Sim-to-Real Interactive Recommendation via Off-Dynamics Reinforcement Learning

Junda Wu
New York University
New York City, NY, USA
jw6466@nyu.edu

Zhihui Xie
Shanghai Jiao Tong University
Shanghai, China
fffffarmer@sjtu.edu.cn

Tong Yu
Adobe Research
San Jose, CA, USA
tyu@adobe.com

Qizhi Li
Shanghai Jiao Tong University
Shanghai, China
qizhili@sjtu.edu.cn

Shuai Li*
Shanghai Jiao Tong University
Shanghai, China
shuaili8@sjtu.edu.cn

Abstract

Interactive recommender systems (IRS) have received growing attention due to its awareness of long-term engagement and dynamic preference. Although the long-term planning perspective of reinforcement learning (RL) naturally fits the IRS setup, RL methods require a large amount of online user interaction, which is restricted due to economic considerations. To train agents with limited interaction data, previous works often count on building simulators to mimic user behaviors in real systems. This poses potential challenges to the success of sim-to-real transfer. In practice, such transfer easily fails as user dynamics is highly unpredictable and sensitive to the type of recommendation task. To address the above issue, we propose a novel method, S2R-Rec, to bridge the sim-to-real gap via off-dynamics RL. Generally, we expect the policy learned by only interacting with the simulator can perform well in the real environment. To achieve this, we conduct dynamics adaptation to calibrate the difference of state transition using reward correction. Furthermore, we align representation discrepancy of items by representation adaptation. Instead of separating the above into two stages, we propose to jointly adapt the dynamics and representations, leading to a unified learning objective. Experiments on real-world datasets validate the superiority of our approach, which achieves about 33.18% improvements compared to the baselines.

1 Introduction

Recent years have featured a trend towards building interactive recommender systems (IRS) [3, 4, 32, 17, 27]. Compared with traditional recommender systems, IRS consider a more realistic scenario where the current user preference drifts over time dynamically. To implement IRS, previous works mainly focus on Multi-armed Bandit (MAB) and Reinforcement Learning (RL). However, the MAB approaches [16, 25, 26] assume little drift of user preferences over time, which may fail to model the dynamics in IRS [3, 32]. An alternative formulation of IRS is Markov Decision Process (MDP), which explicitly models state transition along with the planning procedure. In IRS, RL techniques have been recently gaining attention, showing their advantages in accommodating dynamic user preferences [33, 5]. To train agents with limited interaction data, existing RL models in IRS often assume that the simulation task and the real task are the same recommendation task, and count on building simulators to mimic user behaviors in real systems. However, the assumption may not hold

*Corresponding author

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Online.

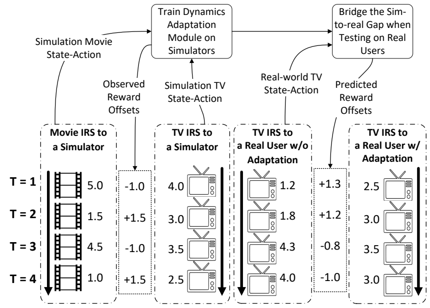


Figure 1: Dynamics adaptation in the sim-to-real IRS. In training on simulators, the Movie recommender and TV recommender interact with two user simulators. By learning the dynamics difference between interacting with the two simulators, the dynamics adaptation module is trained. In testing on real users, the predicted ratings are adjusted with reward offsets by the dynamics adaptation module.

in practice and, without any explicit dynamics adaptation considered, the trained models can easily achieve poor performances when recommending to new users.

The above issue of sim-to-real dynamics adaptation brings critical challenges to the success of applying RL for IRS. While many applications have met the difficulties in adapting model's dynamics from simulation to realistic tasks [20, 22, 1], this sim-to-real dynamics adaptation problem can be even more severe in IRS. Conventionally, simulation of robotic tasks is designed to imitate the same in the real world [29, 7]. In IRS, however, when new businesses are established, the new products in the real task may have never appeared to any users in the simulation. More importantly, consumer buying behaviors in different categories of goods can be hugely varied. For example, in a simple IRS use session in Figure 1, the IRS simulates on the movie recommendation task and is required to perform real recommendation for newly released TV series. By detecting invariant representations from movies and TVs, the user may express a consistent preference from training on simulators to testing on real users. However, we can also observe user dynamics differences between interacting with the movie simulator and the TV simulator. In Movie IRS to a simulator, the user expresses stronger preferences over movies with either high ratings or low ratings, while the ratings regarding TVs are less polarized. This indicates that the user may have a more indifferent taste to TVs, but more personalized inclination to certain movies. Such differences in rating dynamics, caused by the nature of different recommendation tasks in simulation and real-world, cannot be easily mitigated through user representation adaptation.

To explicitly solve the dynamics adaptation problem in sim-to-real IRS, we propose S2R-Rec. Inspired by [7], we imitate the real user dynamics by adjusting the reward objectives in simulation, and thus let the transition to be more smoothly adapted. Unlike tasks in robotics or games that can rely on simulators guided by explicit physical laws and game rules, IRS are usually faced with dynamics that only exist in latent spaces of user preference and item representations. Furthermore, as normally no items shared between sim-to-real, representation adaptation is also required to extract shared item features between simulation and real tasks. We introduce a representation adaptation method to extract invariant item representations. In order to recognize the patterns in item representations more relevant to user dynamics, we propose a collaborative adaptation method to jointly adapt the dynamics and item representations.

