DRN: A Deep Reinforcement Learning Framework for News Recommendation

Guanjie Zheng[†], Fuzheng Zhang[§], Zihan Zheng[§], Yang Xiang[§]

Nicholas Jing Yuan[§], Xing Xie[§], Zhenhui Li[†]

Pennsylvania State University[†], Microsoft Research Asia[§]
University Park, USA[†], Beijing, China[§]
gjz5038@ist.psu.edu,{fuzzhang,v-zihzhe,yaxian,nicholas.yuan,xingx}@microsoft.com,jessieli@ist.psu.edu

ABSTRACT

In this paper, we propose a novel Deep Reinforcement Learning framework for news recommendation. Online personalized news recommendation is a highly challenging problem due to the dynamic nature of news features and user preferences. Although some online recommendation models have been proposed to address the dynamic nature of news recommendation, these methods have three major issues. First, they only try to model current reward (e.g., Click Through Rate). Second, very few studies consider to use user feedback other than click / no click labels (e.g., how frequent user returns) to help improve recommendation. Third, these methods tend to keep recommending similar news to users, which may cause users to get bored. Therefore, to address the aforementioned challenges, we propose a Deep Q-Learning based recommendation framework, which can model future reward explicitly. We further consider user return pattern as a supplement to click / no click label in order to capture more user feedback information. In addition, an effective exploration strategy is incorporated to find new attractive news for users. Extensive experiments are conducted on the offline dataset and online production environment of a commercial news recommendation application and have shown the superior performance of our methods.

KEYWORDS

Reinforcement learning, Deep Q-Learning, News recommendation

1 INTRODUCTION

The explosive growth of online content and services has provided tons of choices for users. For instance, one of the most popular online services, news aggregation services, such as Google News [15] can provide overwhelming volume of content than the amount that users can digest. Therefore, personalized online content recommendation are necessary to improve user experience.

Several groups of methods are proposed to solve the online personalized news recommendation problem, including content based methods [19, 22, 33], collaborative filtering based methods [11, 28, 34], and hybrid methods [12, 24, 25]. Recently, as an extension and integration of previous methods, deep learning models [8, 45, 52] have become the new state-of-art methods due to its capability of modeling complex user item (i.e., news) interactions. However, these methods can not effectively address the following three challenges in news recommendation.

First, the dynamic changes in news recommendations are difficult to handle. The dynamic change of news recommendation can be shown in two folds. First, news become outdated very fast. In our dataset, the average time between the time that one piece of news is published and the time of its last click is 4.1 hours. Therefore, news features and news candidate set are changing rapidly. Second, users' interest on different news might evolve during time. For instance, Figure 1 displays the categories of news that one user has read in 10 weeks. During the first few weeks, this user prefers to read about "Politics" (green bar in Figure 1), but his interest gradually moves to "Entertainment" (purple bar in Figure 1) and "Technology" (grey bar in Figure 1) over time. Therefore, it is necessary to update the model periodically. Although there are some online recommendation methods [11, 24] that can capture the dynamic change of news features and user preference through online model updates, they only try to optimize the current reward (e.g., Click Through Rate), and hence ignore what effect the current recommendation might bring to the future. An example showing the necessity of considering future is given in Example 1.1.

Example 1.1. When a user Mike requests for news, the recommendation agent foresees that he has almost the same probability to click on two pieces of news: one about a thunderstorm alert, and the other about a basketball player Kobe Bryant. However, according to Mike's reading preference, features of the news, and reading patterns of other users, our agent speculates that, after reading about the thunderstorm, Mike will not need to read news about this alert anymore, but he will probably read more about basketball after reading the news about Kobe. This suggests, recommending the latter piece of news will introduce larger future reward. Therefore, considering future rewards will help to improve recommendation performance in the long run.

Second, current recommendation methods [23, 35, 36, 43] usually only consider the click / no click labels or ratings as users' feedback. However, how soon one user will return to this service [48] will also indicate how satisfied this user is with the recommendation.

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3185994

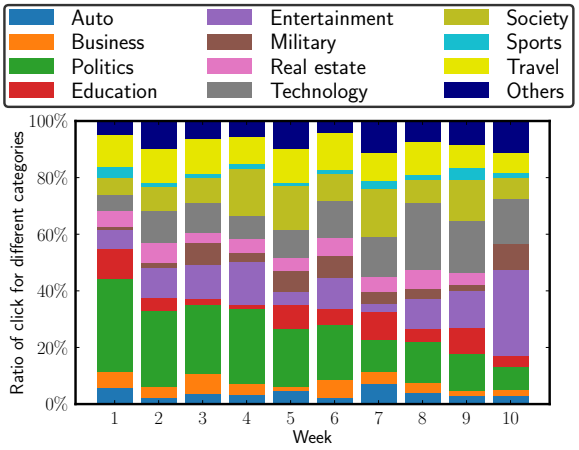


Figure 1: Distribution of clicked categories of an active user in ten weeks. User interest is evolving over time.

Nevertheless, there has been little work in trying to incorporate user return pattern to help improve recommendation.

The third major issue of current recommendation methods is its tendency to keep recommending similar items to users, which might decrease users' interest in similar topics. In the literature, some reinforcement learning methods have already proposed to add some randomness (i.e., exploration) into the decision to find new items. State-of-art reinforcement learning methods usually apply the simple ϵ-*greedy* strategy [31] or *Upper Confidence Bound* (*UCB*) [23, 43] (mainly for Multi-Armed Bandit methods). However, both strategies could harm the recommendation performance to some extent in a short period. ϵ-*greedy* strategy may recommend the customer with totally unrelated items, while *UCB* can not get a relatively accurate reward estimation for an item until this item has been tried several times. Hence, it is necessary to do more effective exploration.

