RL4RS : A Real-World Benchmark for Reinforcement Learning based Recommender System

Kai Wang
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
wangkai02@corp.netease.com

Zhene Zou
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
zouzhene@corp.netease.com

Qilin Deng
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
dengqilin@corp.netease.com

Yue Shang
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
shangyue@corp.netease.com

Minghao Zhao
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
zhaominghao@corp.netease.com

Runze Wu
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
wurunze1@corp.netease.com

Xudong Shen
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
hzshenxudong@corp.netease.com

Tangjie Lyu
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
hzlvtangjie@corp.netease.com

Changjie Fan
Fuxi AI Lab, NetEase Games
Hangzhou, Zhejiang, China
fanchangjie@corp.netease.com

ABSTRACT

Reinforcement learning based recommender systems (RL-based RS) aims at learning a good policy from a batch of collected data, with casting sequential recommendation to multi-step decision-making tasks. However, current RL-based RS benchmarks commonly have a large reality gap, because they involve artificial RL datasets or semi-simulated RS datasets, and the trained policy is directly evaluated in the simulation environment. In real-world situations, not all recommendation problems are suitable to be transformed into reinforcement learning problems. Unlike previous academic RL researches, RL-based RS suffer from extrapolation error and the difficulties of being well validated before deployment.

In this paper, we introduce the RL4RS (Reinforcement Learning for Recommender Systems) benchmark - a new resource fully collected from industrial applications to train and evaluate RL algorithms with special concerns on the above issues. It contains two datasets, tuned simulation environments, related advanced RL baselines,data understanding tools, and counterfactual policy evaluation algorithms. The RL4RS suit can be found at https://github.com/fuxiAIlab/RL4RS. In addition to the RL-based recommender systems, we expect the resource to contribute to research in reinforcement learning and neural combinatorial optimization.

1 INTRODUCTION

In 2020, retail e-commerce sales worldwide amounted to 5.23 trillion US dollars and e-retail revenues are projected to grow to 6.54 trillion US dollars in 2022. Such rapid growth promises a great future for the worldwide e-commerce industry signifying a strong market and increased customer demand. Besides the huge increment of the traffic volume, there has been a rapid growth of various recommendation scenarios, including slate recommendation, bundle recommendation (a collection of items that should be purchased simultaneously), sequential item recommendation, and many others, as shown in Figure 1. It is worth exploring the various challenges that the modern e-commerce industry faces today. Most current e-commerce and retail companies build their recommender systems by implementing supervised learning based algorithms on their websites to maximize immediate user satisfaction in a greedy manner. However, the item-wise greedy recommendation strategy is imperfect fitting to real recommendation systems. With more and more new upcoming recommendation scenarios, more and more challenges have to be solved. For instance, in sequential recommendation scenarios, traditional methods often consider different ranking steps in a session to be independent and fail to maximize the expected accumulative utilities in a recommendation session. In the slate recommendation or bundle recommendation, the conversion rate of an item does not solely depend on itself. If an item is surrounded by similar but expensive items, the conversion rate increases, known as the decoy effect [17]. However, the possible combinations of all the items can be billions, which is an NP-hard problem and less explored in traditional supervised learning.

To deal with these challenges, recent researches resort to adopting reinforcement learning for recommendations, in which the recommendation process is formulated as a sequential interaction between the user (environment) and the recommendation agent (RL agent). Reinforcement learning is a promising direction since the RL paradigm is inherently suitable for optimizing long-term user satisfaction directly, exploring the combination spaces efficiently, and tackling multi-step decision-making problems - but there remain two main problems in recent researches.

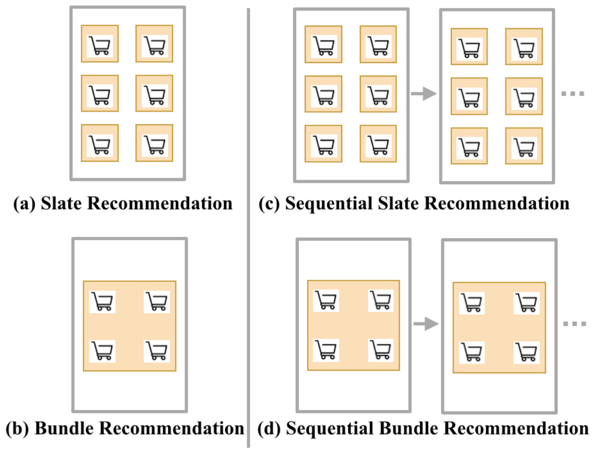
(a) Slate Recommendation
(c) Sequential Slate Recommendation
(b) Bundle Recommendation
(d) Sequential Bundle Recommendation

Figure 1: several novel Recommendation scenarios.

