Deep Reinforcement Learning for Online Advertising in Recommender Systems

Xiangyu Zhao[1], Changsheng Gu[2], Haoshenglun Zhang[2],
Xiaobing Liu[2], Xiwang Yang[2], Jiliang Tang[1]
[1]Michigan State University, [2]Bytedance.com

Abstract

With the recent prevalence of Reinforcement Learning (RL), there have been tremendous interests in utilizing RL for online advertising in recommendation platforms (e.g. e-commerce and news feed sites). However, most RL-based advertising algorithms focus on solely optimizing the revenue of ads while ignoring possible negative influence of ads on user experience of recommended items (products, articles and videos). Developing an optimal advertising algorithm in recommendations faces immense challenges because interpolating ads improperly or too frequently may decrease user experience, while interpolating fewer ads will reduce the advertising revenue. Thus, in this paper, we propose a novel advertising strategy for the rec/ads trade-off. To be specific, we develop a reinforcement learning based framework that can continuously update its advertising strategies and maximize reward in the long run. Given a recommendation list, we design a novel Deep Q-network architecture that can determine three internally related tasks jointly, i.e., (i) whether to interpolate an ad or not in the recommendation list, and if yes, (ii) the optimal ad and (iii) the optimal location to interpolate. The experimental results based on real-world data demonstrate the effectiveness of the proposed framework.

Introduction

Online advertising is a form of advertising that leverages the Internet to deliver promotional marketing messages to consumers. The goal of online advertising is to assign the right ads to the right consumers so as to maximize the revenue, click-through rate (CTR) or return on investment (ROI) of the advertising campaign. The two main marketing strategies in online advertising are guaranteed delivery (GD) and real-time bidding (RTB). For guaranteed delivery, ad exposures to consumers are guaranteed by contracts signed between advertisers and publishers in advance (Jia et al. 2016). For real-time bidding, each ad impression is bid by advertisers in real-time when an impression is just generated from a consumer visit (Cai et al. 2017). However, the majority of online advertising techniques are based on offline/static optimization algorithms that treat each impression independently and maximize the immediate revenue for each impression, which is challenging in real-world business, especially when the environment is unstable. Therefore, great efforts have been made on developing reinforcement learning based online advertising techniques (Cai et al. 2017; Wang et al. 2018a; Zhao et al. 2018b; Rohde et al. 2018; Wu et al. 2018b; Jin et al. 2018), which can continuously update their advertising strategies during the interactions with consumers and the optimal strategy is made by maximizing the expected long-term cumulative revenue from consumers. However, most existing works focus on maximizing the income of ads, while ignoring the negative influence of ads on user experience for recommendations.

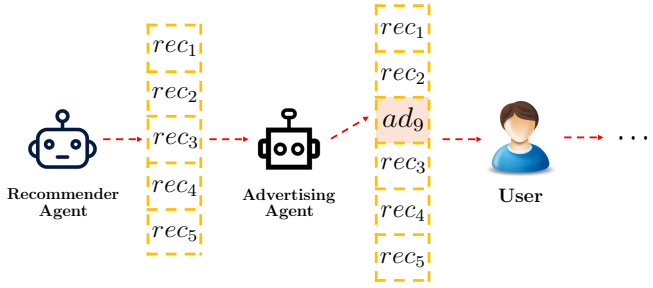

Figure 1: An example of online advertising within recommendation list for one user request

Designing an appropriate advertising strategy is a challenging problem, since (i) displaying too many ads or improper ads will degrade user experience and engagement; and (ii) displaying insufficient ads will reduce the advertising revenue of the platforms. In real-world platforms, as shown in Figure 1, ads are often displayed with normal recommended items, where recommendation and advertising strategies are typically developed by different departments, and optimized by different techniques with different metrics (Feng et al. 2018). Upon a user's request, the recommendation system firstly generates a list of recommendations according to user's interests, and then the advertising system needs to make three decisions (sub-actions), i.e., whether to interpolate an ad in current *recommendation list* (rec-list); and if yes, the advertising system also needs to choose the optimal ad and interpolate it into the optimal location (e.g. in Figure 1 the *advertising agent* (AA) decides to interpolate an ad ad_9 between rec_2 and rec_3 of the rec-list). The first sub-action maintains the frequency of ads, while the other two sub-actions aims to control the appropriateness of ads. The goal of advertising strategy is to si-

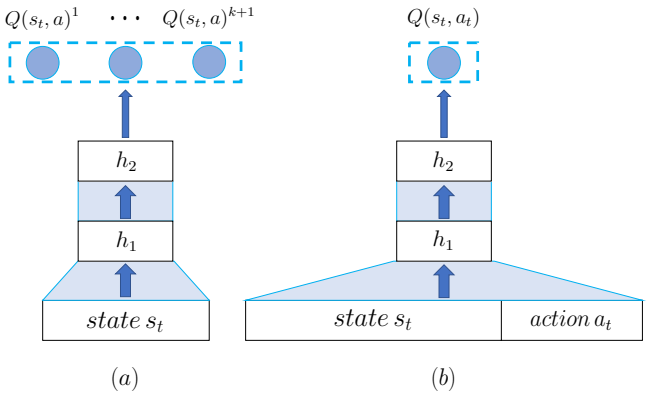

Figure 2: Classic DQN architectures for online advertising.

multaneously maximize the income of ads and minimize the negative influence of ads on user experience.

