Recommender System using Reinforcement Learning: A Survey

Mehrdad Rezaei and Nasseh Tabrizi
Department of Computer Science, East Carolina University, East 5th Street, Greenville, NC, U.S.A.

Keywords: Recommender Systems, Reinforcement Learning, Agent, Rewards.

Abstract: Recommender systems are rapidly becoming an integral part of our daily lives. They play a crucial role in overcoming the overloading problem of information by suggesting and personalizing the recommended items. Collaborative filtering, content-based filtering, and hybrid methods are examples of traditional recommender systems which had been used for straightforward prediction problems. More complex problems can be solved with new methods which are applied to recommender systems, such as reinforcement learning algorithms. Markov decision process and reinforcement learning can take part in solving these problems. Recent developments in applying reinforcement learning methods to recommender systems make it possible to use them in order to solve problems with the massive environment and states. A review of the reinforcement learning recommender system will follow the traditional and reinforcement learning-based methods formulation, their evaluation, challenges, and recommended future work.

1 INTRODUCTION

The enormous amount of information existing on the Internet causes the information overload problem, making it hard to make the right decision. It can be realized in our everyday online shopping when we have an extended list of possible items to be purchased. If the list grows longer, it will be harder to select from the list. Recommender Systems (RSs) are algorithms and software tools designed to assist users in finding items of interest by anticipating their preferences or ratings. The development of RSs helps users find the item they are interested in by predicting the rating on the items and their previous preferences. Today, RSs are a crucial part of enormous companies like Netflix, Amazon, Facebook, and Google where a vast range of applications of RSs is employed, such as e-learning (Aleksandra et al., 2015), e-commerce (Ben et al., 1999), healthcare (Emre and Sevgi, 2013), and news (Mozhgan et al., 2018). Different techniques such as content-based filtering, collaborative filtering, and hybrid methods are also proposed to address the recommender system problem. By the introduction of matrix factorization, some success was achieved in the field of providing appropriate recommendations. Still, the mentioned methods have problems, i.e., cold start, scalability, serendipity, proper computational expense, and recommendation quality (Francesco et al. 2011). Deep learning (DL) has recently gained approval in the RSs application field because of the potential in complex relationships of users, items, and their accurate performance in the recommendation. Non-interpretable, computationally expensive, and data-hungry are properties of DL models (Shuai et al., 2019). Above all, prior RSs methods are not beneficial in the interaction between users and items, which can be better handled with Reinforcement Learning (RL) and its training agent in the environment that is a semi-supervised machine learning field (Shuai et al., 2019). The most critical point in the RL can be a combination of traditional RL methods and DL together; this combination is known as Deep Reinforcement Learning (DRL). This allowed RL to be used in problems with large state and action spaces, such as robotics (Jens et al.,2013), industry automation (Richard et al., 2017), self-driving cars (Ahmad et al. 2017; Changxi et al, 2019), finance (Zhengyao et al. 2017), healthcare (Arthur et al., 2008).

The RL is a seamless match for the recommendation problem because it has the ability to reward learning without any training data, which is a unique specification. These days, the power of RL is used to recommend better items to the customers by many companies, i.e., the video recommender system on YouTube uses RL (Minmin et al., 2019). The use of RL in the RSs is becoming more popular not only in the industry but also in academia. The importance of

Rezaei, M. and Tabrizi, N.
Recommender System using Reinforcement Learning: A Survey.
DOI: 10.5220/0011300300003277
In *Proceedings of the 3rd International Conference on Deep Learning Theory and Applications (DeLTA 2022)*, pages 148-159
ISBN: 978-989-758-584-5; ISSN: 2184-9277

this topic motivated us to write this paper in the field of reinforcement for RSs. Our paper's major goal is to show the progress in utilizing RL in RSs to depict the trend that has been changed during recent years. The sample chart shown in Fig. 1 depicts the number of papers published from 2010 through January 2022.

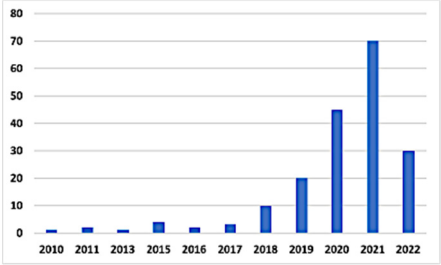

Figure 1: Number of papers published from 2010 to 2022.

