Reinforcement Learning based Recommender Systems: A Survey

M. Mehdi Afsar*, Trafford Crump, and Behrouz Far

University of Calgary,
Calgary, AB
Canada

January 19, 2021

Abstract

Recommender systems (RSs) are becoming an inseparable part of our everyday lives. They help us find our favorite items to purchase, our friends on social networks, and our favorite movies to watch. Traditionally, the recommendation problem was considered as a simple classification or prediction problem; however, the sequential nature of the recommendation problem has been shown. Accordingly, it can be formulated as a Markov decision process (MDP) and reinforcement learning (RL) methods can be employed to solve it. In fact, recent advances in combining deep learning with traditional RL methods, i.e. deep reinforcement learning (DRL), has made it possible to apply RL to the recommendation problem with massive state and action spaces. In this paper, a survey on reinforcement learning based recommender systems (RLRSs) is presented. We first recognize the fact that algorithms developed for RLRSs can be generally classified into RL- and DRL-based methods. Then, we present these RL- and DRL-based methods in a classified manner based on the specific RL algorithm, e.g., Q-learning, SARSA, and REINFORCE, that is used to optimize the recommendation policy. Furthermore, some tables are presented that contain detailed information about the MDP formulation of these methods, as well as about their evaluation schemes. Finally, we discuss important aspects and challenges that can be addressed in the future.

1 Introduction

We are living in the *Zettabyte Era* [1]. The massive volume of information available on the web leads to the problem of *information overload*, which makes it difficult for a decision maker to make right decisions. The realization of this in our everyday lives is when we face a long list of items in an online shopping store; the more items in the list, the tougher it becomes to select among them. Recommender systems (RSs) are software tools and algorithms that have been developed with the idea of helping users find their items of interest, through predicting their preferences or ratings on items. In fact, the idea is to know the users to some extent (i.e., making a *user profile*), based on their feedback on items in the past, and to recommending those items that match their preferences. Today, RSs are an essential part of most giant companies, like Google, Facebook, Amazon, and Netflix, and employed in a wide range of applications, including e-commerce [2], news [3], e-learning [4], and healthcare [5].

*E-mail: `mehdi.afsar@ucalgary.ca`

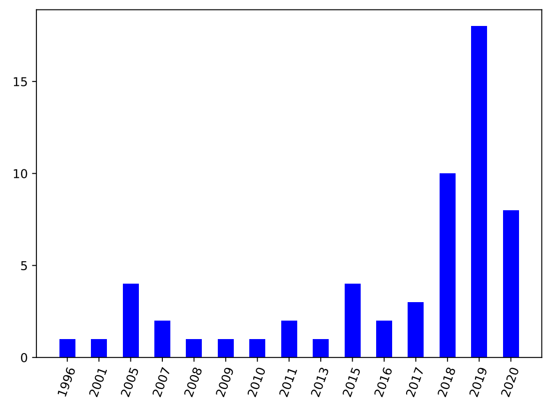

Figure 1: The number of publications in the RLRS field per year until July 2020

Numerous techniques have been proposed to tackle the recommendation problem; traditional techniques include collaborative filtering, content-based filtering, and hybrid methods. Despite some success in providing relevant recommendations, specifically after the introduction of *matrix factorization* [6], these methods have severe problems, such as *cold start*, *serendipity*, scalability, low quality recommendation, and great computational expense [7]. Recently, *deep learning (DL)* has also gained popularity in the RS field due to its abilities in finding complex and non-linear relationships between users and items and its cutting edge performance in recommendation. Nonetheless, DL models are usually non-interpretable, data hungry, and computationally expensive [8]. These problems are compounded when we realize that the amount of data (i.e., rating or user feedback) in the RS field is scarce. Above all, previous RS methods are static and can not handle the sequential nature of user interaction with the system, something that reinforcement learning (RL) can handle well.

RL is a semi-supervised machine learning field in which the agent learns what to do through interaction with the environment. The milestone in the RL field is the combination of DL with traditional RL methods, which is known as deep reinforcement learning (DRL). This made it possible to apply RL in problems with enormous state and action spaces, including self-driving cars [9, 10], robotics [11], industry automation [12], finance [13], healthcare [14, 15], and RSs. The unique ability of an RL agent in learning from a reward from the environment without any training data makes RL specifically a perfect match for the recommendation problem. Today, more and more companies are utilizing the power of RL to recommend better contents to their customers. For example, Google uses RL to recommend better video content to their users on YouTube [16]. In fact, the use of RL in the RS community is not limited to the industry, but it is becoming a trend in academia as well. Fig. 1 illustrates this fact.

This trend and the importance of topic motivated us to prepare this survey paper, which aims at providing a comprehensive and state-of -the-art overview on all algorithms in the field of reinforcement learning based recommender systems (RLRSs). Our main purpose is to depict a high-level picture from the progress in the field since the beginning and show how this trend has been significantly changed with the advent of DRL. At the same time, we provide detailed information about each method in the form of tables so as the reader can easily observe the similarities and differences between methods.

Paper Collection Methodology. The focus of this survey paper is specifically on RSs whose approach is explicitly based on an RL algorithm. Accordingly, in order to find relevant articles, we performed an exhaustive search through all related databases with the keywords recommender systems/engine/platform, recommendation, collaborative filtering, reinforcement learning, and deep reinforcement learning. We first explored important databases that usually publish these articles, including ACM digital library, IEEE Xplore, SpringerLink, ScienceDirect, to name a few. Then, we separately screened popular conferences in the field, including RecSys, KDD, WWW, SIGIR, and WSDM. Moreover, we searched Google Scholar to make sure we are not missing any item. Finally, for the sake of completeness, we have also included papers mentioned in the related work section of found papers. After paper collection, we reviewed them to make sure that all articles are relevant. Thus, we filtered out those papers that use RL for a technology other than or related to RSs, including chatbots/conversational agents/dialogue management systems[1] [17, 18, 19, 20], information retrieval systems [21], and search engines [22]. We also filtered out those papers that use RL for an RS, but RL is not the main recommendation method [23, 24, 25, 26, 27, 28]. For example, Ref. [27] tries different recommendation strategies and uses RL to select the best method among them. Moreover, we do not cover RSs based on multi-armed bandits [29, 30] in this paper as they have been extensively studied in the past and they are different from full RL methods [31].

Our contribution. The goal is to provide the reader with a vista toward the field so that they can quickly understand the topic and major trends and algorithms presented so far. This helps researchers see the big picture, compare algorithms' strengths and weaknesses, and shed some light on ways to advance them in the future. Our main contributions can be summarized as:

- Presenting a classification for RLRSs. We first generally divide the algorithms in the field into RL- and DRL-based methods. Then, each category is subdivided into specific RL algorithms used in the papers.
- Surveying all algorithms in the field. We first provide a concise but complete description about each algorithm to give the reader the main idea and contribution of the work. Then, we present two large tables to give the detailed information about every method, including information about MDP formulation, RL algorithm used, dataset, experiments and performance metrics.
- Suggesting some open research directions for the future. In order to consolidate our survey paper, we finally present some observations about ongoing research in the RLRS field and propose some open research directions to advance the field.

Although some RLRSs have been discussed and reviewed in [32, 8], to the best of our knowledge, this is the first comprehensive survey particularly developed for RLRSs.

The remaining of this paper is organized as follows. In section 2, to help the reader better understand the topic, we provide a brief introduction to the concepts discussed in the paper and explain the importance of using RL for the recommendation problem. Section 3 presents the published algorithms by first classifying them and then critically reviewing them. In section 4, the algorithms are discussed and some open research directions are suggested for the future work. Finally, the paper is concluded in section 5.

2 Preliminaries

2.1 Recommender Systems

In everyday life, it is not very uncommon to face situations in which we have to make decisions while we have no *a priori* information about options. In such a case, relying on recommendations from others, who are experienced in that aspect, seems quite necessary [33]. This was the rationale behind the first RS, Tapestry [34], and they termed it as *collaborative filtering (CF)*. Later, this term was broadened to *recommender systems* to reflect two facts [33]: 1) the method may not collaborate with users, 2) the method may suggest interesting items, not filter them. By definition, RSs are software tools and algorithms that suggest items that might be of interest

[1] We cover dialogue management systems that are used in a recommendation scenario

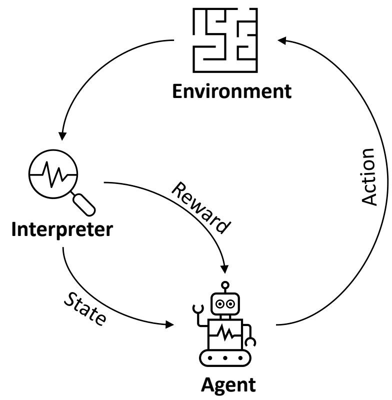

Figure 2: The agent-environment RL interface

to the users [7]. Another important approach toward the recommendation problem is *content-based filtering (CBF)*. In CBF, the idea is to recommend items similar to the user *profile*, which is a structured representation of user interests [35, 36]. Due to the problems of the two methods (i.e., CF and CBF), specifically cold-start (when user/item is new) and serendipity (having a diverse range of recommendations), *hybrid* methods was proposed to cover these problems [7]. In general, we call these methods *traditional* RSs as, because of their acute problems, namely cold start, serendipity, scalability, low quality and static recommendation, and great computational expense, it is unlikely that they are able to handle today's recommendation problems with a huge number of users and items.

2.2 Reinforcement Learning and Deep Reinforcement Learning

Reinforcement learning (RL) is a machine learning field that studies problems and their solutions in which agents, through interaction with their environment, learn what to do in order to maximize a numerical reward. According to Sutton and Barto [31], three characteristics distinguish an RL problem: (1) the problem is closed-loop, (2) the learner does not have a tutor to teach it what to do, but it should figure out what to do through trial-and-error, and (3) actions influence not only the short term results, but also the long-term ones. The most common interface to model an RL problem is the agent-environment interface, depicted in Fig. 2. The learner or decision maker is called *agent* and the *environment* is everything outside the agent. Accordingly, at time step t, the agent sees some representations/information about the environment, called *state*, and based on the current state it takes an *action*. On taking this action, it receives a numerical *reward* from the environment and finds itself in a new state. More formally, the RL problem is typically formulated as a Markov decision process (MDP) in the form of a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, where $\mathcal{S}$ is the set of all possible states, $\mathcal{A}$ is the set of available actions in all states, $\mathcal{R}$ is the reward function, $\mathcal{P}$ is the transition probability, and γ is the discount factor. The goal of the RL agent is to find a policy $\pi(a|s)$ that takes action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ in order to maximize the *expected, discounted* cumulative reward

$$\max \mathbb{E}[R(\tau)],$$
$$\text{where } R(\tau) = \sum_{t=0}^{\tau} \gamma^t r(a_t, s_t),$$
$$0 \leq \gamma \leq 1.$$

The main elements of an RL system are [31]: (1) *Policy*: policy is usually indicated by π and gives the probability of taking action a when the agent is in state s. Regarding the policy, RL algorithms can be generally divided into *on-policy* and *off-policy* methods. In the former, RL methods aim at evaluating or improving the policy they are using to make decisions. In the latter, they improve or evaluate a policy that is different from the one used to generate the data. (2) *Reward signal*: upon selecting actions, the environment provides a numerical signal reward to inform the agent how good or bad are the actions selected. (3) *Value function*: the reward signal is merely able to tell what is good immediately, but the value function defines what is good in the long run. (4) *Model*: model is an inference about the behaviour of the environment in different states.

