UNEX-RL: Reinforcing Long-Term Rewards in Multi-Stage Recommender Systems with UNidirectional EXecution

Gengrui Zhang[1*], **Yao Wang**[1*], **Xiaoshuang Chen**[1*], **Hongyi Qian**[2], **Kaiqiao Zhan**[1], **Ben Wang**[1†]

[1]Kuaishou Technology, Beijing, China
[2]Beihang University, Beijing, China
{zhanggengrui, wangyiyan, chenxiaoshuang, zhankaiqiao, wangben}@kuaishou.com, qianhongyi@buaa.edu.cn

Abstract

In recent years, there has been a growing interest in utilizing reinforcement learning (RL) to optimize long-term rewards in recommender systems. Since industrial recommender systems are typically designed as multi-stage systems, RL methods with a single agent face challenges when optimizing multiple stages simultaneously. The reason is that different stages have different observation spaces, and thus cannot be modeled by a single agent. To address this issue, we propose a novel UNidirectional-EXecution-based multi-agent Reinforcement Learning (UNEX-RL) framework to reinforce the long-term rewards in multi-stage recommender systems. We show that the unidirectional execution is a key feature of multi-stage recommender systems, bringing new challenges to the applications of multi-agent reinforcement learning (MARL), namely the observation dependency and the cascading effect. To tackle these challenges, we provide a cascading information chain (CIC) method to separate the independent observations from action-dependent observations and use CIC to train UNEX-RL effectively. We also discuss practical variance reduction techniques for UNEX-RL. Finally, we show the effectiveness of UNEX-RL on both public datasets and an online recommender system with over 100 million users. Specifically, UNEX-RL reveals a 0.558% increase in users' usage time compared with single-agent RL algorithms in online A/B experiments, highlighting the effectiveness of UNEX-RL in industrial recommender systems.

1 Introduction

Recent recommender systems have achieved great success in optimizing immediate engagement, such as click-through rates (Pi et al. 2020; Lian et al. 2018). However, in real-life applications, it is also necessary to improve the long-term reward, including the total watch time, the session length, the users' retention, etc. The long-term reward is more desirable than immediate engagement because it directly affects some important operational metrics, e.g., daily active users (DAUs) and dwell time (Cai et al. 2023a). Different from the immediate engagement, which has been extensively investigated (Wang et al. 2021; Zhou et al. 2018, 2019; Pi et al. 2020; Lian et al. 2018), it is more challenging to improve the long-term reward in recommender systems, since the long-term reward is affected by users' overall satisfaction of the
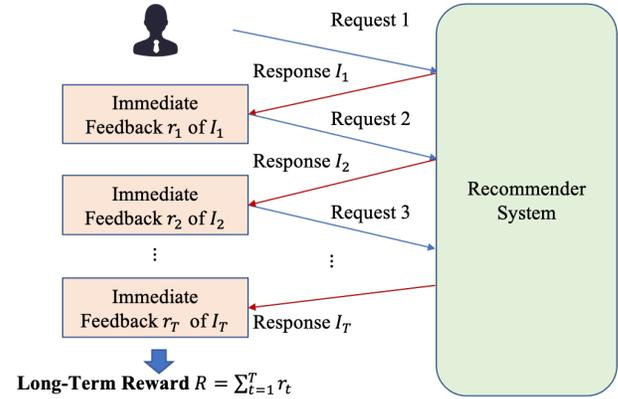
Figure 1: Long-term rewards in recommender systems.

entire set of recommended items (Xue et al. 2022; Wang et al. 2022), and it is difficult to relate the long-term reward to a single recommended item. Figure 1 shows the long-term reward of a recommender system.

Recently, a growing literature has focused on applying reinforcement learning (RL) to recommender systems due to its ability to improve the long-term reward (Chen et al. 2019; Gao et al. 2022a; Zou et al. 2019; Cai et al. 2023a). RL-based methods treat users as the environment and the recommender system as the agent, and then model the sequential interactions between users and the recommender system as Markov Decision Processes (MDP). After such modeling, several kinds of RL methods (Fan et al. 2020; Lillicrap et al. 2015; Fujimoto, Hoof, and Meger 2018) can be employed to optimize users' long-term rewards.

Despite the growing interest in RL-based recommendation, it is still challenging to apply RL to industrial-level recommender systems. A significant challenge arises from the fact that industrial recommender systems are typically built upon a multi-stage structure, including matching, pre-ranking, ranking, and re-ranking, to provide real-time recommendations from tens of millions of candidate items in a low latency (Wang et al. 2020b). The upstream stages utilize lightweight models to provide a subset of the candidates to the downstream stages for more precise estimation. Cooperation of the multiple stages is essential for optimizing the long-term rewards (Zhang et al. 2023). However, it is challenging to apply RL to the joint optimization of multi-

ple stages due to the fact that different stages have different observation spaces, e.g., the predictions or statistics of the candidate sets at each stage. This feature makes it inappropriate to model multiple stages in a single-agent RL model.

Recently, multi-agent reinforcement learning (MARL) has reached success in RL problems with multiple agents. Particularly, the centralized training with decentralized execution (CTDE) paradigm has made significant progress in cooperative MARL tasks (Rashid et al. 2020). However, challenges arise when training MARL in multi-stage recommender systems. The key problem is that CTDE assumes that observations of all the agents are simultaneously sampled from the replay buffer in the training stage, but in a multi-stage recommender system, even a slight change of the actions in upstream stages may cause different candidates in downstream stages, leading to different observations and actions in downstream stages. This feature contradicts the assumption of CTDE and deteriorates the performance when training MARL with CTDE.

