Reinforcement Learning with a Disentangled Universal Value Function for Item Recommendation

Kai Wang,[1] Zhene Zou,[1] Qilin Deng,[1] Runze Wu,[1,*] Jianrong Tao,[1]
Changjie Fan,[1] Liang Chen,[2] Peng Cui[3]
[1]Fuxi AI Lab, NetEase Inc., Hangzhou, China
[2]School of Data and Computer Science, Sun Yat-sen University, China
[3]Tsinghua University, China
{wangkai02,zouzhene,dengqilin,wurunze1,hztaojianrong,fanchangjie}@corp.netease.com,
chenliang6@mail.sysu.edu.cn, cuip@tsinghua.edu.cn

Abstract

In recent years, there are great interests as well as challenges in applying reinforcement learning (RL) to recommendation systems (RS). In this paper, we summarize three key practical challenges of large-scale RL-based recommender systems: massive state and action spaces, high-variance environment, and the unspecific reward setting in recommendation. All these problems remain largely unexplored in the existing literature and make the application of RL challenging.

We develop a model-based reinforcement learning framework with a disentangled universal value function, called GoalRec. Combining the ideas of world model (model-based), value function estimation (model-free), and goal-based RL, we propose a novel model-based value function formalization. It can generalize to various goals that the recommender may have, and disentangle the stochastic environmental dynamics and high-variance reward signals accordingly. As a part of the value function, free from the sparse and high-variance reward signals, a high-capacity reward-irrelevant world model is trained to simulate complex environmental dynamics under a certain goal. Based on the predicted environmental dynamics, the disentangled universal value function is related to the user's future trajectory instead of a monolithic state and a scalar reward. We demonstrate the superiority of GoalRec over previous approaches in terms of the above three practical challenges in a series of simulations and a real application.

Introduction

With the recent tremendous development of reinforcement learning (RL), there has been increasing interest in adopting RL for recommendations (Shani, Heckerman, and Brafman 2005; Hu et al. 2018; Chen et al. 2018). The RL-based recommender systems treat the recommendation process as a sequential interaction between the user (environment) and the recommendation agent (RL agent). And a part of user feedback (e.g., user purchase) is regarded as reward signals. The RL-based recommender systems can achieve two key advantages: (i) the recommendation agent can explore and exploit extremely sparse user-item feedback through limited user-agent interactions; (ii) the best strategy is to maximize users' overall long-term satisfaction without sacrificing the recommendations' short-term utility.

While RL has shown considerable success in games (Mnih et al. 2013; Colas et al. 2018) and robotics (Lillicrap et al. 2015; Andrychowicz et al. 2017), large-scale deployment of RL in real-world applications has proven challenging (Dulac-Arnold, Mankowitz, and Hester 2019). Compared to other machine learning methods, deep reinforcement learning has a reputation for being data-hungry and is subject to instability in its learning process (Henderson et al. 2018). In this paper, we first summarize three key challenges when applying reinforcement learning to a large-scale recommender system:

- **Massive state and action spaces.** In an industrial setting, the dimension of user features (state space) is extremely large, and the item is usually represented by high-dimensional item features (action space). However, most model-free RL methods as well as RL-based RS methods (Shi et al. 2019; Hu et al. 2018) in the literature learn from reward signals directly and often only use small neural networks with few parameters. As discussed in Ha and Schmidhuber (2018), the famous "credit assignment problem" and low-quality reward signals are the two main bottlenecks that make it hard for RL algorithms to learn millions of weights of a large model.
- **High-variance environment.** Different from static gym environments, real-world recommendation environments are commonly high-variance and uncertain. On the one hand, the feedback reward of users is usually sparse (e.g., 0.2% conversion rate) and unbalanced (e.g., a wide range of deal price). On the other hand, the random page jumps of users make the calculation of expected rewards difficult. These difficulties lead to a high-variance and biased estimation of the expected reward, which probably misleads the reinforcement learning towards poor performance.
- **Unspecific reward setting in recommendation.** Unlike reward-given environments, there is no specific reward setting given in real applications. Actually, it is unclear how to do event-based reward shaping (assigning reward to user view, click, exit, etc.) to maximize business met-

[*]Corresponding author.

rics (stay time, click-through rate, etc.) as well as alleviate the problem of credit assignment. It is common that several agents with different reward settings are deployed online simultaneously. The RL agent should be able to generalize over different reward settings and able to learn from the experiences generated by other policies, even the experiences generated by myopic supervised based recommenders.

In this paper, we propose a novel reinforcement learning framework called GoalRec, which takes special care of the above mentioned three practice difficulties with a disentangled universal value function:

- **Handling massive state and action spaces.** Complex real applications call for specially-designed RL models which are data-efficiency and high-capacity. Motivated by the observation that the rich and temporally dense multidimensional supervision is available in recommender systems, we resort to a recently developed model-based RL technique, World Model (Ha and Schmidhuber 2018; Feinberg et al. 2018). World model is self-supervised, high-capacity, irrelevant to environmental rewards, and only predicts how the states and actions would evolve afterwards. Different from previous one-step world model methods which only considers the next state, our RS-specific world model is designed to reconstruct the "measurement" of user's long-term future trajectories under a certain goal g.
- **Handling high-variance environment.** Previous value-based RL algorithms (Schulman et al. 2017; Mnih et al. 2016) approximate value functions directly without explicitly handling the coupling of environmental dynamics and reward signals. In contrast to the coupling manner, we develop a method of decoupling state transitions and reward functions in value function estimation, i.e., we incorporate the powerful world model into value function estimation. This disentangled formalization distinguishes the uncertainty of state transitions from the variance of reward signals, and thus helping to stabilize the value estimation process. Note that the standard model-based planning technique, Model Predictive Control (Hafner et al. 2018; Ke et al. 2019), is time-consuming and not practicable for online recommender systems.
- **Handling unspecific reward setting.** The problem setting that different reward settings with the same environmental dynamics is well studied in the goal-based RL domain. In this paper, we borrow the idea of goal-based RL by dealing with a more general setting of vectorial rewards and parameterized goals. The policy implied in users' trajectories are represented by goal vectors, and the environmental reward can be defined as the "distance" between current state and agent's desired goal. Specifically, we extend the disentangled value function to both states and goals by using universal value function (UVF). By universal, it means that the value function can learn from the experiences generated by other goals, and generalize to any goal g that the recommender may have. To the best of our knowledge, it is the first attempt to think multi-step RS problems in the lens of goal-based RL.