Therefore, in this paper, we propose a Deep Reinforcement Learning framework that can help to address these three challenges in online personalized news recommendation. *First, in order to better model the dynamic nature of news characteristics and user preference, we propose to use Deep Q-Learning (DQN) [31] framework.* This framework can consider current reward and future reward simultaneously. Some recent attempts using reinforcement learning in recommendation either do not model the future reward explicitly (MAB-based works [23, 43]), or use discrete user log to represent state and hence can not be scaled to large systems (MDP-based works [35, 36]). In contrast, our framework uses a DQN structure and can easily scale up. *Second, we consider user return as another form of user feedback information, by maintaining an activeness score for each user.* Different from existing work [48] that can only consider the most recent return interval, we consider multiple historical return interval information to better measure the user feedback. In addition, different from [48], our model can estimate user activeness at any time (not just when user returns). This property enables the experience replay update used in DQN. *Third, we propose to apply a Dueling Bandit Gradient Descent (DBGD) method [16, 17, 49] for exploration, by choosing random item candidates in the neighborhood of the current recommender.* This exploration strategy can avoid recommending totally unrelated items and hence maintain better recommendation accuracy.

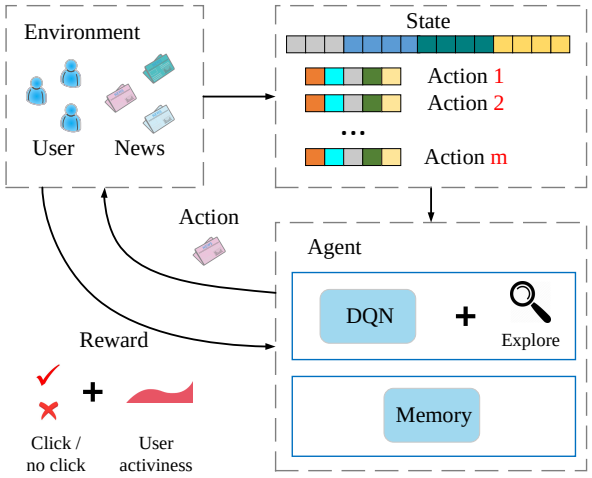


Figure 2: Deep Reinforcement Recommendation System

Our deep reinforcement recommender system can be shown as Figure 2. We follow the common terminologies in reinforcement learning [37] to describe the system. In our system, user pool and news pool make up the environment, and our recommendation algorithms play the role of agent. The state is defined as feature representation for users and action is defined as feature representation for news. Each time when a user requests for news, a state representation (i.e., features of users) and a set of action representations (i.e., features of news candidates) are passed to the agent. The agent will select the best action (i.e., recommending a list of news to user) and fetch user feedback as reward. Specifically, the reward is composed of click labels and estimation of user activeness. All these recommendation and feedback log will be stored in the memory of the agent. Every one hour, the agent will use the log in the memory to update its recommendation algorithm.

Our contribution can be summarized as below:

- We propose a reinforcement learning framework to do online personalized news recommendation. Unlike previous studies, this framework applies a DQN structure and can take care of both immediate and future reward. Although we focus on news recommendation, our framework can be generalized to many other recommendation problems.
- We consider user activeness to help improve recommendation accuracy, which can provide extra information than simply using user click labels.
- A more effective exploration method *Dueling Bandit Gradient Descent* is applied, which avoids the recommendation accuracy drop induced by classical exploration methods, e.g., ϵ-*greedy* and *Upper Confidence Bound*.
- Our system has been deployed online in a commercial news recommendation application. Extensive offline and online experiments have shown the superior performance of our methods.

The rest of the paper is organized as follows. Related work is discussed in Section 2. Then, in Section 3 we present the problem definitions. Our method is introduced in Section 4. After that, the experimental results are shown in Section 5. Finally, brief conclusions are given in Section 6.

2 RELATED WORK

2.1 News recommendation algorithms

Recommender systems [3, 4] have been investigated extensively because of its direct connection to profits of products. Recently, due to the explosive grow of online content, more and more attention has been drawn to a special application of recommendation – online personalized news recommendation. Conventional news recommendation methods can be divided into three categories. Content-based methods [19, 22, 33] will maintain news term frequency features (e.g., TF-IDF) and user profiles (based on historical news). Then, recommender will select news that is more similar to user profile. In contrast, collaborative filtering methods [11] usually make rating prediction utilizing the past ratings of current user or similar users [28, 34], or the combination of these two [11]. To combine the advantages of the former two groups of methods, hybrid methods [12, 24, 25] are further proposed to improve the user profile modeling. Recently, as an extension and integration of previous methods, deep learning models [8, 45, 52] have shown much superior performance than previous three categories of models due to its capability of modeling complex user-item relationship. *Different from the effort for modeling the complex interaction between user and item, our algorithm focuses on dealing with the dynamic nature of online news recommendation, and modeling of future reward. However, these feature construction and user-item modeling techniques can be easily integrated into our methods.*

2.2 Reinforcement learning in recommendation

2.2.1 Contextual Multi-Armed Bandit models. A group of work [5, 7, 23, 40, 44, 50] begin to formulate the problem as a Contextual Multi-Armed Bandit (MAB) problem, where the context contains user and item features. [23] assumes the expected reward is a linear function of the context. [39] uses an ensemble of bandits to improve the performance, [40] proposes a parameter-free model, and [50] addresses the time-varying interest of users. Recently, some people try to combine bandit with clustering based collaborative filtering [14], and matrix factorization [6, 21, 32, 42, 43, 51], in order to model more complex user and item relationship, and utilize the social network relationship in determining the reward function. *However, our model is significantly different from these works, because by applying Markov Decision Process, our model is able to explicitly model future rewards. This will benefit the recommendation accuracy significantly in the long run.*

2.2.2 Markov Decision Process models. There are also some literature trying to use Markov Decision Process to model the recommendation process. In contrast to MAB-based methods, MDP-based methods can not only capture the reward of current iteration, but also the potential reward in the future iterations. [26, 27, 35, 36, 38] try to model the item or n-gram of items as state (or observation in Partially Observed MDP), and the transition between items (recommendation for the next item) as the action. However, this can not scale to large dataset, because when the item candidate set becomes larger, the size of state space will grow exponentially. In addition, the state transitions data is usually very sparse, and can only be used to learn the model parameters corresponding to certain state transitions. Therefore, the model is really hard to learn. *Different from the literature, we propose a MDP framework with continuous state and action representation, which enables the system to scale up and the effective learning of model parameters by using all the state, action, reward tuples.*

3 PROBLEM DEFINITION

We define our problem as follows:

When a user u sends a news request to the recommendation agent G at time t, given a candidate set I of news, our algorithm is going to select a list L of top-k appropriate news for this user. The notations used in this paper are summarized in Table 1.