The first problem is the lack of real datasets for RL-based RS problems. There are mainly two alternatives, one is artificial datasets, such as RecoGym [28] and RECSIM [18]. The main disadvantage is that they are not the real feedback of users in real applications. Another one is semi-simulated RS datasets, i.e., transforming traditional datasets such as MovieLens to RL data format. Take MovieLens dataset as an example, to meet the requirements of RL data format, Adversarial User Model [5] assumes the context of user's choice as the movies released within a month and the maximal size of each displayed set as 40. The main disadvantage of semi-simulated datasets is that most forced data transformations are not reasonable.

The second problem is the lack of unbiased evaluation methods. In current researches, there are mainly two kinds of evaluation indicators: traditional recommendation indicators (recall rate, accuracy, etc.) and pure reinforcement learning indicators (e.g., cumulative rewards). However, the former ones are indirect evaluation indicators, and the latter ones highly depend on the accuracy of the simulation environment. The bias of offline policy evaluation mainly comes from "extrapolation error", which is a phenomenon in which unseen state-action pairs are erroneously estimated to have unrealistic values. In this paper, we further explore the other two recently developed methods to tackle "extrapolation error", counterfactual policy evaluation and RL.

With these in mind, we introduce RL4RS - an open-source benchmark for applied RL developed and deployed at Netease. RL4RS is built in Python and uses TensorFlow for modeling and training. It aims to fill the rapidly-growing need for RL systems that are tailored to work on novel recommendation scenarios. It consists of (1) two large-scale raw logged data, reproducible simulation environments, and related RL baselines. (2) data understanding tools for testing the proper use of RL, and systematic evaluation process, including environment fitting evaluation, policy evaluation on the simulation environment, and counterfactual policy evaluation. (3) the separated data before and after reinforcement learning deployment for each dataset. Based on them, we are able to measure the extent of extrapolation error and evaluate the effectiveness of different RL algorithms, including RL algorithms.

2 IMPACT

In this section, we assess the benefits of the RL4RS resource in relation to existing resources for evaluating RL-based RS. We collect all relevant works that have been open-sourced at present, including RecoGym[1][28], Recsim[2][18], Top-k off-policy[3][4], SlateQ[4][19], Adversarial User Model[5][5], List-wise[6][37], Virtual Taobao[7][32], and Model-Based RS[8][1].

We consider the following dimensions to evaluate the benefit of these related works:

- **Artificial Datasets**: RecoGym and Recsim are two representative artificial datasets, which are employed in the experiments of Top-K and SlateQ.
- **Semi-simulated Datasets**: Traditional RS datasets such as MovieLens are designed for item-wise supervised learning and are not suitable for RL-based RS experiments. As a suboptimal solution, Adversarial User Model, List-wise and Model-based RS make a lot of assumptions and manual transformations on these datasets to fit the RL requirements.
- **Real Industry Datasets Without Transformation**: Though many works build their online experiments on real industrial scenarios, there are few works providing the reproductive offline experiment result on real offline datasets.
- **Code Release**: We list the GitHub pages of each algorithm at the beginning of this section. Top-k off-policy and SlateQ are non-official implementations.
- **Dataset Release**: In addition to artificial datasets and semi-simulated RS datasets, virtual Taobao builds experiments on a real dataset but without open-sourcing the raw logged data. Raw logged data is necessary for the reproduction of the simulation environment and the implementation of offline policy learning and evaluation.
- **Simulation Environment Release**: Virtual Taobao has open-sourced a low-dimensional version of the pre-trained simulation environment, which is associated with 11-dimensional user features and 27-dimensional item features.
- **Offline policy learning**: Most current works firstly train the environmental model and then learn the policy through the interaction with the trained environment, except Top-k off-policy. In this paper, we further explore the extrapolation error and RL algorithms.

[1] https://github.com/criteo-research/reco-gym
[2] https://github.com/google-research/recsim
[3] https://github.com/awarebayes/RecNN
[4] https://github.com/ray-project/ray/blob/master/rllib/agents/slateq/
[5] https://github.com/xinshi-chen/GenerativeAdversarialUserModel
[6] https://github.com/luozachary/drl-rec
[7] https://github.com/eyounx/VirtualTaobao
[8] https://github.com/XueyingBai/Model-Based-Reinforcement-Learning-for-Online-Recommendation

Table 1: A comparison between RL4RS and other resources.

	Dataset			Opensource			Others	
	Artifical dataset	Semi-simulated dataset	Real dataset	Code	Raw logged data	simulation environment	Offline policy learning	Offline policy evaluation
RecoGym[28]	✓	×	×	✓	×	✓	×	×
Recsim[18]	✓	×	×	✓	×	✓	×	×
Top-k Off-policy[4]	✓	×	×	✓	×	×	✓	×
SlateQ[19]	✓	×	×	✓	×	×	×	×
Adversarial[5]	×	✓	×	✓	×	×	×	×
List-wise[37]	×	✓	×	✓	×	×	×	×
Virtual-Taobao[32]	×	×	✓	✓	×	✓	×	×
Model-based[1]	✓	✓	×	✓	×	×	✓	×
Ours	×	×	✓	✓	✓	✓	✓	✓

- **Offline policy evaluation**: Offline policy evaluation aims to predict the performance of a newly learned policy without having to deploy it online. Rather than test the policy in a simulation environment, in this paper, we introduce the counterfactual policy evaluation (CPE).