The above-mentioned three decisions (sub-actions) are internally related, i.e., (only) when the AA decides to interpolate an ad, the locations and candidate ads together determine the rewards. Figure 2 illustrates the two conventional Deep Q-network (DQN) architectures for online advertising. Note that in this paper we suppose (i) there are $|A|$ candidate ads for each request, and (ii) the length of the recommendation list (or rec-list) is L. The DQN in Figure 2(a) takes the state space and outputs Q-values of all locations. This architecture can determine the optimal location but cannot choose the specific ad to interpolate. The DQN in Figure 2(b) inputs a state-action pair and outputs the Q-value corresponding to a specific action (ad). This architecture can select a specific ad but cannot decide the optimal location. Taking a representation of location (e.g. one-hot vector) as the additional input is an alternative way, but $O(|A| \cdot L)$ evaluations are necessary to find the optimal action-value function $Q^*(s, a)$, which prevents the DQN architecture from being adopted in practical advertising systems. It is worth to note that both architectures cannot determine whether to interpolate an ad (or not) into a given rec-list. Thus, in this paper, we design a new **DE**ep reinforcement learning framework with a novel DQN architecture for online **A**dvertising in **R**ecommender systems (DEAR), which can determine the aforementioned three tasks simultaneously with reasonable time complexity. We summarize our major contributions as follows:

- We identify the phenomena of online advertising with recommendations and provide a principled approach for better advertising strategy;
- We propose a deep reinforcement learning based framework DEAR and a novel Q-network architecture, which can simultaneously determine whether to interpolate an ad, the optimal location and which ad to interpolate;
- We demonstrate the effectiveness of the proposed framework in real-world short video site.

Problem Statement

In this paper, we study the advertising problem within a rec-list as a Markov Decision Process (MDP), in which an Advertising-Agent (AA) interacts with environment $\mathcal{E}$ (or users) by sequentially interpolating ads into a sequence of rec-lists over time, so as to maximize the cumulative reward from the environment. Formally, the MDP consists of a tuple of five elements $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:

- **State space** $\mathcal{S}$: A state $s_t \in \mathcal{S}$ is defined as a user's browsing history before time t and the information of current request at time t. More specifically, a state s_t consists of a user's recommendation and ad browsing history, the rec-list and contextual information of current request.
- **Action space** $\mathcal{A}$: The action $a_t = (a_t^{ad}, a_t^{loc}) \in \mathcal{A}$ of AA is to determine three internally related tasks, i.e., whether interpolate an ad in current rec-list (that is considered in a_t^{loc}, more details are presented in following sections); if yes, the AA needs to choose a specific ad a_t^{ad*} and interpolate it into the optimal location a_t^{loc*} in the rec-list. Without the loss of generality, we assume that the AA could interpolate at most one ad into a rec-list, but it is straightforward to extend it with multiple ads.
- **Reward** $\mathcal{R}$: After the AA taking an action a_t at the state s_t, i.e., (not) interpolating an ad into a rec-list, a user browses this mixed rec-ad list and provides her feedback. The AA will receive the immediate reward $r(s_t, a_t)$ based on user's feedback. The reward $r(s_t, a_t)$ is two-fold: (i) the income of ad that depends on the quality of the ad, and (ii) the influence of an ad on the user experience.
- **Transition probability** $\mathcal{P}$: Transition probability $p(s_{t+1}|s_t, a_t)$ defines the state transition from s_t to s_{t+1} after taking action a_t. We assume that the MDP satisfies $p(s_{t+1}|s_t, a_t, ..., s_1, a_1) = p(s_{t+1}|s_t, a_t)$.
- **Discount factor** γ: Discount factor $\gamma \in [0, 1]$ is introduced to measure the present value of future reward. When $\gamma = 1$, all future rewards will be fully counted into current action; on the contrary, when $\gamma = 0$, only the immediate reward will be considered.

With the above-mentioned notations and definitions, the problem of ad interpolation into recommendation lists can be formally defined as follows: *Given the historical MDP, i.e., $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, the goal is to find an advertising policy $\pi : \mathcal{S} \to \mathcal{A}$, which can maximize the cumulative reward from users*, i.e., maximizing the income of ads and minimizing the negative influence on user experience.

The Proposed Framework

In this section, we will propose a deep reinforcement learning framework for online advertising in recommender systems. To be more specific, we will first propose a novel DQN architecture, which could tackle the aforementioned three tasks simultaneously. Then, we discuss how to train the framework via offline users' behavior log.

The DQN Architecture for Online Advertising

As aforementioned the online advertising in recommender system problem is challenging because (i) the action of the advertising agent (AA) is complex which consists of three sub-actions, i.e., whether interpolate an ad into current rec-list, if yes, which ad is optimal and where is the best location; (ii) the three sub-actions are internally related, i.e.,

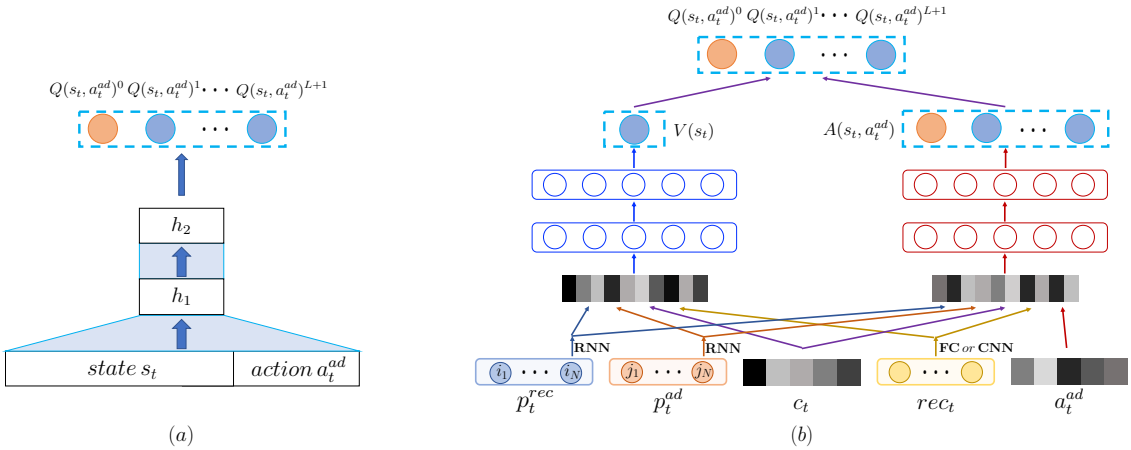

Figure 3: (a) Overview of the proposed DQN architecture. (b) The detailed architecture of the proposed DQN.

when the AA decides to interpolate an ad, the candidate ads and locations are interactive to maximize the reward, which prevents traditional DQN architectures from being employed in online advertising systems; and (iii) the AA should simultaneously maximize the income of ads and minimize the negative influence of ads on user experience. To address these challenges, we propose a deep reinforcement learning framework with a novel Deep Q-network architecture. In the following, we first introduce the processing of state and action features, and then we illustrate the proposed DQN architecture with optimization algorithm.