Many algorithms have been proposed to solve an RL problem; they can be generally divided into *tabular* and *approximate* methods. In tabular methods, value functions can be represented as tables, since the size of action and state spaces is small, so that exact optimal policy can be found. Popular tabular methods include *dynamic programming (DP)*, *Monte Carlo (MC)*, and *temporal difference (TD)*. DP methods assume a perfect model of the environment and use a value function to search for good policies. Two important algorithms from this class are *policy iteration* and *value iteration*. Unlike DP, MC methods do not need a complete knowledge assumption about the environment. They only need a sample sequence of states, actions, and rewards from the environment, which could be real or simulated. Monte Carlo Tree Search (MCTS) is an important algorithm from this family. Moreover, TD methods are a combination of DP and MC methods. While they do not need a model from environment, they can *bootstrap*, which is the ability to update estimates based on other estimates [31]. From this family, Q-learning [37], which is an off-policy algorithm and simply one of the most popular RL algorithms ever, and SARSA, an on-policy method, are very popular.

On the other hand, in approximate methods, since the size of state space is enormous, the goal is to find a good approximate solution with the constraint of limited computational resources. In approximate methods, a practical approach is to *generalize* from previous experiences (already seen states) to unseen states. *Function approximation* is the type of generalization required in RL and many techniques could be used to approximate the function, including *artificial neural networks*. Among the approximate solutions, policy gradient methods have been very popular, which learn a *parameterized* policy and can select actions without the need to a value function. REINFORCE [38] and *actor-critic* are two important methods in this family.

DL, which is based on artificial neural networks, has recently gained the attention of researchers in many fields due to their superior performance [39, 40, 41, 42, 43]. This astonishing success inspired researchers at Google DeepMind to use DL as the function approximator in RL and propose deep Q-network (DQN) [44, 45], which is an approximate method for Q-learning. Later in deep deterministic policy gradient (DDPG) [46], they extended this idea for continuous spaces, which is a combination of DQN and deterministic policy gradient (DPG) [47]. Other popular DRL methods used by RS community are double DQN (DDQN) [48] and dueling Q network [49].

2.3 Reinforcement Learning for Recommendation

The nature of user interaction with an RS is sequential [50] and the problem of recommending the best items to a user is not only a prediction problem, but a sequential decision problem [51]. This suggests that the recommendation problem could be modelled as an MDP and solved by RL methods. As mentioned earlier, in a typical RL setting, an agent aims at maximizing a numerical reward through interaction with an environment. This is analogous to the recommendation problem where the RS algorithm tries to recommend the best items to the user and to maximize the user's satisfaction. Therefore, the RS algorithm can play the role of the RL agent and every thing outside this agent, including the users of the system and items, can be considered as the environment for this agent. It is almost infeasible to apply traditional tabular RL algorithms to today's RSs with huge action and state spaces [52]. Instead, with the development of DRL algorithms, it is becoming an emerging trend among the RS community to employ RL techniques.

Table 1: Algorithms

Category	Algorithm	Ref.
RL	TD (Q-learning & SARSA)	[53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]
	MC (MCTS)	[65, 66, 67]
	DP (Value/Policy iteration)	[68, 51, 69, 70]
	Fitted Q	[71, 72, 73, 74]
DRL	Q-learning (DQN)	[75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86]
	Actor-Critic	[52, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96]
	REINFORCE	[97, 98, 16, 99, 100, 101, 102]
	Compound	[103, 104, 105, 106]

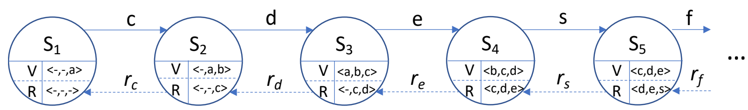

Figure 3: The concept of sliding window used in [57]. V and R indicate visited and previously recommended pages, respectively.

3 Algorithms

In this section, we present algorithms in a classified manner. After scrutinizing all the algorithms, we have recognized that the research on RLRSs has been significantly changed with the emergence of DRL. This change is also observable in Fig. 1. Thus, we have divided RLRS methods into two general groups: RL- and DRL-based algorithms. Table 1 provides a quick overview on the algorithms and publications. We first target RL-based methods.

3.1 RL-based Methods

By RL-based methods, we mean those RLRSs that use an RL algorithm for recommendation policy optimization where DL is not used for parameter estimation. RL-based methods include RL algorithms from both tabular and approximate approaches, including TD, DP, MC, and Fitted Q.

3.1.1 TD Methods

Q-learning has been a popular RL algorithm among the RS community [53, 54, 57, 58, 60, 61, 62]. WebWatcher [53] is probably the first RS algorithm that uses RL to enhance the quality of recommendations. They simply model the web page recommendation problem as an RL problem and adopt Q-learning to improve the accuracy of their basic web RS, which uses a similarity function (based on TF-IDF) to recommend pages similar to the interest of the user. A decade later, Taghipour and Kardan [57] extend this idea to recommend personalized web pages to the users. More precisely, to tackle the state dimensionality problem, they borrow the *N-gram* model from the *web usage mining* literature [107] and use a sliding window to represent states, depicted in Fig. 3. Later in [58], they enhance their work through incorporating conceptual information to their usage-based web RS. Another TD method is [54] in which a travel agent is developed that recommends personalized trips to tourists. The method composes of two main modules: *personalization learner*, responsible to learn static and dynamic information from users, and *personalization ranking*, responsible to generate the recommendations using Q-learning. While the work is among the first attempts that conducts a small scale, online experiment with real users, it is not clear how they handle a recommendation problem with large state and action spaces. Also, some technical details related to using RL, including reward function, are unclear. A conversational RS based on RL is proposed in [60] — a possible extension over their previous works [69, 70] that are discussed in Section 3.1.2. The main difference here is that Q-learning is used instead of *policy iteration* algorithm to optimize the policy. They keep the state and action spaces manageable by limiting them to a predefined number. The main contribution in RLWRec [61] is to present a state compression model to tackle the dimensionality problem of state space. In particular, the idea is to cluster songs based on similar user's performance and then replace songs with song clusters in the learning phase. Popular K-means algorithm is used to cluster the songs. RPMRS [62] utilizes effective methods, like WaveNet [108] and Word2Vec [109], to extract features from the audio and lyrics of songs. These features are used by a CBF module to shortlist an initial set of recommendations and are then refined by Q-learning.

SARSA is another TD algorithm used by some RSs [55, 56, 63]. The web RS in [55] has two main units: global and local. While the global unit is responsible to learn the global trend of the system, e.g., the most popular products, the local unit tracks each customer individually. The system uses a weighted method to combine the local and global models in order to select the next page to recommend. An obvious problem of this work is scalability as it is not clear how they want to keep the track of all users and in a global level. SARSA (λ) is an approximate solution version of the original SARSA algorithm and used in [56] to develop a personalized ontology-based web RS. The main goal of the work is to recommend the best concepts on a website to the user using RL techniques and epistemic logic programs. In fact, the contribution of the work lies in transforming epistemic information into arrays of real numbers, which are suitable for approximate RL methods to work on. Ref. [63] uses RL for online learning. The goal of the RS here is to provide the learning path for students, adapted to their specific requirements and characteristics. Similar to [57], they use the N-gram model to tackle the state dimensionality problem.

There are also some works that test both Q-learning and SARSA for their policy optimization [59, 64]. For instance, emotion-based playlist generation is formulated as an RL problem in [59]. To manage the state space, similar to [57], an *N-gram* (sliding window) model is used to model the states, i.e., every state contains information about user's last m songs' emotion classes. Ref. [64] formulates the recommendation problem as a *gridworld* game using biclustering. First, biclusters are formed from user-item matrix using Bimax [110] and Bibit [111] algorithms. Every bicluster is then mapped to one of states in the gridworld. Any state in the gridworld can be the starting state, which is the most similar state to the user based on the Jaccard distance.

3.1.2 DP Methods

DP is another tubular method that has been utilized in [68, 51, 69, 70]. Among the early works that formulates the recommendation problem as an MDP is [68]. In fact, the work discusses the possible advantages of using MDP for the recommendation problem through an example of guiding a user in an airport. Similarly, one of the early valuable attempts trying to model the recommendation problem as an MDP is [51]. Since the model parameters of an MDP-based recommender is unknown and deploying it on the real to learn them is very costly, they suggest a predictive model that is able to provide initial parameters for the MDP. This predictive model is a Markov chain in which they model the state and transition function based on the observations in the dataset. The basic version of this Markov chain uses *maximum likelihood* to estimate the transition function, but they argue that it faces the *data sparsity* problem. Accordingly, using three techniques, *skipping*, clustering, and *mixture modelling*, the basic version is improved. Then, this predictive model is used to initialize the MDP-based recommender. To tackle the dimensionality problem, last k items is used to encode state information. They test the performance of their method using an online study. In [69], Mahmood et al. develop an RL-based conversational RS and they show in a case-study (within a travel assistant called NutKing and simulated user interactions) that it can improve the quality of recommendations compared to a rigid policy. Later in [70], they extend their work and implement it on a real user study.

3.1.3 MC Methods

MC is the last tabular method and has been employed in some RLRSs [65, 66, 67]. DJ-MC [65] is an RL-based music playlist recommender. To cope with the dimensionality issue, each song is modelled as a vector of song (spectral auditory) descriptors, which include information about spectral fingerprint of the song, its rhythmic characteristics, overall loudness, and their change over time. Also, in order to accelerate the learning process, the reward function is factored as the listener's preference over individual songs and his song transition pattern. The DJ-MC architecture composes of two main components: learning listener parameters (his preference over songs and transitions) and planning a sequence of songs. The learning unit is divided into two parts: initialization and learning on the fly. In the initialization step, the listener is asked about his song and transition preferences. After initialization, the learning process begins by playing songs for the listener and requesting his feedback about them. The planning step is responsible to select the best song to recommend. To do this, MCTS is employed. In the case that the song space is too large or search time is limited, song clustering (using k-means) is performed. A work similar to DJ-MC is PHRR [66], where they use a hybrid of weighted matrix factorization (WMF) [112] and convolutional neural networks (CNNs) for song feature extraction. The goal of Div-FMCTS [67] is to propose a method that optimizes the diverse top-N recommendation problem. The method composes of two cyclic stages. It first heuristically searches the item space to find the optimal top-N recommendations using MCTS algorithm. Then, they generalize these findings by neural networks. To solve the scalability problem in searching all items, two methods are used: structure pruning and problem decomposition. Also, gated recurrent units (GRU), a DL model, is used to encode user preference information into states.