In Table 1, we summarize the characteristics of existing works in terms of these dimensions. It can be seen that our RL4RS benchmark is the only one that meets all requirements. In addition to the contribution of open-sourcing industrial datasets, we further explore the topics such as offline policy training, extrapolation error, and counterfactual policy evaluation, hoping to enhance the development of RL-based RS field.

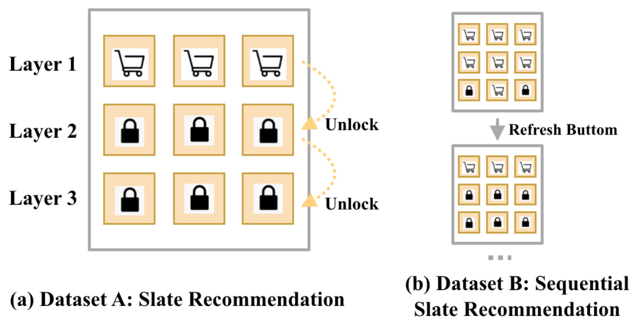
Layer 1
Layer 2
Layer 3
Unlock
Unlock
Refresh Buttom
(a) Dataset A: Slate Recommendation
(b) Dataset B: Sequential Slate Recommendation

Figure 2: A graphic description of RL4RS datasets.

3 DATA DESCRIPTION

We collect the raw logged data from one of the most popular games released by *NetEase Games*[9]. The item recommendation task in this game is characterized by its 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. Obviously, thanks to the special 'unlock' rule, the users' response to an item depends on not only that item but also the items of other lists. If users are not satisfied with the existing item slate, they can refresh the page (i.e. the item slate) through a refresh button up to 3 times per day.

Here, we provide two benchmark datasets, Dataset A and Dataset B. Dataset A focus on the slate recommendation. It regards the user's behavior on a single page as an MDP process. Dataset B focuses on the sequential slate recommendation. It not only considers how to recommend a single page but also considers the relationship between pages to maximize the total reward of pages. We will make data exploration in Section 4 to compare the MDP properties of our datasets and traditional RS datasets. A brief statistics of RL4RS datasets are shown in Table 2.

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

3.1 Logged Data

Roughly speaking, the logs mainly record the recommendation context, the user's behavior sequence, and the deployed behavior policy at that time. The context includes the metadata of some users and items to form the features of users and items. The recorded behavior policy is used to reproduce the probability of each action at that time and to evaluate the counterfactual strategy. As shown in Figure 3, the recommendation engine contains five major components: an RS agent, an item pool, a logging center, a training 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 3, the online planning loop is the loop of the interactions between the recommendation agent and users. We will record the item information of each session and the user's behavior for each item. The second one is a learning loop (on the right in Figure 3) where the training process happens. Whenever the model is updated, it will be rewritten to the online KV system. We will record the network architecture and network parameters at that time. The two working loops are connected through the logging center and the online KV system. We will record the corresponding user and item features stored in the online KV system for each user log.

After aligning the user log, real-time features and behavior

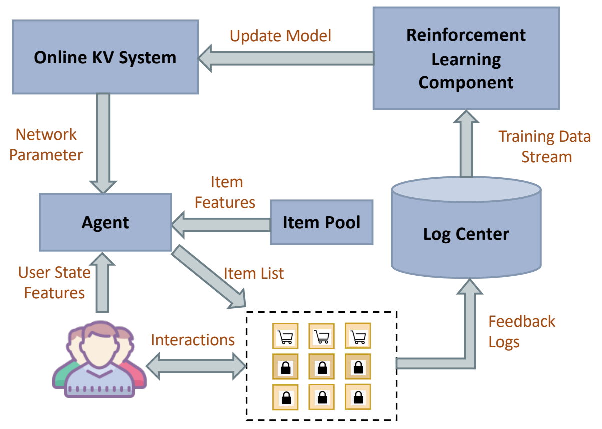

Figure 3: The architecture of recommender system.

policy, the format of raw logged data is:

- **TimeStamp**: The timestamp when the event happens.
- **Session ID**: A unique number which uniquely identifies an user session.
- **Sequence ID**: A unique number representing the location of the state in the session (i.e., a page order).
- **Exposed items**: A nine-length space-delimited list representing the nine items exposed to users (left to right top to bottom).
- **User feedback**: A nine-length space-delimited list representing the user's reponses to the nine exposed items (left to right top to bottom).
- **User Feature**: The anonymized user features which consists of user portrait and features of the item that the user clicked.
- **Item Feature**: The anonymized item features that describe whether certain property appears in this item, such as item's id, item's name, category, item embedding, and historical CTR.
- **Behavior Policy ID**: A model file that records the network architecture and network parameters of the behavior policy (supervised learning strategy) at that time.