We note that GoalRec is a specially-designed unified solution rather than a trivial combination of existing techniques. The decoupling of stochastic environmental dynamics and low-quality reward signals makes the training of high-capacity RL models possible. As the cornerstone of the whole algorithm, the world model, which trained by temporally dense supervisory signals, can effectively alleviate the optimization issues caused by the problems mentioned above. Instead of employing time-consuming model predictive control, we then incorporate world model into value function estimation. Recall that the value function is policy-dependent and concerns the expected cumulative reward over trajectories, that is why the goal-based RL and trajectory-based world model are technically necessary.

Background

Reinforcement Learning

Consider a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0, \gamma, T \rangle$, defined with a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, the transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\in[0,1]}$, the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the initial state distribution $\rho_0 : \mathcal{S} \to \mathbb{R}_{\in[0,1]}$, the discounted factor $\gamma \in [0, 1]$, and the finite horizon T. An agent interacts with the MDP at discrete time steps by performing its policy $\pi : \mathcal{S} \to \mathcal{A}$, generating a trajectory of states and actions, $\tau = \tau_{0:T} = (s_0, a_0, \ldots, s_T, a_T)$, where $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi(s_t)$ and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$. The objective of the agent is to maximize the expected cumulative discounted reward, denoted by $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t | \pi\right]$ where $r_t = \mathcal{R}(s_t, a_t)$.

Universal Value Function Approximation

Consider for example the case where the agent's goal is described by a single desired state: it is clear that there is just as much similarity between the value of nearby goals as there is between the value of nearby states. A sufficiently expressive function approximator can in principle identify and exploit structure across both s and g. By universal, it means that the value function can generalize to any goal g in a set G of possible goals.

Specifically, for any goal $\mathrm{g} \in \mathcal{G}$, we define a pseudo-reward function $R_{\mathrm{g}}(s, a, s')$ and a pseudo-discount function $\gamma_{\mathrm{g}}(s)$. For any policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ and each g, and under some technical regularity conditions, we define a general value function that represents the expected cumulative pseudo-discounted future pseudo-return, i.e.,

$$V_{\mathrm{g},\pi}(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} R_{\mathrm{g}}\left(s_{t+1}, a_t, s_t\right) \prod_{k=0}^{t} \gamma_{\mathrm{g}}\left(s_k\right) | s_0 = s\right] \quad (1)$$

where the actions are generated according to π, as well as an state-action value function:

$$Q_{\mathrm{g},\pi}(s, a) := \mathbb{E}_{s'}\left[R_{\mathrm{g}}\left(s, a, s'\right) + \gamma_{\mathrm{g}}\left(s'\right) \cdot V_{\mathrm{g},\pi}\left(s'\right)\right] \quad (2)$$

Related Work

In the RL-based RS domain, past efforts mainly focused on item-list recommendation (Zhao et al. 2017, 2018; Huzhang et al. 2020; Ie et al. 2019), simulation environment construction (Chen et al. 2018; Shi et al. 2019; Bai, Guan, and Wang 2019), variance reduction (Hu et al. 2018; Chen et al. 2019) and long-term reward modeling (Zou et al. 2019; Zheng et al. 2018). Few works deal with the critical practice challenges mentioned above. To our best knowledge, the only existing research on world model-based RS is the Pseudo Dyna-Q conducted by Zou et al. (2020), in which the static simulation environment is replaced by a constantly updated world model and the overall learning process is the same as the vanilla Dyna-Q (Sutton 1991). Not only are the problems concerned different between our work and Pseudo Dyna-Q, but also the use of world model: (i) our world model is trained using the whole trajectories of users while theirs only use users' immediate responses; (ii) our world model is a part of the value function while theirs acts as a pseudo sample provider.

Outside of the RL-based RS domain, the ideas of combining model-based learning and model-free value function estimation are also proposed in Successor Feature (Borsa et al. 2018) and Model-Based Value Expansion (Feinberg et al. 2018). The main difference is that our value function is based on the representation of user's long-term future trajectory rather than the prediction of the next state.

Proposed Approach

In this section, we first provide an abstract description of the proposed disentangled universal value function and prove a lower-bound approximation of the Q-function. Then, we discuss the specific design of value function for recommender systems and the corresponding world model approximator–a scalable Dueling Deep Q-Network (DDQN)-like model. An illustration of the GoalRec is shown in the right part of Figure 1. The overall algorithm is summarized in Algorithm 1 in Appendix A.

Disentangled Formalization

Most existing world model studies in the RL community focus on pixel-based game environments and are not suitable for recommender systems. Instead of employing generative models to reconstruct the whole state space, we consider the "measurement" of user trajectories. Given a trajectory $\tau_{0:t} = (s_0, a_0, ..., s_t, a_t)$ with $t \geq 0$, we consider a measurement function f that $\mathrm{m} = f(\tau_{0:t})$, and then introduce the following definitions:

Definition 1 *The measurement* m *of trajectory* τ *is a set of sufficient statistic indicators that are predictive of user response (long-term and short-term) or self-predictive (i.e., summarizes user history in a way that renders the implied environment dynamics).*

Definition 2 *Given the measurement function* f*, the function* M *denotes the expected measure of the future trajectory for performing action* $a \in \mathcal{A}$ *in state* $s \in \mathcal{S}$*, then following a policy* π*:*

$$\begin{aligned} M^\pi(s,a) &= \mathbb{E}[\mathrm{m}|s_0 = s, a_0 = a; \pi] \\ &= \mathbb{E}\big[f(\tau_{0:t})|s_0 = s, a_0 = a; \pi\big]. \end{aligned} \tag{3}$$

The future measurement predictor (i.e., world model) M is irrelevant to reward and only predicts how the states and actions would evolve afterwards. It can be seen as a partial dynamics model of the environment which depends on the policy π. Free from the "credit assignment problem" and the low-quality reward signals, a high-capacity world model M with millions of weights can be trained effectively to capture the complex environment dynamics. We note that this improvement is crucial for real complex applications such as recommender systems.