3.1.4 Fitted Q Methods

There are also some RL-based algorithms [71, 72, 73, 74] that use an approximate method (fitted Q) for policy optimization. In a clinical application [71], RL is used to recommend treatment options for lung cancer patients with the objective of maximum survival for patients. They consider the treatment for patients with advanced non-small cell lung cancer (NSCLC) as a two-line treatment, where the task of RL agent is to recommend the best treatment option in every treatment line as well as the optimal time to initiate the second line therapy. For the RL agent, support vector regression (SVR) is used to fit the Q-function. Since the original SVR cannot be applied to *censored data*, they modify SVR with ϵ-insensitive loss function [113]. Ref. [72] utilizes RL to recommend the best treatment options for schizophrenia patients. First, they use *multiple imputation* [114] to address the missing data problem, which can introduce bias and increase variance in the estimates of Q-values and is caused in light of patient dropout or item missingness. The second problem they address is the fact that the clinical data is highly variable with few trajectories and makes function approximation difficult. Accordingly, they use fitted Q-iteration (FQI) [115] with a simple linear regression model to learn the Q-function. The motivation behind the work presented in [73] is that current ad recommendation algorithms do not discriminate between a visit and a visitor and assume all the visits to a website belongs to a new visitor. Accordingly, they argue that while click through rate (CTR) is a reasonable choice for greedy performance, life-time value (LTV) is a true choice for long-term performance, defined as (total number of clicks/total number of visitors) $\times 100$. In order to tackle the *off-policy evaluation* problem in the RS field, they use a model-free approach, called HCOPE and proposed by the same authors in [116], that computes a lower bound on the expected return of a policy using a concentration inequality. Two types of algorithms are developed for ad recommendation. The first one targets the greedy optimization (CTR) and uses *random forest* to learn a mapping from features to actions. The second one targets LTV optimization using FQI. To alleviate the oscillation problem of FQI, HCOPE is used within the training loop. Lu et al. [74] propose to use partially observable MDPs (POMDPs) for the recommendation problem. Using *low-dimensional factor model*, they estimate belief states, which are a linear combination of latent item features and latent user interests. With this definition, the POMDP becomes equivalent to the MDP and can be solved by RL methods. To optimize the policy, they use a variant of FQI, which uses a neural network instead of linear regression to approximate the value of Q-function.

3.1.5 Summary

RL methods presented in this section could be divided into tabular and approximate methods. Among the tabular methods, DP methods are usually impractical due to their great computational expense and the need to perfect knowledge about the environment. While these algorithms are polynomial in the number of states, performing even one iteration of policy or value iteration methods is often infeasible [117]. In the RLRS literature, DP is only utilized by two RS methods [51, 69]. To make it practical, Ref. [51] uses a couple of features in their state space and makes some approximations. Similarly, Ref. [69] keeps the number of policy iteration run to a limited number. In contrast to DP, MC methods do not need a perfect knowledge (or model) of the environment. Instead, they only need sampled experience, i.e., some interactions with the environment. However, MC methods have some limitations; they do not bootstrap. Also, they update the value function only after a complete episode, so their convergence is slow. MCTS is a successful, enhanced MC algorithm that has been employed by [65, 66, 67]. In fact, MCTS is decision-time planning algorithm that benefits from online, incremental, sample-based value estimation and policy improvement [31]. On the other hand, TD methods have been very popular among the RS community [53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]. The main reason of this popularity is their simplicity; that is, they are online, model-free, need minimal amount of computation, and can be expressed by a single equation [31]. In general, although tabular methods can find the exact solution, i.e., the optimal value function and policy, they lead to the *curse of dimensionality* as the state and action spaces grow, which make them inefficient in learning. RLRSs that employ DP and TD methods try to address this issue by keeping the state space small. Methods based on MCTS also need to only keep the information of a sampled experience, not all the environment.

On the other hand, it is interesting to see that, apart from SARSA (λ) used by [56], the only type of approximate method used by RL-based RSs is fitted Q method, which is a flexible framework that can fit any approximation architecture to Q-function [115]. Accordingly, any batch-mode supervised regression algorithms can be used to approximate the Q-function, which can scale well to high dimensional spaces [31]. However, one problem with this method is that it could have a high computational and memory overhead with the increase in the number of *four-tuples* ((x_t, u_t, r_t, x_{t+1}), where x_t indicates the system state at time t, u_t the control action taken, r_t the immediate reward, and x_{t+1} the next state of the system) [115]. This algorithm has been used by several RLRSs [71, 72, 73, 74]. Table 2 chronologically lists the RL-based algorithms presented. Note that some algorithms have an *offline (simulation)* evaluation method. Generally speaking, we consider a simulation study as an offline evaluation method. Online study, in this paper, indicates the study conducted on real users. For an algorithm in the table, if the Dataset column is empty, it means the paper uses a pure artificial method for evaluation.

Table 2: RL-based RSs

Paper	**Year**	**RL algorithm**	**State**	**Action**	**Reward**	**Dataset**	**Evaluation Method**	**Metrics**
Joachims et al. [53]	1996	Q-learning	Pages	Selecting hyperlinks	TFIDF score	CMU CS school's web logs	Offline	Accuracy
Bohnenberger et al. [68]	2001	Value iteration	Features about location and buying a gift	A subsequent destination and a presentation mode	Reaching the gate without a gift: 0, otherwise: a positive number	N/A	N/A	N/A
Rojanavasu et al. [55]	2005	SARSA	Pages	Recommending a page	+1 for each click, +3 when the product is purchased	N/A	N/A	CTR
Preda et al. [56]	2005	SARSA (λ)	A set of visited concepts	Selecting a set of concepts	A positive reward for downloading a book	N/A	Online	Recommended download rate
Srivihok et al. [54]	2005	Q-learning	A trip list	Filtering the travel list	N/A	N/A	Online	Precision, recall, F1-score
Shani et al. [51]	2005	Policy iteration	k-tuples of items	Recommending an item	Net profit	Mitos (online book store)	Online	Averaged profit
Taghipour et al. [57]	2007	Q-learning	Last w pages requested by the user	Recommending a single page	A function of the rank and visiting time	DePaul university's web logs	Offline	Accuracy, coverage, shortcut gains
Mahmood et al. [69]	2007	Policy iteration	A set of variables from user, agent, and interaction session	Multiple actions (e.g., show the query or suggest tightening features)	Each terminal state: +1, non-terminal states: a negative number	N/A	Offline (simulation)	Optimal policy
Taghipour et al. [58]	2008	Q-learning	A sequence of concepts visited by the user	Recommending pages belonging to a specific concept	A function of the rank, visiting time, and content	DePaul university's web logs	Offline	Hit ratio, predictive ability, click reduction, recommendation quality
Mahmood et al. [70]	2009	Policy iteration	A set of variables from user, agent, and interaction session	Multiple actions (e.g., show the query or suggest tightening features)	Adding product to travel plan: +5, showing a result page: +1, otherwise: 0	N/A	Online	Pre-defined performance variables
Chi et al. [59]	2010	Q-learning & SARSA	Last m songs' emotion classes	Choosing an emotion class	Implicit (listening time & number of replays) & explicit (user rating) feedbacks	Music dataset developed by [118]	Offline (simulation) & online	Miss ratio, Miss-to-hit, Listening-time ratio, user rating

Shortreed et al. [72]	2011	FQI	Two types of variables measured from patients	Choosing treatment	Negative of the area under the PANSS[2] score curve	CATIE[3]	Offline	PANSS score
Zhao et al. [71]	2011	Fitted Q	Patient covariate values	Choosing possible treatments & timing	Overall survival time	N/A	Offline (simulation)	Overall survival
Mahmood et al. [60]	2014	Q-learning	User request	Displaying a page	Buying a book: +100, user quit without buying: -30, otherwise: 0	Amazon	Offline (simulation) & online	Pre-defined performance variables
Liebman et al. [65]	2015	MCTS	An ordered list of all songs (up to k) in the playlist	Selecting the next song	A function of user's preferences over individual songs and song transitions	Million Song Dataset [119]	Offline (simulation) & online	Cumulative reward
Theocharous et al. [73]	2015	FQI	Feature vector from the last visit of the user	Recommending an ad	user click: +1, otherwise: 0	Two datasets from banking industry	Offline	CTR & LTV [4]
Lu et al. [74]	2016	Fitted Q	Belief state (latent features from items and users)	Recommending an item	A function of user and item interaction	MovieLens 1M [120] & Yahoo Music [121]	Offline	RMSE[5]
Hu et al. [61]	2017	Q-learning	Last K songs listened	Recommending a song	A function of user listening, collecting, and downloading	A real music dataset	Offline	F1-score
Choi et al. [64]	2018	Q-learning & SARSA	A set of users and items	Selecting a direction in the gridworld	Jaccard distance between two states	MovieLens 100K & 1M	Offline	Precision & recall
Intayoad et al. [63]	2018	SARSA	Learning object	Recommending a learning object	Recommendation selected: +0.5, otherwise: -0.5	Logs from a real e-learning system	Offline	RMSE
Chang et al. [62]	2019	Q-learning	Last N songs listened	Recommending K songs	Recommended listening levels = real listening levels: +1, otherwise: 0	A real music dataset	Offline	User rating

[2] Positive and Negative Syndrome Scale
[3] Clinical Antipsychotic Trials of Intervention Effectiveness
[4] Life Time Value
[5] Root-Mean-Square Error

Zou et al. [67]	2019	MCTS	A tuple of rated history, user id, previous recommendations	Recommending a list of top-N	A function of accuracy and diversity	MovieLens 100K, 1M, 10M, and 20M	Offline	NDCG[6] & F1-score
Wang et al. [66]	2020	MCTS	A song sequence	Listening to a song	A function of user's preferences for songs and song transitions	Million Song Dataset, Taste Profile Subset [119], Historical Song Playlist [122]	Offline	Hit ratio & F1-score

[6] Normalized Discounted Cumulative Gain

3.2 DRL-based methods

In this subsection, we review DRL-based RSs in which DL is used to approximate the value function or policy. These methods use three important RL algorithms for their policy optimization, including Q-learning, actor-critic, and REINFORCE. There are also some works that test several RL algorithms for their policy optimization and compare their performance. We call them *compound* methods and present them in subsection 3.2.4.