3.2 Data Preprocessing

The raw logged data need to further be transformed to consecutive pairs of state/action tuple for RL models training. Besides, there are some differences for RL4RS datasets, such as time difference between states, action mask, action probability (used in counterfactual policy evaluation), and so on.

Specifically, we transforms the logged data collected in the following row format:

- **MDP ID**: A unique ID for the Markov Decision Process (MDP) chain that this training example is a part of.
- **Sequence ID**: A number representing the location of the state in the MDP (i.e. a timestamp).
- **State Features**: The features of the current step that consists of user features and context features.
- **Action**: The action taken at the current step.
- **Action Feature**: The features of item taken at the current step.
- **Action Probability**: The probability that the behavior policy took the action.
- **Action Mask**: An list of possible items at the current step.
- **Reward**: The reward of the current step.
- **Next State Features**: The state features after acting the logged action.
- **Next Action**: The item recommended at the next step.
- **Next Action Feature**: The features of item recommended at the next step.
- **Next Action Probability**: The probability of the item that is recommended at the next step.
- **Next Action Mask**: A list of items that are allowed to recommend at the next step.
- **Terminal**: A 0/1 number representing whether it is the last state.
- **TimeDiff (Optional)**: A number representing the time difference between the current state and next state.

4 DATA UNDERSTANDING

One big challenge of applied RL is problem formulation. Traditional recommendation scenarios and datasets are not always suitable for modeling as MDP problems where some sort of long-term reward is optimized in a sequential setting. It is easy to accidentally prepare data that does not conform well to the MDP definition and applying RL on ill-formulated problems is a costly process. Here, we develop a data understanding tool. Using a data-driven method together with heuristics, it checks whether the properties of RS datasets conform to the RL framework.

4.1 Long-term Impact

A optimal policy should take into account the long-term impact of a recommendation on the user's future choices. To maximize the accumulative rewards, we might suggest an item whose immediate reward is lower but leads to more likely or more profitable rewards in the future. For example, when the products sold are books, by recommending a book for which there is a sequel, we may increase the likelihood that this sequel will be purchased later. Heuristically, the way to measure whether a recommendation problem should be modeled as an RL problem is to see whether the recommendation decision in one step has a long-term impact. In terms of reinforcement learning formula, the recommendation of each step is to maximize $r(s_t, a_t) + V^*(s_{t+1})$, where $r(s_t, a_t)$ is the expected reward when recommending item a_t at state s_t, and $V^*(s_{t+1})$ represents the maximum reward of next state under current policy. We further denote $A(s_t, a_t) = V^*(s_{t+1}) - V^*(s_t, \cdot)$ as the difference between the future rewards of performing a_t and the averaged future rewards, i.e., the advantage of performing a_t at state s_t. When there is no long-term impact or the long-term impact is small, the RL problem degenerates into an ordinary sequential recommendation problem, which only maximizes the reward of the current step.

Table 2: RL4RS - Dataset Details

Num. of	RL4RS-A	RL4RS-A-SL	RL4RS-A-RL	RL4RS-B	RL4RS-B-SL	RL4RS-B-RL
Users	156179	109715	94563	156179	109715	94563
Items	381	381	381	381	381	381
Sessions	1773419	879514	893905	998672	484855	513817
Items per session	9.0	9.0	9.0	16.0	16.3	15.7
Purchase per session	5.36	5.25	5.46	9.53	9.51	9.53
Rewards per session	90.5	80.8	100.0	160.7	146.5	174.4

We establish a data understanding tool to quantify the long-term impact. without the requirement of establishing complex environment models or learning a value function, this tool is easy to use by simplifing RL as a sequence modeling problem. The similar ideas are developed in Trajectoy Transformer [20] and Decision Transformer [3], in which states, actions, and returns are fed into a GPT [27] architecture and actions are decoded autoregressively.

First, the tool fits a Transformer-based [36] sequence-to-sequence model on each offline RS dataset. It encodes the user context features (user historical behavior and user portrait feature) and decodes K items that users may click, which means that the recommendation system will recommend these K items in turn in the next K steps (considering that most RS datasets do not provide a complete page context, here, only one item is recommended at a time, eliminating the slate effect between items within a page). We consider using the decode sequence generated by greedy search to represent greedy recommendation (SL-based strategy), and the sequence generated by beam search to represent the recommendation result generated by the optimal RL policy. We use beam-search width as 1000 in each step. According to the previous discussion, when there is a significant long-term impact, the items recommended in the previous steps may not have high immediate impact, but the long-term impact is the large. It means that the immediate reward of the first item in the decode sequence accounts for a small proportion in the final score of the sequence (experiment I). On the other hand, we can compare the score of sequences composed of only hot items (with high immediate reward) with the optimal sequence score to check the greedy strategy is good enough (experiment II).