Table 1: Notations

Notation	Meaning
G	Agent
u, U	User, User set
a	Action
s	State
r	Reward
i, I	News, Candidate news pool
L	List of news to recommend
B	List of feedback from users
Q	Deep Q-Network
W	Parameters of Deep Q-Network

4 METHOD

Personalized news recommendation has attracted a lot of attention in recent years [11, 23, 45]. The current methods can be generally categorized as content based methods [19, 22, 33], collaborative filtering based methods [11, 28, 34], and hybrid methods [12, 24, 25]. Recently, many deep learning models [8, 45, 52] are further proposed in order to model more complex user item interactions. News recommendation problem becomes even more challenging when it happens in an online scenario due to three reasons. First, online learning are needed due to the highly dynamic nature of news characteristics and user preference. Second, only using click / no click labels will not capture users' full feedback towards news. Third, traditional recommendation methods tend to recommend similar items and will narrow down user's reading choices. This will make users bored and lead to decrease of user satisfaction in the long run.

To address these three challenges, we propose a DQN-based Deep Reinforcement Learning framework to do online personalized news recommendation. Specifically, we use a continuous state feature representation of users and continuous action feature representation of items as the input to a multi-layer Deep Q-Network to predict the potential reward (e.g., whether user will click on this piece of news). First, this framework can deal with the highly dynamic nature of news recommendation due to the online update of DQN. Meanwhile, DQN is different from common online methods, because of its capability to speculate future interaction between user and news. Second, we propose to combine user activeness (i.e.,

how frequent a user returns to the App after one recommendation) and click labels as the feedback from users. Third, we propose to apply *Dueling Bandit Gradient Descent* exploration strategy [16, 49] to our algorithm which can both improve recommendation diversity and avoid the harm to recommendation accuracy induced by classical exploration strategies like ϵ-*greedy* [31] and *Upper Confidence Bound* [23].

Our method is significantly different from the MAB group of methods [5, 7, 23, 40, 44, 50] due to its explicit modeling of future rewards, and different from previous MDP methods [27, 35, 36, 38] using user log due to its continuous representation of state and action, and the capability to scale to large systems.

In this section, we will first introduce the model framework in Section 4.1. Then, we will illustrate the feature construction in Section 4.2 and the deep reinforcement learning model in Section 4.3. After that, the design of user activeness consideration is discussed in Section 4.4. Finally, the exploration module is introduced in Section 4.5.

4.1 Model framework

As shown in Figure 3, our model is composed of offline part and online part. In offline stage, four kinds of features (will be discussed in Section 4.2) are extracted from news and users. A multi-layer Deep Q-Network is used to predict the reward (i.e., a combination of user-news click label and user activeness) from these four kinds of features. This network is trained using the offline user-news click logs. Then, during the online learning part, our recommendation agent G will interact with users and update the network in the following way:

(1) **PUSH:** In each timestamp (t_1, t_2, t_3, t_4, t_5, ...), when a user sends a news request to the system, the recommendation agent G will take the feature representation of the current user and news candidates as input, and generate a top-k list of news to recommend L. L is generated by combining the exploitation of current model (will be discussed in Section 4.3) and exploration of novel items (will be discussed in Section 4.5).
(2) **FEEDBACK:** User u who has received recommended news L will give their feedback B by his clicks on this set of news.
(3) **MINOR UPDATE:** After each timestamp (e.g., after timestamp t_1), with the feature representation of the previous user u and news list L, and the feedback B, agent G will update the model by comparing the recommendation performance of exploitation network Q and exploration network $\tilde{Q}$ (will be discussed in Section 4.5). If $\tilde{Q}$ gives better recommendation result, the current network will be updated towards $\tilde{Q}$. Otherwise, Q will be kept unchanged. Minor update can happen after every recommendation impression happens.
(4) **MAJOR UPDATE:** After certain period of time T_R(e.g., after timestamp t_3), agent G will use the user feedback B and user activeness stored in the memory to update the network Q. Here, we use the experience replay technique [31] to update the network. Specifically, agent G maintains a memory with recent historical click and user activeness records. When each update happens, agent G will sample a batch of records to update the model. Major update usually happens after a certain time interval, like one hour, during which thousands of recommendation impressions are conducted and their feedbacks are collected.
(5) Repeat step (1)-(4).

4.2 Feature construction

In order to predict whether user will click one specific piece of news or not, we construct four categories of features:

- **News features** includes 417 dimension one hot features that describe whether certain property appears in this piece of news, including headline, provider, ranking, entity name, category, topic category, and click counts in last 1 hour, 6 hours, 24 hours, 1 week, and 1 year respectively.
- **User features** mainly describes the features (i.e., headline, provider, ranking, entity name, category, and topic category) of the news that the user clicked in 1 hour, 6 hours, 24 hours, 1 week, and 1 year respectively. There is also a total click count for each time granularity. Therefore, there will be totally $413 \times 5 = 2065$ dimensions.
- **User news features**. These 25-dimensional features describe the interaction between user and one certain piece of news, i.e., the frequency for the entity (also category, topic category and provider) to appear in the history of the user's readings.
- **Context features**. These 32-dimensional features describe the context when a news request happens, including time, weekday, and the freshness of the news (the gap between request time and news publish time).

In order to focus on the analysis of the reinforcement learning recommendation framework, we did not try to add more features, e.g., textual features [45]. But they can be easily integrated into our framework for better performance.

