (AUC) Fawcett [2006] and Normalised Capped Importance Sampling (NCIS) Swaminathan and Joachims [2015]. AUC is used to measure the model's ranking ability and NCIS is used to approximate the model's online performance Gilotte et al. [2018]. Formally, the score of NCIS can be calculated by: $\tilde{J}^{NCIS}(\mathcal{M}) = \frac{\sum_{i=1}^{n} \tilde{\rho}_i(\mathcal{M}) * L_i}{\sum_{i=1}^{n} \tilde{\rho}_i(\mathcal{M})}$, where $\rho_i(\mathcal{M}) = \prod_{t \in \mathcal{T}} p_t(\mathcal{M})$ is the probability that the CTR model $\mathcal{M}$ follows the request trajectory of the user i, $p_t(\mathcal{M})$ is the click-through rate estimate of model $\mathcal{M}$ for item t and n is the number of users in the test set for NCIS. In the experiments, we use the complete trajectories of 10% of users to calculate NCIS. Moreover, we obtain the final NCIS score by substracting the NCIS of the untrained model from that of the trained model to eliminate the impact of random parameters among different models. Intuitively, $\tilde{J}^{NCIS}(\mathcal{M})$ awards a CTR model with a high score if the model has large probability to follow long trajectories.

Results on Amazon datasets. Table 1 summarizes the experimental results on the two Amazon datasets. The suffix "IRL" indicates that the mental reward model is learned by a non-distributional IRL algorithm IQ-Learn Garg et al. [2021], while the suffix "MTRec" indicates that the mental reward model is learned by our algorithm QR-IQL. It can be observed that integrating MTRec with existing models consistently improves their AUC and NCIS across almost all baseline models. Moreover, the improvements on NCIS are generally more significant than AUC. This aligns with the motivation of MTRec, that is, to maximize overall user satisfaction and long-term engagement. Note that the AUC is also slightly improved, demonstrating that the improvement of users' long-term engagement is not at the sacrifice of the model's ranking ability. In addition, the comparisons between IRL and MTRec demonstrates the benefit of learning a distributional version of mental reward model.

Model	Electronics		Books	
	AUC	NCIS	AUC	NCIS
Wide&Deep	0.8290	0.6749	0.8605	2.0967
Wide&Deep-IRL	0.8342	0.8738	**0.8661**	3.1264
Wide&Deep-MTRec	**0.8351**	**0.9063**	0.8657	**3.2043**
PNN	0.8396	0.6618	0.8603	2.3316
PNN-IRL	**0.8547**	0.9297	0.8673	3.4567
PNN-MTRec	0.8542	**0.9515**	**0.8679**	**3.4971**
DeepFM	0.8424	0.6993	0.8634	2.3146
DeepFM-IRL	0.8458	0.8542	0.8715	3.7296
DeepFM-MTRec	**0.8468**	**0.8961**	**0.8742**	**3.7904**
SASRec	0.8325	0.8243	0.8675	3.1755
SASRec-IRL	**0.8366**	0.8681	0.8681	3.1946
SASRec-MTRec	0.8328	**0.8833**	**0.8798**	**3.4634**
DIN	0.8523	0.6044	0.8653	2.1324
DIN-IRL	0.8533	0.8168	0.8701	3.1675
DIN-MTRec	**0.8542**	**0.8728**	**0.8732**	**3.2208**
DIEN	0.8448	0.7766	0.8686	2.2685
DIEN-IRL	0.8461	0.8857	0.8723	3.0476
DIEN-MTRec	**0.8472**	**0.9324**	**0.8757**	**3.1185**
LinRec	0.8579	0.8077	0.8754	2.4653
LinRec-IRL	**0.8597**	0.9365	0.8771	**3.8005**
LinRec-MTRec	0.8594	**0.9782**	**0.8792**	3.7928
SIGMA	0.8581	0.7946	0.8762	2.3975
SIGMA-IRL	0.8592	0.9261	0.8802	3.7556
SIGMA-MTRec	**0.8604**	**0.9563**	**0.8814**	**3.8025**

Table 1: Experimental results on Amazon datasets.