This paper makes three major contributions. (i) We propose a dynamics adaptation recommendation method by aligning sim-to-real MDPs with reward offsets. This mitigates user interactive behaviour differences between training on the simulators and testing on the real users. (ii) We introduce a representation adaptation method to extract shared item features between simulation and real tasks, which is critical for dynamics adaption since items from two tasks may have no overlap. (iii) We propose a joint adaptation method S2R-Rec to jointly align interactive dynamics and item representations. This improves traditional item representations adaptation by taking consideration of the user dynamics difference between simulation and real tasks.

2 Related Work

RL for Interactive Recommender Systems To capture the interactive nature of IRS, extensive effort [6, 30, 12] has been made to model the recommendation environment as a Markov Decision

Process (MDP) and then utilize RL algorithms to deliver the optimal policy. Previous works have proposed various kinds of RL approaches to maximize the expected recommendation reward in the long run. Deep Q-Network (DQN) is proposed for news recommendation [31]. Actor-critic approaches with a state representation module are developed to explicitly model user-item interactions [17]. REINFORCE is utilized to get rid of problems with regard to a large corpus of items [4]. In [10], the problem of learning to rank is formulated via an MDP and the agent learns by deterministic policy gradient. The auxiliary tasks such as user response prediction to construct useful representations and augment the training process in [5] . Nevertheless, previous works usually rely on simulators for policy training, which poses challenges to the success of sim-to-real transfer. Therefore, we aim to address the discrepancy between the simulation and real environment to boost RL methods.

Sim-to-Real Gap and Off-Dynamics RL To construct a more accurate simulation environment for robotics, a previous work [23] develops an accurate actuator and simulate latency via thorough system identification. Another work [2] interleaves real-world roll-outs with simulation samples to adapt the simulation parameter distribution. In [24, 14, 19], domain randomization techniques are leveraged to produce a robust controller, training policies on a large variety of simulated scenarios. To further bridge the sim-to-real gap of dynamics, an intuitive approach is proposed to adapt to dynamics changing when conducting sim-to-real transfer [7]. Cycle-GAN is developed to provide sim-to-real translation [13, 20, 29]. For IRS, a number of prior works explore the importance of configurable simulators. RecoGym [21] supports environment configuration for sequential interaction but lacks the support for user state transition. An authoring simulation platform is also designed to mimic specific aspects of user behavior [11]. While previous works in IRS provide opportunity to create stylized environments, none of these address the gap between the simulation and real-world environment.

3 Problem Formulation

Let $\mathcal{U}$ be a set of users ($|\mathcal{U}| = m$) and $\mathcal{V}$ be a set of items ($|\mathcal{V}| = n$). The essence of interactive recommendation for user $u \in \mathcal{U}$ is a multi-step decision-making process, which can be naturally formulated as a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho, \gamma)$ where

- $\mathcal{S}$ is a continuous state space which captures the interaction characteristics of the user;
- $\mathcal{A}$ is a discrete action space including all available items, i.e., $\mathcal{A} = \mathcal{V}$;
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the probability of state transition, defining the user dynamics;
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, where $r(\boldsymbol{s}, \boldsymbol{a})$ is the immediate reward when performing action $\boldsymbol{a}$ at state $\boldsymbol{s}$;
- ρ is the initial user state distribution;
- γ is the discount rate which determines the present value of future rewards.

At each step t, the system selects an item $\boldsymbol{a}_t \in \mathcal{A}$ to recommend based on the interaction history, and then receives feedback (e.g., click or purchase behavior) as the reward. The target of the recommender system is to learn the optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ which can maximize the cumulative reward in the long run: $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{|\tau|} \gamma^t r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) \right]$, where $\tau = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \dots)$ represents an interaction trajectory obtained via $\boldsymbol{s}_0 \sim \rho$, $\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t)$, $\boldsymbol{s}_{t+1} \sim P(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$.

What we really desire is a policy that performs well in the real world, while the potentially expensive trial-and-error experimentation prohibits us to directly train the policy by interacting with the users. In other words, the real $\mathcal{M}_{\text{real}}$ is usually not accessible. To circumvent the issue, we desire to leverage available interaction history to construct a simulation environment $\mathcal{M}_{\text{sim}}$ from which we can train a policy that also achieves good performance in $\mathcal{M}_{\text{real}}$. In practice, we only have access to explicit rating feedback. Therefore, we consider the following problem:

Definition 3.1. Given a large amount of interaction history from the simulation environment $\mathcal{D}_{\text{sim}} = \{(u_i^S, v_j^S, y_{i,j}^S)_k\}_{k=1}^{N_{\text{sim}}}$ as well as a small fraction of data from the real environment $\mathcal{D}_{\text{real}} = \{(u_i^R, v_j^R, y_{i,j}^R)_k\}_{k=1}^{N_{\text{real}}}$ where $N_{\text{real}} \ll N_{\text{sim}}$, we want to acquire a policy π^* that achieves high rewards in the real environment $\mathcal{M}_{\text{real}}$ by only interacting with the simulator $\mathcal{M}_{\text{sim}}$.