4.3 Deep Reinforcement Recommendation

Considering the previous mentioned dynamic feature of news recommendation and the need to estimate future reward, we apply a Deep Q-Network (DQN) [31] to model the probability that one user may click on one specific piece of news. Under the setting of reinforcement learning, the probability for a user to click on a piece of news (and future recommended news) is essentially the reward that our agent can get. Therefore, we can model the total reward as Equation 1.

$$y_{\mathrm{s,a}} = Q(\mathrm{s,a}) = \mathrm{r}_{immediate} + \gamma \mathrm{r}_{future} \tag{1}$$

where state s is represented by context features and user features, action a is represented by news features and user-news interaction features, $\mathrm{r}_{immediate}$ represents the rewards (e.g., whether user click on this piece of news) for current situation, and r_{future} represents the agent's projection of future rewards. γ is a discount factor to balance the relative importance of immediate rewards and future rewards. Specifically, given s as the current state, we use the *DDQN* [41] target to predict the total reward by taking action a at timestamp t as in Equation 2.

$$y_{\mathrm{s},a,t} = \mathrm{r}_{\mathrm{a},t+1} + \gamma \mathrm{Q}(\mathrm{s}_{\mathrm{a},t+1}, \arg\max_{\mathrm{a}'} \mathrm{Q}(\mathrm{s}_{\mathrm{a},t+1}, \mathrm{a}'; \mathrm{W}_t); \mathrm{W}'_t) \tag{2}$$

where $\mathrm{r}_{\mathrm{a},t+1}$ represents the immediate reward by taking action a (the subscript $t+1$ is because the reward is always delayed 1

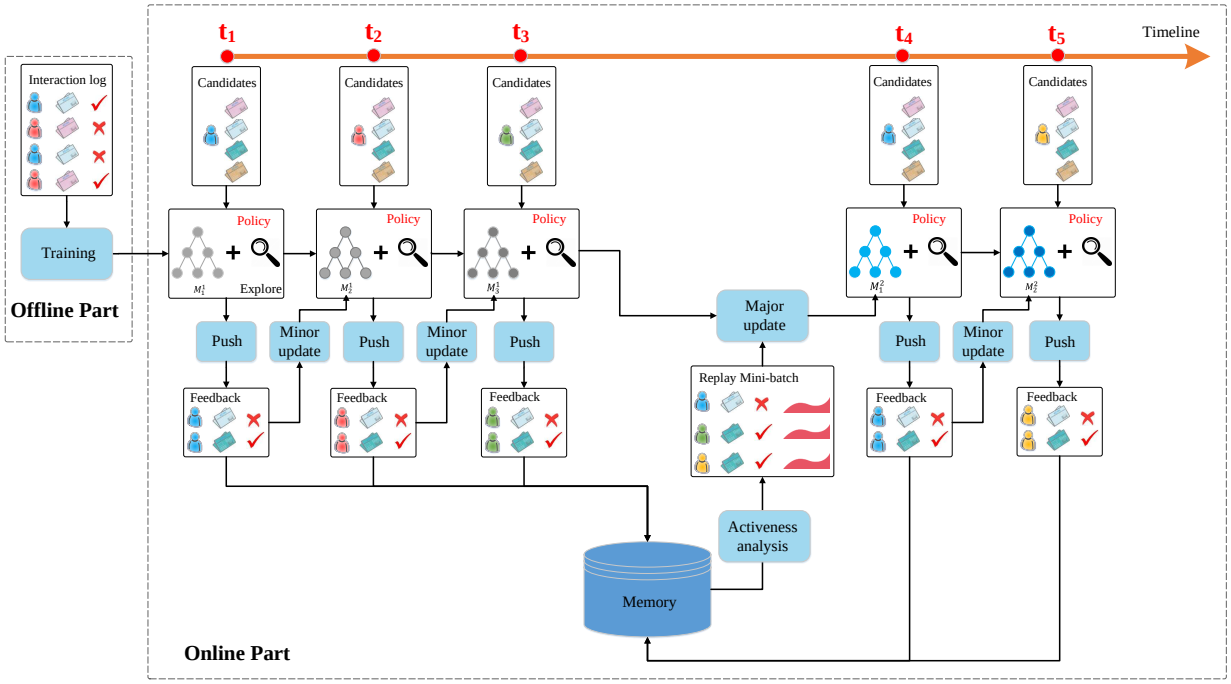


Figure 3: Model framework

timeslot than the action). Here, W_t and W'_t are two different sets of parameters of the DQN. In this formulation, our agent G will speculate the next state $s_{a,t+1}$, given action a is selected. Based on this, given a candidate set of actions $\{a'\}$, the action a' that gives the maximum future reward is selected according to parameter W_t. After this, the estimated future reward given state $s_{a,t+1}$ is calculated based on W'_t. Every a few iterations, W_t and W'_t will be switched. This strategy has been proven to eliminate the overoptimistic value estimates of Q [41]. Through this process, DQN will be able to make decision considering both immediate and future situations.

As shown in Figure 4, we feed the four categories of features into the network. **User features** and **Context features** are used as state features, while **User news features** and **Context features** are used as action features. On one hand, the reward for taking action a at certain state s is closely related to all the features. On the other hand, the reward that determined by the characteristics of the user himself (e.g., whether this user is active, whether this user has read enough news today) is more impacted by the status of the user and the context only. Based on this observation, like [47], we divide the Q-function into value function $V(s)$ and advantage function $A(s, a)$, where $V(s)$ is only determined by the state features, and $A(s, a)$ is determined by both the state features and the action features.

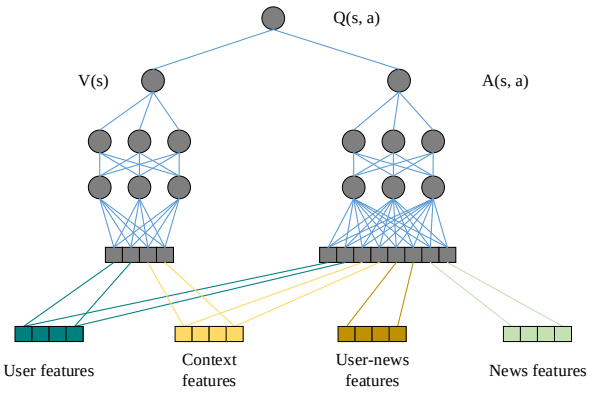


Figure 4: Q network

4.4 User Activeness

Traditional recommender systems only focus on optimizing CTR-like metrics (i.e., only utilizing click / no click labels), which only depicts part of the feedback information from users. The performance of recommendation might also influence whether users want to use the application again, i.e., better recommendation will increase the frequency for users to interact with the application. Therefore, the change of user activeness should also be considered properly.

Users request for news in a non-uniform pattern. Users usually read news for a short period (e.g., 30 minutes), during which they will request or click news with high frequency. Then they might leave the application and return to the application when they want to read more news after several hours. A user return happens when a user requests for news (users will always request for news before they click on news, therefore, user click is also implicitly considered).