Given the definition of measurement m, it is straightforward to define a return function U that maps the expected measurement $m = f(\tau_{0:t})$ to the cumulative discounted reward for any trajectory $\tau_{0:t}$. Here, we provide a lower-bound approximation of the Q-function by proving a lemma:

Lemma 1 *Given a policy* π*, the following lower bound of the Q-function holds for all* $s \in \mathcal{S}$ *and* $a \in \mathcal{A}$*, when function* U *is convex:*

$$\begin{aligned} &Q^\pi(s,a) \geq U\big(M^\pi(s,a)\big), \\ \textit{where} \quad &U(\mathrm{m}) = r_0 + \gamma r_1 + \cdots + \gamma^t r_t \end{aligned} \tag{4}$$

The proof can be easily obtained with *Jensen's Inequality* by exchanging the expectation and function. Lemma 1 provides a lower-bound approximation of the Q-function as a composite function of U and M. When U is a linear function, the equality guarantees that we can also obtain the optimal policy through optimizing the composite function. Since the input $M^\pi(s,a)$ can be non-linear, it still ensures the representation ability of the composite function with a linear U.

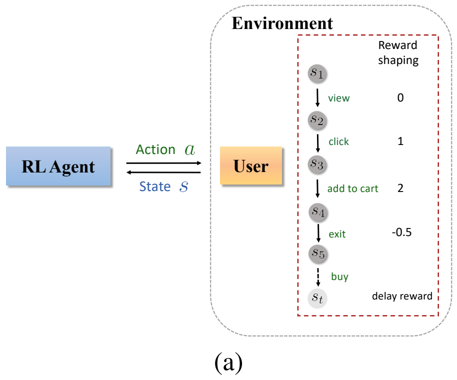
Now, we extend the disentangled value function to both states and goals by using a universal value function. And the above lemma still holds for a fixed goal g. Specifically, we replace the vanilla Q-function with a goal-based universal Q-function $Q\left(M_{\mathrm{g}}(s,a), \mathrm{g}\right)$, where $M_{\mathrm{g}}(s,a)$ is the expected measurement of future trajectories for performing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ with the goal g. Different from the previous definition in which the world model M^π is specific to the policy π, in the disentangled universal value function, the world model M_{g} is related to the goal g and can generalize to different goals. Given the composite function of U and M which is a strict lower-bound approximation of the Q-function, the best action a' in state s can be selected as follows:

$$a' = \max_a U\big(M_{\mathrm{g}}(s,a), \mathrm{g}\big) \tag{5}$$

we can also obtain the value-decomposed policy gradient $\nabla_\theta J(\pi_\theta)$ for a specific goal g with the Chain Rule as follows:

$$\begin{aligned} \nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi} \big[&\nabla_\theta \pi_\theta(s) \nabla_a M_{\mathrm{g}}(s,a)|_{a=\pi_\theta(s)} \\ &\nabla_{\mathrm{m}} U(\mathrm{m},\mathrm{g})|_{\mathrm{m}=M_{\mathrm{g}}(s,a)}\big]. \end{aligned} \tag{6}$$

Overall, our decomposition of value functions induces a goal-based Q-function and a world model M. The above



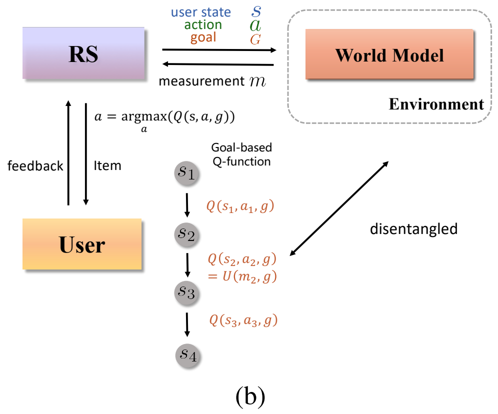


Figure 1: (a) The traditional reinforcement learning framework. (b) The proposed disentangled framework. In the traditional framework, the RL agent receives the feedback of environment (user) and corresponding shaping rewards, and then train in a fully real-time interactive manner. For the GoalRec, the RS agent requests the future measurement predictor (world model) with a query consists of state s, action (item) a and a specific goal g, and the user is recommended for one or more items to maximize the long-term utility $Q(s, a, \mathrm{g})$.

modeling can be understood in a two-step way: it first predicts the expected future dynamics under the goal g (world model M) then evaluates the benefits of future prediction (Q-function).

RS-specific Value Function Design

We have described the general framework of GoalRec. In this subsection, we first introduce the definitions of state and action, measurement, and goal in recommender systems, respectively. Then, we present a RS-specific instantiation of the goal-based Q-function.

State and action. We construct four categories of features to represent state and action: (a) Item features that describe whether certain property appears in this item, such as item's id, shop's id, brand, item's name, category, item embedding, and historical CTR in the last 1 hour, 6 hours, 24 hours and 1 week respectively. (b) User features consist of user portrait and features of the item that the user clicked in 1 hour, 6 hours, 24 hours, and 1 week respectively. (c) User-item features describe the interaction between user and one certain item, i.e., the frequency for the item (also category, brand and shop) to appear in the history of the user's clicks. (d) Context features describe the context when a request happens, including time, weekday, and the freshness of the item (the gap between the request time and item publish time). State s is represented by context features and user features. Action a is represented by item features and user-item interaction features.

Measurement. The definition of measurement is flexible and can be user-defined based on specific scenarios. Take the typical recommender system shown in Figure 2 as an example: the measurement can be designed as a vector composed by (a) rewarded user immediate responses (user purchase, etc.) and unrewarded user immediate responses (view, exit, etc.), (b) session-level business metrics (stay time, etc.), and (c) user's future trajectory representation (e.g., the averaged embedding of click items). The vectorial rewards (i.e., session-level business metrics and rewarded user responses) generalize the reward setting of previous RL-based methods: the immediate user satisfaction (e.g., user purchase) or long-term user satisfaction (e.g., stay time) can be viewed as a measurement. Moreover, the prediction of vectorial rewards, which are high-variance and sparse, can benefit from the prediction of other self-predictive measurements through the shared hidden layers.