The Processing of State and Action Features The state s_t consists of a user's rec/ads browsing history, the contextual information and rec-list of current request. The recommendation (or ad) browsing history is a sequence of recommendations (or ads) the user has browsed. We leverage two RNNs with Gated Recurrent Units (GRU) to capture users' sequential preference of recommendations and ads separately. The inputs of RNN are the features of user's recently browsed recommendations (or ads), while we use the final hidden state of RNN as the representation of user's dynamic preference of recommendations p_t^{rec} (or ads p_t^{ad}). Here we leverage GRU rather than Long Short-Term Memory (LSTM) because of GRU's simpler architecture and fewer parameters.

The contextual information feature c_t of current user request consists of information such as the OS (ios or android), app version and feed type (swiping up/down the screen) of user's current request. Next, we represent the rec-list of current request by the concatenated features of L recommended items that will be displayed in current request, and we transform them into a low-dimensional dense vector $rec_t = tanh(W_{rec} concat(rec_1, \cdots, rec_L) + b_{rec})$. Note that other architectures like CNN for NLP (Kim 2014) can also be leveraged. Finally, we get a low-dimensional representation of state s_t by concatenating p_t^{rec}, p_t^{ad}, c_t and rec_t:

$$s_t = concat(p_t^{rec}, p_t^{ad}, c_t, rec_t) \tag{1}$$

For the transition from s_t to s_{t+1}, the recommendations and ads browsed at time t will be added into browsing history to generate p_{t+1}^{rec} and p_{t+1}^{ad}, c_{t+1} depends on user's behavior at time $t+1$, and rec_{t+1} comes from the recommendation system. For the action $a_t = (a_t^{ad}, a_t^{loc}) \in \mathcal{A}$, a_t^{ad} is the feature of a candidate ad, and $a_t^{loc} \in \mathbb{R}^{L+1}$ is the location to interpolate the selected ad (given a list of L recommendations, there exist $L+1$ possible locations). Next, we will elaborate the architecture of the proposed DQN architecture.

The Proposed DQN Architecture Given the state s_t, the action a_t of AA consists three sub-actions, i.e., whether to interpolate an ad, if yes, (ii) where is the optimal location and (iii) which ad is optimal.

We first consider to simultaneously tackle the sub-action (ii) and (iii). In other words, we aim to estimate the Q-values of all possible locations a_t^{loc} for any given candidate ad a_t^{ad}. To incorporate these two sub-actions into one framework, we proposed a novel DQN architecture, as illustrated in Figure 3(a), which is on the top of the two conventional Deep Q-network architectures shown in Figure 2. The inputs are the representations of state s_t and any candidate ad a_t^{ad}, while the output is the action-value (Q-value) corresponding to $L+1$ locations. In this way, the proposed DQN architecture could take advantage of both two traditional DQN architectures, which could simultaneously evaluate the Q-values of two types of internally related sub-actions, i.e., evaluating the Q-values of all possible locations for an ad.

To incorporate the first sub-action (whether to interpolate an ad or not) into the above DQN architecture, we consider not interpolating an ad as a special $location\,0$, and extend the length of output layer from $L+1$ to $L+2$, where $Q(s_t, a_t^{ad})^0$ corresponds to the Q-value of not incorporating an ad into current rec-list. Therefore, the proposed DQN architecture could take the three sub-actions simultaneously, where the Q-value depends on the combination of ad-location pair; and when $Q(s_t, a_t^{ad})^0$ of any candidate ads corresponds to the maximal Q-value, the AA will not interpolate an ad into current rec-list.

The detailed DQN architecture is illustrated in Figure 3(b). On one hand, whether to interpolate an ad into current rec-list is mainly impacted by the state s_t (the browsing history, the contextual information and especially the quality of current rec-list), e.g., if a user has good experience for current rec-list, the advertising agent may prefer to in-

terpolate an ad into the current rec-list; while if a user has bad experience for current rec-list, the user has high possibility to leave, then the AA will not insert an ad to increase this possibility. On the other hand, the reward for choosing an ad and location is closely related to all the features (both current rec-list and the ads). According to this observation, we divide the Q-function into value function $V(s_t)$, which is determined by the state features, and the advantage function $A(s_t, a_t)$, which is determined by the features of both state and action (Wang, Freitas, and Lanctot 2015).

Discussion There exist two classic DQN architectures as illustrated in Figure 2, where (i) the left one takes state as input and outputs Q-values of all actions, and (ii) the right one takes a state-action pair and outputs the Q-value of this pair. These two conventional architectures can only evaluate Q-values for one level of actions, e.g., the agent in Maze environment can only choose to go up, down, left, or right (Brockman et al. 2016). Compared with these two traditional architectures, the proposed DEAR takes a state-action pair of one level of actions, and outputs the Q-values corresponding to the combination of this state-action pair and another level of actions. Hierarchical reinforcement learning (HRL) architectures like (Kulkarni et al. 2016) can also handle multiple levels of tasks. However, HRL frameworks suffer from the instability problem when training multiple levels jointly (Nachum et al. 2018). To the best of our knowledge, the proposed DEAR architecture is the first individual DQN architecture that can evaluate the Q-values of two or more levels of internally related actions simultaneously with reasonable time complexity. This design is general which has many other possible applications. For example, in Maze environment the input of DEAR can be the pair of agent's location (state) and the direction to go (action), then the DEAR can output the Q-values corresponding to the location, direction and how many steps to go in this direction (another level of related actions).

The Reward Function

After the AA executing an action a_t at the state s_t, i.e., interpolating an ad into a rec-list (or not), a user browses this mixed rec-ad list and provides her feedback. In online advertising with normal recommendation problem, the AA aims to simultaneously maximize the income of ads and minimize the negative influence of ads on user experience (i.e. to optimize user experience). Thus the immediate reward $r_t(s_t, a_t)$ is two-fold: (i) the income of ad r_t^{ad}, and (ii) the user experience r_t^{ex}.