We use survival models [18, 30] to model user return and user activeness. Survival analysis [18, 30] has been applied in the field of estimating user return time [20]. Suppose T is the time until next event (i.e., user return) happens, then the hazard function (i.e., instantaneous rate for the event to happen) can be defined as Equation 3 [1, 30]

$$\lambda(t) = \lim_{dt \to 0} \frac{Pr\{t \leq T < t + dt | T \geq t\}}{dt} \tag{3}$$

Then the probability for the event to happen after t can be defined as Equation 4 [1, 30]

$$S(t) = e^{-\int_0^t \lambda(x)dx} \tag{4}$$

and the expected life span T_0 can be calculated as [1, 30]

$$T_0 = \int_0^\infty S(t)dt \tag{5}$$

In our problem, we simply set $\lambda(t) = \lambda_0$, which means each user has a constant probability to return. Every time we detect a return of user, we will set $S(t) = S(t) + S_a$ for this particular user. The user activeness score will not exceed 1. For instance, as shown in Figure 5, user activeness for this specific user starts to decay from S_0 at time 0. At timestamp t_1, the user returns and this results in a S_a increase in the user activeness. Then, the user activeness continues to decay after t_1. Similar things happen at t_2, t_3, t_4 and t_5. Note that, although this user has a relatively high request frequency during t_4 to t_9, the maximum user activeness is truncated to 1.

The parameters S_0, S_a, λ_0, T_0 are determined according to the real user pattern in our dataset. S_0 is set to 0.5 to represent the random initial state of a user (i.e., he or she can be either active or inactive). We can observe the histogram of the time interval between every two consecutive requests of users as shown in Figure 6. We observe that besides reading news multiple times in a day, people usually return to the application on a daily regular basis. So we set T_0 to 24 hours. The decaying parameter λ_0 is set to $1.2 \times 10^{-5} second^{-1}$ according to Equation 4 and Equation 5. In addition, the user activeness increase S_a for each click is set to 0.32 to make sure user will return to the initial state after one daily basis request, i.e., $S_0 e^{-\lambda_0 T_0} + S_a = S_0$.

The click / no click label r_{click} and the user activeness r_{active} are combined as in Equation 6.

$$r_{total} = r_{click} + \beta r_{active} \tag{6}$$

Although we use survival models here to estimate the user activeness, other alternatives like Poisson point process [13] can also be applied and should serve similar function.

4.5 Explore

The most straightforward strategies to do exploration in reinforcement learning are ϵ-*greedy* [31] and *UCB* [23]. ϵ-*greedy* will randomly recommend new items with a probability of ϵ, while *UCB* will pick items that have not been explored for many times (because these items may have larger variance). It is evident that these trivial exploration techniques will harm the recommendation performance in a short period. Therefore, rather than doing random exploration, we apply a *Dueling Bandit Gradient Descent* algorithm [16, 17, 49] to do the exploration. Intuitively, as shown in Figure 7, the agent G is going to generate a recommendation list L using the current

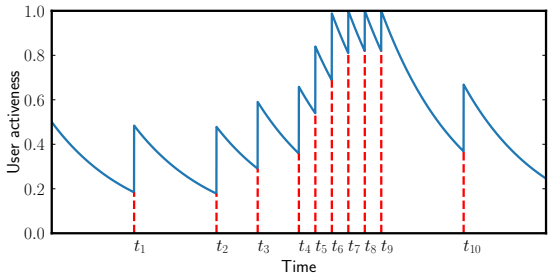


Figure 5: User activeness estimation

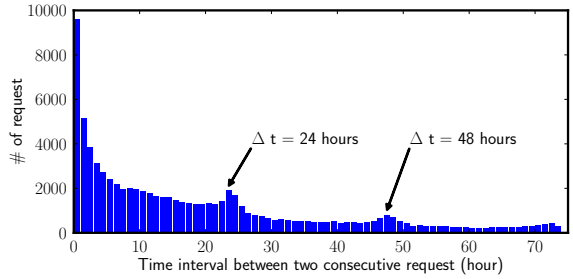


Figure 6: Time interval between two consecutive requests of one week request data.

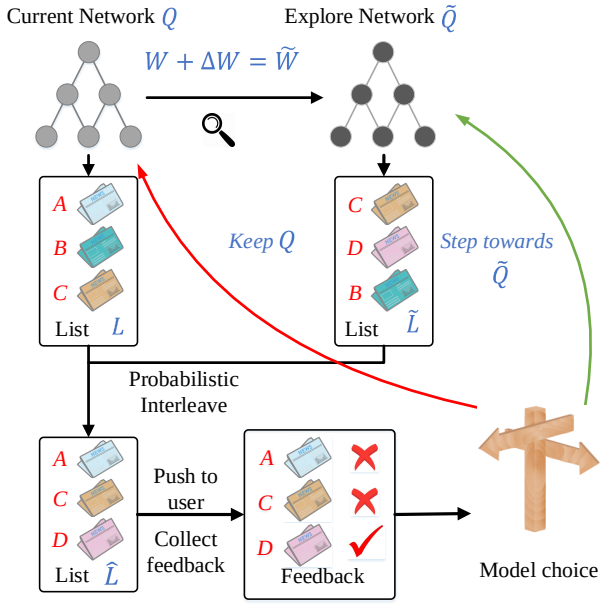


Figure 7: Exploration by *Dueling Bandit Gradient Descent*

network Q and another list $\tilde{L}$ using an explore network $\tilde{Q}$. The parameters $\tilde{W}$ of network $\tilde{Q}$ can be obtained by adding a small disturb ΔW (Equation 7) to the parameters W of the current network

Q.

$$\Delta W = \alpha \cdot rand(-1, 1) \cdot W \tag{7}$$

where α is the explore coefficient, and $rand(-1, 1)$ is a random number between -1 and 1. Then, the agent G will do a probabilistic interleave [16] to generate the merged recommendation list $\hat{L}$ using L and $\tilde{L}$. To determine the item for each position in the recommendation list $\hat{L}$, the probabilistic interleave approach basically will first randomly select between list L and $\tilde{L}$. Suppose L is selected, then an item i from L will be put into $\hat{L}$ with a probability determined by its ranking in L (items with top rankings will be selected with higher probability). Then, list $\hat{L}$ will be recommended to user u and agent G will obtain the feedback B. If the items recommended by the explore network $\tilde{Q}$ receive a better feedback, the agent G will update the network Q towards $\tilde{Q}$, with the parameters of the network being updated as Equation 8

$$W' = W + \eta \tilde{W}. \tag{8}$$

Otherwise, the agent G will keep network Q unchanged. Through this kind of exploration, the agent can do more effective exploration without losing too much recommendation accuracy.