5.2 Experiments on Virtual Taobao (RQ2)

In this set of experiments, we choose RL-based recommendation models as our baselines and combine them with MTRec to test their performance. Both training and testing of the algorithms are conducted in simulated interactive recommendation environments on Virtual Taobao Shi et al. [2019]. We construct an expert dataset containing 100,000 high-quality trajectories by recording trajectories with averaged CTR>0.5. After training the mental reward model using the expert data, we add the predicted mental rewards to the original rewards and train the baselines again.

Baselines. Virtual Taobao allows training recommendation policies by RL. We use Proximal Policy Optimization (PPO) Schulman et al. [2017] and Soft Actor-Critic (SAC) Haarnoja et al. [2018] as baselines. According to Equation 12, RL-based recommendation models can be adjusted by simply adding the mental rewards to the base rewards $r_{final} = r_{env} + \kappa \cdot r_{mental}$. In Virtual Taobao, r_{env} is provided by a pre-trained user model. We set $\kappa = 0.2$ for trade-off between the two reward signals.

Evaluation metric. The major evaluation metric used in Virtual Taobao is the episodic Click-Through-Rate (eCTR) during the simulated online interaction. The eCTR is calculated as: eCTR $= \frac{r_{episode}}{10 * N_{step}}$, where 10 is the number of items recommended in a single page, $r_{episode}$ is the total number of clicks in an episode and N_{step} is the total number of steps.

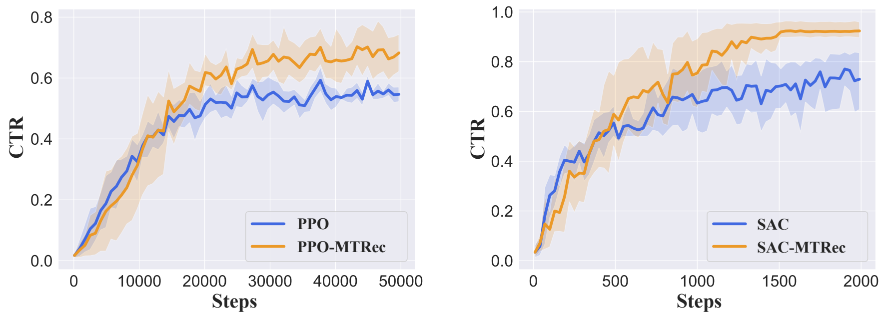

Figure 2: Training curves of RL models. Averaged CTR is reported with 95% confidence interval.

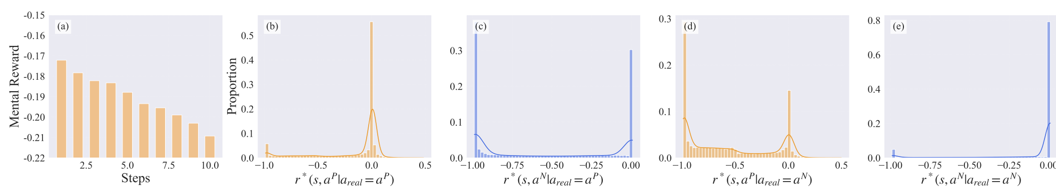

Figure 3: Illustrations of the predicted mental rewards. (a) Averaged mental rewards by steps in all trajectories; (b-e) Expected and counterfactual mental rewards given actual user actions.

Results on Virtual Taobao. From Fig 2 we can see that the baselines attain average CTRs of 0.5435 and 0.7055 respectively. By incorporating the mental rewards during training, PPO and SAC show significantly improved performance at average CTRs of 0.678 and 0.909 respectively. This demonstrates that the mental rewards provide more useful information about users' real preferences and successfully boost the performance of RL-based recommendation models.