In practical platforms, the major risk of interpolating ads improperly or too frequently is that user will leave the platforms. Thus, user experience is measured by whether she/he will leave the platform after browsing the current rec-ad list, and we have:

$$r_t^{ex} = \begin{cases} 1 & continue \\ -1 & leave \end{cases} \tag{2}$$

in other word, the AA will receive a positive reward if the user continue to browse the next list, otherwise negative reward. Then, we design the reward function as follows:

$$r_t(s_t, a_t) = r_t^{ad} + \alpha \cdot r_t^{ex} \tag{3}$$

Algorithm 1 Off-policy Training of DEAR Framework.

Initialize the capacity of replay buffer $\mathcal{D}$
Initialize action-value function Q with random weights
for session $= 1, M$ **do**
  Initialize state s_0 from previous sessions
  for $t = 1, T$ **do**
    Observe state $s_t = concat(p_t^{rec}, p_t^{ad}, c_t, rec_t)$
    Execute action a_t following off-policy $b(s_t)$
    Calculate reward $r_t = r_t^{ad} + \alpha r_t^{ex}$ from offline log
    Update state to s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) into the replay buffer $\mathcal{D}$
    Sample mini-batch of transitions (s, a, r, s') from the replay buffer $\mathcal{D}$
    Set $y = \begin{cases} r & \text{terminal } s' \\ r + \gamma \max_{a'} Q(s', a'; \theta) & \text{non} - \text{terminal } s' \end{cases}$
    Minimize $\big(y - Q(s, a; \theta)\big)^2$ according to Eq.(6)
  end for
end for

where the r_t^{ad} is the income of ad, which is a positive value if interpolate an ad, otherwise 0. The hyper-parameter α controls the importance of the second term, which measures the influence of an ad on user experience. Based on the reward function, optimal action-value function $Q^*(s_t, a_t)$, which has the maximum expected return achievable by the optimal policy, should follow the Bellman equation (Bellman 2013) as:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\big[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t\big], \tag{4}$$

where the operation $\max_{a_{t+1}}$ needs to look through all candidate ads $\{a_{t+1}^{ad}\}$ (input) and all locations $\{a_{t+1}^{loc}\}$ (output).

The Optimization Task

The Deep Q-network, i.e., action-value function $Q(s_t, a_t)$, can be optimized by minimizing a sequence of loss functions $L(\theta)$ as:

$$L(\theta) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1}}\big(y_t - Q(s_t, a_t; \theta)\big)^2, \tag{5}$$

where $y_t = \mathbb{E}_{s_{t+1}}[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1},\ a_{t+1}; \theta^T) | s_t, a_t]$ is the target for the current iteration. We introduce separated evaluation and target networks (Mnih et al. 2013) to help smooth the learning and avoid the divergence of parameters, where θ represents all parameters of the evaluation network, and the parameters of the target network θ^T are fixed when optimizing the loss function $L(\theta)$. The derivatives of loss function $L(\theta)$ with respective to parameters θ are presented as follows:

$$\nabla_\theta L(\theta) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1}}\big(y_t - Q(s_t, a_t; \theta)\big) \nabla_\theta Q(s_t, a_t; \theta). \tag{6}$$

where $y_t = \mathbb{E}_{s_{t+1}}[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1},\ a_{t+1}; \theta^T) | s_t, a_t]$, and $\max_{a_{t+1}}$ will look through the candidate ad set $\{a_{t+1}^{ad}\}$

Table 1: Statistics of the dataset.

session	user	normal video	ad video
1,000,000	188,409	17,820,066	10,806,778
session dwell time	session length	session ad revenue	rec-list with ad
17.980 min	55.032 videos	0.667	55.23%

Algorithm 2 Online Test of the DEAR Framework.

1: Initialize the proposed DQN with well trained weights
2: **for** session $= 1, M$ **do**
3: Initialize state s_0
4: **for** $t = 1, T$ **do**
5: Observe state $s_t = concat(p_t^{rec}, p_t^{ad}, c_t, rec_t)$
6: Execute action a_t following $Q^*(s_t, a_t)$
7: Observe rewards $r_t(s_t, a_t)$ from user
8: Update the state from s_t to s_{t+1}
9: **end for**
10: **end for**

and all locations $\{a_{t+1}^{loc}\}$ (including the location that represents not interpolating an ad). Note that a recall mechanism is employed by the platform to select a subset of ads that may generate maximal revenue, and filter out ads that run out of their budget (RTB) or have fulfilled the guaranteed delivery amount (GD). In this paper, we mainly focus on the income of platform and user experience.

Off-policy Training Task

We train the proposed framework based on users' offline log, which records the interaction history between behavior policy $b(s_t)$ (the advertising strategy in use) and users' feedback. Our AA takes the action based on the off-policy $b(s_t)$ and obtains the feedback from the offline log. We present our off-policy training algorithm in details in Algorithm 1.

In each iteration of a training session, there are two stages. For storing transitions stage: given the state s_t (line 6), the AA takes action a_t according to the behavior policy $b(s_t)$ (line 7), which follows a standard off-policy way (Degris, White, and Sutton 2012); then the AA observes the reward r_t from offline log (line 8) and updates the state to s_{t+1} (line 9); and finally the AA stores transition (s_t, a_t, r_t, s_{t+1}) into replay buffer $\mathcal{D}$ (line 10). For model training stage: the AA samples minibatch of transitions (s, a, r, s') from replay buffer $\mathcal{D}$ (line 11), and then updates the parameters according to Equation (6) (lines 13). Note that in line 7, when the behavior policy $b(s_t)$ decides not to interpolate an ad, we use an all-zero vector as a_t^{ad}.