This paper proposes a novel UNidirectional-EXecution-based multi-agent Reinforcement Learning (UNEX-RL) framework to apply MARL to multi-stage recommender systems. The unidirectional execution is a unique feature of multi-stage recommender systems compared to typical MARL applications. It leads to two effects in the training process of MARL, namely the observation dependency (OD) in critic learning and the cascading effect (CE) in actor learning. OD and CE deteriorate the performance of traditional CTDE-based approaches in the training of MARL. To mitigate these two effects, we propose a novel cascading information chain (CIC) approach to train UNEX-RL effectively. Moreover, we provide practical variance reduction techniques to further improve the performance. UNEX-RL shows significant improvement in optimizing the long-term rewards in multi-stage recommender systems.

In summary, our contributions are as follows:

- We introduce a novel UNEX-RL framework to improve the long-term rewards of multi-stage recommender systems. To the best of the authors' knowledge, we are the first to investigate the application of RL in industrial multi-stage recommender systems.

- We show that the unidirectional execution in multi-stage recommender systems leads to two effects, namely OD and CE. Then we provide a CIC approach to solve these problems. CIC shows a significant improvement over traditional CTDE-based training methods of MARL.

- We also discuss practical variance reduction techniques, i.e. stopping gradient (SG) and category-quantile rescaling (CQR), to improve the performance of UNEX-RL.

- We conduct extensive experiments in both public datasets and an online recommender system serving over 100 million users. UNEX-RL achieved a 0.558% increase in users' usage time in online A/B experiments of a real-world multi-stage recommender system.

2 Related Work

2.1 RL in Recommender Systems

Due to the ability to improve cumulative reward, RL has gathered increasing attention in the realm of recommender systems (Shani et al. 2005; Zou et al. 2019; Zheng et al.
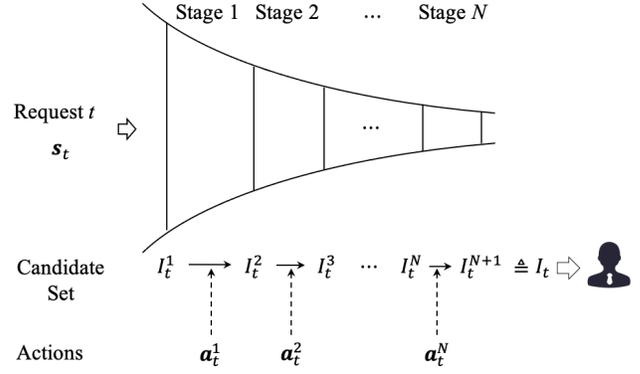
Figure 2: A multi-stage recommender system.

2018; Chen et al. 2019). RL models multiple interactions between users and recommender systems as MDP, aiming to optimize the long-term cumulative reward of the recommender system. Value-based approaches usually estimate users' preference for each item by Q networks, and select the top k items for users(Zheng et al. 2018; Zhao et al. 2018). Policy-based approaches usually use actor-critic-based methods to learn a policy that aims to improve users' satisfaction(Cai et al. 2023a,b; Xue et al. 2022, 2023). However, previous studies focused on improving a certain part of recommender systems with a single agent, and did not consider the multi-stage structure of industrial-level recommender systems, which is the contribution of our work.

2.2 MARL

In fully cooperative MARL, agents are trained to collaborate and achieve better cooperation by sharing the same reward. The CTDE paradigm learns a centralized critic with decentralized actor, has been increasingly used in recent years (Foerster et al. 2018; Peng et al. 2021; Wang et al. 2020a). In contrast, communication-based methods are used to transfer information between agents to facilitate the completion of cooperative tasks (Sukhbaatar, Fergus et al. 2016; Foerster et al. 2016). However, there are still challenges when applying MARL to multi-stage recommender systems (see Sec. 4), and to the best of the authors' knowledge, we are the first to discuss and solve these challenges.

3 Preliminary

3.1 Multi-Stage Recommender System

A typical multi-stage recommender system includes the matching, pre-ranking, ranking, and re-ranking processes. For the sake of convenience, we utilize the term "stages" to uniformly describe these processes in the following discussions. Specifically, there are N stages in the system, as shown in Figure 2. The system receives a user request at each time step t, with an observable user state s_t, including the user profile, the browsing history, etc. The system returns an item set I_t out of the universal candidate set I_U. The i-th stage takes a candidate set I_t^i as input, and outputs a subset I_t^{i+1}. We have $I_U = I_t^1 \supset I_t^2 \cdots \supset I_t^N \supset I_t^{N+1} = I_t$.

Generally, Stage i can be modeled as a selecting function
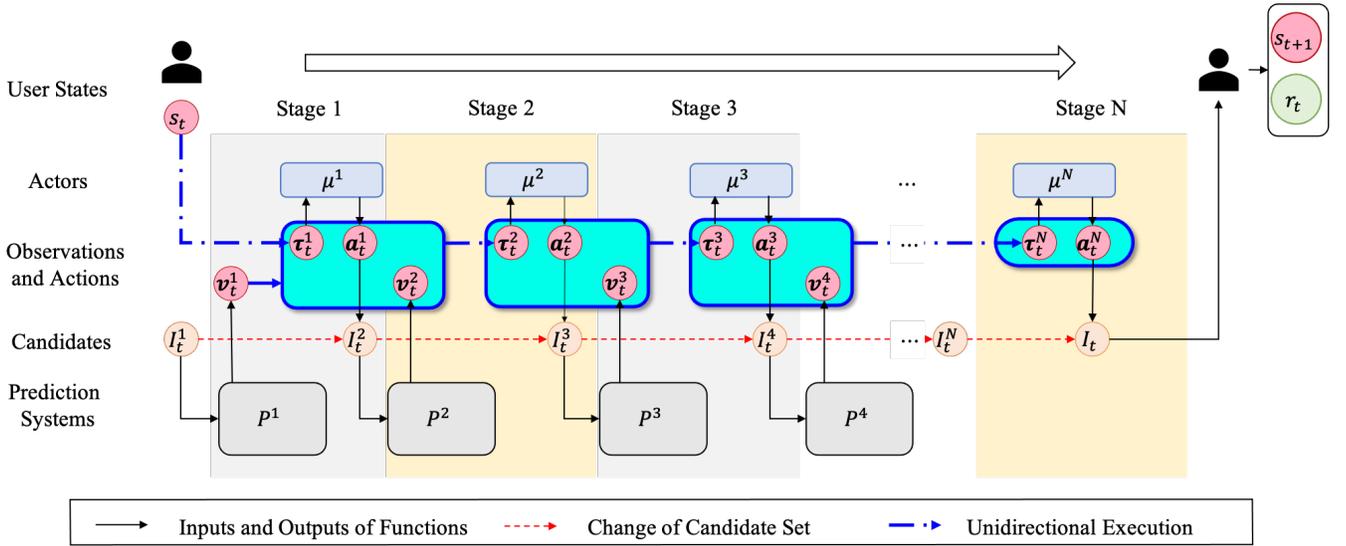
Figure 3: Overall framework of UNEX-RL.