3.2.1 Q-learning (DQN) Methods

Slate-MDP [75] is perhaps the first work that utilizes DQN for slate recommendation. In order to tackle the combinatorial action space due to slates (tuples) of actions, they introduce agents that learn the value of full slates using a *sequential greedy* method. In fact, it is assumed that the slates of items have the *sequential presentation* property in which recommended items are presented to the user one by one. This assumption is combined with another assumption in which it is assumed that one action from the *primitive actions* should be executed. They also use an attention mechanism using DDPG for each slot in the slate, to guide the search towards a small area of action space with the highest value. However, as indicated in [103], the second assumption made is not very realistic in common recommendation scenarios, as it is analogous to the condition in which we can force a user to consume a specific item.

Nemati et al. [76] use DQN to optimize heparin dosage recommendation. They first model the problem as a POMDP and use discriminative hidden Markov model to estimate the belief states. Then, DQN is used to optimize the policy. In another clinical application [77], a variant of DQN is used to optimize dosage recommendation for sepsis treatment. They use a continuous state space and a discrete action space. Due to the inherent problems in the original DQN algorithm, including overestimation of Q values, they modify DQN as follows. First, DDQN is used to estimate Q values, in order to address the overestimation problem. Also, they separate the Q function into value and advantage streams using dueling Q Network, which helps better generalize learning across actions. Finally, instead of original experience replay in DQN, *prioritized experience replay* [123] is used to accelerate the learning process.

Since the number of negative feedback, like skipping items, is much larger than that of positive feedback, Zhao et al. [78] propose a DRL framework, called DEERS and depicted in Fig. 4, to incorporate both feedbacks into the system. In fact, they modify the loss function of DQN such that it incorporates the effect of both feedbacks. Also, a regularization term is added to the loss function such that it maximizes the difference of Q-values of items from the same category with different feedbacks.

Authors in [80] try to address the high variance and biased estimation of reward in existing RSs through: 1) using *stratified sampling replay strategy* instead of original experience replay in DQN, 2) using approximate regretted reward. Instead of uniform sampling, they propose to use stratified sampling in which some stable features, like customers gender, age, and geography, are used as the strata. Moreover, a bandit-inspired method is used to estimate the reward, where the ultimate reward is the difference between two estimated rewards obtained from applying two different RL methods to the data.

In DRN [81], to tackle the dynamics in news and user's preferences, DDQN is used. Moreover, they utilize dueling Q network and separate the Q-function into value and advantage functions. In addition to common metrics, e.g., CTR, a new metric called *user activeness* is incorporated into the reward function, which reflects how often the user returns and uses the system again. Finally, instead of ϵ-greedy or upper confidence bound (UCB) family algorithms [124], *dueling bandit gradient descent* algorithm [125] is used for exploration.

The main idea in [82] is to build a user model using generative adversarial networks (GANs) and then recommend the best items using a cascading DQN algorithm. In user modelling, a *mini-max* optimization approach is used to concurrently learn user behavior and reward function. Then, using the learned user model, DQN is employed to learn the optimal recommen-

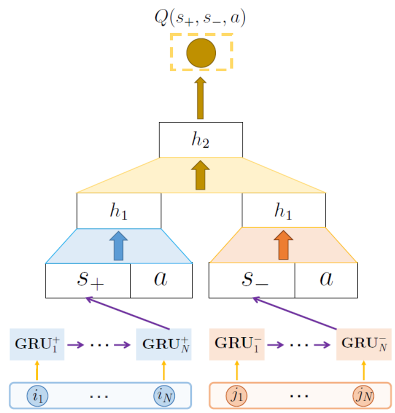

Figure 4: The architecture of DEERS [78], which incorporates both positive and negative feedbacks

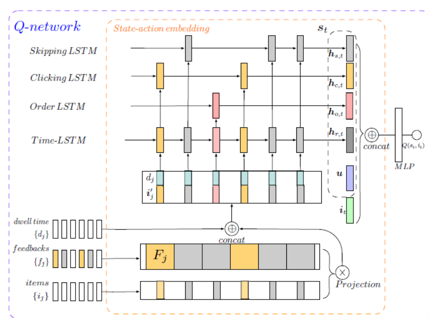

Figure 5: Q-Network (a different Q function approximator) proposed in [126]

dation policy. Unlike other methods, instead of one Q-network, to tackle the combinatorics of recommending a list of items (k items), k Q-networks are used in a cascaded manner to find k optimal actions. More precisely, the optimal actions are calculated using the following fact:

$$\max_{a_{1:k}} Q^*(s, a_{1:k}) = \max_{a_1}\big(\max_{a_{2:k}} Q^*(s, a_{1:k})\big).$$

Different from DQN-based methods, authors in [126] propose a new DL-based Q function approximator, called Q-Network and depicted in Fig. 5. The proposed Q-Network is divided into two layers: *raw behavior embedding layer* (below part in the figure) and *hierarchical behavior layer* (top part). The idea behind this division is to better capture major user behaviors, like skipping, clicking, and ordering, separately from raw behaviors, like dwell time. To better train the Q-Network offline, a user simulator (called S-Network) is also developed that receives the same information as the Q-Network and generates real user's feedback (e.g., response, dwell time). They later extend their work and propose a more general framework, called pseudo dyna-Q (PSD) [127]. PSD has two modules: *world model* and *recommendation policy*. Similar to S-Network, the world model tries to imitate the customer's feedback. On the other hand, the recommendation policy recommends an item and updates itself based on the feedback from world model.

There are also some works that simply use DQN, or its extensions, in a specific RS application without notable change or adaptation. The problem of proactive caching in mobile networks is formulated as an RL problem in [83]. To address the dimensionality, the problem is decomposed into two problems content recommendation and pushing. In recommendation, a

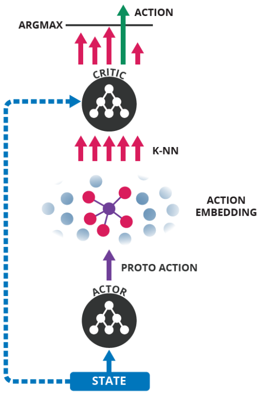

Figure 6: Wolpertinger architecture [52]

based station (BS) recommends a content from a content candidate set. Meanwhile, the BS can proactively push a content to the cache of mobile device of the user. Accordingly, two RL agents are developed for content recommendation and pushing to users which are trained sequentially and work together to reach the final goal. It is worth mentioning that for both agents DDQN with dueling Q network is utilized. In another application study [84], movie recommendation is formulated as an RL problem. The main contribution of the work is to prioritize experience replay in DQN by capturing user's interest change using cross entropy. In an internet-of-things (IoT) application [86], recEnergy uses deep Q-learning to optimize energy consumption in commercial buildings. It achieves this by making four types of recommendation: move, schedule change, personal resources, and coerce, which are some recommendations to both occupants and building management system (BMS). Finally, authors in [85] argue the effect of recommendation based on trust. In particular, RL is used to increase the trust level of users to the system, so the trust is used as the reward for the RL agent to be maximized.

3.2.2 Actor-Critique Methods

Wolpertinger [52] is an actor-critic framework that is able to handle large action spaces (up to one million). The idea is to provide a method that has sub-linear complexity w.r.t. action space and generalizable over actions. As depicted in Fig. 6, Wolpertinger consists of two parts: action generation and action refinement. In the first part, proto-actions are generated by the actor in continuous space and then are mapped to discrete space using k-nearest neighbor method. In the second part, outlier actions are filtered using a critic, which selects the best action that has the maximum Q value. Also, DDPG is used to train their method. Wolpertinger is not specifically designed for RSs, but they show in a simulation study that it can handle a recommendation task.

The main idea in SRL-RNN [89] is to combine RL with supervised learning (SL) to recommend treatment based on electronic health records (EHRs). More specifically, first the problem is formulated as a POMDP using a long short-term memory (LSTM) network. The actor is responsible to recommend the best prescription. To do so, it optimizes an objective function that is a combination of RL and SL; while it should maximize the expected return, it should minimize the difference from doctors' prescriptions. A weight is used to balance this trade-off. The critic, on the other hand, judges the actor's action.

In LIRD [87], authors first propose a stochastic environment simulator and then use an actor-critic framework for generating recommendations. The developed simulator generates a reward for a given state-action pair based on the intuition used in collaborative filtering: users with similar interests like similar items. The actor maps the state space into weight representation space and generate a score for all items, selecting the items with the highest scores as the

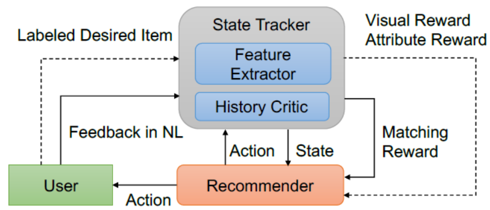

Figure 7: The agent-environment framework for [93]

actions. Also, DDPG is utilized for parameter training. One problem with this work is that they do not address the combinatorics of action space while generating a list of items instead of proposing one item. Later in [90], they propose *page-wise* recommendation. By 'page-wise' they mean recommending a set of complementary items and display them in a page. The actor is responsible to generate a page of items. First, two encoders are used to generate initial and current states. Then, a decoder, specifically a deconvolutional neural network, is used to generate the actions. On the other hand, current state (with the same strategy) and action made by actor is fed into the critic, which utilizes a DQN architecture. Again, DDPG is used for model training. They also extend their work for whole-chain recommendation in e-commerce [92]. Instead of having several scenarios in a user session, like welcome page and item pages, they use a multi-agent system with a shared memory that optimizes all these scenarios jointly (actually they only consider two pages in their studies: entrance and item pages). Agents (actors) sequentially interact with the user and cooperate with each other to maximize the cumulative reward. On the other hand, the global critic is responsible to control these actors. To capture user's preference in different scenarios, the global critic uses an attention mechanism and each attention is activated only in its specific scenario.

The idea in [93] is to propose an RL-based conversational RS that uses visual data in addition to the natural language provided by the user. The environment in the work composes of a feature extractor and a history critic, depicted in Fig. 7. The feature extractor unit is responsible to generate the state using three types of features, including visual, text, and context. The recommender receives these states and generates recommendations using a simple *nearest neighbor* method. Finally, a history critic is used to reduce the number of violations to the user's preferences described in the previous comments of the user. While the idea of using multi-modal data in this work is creative, a clear explanation about the actor-critic framework used in the work is missed.

The actor in DRR framework [88] receives the state from state representation module and generates the actions using two *ReLU* layers and one *Tanh* layer. The state representation module features three structures: product-based item, product-based item-user, and averaged (pooled) product-based item-user. The critic used is a DQN module, which uses two ReLU layers to judge the action generated by the actor. Finally, to train the model, DDPG is used. In DRRS [94] by the same authors, the actor-critic architecture is used again to generate the recommendations. The main difference here is that the recommendations are generated using a ranking function, which composes of CTR and *bid*, and the action generated by the actor is used to modify the weight of CTR.