Goal. In the traditional goal-based RL framework, the agent's goal is described by a single desired state, and the environmental reward can be defined as the "distance" between the current state and the desired goal. In our model, the goal is the desired user's future trajectories that maximize vectorial rewards. The goal plays two roles. First, through a distance function, the goal can be used to calculate the cumulative reward. Second, the goal can be regarded as a representation for the policies and thus determines future trajectories.

Q-function. Given the definitions of measurement and goal, the universal state-action value function $Q(s, a, \mathrm{g})$ can be defined as follows:

$$Q(s, a, \mathrm{g}) = U\big(M_{\mathrm{g}}(s, a), \mathrm{g}\big) = U\big(\mathrm{m}, \mathrm{g}\big) = \mathrm{g}^\top \mathrm{m} \quad (7)$$

where the unit-length vector g parameterizes the goal and has the same dimensionality as m.

World Model

As discussed in the previous subsection, the problem of universal Q-function estimation for a specific goal boils down to predicting the measurement m by the world model M:

$$Q(s, a, \mathrm{g}) = \mathrm{g}^\top \mathrm{m} = \mathrm{g}^\top M\left(s, a, \mathrm{g}; \boldsymbol{\theta}\right) \quad (8)$$

Here $a \in \mathcal{A}$ is an action, $\boldsymbol{\theta}$ are the learned parameters of M, and m is the predicted future measurement. Note that the resulting prediction is a function of the current observation s, the considered action a, and the goal g.

The network architecture we use is shown in Figure 3. The network has three input modules: a user representation

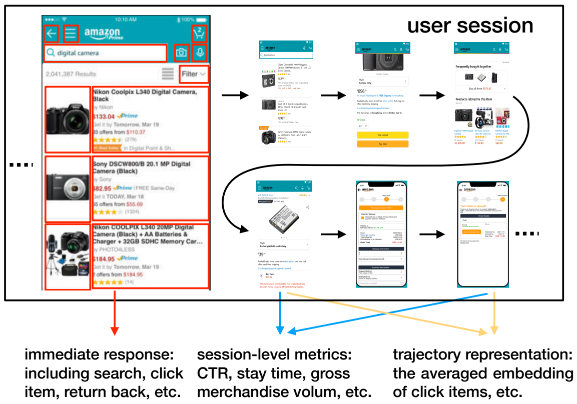

user session
immediate response: including search, click item, return back, etc.
session-level metrics: CTR, stay time, gross merchandise volum, etc.
trajectory representation: the averaged embedding of click items, etc.

Figure 2: The definition of measurement for a typical item recommender system.

(state) module $S(\mathbf{s})$, an item representation (action) module $E(a)$ and a goal module $G(\mathbf{g})$. The state $\mathbf{s}$ is represented by context features and user features, and the state module S can be any state-of-the-art recommendation models such as Wide&Deep and DIEN. The action and goal modules are fully-connected networks.

Built on the ideas of Dueling DQN (Wang et al. 2015), a similar structure is employed to enhance the ability to learn the differences between the outcomes of different actions. We split the prediction module into two parts: a state value part $V(s, \mathbf{g})$ and an advantage part $A(s, \mathbf{g}, a)$. The state value part predicts the average of future measurements over all potential actions. The action advantage part concentrates on the differences between actions. However, different from the vanilla dueling DQN, the world model does not take all actions into consideration, which means the average of the predictions over all actions is not available. To this end, we introduce an auxiliary loss to learn the state value part $V(s, \mathbf{g})$, and thus the average of the predictions of the action advantage part is nearly zero for each future measurement. The output of these two parts has dimensionality $\dim(\mathrm{m})$, where m is the vector of future measurements. Finally, the output of the network is a prediction of future measurements for action a, composed by summing the output of the state value part and the action-conditional output of the action advantage part.

Hindsight Experience Replay

We employ a re-labeling technique, called hindsight experience replay (HER) (Andrychowicz et al. 2017), to induce the corresponding parameterized goals (i.e., reward settings) given any trajectories. HER is widely used in goal-based RL, especially on the situation when the exploration of different goals is impossible (e.g., offline datasets) or costful (e.g., real applications). Its key insight is that the agent can transform failed trajectories with no reward into successful ones by assuming that a state it saw in the trajectory was the actual goal. In other words, since the goal does not affect the dynamics of the environment, we can re-label the trajectory afterwards to achieve different goals and thus improve sample utilization. For every episode the agent experiences, we store it in the replay buffer twice: once with the original goal pursued in the episode and once with the goal corresponding to the final measurement achieved in the episode, as if the agent intended on reaching this goal from the very beginning. We refer to Andrychowicz et al. (2017) for the full details of hindsight experience replay.

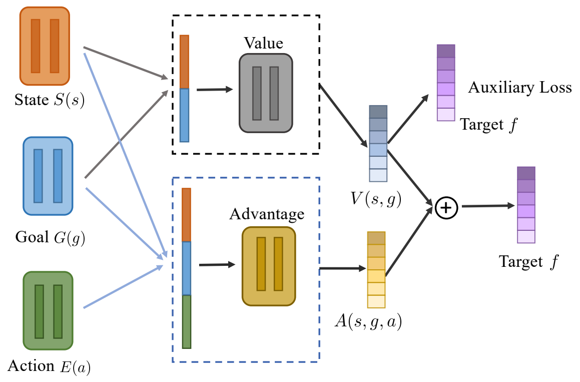


Figure 3: The network structure of world model. It can be regarded as a variant of the classic two-tower recommendation model, which is scalable and able to utilize high-dimensional item features.