Online Test Task

The online test algorithm is presented in Algorithm 2, which is similar to the transition generating stage in Algorithm 1. In each iteration of the session, given the current state s_t (line 5), the AA decides to interpolate an ad into the rec-list (or not) by the well-trained advertising policy $Q^*(s_t, a_t)$ (line 6), then the target user browses the mixed rec-ad list and provides her/his feedback (line 7). Finally, the AA updates the state to s_{t+1} (line 8) and goes to the next iteration.

Experiments

In this section, we conduct extensive experiments on a real short video site to evaluate the effectiveness of the proposed framework. We mainly focus on three questions: (i) how the proposed framework performs compared to representative baselines; (ii) how the components in the framework contribute to the performance; and (iii) how the hyper-parameters impact the performance. We first introduce experimental settings. Then we seek answers to the above three questions.

Experimental Settings

Since there are no public dataset consists of both recommended and advertised items, we train our model on the dataset of March, 2019 collected in a short video site, where there are two types of videos, i.e., normal videos (recommended items) and ad videos (advertised items). The features for a normal video contain: id, like score, finish score, comment score, follow score and group score, where the scores are predicted by the platform. The features for an ad video consist of: id, image size, pricing, hidden-cost, rc-preclk and recall-preclk, where the last four are predicted by the platform. Note that (i) the predicted features are widely and successfully used in many applications such as recommendation and advertising in the platform, (ii) we discretize each feature as a one-hot vector, and (iii) the same features are used by baselines for a fair comparison. We collect 1,000,000 sessions in temporal order to train the proposed framework via an off-policy manner. More statistics about the dataset are shown in Table 1.

For the architecture of DEAR, the dimension of ad and normal videos' features is 60, the length of a rec-list is $L = 6$, and the size of ad candidate set for a request is $5 \sim 10$. For a given session, the initial user browsing history is collected from the first three requests of the session. The dimensions of $p_t^{rec}, p_t^{ad}, c_t, rec_t, a_t^{ad}$ are $64, 64, 13, 360, 60$. We leverage two 2-layer neural network to generate $V(s_t)$ and $A(s_t, a_t^{ad})$, respectively. The length of the output layer is $L + 2 = 8$, i.e., there are 8 possible locations including the one representing not to interpolate an ad. We set the discounted factor $\gamma = 0.95$, and the size of replay buffer is 10,000. For the hyper-parameters of the proposed framework such as α, we select them via cross-validation. Correspondingly, we also do parameter-tuning for baselines for a fair comparison. We will discuss more details about hyper-parameter selection for the DEAR framework in the following subsections. Reward r_t^{ad} is the revenue of ad videos, and

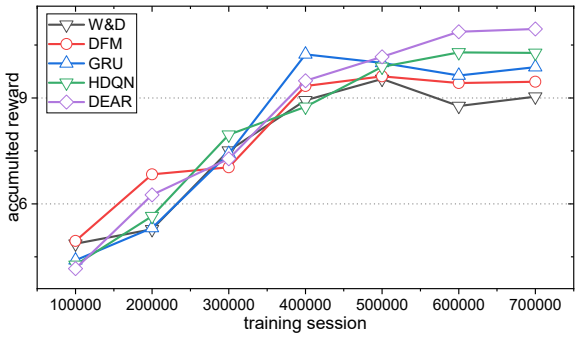

Figure 4: Overall performance comparison.

r_t^{ex} is 1 if user continue to browse next list and 0 otherwise. To measure the online performance, we leverage the *accumulated rewards* in the session $R = \sum_1^T r_t$ as the metric.

Overall Performance Comparison

We compare the proposed framework with the following representative baseline methods:

- **W&D** (Cheng et al. 2016): This baseline is a wide & deep model for jointly training feed-forward neural networks with embeddings and linear model with feature transformations for generic recommender systems with sparse inputs. We further augment it W&D to predict whether interpolate an ad and estimate the CTR of ads. W&D is the behavior policy $b(s_t)$ in use of the video platform.
- **DFM** (Guo et al. 2017): DeepFM is a deep neural network model that integrates the architectures of factorization-machine (FM) and wide & deep model. It models low-order feature interactions like FM and models high-order feature interactions like W&D.
- **GRU** (Hidasi et al. 2015): GRU4Rec utilizes RNN with Gated Recurrent Units (GRU) to predict what user will click/order next based on the clicking/ordering histories. We also augment it for ads interpolation.
- **HDQN** (Kulkarni et al. 2016): This baseline is a hierarchical DQN framework where the high-level DQN determines the locations (including the location of not interpolating ad), and the low-level DQN selects a specific ad.

The results are shown in Figure 4. We make the following observations:

1. The DFM achieves better performance than W&D, where DeepFM can be trained end-to-end without any feature engineering, and its wide part and deep part share the same input and also the embedding vector.
2. GRU outperforms W&D and DFM, since GRU can capture the temporal sequence of user behaviors within one session, while W&D and DFM neglect it.
3. HDQN performs better than GRU, since GRU is designed to maximize the immediate reward of each request, while HDQN aims to maximize the rewards in the long run. This result suggests that introducing reinforcement learning can improve the long-term performance of online recommendation and advertising.

Table 2: Component study results.

	reward	improvement	p-value
DEAR-1	9.936	10.32%	0.000
DEAR-2	10.02	9.056%	0.000
DEAR-3	10.39	5.495%	0.001
DEAR-4	10.57	3.689%	0.006
DEAR	**10.96**	-	-

4. DEAR outperforms HDQN, since HRL frameworks like HDQN are not stable when multiple levels are jointly trained by an off-policy manner (Nachum et al. 2018).

To sum up, DEAR outperforms representative baselines, which demonstrates its effectiveness in online advertising. Note that the improvement of DEAR is significant ($p - value < 0.01$), we omit the results of hypothesis test because of the space limitation.

Component Study

To answer the second question, we systematically eliminate the corresponding components of DEAR by defining the following variants:

- **DEAR-1**: This variant shares the same architectures with the proposed model, while we train the framework through a supervised learning manner.
- **DEAR-2**: This variant is to evaluate the effectiveness of RNNs, hence we replace each RNN by two fully-connected layers (FCNs), concatenate recommended or advertised items as one vector and feed it into the corresponding FCN.
- **DEAR-3**: This baseline leverages the DQN architecture in Figure 2(b) with an additional input, which represents the location by a one-hot vector.
- **DEAR-4**: The architecture of this variant does not divide the Q-function into the value function $V(s)$ and the advantage function $A(s, a)$ for the AA.