There are also some other works that simply use DDPG to optimize the recommendation policy. In [91], store recommendation is formulated as an RL problem. First, store information is converted to a continuous space using latent Dirichlet allocation (LDA). Then, DDPG is applied to recommend a store to the user. The main contribution in CapDRL [95] is to use CapsNet [128] for feature extraction. Specifically, CapsNet is fed with both positive and negative feedbacks from the user, which converts them to the continuous space in the form of latent weight vectors. Then, DDPG is used to generate the recommendations. Also, Ref. [96] shows a possible implementation of an RS problem, using DDPG, in OpenAI gym [129].

3.2.3 REINFORCE methods

Authors in CEI [97] develop a conversational RS based on hierarchical RL [130]. In the framework, there is a module called meta-controller, which receives the dialogue state and predicts the goal for that state. There are two kinds of goals supported in the work: chitchat and recommendation. A goal-specific representation module converts the dialogue state to a score vector, which is then refined by an attention module to feature more important parts. Finally, a module called controller uses these refined scores and takes an action to satisfy the goal given. There are two critics in the framework: while an external critic evaluates the reward for the meta-controller generated by the environment, an internal critic gives reward to controller regarding the goal defined.

A conversational RLRS is proposed in [98]. The system is composed of three main parts: a belief tracker, an RS, and a policy network. The belief tracker unit is responsible to extract *facet-value pairs* (some constraints) from user utterances and convert them to belief using an LSTM network. *Factorization machine* is used in the RS to generate a set of recommendations. Finally, a neural policy network is used to manage the conversational system. More specifically, it decides to either ask for more information from the user or recommend the items.

A valuable work in the field of video recommendation using RL is presented in [16]. The main contribution of the work is to adapt REINFORCE algorithm to a neural candidate generator with a very large action space. In an online RL setting, the estimator of the policy gradient can be expressed as:

$$\sum_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{|\tau|} R_t \Delta_\theta \log\pi_\theta(a_t|s_t)\Big], \tag{1}$$

where π_θ is the parametrized policy, $\tau = (s_0, a_0, s_1, ...)$, and R_t is the cumulative reward. Since in the RS setting, unlike classical RL problems, the online or real time interaction between the agent and environment is infeasible and usually only logged feedback is available, applying the policy gradient in Eq. (1) is biased and needs correction. The off-policy-corrected policy gradient estimator is then:

$$\sum_{\tau\sim\beta}\frac{\pi_\theta(\tau)}{\beta(\tau)}\Big[\sum_{t=0}^{|\tau|} R_t \Delta_\theta \log\pi_\theta(a_t|s_t)\Big], \tag{2}$$

where β is the behavior policy and $\frac{\pi_\theta(\tau)}{\beta(\tau)}$ is the importance weight. Since this correction generates a huge variance for the estimator, they use first-order approximation, leading to the following biased estimator with a lower variance for the estimator:

$$\sum_{\tau\sim\beta}\Big[\sum_{t=0}^{|\tau|} \frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)} R_t \Delta_\theta \log\pi_\theta(a_t|s_t)\Big]. \tag{3}$$

Fig. 8 illustrates the neural architecture of the parametrized policy π_θ in Eq. (3). The last contribution of the work is top-K off-policy correction. Set (top-K) recommendation leads to exponentially growing action space. Under two assumptions, the off-policy corrected estimator presented in Eq. (3) is modified to the following estimator for top-K recommendation:

$$\sum_{\tau\sim\beta}\Big[\sum_{t=0}^{|\tau|} \frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)} \frac{\partial\alpha(a_t|s_t)}{\partial\pi(a_t|s_t)} R_t \Delta_\theta \log\pi_\theta(a_t|s_t)\Big], \tag{4}$$

where α is the probability that an item a appears in the final non-repetitive set A (top-K items). They later in [101] extend their work for a two-stage RS, where there is a candidate

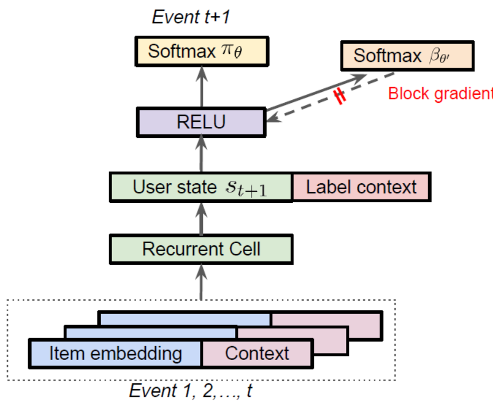

Figure 8: The neural architecture of policy π_θ [16]

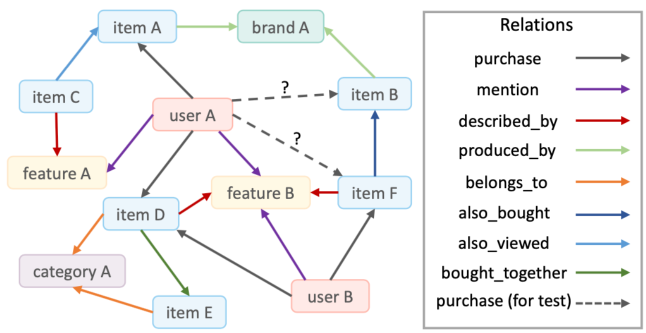

Figure 9: Graph-based reasoning example [99]

generator in the first stage and a more powerful ranking tool in the second stage. Specifically, it is assumed that the ranking model has been given so that they focus on the candidate generator. They use a sampling (*sampling with replacement*) method to make it feasible to compute the gradient for the two-stage model. Then, to reduce the variance in the importance weights of off-policy training, various techniques are used, including *weight capping*, normalization, and regularization.

Xian et al. [99] propose a graph-based explainable recommendation framework based on RL. The idea is to not only recommend a set of items, but also the paths in the knowledge graph to show the reason why the method has made these recommendations. An example of this graph reasoning is depicted in Fig. 9. For a given user A purchased item A, the RS might recommend item B with the reasoning path $\{$User $A \rightarrow$ Item $A \rightarrow$ Brand $A \rightarrow$ Item B $\}$. Obviously, graph based techniques face the scalability problem as the number of nodes and links can significantly grow, proportional to the number of users and items. To address this problem, a *user-conditional action pruning strategy* is used, which uses a scoring function to only keep important edges conditioned on the starting user. A neural policy network is used to predict the probability of items of interest to the user, which receives the state vector and pruned action space as the input and outputs the probability of each action. Then, a policy-guided path reasoning unit is used to provide the recommended items with their paths.

The main idea in TPGR [100] is to represent the item space in the form of a balanced tree and learn a strategy, using policy networks, to select the best child nodes for every non-l eaf nodes. To build the balanced tree, they use *hierarchical clustering*. Also, for the purpose of hierarchical clustering, PCA-based and K-means clustering methods are utilized. On the other hand, the recommendation task is to traverse the tree from the root to a leaf node. Every non-leaf node in this structure is associated with a policy network. Using this method, they claim that the time complexity is reduced from $O(|A|)$ to $O(d \times |A|^{1/d})$, where d indicates the depth of the tree.

Finally in [102], the environment is modelled as a heterogeneous information network (graph), which composes of users, items, and different information sources, like content, tags, reviews, friends, etc. The idea is to find a path between a user and an unobserved item in the graph. The paper uses a *multi-iteration* training process as follows. There is a *meta-path base* (like a knowledge-base) that keeps the meta-path (a sequence of node types and relation types in the graph) generated at each iteration. Initially the meta-path base is empty and filled by the meta-paths given by experts. Then, meta-paths tried in every iteration by the RL agent is added to the meta-path base. The updated meta-path base is used to train the RL agent at the next iteration. This process is repeated until no new knowledge can be achieved or it reaches the maximum number of iterations. For top-K recommendation, a nearest neighbor method is used.

3.2.4 Compound Methods

In an uncommon but interesting application, Liu et al. [79] use RL to recommend learning activities in a smart class. In particular, a cyber-physical-social system is built that monitors the learning status of students, through collecting their multi-modal data like test scores, heartbeat, and facial expression, and then recommends a learning activity suitable for them.

The main contribution in [103] is to propose SlateQ, a slate-based RS. In order to decompose the Q-value of a slate into a combination of item-wise Q-values, they assume: 1) only one item from slate is consumed by the user, 2) the reward depends only on the item consumed. Using this decomposition, they show that TD methods, like SARSA and Q-learning, can be used to maximize long-term user engagement. They also propose a flexible environment simulator, called RecSim, which simulates the dynamics of both users and RSs. In an interesting RS application that is based on SlateQ, Fotopoulou et al. [105] design an RL-like framework for an activity recommender for *social-emotional learning* of students. More precisely, the RS agent recommends a slate of activities from an activity database. The tutor then selects one activity from the list and the group accomplishes it. On response, the group provides a feedback for the activity performed and the agent updates its policy accordingly.

In [104], a task-oriented dialogue management system is proposed and applied to different recommendation tasks. Two methods are proposed for dialogue management: segmentation-based and state-based. In the former, the user population is segmented based on the context, e.g., demographics and purchase history, and every segment has a separate policy. The latter method is based on concatenation of agent belief about dialogue history, user intentions, and context. Then, a unique policy for all users is fed with this belief vector. The work is tested on a benchmark [131] in the field, which consists of several recommendation tasks, like recommending restaurants in Cambridge or San Francisco.

Finally, authors in EDRR [106] discuss that there are three components that are common in all RL approaches: embedding, state representation, and policy. They argue that directly training the embedding module with the two other modules is not possible as RL methods have gradients with high variance. They aim at alleviating this problem by adding a supervised learning (SL) unit to EDRR. To incorporate this SL signal into their method, three strategies are proposed and the difference between them is if SL signal is used to update the state representation component.

3.2.5 Summary

The foundation of DRL was a turning point in the field of RLRSs. Fig. 1 clearly illustrates this trend. The unique ability of DRL in handling high dimensional spaces makes it particularly suitable for RSs with large state and action spaces. Among the DRL algorithms employed by RLRSs, DQN has been the most popular [75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 103, 104, 105, 86, 106]. According to [31], DQN modified the original Q-learning algorithm in three ways: 1) It uses experience replay, first proposed in [132] and a method that keeps agents' experiences over various time steps in a replay memory and uses them to update weights in the training phase. 2) In order to reduce the complexity in updating weights, current updated weights are kept fixed and fed into a second (duplicate) network whose outputs are used as the Q-learning targets. 3) To limit the scale of error derivatives, the reward function is clipped such that it is 1 for positive rewards, -1 for negative rewards, and leaving 0 rewards unchanged. All these modifications turned out to improve the stability of DQN. However, as mentioned earlier, DQN has some problems; first, following Q-learning algorithm, DQN overestimates action values under certain circumstances, which makes learning inefficient and can lead to sub-optimal policies [133]. DDQN was proposed to alleviate this problem and is employed by several RLRSs [77, 81, 80, 83]. Second, DQN selects experiences randomly uniformly to replay regardless of their significance, which makes the learning process slow and inefficient. Accordingly, prioritized experience replay was proposed to solve the problem [123]. While the majority of DQN-based RLRSs use the original experience replay method of DQN, only four RLRS algorithms use an improved version. In particular, Refs. [77, 78] use prioritized experience replay [123], Ref. [80] uses stratified sampling instead of uniform sampling, and cross entropy of user interest is used in [84] to prioritize experiences. Third, DQN cannot handle continuous domains as it needs an iterative optimization process at every step, which is computationally very expensive and even infeasible. To solve this problem, DDPG has been proposed that combines DQN with DPG.