4.2 Experiment

We build experiments on the following datasets, MovieLens, RecSys15, RL4RS-A (has the same result of RL4RS-B under this experiment setting). More details are in Appendix C. Without losing generality, we only consider the long-term impact within 5 steps. For each dataset, we choose 10000 users randomly as test set. For each user, we calculate the greedy search result and the top 100 item sequences (beam search width as 100). In the first experiment, we report the Pearson Correlation Coefficient and Spearman Rank Correlation Coefficient between the score (immediate reward) of the first k items and the final score of the sequence, denoted as k-Pearson and k-Spearman respectively. If the item's long-term impact is significant, the coefficient should be small. We report the results on the item sequences generated by beam search, as shown in Table 3. It can be seen that, RL4RS dataset achieves a significant lower spearman rank coefficient at the first one and two items. Although the first experiment can not absolutely indicate whether a dataset is suitable for reinforcement learning, it provides a tool for comparison of the degree of suitability between different datasets.

In the second experiment, we aim to provide a tool to classify the dataset property definitely by checking whether the greedy strategy is already good enough. We report the averaged score of the top 5 (5% quantile) decode sequence, the top 20 (20% quantile) sequence and the score of sequence generated by greedy strategy. We also calculate the averaged score of the first 5% quantile and 20% quantile sequences when limiting the candidate items to hot items (20 items with the highest probability at the first decode step). All the scores are normalized by the averaged score of the top 5 sequence for each dataset. The results are shown in Table 4. It can be seen that there is a significant gap on the scores between the best 5% item sequences and greedy strategy in RL4RS-A, let alone the sequences composed of hot items. The result shows that it is necessary to model the RL4RS dataset as a RL problem. For traditional RS datasets, take randomness into consideration, there is no significant difference between the best RL policy and the greedy policy. It indicates that the greedy strategy (i.e., sequential RS methods) is well enough to model these scenarios.

Table 3: The comparison of item's long-term impact between different datasets.

Score. of	RecSys15	MovieLens	RL4RS-A
1-Pearson	0.63	0.54	0.43
1-Spearman	0.53	0.49	0.18
2-Pearson	0.67	0.62	0.50
2-Spearman	0.63	0.63	0.37
3-Pearson	0.75	0.70	0.75
3-Spearman	0.76	0.75	0.76
4-Pearson	0.84	0.82	0.85
4-Spearman	0.87	0.85	0.87
5-Pearson	1.00	1.00	1.00
5-Spearman	1.00	1.00	1.00

Table 4: The comparison of decode sequence scores between different datasets.

Score. of	5%	20%	greedy.	hot 5%	hot 20%
RecSys15	1.00	0.44	1.73	0.97	0.50
MovieLens	1.00	0.51	1.40	1.64	1.06
RL4RS-A	1.00	0.70	0.50	0.01	0.01

5 POLICY LEARNING

Unlike the academic environments which have perfect simulators, in real applications, RL has to learn from offline datasets considering the cost of online exploration. Given an offline dataset, we have two training methods. The first is training RL policies in the pre-trained simulation environment, which is the method adopted by most relevant works. The second is batch RL, i.e., training policies from offline datasets directly.

5.1 Online Policy Learning

A straightforward way to train an RL policy is to interact with the pre-trained simulation environment. Most existing RL-based RS works follow this online training manner, but introduce a new problem of how to simulate the environment.

Simulation Environment Construction. For recommendation systems, building a simulation environment is to build a perfect user behavior model. Different recommendation scenarios require different user behavior models. For RL4RS datasets, we predict the user's response to item slates, and measure the fitting effect of the model by predicting user purchase behavior. Specifically, we consider the following supervised learning baselines, including simple DNN, Wide&Deep [6], GRU4Rec [15], DIEN [39] and Adversarial User Model [5]. The specific network designs are detailed in the appendix. We consider three supervised learning tasks, namely slate-wise classification (multi-class classification), item-wise classification (binary classification) and item-wise rank (ranking task). The task description are detailed in the appendix. From the results shown in Table 7, DIEN (attention-based model) achieves the best predictive performance, and the indicators of the three tasks are represented positive correlation. Note that due to the high purchase rate of this scenario (more than 5 items are purchased per session), the indicators such as item-wise predictive accuracy are higher than the general RS scenarios.

RL Model Training. Given the simulation environment, we can employ any classical RL algorithms. When working on large discrete action spaces (more than millions of possible actions), one alternative is to address it with a continuous-action RL algorithm and combine policy gradients with a K-NN search [8]. When only sorting a few candidate items, we can choose to create a variant of DQN called Parametric-Action DQN, in which we input concatenated state-action pairs and output the Q-value for each pair. The Parametric-Action DQN [12] can make full use of rich item features, which is important for recommendation systems. Here, RL4RS suit provides both discrete-action and continuous-action baselines, includeing PG [33], DQN [26], parametric action DQN, A2C [25], DDPG [24] and PPO [29]. All these algorithms are implemented based on the open-source reinforcement learning library RLlib [23]. The online evaluation results built on the learned DIEN based simulation environment are shown in Table 5.