The results are shown in Table 2. It can be observed:

1. DEAR-1 validates the effectiveness of introducing reinforcement learning for online advertising.
2. DEAR-2 demonstrates that capture user's sequential preference over recommended and advertised items can boost the performance.
3. DEAR-3 validates the effectiveness of the proposed DEAR architecture over conventional DQN architecture that takes an ad a_t^{ad} as input while outputs the Q-value corresponding to all possible locations $\{a_t^{loc}\}$ for the given ad a_t^{ad}.
4. DEAR-4 proves that whether interpolate an ad into rec-list is mainly depended on state (especially the quality of rec-list), while the reward for selecting an ad and location depends on both s_t and a_t (ad). Thus dividing $Q(s_t, a_t)$ into the value function $V(s_t)$ and the advantage function $A(s_t, a_t)$ can improve the performance.

In summary, introducing RL and appropriately designing neural network architecture can boost the performance.

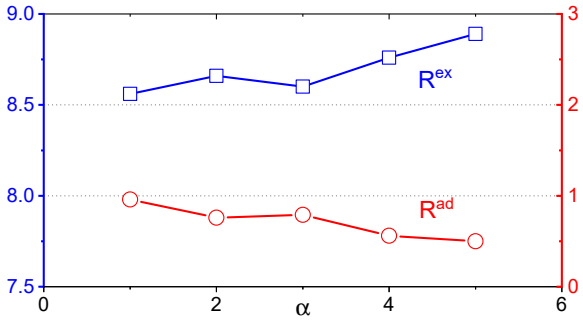

Figure 5: Parameter sensitivity analysis.

Parameter Sensitivity Analysis

In this section, we investigate how the proposed framework DEAR performs with the changes of α in Equation (3), while fixing other parameters. We select the accumulated $R^{ad} = \sum_1^T r_t^{ad}$ and accumulated $R^{ex} = \sum_1^T r_t^{ex}$ of the whole session as the metrics to evaluate the performance.

Figure 5 demonstrates the parameter sensitivity of α. We find that with the increase of α, the performance of R^{ex} improves, while R^{ad} decreases. On one hand, when we increase the importance of the second term in Equation (3), the AA tends to interpolate fewer ads or select the ads that will not decrease user's experience, although they may generate suboptimal revenue. On the other hand, when we decrease the importance of the second term of Equation (3), the AA prefers to interpolate more ads or choose the ads that will lead to maximal revenue, while ignoring the negative impact of ad on user's experience.

Related Work

In this section, we briefly review works related to our study. In general, the related work can be mainly grouped into the following categories.

The first category related to this paper is guaranteed delivery, where ads that share a single idea and theme are grouped into campaigns, and are charged on a pay-per-campaign basis for the pre-specified number of deliveries (click or impressions) (Salomatin, Liu, and Yang 2012). Most popular GD (Guaranteed Delivery) solutions are based on offline optimization algorithms, and then adjusted for online setup. However, deriving the optimal strategy to allocate impressions is challenging, especially when the environment is unstable in real-world applications. In (Wu et al. 2018a), a multi-agent reinforcement learning (MARL) approach is proposed to derive cooperative policies for the publisher to maximize its target in an unstable environment. They formulated the impression allocation problem as an auction problem where each contract can submit virtual bids for individual impressions. With this formulation, they derived the optimal impression allocation strategy by solving the optimal bidding functions for contracts.

The second category related to this paper is RTB, which allows an advertiser to submit a bid for each individual impression in a very short time frame. Ad selection task is typically modeled as multi-armed bandit (MAB) problem with the setting that samples from each arm are iid, feedback is immediate and rewards are stationary (Yang and Lu 2016; Nuara et al. 2018; Gasparini et al. 2018; Tang et al. 2013; Xu, Qin, and Liu 2013; Yuan, Wang, and van der Meer 2013; Schwartz, Bradlow, and Fader 2017). The problem of multi-armed bandits with budget constraints and variable costs is studied in (Ding et al. 2013). In this case, pulling the arms of bandit will get random rewards with random costs, and the algorithm aims to maximize the long-term reward by pulling arms with a constrained budget. Under the MAB setting, the bid decision is considered as a static optimization problem of either treating the value of each impression independently or setting a bid price to each segment of ad volume. However, the bidding for a given ad campaign would repeatedly happen during its life span before the budget running out. Thus, the MDP setting has also been studied (Cai et al. 2017; Wang et al. 2018a; Zhao et al. 2018b; Rohde et al. 2018; Wu et al. 2018b; Jin et al. 2018). A model-based reinforcement learning framework is proposed to learn bid strategies in RTB advertising (Cai et al. 2017), where neural network is used to approximate the state value, which can better deal with the scalability problem of large auction volume and limited campaign budget. A model-free deep reinforcement learning method is proposed to solve the bidding problem with constrained budget (Wu et al. 2018b): the problem is modeled as a λ-control problem, and RewardNet is designed for generating rewards to solve reward design trap, instead of using the immediate reward. A multi-agent bidding model takes the other advertisers' bidding in the system into consideration, and a clustering approach is introduced to solve the large number of advertisers challenge (Jin et al. 2018).