In contrast with action-value methods, like DQN, policy gradient methods learn a parameterized policy without the need to a value function. There are three advantages when employing policy-based methods over action-value methods [31]: 1) Policy approximate methods can approach determinism, 2) Policy approximation could be easier than value function approximation, and 3) Policy approximation methods can find stochastic optimal policies, while value based methods can not. The two popular policy gradient methods used by RSs are REINFORCE and actor-critic methods. REINFORCE is an MC, stochastic gradient method that directly updates the policy weights. A major problem of REINFORCE algorithm is its high variance and slow learning. These problems come from the MC nature of REINFORCE, as it takes samples randomly. In REINFORCE-based RLRSs reviewed, different techniques have been utilized to tackle the high variance problem, including a neural network-based baseline [97], a rule-based policy to initialize parameters [98], first-order approximation [16], *REINFORCE with baseline* algorithm [99], weight capping, normalization, and regularization [101]. However, it is not clear how other REINFORCE-based RLRSs, i.e., Refs. [100] and [102], address this problem. On the other hand, instead of having a baseline, actor-critic algorithm uses a critic to alleviate the problems of REINFORCE. More precisely, the critic is used to criticize the policy generated by the actor; that is, it computes the value of the state-action pair given by the actor and provides feedback on how good is the action selected. This adds bootstrapping to the policy gradient method. While this introduces an acceptable bias, it reduces variance and accelerates learning [31]. As mentioned earlier, DDPG is a well-known DRL algorithm, which uses actor-critic algorithm to handle continuous spaces. It is noteworthy to mention that actor-critic is the second popular RL algorithm among RLRSs [52, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96]. Table 3 presents the DRL-based RSs reviewed.

Table 3: DRL-based RSs

Paper	Year	RL algorithm	State	Action	Reward	Dataset	Evaluation Method	Metrics
Sunehag et al. [75]	2015	DQN	N/A	Recommending a slate of items	N/A	N/A	Offline (simulation)	Averaged total reward
Dulac et al. [52]	2016	Actor-critic	Current Item	Recommending an item	N/A	N/A	Offline (simulation)	Return
Nemati et al. [76]	2016	DQN	Belief states estimated by DHMM	Recommending heparin dosage	A function of aPTT[7] range (maximum +1 when aPTT is within the therapeutic window, minimum -1 if becomes farther)	MIMIC[8] II [134]	Offline	Accumulated reward
Raghu et al. [77]	2017	DQN	Patient physiological data from ICU	Dosage recommendation	A function of SOFA[9] score, lactate levels, and survival in terminal state (+15 for survival, -15 otherwise)	MIMIC III [135]	Offline	Mortality rate & SOFA scores
Zhao et al. [87]	2017	Actor-critic	Browsing history of a user	Recommending a list of items	Skip: 0, click: +1, purchase: +5	A real e-commerce site	Offline (simulation)	MAP[10] & NDCG
Greco et al. [97]	2017	REINFORCE	Dialogue state	Selecting a dialogue turn	+50 for meta-controller if user satisfied, -5 for meta-controller and 0 for controller otherwise	MovieLens & Movie Tweetings [136]	Offline	Reward
Zhao et al. [78]	2018	DQN	Browsing history of a user (clicked, purchased, or skipped) ordered chronologically	Recommending an item	A numerical reward based on user's skip, click, order	JD.com	Offline (simulation)	MAP & NDCG

[7] activated Partial Thromboplastin Time
[8] Multiparameter Intelligent Monitoring in Intensive Care
[9] Sequential Organ Failure Assessment
[10] Mean Average Precision

Chen et al. [80]	2018	DQN	Customer's features, preference, recent behaviour (age, gender, purchasing power, etc.)	Recommending a tip	A function of user click and tip page number	Taobao	Offline & online	CTR & UV CTR
Zheng et al. [81]	2018	DQN	User and context features	Recommending top K news	Combination of user's click and activeness	A commercial news recommendation application	Offline & online	CTR, precision, NDCG
Liu et al. [79]	2018	Compound (DQN & Q-learning)	Student's physiological data	Selection of learning activity	Learning performance	N/A	Offline (simulation)	Utility & test scores
Wang et al. [89]	2018	Actor-critic	Belief states estimated by LSTM	Recommending a medication	Patient survival: +15, patient death: -15, otherwise: 0	MIMIC III	Offline	Mortality rate & mean Jaccard coefficient
Zhao et al. [90]	2018	Actor-critic	User's browsing history and feedback	Recommending a page of M items	Skip: 0, click: +1, purchase: +5	A real e-commerce company	Offline (simulation)	Precision, recall, F1-score, NDCG, MAP, overall reward in one session
Munemasa et al. [91]	2018	Actor-critic	Browsing history and area information	Recommending a store	[-0.5, 0.5]	log data from Ekiten website	Offline	MRR[11] & recall
Liu et al. [88]	2018	Actor-critic	User's positive interaction history and demographic information	Recommending an item	[-1, 1]	MovieLens, Yahoo Music [137], Jester [138]	Offline (simulation)	Precision, NDCG, cumulative reward
Sun et al. [98]	2018	REINFORCE	Belief state estimated by LSTM	Requesting the value of a facet or make a recommendation	Successful recom.: linear, NDCG, cascade, user quit: -10, each dial. turn: -1	Yelp [139]	Offline (simulation) & Online	Average reward, success rate, number of turns, wrong quit rate, low rank rate
Chen et al. [82]	2019	DQN	Ordered sequence of user clicks	Recommending k items	User satisfaction/utility	MovieLens, Last.fm [140], Yelp, RecSys15YooChoose [141], Ant Financial News dataset	Offline (user simulation)	Cumulative reward & CTR

[11] Mean Reciprocal Rank

Liu et al. [83]	2019	DQN	Last consumed content, cache status, costs to transmit a content	Recommending and pushing a content	Net profit	N/A	Offline (simulation)	Return & achievable rate
Yuyan et al. [84]	2019	DQN	User and movie information (e.g., gender, age, etc.)	Predicting user's rating	Difference between true and predicted rating	Movielens-100k & Movielens-1M	Offline	RMSE
Qi et al. [85]	2019	DQN	N/A	Recommending an item	User's trust	Douban	Offline	Succes rate & Trust value
Zou et al. [126]	2019	DQN	The browsing sequence	Recommending an item	A weighted vector of instant, delayed, depth, and return time metrics	Logs collected from a commercial website	Offline	Clicks, depth, return time
Yu et al. [93]	2019	Actor-critic	Features extracted from item images, visual attributes, and user comments	Recommending a list of items	Visual, attribute, history matching reward	UT-Zappos50K (shoe dataset) [142]	Offline (user simulation)	Task success rate, # of violated attributes, # of user interaction before success, ranking percentile
Han et al. [94]	2019	Actor-critic	User's click history sorted chronologically	Recommending an item	User's click	MovieLens	Offline	Cumulative reward
Zhao et al. [92]	2019	Actor-critic	User click history or purchased items	Recommending an item	Click: 1, skip: 0, leave: -2	A real e-commerce company	Offline & online	MAP, NDCG, overall reward in one session
Zhao et al. [95]	2019	Actor-critic	Previous N movies rated by the user	Recommending top K movies	+1 or 0	Movielens-100k & Movielens-1M	Offline	NDCG & precision
Dorozhko et al. [96]	2019	Actor-critic	Previous N items liked by the user	Recommending an item	[-1, 1]	MovieLens	Offline	Precision & NDCG
Chen et al. [16]	2019	REINFORCE	A sequence of user interaction with the system	Recommending top K videos	Empirical values based on the experiment	YouTube	Offline (simulation) & online	View time
Xian et al. [99]	2019	REINFORCE with baseline	A tuple from user, agent, and history entities	Selecting an outgoing edges from the current node	Terminal state reward $\in [0, 1]$	Amazon e-commerce dataset	Offline	NDCG, recall, hit ratio, precision
Chen et al. [100]	2019	REINFORCE	Historical interactions between a user and the RS	Recommend an item	A combination of normalized [-1, 1] and a sequential reward	MovieLens & Netflix	Offline (simulation)	Precision, recall, F1-score, average reward

Hengst et al. [104]	2019	Compound (DQN, GP-SARSA, A2C, eNAC)	Bayesian belief state	Recommending an item or asking for more information	Recommending an item: +1, otherwise: 0	N/A	Offline (simulation)	Cumulative reward
Ie et al. [103]	2019	Compound (DQN, Q-learning, SARSA)	User features (user's behavior plus static features, like age)	Recommending a slate of items	User engagement	YouTube	Offline (simulation) & online	Return, quality, and user engagement
Fotopoulou et al. [105]	2020	Compound (SlateQ)	Social and emotional profile of educational group	Recommending a slate of activities	A function of group interest, activity quality, and group receptiveness level	N/A	Offline (simulation)	Reward & Huber loss
Liu et al. [106]	2020	Compound (DQN & DDPG)	User's positive interaction history and demographic information	Recommending an item	A function of user ratings and promotion of NDCG	MovieLens & Jester	Offline (simulation)	Precision, NDCG, MAP
Zou et al. [127]	2020	DQN	Past interactions	Recommending an item	Customer feedback (click or not) and customer leaving (the session)	Taobao, Retailrocket [143]	Offline (simulation)	Clicks, diversity, horizon
Wei et al. [86]	2020	DQN	A set of features impacting energy saving (e.g., occupant location)	Move, shift schedule, reduce, coerce	Saved energy	N/A	Online	Saved energy
Ma et al. [101]	2020	REINFORCE	Contextual status of the user or query text given by the user	Recommending an item	click: +1, otherwise: 0	MovieLens & Wiki10-31K [144]	Offline (simulation)	Precision, recall, NDCG
Liang et al. [102]	2020	REINFORCE	Nodes in the information network	Finding the best paths	The combination of global reward (+1 or -1) and path efficiency reward (1/path length)	MovieLens & Last.fm	Offline	Precision, recall, hit ratio

4 Discussion and Open Research Directions

Based on the Tables 2 and 3, there are several observations that we can make about RLRSs presented. First, in terms of RL algorithms used by RLRSs, Q-learning has been the most popular RL algorithm in the realm of RSs. The main reason of this popularity is its great simplicity [31]. This observation is consistent with the fact mentioned in [31]: "TD methods are the most widely used RL algorithms." The second popular RL algorithm has been actor-critic, mainly due to its ability in handling continuous spaces so that it can be applied to large state and action spaces. In general, as mentioned earlier, DRL has significantly changed the research direction in the RS field and we can generally divide RLRSs into before and after the foundation of DRL.