Training

Consider a set of samples collected by the agent, yielding a set $\mathcal{D}$ of training trajectories. We then generate a training mini-batch $\mathcal{B} = \{\langle s_i, a_i, \mathrm{g}_i, \mathrm{m}_i\rangle\}_{i=1}^{N}$ from the set $\mathcal{D}$ (the steps 5 to 14 of Algorithm 1 shown in Appendix A). Here $\langle s_i, a_i, \mathrm{g}_i\rangle$ is the input and m_i is the output. Instead of manually tuning hyper-parameters to balance the importance of each measurement, we employ homoscedastic uncertainty trick (Kendall and Gal 2017) to tune the loss weights adaptively during training. Following the work of (Kendall, Gal, and Cipolla 2018), the regression loss is defined as:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{j} \left(\frac{1}{\sigma_j^2} \left\| M_j\left(\mathrm{s}_i, a_i, \mathrm{g}_i; \boldsymbol{\theta}\right) - \mathrm{m}_i^j \right\|^2 + \log \sigma_j^2 \right) \tag{9}$$

where the j indicates the j-th measurement and the σ_j^2 is the variance of estimated Gaussian distribution of the j-th measurement. When σ_j^2 increases, the corresponding weight decreases. Additionally, $\log \sigma_j^2$ serves as a regularizer to avoid overfitting. More complex loss functions such as classification loss or reconstruction loss can be used for predicting categorical or high-dimensional measurements, but are not necessary for our experiments. The agent follows an ϵ-greedy policy: it acts greedily according to the current goal with probability $1 - \epsilon$, and selects a random goal with probability ϵ. The parameters of the predictor used by the agent are updated after every N new experiences.

Offline Experiments

Following the setting of Shi et al. (2019) and Chen et al. (2018), we demonstrate how the proposed method would perform on a real-world recommender system by constructing a simulated user model on public recommendation datasets. We empirically evaluate GoalRec to address the following questions:

Q1. Is it appropriate to apply the GoalRec on multi-step recommendation problems? How does it compare with the existing supervised learning or reinforcement learning methods?

Q2. Is the GoalRec capable of tackling the high-dimensional state and action spaces problem in recommender systems? How about the high-variance environment?

Q3. Is the GoalRec capable of learning a promising policy from the offline dataset generated by other policies (e.g., a myopic recommendation model)?

Experiment Settings

Baselines. We compare methods ranging from Supervised Learning, Model-free RL to Model-based RL, including:

- **Wide&Deep** (Cheng et al. 2016): a widely used state-of-the-art deep learning model combining a logistic regression and a deep neural network to predict the click label.
- **DIEN** (Zhou et al. 2019): another widely used state-of-the-art deep learning model which employs the attention mechanism and takes user sequence features as input.
- **Rainbow** (Hessel et al. 2018): a popular off-policy value approximation paradigm, which combines several significant improvements of the DQN algorithm.
- **TD3** (Fujimoto, Van Hoof, and Meger 2018): an off-policy actor-critic method. The action space of TD3 is continuous. For inference, the continuous action finds its nearest neighbors in the actual item space.
- **Deep Dyna-Q** (Peng et al. 2018): a representative model-based approach combining one-step deep Q-learning and k-step Q-planning.
- **GoalRec**: the vanilla GoalRec which is training from scratch and uses a goal that maximizes the cumulative reward (number of clicks) for a fair comparison. The measurement used for different datasets are listed in Appendix B. Roughly, it is a vector composed of user click/purchase/exit action, the amount/ratio of different actions, and averaged embedding of items. The state module is implemented with open-sourced DIEN algorithm[4].
- **GoalRec-off**: a variant that is trained on offline training datasets instead of simulation environments.

For all compared algorithms, the recommendation item list is generated by selecting the items with top-k estimated potential reward or click probability of each item. For a fair comparison, RL-based methods employ similar state network structures and input state features, and the item features are not considered.

[4]https://github.com/mouna99/dien

Dataset. We use three real-world datasets: MovieLens-25m, Taobao and RecSys15 YooChoose. The dataset descriptions and detailed statistics are given in Appendix B.

Simulation Setting. Following the work of Zou et al. (2020), we regard "*positive rating*" or "*transactions*" as positive feedback (clicks), and conduct the simulation with data collected from the public recommendation dataset to fit a RNN-based user model (Jannach and Ludewig 2017). Apart from the feedback, the user model also predicts the users' exit probability, i.e., when to end the trajectory after losing patience. For each dataset, we randomly select 100,000 users as the simulation environment. The users are divided as 70% for training and 30% for testing. For each user, the number of interaction trajectories used for training is limited to 30 to simulate the data sparsity of real applications. We also set up two variants of the simulation environment to evaluate algorithms in the high-dimensional state/action environment and the high-variance environment, respectively. For the high-dimensional environment, the action dimension is increased to 10,000 with the state dimension expanded to 30,000 through the discretization and intersection of features. For high-variance environment, we delayed reward signals 3 steps and add a stochastic trend noise $y_t = y_{t-1} + y_{t-2} + \epsilon_t$, where ϵ_t is white noise with variance $\sigma^2 = 0.2$, on the output click probability of the estimated user model.

Evaluation Metrics. The performances are evaluated by two metrics: (a) Cumulative reward: we calculate the cumulative reward by averaging the number of clicks over all users. (b) CTR: the ratio of the number of clicks and the average browsing depth of users.

Parameter Setting. Grid search is applied to find the optimal settings for all methods, and we report the result of each method with its optimal hyper-parameter settings. The details of grid search reported in Appendix C. The exploration factor ϵ decays from 1 to 0 during the training. We used TensorFlow to implement the pipelines and trained networks with an Nvidia GTX 1080 ti GPU card.

Experiment Results

To address Q1, we first compare our methods with non-RL and RL-based methods, and the performance is shown in Table 1. The results show that all non-RL methods, compared with DRL methods, are stable but at a lower performance level in terms of reward. This is because they mainly focus on the item-level performance (a higher CTR) and are unable to improve the overall performance on trajectory level. Though only equipped with the vanilla DQN, the model-based approach (Deep Dyna-Q) achieves a higher reward than Rainbow on the Taobao dataset. Compared with Rainbow, Deep Dyna-Q, and TD3, GoalRec further improves the cumulative reward (an averaged increase of 3.7%, 6.1%, 10.7%, respectively), especially on Taobao dataset.

To address Q2, we compare the performance of different models between the vanilla environment and the high-dimensional environment. As shown in Table 1 and Table 2, TD3 works badly in the high-dimensional environment because of the increase of action space. The performance of Rainbow and Deep Dyna-Q also degrade because of the

Table 1: Performance comparison on the vanilla simulation environment.