5.3 Evaluation of the Mental Reward Model (RQ3)

It is almost impossible to evaluate the learned mental reward model directly due to the lack of ground-truth mental reward labels. Yet, we evaluate the mental reward model indirectly by its correlation with the datasets. First, we calculate the averaged mental rewards at different steps in all trajectories. Figure 3 (a) shows the results of the first ten steps on Amazon Book dataset. We can see that the averaged mental rewards decrease obviously with the increase of steps. This result aligns with our intuition that users may get tired and receive less mental rewards during their interaction with the recommender system. Second, we visualize the distribution of the mental rewards conditioned on the users' real responses. For instance, Figure 3 (b) and (c) show the distributions of expected mental reward $r^*(s, a^P | a_{real} = a^P)$ and the counterfactual mental reward $r^*(s, a^N | a_{real} = a^P)$ when the users take a positive action $a_{real} = a^P$. Specifically, $r^*(s, a^P | a_{real} = a^P)$ represents the predicted user's mental reward after she takes action a^P, while $r^*(s, a^N | a_{real} = a^P)$ represents the counterfactual mental reward if she had taken a negative action a^N. Intuitively, an user should receive a relatively high reward for the action she actually taken (a high $r^*(s, a_{real} | a_{real})$). Yet, the experimental results show that a non-negligible proportion of $r^*(s, a_{real} | a_{real})$ is relatively low, which suggests that there is a mismatch between users' actions and their real preferences (recall the example in the abstract: a user might click on a news article because of its attractive headline, but end up feeling uncomfortable after reading it). These observations indirectly validate the effectiveness of the learned mental reward model.

5.4 Online A/B test on Industrial Platform (RQ4)

To further validate the effectiveness of MTRec, we deploy it on an industrial short-video recommendation platform, which has tens of millions of Daily Active Users (DAU). Short-video recommendation is a typical sequential recommendation scenario, where the goal is to improve the user engagement with the platform. The baseline model is a Deep Cross Network (DCN) Wang et al. [2017], which is trained using binary labels indicating whether the users click on videos. However, intuitively, clicking on a video does not necessarily mean that the user is satisfied after watching the video. Therefore, we expect that MTRec could help to improve the overall users' satisfaction on the recommended videos and hence improve their engagement.

We improve the DCN model by incorporating the alignment loss, as illustrated in Equation 10. We find that the averaged video viewing time stably improve by about 7% compared with the baseline model, which demonstrates that MTRec indeed improves the overall recommendation quality and leads to better user engagement. Note that MTRec is quite industrial friendly because of two reasons: (1) the dataset used to train MTRec could be same with that used to train the recommendation model; (2) we only need to add an auxiliary alignment loss to the original recommendation loss.

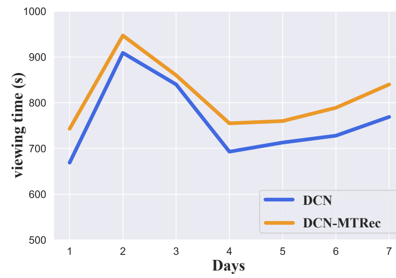

Figure 4: Online A/B test results.

6 Conclusions

The general goal of recommender systems is to satisfy users by providing items that align with their real preferences. However, existing works focus on optimizing surrogate objectives based on users' implicit feedback, ignoring that the implicit feedback may not accurately reflect their real preferences. Consequently, recommendation models could be systematically biased. In this work, we aim to fill this gap by studying the users' behaviors and uncovering the distribution of their mental rewards. We propose a novel distributional IRL algorithm to learn the mental reward model and use it to guide the training of recommendation models. Finally we validate the effectiveness of MTRec via both offline and online experiments, including the A/B on an industrial short video recommendation platform.

Acknowledgements

This research is supported by Guangdong Basic and Applied Basic Research Foundation (2025A1515010247), the Fundamental Research Funds for the Central Universities (2024ZYGXZR069).