$\mathcal{F}^i$ with input I_t^i and output I_t^{i+1}, parameterized by $\boldsymbol{a}_t^i$:

$$I_t^{i+1} = \mathcal{F}^i(I_t^i; \boldsymbol{a}_t^i), \qquad (1)$$

where $\boldsymbol{a}_t^i$ is to be determined. We would like to emphasize that Eq. (1) is a very general description of Stage i. Here we provide an example:

Example 1: This example is a simple multi-objective ranking module. The $\mathcal{F}^i$ is the combination of the following step:

1. Predicting the user engagement on each item, e.g., the click-through rate, the watch time, etc. The predictions are denoted by $\boldsymbol{p}_t^i$, a $|I_t^i| \times M^i$ matrix, where M^i is the number of predictions. $\boldsymbol{p}_t^i$ can be obtained from any DNN model.

2. Taking a linear combination of $\boldsymbol{p}_t^i$ with the weight vector $\boldsymbol{a}_t^i \in R^{M^i}$, i.e. $\boldsymbol{q}_t^i = \boldsymbol{p}_t^i \boldsymbol{a}_t^i \in R^{|I_t^i|}$.

3. Sorting the items by $\boldsymbol{q}_t^i$ and output the top items as I_t^{i+1}.

In this example, the action $\boldsymbol{a}_t^i$ acts as the weights to merge the predictions of each item in $\boldsymbol{p}_t^i$.

Problem Formulation   We focus on the optimization of the actions $\boldsymbol{a}_t^i$ at all the stages and all the steps, aiming at improving the user's long-term reward. Formally we have:

$$\max_{\boldsymbol{a}_t^{1:N}} R_t = \sum_{t'=t}^{\infty} \gamma^{t'-1} r_{t'}, \qquad (2)$$

where the upper bound "∞" indicates a summation of the rewards until the user leaves, and γ is the discount factor.

MARL-based approaches are usually used to improve the long-term reward with multiple cooperative agents. Before proposing our UNEX-RL framework, we first provide some basic concepts of MARL.

3.2   MARL

Agents in cooperative MARL aim to jointly maximize their collective rewards. It can be modeled as a decentralized partially observable Markov decision process (Dec-POMDP)

(Oliehoek, Amato et al. 2016), consisting of a tuple $\mathcal{G} = \{\mathcal{N}, \mathcal{S}, \mathcal{A}, P, R, \Omega, O, i, \gamma\}$, where $\mathcal{N}$ is the set of N agents, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition function, R is the reward function, Ω is the observation space, and O is the observation function. At Step t, the environment falls into a state $\boldsymbol{s}_t \in \mathcal{S}$, and each agent $i \in \mathcal{N}$ receives an observation $\boldsymbol{\tau}_t^i$ drawn according to the observation function $O(\boldsymbol{s}_t, i)$. Each agent learns a policy function μ^i parameterized by θ^i, and outputs the action $\boldsymbol{a}_t^i \in \mathcal{A}$ as

$$\boldsymbol{a}_t^i = \mu^i\left(\boldsymbol{\tau}_t^i; \theta^i\right). \qquad (3)$$

After executing the actions of N agents, the system transits to the next state according to the transition function: $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t^{1:N})$ and returns the reward $r_t = R(\boldsymbol{s}_t, \boldsymbol{a}_t^{1:N})$. Here we use "$1:N$" to represent a traverse from 1 to N.

CTDE is a training paradigm of MARL with a centralized critic and decentralized actors, and MADDPG (Lowe et al. 2017) is one of the representative examples. MADDPG learns a centralized critic Q^i for each agent i by sharing the global observation and actions in the training stage:

$$Q^i(\boldsymbol{\tau}_t^{1:N}, \boldsymbol{a}_t^{1:N}) = \mathbf{E}_{\boldsymbol{\tau}_{t+1:\infty}^{1:N}, \boldsymbol{a}_{t+1:\infty}^{1:N}}\left[R_t^i|\boldsymbol{\tau}_t^{1:N}, \boldsymbol{a}_t^{1:N}\right], \quad (4)$$

where R_t^i is the long-term reward of Agent i defined in Eq. (2). Given Q^i, the loss of critic learning in CTDE writes:

$$loss(\phi^i) = \mathbf{E}_D[(y^i - Q^i(\boldsymbol{\tau}_t^{1:N}, \boldsymbol{a}_t^{1:N}; \phi^i))^2], \qquad (5)$$

where

$$y^i = r_t^i + \gamma * Q^i(\boldsymbol{\tau}_{t+1}^{1:N}, \boldsymbol{a}_{t+1}^{1:N}; \phi^{i-}), \qquad (6)$$

and ϕ^i is the parameters of Q^i for Agent i. $\boldsymbol{\tau}_{t+1}^{1:N}$ is the observation of Step $t + 1$, which is obtained from the replay buffer, and $\boldsymbol{a}_{t+1}^i = \mu^i(\boldsymbol{\tau}_{t+1}^i; \theta^{i-})$ is the action of target policy with delayed parameters θ^{i-}. ϕ^{i-} is the parameters of the target critic, and D is the replay buffer. The policy gradient is calculated separately for each agent:

$$\nabla_{\theta^i} J = \mathbf{E}_D\left[\nabla_{\theta^i}\mu^i \nabla_{\mu^i} Q^i\left(\boldsymbol{\tau}_t^{1:N}, \boldsymbol{a}_t^{1:N}\right)\big|_{\boldsymbol{a}_t^i = \mu^i(\boldsymbol{\tau}_t^i)}\right] \quad (7)$$

where $\boldsymbol{a}_t^i$, the action of agent i, is obtained of the policy μ^i, while the other actions are sampled from the replay buffer.