2 METHODOLOGIES

We have decided to explore the problems and challenges associated with the RSs with the RL algorithm. As the second goal, methods, and algorithms to tackle these challenges are discussed, highlighting a critical point that involves the introduction of the applications for RL-based and non-RL-based RSs. This paper addresses the followings:

- We first categorize the algorithms in the field into RL- and DRL-based techniques. The categories are then divided into specific RL algorithms employed in the studies.
- To provide the reader with the essential idea and contribution of the study, we first provide a simple yet detailed description of each algorithm.
- Future research directions are suggested. Finally, in order to combine our survey study, we give some insights about active research in the Reinforcement Learning Recommender System (RLRS) field.

We initially searched among all related articles using multiple search engines with the following search queries: recommender systems, recommender engine, recommendation, content filtering, collaborative filtering, reinforcement learning algorithm, deep reinforcement learning algorithm, and reinforcement learning for recommender systems and applied them in different databases including IEEE Xplore, SpringerLink, ScienceDirect, ACM digital library, Lynda.com. We also explored popular conferences in the field of RSs, such as RecSys, SIGIR, and KDD. Following the article collection, we reviewed the publications to find all related articles for our purpose (Aleksandra et al., 2015; Smyth et al., 2000). Therefore, the papers that use RL also for other technologies other than RSs such as dialogue management system/conversational were selected (Satinder et al., 2000; Joel et al., 2006).

3 RECOMMENDER SYSTEMS

On daily basis, we encounter situations where decisions need to be made and there is not enough information on different aspects of them. In these cases, it is necessary to trust others' recommendations who are experienced in those situations (Paul et al., 1992). The early RSs were called collaborative filtering (CF) (David et al., 1992). Then, it was changed to RSs for two reasons: i) CF may not be used by the users ii) item may be recommended not filtered by the method. The recommender system may use another approach termed Content-based Filtering (CBF) that applies the user profile to suggest related items associated with the user's interest (Michael and Daniel 2007; Pasquale et al., 2011). Both CF and CBF cannot be used for some problems, where cold-start and serendipity persist. To address the above issues, the hybrid method was applied (Francesco et al. 2011). All the mentioned methods cannot handle today's RS problems due to a massive number of users and items. We will introduce briefly classic techniques used in RSs in this section and in the next sections, we'll go through RL in further depth.

3.1 Collaborative Filtering

The aim of Collaborative Filtering (CF) RSs is to help users to make a decision based on other users' suggestions with similar interests (Deshpande and Karypis, 2004). The CF approaches can be divided into item-based and user-based (Sarwar et al., 2001), wherein the user-based, the recommended items will be suggested based on items that are liked by similar users. In the item-based, items will be recommended to users based on items in which they were interested. Pearson correlation-based (Resnick et al. 1994), Cosine-based, and Adjusted Cosine-based can be used to calculate the similarity between users or items. Users who rated both items will take part in the calculation of the similarity of recommended items which helps the calculation be more accurate. An improved item-based CF was introduced by the combination of the Adjusted Cosine and Jaccard metric in order to increase the similarity calculation's accuracy (Shambour and Lu, 2011).

3.2 Content-based Filtering

Content-based (CB) RSs recommend items based on similar items which have been liked by the user in the past (Pazzani and Billsus, 2007). The fundamental concepts of CB RSs are i) The specifications of items are used to find the recommended items. In order to detect these attributes, the items description which is preferred by a certain user should be analyzed. ii) For each item, specifications are compared with the profile of the user. Eventually, items with a high degree of similarity to the profile of the user will be recommended (Pazzani and Billsus, 2007). Two techniques are used to create recommendations in CB. Methods with information retrieval, including Cosine similarity measures, are used in the first technique, which generates recommendations. The second technique uses Machine Learning methods to create recommendations.

3.3 Hybrid Recommendation Methods

To tackle the weaknesses regarding traditional RS techniques and to achieve higher efficiency, a hybrid RS technique is used, that merges two or more recommendation techniques (Bruke, 2007). There are seven initial combination approaches that are used to create hybrid methods: Mixed (Smyth and Cotter, 2000), Weighted (Mobasher et al. 2004), Switching (Billsus and Pazzani, 2000), Feature Combination, and Argumentation (Wilson et al. 2003), Meta-level (Pazzani, 1999), and Cascade (Bruke, 2002). The most used hybrid RSs attempt to tackle cold start and scalability issues (Bellogin et al., 2013).