4 Methodology

To bridge the gap between simulation and real world, the user dynamics and the item representations are two main components which need to be aligned. Thus, we propose an IRS consisting of two

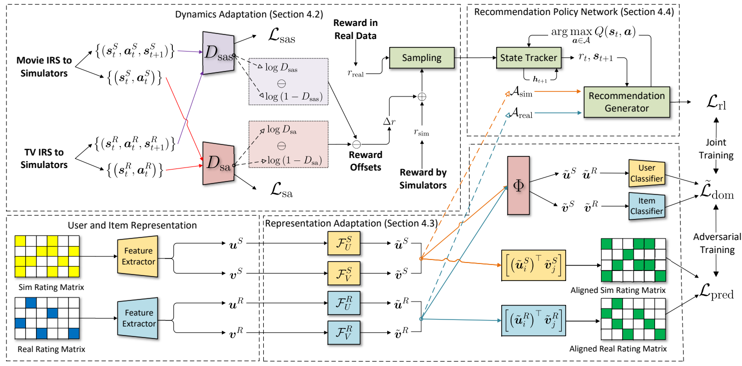


Figure 2: Our model mainly consists of the dynamics adaptation and representation adaptation module to bridge the sim-to-real gap. The movie IRS and TV IRS interact with the user simulators to collect the data trajectory to train the dynamics adaptation module in Figure 1. The representation adaptation, the recommender policy network and the dynamics adaptation modules are jointly trained.

adaptation methods, dynamics adaptation and representation adaptation. In the dynamics adaptation module, the simulation user trajectory data and the real user trajectory data are interleaved to train the dynamics classifiers $D_{\mathrm{sas}}, D_{\mathrm{sa}}$. We construct Δr from the output of those two classifiers and modify the simulation rewards to align with the real user dynamics. In the representation adaptation module, the raw representations of users and items, $\boldsymbol{u}^S, \boldsymbol{u}^R, \boldsymbol{v}^S, \boldsymbol{v}^R$, are extracted by the feature extractor and then mapped into sim-real invariant representations via adversarial training.

The training process can be divided into two stages: representation adaptation and joint adaptation. First, the representation adaptation stage aims to produce invariant item representations via adversarial supervised learning. Loss $\mathcal{L}_{\mathrm{dom}}$ and $\mathcal{L}_{\mathrm{pred}}$ are designed to learn the sim-real classifiers and to achieve better prediction accuracy respectively. By applying a gradient inverse layer Φ, we modify the min-max optimization objective of $\mathcal{L}_{\mathrm{dom}}$ and transform it into a new minimization objective with loss function $\tilde{\mathcal{L}}_{\mathrm{dom}}$. Second, in the process of RL-based interactive learning, the RL model learns behavioral policies with reward correction in dynamics adaptation while the representations updated with adversarial learning. Apart from $\tilde{\mathcal{L}}_{\mathrm{dom}}$ and $\mathcal{L}_{\mathrm{pred}}$ in updating user representations, the RL-based model is trained with its own loss $\mathcal{L}_{\mathrm{rl}}$ and reward correction guided by $\mathcal{L}_{\mathrm{sa}}$ and $\mathcal{L}_{\mathrm{sas}}$.

Dynamics Adaptation To capture the user preference based on the historical interaction, we can extract user states from the user's interacted items as well as the responses. However, user preference is usually unpredictable and user dynamics varies drastically in different recommendation tasks. This poses serious challenges to precisely model state transition for the simulator. In other words, we need to take state transition into account, narrowing user behavior difference between the simulator and the real world by calibrating P_{sim} with P_{real}. Inspired by the work in robotics [7], we propose to align the user dynamics in two environments from the perspective of probabilistic inference [15].

Specifically, the desired distribution over trajectories in the real-world environment is: $p_{\mathrm{real}}(\tau) \propto \rho\left(\prod_t^{|\tau|} P_{\mathrm{real}}\left(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\right)\right) \exp\left(\sum_t^{|\tau|} r_{\mathrm{real}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right)$, and $p_{\mathrm{sim}}(\tau)$ is the distribution over interaction trajectories in the simulation environment: $p_{\mathrm{sim}}(\tau) = \rho \prod_t^{|\tau|} P_{\mathrm{sim}}\left(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\right) \pi\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right)$.

If p_{sim} is close to p_{real}, policies learned by interacting with the simulator can achieve high rewards and behave under similar user dynamics in the real environment. Thus, we aim to minimize the distance between two distributions with the metric of KL divergence:

$$\min_\pi D_{\mathrm{KL}}(p_{\mathrm{sim}} \| p_{\mathrm{real}}) = -\mathbb{E}_{P_{\mathrm{sim}}}\left[\sum_t^{|\tau|} r_{\mathrm{sim}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) + \Delta r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})\right],$$

where $\Delta r\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}\right) \triangleq \log p_{\mathrm{real}}\left(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\right) - \log p_{\mathrm{sim}}\left(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\right)$.