Table 5: Performance comparison between model-free RL algorithms on the learned simulation environment.

Rewards of	RL4RS-A	RL4RS-B
PG	127.2(±0.9)	242.8(±3.6)
DQN	175.2(±2.3)	271.4(±3.5)
A2C	167.8(±2.5)	263.7(±3.2)
DDPG	144.7(±6.0)	253.2(±10.8)

5.2 Offline Policy Learning

Offline policy learning aims to learn a better policy from the offline experience pool collected by poor behavior policies. Corresponding to RL-based RS, we have the user feedback data collected by supervised learning as the offline dataset. RL4RS suit provides two kinds of baselines, imitation learning (behavioral cloning) and batch RL (BCQ [10] and CQL [22]) . Behavioral cloning trains a policy to mimic the behavior policy from the data. We treat behavioral cloning (BC) as a baseline of offline learning methods. BCQ learns a state-conditioned generative model to mimic the behavior policy of the dataset, and a perturbation network to generate actions. The learned policy is constrained near the original behavior policy. CQL penalizes the value function for states and actions that are not supported by the data to prevent overestimation of the training policy. All these algorithms are implemented based on the open-source batch reinforcement learning library d3rlpy [30]. The online evaluation results built on the learned DIEN based simulation environment are shown in Table 6. Besides online evaluation, more evaluation results are reported in Section 6.

Table 6: Performance comparison between batch RL algorithms on the learned simulation environment.

Rewards of	RL4RS-A	RL4RS-B
BC	96.5(±1.0)	161.2(±1.8)
BCQ	166.7(±3.6)	276.1(±5.7)
CQL	164.5(±3.1)	275.9(±5.3)

6 POLICY EVALUATION

In the previous section, we provide a simple performance comparison between model-free RL and batch RL. These results are collected following the traditional RL evaluation framework, that is, training and online evaluating on the same environment. However, in real applications, it is usually rare to have access to a perfect simulator. A policy should be well evaluated before

Table 7: The experiment results of simulation environment construction.

		Slate-wise classification	Item-wise classification		Item-wise rank			
		Accuracy	AUC	Accuracy	Accuracy	Precision	Recall	F1 score
RL4RS-A	DNN	0.366	0.880	0.791	0.869	0.795	0.858	0.825
	Wide&Deep	0.382	0.891	0.806	0.873	0.796	0.863	0.828
	GRU4Rec	0.393	0.909	0.820	0.887	0.816	0.865	0.840
	DIEN	0.402	0.913	0.827	0.893	0.821	0.868	0.844
	Adver.	-	-	-	0.871	0.801	0.841	0.820
RL4RS-B	DNN	0.375	0.911	0.825	0.873	0.798	0.869	0.832
	Wide&Deep	0.391	0.917	0.834	0.878	0.797	0.879	0.836
	GRU4REC	0.408	0.923	0.840	0.885	0.812	0.863	0.836
	DIEN	0.413	0.930	0.849	0.889	0.819	0.877	0.846
	Adver.	-	-	-	0.875	0.802	0.867	0.833

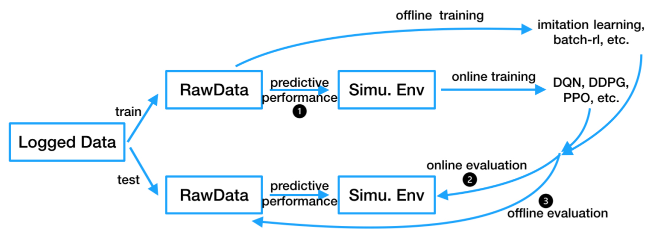

Figure 4: The evaluation framework for the RL-based RS task.

deployment since policy deployment affect the real world and may be costful. In applied setting, policy evaluation may not be a simple task that can be quantified by only a single indicator. As shown in Figure 4, in RL4RS, we propose a comprehensive evaluation framework for RL-based RS, including environment simulation evaluation (see Table 4), counterfactual policy evaluation (offline policy evaluation), and evaluation on simulation environments built from test dataset (online policy evaluation).

6.1 Counterfactual Policy Evaluation

Counterfactual policy evaluation (CPE) is a set of methods used to predict the performance of a newly learned policy without having to deploy it online [16, 7, 21, 35]. RL4RS provides several well known counterfactual policy evaluation (CPE) methods to score trained models offline, including direct method, importance sampling [16], step-wise weighted importance sampling (SWIS), doubly-robust [7], and sequential doubly-robust [21]. The direct method (DM) learns a regression based reward function to estimate expected rewards incur by the evaluated policy. To generalize, the simulation environment evaluation can be seen as a kind of DM method. The importance sampling (IS) uses action propensities of behavior and newly learned policies to scale logged rewards, which tends to have high variance if the action propensities of behavior policy are not logged. An improved step-wise IS version defines the t-step cumulative importance ratio and achieves a lower variance. The doubly-robust (DR) combines the ideas of the previous two methods and is widely used in contextual bandits. The sequential doubly-robust is specifically designed for evaluating policies on longer horizons.