	(1) MovieLens		(2) Taobao		(3) YooChoose	
model	Reward	CTR	Reward	CTR	Reward	CTR
W&D	25.2(±0.31)	**0.601(±0.006)**	2.72(±0.03)	**0.024(±0.001)**	3.52(±0.04)	0.040(±0.001)
DIEN	26.4(±0.27)	0.592(±0.007)	2.87(±0.04)	**0.024(±0.001)**	3.85(±0.05)	**0.041(±0.001)**
Dyna-Q	29.3(±0.80)	0.526(±0.016)	3.05(±0.10)	0.019(±0.002)	3.93(±0.13)	0.032(±0.003)
Rainbow	30.1(±0.67)	0.548(±0.012)	3.02(±0.05)	0.020(±0.002)	4.14(±0.09)	0.037(±0.002)
TD3	28.3(±1.05)	0.513(±0.019)	2.80(±0.10)	0.020(±0.002)	3.91(±0.14)	0.034(±0.003)
GoalRec	**31.6 (±0.50)**	0.544(±0.009)	3.14(±0.06)	0.021(±0.001)	**4.23(±0.07)**	0.037(±0.002)
GoalRec-off	29.1 (±0.37)	0.549(±0.005)	**3.20(±0.04)**	0.020(±0.001)	4.09(±0.05)	0.038(±0.001)

Table 2: Performance comparison on the high-dimensional simulation environment.

	(1) MovieLens		(2) Taobao		(3) YooChoose	
model	Reward	CTR	Reward	CTR	Reward	CTR
Dyna-Q	27.1(±0.74)	0.391(±0.013)	2.73(±0.08)	0.013(±0.002)	3.48(±0.10)	0.024(±0.002)
Rainbow	28.2(±0.72)	0.441(±0.015)	2.61(±0.06)	0.015(±0.002)	3.60(±0.08)	0.029(±0.003)
TD3	16.2(±0.59)	0.312(±0.024)	1.51(±0.05)	0.013(±0.002)	1.86(±0.07)	0.021(±0.003)
GoalRec	**33.8(±0.58)**	**0.567(±0.01)**	**3.24(±0.05)**	**0.022(±0.001)**	**4.35(±0.07)**	**0.037(±0.002)**

Table 3: Performance comparison on the high-variance simulation environment.

	(1) MovieLens		(2) Taobao		(3) YooChoose	
model	Reward	CTR	Reward	CTR	Reward	CTR
Dyna-Q	28.4(±0.81)	0.513(±0.015)	2.82(±0.08)	0.014(±0.002)	3.85(±0.09)	0.023(±0.002)
Rainbow	29.5(±0.73)	0.530(±0.018)	2.74(±0.06)	0.019(±0.001)	3.92(±0.05)	0.032(±0.002)
TD3	27.4(±0.95)	0.491(±0.026)	2.24(±0.08)	0.016(±0.002)	3.14(±0.07)	0.028(±0.003)
GoalRec	**32.1(±0.57)**	**0.536(±0.013)**	**3.02(±0.05)**	**0.020(±0.001)**	**4.05(±0.05)**	**0.034(±0.003)**

"credit assignment" issue caused by the high-dimensional state space. Conversely, GoalRec achieves a better performance in the high-dimensional environment because of the high-capacity world model and the goal-based dense reward signals. For the high-variance environment, the results reported in Table 1 and Table 3 show a similar conclusion. Deep Dyna-Q, Rainbow, and TD3 achieve a worse performance compared with the vanilla environment (an averaged decrease of 4.2%, 5.5%, 14.3%, respectively). The decoupled value function and the adoption of world model help in alleviating the optimization issue caused by the high-variance environment.

To address Q3, we compare the performance of GoalRec-off and GoalRec. As shown in Table 1, GoalRec-off achieves a promising performance with no interactions with the simulator. The ability of learning from offline datasets directly is important since pretraining on simulation environments may be problematic (though it is widely used in existing literatures): (i) the offline dataset is usually generated by a myopic policy and only covers limited state-action space; (ii) the supervised simulator which responded to unseen state-action confidently may mislead the RL algorithms.

Overall, the first two experiments verified the effectiveness of GoalRec when assuming the learned user simulator as ground truth. The third experiment shows that GoalRec offers another solution to learn from offline user trajectories and it works well. GoalRec is able to learn a generalization over different goals (myopic policies, etc.) from offline datasets and be adapted to a new goal (which maximizes the number of clicks) during inference. To overcome the shortcomings of offline evaluation, we also report online experiments with real-world commercial users in Appendix C.

Conclusion and Discussion

This paper presents GoalRec, which is motivated by trying to address challenges due to large state/action spaces, high-variance environment, and unspecific reward setting. The approach decouples the model that predicts the environment dynamics and sub-information, encoded in measurements, from the way that different measurements are then encoded into rewards for various goals that the recommender may have. As a potential instantiation of the disentangled goal-based Q-function, GoalRec factors the value function into the dot-product between the representation of user future trajectories and a vector of parameterized goal. It may more reasonable to think from the perspective of trajectories rather than states: recommendation environments lack significant long-range dependence between the state. By contrast, in video game environments, the key state is extremely important for future states, such as obtaining a key or entering a new scene. The influence of the recommender agent is subtle, and the user's preference changes gradually and slowly.

Several directions are left open in our work, including balancing explore-exploit in the goal-based RL framework and exploring the applicability of our solution in item-list recommendation.

References

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, O. P.; and Zaremba, W. 2017. Hindsight experience replay. In *Advances in neural information processing systems*, 5048–5058.

Bai, X.; Guan, J.; and Wang, H. 2019. A Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation. In *Advances in Neural Information Processing Systems*, 10734–10745.

Borsa, D.; Barreto, A.; Quan, J.; Mankowitz, D.; Munos, R.; van Hasselt, H.; Silver, D.; and Schaul, T. 2018. Universal successor features approximators. *arXiv preprint arXiv:1812.07626* .

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, X.; Li, S.; Li, H.; Jiang, S.; Qi, Y.; and Song, L. 2018. Generative adversarial user model for reinforcement learning based recommendation system. *arXiv preprint arXiv:1812.10613* .