5 EXPERIMENT

5.1 Dataset

We conduct experiment on a sampled offline dataset collected from a commercial news recommendation application and deploy our system online to the App for one month. Each recommendation algorithm will give out its recommendation when a news request arrives and user feedback will be recorded (click or not). The basic statistics for the sampled data is as in Table 2. In the first offline stage, the training data and testing data are separated by time order (the last two weeks are used as testing data), to enable the online models to learn the sequential information between different sessions better. During the second online deploying stage, we use the offline data to pre-train the model, and run all the compared methods in the real production environment.

Table 2: Statistics of the sampled dataset

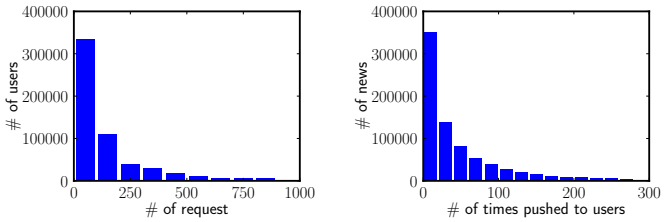
Stage	Duration	# of users	# of news
Offline stage	6 months	541,337	1,355,344
Online stage	1 month	64,610	157,088

As shown in Figure 8, the dataset is very skewed. The number of requests for each user follows a long tail distribution and most users only request news for less than 500 times. The number of times each news are pushed also follow a long tail distribution and most news are pushed to user less than 200 times.

5.2 Evaluation measures

- CTR. [10] Click through rate is calculated as Equation 9.

$$CTR = \frac{\text{number of clicked items}}{\text{number of total items}} \tag{9}$$



(a) User request counts (b) News requested counts

Figure 8: User and news basic illustration

- Precision@k [10]. Precision at k is calculated as Equation 10

$$Precision@k = \frac{\text{number of clicks in top-k recommended items}}{k} \tag{10}$$

- nDCG. We apply the standard Normalized Discounted Cumulative Gain proved in [46] as Equation 11, where r is the rank of items in the recommendation list, n is the length of the recommendation list, f is the ranking function or algorithm, y_r^f is the 1 or 0 indicating whether a click happens and $D(r)$ is the discount.

$$DCG(f) = \sum_{r=1}^{n} y_r^f D(r) \tag{11}$$

with

$$D(r) = \frac{1}{log(1 + r)} \tag{12}$$

5.3 Experiment setting

In our experiment, the parameters are determined by grid search of parameter space to find the ones with best CTR. The detailed settings are shown in Table 3.

Table 3: Parameter setting

Parameter	Setting
Future reward discount γ (Equation 1)	0.4
User activeness coefficient β (Equation 6)	0.05
Explore coefficient α (Equation 7)	0.1
Exploit coefficient η (Equation 8)	0.05
Major update period T_R (for DQN experience replay)	60 minutes
Minor update period T_D (for *DBGD*)	30 minutes

5.4 Compared methods

Variations of our model. Our basic model is named as "*DN*", which uses a dueling-structure [47] Double Deep Q-network [41] without considering future reward. Then, by adding future reward into consideration, this becomes "*DDQN*". After that, we add more components to "*DDQN*". "*U*" stands for user activeness, "*EG*" stands for ϵ-*greedy*, and "*DBGD*" stands for *Dueling Bandit Gradient Descent*.

Baseline algorithms. We compared our algorithms with following five baseline methods. All these five methods will conduct online

update during the testing stage. Some state-of-art methods can not be applied due to their inapplicability to our problem, like [43] (user graph and item graph is oversized and can not be updated incrementally), [45] (similar with *W&D* when textual features are removed), and [48] (user return is not applicable to experience replay update).

- *LR. Logistic Regression* is widely used in industry as baseline methods due to its easy implementation and high efficiency. It takes all the four categories of features as input. It is implemented using Keras [9].
- *FM* [29, 34]. *Factorization Machines* is a state-of-art context-aware recommendation methods. It takes all the four categories of features as input, use the combination of features and their interactions to do the click prediction.
- *W&D* [8]. *Wide & Deep* is a widely used state-of-art deep learning model combining the memorization (through a logistic regression on wide combinations of categorical features) and generalization (through a deep neural network embedding of the raw features) to predict the click label.
- *LinUCB* [23]. *Linear Upper Confidence Bound* [23] can select an arm (i.e., recommend a piece of news) according to the estimated upper confidence bound of the potential reward. Due to the long tail distribution of news request and click counts, we apply the same set of parameters for different news, which actually performs better than the original setting in [23] on our dataset.(An improved version of the original *LinUCB– HLinUCB* will also be compared.)
- *HLinUCB* [42] is another state-of-art bandit-based approach in recommendation problem. *Hidden Linear Upper Confidence Bound* [42] further allows learned hidden feature to model the reward. We follow the original setting of keeping different sets of parameters for different users and different news. However, under this case, only **News features** introduced in Section 4.2 can be directly applied, while the other features describing the interaction between user and news are expected to be learned in the hidden features.

For all compared algorithms, the recommendation list is generated by selecting the items with top-k estimated potential reward (for *LinUCB*, *HLinUCB* and our methods) or probability of click (for *LR*, *FM* and *W&D*) of each item.

5.5 Offline evaluation

We first compare our methods with other baselines on the offline dataset. The offline dataset is static and only certain pairs of user-news interaction have been recorded. As a result, we can not observe the change of user activeness due to different recommendation decisions. Similarly, the exploration strategy can not explore well due to the limited candidate news set (i.e., only the click labels of a few candidate news are recorded). Hence, the benefit of considering user activeness and exploration is not very evident in the offline setting. Therefore, we only show the comparison of recommendation accuracy under this situation.