4 Method

This section provides the UNEX-RL method. Firstly, we provide the overall framework and show that the unidirectional execution leads to several challenges i.e. OD and CE. Then, we propose CIC to tackle these challenges. Finally, practical variance reduction techniques are provided to reduce the variance and to improve the performance.

4.1 Overall Framework

The UNEX-RL framework is illustrated in Figure 3. When a user opens the app, a session begins, which consists of multiple requests until the user leaves the app. At Step t, the system obtains a user state s_t together with the first candidate set I_t^1. The N stages are modeled as N agents. Agent i receives the information available at Stage i, denoted by $\boldsymbol{\tau}_t^i$, and uses the policy function μ^i to provide the action $\boldsymbol{a}_t^i$. The action $\boldsymbol{a}_t^i$ is used to select I_t^{i+1} out of I_t^i (see Eq. (1)). After the N stages, the system outputs the final recommended set I_t. Then, the user provides feedback r_t and determines whether to send the next request or leave. UNEX-RL aims to maximize the long-term reward R_t defined in Eq. (2).

There are three parts in the agent observation $\boldsymbol{\tau}_t^i$:

- The inherit part $\boldsymbol{\tau}_t^{i-1}$ when $i > 1$, which means Stage i possesses all the observations in Stage $i-1$.
- The action of the previous stage, i.e. $\boldsymbol{a}_t^{i-1}$.
- The information obtained in this stage for the first time, e.g., new predictions or statistics about the candidate set I_t^i. We denote it as $\boldsymbol{v}_t^i$, and generally, we assume $\boldsymbol{v}_t^i$ to be obtained from an information extraction process:

$$\boldsymbol{v}_t^i = \mathcal{P}^i\left(I_t^i; P^i\right), \qquad (8)$$

where P^i is the parameter. We do not explicitly model $\mathcal{P}^i$, but we assume it to be accessible in training.

Formally we have:

$$\boldsymbol{\tau}_t^1 = \left[\boldsymbol{s}_t, \boldsymbol{v}_t^1\right], \boldsymbol{\tau}_t^{i+1} = \left[\boldsymbol{\tau}_t^i, \boldsymbol{a}_t^i, \boldsymbol{v}_t^{i+1}\right]. \qquad (9)$$

The **unidirectional execution**, which is the key difference between UNEX-RL and traditional MARL, arises from the information extraction process $\mathcal{P}^i$ in Eq. (8). Since $\boldsymbol{v}_t^i$ must be obtained after the candidate set I_t^i is determined, Agent i must be executed after Agent $i-1$, and all the agents must be executed in series (see the blue arrows in Figure 3).

MARL is usually trained by CTDE (Lowe et al. 2017), but the unidirectional execution brings two challenges:

- **Observation Dependency (OD)**: The OD problem is about the critic learning, i.e., Eq. (5)(6). In the calculation of target actions in Eq. (6), the observations $\boldsymbol{\tau}_{t+1}^{1:N}$ are sampled from the replay buffer, and the actions $\boldsymbol{a}_{t+1}^{1:N}$ are calculated via the target policy. However, if we change the actions $\boldsymbol{a}_{t+1}^i$, the observations $\boldsymbol{\tau}_{t+1}^{i+1}$ in Stage $i+1$ will also change according to Eq. (9), which contradicts the assumption that $\boldsymbol{\tau}_{t+1}^{1:N}$ is predefined in the replay buffer.
- **Cascading Effect (CE)**: The CE problem is about the actor learning part, i.e., Eq. (7), which optimizes the policy function μ^i of the i-th agent under the condition that the actions of other agents are sampled from the replay buffer. However, according to Eq. (9), the action $\boldsymbol{a}_t^i$ is contained in the observation of downstream stages, i.e., $\boldsymbol{\tau}_t^{i+1:N}$, hence will affect the decisions of downstream stage actors, which is not taken into account in Eq. (7).

4.2 Training of UNEX-RL

This subsection presents the training of UNEX-RL, especially the solution to the OD and CE problems, as shown in Algorithm 1. We use the actor-critic structure. Different from Eq. (5)(6), all the agents share the reward r_t from the user, and thus we only use a global critic Q^g.

We first consider the solution to OD. A key dilemma is that the observations from the replay buffer contradict the observations of target actors in the training stage. Thus, we need to find an independent subset of the observations. Our solution is based on the following finding: given all the target actors and all the checkpoints of the system, we are able to replay the total recommendation only based on the first observation in the training stage. Formally we have:

Theorem 4.1. *Denote $\mathcal{E}(\cdot)$ as the information implied by input variables. Assume that the parameters P^i of the information extraction in Eq. (8) are given for all i. Then $\forall i, 2 \leq i \leq N$, the set $\left\{\boldsymbol{\tau}_t^1, \boldsymbol{a}_t^{1:i-1}\right\}$ contains all the information of the observation $\boldsymbol{\tau}_t^i$, i.e. $\mathcal{E}\left(\boldsymbol{\tau}_t^i\right) \subset \mathcal{E}\left(\left\{\boldsymbol{\tau}_t^1, \boldsymbol{a}_t^{1:i-1}\right\}\right)$.*

Please refer to Appendix. A for the detailed proof of Theorem 4.1. Note that the assumption of a given parameter P^i is reasonable since it is common to replay the online recommendation processes based on model checkpoints in the offline training (Chen et al. 2019).