4 REINFORCEMENT LEARNING AND DEEP REINFORCEMENT LEARNING

A machine learning method that studies different problems and solutions to maximize a reward through interaction between agents and their environment is called RL. Three characteristics that discriminate an RL problem (Richard et al., 2017) are: i) closed-loop problem ii) there is no need for a trainer to teach the learner, but it trains what to do to the learner with the trial-and-error method according to the policy iii) the short and long terms results can be influenced by the actions. The crucial part to model the RL problem is the agent's interface and environment as shown in Fig. 2.

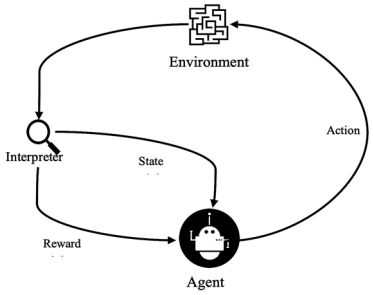

Figure 2: The interface of RL.

An agent is a decision-maker or learner; everything outside of the agent is called the environment. Information and representation that is seen outside of the agent (the environment) at time step t is called state, and the agent makes an action according to the current state. Based on the action taken, the environment is given a numerical reward and goes to a new state. The RL problems are formulated commonly as a Markov Decision Process (MDP) with the form of (S, A, R, P, γ), where S represents all possible states, A indicates actions that are available in all states, R presents reward function, P shows the probability of the transition, and finally, γ is the discount factor. The agent's aim in the RL problem is the best policy $\pi(a|s)$ to make an action which is a member of A in state $s \in S$ to maximize the cumulative reward. An RL system includes four principal parts (Richard et al., 2017): i) Policy: It is presented by π generally, which indicates the probability of doing an action. The RL algorithm may be categorized into on-policy and off-policy techniques depending on the policy. In the first case, RL approaches are used to evaluate or improve the policy that is being used to make judgments. They enhance or assess a policy that is not the same as the one used to create the data in the latter. ii) Numeral rewards: regarding the selected actions, the environment gives a numeral reward in order to send an announcement to the agent about the action that is selected. iii) Value function: the purpose of the value function is to indicate how good or bad is the action in the long run. iv) Model: it indicates the conduct of the environment. There are two types of algorithms that are utilized to address RL challenges: tabular and approximate. In the tabular method, tables are used to represent value functions, and an accurate policy is found because the size of spaces (action and state) is not big. Monte Carlo (MC), Temporal Difference (TD), and Dynamic Programming (DP) are popular tabular methods. The MC methods need only an instance of rewards, states, and actions that will be provided by the environment. Monte Carlo Tree

Search (MCTS) is the most important algorithm of MC methods. The DP methods use an excellent model of the environment and value function in order to find good policies. Policy and value iteration can be good examples of DP methods. The TD method is a blend of the MC sampling method and the DP bootstrapping method. The TD methods, like the MC methods, may learn from the agent's interactions with the world and do not require model knowledge. From this class, Q-learning (Christopher, 1989) and SARSA are the most important ones as they are off-policy and on-policy, respectively. In the approximate method, the aim is to search for sufficient solutions regarding the computational resources constraint because state space has a massive size. To address this, previous experiences are used. Policy Gradient methods are very popular because of their ability to learn policy parametrization and actions selection without the need for a value function. Actor-critic and reinforcement (Roland, 1992) are more significant methods in this category. DL is a field based on an artificial Neural Network that is used as the function in RL and suggests a deep Q-network (DQN) (Alex et al. 2012; Ian et al. 2012). DQN and Deterministic policy gradient (DPG) (David et al. 2014) are combined and used in Deep Deterministic Policy Gradient (DDPG) (Timothy et al., 2015). In RSs, Double DQN (DDQN) and Dueling Q-network are also used (Ziyu et al 2016).