References

Ziebart Brian, Andrew Maas, Andrew Bagnell, and Anind Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.

Hong Chen, Yudong Chen, Xin Wang, Ruobing Xie, Rui Wang, Feng Xia, and Wenwu Zhu. Curriculum disentangled recommendation with noisy multi-feedback. In *Advances in Neural Information Processing Systems*, pages 26924–26936, 2021a.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464, 2019.

Xiaocong Chen, Lina Yao, Aixin Sun, Xianzhi Wang, Xiwei Xu, and Liming Zhu. Generative inverse deep reinforcement learning for online recommendation. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, pages 201–210, 2021b.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

Paul Covington, Jay Adams, Emre Sargin, and Mengchen Zhao. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, page 191–198, 2016.

Xinyi Dai, Jianghao Lin, Weinan Zhang, Shuai Li, Weiwen Liu, Ruiming Tang, Xiuqiang He, Jianye Hao, Jun Wang, and Yong Yu. An adversarial imitation click model for information retrieval. In *Proceedings of the Web Conference*, pages 1809–1820, 2021.

Ramachandran Deepak and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2586–2591, 2007.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment, 2023.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. In *Advances in Neural Information Processing Systems*, pages 4028–4039, 2021.

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

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.

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

Choi Jaedeug and Kee-Eung Kim. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, pages 305–313, 2012.

Chen Jiawei, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, 41(3):1–39, 2023.

Ho Jonathan and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.

Diederik P. Kingma and Jimmy Ba. Deepfm: a factorization-machine based neural network for ctr prediction. In *International Conference on Learning Representations*, 2015.

Jialin Liu, Xinyan Su, Zeyu He, Xiangyu Zhao, and Jun Li. Adversarial batch inverse reinforcement learning: Learn to reward from imperfect demonstration for interactive recommendation. In *arXiv preprint arXiv:2310.19536*, 2023a.

Langming Liu, Liu Cai, Chi Zhang, Xiangyu Zhao, Jingtong Gao, Wanyu Wang, Yifu Lv, Wenqi Fan, Yiqi Wang, Ming He, et al. Linrec: Linear attention mechanism for long-term sequential recommender systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 289–299, 2023b.

Ziwei Liu, Qidong Liu, Yejing Wang, Wanyu Wang, Pengyue Jia, Maolin Wang, Zitao Liu, Yi Chang, and Xiangyu Zhao. Bidirectional gated mamba for sequential recommendation. *arXiv preprint arXiv:2408.11451*, 2024.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, pages 27730–27744, 2022.

Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1149–1154. IEEE, 2016.

Arora Saurabh and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297(103500), 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4902–4909, 2019.

Adith Swaminathan and Thorsten Joachims. The self-normalized estimator for counterfactual learning. *advances in neural information processing systems*, 28, 2015.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, 2017.

Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Denoising implicit feedback for recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 373–381, 2021.

Kang Wang-Cheng and Julian McAuley. Self-attentive sequential recommendation. In *Proceedings of the International Conference on Data Mining*, pages 197–206, 2018.

Dabney Will, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2018.

Jae woong Lee, Seongmin Park, Jongwuk Lee, and mengchen zhao. Dual unbiased recommender learning for implicit feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1647–1651, 2021.

Jiaqi Yang, Xiang Li, Shuguang Han, Tao Zhuang, Dechuan Zhan, Xiaoyi Zeng, and Bin Tong. Capturing delayed feedback in conversion rate prediction via elapsed-time sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4582–4589, 2021.

Ren Yi, Hongyan Tang, Jiangpeng Rong, and Siwen Zhu. Unbiased pairwise learning from implicit feedback for recommender systems without biased variance control. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2461–2465, 2023.

Wenhui Yu and Zheng Qin. Sampler design for implicit feedback data by noisy-label robust learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 861–870, 2020.