The second observation is about formulating the recommendation problem as an MDP. It is observable from the tables that different sources of information are used to represent the environment states. This information can be generally classified into information about users, items, user-system interaction (e.g., browsing), and context (e.g., time of the day). Moreover, it is observable from the action column in the tables that almost all the algorithms reviewed recommend a single item in every episode or round of recommendation, instead of a list of items. Furthermore, in terms of reward, while some methods use simple empirical rewards [87, 95], in other methods, the reward is more complex and could be a function of some observations from the environment [76, 77]. Also, it is not uncommon to see that POMDP is used instead of MDP when the uncertainty in the data or observations is high, specifically in healthcare and dialogue management applications [76, 89, 98, 104].

In terms of evaluation, the vast majority of RLRSs use an offline approach for evaluation, using publicly available datasets or pure simulation. The most popular dataset among RLRSs has been MovieLens [120]. In terms of evaluation metrics, it is interesting to see that only a few schemes have used cumulative reward, which is usually the main metric used by RL algorithms, as their evaluation metric. Metrics from information retrieval, or other machine learning areas like pattern recognition and classification, including precision, recall, F1-score, and NDCG, are the dominant metrics used by RLRSs.

Despite the recent interest in using RL for RSs and the works presented, we believe that the research in this field is in its infancy and needs a lot of advancements. In the following, we point to some directions that could be of interest to the researchers in this field for future work.

First, RL algorithms have been basically developed to select one action from different actions around. However, in the RS field, it is quite common to recommend a list of items. This is usually called *slate*, top-K, or list-wise recommendation. Apart from a few [75, 87, 81, 67, 62, 95, 93, 82, 103, 16], the vast majority of the algorithms reviewed consider the problem of single item recommendation. Among the algorithms that recommend a list of items, only Refs. [75, 103, 16] deeply study this problem and adapt their RL method to handle the list of items. The problem of recommending a list of items should be studied more in the future as the RL agent faces with a huge combinatorial action space. The larger action space, the more exploration, generalization, and optimization is needed [103].

From the architectural point of view, most of the existing RLRSs exploit available DRL architectures that were developed and successfully tested for other domains with different applications, like Atari games [44, 46]. Equally important, previous RL methods are typically designed based on physics or Gaussian processes, not based on complex and dynamic nature of users. Although it makes sense to not reinvent the wheel, sometimes thinking out of the box could make a substantial improvement in the field. For example, instead of usual MDP-based RL algorithms, Ref. [145] uses *evolution strategies* [146] to optimize the recommendation policy. Moreover, as surveyed by [8], there are a lot of DL models developed and used for RSs. Perhaps wisely combining these models with traditional RL algorithms could outperform existing DRL models. Also, as mentioned earlier, some RL algorithms (e.g., Q-learning) have been more popular among RLRSs compared to other RL methods. Nonetheless, there is no clear justification behind the use of a specific RL algorithm for an RS application. Therefore, this is a valuable research direction for the future to possibly find a relationship between the RL algorithm and the RS application.

Explainable recommendation is the ability of an RS to not only provide a recommendation, but also to address why a specific recommendation has been made [147]. Explanation about recommendations made could improve user experience, boost their trust in the system, and help them make better decisions [148, 149, 150]. Explainable methods could be generally divided into two groups: model-intrinsic or model-agnostic [151]. In the former, explanation is part of the recommendation process, while in the latter, the explanation is provided after the recommendation is made. An intrinsic explanation method could be the method we reviewed earlier [99]. On the other hand, as a model-agnostic example [152], RL is used to provide explanation for different recommendation methods. In particular, the method uses *couple agents*; one is responsible to generate explanations and another one predicts if the explanation generated is good enough for the user. One interesting application of explainable recommendation is in debugging the failed RS [152]. That is, through explanations provided, we can track the source of problems in our system and to see which parts are not working properly. Among the RLRSs reviewed in this survey, only Ref. [99] supports an explainable recommendation, so it shows that there is a lack in this aspect and more attention is required in the future. In general, we believe that explainable recommendation is a must in the next generation of RSs and RL can be effectively employed to provide better explanations.

Finally, evaluation for RLRSs should be improved. An RL agent needs to directly interact with the environment to learn what to do. This is equivalent to online study for an RS; i.e., the RS algorithm generates the recommendations and receives the user feedbacks in real-time. Nonetheless, apart from a few methods reviewed [70, 59, 103, 16, 98, 92, 86, 81], most of the works use an offline study for evaluation. This is mainly because online study is expensive and deploying an RLRS to optimize its recommendation policy is very risky for most businesses. Accordingly, offline evaluation, using available datasets or simulation, is of great importance for RLRSs evaluation. In offline evaluation, simulation is specifically important as evaluation using available datasets is usually biased and static, as they reflect user's feedback in that specific RS setting. On the other hand, the majority of user models in the simulation studies are often too simple and may not be able to reflect real user behaviors. The GAN-based user simulator developed in [82] could be a good example. Although some environment simulators have recently been developed for RLRSs [153, 154, 155], we believe that there is an essential need to have a powerful simulator, preferably a strong unified evaluation framework, for RLRSs, and the RS field in general. This need is more tangible when we see that there is no evaluation metric specifically developed for RSs.

5 Conclusion

In this paper, we presented a comprehensive and state-of-the-art survey on RLRSs. We highlighted the important role of DRL in changing the research direction in the RLRS field, and accordingly, classified the algorithms into two general groups, i.e., RL- and DRL-based methods. Subsequently, each general group was divided into sub-categories regarding the specific RL algorithm used, such as Q-learning and actor-critic. We believe that the research on RLRSs is in its infancy and needs plenty of advancements. Both RL and RSs are hot and ongoing research areas and are of specific interest to giant companies and businesses. Accordingly, we can expect to witness new and exciting models to emerge each year. In the end, we hope this survey can assist researchers in understanding the key concepts and help advance the field in the future.

References

[1] Cisco Visual Networking Index. The zettabyte era–trends and analysis. *Cisco white paper*, 2013.

[2] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, 1999.

[3] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. News recommender systems–survey and roads ahead. *Information Processing & Management*, 54(6):1203–1227, 2018.

[4] Aleksandra Klašnja-Milićević, Mirjana Ivanović, and Alexandros Nanopoulos. Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 44(4):571–604, 2015.

[5] Emre Sezgin and Sevgi Özkan. A systematic literature review on health recommender systems. In *2013 E-Health and Bioengineering Conference (EHB)*, pages 1–4. IEEE, 2013.

[6] Netflix update: Try this at home. `https://sifter.org/~simon/journal/20061211.html`.

[7] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. 2011.

[8] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[9] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

[10] Changxi You, Jianbo Lu, Dimitar Filev, and Panagiotis Tsiotras. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems*, 114:1–18, 2019.

[11] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[12] Richard Meyes, Hasan Tercan, Simon Roggendorf, Thomas Thiele, Christian Büscher, Markus Obdenbusch, Christian Brecher, Sabina Jeschke, and Tobias Meisen. Motion planning for industrial robots using reinforcement learning. *Procedia CIRP*, 63:107–112, 2017.

[13] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

[14] Arthur Guez, Robert D Vincent, Massimo Avoli, and Joelle Pineau. Adaptive treatment of epilepsy via batch-mode reinforcement learning. In *AAAI*, pages 1671–1678, 2008.

[15] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nat Med*, 25(1):16–18, 2019.

[16] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464, 2019.

[17] Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, pages 956–962, 2000.

[18] Joel Tetreault and Diane Litman. Using reinforcement learning to build a better model of dialogue state. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

[19] Matthew Frampton and Oliver Lemon. Learning more effective dialogue strategies using limited dialogue move features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 185–192, 2006.

[20] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*, 2016.

[21] Byoung-Tak Zhang and Young-Woo Seo. Personalized web-document filtering using reinforcement learning. *Applied Artificial Intelligence*, 15(7):665–685, 2001.

[22] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4902–4909, 2019.

[23] Maria Salamó, Sergio Escalera, and Petia Radeva. Quality enhancement based on reinforcement learning and feature weighting for a critiquing-based recommender. In *International Conference on Case-Based Reasoning*, pages 298–312. Springer, 2009.

[24] Feng Ge, Yi He, Jin Liu, Xiaoming Lv, Wensheng Zhang, and Yiqun Li. A reinforcement learning based tag recommendation. In *Practical Applications of Intelligent Systems*, pages 251–258. Springer, 2011.

[25] Zhenni Lu, Min Gao, Xinyi Wang, Junwei Zhang, Haider Ali, and Qingyu Xiong. Srrl: Select reliable friends for social recommendation with reinforcement learning. In *International Conference on Neural Information Processing*, pages 631–642. Springer, 2019.

[26] Yan Zheng Wei, Luc Moreau, and Nicholas R Jennings. Learning users' interests in a market-based recommender system. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 833–840. Springer, 2004.

[27] Yan Zheng Wei, Luc Moreau, and Nicholas R Jennings. Learning users' interests by quality classification in market-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1678–1688, 2005.

[28] Nick Golovin and Erhard Rahm. Reinforcement learning architecture for web recommendations. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, volume 1, pages 398–402. IEEE, 2004.

[29] Romain Warlop, Alessandro Lazaric, and Jérémie Mary. Fighting boredom in recommender systems with linear reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1757–1768, 2018.

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

[31] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 2017.

[32] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. " 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):1–15, 2019.

[33] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[34] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[35] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[36] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[37] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[38] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[40] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[41] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[42] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[45] 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. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

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

[47] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

[48] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

[49] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016.

[50] A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. In *17th Conference in Uncertainty in Artificial Intelligence*, pages 580–588, 2001.

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

[52] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[53] Thorsten Joachims, Dayne Freitag, Tom Mitchell, et al. Webwatcher: A tour guide for the world wide web. In *IJCAI (1)*, pages 770–777. Citeseer, 1997.

[54] Anongnart Srivihok and Pisit Sukonmanee. E-commerce intelligent agent: personalization travel support agent using q learning. In *Proceedings of the 7th international conference on Electronic commerce*, pages 287–292, 2005.

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

[56] Mircea Preda and Dan Popescu. Personalized web recommendations: supporting epistemic information about end-users. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 692–695. IEEE, 2005.

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

[58] Nima Taghipour and Ahmad Kardan. A hybrid web recommender system based on q-learning. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1164–1168, 2008.