5 REINFORCEMENT LEARNING FOR RECOMMENDATION

The user's interaction with an RS is sequential in nature. (Zimdars et al., 2001), and recommending the best items to a user is a sequential decision problem (Guy et al, 2005). This implies that the recommendation problem can be modeled as an MDP and solved using RL approaches. As previously stated, an agent in a normal RL situation seeks to maximize a numerical reward through interaction with an environment. This is similar to the recommendation problem, in which the RS algorithm seeks to recommend the best goods to the user while maximizing the user's pleasure. As a result, the RS algorithm can act as the RL agent, and everything outside of this agent, including system users and items, can be regarded as the agent's environment. Applying standard tabular RL algorithms to today's RSs with large action and state spaces is nearly impossible (Gabriel et al. 2015). Instead, with the emergence of DRL algorithms, there is a growing trend in the RS community to use RL approaches.

6 REINFORCEMENT LEARNING ALGORITHMS

We present algorithms in a classified manner in this part. After reviewing all the algorithms, we concluded that the emergence of DRL has significantly altered

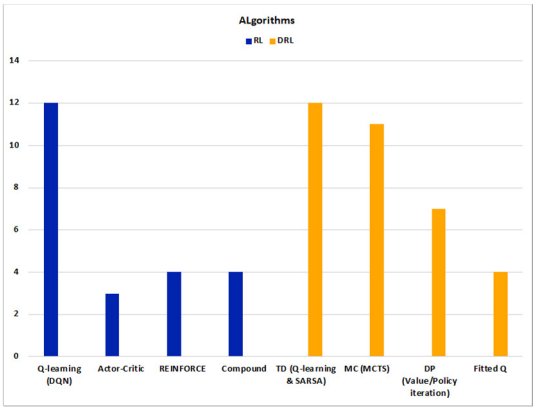

Figure 3: Number of publications using each algorithm.

the study of RLRSs. As a result, we split RLRS approaches into two broad categories: RL- and DRL-based algorithms. Fig. 3. gives a high-level overview of the algorithms and number of publications. We begin with RL-based approaches.

6.1 RL-based Methods

We mean RLRSs that use an RL algorithm for recommendation policy optimization but do not use DL to estimate parameters by RL-based methods. RL-based methods include TD, DP, MC, and Fitted Q RL algorithms from both tabular and approximate approaches.

6.1.1 TD Methods

Q-learning is a well-known RL algorithm in the RS field (Thorsten et al., 1997; Anongnrat and Pisit, 2005). WebWatcher is most likely the first RS method to integrate RL to improve suggestion quality. They simply treat the web page recommendation problem as an RL problem and apply Q-learning to increase the accuracy of their basic web RS, which employs a similarity function (based on TF-IDF) to propose pages that are related to the user's interest. Authors in (Nima et al., 2007) extend this idea a decade later to offer tailored websites to users. To address the state dimensionality issue, they employ the N-gram model from the online usage mining literature (Bamshad et al., 2000) and a sliding window to represent states.

6.1.2 DP Methods

Another tubular approach that has been used in (Thorsten and Anthony, 2001; Guy et al., 2005, Elad et al., 2014) is DP. Ref. (Thorsten and Anthony, 2001) is one of the early studies that formulate the recommendation problem as an MDP. In fact, the paper examines the potential benefits of utilizing MDP for the recommendation problem using the example of guiding a user through an airport. Similarly, ref. (Guy et al., 2005) is one of the early and valuable attempts to model the recommendation problem as an MDP. Because the model parameters of an MDP-based recommender are unknown and deploying it on the actual to learn them is prohibitively expensive, they propose a predictive model capable of providing starting parameters for the MDP. This prediction model is a Markov chain in which the state and transition function are modeled based on the dataset observations. They suggest that the simplest version of this Markov chain faces the data sparsity problem because it uses maximum probability to estimate the transition function. As a result, the basic version is improved by utilizing three techniques: skipping, clustering, and mixture modeling. This prediction model is then utilized to kickstart the MDP-based recommender. To address the dimensionality issue, the last k elements are employed to encode state information. They use an online study to evaluate the effectiveness of their strategy.