The third category related to this paper is reinforcement learning based recommender systems, which typically consider the recommendation task as a Markov Decision Process (MDP), and model the recommendation procedure as sequential interactions between users and recommender system (Zhao et al. 2019b; Zhao et al. 2018a). Practical recommender systems are always with millions of items (discrete actions) to recommend (Zhao et al. 2016; Guo et al. 2016). Thus, most RL-based models will become inefficient since they are not able to handle such a large discrete action space. A Deep Deterministic Policy Gradient (DDPG) algorithm is introduced to mitigate the large action space issue in practical RL-based recommender systems (Dulac-Arnold et al. 2015). To avoid the inconsistency of DDPG and improve recommendation performance, a tree-structured policy gradient is proposed in (Chen et al. 2018a). Biclustering technique is also introduced to model recommender systems as grid-world games so as to reduce the state/action space (Choi et al. 2018). To solve the unstable reward distribution problem in dynamic recommendation environments, approximate regretted reward technique is proposed with Double DQN to obtain a reference baseline from individual customer sample (Chen et al. 2018c). Users' positive and negative feedback, i.e., purchase/click and skip behaviors, are jointly considered in one framework to boost recommendations, since both types of feedback can represent part of users' preference (Zhao et al. 2018c). Architecture aspect and formulation aspect improvement are

introduced to capture both positive and negative feedback in a unified RL framework. A page-wise recommendation framework is proposed to jointly recommend a page of items and display them within a 2-D page (Zhao et al. 2017; Zhao et al. 2018b). CNN technique is introduced to capture the item display patterns and users' feedback of each item in the page. A multi-agent model-based reinforcement learning framework (DeepChain) is proposed for the whole-chain recommendation problem (Zhao et al. 2019c), which is able to collaboratively train multiple recommendation agents for different scenarios by a model-based optimization algorithm. A user simulator RecSimu base on Generative Adversarial Network (GAN) framework is presented for RL-based recommender systems (Zhao et al. 2019a), which models real users' behaviors from users' historical logs, and tackle the two challenges: (i) the recommended item distribution is complex within users' historical logs, and (ii) labeled training data from each user is limited. In the news feed scenario, a DQN based framework is proposed to handle the challenges of conventional models, i.e., (1) only modeling current reward like CTR, (2) not considering click/skip labels, and (3) feeding similar news to users (Zheng et al. 2018). An RL framework for explainable recommendation is proposed in (Wang et al. 2018b), which can explain any recommendation model and can flexibly control the explanation quality based on the application scenario. A policy gradient-based top-K recommender system for YouTube is developed in (Chen et al. 2018b), which addresses biases in logged data through incorporating a learned logging policy and a novel top-K off-policy correction. Other applications includes sellers' impression allocation (Cai et al. 2018a), fraudulent behavior detection (Cai et al. 2018b), and user state representation (Liu et al. 2018).

Conclusion

In this paper, we propose a deep reinforcement learning framework DEAR with a novel Deep Q-network architecture for online advertising in recommender systems. It is able to (i) determine three internally related actions at the same time, i.e., whether to interpolate an ad in a rec-list or not, if yes, which is the optimal ad and location to interpolate; and (ii) simultaneously maximize the revenue of ads and minimize the negative influence of ads on user experience. It is worth to note that the proposed DQN architecture can take advantage of two conventional DQN architectures, which can evaluate the Q-value of two or more kinds of related actions simultaneously. We evaluate our framework with extensive experiments based on a short video site. The results show that our framework can significantly improve online advertising performance in recommender systems.

There are several interesting research directions. First, in addition to only optimizing advertising strategies in recommender systems, we would like to develop a framework that jointly optimizes advertising and recommending strategies simultaneously. Second, the proposed framework DEAR is quite general for evaluating the Q-value of two or more types of internally related actions, we would like to investigate more applications beyond online advertising, such as recommendations and video games.

References

[Bellman 2013] Bellman, R. 2013. *Dynamic programming*. Courier Corporation.

[Brockman et al. 2016] Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

[Cai et al. 2017] Cai, H.; Ren, K.; Zhang, W.; Malialis, K.; Wang, J.; Yu, Y.; and Guo, D. 2017. Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 661–670. ACM.

[Cai et al. 2018a] Cai, Q.; Filos-Ratsikas, A.; Tang, P.; and Zhang, Y. 2018a. Reinforcement mechanism design for e-commerce. In *Proceedings of the 2018 World Wide Web Conference*, 1339–1348. International World Wide Web Conferences Steering Committee.

[Cai et al. 2018b] Cai, Q.; Filos-Ratsikas, A.; Tang, P.; and Zhang, Y. 2018b. Reinforcement mechanism design for fraudulent behaviour in e-commerce. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

[Chen et al. 2018a] Chen, H.; Dai, X.; Cai, H.; Zhang, W.; Wang, X.; Tang, R.; Zhang, Y.; and Yu, Y. 2018a. Large-scale interactive recommendation with tree-structured policy gradient. *arXiv preprint arXiv:1811.05869*.

[Chen et al. 2018b] Chen, M.; Beutel, A.; Covington, P.; Jain, S.; Belletti, F.; and Chi, E. 2018b. Top-k off-policy correction for a reinforce recommender system. *arXiv preprint arXiv:1812.02353*.

[Chen et al. 2018c] Chen, S.-Y.; Yu, Y.; Da, Q.; Tan, J.; Huang, H.-K.; and Tang, H.-H. 2018c. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1187–1196. ACM.

[Cheng et al. 2016] 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. ACM.

[Choi et al. 2018] Choi, S.; Ha, H.; Hwang, U.; Kim, C.; Ha, J.-W.; and Yoon, S. 2018. Reinforcement learning based recommender system using biclustering technique. *arXiv preprint arXiv:1801.05532*.

[Degris, White, and Sutton 2012] Degris, T.; White, M.; and Sutton, R. S. 2012. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*.

[Ding et al. 2013] Ding, W.; Qin, T.; Zhang, X.-D.; and Liu, T.-Y. 2013. Multi-armed bandit with budget constraint and variable costs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

[Dulac-Arnold et al. 2015] Dulac-Arnold, G.; Evans, R.; van Hasselt, H.; Sunehag, P.; Lillicrap, T.; Hunt, J.; Mann, T.; Weber, T.; Degris, T.; and Coppin, B. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

[Feng et al. 2018] Feng, J.; Li, H.; Huang, M.; Liu, S.; Ou, W.; Wang, Z.; and Zhu, X. 2018. Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning. In *Proceedings of the 2018 World Wide Web Conference*, 1939–1948. International World Wide Web Conferences Steering Committee.

