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.
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 [20, 1]. 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, 50, 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 $U$ be a set of users ($|U| = m$) and $V$ be a set of items ($|V| = n$). The essence of interactive recommendation for user $u \in U$ is a multi-step decision-making process, which can be naturally formulated as a MDP $M = (S, A, P, r, \rho, \gamma)$ where

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

At each step $t$, the system selects an item $a_t \in A$ 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 $\pi^* : S \rightarrow A$ which can maximize the cumulative reward in the long run: $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}_{\tau \sim \tau} \left[ \sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right]$, where $\tau = (s_0, a_0, s_1, \ldots)$ represents an interaction trajectory obtained via $s_0 \sim \rho, a_t \sim \pi(\cdot | s_t), s_{t+1} \sim P(\cdot | s_t, a_t)$.

What we really desire is a policy that performs well in the real world, but the potentially expensive trial-and-error experimentation prohibits us to directly train the policy by interacting with the users. In other words, the real $M_{real}$ is usually not accessible. To circumvent the issue, we desire to leverage available interaction history to construct a simulation environment $M_{sim}$ from which we can train a policy that also achieves good performance in $M_{real}$. 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 $D_{sim} = \{(u^S_i, v^S_j, y^S_{i,j})\}_{k=1}^{N_{sim}}$ as well as a small fraction of data from the real environment $D_{real} = \{(u^R_i, v^R_j, y^R_{i,j})\}_{k=1}^{N_{real}}$, where $N_{real} \ll N_{sim}$, we want to acquire a policy $\pi^*$ that achieves high rewards in the real environment $M_{real}$ by only interacting with the simulator $M_{sim}$.

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
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 $D_{naa}, D_{an}$. We construct $\Delta r$ 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, $u^S$, $u^R$, $v^S$, $v^R$, 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 $L_{d_{dom}}$ and $L_{d_{pred}}$ are designed to learn the sim-real classifiers and to achieve better prediction accuracy respectively. By applying a gradient inverse layer $\Phi$, we modify the min-max optimization objective of $L_{d_{dom}}$ and transform it into a new minimization objective with loss function $\hat{L}_{d_{dom}}$. 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 $\hat{L}_{d_{dom}}$ and $L_{d_{pred}}$ in updating user representations, the RL-based model is trained with its own loss $L_{rl}$ and reward correction guided by $L_{naa}$ and $L_{an}$.

**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: $p_{real}(\tau) \propto \rho \left( \prod_{t}^{\tau} P_{real}(s_{t+1} \mid s_t, a_t) \right) \exp \left( \sum_{t}^{\tau} r_{real}(s_t, a_t) \right)$, and $p_{sim}(\tau)$ is the distribution over interaction trajectories in the simulation environment: $p_{sim}(\tau) = \rho \prod_{t}^{\tau} P_{sim}(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t)$.

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:

$$\min_{\pi} D_{KL}(p_{sim} \| p_{real}) = -\mathbb{E}_{p_{sim}} \left[ \sum_{t}^{\tau} r_{sim}(s_t, a_t) + \Delta r(s_t, a_t, s_{t+1}) \right],$$

where $\Delta r(s_t, a_t, s_{t+1}) \triangleq \log p_{real}(s_{t+1} \mid s_t, a_t) - \log p_{sim}(s_{t+1} \mid s_t, a_t)$.

From the above objective, we can reduce the difference of user dynamics between two environments via reward correction. In practice, $\Delta r$ can be estimated by learning two classifiers $D_{naa}$ and $D_{ana}$:
The intuition behind $\Delta r$ 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 $s_{t+1}$. 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 $M_{\text{sim}}$ with $\alpha_{\text{sim}} = \alpha_{\text{real}} + \alpha \Delta r$ to mimic the user dynamics in the real environment, where $\alpha$ controls the impact of reward offset. This leads to competitive policies for $M_{\text{real}}$ with solely the practice under $M_{\text{sim}}$.

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 $D_{\text{sim}}$ and $D_{\text{real}}$ 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 $Y^S = [y_{ij}^S], Y^R = [y_{ij}^R]$ from $D_{\text{sim}}, D_{\text{real}}$ by $U^S, V^S = \text{SVD}(Y^S)$ and $U^R, V^R = \text{SVD}(Y^R)$, where $U^S = [u_1^S, \ldots, u_{m^S}^S]$ and $V^S = [v_1^S, \ldots, v_{n^S}^S]$ denote user and item features in the simulation environment. In the real environment, users and items are represented as $U^R = [u_1^R, \ldots, u_{m^R}^R]$ and $V^R = [v_1^R, \ldots, v_{n^R}^R]$ respectively. Given pretrained item embeddings $v^S \in V^S, v^R \in V^R$, two mappings $F^S_U$ and $F^R_U$ 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 $F^S_U(v^S)$ from $F^R_U(v^R)$. The same procedures also apply for user embeddings with $F^S_V, F^R_V$ and $D_V$, leading to the mappings for both item and user representations: $\tilde{u}^S = F^S_U(u^S), \tilde{u}^R = F^R_U(u^R)$, and $\tilde{v}^S = F^S_V(v^S), \tilde{v}^R = F^R_V(v^R)$. When training, the adversarial learning objective is then:

$$\min_{D_U, D_V, F^S_U, F^R_U} \max_{F^S_V, F^R_V} \mathcal{L}_{\text{dom}}(D_U, D_V, F^S_U, F^S_V, F^R_U, F^R_V)$$

$$\sum_{i=1}^{m^R} \log D_U(\tilde{u}^R_i) - \sum_{i=1}^{m^S} \log (1 - D_U(\tilde{u}^S_i)) - \sum_{j=1}^{n^R} \log D_V(\tilde{v}^R_j) - \sum_{j=1}^{n^S} \log (1 - D_V(\tilde{v}^S_j)) \right).$$