6.1.3 MC Methods

The final tabular approach, MC, has been used in various RLRSs (Elad et al., 2014; Yu, 2020; Lixin et al., 2019). To address the dimensionality issue, each song is represented as a vector of song (spectral auditory) descriptors, which include information about the song's spectral fingerprint, rhythmic features, overall loudness, and change over time. To expedite the process of learning, the reward function is also used to account for the listener's liking for certain songs as well as his song transition behavior. The DJ-MC architecture is made up of two primary parts: learning listener parameters (his preferences for songs and transitions) and arranging a song sequence.

6.1.4 Fitted Q Methods

Some RL-based algorithms (Yufan et al, 2011; Susan et al., 2011; Georgios et al., 2015) also employ an approximation technique (fitted Q) for policy optimization. In a clinical application (Yufan et al, 2011), RL is used to offer treatment alternatives for lung cancer patients with the goal of maximizing patient survival. They regard treatment for patients with advanced non-small cell lung cancer (NSCLC) as a two-line treatment, with the role of the RL agent being to recommend the best treatment option in each treatment line as well as the appropriate time to begin second-line therapy. Support vector regression (SVR) is used to optimize the Q-function for the RL agent. They alter SVR with a ε-insensitive loss function because the original SVR cannot be used to censored data (Vladimir, 2013). (Susan et al., 2011) employs RL to identify the best therapy alternatives for patients suffering from schizophrenia. First, they employ multiple imputations (Roderick et al., 2019) to solve the missing data issue, which can introduce bias and increase variance in Q-value estimations due to patient dropout or item missingness. The second issue they address is that clinical data is highly variable, with few trajectories, making function approximation difficult. As a result, they train the Q-function using fitted Q-iteration (FQI) (Damien et al, 2005) and a basic linear regression model. The impetus for the work presented in (Georgios et al., 2015) is that current ad suggestion algorithms do not distinguish between a visit and a visitor and presume that all visits to a website are from new visitors. As a result, they claim that, while click-through rate (CTR) is a realistic choice for greedy performance, life-time value (LTV), denoted as (total number of clicks/total number of visitors) $\times$ 100, is a true choice for long-term performance. To address the off-policy evaluation problem in the RS field, they employ a model-free approach dubbed HCOPE, developed by the same authors in (Philip et al., 2015), which computes a lower bound on a policy's expected return using a concentration inequality.

6.2 DRL-based Methods

In this part, we look at DRL-based RSs that employ DL to approximate the value function or policy. For policy optimization, these methods employ three essential RL algorithms: Q-learning, actor-critic, and REINFORCE. There are also other works that test and compare the performance of various RL algorithms for policy optimization.

6.2.1 Q-learning Methods

Slate-MDP (Peter, 2015) is possibly the first study to use DQN for a slate recommendation. To deal with the combinatorial action space caused by slates (tuples) of actions, they present agents that use a sequential greedy strategy to learn the value of whole

slates. In fact, it is anticipated that the item slates have the sequential presentation attribute, which means that recommended things are given to the user one at a time. This assumption is paired with another, in which it is assumed that one of the primitive acts will be performed. They also employ an attention mechanism based on DDPG for each slot in the slate to direct the search to a tiny area of action space with the highest value. However, as mentioned in (Eugene et al., 2019), the second assumption is not very realistic in frequent recommendation scenarios because it is akin to the condition in which we can force a user to eat a specific item.

DQN is used in (Shamim et al., 75) to optimize heparin dosage advice. They first represent the problem as a partially observable MDP (POMDP) and then estimate the belief states using a discriminative hidden Markov model. The policy is then optimized using DQN. A variation of DQN is employed in another clinical application (Aniruddh et al., 2017) to optimize dosage recommendations for sepsis treatment. They employ a continuous state space as well as a discrete action space. They alter DQN as follows due to intrinsic flaws in the original DQN algorithm, including the overestimation of Q values.

The fundamental idea of (Xinshi et al., 2019) is to utilize generative adversarial networks (GANs) to develop a user model and then use a cascade DQN algorithm to recommend the best things. A mini-max optimization strategy is used in user modeling to simultaneously learn user behavior and the reward function. DQN is then used to learn the optimal recommendation strategy using the learned user model. Unlike other approaches, instead of tackling the combinatorics of proposing a list of items (k items) with a single Q-network, k Q-networks are employed in a cascaded fashion to identify k best actions. To be more specific, the ideal actions are determined by the following fact:

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

6.2.2 Actor-Critique Methods

Wolpertinger (Nima et al., 2007) is an actor-critic framework capable of dealing with huge action spaces (up to one million). The goal is to create a method that is sub-linear in terms of action space and generalizable across activities. Wolpertinger is divided into two components. The first is action production, and the second is action refinement. In the first half, the actor generates proto-actions in continuous space, which are subsequently mapped to discrete space using the k-nearest neighbor approach. In the second section, outlier actions are filtered using a critic, which chooses the best action with the highest Q value. DDPG is also utilized to train their method. Wolpertinger is not primarily intended for RSs, although it can handle a recommendation task in a simulation study.