[59] Chung-Yi Chi, Richard Tzong-Han Tsai, Jeng-You Lai, and Jane Yung-jen Hsu. A reinforcement learning approach to emotion-based automatic playlist generation. In *2010 International Conference on Technologies and Applications of Artificial Intelligence*, pages 60–65. IEEE, 2010.

[60] Tariq Mahmood, Ghulam Mujtaba, and Adriano Venturini. Dynamic personalization in conversational recommender systems. *Information Systems and e-Business Management*, 12(2):213–238, 2014.

[61] Binbin Hu, Chuan Shi, and Jian Liu. Playlist recommendation based on reinforcement learning. In *International Conference on Intelligence Science*, pages 172–182. Springer, 2017.

[62] Jia-Wei Chang, Ching-Yi Chiou, Jia-Yi Liao, Ying-Kai Hung, Chien-Che Huang, Kuan-Cheng Lin, and Ying-Hung Pu. Music recommender using deep embedding-based features and behavior-based reinforcement learning. *Multimedia Tools and Applications*, pages 1–28, 2019.

[63] Wacharawan Intayoad, Chayapol Kamyod, and Punnarumol Temdee. Reinforcement learning for online learning recommendation system. In *2018 Global Wireless Summit (GWS)*, pages 167–170. IEEE, 2018.

[64] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. Reinforcement learning based recommender system using biclustering technique. *arXiv preprint arXiv:1801.05532*, 2018.

[65] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. *arXiv preprint arXiv:1401.1880*, 2014.

[66] Yu Wang. A hybrid recommendation for music based on reinforcement learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 91–103. Springer, 2020.

[67] Lixin Zou, Long Xia, Zhuoye Ding, Dawei Yin, Jiaxing Song, and Weidong Liu. Reinforcement learning to diversify top-n recommendation. In *International Conference on Database Systems for Advanced Applications*, pages 104–120. Springer, 2019.

[68] Thorsten Bohnenberger and Anthony Jameson. When policies are better than plans: Decision-theoretic planning of recommendation sequences. In *Proceedings of the 6th international conference on Intelligent user interfaces*, pages 21–24, 2001.

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

[70] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82, 2009.

[71] Yufan Zhao, Donglin Zeng, Mark A Socinski, and Michael R Kosorok. Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer. *Biometrics*, 67(4):1422–1433, 2011.

[72] Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84(1-2):109–136, 2011.

[73] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. Personalized ad recommendation systems for life-time value optimization with guarantees. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[74] Zhongqi Lu and Qiang Yang. Partially observable markov decision process for recommender systems. *arXiv preprint arXiv:1608.07793*, 2016.

[75] Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold, Yori Zwols, Daniel Visentin, and Ben Coppin. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. *arXiv preprint arXiv:1512.01124*, 2015.

[76] Shamim Nemati, Mohammad M Ghassemi, and Gari D Clifford. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2978–2981. IEEE, 2016.

[77] Aniruddh Raghu, Matthieu Komorowski, Imran Ahmed, Leo Celi, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*, 2017.

[78] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1040–1048, 2018.

[79] Su Liu, Ye Chen, Hui Huang, Liang Xiao, and Xiaojun Hei. Towards smart educational recommendations with reinforcement learning in classroom. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 1079–1084. IEEE, 2018.

[80] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 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*, pages 1187–1196, 2018.

[81] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, pages 167–176, 2018.

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

[83] Dong Liu and Chenyang Yang. A deep reinforcement learning approach to proactive content pushing and recommendation for mobile users. *IEEE Access*, 7:83120–83136, 2019.

[84] Zhang Yuyan, Su Xiayao, and Liu Yong. A novel movie recommendation system based on deep reinforcement learning with prioritized experience replay. In *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, pages 1496–1500. IEEE, 2019.

[85] Faxin Qi, Xiangrong Tong, Lei Yu, and Yingjie Wang. Personalized project recommendations: using reinforcement learning. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–17, 2019.

[86] Peter Wei, Stephen Xia, Runfeng Chen, Jingyi Qian, Chong Li, and Xiaofan Jiang. A deep reinforcement learning based recommender system for occupant-driven energy optimization in commercial buildings. *IEEE Internet of Things Journal*, 2020.

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

[88] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027*, 2018.

[89] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2447–2456, 2018.

[90] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 95–103, 2018.

[91] Isshu Munemasa, Yuta Tomomatsu, Kunioki Hayashi, and Tomohiro Takagi. Deep reinforcement learning for recommender systems. In *2018 international conference on information and communications technology (icoiact)*, pages 226–233. IEEE, 2018.

[92] Xiangyu Zhao, Long Xia, Dawei Yin, and Jiliang Tang. Model-based reinforcement learning for whole-chain recommendations. *arXiv preprint arXiv:1902.03987*, 2019.

[93] Tong Yu, Yilin Shen, Ruiyi Zhang, Xiangyu Zeng, and Hongxia Jin. Vision-language recommendation via attribute augmented multimodal reinforcement learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 39–47, 2019.

[94] Jianhua Han, Yong Yu, Feng Liu, Ruiming Tang, and Yuzhou Zhang. Optimizing ranking algorithm in recommender system via deep reinforcement learning. In *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pages 22–26. IEEE, 2019.

[95] Chenfei Zhao and Lan Hu. Capdrl: A deep capsule reinforcement learning for movie recommendation. In *Pacific Rim International Conference on Artificial Intelligence*, pages 734–739. Springer, 2019.

[96] Anton Dorozhko. Reinforcement learning for long-term reward optimization in recommender systems. In *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, pages 0862–0867. IEEE, 2019.

[97] Claudio Greco, Alessandro Suglia, Pierpaolo Basile, and Giovanni Semeraro. Converse-et-impera: Exploiting deep learning and hierarchical reinforcement learning for conversational recommender systems. In *Conference of the Italian Association for Artificial Intelligence*, pages 372–386. Springer, 2017.

[98] Yueming Sun and Yi Zhang. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 235–244, 2018.

[99] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294, 2019.

[100] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3312–3320, 2019.

[101] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H Chi. Off-policy learning in two-stage recommender systems. In *Proceedings of The Web Conference 2020*, pages 463–473, 2020.

[102] Huizhi Liang. Drprofiling: deep reinforcement user profiling for recommendations in heterogenous information networks. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[103] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767*, 2019.

[104] Floris Den Hengst, Mark Hoogendoorn, Frank Van Harmelen, and Joost Bosman. Reinforcement learning for personalized dialogue management. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 59–67, 2019.

[105] Eleni Fotopoulou, Anastasios Zafeiropoulos, Michalis Feidakis, Dimitrios Metafas, and Symeon Papavassiliou. An interactive recommender system based on reinforcement learning for improving emotional competences in educational groups. In *International Conference on Intelligent Tutoring Systems*, pages 248–258. Springer, 2020.

[106] Feng Liu, Huifeng Guo, Xutao Li, Ruiming Tang, Yunming Ye, and Xiuqiang He. End-to-end deep reinforcement learning based recommendation with supervised embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 384–392, 2020.

[107] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[108] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[109] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[110] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[111] Domingo S Rodriguez-Baena, Antonio J Perez-Pulido, and Jesus S Aguilar-Ruiz. A biclustering algorithm for extracting bit-patterns from binary datasets. *Bioinformatics*, 27(19):2738–2745, 2011.

[112] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[113] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[114] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[115] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[116] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[117] Andrew G Barto. Reinforcement learning and dynamic programming. In *Analysis, Design and Evaluation of Man–Machine Systems 1995*, pages 407–412. Elsevier, 1995.

[118] Chung-Yi Chi, Ying-Shian Wu, Wei-rong Chu, Daniel C Wu, Jane Yung-jen Hsu, and Richard Tzong-Han Tsai. The power of words: Enhancing music mood estimation with textual input of lyrics. In *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6. IEEE, 2009.

[119] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *12th International Society for Music Information Retrieval Conference,*, pages 591–596, 2011.

[120] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[121] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The yahoo! music dataset and kdd-cup'11. In *Proceedings of KDD Cup 2011*, pages 3–18. PMLR, 2012.

[122] Shuo Chen, Jiexun Xu, and Thorsten Joachims. Multi-space probabilistic sequence modeling. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 865–873, 2013.

[123] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[124] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

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

[126] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2810–2818, 2019.

[127] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. Pseudo dyna-q: A reinforcement learning framework for interactive recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 816–824, 2020.

[128] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

[129] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[130] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[131] Inigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Stefan Ultes, Lina Rojas-Barahona, Steve Young, and Milica Gašić. A benchmarking environment for reinforcement learning based task oriented dialogue management. *arXiv preprint arXiv:1711.11023*, 2017.

[132] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[133] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[134] Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 39(5):952, 2011.

[135] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[136] Simon Dooms, Toon De Pessemier, and Luc Martens. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys*, volume 2013, page 43, 2013.

[137] Yahoo! music. `https://webscope.sandbox.yahoo.com`.

[138] Jester. `http://eigentaste.berkeley.edu/dataset/`.

[139] Yelp. `https://www.yelp.com/academic_dataset`.

[140] Last.fm. `https://www.last.fm/api`.

[141] Recsys15. `https://2015.recsyschallenge.com/`.

[142] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014.

[143] Retailrocket. `https://www.kaggle.com/retailrocket/ecommerce-dataset/home`.

[144] Arkaitz Zubiaga. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*, 2012.

[145] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. Value-aware recommendation based on reinforcement profit maximization. In *The World Wide Web Conference*, pages 3123–3129, 2019.

[146] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[147] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*, 2018.

[148] Dan Cosley, Shyong K Lam, Istvan Albert, Joseph A Konstan, and John Riedl. Is seeing believing? how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 585–592, 2003.

[149] Li Chen and Pearl Pu. Trust building in recommender agents. In *Proceedings of the Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces at the 2nd International Conference on E-Business and Telecommunication Networks*, pages 135–145. Citeseer, 2005.

[150] Nava Tintarev and Judith Masthoff. Effective explanations of recommendations: user-centered design. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 153–156, 2007.

[151] Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

[152] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. A reinforcement learning framework for explainable recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 587–596. IEEE, 2018.

[153] Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Yuchen He, Zachary Kaden, Vivek Narayanan, Xiaohui Ye, Zhengxing Chen, and Scott Fujimoto. Horizon: Facebook's open source applied reinforcement learning platform. *arXiv preprint arXiv:1811.00260*, 2018.

[154] Xiangyu Zhao, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Toward simulating environments in reinforcement learning based recommendations. *arXiv preprint arXiv:1906.11462*, 2019.

[155] Bichen Shi, Makbule Gulcin Ozsoy, Neil Hurley, Barry Smyth, Elias Z Tragos, James Geraci, and Aonghus Lawlor. Pyrecgym: a reinforcement learning gym for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 491–495, 2019.