From the above objective, we can reduce the difference of user dynamics between two environments via reward correction. In practice, Δr can be estimated by learning two classifiers D_{sas} and D_{sa}:

$$p_{\text{real}}\left(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\right) \propto \underbrace{p\left(\text{real} \mid \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}\right)}_{D_{\text{sas}}} / \underbrace{p\left(\text{real} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\right)}_{D_{\text{sa}}}.$$

Given trajectory data from the simulation environment $\tau_{\text{sim}} = \left\{\left(s_t^S, a_t^S, s_{t+1}^S\right)\right\}_{t=0}^{|\tau_{\text{sim}}|-1}$ and those from the real environment $\tau_{\text{real}} = \left\{\left(s_t^R, a_t^R, s_{t+1}^R\right)\right\}_{t=0}^{|\tau_{\text{real}}|-1}$, we learn D_{sas} and D_{sa} to predict whether the transitions come from the simulation or real data:

$$\begin{aligned}\mathcal{L}_{\text{sas}}\left(D_{\text{sas}}\right) = &- \mathbb{E}_{(s_t^R, a_t^R, s_{t+1}^R)\sim\tau_{\text{real}}}\left[\log D_{\text{sas}}(s_t^R, a_t^R, s_{t+1}^R)\right] \\ &- \mathbb{E}_{(s_t^S, a_t^S, s_{t+1}^S)\sim\tau_{\text{sim}}}\left[\log\left(1 - D_{\text{sas}}(s_t^S, a_t^S, s_{t+1}^S)\right)\right],\end{aligned}$$

$$\mathcal{L}_{\text{sa}}\left(D_{\text{sa}}\right) = -\mathbb{E}_{(s_t^R, a_t^R)\sim\tau_{\text{real}}}\left[\log D_{\text{sa}}(s_t^R, a_t^R)\right] - \mathbb{E}_{(s_t^S, a_t^S)\sim\tau_{\text{sim}}}\left[\log\left(1 - D_{\text{sa}}(s_t^S, a_t^S)\right)\right].$$

The intuition behind Δr is that, for the task of predicting whether a transition came from the simulation or real interaction data, how much better we can perform after observing $\boldsymbol{s}_{t+1}$. For transitions that are likely in the simulation environment but are unlikely in the real environment, $\Delta r < 0$, the agent is penalized for "exploiting" inaccuracies or discrepancies in the simulation task by taking these transitions. As a result, we can calibrate $\mathcal{M}_{\text{sim}}$ with $\tilde{r}_{\text{sim}} = r_{\text{sim}} + \alpha\Delta r$ to mimic the user dynamics in the real environment, where α controls the impact of reward offset. This leads to competitive policies for $\mathcal{M}_{\text{real}}$ with solely the practice under $\mathcal{M}_{\text{sim}}$.

Representation Adaptation Another issue that is critical when performing sim-to-real transfer is the discrepancy of item representation. We begin by extracting pretrained user and item representations via Singular Value Decomposition (SVD). Assume $\mathcal{D}_{\text{sim}}$ and $\mathcal{D}_{\text{real}}$ contain rating data from m^S users to n^S items and m^R users to n^R items respectively. We reconstruct both rating matrices $Y^S = \left[y_{i,j}^S\right], Y^R = \left[y_{i,j}^R\right]$ from $\mathcal{D}_{\text{sim}}, \mathcal{D}_{\text{real}}$ by $U^S, V^S = \text{SVD}(Y^S)$ and $U^R, V^R = \text{SVD}(Y^R)$, where $U^S = \left[\boldsymbol{u}_1^S, \cdots, \boldsymbol{u}_{m^S}^S\right]$ and $V^S = \left[\boldsymbol{v}_1^S, \cdots, \boldsymbol{v}_{n^S}^S\right]$ denote user and item features in the simulation environment. In the real environment, users and items are represented as $U^R = \left[\boldsymbol{u}_1^R, \cdots, \boldsymbol{u}_{m^R}^R\right]$ and $V^R = \left[\boldsymbol{v}_1^R, \cdots, \boldsymbol{v}_{n^R}^R\right]$ respectively. Given pretrained item embeddings $\boldsymbol{v}^S \in V^S$, $\boldsymbol{v}^R \in V^R$, two mappings $\mathcal{F}_V^S$ and $\mathcal{F}_V^R$ can be learned with a classifier D_V with the adversarial objective, where the mappings aim to produce invariant item representations from sim to real, while D_V tries to distinguish $\mathcal{F}_V^S(\boldsymbol{v}^S)$ from $\mathcal{F}_V^R(\boldsymbol{v}^R)$. The same procedures also apply for user embeddings with $\mathcal{F}_U^S$, $\mathcal{F}_U^R$ and D_U, leading to the mappings for both item and user representations: $\tilde{\boldsymbol{u}}^S = \mathcal{F}_U^S(\boldsymbol{u}^S)$, $\tilde{\boldsymbol{v}}^S = \mathcal{F}_V^S(\boldsymbol{v}^S)$, $\tilde{\boldsymbol{u}}^R = \mathcal{F}_U^R(\boldsymbol{u}^R)$, and $\tilde{\boldsymbol{v}}^R = \mathcal{F}_V^R(\boldsymbol{v}^R)$. When training, the adversarial learning objective is then:

$$\begin{aligned}&\min_{D_U, D_V} \max_{\mathcal{F}_U^S, \mathcal{F}_V^S, \mathcal{F}_U^R, \mathcal{F}_V^R} \mathcal{L}_{\text{dom}}\left(D_U, D_V, \mathcal{F}_U^S, \mathcal{F}_V^S, \mathcal{F}_U^R, \mathcal{F}_V^R\right) \\ &= -\sum_{i=1}^{m^R}\left[\log D_U\left(\tilde{\boldsymbol{u}}_i^R\right)\right] - \sum_{i=1}^{m^S}\left[\log\left(1 - D_U\left(\tilde{\boldsymbol{u}}_i^S\right)\right)\right] - \sum_{j=1}^{n^R}\left[\log D_V\left(\tilde{\boldsymbol{v}}_j^R\right)\right] - \sum_{j=1}^{n^S}\left[\log\left(1 - D_V\left(\tilde{\boldsymbol{v}}_j^S\right)\right)\right].\end{aligned} \tag{1}$$

To retain sufficient information, we introduce the prediction loss to reconstruct the rating patterns for two environments separately:

$$\min_{\mathcal{F}_U^S, \mathcal{F}_V^S, \mathcal{F}_U^R, \mathcal{F}_V^R} \mathcal{L}_{\text{pred}}\left(\mathcal{F}_U^S, \mathcal{F}_V^S, \mathcal{F}_U^R, \mathcal{F}_V^R\right) = \sum_{i=1}^{m^S}\sum_{j=1}^{n^S}\left\|\langle\tilde{\boldsymbol{u}}_i^S, \tilde{\boldsymbol{v}}_j^S\rangle - y_{i,j}^S\right\|_{\mathcal{O}} + \sum_{i=1}^{m^R}\sum_{j=1}^{n^R}\left\|\langle\tilde{\boldsymbol{u}}_i^R, \tilde{\boldsymbol{v}}_j^R\rangle - y_{i,j}^R\right\|_{\mathcal{O}}, \tag{2}$$

where $\|\cdot\|_{\mathcal{O}}$ is the norm $\|\cdot\|_{l^2}$ only on the observed data.

Recommendation Policy Network We apply deep RL algorithms to learn a recommendation policy by interacting with the simulator. We use the LSTM model [9] to distill the user state $\boldsymbol{s}_t$. At each time step t, the system recommends item $\boldsymbol{a}_t$ to the user and receives reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. To aggregate

No	Dataset			User			Item			Rating		
	Name	Sim	Real	Train		Test	Train		Test	Train		Test
				Sim-only	Shared	Real	Sim	Real	Real	Sim	Real	Real
1	Movielens	Adventure	Crime	24,948	200	22,633	652	508	508	2,865,694	14,568	1,699,513
2		Drama	Comedy	38,984	200	35,265	1,356	1,035	1,035	4,955,437	19,402	3,617,467
3		Thriller	Action	18,148	200	16,513	548	544	544	1,461,082	20,248	1,648,370
4	ANIME	Movie	TV	16,361	200	14,904	227	969	969	446,470	32,316	2,341,984
5		TV	OVA	15,072	200	13,744	969	145	145	2,502,212	2,825	186,889
6		OVA	Movie	2,353	200	2,297	145	227	227	77,539	12,058	136,340

Table 1: Statistics of all datasets used in our experimental evaluation.

historical user behaviors, the representation of the delivered item and the reward are concatenated as the input of LSTM. The state is updated recursively as $\boldsymbol{s}_{t+1}, \boldsymbol{h}_{t+1} = F([\boldsymbol{a}_t, r_t]; \boldsymbol{h}_t)$, where F denotes an LSTM cell which updates the hidden state $\boldsymbol{h}$ and outputs $\boldsymbol{s}$. We implement a vanilla Deep Q-Network (DQN), which aims to find the optimal policy via iterating the estimation of state-action value $Q(\boldsymbol{s}, \boldsymbol{a})$ parameterized by neural networks. The state-action value function estimates the long-term user engagement after acting $\boldsymbol{a}$ at state $\boldsymbol{s}$ and then following the learned policy, which leads to a deterministic recommendation strategy: $\pi(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}\in\hat{\mathcal{A}}} Q(\boldsymbol{s}, \boldsymbol{a})$, where $\hat{\mathcal{A}}$ denotes the action space with items that have already delivered masked to explicitly avoid repeated recommendation.

Joint Training for Joint Adaptation Furthermore, we propose a joint adaptation strategy by collaboratively optimizing the task-specified losses and finetuning the representation adaptation module under a unified objective. Different from [7] in which the simulation task and the real task are in the same environment, our sim-to-real setup contains two completely different recommendation tasks with no shared items. Thus, joint training is essential to adapt dynamics and representations simultaneously. Besides, joint training can further help to capture the shared dynamics patterns through backpropagation from downstream tasks, which is not viable by applying common domain adaptation methods like [28]. Finally, we formulate

$$\begin{aligned}\min_{D_{\mathrm{sas}}, D_{\mathrm{sa}}, F, Q, \mathcal{F}_V^S, \mathcal{F}_V^R} \max_{D_V} \mathcal{L}_{\mathrm{joint}} = {} & \mathcal{L}_{\mathrm{rl}}\left(F, Q, \mathcal{F}_V^S, \mathcal{F}_V^R\right) + \mathcal{L}_{\mathrm{sa}}\left(D_{\mathrm{sa}}, \mathcal{F}_V^S, \mathcal{F}_V^R\right) \\ & + \mathcal{L}_{\mathrm{sas}}\left(D_{\mathrm{sas}}, \mathcal{F}_V^S, \mathcal{F}_V^R\right) + \mathcal{L}_{\mathrm{dom}}\left(D_U, D_V, \mathcal{F}_U^S, \mathcal{F}_V^S, \mathcal{F}_U^R, \mathcal{F}_V^R\right),\end{aligned}$$

as the overall joint training objective.