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

$$\min_{F^S_U, F^S_V, F^R_U, F^R_V} \mathcal{L}_{\text{pred}}(F^S_U, F^S_V, F^R_U, F^R_V) = \sum_{i=1}^{m^S} \sum_{j=1}^{n^S} \| (\tilde{u}^S_i, \tilde{v}^S_j) - y_{i,j}^S \|_\Theta + \sum_{i=1}^{m^R} \sum_{j=1}^{n^R} \| (\tilde{u}^R_i, \tilde{v}^R_j) - y_{i,j}^R \|_\Theta,$$

where $\| \cdot \|_\Theta$ is the norm $\| \cdot \|_2$ 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 $s_t$. At each time step $t$, the system recommends item $a_t$ to the user and receives reward $r_t = r(s_t, a_t)$. To aggregate
Table 1: Statistics of all datasets used in our experimental evaluation.

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Dataset</th>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sim</td>
<td>Real</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Train</td>
<td>Test</td>
<td>Train</td>
<td>Test</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Sim-only</td>
<td>Shared</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

historical user behaviors, the representation of the delivered item and the reward are concentrated as the input of LSTM. The state is updated recursively as $s_{t+1}, h_{t+1} = F([a_t, r_t]; h_t)$, where $F$ denotes an LSTM cell which updates the hidden state $h$ and outputs $s$. We implement a vanilla Deep Q-Network (DQN), which aims to find the optimal policy via iterating the state-action value $Q(s, a)$ parameterized by neural networks. The state-action value function estimates the long-term user engagement after acting $a$ at state $s$ and then following the learned policy, which leads to a deterministic recommendation strategy: $\pi(s) = \arg\max_{a \in A} Q(s, a)$, where $A$ 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 [2] 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

$$\min_{D_{sa}, D_{sr}, F, Q, F_S, F_V, F_R} \max_{D_V} \mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{rl}}(F, Q, F_S, F_V, F_R) + \mathcal{L}_{\text{na}}(D_{sa}, F_S, F_V, F_R)$$

$$+ \mathcal{L}_{\text{ras}}(D_{sr}, F_S, F_V, F_R) + \mathcal{L}_{\text{dom}}(D_U, D_V, F_S, F_V, F_R),$$

as the overall joint training objective.

5 Experiment

**Datasets**

We conduct experiments on MovieLens-25M and ANIME. 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

---

https://grouplens.org/datasets/movielens/
https://www.kaggle.com/CooperUnion/anime-recommendations-database
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 $D_{rep} = 128$. For DARRec, 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. DARRec 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 $N_{batch} = \{32, 64\}$. 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 $K_{interval} = \{2, 3, 4\}$ iterations. Except for S2R-Rec, for methods involving representation adaptation, we use the same item representations from DARRec. Otherwise, we use the same item representations from MF. For methods involving dynamics adaptation, the coefficient for $\Delta r$ is set to $\alpha_{dyn} = \{0.6, 0.7, 0.8, 0.9\}$. 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 DARRec.

### 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%, 25.50%, and 13.22% (48.56%, 26.63%, and 17.56%) better than DQN-R on the Movielens-25M (ANIME) dataset.

![Table 2: Experimental results on three sim-2-real combinations from Movielens-25M.](image)

<table>
<thead>
<tr>
<th>Movie (sim)</th>
<th>TV (real)</th>
<th>OVA (real)</th>
<th>Movie (real)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MF</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward</td>
<td>P@32</td>
<td>R@32</td>
<td>F1@32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DARRec</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward</td>
<td>P@32</td>
<td>R@32</td>
<td>F1@32</td>
</tr>
<tr>
<td></td>
<td>0.2509</td>
<td>0.1039</td>
<td>0.2537</td>
</tr>
<tr>
<td></td>
<td>0.0974</td>
<td>0.1217</td>
<td></td>
</tr>
<tr>
<td><strong>DQN-R</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward</td>
<td>P@32</td>
<td>R@32</td>
<td>F1@32</td>
</tr>
<tr>
<td></td>
<td>0.0997</td>
<td>0.2321</td>
<td>0.1217</td>
</tr>
<tr>
<td></td>
<td>0.2359</td>
<td>0.1332</td>
<td></td>
</tr>
<tr>
<td><strong>S2R-Rec w/o DynAda</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward</td>
<td>P@32</td>
<td>R@32</td>
<td>F1@32</td>
</tr>
<tr>
<td></td>
<td>0.0953</td>
<td>0.2932</td>
<td>0.1428</td>
</tr>
<tr>
<td></td>
<td>0.2471</td>
<td>0.3940</td>
<td></td>
</tr>
<tr>
<td><strong>S2R-Rec w/o RepAda</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward</td>
<td>P@32</td>
<td>R@32</td>
<td>F1@32</td>
</tr>
<tr>
<td></td>
<td>0.0974</td>
<td>0.2932</td>
<td>0.1428</td>
</tr>
<tr>
<td></td>
<td>0.2471</td>
<td>0.3940</td>
<td></td>
</tr>
<tr>
<td><strong>S2R-Rec w/o JntTrn</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward</td>
<td>P@32</td>
<td>R@32</td>
<td>F1@32</td>
</tr>
<tr>
<td></td>
<td>0.0953</td>
<td>0.2932</td>
<td>0.1428</td>
</tr>
<tr>
<td></td>
<td>0.2471</td>
<td>0.3940</td>
<td></td>
</tr>
</tbody>
</table>
We observe significant improvements by applying joint training in S2R-Rec. Moreover, the improvement 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 promingly perform policy training for IRS by only interacting with the simulators.
References