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 7–10.

Colas, C.; Fournier, P.; Sigaud, O.; Chetouani, M.; and Oudeyer, P.-Y. 2018. CURIOUS: intrinsically motivated modular multi-goal reinforcement learning. *arXiv preprint arXiv:1810.06284* .

Dulac-Arnold, G.; Mankowitz, D.; and Hester, T. 2019. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901* .

Feinberg, V.; Wan, A.; Stoica, I.; Jordan, M. I.; Gonzalez, J. E.; and Levine, S. 2018. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101* .

Fujimoto, S.; Van Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477* .

Ha, D.; and Schmidhuber, J. 2018. World models. *arXiv preprint arXiv:1803.10122* .

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2018. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551* .

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hu, Y.; Da, Q.; Zeng, A.; Yu, Y.; and Xu, Y. 2018. 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*, 368–377.

Huzhang, G.; Pang, Z.-J.; Gao, Y.; Zhou, W.-J.; Da, Q.; Zeng, A.-x.; and Yu, Y. 2020. Validation Set Evaluation can be Wrong: An Evaluator-Generator Approach for Maximizing Online Performance of Ranking in E-commerce. *CoRR* .

Ie, E.; Jain, V.; Wang, J.; Narvekar, S.; Agarwal, R.; Wu, R.; Cheng, H.-T.; Chandra, T.; and Boutilier, C. 2019. SlateQ: A tractable decomposition for reinforcement learning with recommendation sets .

Jannach, D.; and Ludewig, M. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 306–310.

Ke, N. R.; Singh, A.; Touati, A.; Goyal, A.; Bengio, Y.; Parikh, D.; and Batra, D. 2019. Learning dynamics model in reinforcement learning by incorporating the long term future. *arXiv preprint arXiv:1903.01599* .

Kendall, A.; and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, 5574–5584.

Kendall, A.; Gal, Y.; and Cipolla, R. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7482–7491.

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* .

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* .

Peng, B.; Li, X.; Gao, J.; Liu, J.; Wong, K.-F.; and Su, S.-Y. 2018. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176* .

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6(Sep): 1265–1295.

Shi, J.-C.; Yu, Y.; Da, Q.; Chen, S.-Y.; and Zeng, A.-X. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4902–4909.

Sutton, R. S. 1991. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin* 2(4): 160–163.

Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; and De Freitas, N. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581* .

Zhao, X.; Xia, L.; Zhang, L.; Ding, Z.; Yin, D.; and Tang, J. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 95–103.

Zhao, X.; Zhang, L.; Xia, L.; Ding, Z.; Yin, D.; and Tang, J. 2017. Deep reinforcement learning for list-wise recommendations. *arXiv preprint arXiv:1801.00209* .

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.

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.

Zou, L.; Xia, L.; Du, P.; Zhang, Z.; Bai, T.; Liu, W.; Nie, J.-Y.; and Yin, D. 2020. Pseudo Dyna-Q: A Reinforcement Learning Framework for Interactive Recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 816–824.

Appendix A. Algorithm Description

The training procedure for GoalNet is shown in Algorithm 1.

```
Algorithm 1: The training procedure of GoalRec
Input: offline experience dataset $\mathcal{D}$; a goal $\mathbf{g}$; a strategy $H$
       for sampling states for hindsight experience replay;
       future measurement predictor (i.e., world model) $M$
1  Randomly Initialize parameters;
2  # Pretraining the world model.
3  for ii=1: N do
4      Sample a trajectory $\tau_{0:t}$ from $\mathcal{D}$;
5      for jj=1: K do
6          Sample a $(s_i, a_i, \mathbf{g}_i, s_{i+1})$ from $\tau_{0:t}$
7          Calculate the future measurement $m_i$ for
             trajectory $\tau_{i:t}$;
8          Append $(s_i, a_i, \mathbf{g}_\mathbf{i}, m_i)$ into mini-batch;
9          # Hindsight experience replay.
10         Sample a additional state $s_k$ for replay
             $s_k := H(T)$;
11         Calculate the future measurement $m_i'$ for
             trajectory $\tau_{i:k}$;
12         Calculate the hindsight goal $\mathbf{g}_i'$ that maximizes
             $\mathbf{g}_i^\top m_i'$;
13         Append $(s_i, a_i, \mathbf{g}_i', m_i')$ into mini-batch;
14     end
15     Update $\theta_M$ via mini-batch SGD w.r.t. the loss in
         Equation (??);
16 end
17 # Online training of GoalRec
18 while True do
19     for j=1: N do
20         given a customer $u$ and initialize $s = \{u\}$;
21         while customer not exit do
22             given item candidates $I$;
23             calculate the future measurement $m$ for each
                 item in $I$ using world model $M$;
24             calculate the Q-value according to the goal $\mathbf{g}$;
25             sample an action $a$ by $\epsilon$ -greedy w.r.t Q-value;
26             execute $a$;
27             customer $u$ responds with new state $s'$;
28             store $(s, a, \mathbf{g}, s')$ in customer trajectory $\mathcal{T}$;
29             update $s \leftarrow s'$;
30         end
31         store trajectory $\mathcal{T}$ in dataset $\mathcal{D}$;
32     end
33     # Updating the world model.
34     repeat
35         perform step (3)~(15);
36     until convergence;
37 end
```

Appendix B. Dataset Description

We used three real-world datasets, including:

(1) MovieLens public dataset[1] contains large amounts of movie ratings collected from their website. On average, each of these active users rated more than 500 movies (including short films), so we assume they rated almost every movie that they watched and thus equate their rating behavior with watching behavior. MovieLens dataset is the most suitable public dataset for our experiments, but it is still not perfect. In fact, only the Taobao dataset provides the context in which a user's choice is made. Thus, following the setting of (Chen et al. 2018), we simulate this missing information in a reasonable way for the MovieLens dataset. For each movie watched(rated) on the date d, we collect a list of movies released within a month before that day d. On average, movies run for about four weeks in the theater. Even though we don't know the actual context of user's choice, at least the user decided to watch the rated movie instead of other movies in theater. Besides, we control the maximal size of each display set by 40. **Features:** In the MovieLens dataset, only titles and IDs of the movies are given, so we collect detailed movie information from Internet Movie Database(IMDB). Categorical features as encoded as sparse vectors, and descriptive features are encoded as dense vectors. The combination of such two types of vectors produces 722-dimensional raw feature vectors. We conduct the simulation with the MovieLens dataset to fit an RNN-based user model (Jannach and Ludewig 2017). The number of hidden units is set to 256, the learning rate is 0.01, and the maximum iteration is 200,000 before convergence.