5 Experiment

Datasets We conduct experiments on **Movielens-25M**[2] and **ANIME**[3]. The sim-real dataset split is across different genres. To split the datasets, we collect rating data from 90% of users shown in the real environment as the test set. Along the remaining users in the real environment, 20% of users are randomly selected as the validation set for hyperparameter tuning, while 200 other users are sampled for training. All ratings from the simulation data are also served as the training set. To evaluate the effectiveness of sim-to-real adaptation, we choose pairs of unrelated categories within the dataset for the simulation and real-world tasks. The statistics for the datasets are summarized in Table 1.

Metrics To evaluate the long-term performance, we use the average cumulative reward over each user session for each user in the real task as one metric. Additionally, we adopt three other metrics, Precision@T, Recall@T and F1@T, that are commonly used in traditional IRS tasks. Specifically, we set the length of each user session as $T = 32$. On Movielens-25M, the item ratings are ranged from 1.0 to 5.0 and those higher than 3.0 are regarded as the relevant items. On ANIME, the item ratings are ranged from 0.0 to 10.0 and those higher than 5.0 are regarded as the relevant items.

Baselines We evaluate the following baselines in our experiments. (i) **MF**: a simple matrix factorization method via conducting singular value decomposition on the rating patterns. When interacting with users, the model greedily selects the item with highest predicted rating to recommend in the real task. (ii) **DARec** [28]: a cross-domain model which regards items of simulation and real tasks are from two domains and trains a domain classifier via adversarial training to produce domain-invariant item representation. (iii) **DQN-R** [31]: a DQN-based method which learns by interacting with the simulator without any adaptation and recommends the item with highest Q-value when evaluation. To understand the importance of different components in our algorithm, we also compare our algorithm with three variants of our algorithm S2R-Rec. (i) **S2R-Rec w/o DynAda**: A variant of our

[2]`https://grouplens.org/datasets/movielens/`

[3]`https://www.kaggle.com/CooperUnion/anime-recommendations-database`

	Adventure (sim) → Crime (real)				Drama (sim) → Comedy (real)				Thriller (sim) → Action (real)			
	Reward	P@32	R@32	F1@32	Reward	P@32	R@32	F1@32	Reward	P@32	R@32	F1@32
MF	0.1554	0.2266	0.1032	0.1331	0.1325	0.1991	0.0729	0.0977	0.1671	0.2553	0.0883	0.1258
DARec	0.2325	0.3210	0.1530	0.1951	0.1572	0.2275	0.0876	0.1161	0.2140	0.3066	0.1098	0.1551
DQN-R	0.2909	0.4010	0.1927	0.2450	0.1879	0.2726	0.1051	0.1393	0.3002	0.4235	0.1527	0.2153
S2R-Rec w/o DynAda	0.3320	0.4507	0.2158	0.2756	0.2182	0.3093	0.1196	0.1585	0.3211	0.4513	0.1634	0.2300
S2R-Rec w/o RepAda	0.3103	0.4243	0.2016	0.2580	0.2181	0.3108	0.1207	0.1595	0.3109	0.4424	0.1602	0.2256
S2R-Rec w/o JntTrn	0.3363	0.4579	0.2208	0.2809	0.2264	0.3232	0.1269	0.1671	0.3336	0.4705	0.1697	0.2394
S2R-Rec	**0.3413**	**0.4654**	**0.2261**	**0.2867**	**0.2359**	**0.3370**	**0.1312**	**0.1734**	**0.3399**	**0.4777**	**0.1722**	**0.2430**

Table 2: Experimental results on three sim-2-real combinations from Movielens-25M.

	Movie (sim) → TV (real)				TV (sim) → OVA (real)				OVA (sim) → Movie (real)			
	Reward	P@32	R@32	F1@32	Reward	P@32	R@32	F1@32	Reward	P@32	R@32	F1@32
MF	0.1509	0.2464	0.0527	0.0839	0.0498	0.0974	0.2321	0.1217	0.1448	0.2568	0.1414	0.1745
DARec	0.2550	0.3576	0.0809	0.1277	0.0539	0.1039	0.2537	0.1306	0.2025	0.3292	0.1839	0.2260
DQN-R	0.2286	0.3453	0.0774	0.1225	0.0597	0.1125	0.2932	0.1428	0.2471	0.3940	0.2186	0.2700
S2R-Rec w/o DynAda	0.3194	0.4560	0.1045	0.1647	0.0695	0.1259	0.3340	0.1611	0.2590	0.4078	0.2273	0.2802
S2R-Rec w/o RepAda	0.2518	0.3881	0.0864	0.1371	0.0669	0.1201	0.3125	0.1529	0.2521	0.4000	0.2232	0.2747
S2R-Rec w/o JntTrn	0.3373	0.4818	0.1102	0.1738	0.0725	0.1299	0.3374	0.1655	0.2855	0.4449	0.2484	0.3062
S2R-Rec	**0.3396**	**0.4868**	**0.1122**	**0.1767**	**0.0756**	**0.1306**	**0.3493**	**0.1672**	**0.2905**	**0.4505**	**0.2529**	**0.3108**