[Gasparini et al. 2018] Gasparini, M.; Nuara, A.; Trovò, F.; Gatti, N.; and Restelli, M. 2018. Targeting optimization for internet advertising by learning from logged bandit feedback. In *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

[Guo et al. 2016] Guo, H.; Li, X.; He, M.; Zhao, X.; Liu, G.; and Xu, G. 2016. Cosolorec: Joint factor model with content, social, location for heterogeneous point-of-interest recommendation. In *International Conference on Knowledge Science, Engineering and Management*, 613–627. Springer.

[Guo et al. 2017] Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

[Hidasi et al. 2015] Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

[Jia et al. 2016] Jia, Z.; Zheng, W.; Qian, L.; Zhang, J.; and Sun, X. 2016. Efficient delivery policy to minimize user traffic consumption in guaranteed advertising.

[Jin et al. 2018] Jin, J.; Song, C.; Li, H.; Gai, K.; Wang, J.; and Zhang, W. 2018. Real-time bidding with multi-agent reinforcement learning in display advertising. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2193–2201. ACM.

[Kim 2014] Kim, Y. 2014. Convolutional neural networks for sentence classification. *Eprint Arxiv*.

[Kulkarni et al. 2016] Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, 3675–3683.

[Liu et al. 2018] Liu, F.; Tang, R.; Li, X.; Zhang, W.; Ye, Y.; Chen, H.; Guo, H.; and Zhang, Y. 2018. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027*.

[Mnih et al. 2013] 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*.

[Nachum et al. 2018] Nachum, O.; Gu, S. S.; Lee, H.; and Levine, S. 2018. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 3303–3313.

[Nuara et al. 2018] Nuara, A.; Trovo, F.; Gatti, N.; and Restelli, M. 2018. A combinatorial-bandit algorithm for the online joint bid/budget optimization of pay-per-click advertising campaigns. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Rohde et al. 2018] Rohde, D.; Bonner, S.; Dunlop, T.; Vasile, F.; and Karatzoglou, A. 2018. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720*.

[Salomatin, Liu, and Yang 2012] Salomatin, K.; Liu, T.-Y.; and Yang, Y. 2012. A unified optimization framework for auction and guaranteed delivery in online advertising. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2005–2009. ACM.

[Schwartz, Bradlow, and Fader 2017] Schwartz, E. M.; Bradlow, E. T.; and Fader, P. S. 2017. Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science* 36(4):500–522.

[Tang et al. 2013] Tang, L.; Rosales, R.; Singh, A.; and Agarwal, D. 2013. Automatic ad format selection via contextual bandits. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 1587–1594. ACM.

[Wang et al. 2018a] Wang, W.; Jin, J.; Hao, J.; Chen, C.; Yu, C.; Zhang, W.; Wang, J.; Wang, Y.; Li, H.; Xu, J.; et al. 2018a. Learning to advertise with adaptive exposure via constrained two-level reinforcement learning. *arXiv preprint arXiv:1809.03149*.

[Wang et al. 2018b] Wang, X.; Chen, Y.; Yang, J.; Wu, L.; Wu, Z.; and Xie, X. 2018b. A reinforcement learning framework for explainable recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, 587–596. IEEE.

[Wang, Freitas, and Lanctot 2015] Wang, Z.; Freitas, N. D.; and Lanctot, M. 2015. Dueling network architectures for deep reinforcement learning.

[Wu et al. 2018a] Wu, D.; Chen, C.; Yang, X.; Chen, X.; Tan, Q.; Xu, J.; and Gai, K. 2018a. A multi-agent reinforcement learning method for impression allocation in online display advertising. *arXiv preprint arXiv:1809.03152*.

[Wu et al. 2018b] Wu, D.; Chen, X.; Yang, X.; Wang, H.; Tan, Q.; Zhang, X.; Xu, J.; and Gai, K. 2018b. Budget constrained bidding by model-free reinforcement learning in display advertising. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1443–1451. ACM.

[Xu, Qin, and Liu 2013] Xu, M.; Qin, T.; and Liu, T.-Y. 2013. Estimation bias in multi-armed bandit algorithms for search advertising. In *Advances in Neural Information Processing Systems*, 2400–2408.

[Yang and Lu 2016] Yang, H., and Lu, Q. 2016. Dynamic contextual multi arm bandits in display advertisement. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 1305–1310. IEEE.

[Yuan, Wang, and van der Meer 2013] Yuan, S.; Wang, J.; and van der Meer, M. 2013. Adaptive keywords extraction with contextual bandits for advertising on parked domains. *arXiv preprint arXiv:1307.3573*.

[Zhao et al. 2016] Zhao, X.; Xu, T.; Liu, Q.; and Guo, H.

2016. Exploring the choice under conflict for social event participation. In *International Conference on Database Systems for Advanced Applications*, 396–411. Springer.

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

[Zhao et al. 2018a] Zhao, X.; Xia, L.; Tang, J.; and Yin, D. 2018a. Reinforcement learning for online information seeking. *arXiv preprint arXiv:1812.07127*.

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

[Zhao et al. 2018c] Zhao, X.; Zhang, L.; Ding, Z.; Xia, L.; Tang, J.; and Yin, D. 2018c. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1040–1048. ACM.

[Zhao et al. 2019a] Zhao, X.; Xia, L.; Ding, Z.; Yin, D.; and Tang, J. 2019a. Toward simulating environments in reinforcement learning based recommendations. *arXiv preprint arXiv:1906.11462*.

[Zhao et al. 2019b] Zhao, X.; Xia, L.; Tang, J.; and Yin, D. 2019b. Deep reinforcement learning for search, recommendation, and online advertising: a survey by xiangyu zhao, long xia, jiliang tang, and dawei yin with martin vesely as coordinator. *ACM SIGWEB Newsletter* (Spring):4.

[Zhao et al. 2019c] Zhao, X.; Xia, L.; Zhao, Y.; Yin, D.; and Tang, J. 2019c. Model-based reinforcement learning for whole-chain recommendations. *arXiv preprint arXiv:1902.03987*.

[Zheng et al. 2018] 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 on World Wide Web*, 167–176. International World Wide Web Conferences Steering Committee.