(2) Taobao[2] is a large dataset collected from Taobao online platform anonymously. It contains a subset of customer behaviors including click, purchase, adding item to shopping cart and item favoring from November 25, 2017 to December 03, 2017. It consists of 100,144,665 users' purchase and impression logs, involving dozens of thousands of items. It is a time-stamped dataset that contains user features, item features, and the context where the user clicks the item. We consider buying behaviors as positive events. **Features:** As described in the 'RS-specific value function design' subsection, we extract item features, user features, user-item features, and context features from the customer behavior sequences for each kind of user action. The overall features are approximately of dimension 300 after the embedding layer.

(3) RecSys15[3] contains click-streams that sometimes end with purchase events. Since the dataset only recorded the click and purchase events, we consider the click behaviors as impression events and buying behaviors as positive events (user clicks). **Features:** Similar to the Taobao dataset, we extract item features, user-item features, and context features from the user click sequences. The overall features are approximately of dimension 100 after the embedding layer.

Detailed statistics, including the number of customers, items, and behaviors, of these datasets is given in Table 1.

Table 1: Statistics of the datasets.

Dataset	#Users	#Items	#Total Behaviors	#Behaviors per User	#Behaviors per Item
MovieLens	162,541	62,423	25,000,095	153.81	400.49
Taobao	986,240	4,161,799	100,144,665	101.54	24.06
RecSys15	509,696	52,739	1,818,645	3.57	34.48

Appendix C. Online Experiments

We implement Wide&Deep, TD3, Rainbow and GoalRec algorithms in *Love is Justice*[5] which is one of the most popular games released by *NetEase Games*[6] for providing online item recommendation service. The item recommendation task in *Love is Justice* is

[1]https://grouplens.org/datasets/movielens

[2]https://tianchi.aliyun.com/datalab

[3]https://2015.recsyschallenge.com

[5]http://leihuo.163.com/en/games.html?g=game-2#nsh

[6]http://leihuo.163.com/en/index.html

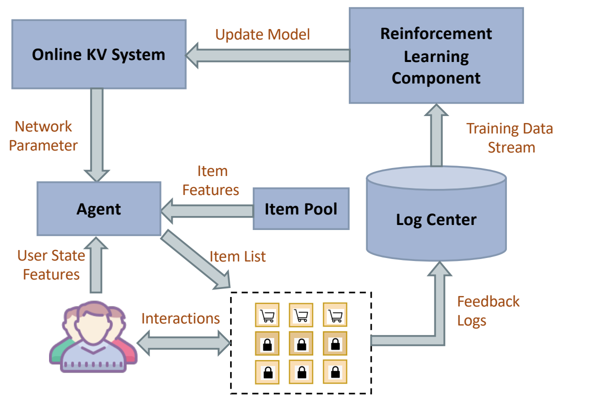


Figure 1: RL-based recommender system of *Love is Justice*.

characterized by its mid-dimensional action spaces (roughly 3000 item candidates) and special interaction rules. In each second, the recommendation engine should respond to thousands of users' requests with 3 item lists (3 items per list), and the next item list is locked until the items of current list are sold out. The users' response depended on not only the current item but also the items of next list. We formalize this item recommendation problem as a three-step decision-making problem in which the agent recommends three items as an item list in each step.

We design a data stream-driven RL recommender system for implementing our algorithm GoalNet. As shown in Figure 1, this system contains five major components: an agent (i.e., the GoalNet), an item pool, a logging center, a reinforcement learning component, and an online KV system. The workflow of our system mainly consists of two loops: the online planning loop and the online learning loop. As is shown in the left bottom of Figure 1, the online planning loop is the loop of the interactions between the recommendation agent and users. The second one is a learning loop (on the right in Figure 1) where the training process happens. The two working loops are connected through the logging center and the online KV system, which are used for collecting user logs and storing the network parameters, respectively. In the first loop, every time a user requests an item page, the agent will extract state features and item features, get the parameters of the model from the online KV system, and compute a ranking action for the current state (with exploration). The agent will apply the computed action to the unranked items and derive the top-3 items. After three steps, the ranker generates three item lists (3 items per list), where the user will give feedback. In the meanwhile, the log data produced in the online planning loop is injected into the learning loop for constructing training data source. In the logging center, the user logs collected from different search sessions are transformed to training samples $\langle s_i, a_i, \mathbf{g}_i, \mathbf{m}_i \rangle$. These samples are output continuously in the form of data stream and utilized by our algorithm to update the network parameters in the learning component. Whenever the model is updated, it will be rewritten to the online KV system.

The action of the recommendation engine is a 677-dim item feature vector, which contains the item features and user-item interaction features. The state of the environment is represented by an 8352-dim user feature vector, which contains the user portrait features and sequence features described by item features of the current session or user history. The measurement is a 30-dim vector, including user exit action, user responses for each slot (user click/purchase, etc.), session-level business metrics (user click/purchase amount, etc.), and user's future trajectory representation. The goal is set to maximize the expected user purchase amount. We employ metrics including IPV (Individual Page View), PAY (number of payments), and GMV (Gross Merchandise Volume). The averaged results are presented in Table 5. On each day of the test, the GoalRec algorithm can lead to more transaction amount than the Wide&Deep, TD3, and Rainbow (an increase of $15.2\%, 10.4\%, 6.8\%$, respectively).

Table 2: Comparison between GoalRec and other baselines in online experiments.

Method	GMV	PAY	IPV
GoalRec	**1.000**	**0.823**	**1.069**
Wide&Deep	0.868	0.623	0.815
TD3	0.906	0.701	0.870
Rainbow	0.936	0.752	0.904