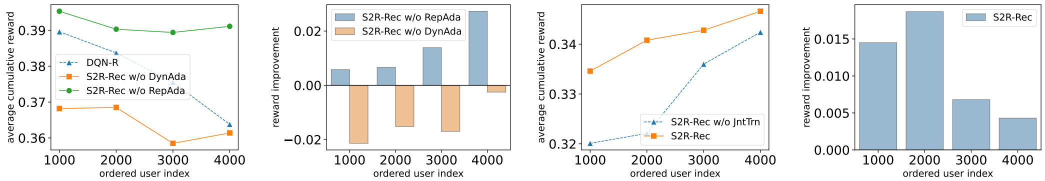
Table 3: Experimental results on three sim-2-real combinations from ANIME.

algorithm which only conducts representation adaptation via domain-adversarial training [8], while user dynamics is misaligned. (ii) **S2R-Rec w/o RepAda**: A variant that only adapt user dynamics with reward correction. (iii) **S2R-Rec w/o JntTrn**: A variant that performs both dynamics adaptation and representation adaptation, but in a separate procedure.

Implementation Details For MF, we use SVD to extract the user representations from the simulation training data and extract item representations from the real training data. The size of the representation vectors is $D_{\text{rep}} = 128$. For DARec, we use the same way as MF to first extract the original representations. Following the method in [28], we apply linear projections to the original representations and learn the invariant user and item representations by adversarial learning. DARec predicts the user ratings in the test data similar to MF but uses the adapted representations. For RL-based models, DQN-R, S2R-Rec and its variants, the agent is trained on the batch size of $N_{\text{batch}} = \{32, 64\}$. The batches are randomly sampled from both simulation and real data. Since the size of real data is significantly smaller, we sample from the real data in every $K_{\text{interval}} = \{2, 3, 4\}$ iterations. Except for S2R-Rec, for methods involving representation adaptation, we use the same item representations from DARec. Otherwise, we use the same item representations from MF. For methods involving dynamics adaptation, the coefficient for Δr is set to $\alpha_{\text{dyn}} = \{0.6, 0.7, 0.8, 0.9\}$. For S2R-Rec, the item representations are fine-tuned in the training of the IRS. Without the joint training, the item representations are fixed and the same as in DARec.

Research Questions We seek to answer the following research questions: **RQ1**: How does S2R-Rec perform compared with the baseline methods? **RQ2**: How is the effectiveness of dynamics adaptation for aligning sim-to-real user behaviour differences? **RQ3**: How is the effectiveness of joint learning to achieve representation-dynamics collaborative adaptation?

Performance of S2R-Rec (RQ1) We compare our approach to the baselines. The results are reported in Table 2 and 3. There are several observations. First, comparing S2R-Rec w/o RepAda with DQN-R, the improvements brought by dynamics adaptation are consistent. On the Movielens-25M dataset, improvement of reward is 6.67%, 16.07%, and 3.56% on Adventure and Crime, Drama and Comedy, Thriller and Action respectively. Meanwhile, on the ANIME dataset, the S2R-Rec w/o RepAda achieves improvements of 10.15%, 12.06% and 2.02% on Movie and TV, TV and OVA, OVA and Movie. Second, shared item features can be transferred to better identify some shared user preferences to the items in the sim-to-real scenario. Consequently, S2R-Rec w/o DynAda performs 14.13%, 16.13%, and 6.96% on reward improvement as expected on the Movielens-25M. In addition, the improvement of the aligning item representations is 39.72%, 16.42% and 4.82% on the ANIME dataset. By learning shared item features related to user preferences in sim-to-real environments, the effects of dynamics adaptation can be further boosted. Compared with DQN-R, S2R-Rec w/o JntTrn achieves improvement 15.61%, 20.49%, and 11.13% in Movielens, while performs 47.55%, 21.44% and 15.54% more than DQN-R on reward. Third, the joint training helps to better extract item features relevant to the dynamics patterns existing in both simulation and real recommendation processes. We observe further improvements, S2R-Rec performs 17.32%, 25.55%, and 13.22% (48.56%, 26.63%, and 17.56%) better than DQN-R on the Movielens-25M (ANIME) dataset.



(a) (b) (c) (d)

Figure 3: The effectiveness of the dynamics adaptation is in (a) (b). Figure (b) shows the improvement of S2R-Rec w/o DynAda, S2R-Rec w/o RepAda, over DQN-R. The effectiveness of the joint adaptation is in (c) (d). Figure (d) shows the improvement of S2R-Rec over S2R-Rec w/o JntTrn.