Theorem 4.1 models the information flow of the unidirectional execution in Fig. 3. Based on this finding, we propose a cascade information chain (CIC) method, described in Algorithm 2, to obtain $\boldsymbol{\tau}_t^{1:N}$ and $\boldsymbol{a}_t^{1:N}$ only from the first observation $\boldsymbol{\tau}_t^1$. CIC executes iteratively, leveraging the information of upstream stages to obtain the observations of downstream stages until the whole observations and actions have been obtained. For simplicity, we represent CIC as:

$$\left[\boldsymbol{\tau}_t^{1:N}, \boldsymbol{a}_t^{1:N}\right] = C\left(\boldsymbol{\tau}_t^1; \theta^{1:N}, P^{1:N}\right). \qquad (10)$$

Via CIC, the critic learning can be formulated as:

$$\mathcal{L}(\phi) = \mathbf{E}_D[(y^g - Q^g(\boldsymbol{\tau}_t^{1:N}, \boldsymbol{a}_t^{1:N}; \phi))^2], \qquad (11)$$

where

$$y_t^g = r_t + \gamma * Q^g(C\left(\boldsymbol{\tau}_{t+1}^1; \theta^{1:N-}, P^{1:N}\right); \phi^-), \qquad (12)$$

ϕ and ϕ^- are parameters of the critic and the target critic. Here the only independent observation which needs to be obtained from the replay buffer is $\boldsymbol{\tau}_{t+1}^1$, while the observations of the downstream stages are obtained from CIC. Therefore, CIC provides a solution to the OD problem.

Now we discuss the actor learning of UNEX-RL and show that CIC can also be used to solve the CE problem. Note that traditional CTDE in Eq. (7) does not consider the influence of $\boldsymbol{a}_t^i$ to the actions of its downstream agents. In contrast, any change of μ^i will change not only the corresponding action $\boldsymbol{a}_t^i$, but also the actions of succeeding stages $\boldsymbol{a}_t^{i+1:N}$ in CIC. Therefore, we just need to use CIC in the policy gradient:

$$\nabla_{\theta^i} J_i = \mathbf{E}_D\left[\nabla_{\theta^i}\mu^i \nabla_{\mu^i} Q^g\left(C\left(\boldsymbol{\tau}_t^1; \theta^{1:N}, P^{1:N}\right)\right)\right] \qquad (13)$$

In a word, we use CIC to replay the execution process in the training stage to solve OD and CE problems. The difference between CTDE and CIC is shown in Figure 4.

Algorithm 1: The training process of UNEX-RL.

1: **Input:** $\{s_{1:T}, \tau_{1:T}^{1:N}, a_{1:T}^{1:N}, r_{1:T}\}$ for each user.
2: **Output:** A critic function $Q^g\left(\tau_t^{1:N}, a_t^{1:N}; \phi\right)$ parameterized by ϕ; N policies $\mu_i\left(\tau_t^i; \theta^i\right)$, parameterized by $\theta^i, 1 \leq i \leq N$.
3: **for** each user session with T requests from the replay buffer **do**
4:     **for** $t = 1, \cdots, T$ **do**
5:         Collect the reward r_t, the observation $\tau_t^{1:N}$, and all the actions $a_t^{1:N}$ from the replay buffer.
6:         Collect the first-stage observation of the next step τ_{t+1}^1 from the replay buffer.
7:         Calculate the CIC according to Eq. (10) to obtain the target action $a_{t+1}^{1:N}$ and the updated observation $\tau_{t+1}^{1:N}$ from τ_t^1.
8:         Critic learning: $\phi \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}$, where $\mathcal{L}(\phi)$ is defined in Eq. (11)(12), and α is the learning rate.
9:         Actor learning: $\theta^i \leftarrow \theta^i - \beta \nabla_{\theta^i} J_i$, where $\nabla_{\theta^i} J_i$ is defined in (13), and β is the learning rate.
10:     **end for**
11: **end for**

(a) Train with CTDE
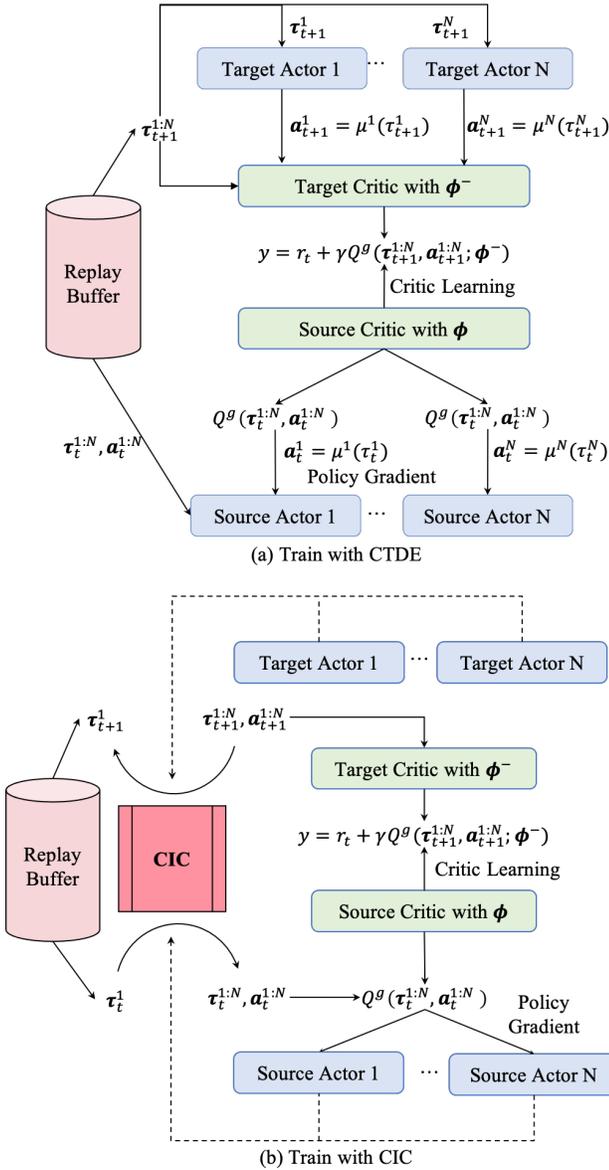
(b) Train with CIC

Figure 4: Train with CTDE and CIC.