For the offline experiment, we down-sample the click / no-click to approximately 1:11 for better model fitting purpose.

We design the algorithm to recommend the top-5 news, and show the results in terms of CTR and nDCG (we omit top-5 precision because it will be the same with CTR).

5.5.1 Accuracy. The accuracy result is shown in Table 4. As expected, our algorithms outperform all the baseline algorithms. Our base model *DN* already achieves very good results compared with the baselines. This is because the dueling network structure can better model the interaction between user and news. Adding future reward consideration (*DDQN*), we achieve another significant improvement. Then, incorporating user activeness and exploration do not necessarily improve the performance under the offline setting, which might because under offline setting, the algorithm can not make the best interaction with user due to the limited static set of candidate news. (It is possible that our agent G want to recommend user u a news i for user activeness or exploration consideration, but actually the information about whether user u will click on news i or not does not exist in the offline log.) In addition, naive random exploration like ϵ-greedy will harm the recommendation accuracy.

Table 4: Offline recommendation accuracy

Method	CTR	nDCG
LR	0.1262	0.3659
FM	0.1489	0.4338
W&D	0.1554	0.4534
LinUCB	0.1447	0.4173
HLinUCB	0.1194	0.3491
DN	0.1587	0.4671
DDQN	**0.1662**	**0.4877**
DDQN + U	**0.1662**	**0.4878**
DDQN + U + EG	0.1609	0.4723
DDQN + U + DBGD	**0.1663**	**0.4854**

5.5.2 Model converge process. We further show the cumulative CTR of different methods in Figure 9 to illustrate the convergence process. The offline data are ordered by time and simulate the process that users send news request as time goes by. All the compared methods will update their models every 100 request sessions. As expected, our algorithm (*DDQN + U + DBGD*) converges to a better CTR faster than other methods.

5.6 Online evaluation

In the online evaluation stage, we deployed our models and compared algorithms on a commercial news recommendation application. Users are divided evenly to different algorithms. In online setting, we can not only measure the accuracy of recommendation, but also observe the recommendation diversity for different algorithms. All the algorithms are designed to recommend the top-20 news to a user when a news request is received.

5.6.1 Accuracy. We compare different algorithms in terms of CTR, Precision@5, and nDCG. As shown in Table 5, our full model *DDQN + U + DBGD* outperforms all the other models significantly in terms of CTR, Precision@5 and nDCG. Here are the observations

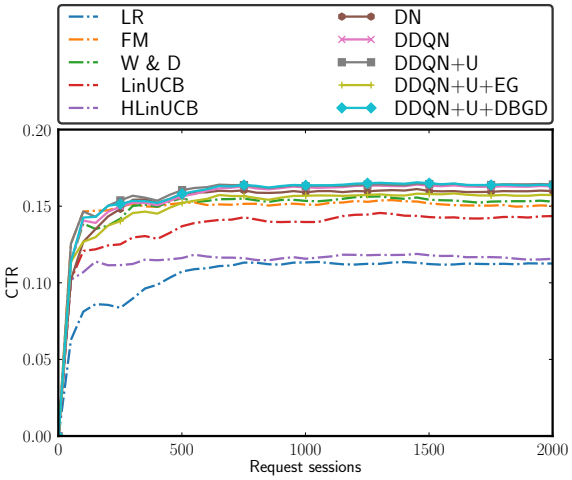


Figure 9: Offline cumulative CTR of different methods

for adding each component. Adding future reward (*DDQN*) does improve the recommendation accuracy over basic *DN*. However, further adding user activeness consideration *U* seems not very helpful in terms of recommendation accuracy. (But this component is helpful for improving user activeness and recommendation diversity. This will be demonstrated later.) In addition, using *DBGD* as exploration methods will help avoid the performance loss induced by classic ϵ-greedy methods.

Table 5: Online recommendation accuracy

Method	CTR	Precision@5	nDCG
LR	0.0059	0.0082	0.0326
FM	0.0072	0.0078	0.0353
W&D	0.0052	0.0067	0.0258
LinUCB	0.0075	0.0091	0.0383
HLinUCB	0.0085	0.0128	0.0449
DN	0.0100	0.0135	0.0474
DDQN	0.0111	0.0139	0.0477
DDQN + U	0.0089	0.0110	0.0425
DDQN + U + EG	0.0083	0.0100	0.03391
DDQN + U + DBGD	**0.0113**	**0.0149**	**0.0492**

5.6.2 Recommendation diversity. Finally, in order to evaluate the effectiveness of exploration, we calculate the recommendation diversity of different algorithms using *ILS*. [2, 53]. It is calculated by Equation 13

$$ILS(\mathsf{L}) = \frac{\sum_{b_i \in \mathsf{L}} \sum_{b_j \in \mathsf{L}, b_j \neq b_i} S(b_i, b_j)}{\sum_{b_i \in \mathsf{L}} \sum_{b_j \in \mathsf{L}, b_j \neq b_i} 1} \tag{13}$$

where $S(b_i, b_j)$ represents the cosine similarity between item b_i and item b_j. We show the diversity for the news clicked by users as in Table 6. In general, users in our algorithm *DDQN + U + DBGD* achieves the best click diversity. Interestingly, adding *EG* seems not improving the recommendation diversity. This is probably because, when random exploration (i.e., *EG*) is conducted, the recommender might recommend some totally unrelated items to users. Although these items have high diversity, users might be not interested in reading them and turn back to read more about the items that fit their interest better. This way, this exploration will not help improve the recommendation diversity. To our surprise, some baseline methods, like *HLinUCB*, also achieve comparable recommendation diversity, which indicates that *UCB* can also achieve reasonable exploration result (but this kind of unguided exploration will harm the recommendation accuracy).

Table 6: Diversity of user clicked news in the online experiment. Smaller *ILS* indicates better diversity. Similarity between news is measured by the cosine similarity between the bag-of-words vectors of news.