Effectiveness of the Dynamics Adaptation (RQ2) To evaluate the dynamics adaptation in aligning sim-to-real user behaviors, we conduct this experiment to recommend the same item pool but to let the system adapt between different groups of users. We choose movies in the category Thriller from the MovieLens-25M dataset as the item pool. By the clustering method k-means [18], we categorize all users' first 32 rating behaviors to 10 clusters, indicating 10 different user groups with different patterns of interaction. We choose the cluster with the most users as the user group existing in the real-world task. By calculating the euclidean distances between the centroids of all other clusters and the centroid of the chosen one, we are able to measure the distances of user behaviors between clusters. We order the clusters by the distances from small to large and then get the ordered user indexes, which indicate the sim-to-real gaps of dynamics that increase.

Compared with the DQN-R in Figure 3(b), in general, dynamics adaptation plays a part in S2R-Rec w/o RepAda, while the representation adaptation in S2R-Rec w/o DynAda has a negative impact. As we can observe in Figure 3(a), from the group 1 to group 4, with the sim-to-real gaps increasing, the performances of DQN-R and S2R-Rec w/o DynAda decrease constantly. Remarkably, S2R-Rec w/o RepAda always achieves higher performance than DQN-R, and is less affected by the sim-to-real gaps increasing. It implies the representation alignment is no longer needed in S2R-Rec w/o DynAda and the user dynamics could be catch up with the dynamics adaptation as S2R-Rec w/o RepAda. Furthermore, from the group 1 to group 4, both the dynamics adaptation achieves higher improvements while the performance of DQN-R declining. It is largely explained by both representation adaptation and dynamics adaptation align the user between different groups.

Effectiveness of the Joint Adaptation (RQ3) We further evaluate the effectiveness of the joint adaptation. We choose the same sim-to-real tasks to adapt from the category Thriller to the category Action on the MovieLens-25M dataset. Also with the clustering method k-means, we divide all users' first 32 rating behaviors in both categories into 10 clusters. We measure each user's sim-to-real difference by calculating the euclidean distance between each user's trajectory centroids in those two categories. By ordering this distance of each user from large to small, we are able to choose the first 4000 users and equally assign them to 4 experiments in this order. In this way, we simulate 4 experiments with descending difficulties in sim-to-real adaptation.

We observe significant improvements by applying joint training in S2R-Rec. Moreover, the improvement with the joint train in Figure 3(d) reveals the effective of S2R-Rec w/o JntTrn and S2R-Rec are related to the distance of users. When the distance of user behaviours in sim-to-real is closer, S2R-Rec provides improvement ratio of the 4th group reduces to 1.241%. It reflects that representation initialized with S2R-Rec w/o DynAda is sufficient in providing the information when the user's distance is small. On the contrary, when the distance of user is greater in the first 1000 users, the joint learning in collaboration adaptation plays the leading role while the improvement achieves 4.52%. It is because that from group 2 to group 3, as the user representation is getting closer, S2R-Rec w/o JntTrn receives greater improvement with respect to reward.

6 Conclusion

In this paper, we present a unified framework towards sim-to-real interactive recommender system (IRS). Concretely, we first devise to calibrate the difference between state transitions of the simulation and real environment via reward correction. Next, we align item representations to further remove discrepancy. These two stages of adaptation are then unified via a joint optimization target, which further boosts the performance of our proposed approach. By bridging the sim-to-real gap in IRS, we can promisingly perform policy training for IRS by only interacting with the simulators.

References

[1] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. *arXiv preprint arXiv:2011.03807*, 2020.

[2] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[3] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3312–3320, 2019.

[4] 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.

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

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

[7] Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. In *International Conference on Learning Representations*, 2021.

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 368–377, 2018.

[11] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847*, 2019.

[12] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. 2019.

[13] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[14] Manuel Kaspar, Juan D Muñoz Osorio, and Jürgen Bock. Sim2real transfer for reinforcement learning without dynamics randomization. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4383–4388. IEEE, 2020.

[15] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

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

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

[18] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[19] Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Data-efficient domain randomization with bayesian optimization. *IEEE Robotics and Automation Letters*, 6(2):911–918, 2021.

[20] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

[21] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720*, 2018.

[22] Timo Stoffregen, Cedric Scheerlinck, Davide Scaramuzza, Tom Drummond, Nick Barnes, Lindsay Kleeman, and Robert Mahony. Reducing the sim-to-real gap for event cameras. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pages 534–549. Springer, 2020.

[23] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[24] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[25] Huazheng Wang, Qingyun Wu, and Hongning Wang. Factorization bandits for interactive recommendation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[26] Qing Wang, Chunqiu Zeng, Wubai Zhou, Tao Li, S Sitharama Iyengar, Larisa Shwartz, and Genady Ya Grabarnik. Online interactive collaborative filtering using multi-armed bandit with dependent arms. *IEEE Transactions on Knowledge and Data Engineering*, 31(8):1569–1580, 2018.

[27] Tong Yu, Yilin Shen, and Hongxia Jin. A visual dialog augmented interactive recommender system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 157–165, 2019.

[28] Feng Yuan, Lina Yao, and Boualem Benatallah. Darec: Deep domain adaptation for cross-domain recommendation via transferring rating patterns. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4227–4233. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[29] Qiang Zhang, Tete Xiao, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Learning cross-domain correspondence for control with dynamics cycle-consistency. In *International Conference on Learning Representations*, 2021.

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

[31] 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.

[32] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 179–188, 2020.

[33] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 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*, pages 2810–2818, 2019.