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

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

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

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main contribution is the MTRec framework with the QR-IQL algorithm, which is described in Section 4 and justified in Section 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation is discussed in Appendix A.5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.

- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The derivations details are provided in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The implementation details are provided in Appendix A.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We provide implementation details in Appendix A.3 and will release the code upon acceptance of this paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All the data processing, training and evaluation details are described in Section 5 and Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the mean CTR with 95% confidence interval in Figure 2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The computational resources and training times are provided in Appendix A.3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We respect the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This work focuses on the technical foundations of sequential recommendation.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper does not release data or models with a high risk for misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the baselines are properly cited and introduced in Section 5.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Technical Appendices and Supplementary Material

A.1 Derivations of Problem 6

In this section, we show that Problem 4 can be simplified and translated to Problem 6, under the assumption that $Q(s,a)$ follows a quantile distribution $Z_\lambda(s,a)$. Given Problem 4:

$$\max_{Q\in\Omega} \mathbb{E}_{s,a\sim\rho_E}[\psi(Q(s,a) - \gamma\mathbb{E}_{s'\sim P(s,a)}V^*(s'))] - (1-\gamma)\mathbb{E}_{s_0\sim\rho_0}[V^*(s_0)], \tag{4}$$

The second term can be expanded as:

$$\begin{aligned}(1-\gamma)\mathbb{E}_{s_0\sim\rho_0}[V^*(s_0)] &= (1-\gamma)\sum_{t=0}^{\infty}\mathbb{E}_{D_E}[V^*(s_t)] - (1-\gamma)\sum_{t=1}^{\infty}\mathbb{E}_{D_E}[V^*(s_t)] \\ &= \mathbb{E}_{\rho_E}[V^*(s) - \gamma V^*(s')], \end{aligned} \tag{13}$$

Since we already have the expert trajectory dataset D_E, we will use ρ_E to estimate Equation 13. Then, on substituting $\psi(x) = x - \frac{1}{4\alpha}x^2$ and $V^*(s) = \log \sum_a \exp Q(s,a)$ in Problem 4, we have:

$$\max_Q \mathbb{E}_{\rho_E}[Q(s,a) - \log \sum_a \exp Q(s,a)] - \frac{1}{4\alpha}\mathbb{E}_{\rho_E}[(Q(s,a) - \log \sum_a \exp Q(s',a))^2]. \tag{14}$$

As $Q(s,a)$ is parameterized by quantiles $\{\lambda_i(s,a)\}_{i=1}^N$, we replace Q with Q_λ and obtain Problem 6.

A.2 Optimization steps for QR-IQL

In order to learn the distributional Q-function, we change the output layer of the Q-network to be of size $|A| \times N$, where $|A|$ denotes the size of the action space and N denotes the number of quantiles. For a given action, each of the N heads implicitly correlates to a λ_i. We adopt the widely used Pinball loss to learn the positions of $\{\lambda_i\}_{i=1}^N$, which is defined as:

$$p_\lambda(u) = \begin{cases} \lambda \cdot u, & u \geq 0 \\ (\lambda - 1) \cdot u, & u < 0 \end{cases}$$

where u represents the error between the predicted value and the target value at quantile λ. Intuitively, the Pinball loss pushes the quantile λ to the right position so that the predicted value distribution matches the target distribution. Based on the objective of Problem 6, we define two errors as:

$$u^1_{\lambda_i}(s,a) = \lambda_i(s,a) - \log \sum_a \exp \lambda_i(s,a),$$

$$u^2_{\lambda_i}(s,a,s') = \frac{1}{4\alpha}(\lambda_i(s,a) - \log \sum_a \exp \lambda_i(s',a))^2,$$

where λ_i is the i-th quantile and $\lambda_i(s,a)$ is the corresponding head of the Q_λ-network. Complete training procedures are provided in Algorithm 1.