Algorithm 2: The process of CIC.

1: **Input:** The first observation τ_t^1, actor parameters $\theta^{1:N}$, and prediction system parameters $P^{1:N}$.
2: **Output:** Actions $a_t^{1:N}$ and observations $\tau_t^{1:N}$.
3: $a_t^1 = \mu^1\left(\tau_t^1; \theta^1\right)$
4: **for** $i = 2, \cdots, N$ **do**
5:     $I_t^i = \mathcal{F}^i\left(I_t^{i-1}; a_t^{i-1}\right)$, as shown in Eq. (1).
6:     $v_t^i = \mathcal{P}^i\left(I_t^i; P^i\right)$, as shown in Eq. (8).
7:     $\tau_t^i = \left[\tau_t^{i-1}, a_t^{i-1}, v_t^i\right]$, as shown in Eq. (9).
8:     $a_t^i = \mu^i\left(\tau_t^i; \theta^i\right)$
9: **end for**

4.3 Variance Reduction Techniques

RL in recommender systems suffers from large variance due to the highly sparse data (Chen et al. 2019), and MARL makes things even worse due to the decentralized actions of multiple agents (Lowe et al. 2017). Here we provide some practical variance reduction techniques.

Stopping Gradient (SG) The CIC function provides multiple paths from Q^g to μ^i, which will make the gradient somehow uncontrollable in backward propagation. Here we stop the gradient from $\mu^{i+1:N}$ to μ^i, and only use the direct gradient from Q^g to μ^i in critic learning.

Category Quantile Rescale (CQR) The user's feedback r_t depends not only on the user's interest in the recommended item but also on the user's or the item's bias feature. Such bias leads to a large range of r_t, resulting in a large variance (Zhan et al. 2022). Please refer to Appendix. B for more details about these biases. Here we propose CQR, a reward-reshaping method to solve this problem. We obtain the group of users similar to the user u, denoted by $\mathcal{G}_u$, and the group of items similar to the item i, denoted by $\mathcal{G}_i$. There are sufficient discussions on how to find similar users and items (Pi et al. 2020; Chen et al. 2021). The CQR reward is the quantile of the reward conditioned on $\mathcal{G}_u$ and $\mathcal{G}_i$:

$$\tilde{r}_{t,ui} = \Phi\left(r_{t,ui} | \mathcal{G}_u, \mathcal{G}_i\right) \qquad (14)$$

where Φ denotes the cumulative density function.

Clearly, $\tilde{r}_{t,ui}$ obeys a uniform distribution, and this uniformity remains when conditioned on any $\mathcal{G}_u$ and $\mathcal{G}_i$. Thus, the range of $\tilde{r}_{t,ui}$ will be moderate compared with r_t, which can reduce the variance of critic learning.

5 Evaluation

We conduct extensive offline and online experiments, and answer the following research questions (RQs):

- **RQ1**: How does the UNEX-RL framework perform compared to other state-of-the-art (SOTA) methods?
- **RQ2**: Does UNEX-RL with CIC effectively solve the problems in multi-stage recommender systems?
- **RQ3**: How do the variance reduction techniques contribute to the performance?
- **RQ4**: Can the UNEX-RL framework lead to improvements in real-world recommender systems?

5.1 Preparing the Dataset

Dataset and Simulator We select KuaiRand (Gao et al. 2022b) as the offline experimental dataset. KuaiRand is a public dataset obtained from Kuaishou, containing 27,285 users and 32,038,725 items. It encompasses contextual features of both users and items, along with multiple feedback signals from users.

To emulate real user behavior, we construct a user simulator to act as the environment and mimic the user's interaction with the recommender system. After receiving the recommended items, the user simulator provides immediate feedback and determines whether to send the next request. We follow the work in (Xue et al. 2023) to introduce a quitting mechanism: the user will exit the session once they have exhausted their satisfaction, and an early quit will also lead to inferior performance in terms of reward metrics.

Offline Experiment Details Align with industrial recommender systems, we consider three stages i.e., matching, pre-ranking, and ranking. The operation in each stage is the same as Example 1 in Sec. 3.1, i.e., a linear ranking function $q_t^i = p_t^i a_t^i$, where $p_t^i \in |I_t^i| \times 3$ represents the prediction of three feedbacks, i.e., like, long view and watch time. The action $a_t^i \in R^3$ is the weight of the linear ranking. The state s_t comprises the user profile, the behavior history, and the request context. The observation v_t^i includes statistics of the candidate items. To ensure fairness, we use a consistent network architecture, i.e., a multi-layer perceptron with 5 layers, for actors and critics of all the compared methods. The hyperparameters can be found in Table 3 in Appendix. C.

5.2 Compared Methods

- **Cross Entropy Method (CEM)** (Rubinstein and Kroese 2004): a black-box optimization method commonly used for hyper-parameter optimization. We use CEM to search the best parameters a_t^i at each stage.
- **DDPG** (Lillicrap et al. 2015): Since DDPG is a single-agent method, We deploy it at the final ranking stage.
- **TD3** (Fujimoto, Hoof, and Meger 2018): Similar to DDPG, we deploy TD3 at the final ranking stage.
- **UNEX-RL-CTDE**: UNEX-RL trained by CTDE along with CQR in a multi-stage recommender system.
- **UNEX-RL-CTDE (w/o CQR)**: UNEX-RL-CTDE without CQR.
- **UNEX-RL-CIC**: UNEX-RL trained under CIC in Algorithm 1, along with SG and CQR.
- **UNEX-RL-CIC (w/o CQR)**: UNEX-RL-CIC without CQR.