We use counterfactual policy evaluation to evaluate the trained strategies described in Section 5.1 and 5.2 in offline training set and test set respectively. The results are shown in Table 8.

The CPE results indicates the expected performance of the newly trained policy relative to the policy that generated the training data. A score of 1.0 means that the RL and the logged policy match in performance. The sequential DR score of CQL (1.52) means that the CQL policy should achieve roughly 1.52x as much cumulative reward as the behavior policy. From the overall results, the IS-based method is relatively unstable in value. It is easy to reach the upper and lower clipped limits after action probabilities are multiplied. The DR-based method performs relatively well, thanks to the value function estimated by simulation environment. We will explore more state-of-the-art CPE methods in the future.

Table 8: Counterfactual policy evaluation on learned model-free RL and batch RL algorithms.

On RL4RS-A.	IS	SWIS	DR	Seq. DR
BCQ	18.9(±0.9)	0.99(±0.01)	1.31(±0.03)	1.39(±0.04)
CQL	188(±34.6)	1.01(±0.01)	1.52(±0.02)	1.46(±0.04)

6.2 Evaluation On Simulation Environment

The second method is to establish an environment on the test set to evaluate the algorithm. Different from testing in the same environment, we emphasize the establishment of environment model on test set according to the evaluation framework of supervised learning. In many recommendation scenarios, the training set and test set need to be divided according to time, which makes this more important. Another popular evaluation method adopted by most current works is to divide users before policy training. Although the training and test data are not overlapped, they are sharing the same environment. This method still exists a risk of over-fitting, because the estimated reward of the test set is affected by the training set through the shared environment.

According to the method described in Section 5.2, we use DIEN algorithm to train the environment model on the training set and the test set respectively. We use the most strict dataset dividing setting, that is, RL4RS-SL as the training set and RL4RS-RL as the test set. Not only they are not overlapped on time, but also there are considerable differences in data distribution between them. We compare the performance of the following three settings: (1) constructing environment model on RL4RS-SL plus RL4RS-RL, training the policy on RL4RS-SL, evaluating on RL4RS-RL; (2) constructing environment model on RL4RS-SL, training the policy on RL4RS-SL, evaluating on RL4RS-RL; (3) constructing environment model on RL4RS-SL, training the policy on RL4RS-SL, evaluating on the environment model built from RL4RS-RL.

The results are as follows.

7 AVAILABILITY

An old version of RL4RS is available at `https://www.kaggle.com/c/bigdata2021-rl-recsys/data` for direct download and use. To support findability and sustainability, the RL4RS dataset is published as an online resource at `https://fuxi-up-research.gitbook.io/open-project/`. A separate page with detailed explanations and illustrations is available at `https://fuxi-up-research.gitbook.io/fuxi-up-challenges/` to promote ease-of-use. The project GitHub repository contains the complete source code for the system and generation script is available at `https:/github.com/fuxiAIlab/RL4RS`. Documentation includes all relevant metadata specified to the research community and users. It is freely accessible under the *Creative Commons Attribution 4.0 International license*, making it reusable for almost any purpose.

7.1 Updating and Reusability

RL4RS is supported by a team of researchers from the Fuxi AI Lab, the Netease Inc. The resource is already in use for individual projects and as a contribution to the data challenge of BigData conference 2021. In addition to the steps above that make the resource available to the wider community, usage of RL4RS will be promoted to the network of researchers in this project. We will add more datasets from other novel RS scenarios to support the research of RL-based RS. RL4RS benchmark will be supported and maintained for three years and a second release of the dataset is already scheduled.

7.2 System Configuration

Pipelines for all tasks are trained for at most 100 epochs with an early stop on a single GeForce GTX 1080 Ti GPU, 8CPU, and 64GB memory. The code for the dataset splits is also available. For all tasks, consistent hyperparameter settings are maintained to enable comparative analysis of the results. The batch size is set at 128 with a learning rate of 0.001, a probability of 0.1 in dropout layers and the embedding size is 128.

8 RELATED RESEARCH

This benchmark is oriented to the reinforcement learning based recommender systems domain. In 2015, Shani et al. [31] proposes to model the recommendation system problem as an MDP process for the first time. In 2018, DRN [38] first apply deep RL into recommendation problems and inspire a series of subsequent works. SlateQ [19] is the first work to concern the novel slate recommendation scenario. Top-k off-policy [4] learn the policy from the offline RS dataset directly for the first time. Adversarial User Model [5] first popularizes semi-simulated RS datasets and simulates the environment using adversarial learning. Virtual-Taobao [32] is the only work to open-source a pre-trained simulation environment (but without raw logged data) that comes from a real RS scenario. A detailed comparison is listed in Section 2.