Method	ILS
LR	0.1833
FM	0.2014
W&D	0.1647
LinUCB	0.2636
HLinUCB	0.1323
DN	0.1546
DDQN	0.1935
DDQN + U	0.1713
DDQN + U + EG	0.1907
DDQN + U + DBGD	**0.1216**

6 CONCLUSION

In this paper, we propose a DQN-based reinforcement learning framework to do online personalized news recommendation. Different from previous methods, our method can effectively model the dynamic news features and user preferences, and plan for future explicitly, in order to achieve higher reward (e.g., CTR) in the long run. We further consider user return pattern as a supplement to click / no click label in order to capture more user feedback information. In addition, we apply an effective exploration strategy into our framework to improve the recommendation diversity and look for potential more rewarding recommendations. Experiments have shown that our method can improve the recommendation accuracy and recommendation diversity significantly. Our method can be generalized to many other recommendation problems.

For the future work, it will be more meaningful to design models for different users correspondingly (e.g., heavy users and one-time users), especially the user-activeness measure. It can bring more insights if different patterns are observed for different groups of users.

ACKNOWLEDGMENTS

The work was supported in part by NSF awards #1639150, #1544455, #1652525, and #1618448. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] 2007. Lecture Notes on Generalized Linear Models. http://data.princeton.edu/wws509/notes/. (2007).

[2] Gediminas Adomavicius and YoungOk Kwon. 2012. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering* 24, 5 (2012), 896–911.

[3] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.

[4] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* 46 (2013), 109–132.

[5] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Gançarski. 2012. A contextual-bandit algorithm for mobile context-aware recommender system. In *Neural Information Processing*. Springer, 324–331.

[6] Nicolo Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. 2013. A gang of bandits. In *Advances in Neural Information Processing Systems*. 737–745.

[7] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*. 2249–2257.

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

[9] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras. (2015).

[10] D Manning Christopher, Raghavan Prabhakar, and SCHÜTZE Hinrich. 2008. Introduction to information retrieval. *An Introduction To Information Retrieval* 151 (2008), 177.

[11] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 271–280.

[12] Gianmarco De Francisci Morales, Aristides Gionis, and Claudio Lucchese. 2012. From chatter to headlines: harnessing the real-time web for personalized news recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 153–162.

[13] Nan Du, Yichen Wang, Niao He, Jimeng Sun, and Le Song. 2015. Time-sensitive recommendation from recurrent user activities. In *Advances in Neural Information Processing Systems*. 3492–3500.

[14] Claudio Gentile, Shuai Li, and Giovanni Zappella. 2014. Online Clustering of Bandits.. In *ICML*. 757–765.

[15] Google. 2017. Google News. https://news.google.com/. (2017).

[16] Artem Grotov and Maarten de Rijke. 2016. Online learning to rank for information retrieval: SIGIR 2016 tutorial. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 1215–1218.

[17] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. Reusing historical interaction data for faster online learning to rank for IR. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 183–192.

[18] Joseph G Ibrahim, Ming-Hui Chen, and Debajyoti Sinha. 2005. *Bayesian survival analysis*. Wiley Online Library.

[19] Wouter IJntema, Frank Goossen, Flavius Frasincar, and Frederik Hogenboom. 2010. Ontology-based news recommendation. In *Proceedings of the 2010 EDBT/ICDT Workshops*. ACM, 16.

[20] How Jing and Alexander J Smola. 2017. Neural survival recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 515–524.

[21] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. 2015. Efficient Thompson Sampling for Online Matrix-Factorization Recommendation. In *Advances in Neural Information Processing Systems*. 1297–1305.

[22] Michal Kompan and Mária Bieliková. 2010. Content-Based News Recommendation.. In *EC-Web*, Vol. 61. Springer, 61–72.

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

[24] Lei Li, Dingding Wang, Tao Li, Daniel Knox, and Balaji Padmanabhan. 2011. SCENE: a scalable two-stage personalized news recommendation system. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 125–134.

[25] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 31–40.

[26] Zhongqi Lu and Qiang Yang. 2016. Partially Observable Markov Decision Process for Recommender Systems. *arXiv preprint arXiv:1608.07793* (2016).

[27] Tariq Mahmood and Francesco Ricci. 2007. Learning and adaptivity in interactive recommender systems. In *Proceedings of the ninth international conference on Electronic commerce*. ACM, 75–84.

[28] Benjamin Marlin and Richard S Zemel. 2004. The multiple multiplicative factor model for collaborative filtering. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 73.

[29] Alexander Novikov Mikhail Trofimov. 2016. tffm: TensorFlow implementation of an arbitrary order Factorization Machine. https://github.com/geffy/tffm. (2016).

[30] Rupert G Miller Jr. 2011. *Survival analysis*. Vol. 66. John Wiley & Sons.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[32] Atsuyoshi Nakamura. 2015. A ucb-like strategy of collaborative filtering. In *Asian Conference on Machine Learning*. 315–329.

[33] Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. 2011. Terms of a feather: Content-based news recommendation and discovery using twitter. *Advances in Information Retrieval* (2011), 448–459.

[34] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.

[35] Pornthep Rojanavasu, Phaitoon Srinil, and Ouen Pinngern. 2005. New recommendation system using reinforcement learning. *Special Issue of the Intl. J. Computer, the Internet and Management* 13, SP 3 (2005).

[36] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.

[37] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[38] Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. 2007. Usage-based web recommendations: a reinforcement learning approach. In *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 113–120.

[39] Liang Tang, Yexi Jiang, Lei Li, and Tao Li. 2014. Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM, 73–80.

[40] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. 2015. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 323–332.

[41] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning.. In *AAAI*. 2094–2100.

[42] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2016. Learning Hidden Features for Contextual Bandits. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1633–1642.

[43] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2017. Factorization Bandits for Interactive Recommendation.. In *AAAI*. 2695–2702.

[44] Xinxi Wang, Yi Wang, David Hsu, and Ye Wang. 2014. Exploration in interactive personalized music recommendation: a reinforcement learning approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11, 1 (2014), 7.

[45] Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2051–2059.

[46] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of ndcg type ranking measures. In *Conference on Learning Theory*. 25–54.

[47] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581* (2015).

[48] Qingyun Wu, Hongning Wang, Liangjie Hong, and Yue Shi. 2017. Returning is Believing: Optimizing Long-term User Engagement in Recommender Systems. (2017).

[49] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 1201–1208.

[50] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. 2016. Online Context-Aware Recommendation with Time Varying Multi-Armed Bandit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2025–2034.

[51] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. 2013. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 1411–1420.

[52] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 425–434.

[53] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.