Methods	WatchTime (s)	Session Length
CEM	654.0	15.3
DDPG	732.6	18.2
TD3	763.2	18.9
UNEX-RL-CTDE(w/o CQR)	782.5	19.7
UNEX-RL-CTDE	887.2	21.6
UNEX-RL-CIC(w/o CQR)	974.3	22.7
UNEX-RL-CIC(w/o SG)	752.1	18.4
UNEX-RL-CIC	**1056.2**	**24.2**

Table 1: The overall performance of different methods.

- **UNEX-RL-CIC (w/o SG)**: UNEX-RL-CIC without SG.

5.3 Evaluation Metric

We measure two kinds of long-term rewards, namely the **WatchTime**, i.e., the accumulated watching time of all the items watched in the session, and the **Session Length**, i.e., the number of items watched in the session.

5.4 Performance Comparison (RQ1)

The results of the offline experiments are shown in Table 1. CEM searches a global weight vector a^i for each stage. Since DDPG and TD3 are single-agent methods, we deploy them at the final ranking stage. Compared to CEM, RL-based algorithms perform much better due to their advantage of optimizing the long-term rewards of the users. Furthermore, UNEX-RL achieves significantly better performance, showing that MARL can better release the ability of RL in a multi-stage recommendation system. Moreover, the comparison between UNEX-RL-CIC and UNEX-RL-CTDE shows the effectiveness of our proposed CIC approach.

5.5 Impacts of Multiple Stages (RQ2)

To demonstrate that the UNEX-RL framework can better adapt to multi-stage recommender systems, we analyze the impact of the number of agents on different methods, as shown in Figure 5. CEM can be regarded as a method with 0 agents. For all other RL-based methods, we adjust the number of agents from 1 to 3. Specifically for DDPG, we deploy an independent DDPG at each stage.

When there is only one agent, UNEX-RL-CTDE and UNEX-RL-CIC degrade to DDPG, and the three methods exhibit the same performance. As the number of agents increases, both UNEX-RL-CTDE and UNEX-RL-CIC show significant improvements, proving the effectiveness of UNEX-RL. However, multiple independent DDPGs show even lower performance since the lack of cooperation among the agents. When there are multiple agents, UNEX-RL-CIC outperforms UNEX-RL-CTDE, showing that the CIC training paradigm can better facilitate cooperation among the agents in multi-stage recommender systems.

5.6 Impacts of Variance Reduction (RQ3)

To analyze the contribution of the variance reduction techniques, we deploy the ablation study of UNEX-RL, shown in Figure 6. By removing CQR from UNEX-RL-CTDE and UNEX-RL-CIC, the performance significantly deteriorates,
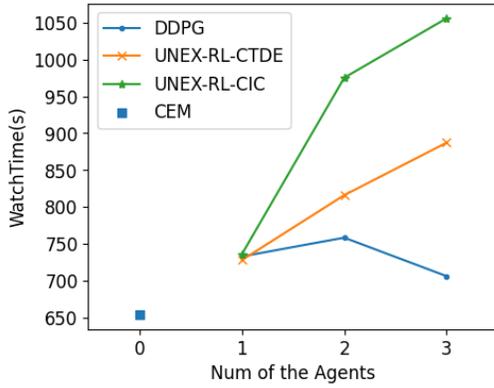
Figure 5: Performance of different numbers of agents.
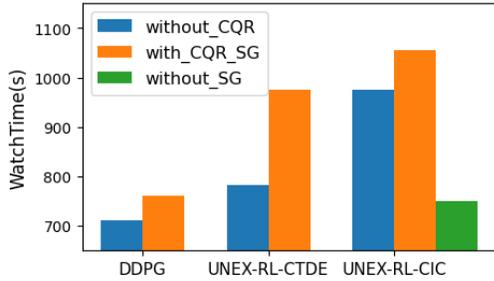
Figure 6: Performance of Variance Reduction Techniques.

Figure 7: Results of a long period online experiment.

Algorithms	Session WatchTime	Daily WatchTime
CEM	-	-
DDPG	+ 0.233%	+ 0.219%
TD3	+ 0.414%	+ 0.395%
UNEX-RL-CTDE	+ 0.610%	+ 0.602%
UNEX-RL-CIC	**+ 0.970%**	**+ 0.953%**

Table 2: Results of online experiments compared with CEM.

proving that CQR can significantly improve the performance of UNEX-RL. Also, by removing SG from UNEX-RL-CIC, the performance will be much worse, showing that SG is a key technique to improve the training of CIC. We do not discuss the impacts of SG on CTDE and DDPG since these two methods do not contain gradients from μ^{i+1} to μ^i.

5.7 Online A/B Experiments (RQ4)

We deploy UNEX-RL on Kwai, a short video app with over 100 million users. Through online experiments, we compare UNEX-RL with the current SOTA methods. The baseline is CEM, and we successively experiment with DDPG, TD3, UNEX-RL-CTDE, and UNEX-RL-CIC. The evaluation metrics are the relative improvements of daily WatchTime and session-wise WatchTime.

In the online experiment, we deploy agents in the pre-ranking and ranking stages, and the two agents return actions as the weight vector of the ranking functions. The observations include user information that all the agents share, and stage-wise observations, i.e., the statistics of the predicted scores in the current stage.

Table 2 shows the comparisons of different methods. According to Table 2, DDPG and TD3 show significant improvements compared to CEM, showing the effectiveness of RL to improve users' long-term rewards. More importantly, both UNEX-RL-CTDE and UNEX-RL-CIC perform better than the RL methods with a single agent, indicating that the proposed UNEX-RL framework leads to improvements in real-world recommender systems. Notably, UNEX-RL-CIC achieves the best performance compared to other meth-

ods, i.e., a 0.953% gain of daily WatchTime compared with CEM, and a 0.558% gain compared with TD3. We emphasize that a 0.1% improvement holds statistical significance in our system.