Outside of the RS domain, research on reinforcement learning for the online environment has an extensive history and we focus here on resources and benchmark tasks where samples are collected from real applications and present in batched offline data, such as DeepMind Control Suite [34], D4RL [9] and RL unplugged [14]. All of these resources exclude recommendation scenarios. And most batched datasets of these resources are collected by mixed, nearly random policies, and usually equipped with a deterministic environment. These characteristics are inconsistent with the characteristics of RS data. We are motivated by these researches to address offline policy evaluation and extrapolation errors in this resource.

RL-based RS can also be regarded as a neural combinatorial optimization (NCO) problem, especially for the slate recommendation scenario where the item appears as a matrix on the page. several works [13] try to use reinforcement learning to generate item sets and avoid enumerating exponential possible combinations. Like RL-based RS, current NCO works [11, 2] also lack attention to real datasets, offline policy evaluation, and extrapolation error. We expect RL4RS resources can also contribute to research in neural combinatorial optimization.

9 CONCLUSION AND FUTURE WORK

In this paper, we present the RL4RS resource, a real-world benchmark for RL-based RS. Since RS datasets are lack of ground-truth environment and usually collected with deployed supervised learning policies, for real-world considerations, RL4RS focuses on simulation environment construction, extrapolation error, offline policy learning, and offline policy evaluation, which are ubiquitous and crucial in recommendation scenarios. So far, RL4RS has included two novel scenarios, slate recommendation, and sequential slate recommendation. The separated data before and after RL deployment are also available for each dataset. We benchmark some state-of-the-art online and offline RL algorithms on RL-based RS task, including RL algorithms, in both online and offline policy evaluation manner. In addition, we further explore the quantitative analysis for extrapolation error and the use of counterfactual policy evaluation in RL-based RS problems.

In the future, we will constantly provide new real-world RS datasets and step further towards real-world RS challenges. We also hope the RL4RS benchmark will shed some light on future research of reinforcement learning and neural combinatorial optimization.

REFERENCES

[1] Xueying Bai, Jian Guan, and Hongning Wang. Model-based reinforcement learning with adversarial training for online recommendation. In *NeurIPS*, 2019.

[2] Quentin Cappart, D. Chételat, Elias Boutros Khalil, A. Lodi, Christopher Morris, and Petar Velickovic. Combinatorial optimization and reasoning with graph neural networks. *ArXiv*, abs/2102.09544, 2021.

[3] Lili Chen, Lu Kevin, Rajeswaran Aravind, Lee Kimin, Grover Aditya, Laskin Michael, Abbeel Pieter, Srinivas Aravind, and Mordatch Igor. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

[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] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. Generative adversarial user model for reinforcement learning based recommendation system. In *International Conference on Machine Learning*, pages 1052–1061. PMLR, 2019.

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

[7] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011.

[8] 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.

[9] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv*, 2020.

[10] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[11] Maxime Gasse, D. Chételat, Nicola Ferroni, Laurent Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *NeurIPS*, 2019.

[12] Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Yuchen R He, Zachary Kaden, Vivek Narayanan, Xiaohui Ye, and Scott Fujimoto. Horizon: Facebook's open source applied reinforcement learning platform. *ArXiv*, abs/1811.00260, 2018.

[13] Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q. Zhu. Exact-k recommendation via maximal clique optimization. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[14] C. Gulcehre, Z. Wang, A. Novikov, T. L. Paine, and N. D. Freitas. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv*, 2020.

[15] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *Computer ence*, 2015.

[16] Daniel G Horvitz and Donovan J Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

[17] Guangda Huzhang, Zhen-Jia Pang, Yongqing Gao, Wen-Ji Zhou, Qing Da, Anxiang Zeng, and Yang Yu. Validation set evaluation can be wrong: An evaluator-generator approach for maximizing online performance of ranking in e-commerce. *arXiv e-prints*, pages arXiv–2003, 2020.

[18] 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.

[19] Eugene Ie, Vihan Jain, Jing Wang, S. Narvekar, R. Agarwal, R. Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. In *IJCAI*, 2019.

[20] Michael Janner, Qiyang Li, and Sergey Levine. Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039*, 2021.

[21] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 652–661. PMLR, 2016.

[22] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *ArXiv*, 2020.

[23] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[25] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[27] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[28] 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*, abs/1808.00720, 2018.

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

[30] Takuma Seno. d3rlpy: An offline deep reinforcement library. `https://github.com/takuseno/d3rlpy`, 2020.

[31] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(9), 2005.

[32] J. Shi, Yang Yu, Qing Da, Shi-Yong Chen, and Anxiang Zeng. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *AAAI*, 2019.

[33] Richard S. Sutton, David A. McAllester, Satinder Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1999.

[34] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, Ddl Casas, D. Budden, A. Abdolmaleki, J. Merel, and A. Lefrancq. Deepmind control suite. *arXiv*, 2018.

[35] Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148. PMLR, 2016.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[37] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for list-wise recommendations. *arXiv preprint arXiv:1801.00209*, 2017.

[38] 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.

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