Algorithm 1 QR-IQL Optimization Steps

Input: Interaction (expert) data D_E, number of quantiles N

1: Initialize network Q_λ;
2: **repeat**
3: Sample (batched) data (s,a,s');
4: Compute errors $u^1_{\lambda_i}(s,a)$ and $u^2_{\lambda_i}(s,a,s')$ for each quantile in $\{\lambda_i\}_{i=1}^N$;
5: Compute the Pinball losses $p_{\lambda_i}(u^1)$ and $p_{\lambda_i}(u^2)$ for each quantile in $\{\lambda_i\}_{i=1}^N$;
6: Compute the total loss as: $\sum_{i=1}^N [p_{\lambda_i}(u^1) + p_{\lambda_i}(u^2)]$;
7: Minimize the total loss by Adam Kingma and Ba [2015];
8: **until** convergence

Output: Q^*_λ

A.3 Implementation Details

Our experiments are run on a server with 2×AMD EPYC 7542 32-Core Processor CPU and 2×NVIDIA RTX 3090 graphics. For the offline experiments on Amazon datasets, it takes about 3 hours for 50,000 iterations of training with a 4000 batch size. For online experiments on Virtual Taobao, it takes about 4 hours for 50,000 RL training steps.

Algorithm 2 describes an overview of the implementation procedures. Basically, there are two stages. At stage 1 we focus on learning Q^*_λ and at stage 2 we focus on learning F^*_ζ. In practice, the architecture of the recommendation model F_ζ could be of various types. For example, in Section 5.1, we test seven widely used recommendation models: Wide&Deep Cheng et al. [2016], PNN Qu et al. [2016], DeepFM Guo et al. [2017], DIN Zhou et al. [2018], DIEN Zhou et al. [2019], LinRec Liu et al. [2023b] and SIGMA Liu et al. [2024]. We will use the same network architecture of F_ζ to construct Q_λ (except the output layer) to ensure that the features are processed properly. All the hyper-parameters of the backbone models follow their official codes. For implementation of MTRec, we select the number of quantiles $N = 10$ and the weight $\alpha = 0.5$ in Problem 6.

Actually, our MTRec framework is industrial friendly due to the following merits. First, all the training procedures are run in an offline manner, saving the cost of building online user-system interaction environments. Second, since Q_λ shares the model architecture with F_ζ, we do not need to build a mental reward model from scratch, saving a lot of work for adaptation.

Algorithm 2 Overall Implementation of MTRec

Input: Interaction (expert) data D_E

1: Initialize networks Q_λ and F_ζ;
2: Learn Q_λ^* according to Algorithm 1;
3: Obtain the mental rewards by
$r(s,a) \leftarrow Q_\lambda^*(s,a) - \gamma V^*(s'), \forall s, a \sim D_E$;
4: Patch the mental rewards to D_E;
5: Learn F_ζ^* according to Equation 10 or Equation 12;

Output: Aligned Recommendation Model F_ζ^*

A.4 Details on the Amazon datasets

The Amazon dataset McAuley et al. [2015] collects user review data from amazon.com. The crawled reviews have a time span from May 1996 to July 2014. The dataset can be divided into many subsets according to the various product categories. To verify the effectiveness of MTRec, we utilized two subsets of the Amazon dataset: Books and Electronics. We treated the reviews as user behaviors and sorted the reviews from each user chronologically. Based on a user's historical behaviors, our goal was to predict whether the user would write a review.

Dataset	User	Item	Categoriy	Sample
Books	603,668	367,982	1,600	603,668
Electronics	192,403	63,001	801	192,403

Table 2: The statistics of the Amazon datasets.

A.5 Limitations

While our work opens a door for studying users' intrinsic rewards during their interaction with the recommender systems, it still lacks a systematic method to thoroughly evaluate the learned mental reward model. In future works, we plan to construct a comprehensive benchmark involving large-scale human studies to further evaluate the mental reward model.