Furthermore, we conduct a 150-day online experiment, with the baseline group remaining with the CEM approach, while the experimental group successively introduced TD3, UNEX-RL-CTDE, and UNEX-RL-CIC. DDPG is not introduced to the experimental group since it is tested in the same period as TD3 with a lower performance. We mark the deployment time of these methods, showing significant gains when deploying UNEX-RL. The results shows the improvement of UNEX-RL over existing methods.

6 Conclusion

This paper provides a general framework, i.e. UNEX-RL, for applying MARL to multi-stage recommender systems. UNEX-RL differs from traditional MARL in the unidirectional execution, which brings challenges to the training of UNEX-RL. To tackle the challenges of OD and CE arising from the unidirectional execution, we provide the CIC approach to effectively train UNEX-RL, and provide practical variance reduction techniques, i.e. SG and CQR, to further improve the performance. UNEX-RL has shown its effectiveness on both public datasets and a real-world recommender system, and has been deployed online, serving over 100 million users.

References

Cai, Q.; Liu, S.; Wang, X.; Zuo, T.; Xie, W.; Yang, B.; Zheng, D.; Jiang, P.; and Gai, K. 2023a. Reinforcing User

Retention in a Billion Scale Short Video Recommender System. *arXiv preprint arXiv:2302.01724*.

Cai, Q.; Xue, Z.; Zhang, C.; Xue, W.; Liu, S.; Zhan, R.; Wang, X.; Zuo, T.; Xie, W.; Zheng, D.; et al. 2023b. Two-Stage Constrained Actor-Critic for Short Video Recommendation. In *Proceedings of the ACM Web Conference 2023*, 865–875.

Chen, M.; Beutel, A.; Covington, P.; Jain, S.; Belletti, F.; and Chi, E. H. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 456–464.

Chen, Q.; Pei, C.; Lv, S.; Li, C.; Ge, J.; and Ou, W. 2021. End-to-end user behavior retrieval in click-through ratepre-diction model. *arXiv preprint arXiv:2108.04468*.

Fan, J.; Wang, Z.; Xie, Y.; and Yang, Z. 2020. A theoretical analysis of deep Q-learning. In *Learning for dynamics and control*, 486–489. PMLR.

Foerster, J.; Assael, I. A.; De Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Gao, C.; Li, S.; Lei, W.; Chen, J.; Li, B.; Jiang, P.; He, X.; Mao, J.; and Chua, T.-S. 2022a. KuaiRec: A fully-observed dataset and insights for evaluating recommender systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 540–550.

Gao, C.; Li, S.; Zhang, Y.; Chen, J.; Li, B.; Lei, W.; Jiang, P.; and He, X. 2022b. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 3953–3957.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247*.

Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2333–2338.

Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; and Sun, G. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1754–1763.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lowe, R.; Wu, Y. I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30.

Oliehoek, F. A.; Amato, C.; et al. 2016. *A concise introduction to decentralized POMDPs*, volume 1. Springer.

Peng, B.; Rashid, T.; Schroeder de Witt, C.; Kamienny, P.-A.; Torr, P.; Böhmer, W.; and Whiteson, S. 2021. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34: 12208–12221.

Pi, Q.; Zhou, G.; Zhang, Y.; Wang, Z.; Ren, L.; Fan, Y.; Zhu, X.; and Gai, K. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2685–2692.

Rashid, T.; Samvelyan, M.; De Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1): 7234–7284.

Rendle, S. 2010. Factorization machines. In *2010 IEEE International conference on data mining*, 995–1000. IEEE.

Rubinstein, R. Y.; and Kroese, D. P. 2004. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer.

Shani, G.; Heckerman, D.; Brafman, R. I.; and Boutilier, C. 2005. An MDP-based recommender system. *Journal of Machine Learning Research*, 6(9).

Sukhbaatar, S.; Fergus, R.; et al. 2016. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29.

Wang, R.; Shivanna, R.; Cheng, D.; Jain, S.; Lin, D.; Hong, L.; and Chi, E. 2021. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, 1785–1797.

Wang, Y.; Han, B.; Wang, T.; Dong, H.; and Zhang, C. 2020a. Dop: Off-policy multi-agent decomposed policy gradients. In *International conference on learning representations*.

Wang, Y.; Sharma, M.; Xu, C.; Badam, S.; Sun, Q.; Richardson, L.; Chung, L.; Chi, E. H.; and Chen, M. 2022. Surrogate for long-term user experience in recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4100–4109.

Wang, Z.; Zhao, L.; Jiang, B.; Zhou, G.; Zhu, X.; and Gai, K. 2020b. Cold: Towards the next generation of pre-ranking system. *arXiv preprint arXiv:2007.16122*.

Xue, W.; Cai, Q.; Xue, Z.; Sun, S.; Liu, S.; Zheng, D.; Jiang, P.; Gai, K.; and An, B. 2023. PrefRec: Recommender Systems with Human Preferences for Reinforcing Long-term User Engagement.

Xue, W.; Cai, Q.; Zhan, R.; Zheng, D.; Jiang, P.; and An, B. 2022. ResAct: Reinforcing Long-term Engagement in Sequential Recommendation with Residual Actor. *arXiv preprint arXiv:2206.02620*.

Zhan, R.; Pei, C.; Su, Q.; Wen, J.; Wang, X.; Mu, G.; Zheng, D.; Jiang, P.; and Gai, K. 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.

Zhang, Z.; Huang, Y.; Ou, D.; Li, S.; Li, L.; Liu, Q.; and Zeng, X. 2023. Rethinking the Role of Pre-ranking in Large-scale E-Commerce Searching System. *arXiv preprint arXiv:2305.13647*.

Zhao, X.; Zhang, L.; Ding, Z.; Xia, L.; Tang, J.; and Yin, D. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1040–1048.

Zheng, G.; Zhang, F.; Zheng, Z.; Xiang, Y.; Yuan, N. J.; Xie, X.; and Li, Z. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*, 167–176.

Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; and Gai, K. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 5941–5948.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1059–1068.

Zou, L.; Xia, L.; Ding, Z.; Song, J.; Liu, W.; and Yin, D. 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.