DDPG is also used for parameter training. One issue with this work is that it does not handle the combinatorics of action space when generating a list of items rather than proposing a single item. They later propose page-wise advice in (Nima et al., 2007). By recommending a group of complementary things and displaying them on a website, they mean recommending a set of complementary items and displaying them on a page. The actor is in charge of creating a page of stuff. To begin, two encoders are utilized to produce initial and current states. The actions are then generated by a decoder, namely a deconvolutional neural network. On the other hand, the current state (as determined by the same approach) and action was taken by the actor are sent into the critic, which employs a DQN architecture. DDPG is employed for model training once more. They also expand their work in e-commerce to a whole-chain recommendation (Xiangyu et al., 2019). Instead of having many scenarios in a user session, such as a welcome page and item pages, they employ a multiagent system with shared memory that optimizes all these situations at the same time (in fact, they only consider two pages in their studies: entry and item pages). Agents (actors) interact with the user sequentially and collaborate with one another to optimize the cumulative reward. On the other side, it is the global critic's responsibility to exert control over these actors. The global critic employs an attention method to capture user preferences in various settings, and each attention is active only in its specific scenario.

The recommender receives these states and uses a basic closest neighbor algorithm to provide recommendations. Finally, a historical critic is employed in order to limit the number of infractions to the user's preferences as specified in the user's earlier comments. While the use of multimodal data in this work is innovative, a thorough explanation of the actor-critic paradigm is lacking.

6.2.3 REINFORCE Methods

Authors in (Claudio et al., 2017) create a conversational RS based on hierarchical RL (Tejas, et al., 2016). There is a module in the framework called meta-controller that receives the dialogue state and anticipates the goal for that state. The work supports two types of goals: chitchat and recommendation.

The dialogue state is converted to a score vector by a goal-specific representation module, which is then refined by an attention module to highlight more relevant areas. Finally, a module known as the controller employs these revised scores to take action in order to meet the provided goal. In the framework, there are two critics: an external critic reviews the reward for the meta controller created by the environment, and an internal critic rewards the controller based on the aim set.

Ref. (Minmin et al., 2019) presents a useful study in the field of video recommendation using RL. The work's key contribution is the adaptation of the REINFORCE method to a neural candidate generator with a very wide action space. In an online RL scenario, the policy gradient estimator can be written as:

$$\sum_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{|r|} R_t \Delta_\theta \log \pi_\theta(a_t|s_t) \right] \quad (1)$$

where R_t is the total reward, π_θ is called the parametrized policy, and $\tau = (s_0, a_0, s_1, \ldots)$. Because, unlike in conventional RL situations, online or real-time interaction between the agent and environment is infeasible and frequently only logged feedback is provided, applying the policy gradient in Eq. (1) is biased and requires rectification. The policy gradient estimator that has been off-policy-corrected is then:

$$\sum_{\tau \sim \beta} \frac{\pi_\theta(\tau)}{\beta(\tau)} \left[\sum_{t=0}^{|r|} R_t \Delta_\theta \log \pi_\theta(a_t|s_t) \right] \quad (2)$$

where β is the importance weight and $\frac{\pi_\theta(\tau)}{\beta(\tau)}$ is the behavior policy Because this adjustment yields a large variance for the estimator, they utilize first-order approximation, resulting in the biased estimator with the reduced variance:

$$\sum_{\tau \sim \beta} \left[\sum_{t=0}^{|r|} \frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)} R_t \Delta_\theta \log \pi_\theta(a_t|s_t) \right] \quad (3)$$

The work's last contribution is top-K off-policy rectification. Setting (top-K) recommendations result in an exponentially increasing action space. The off-policy adjusted estimator described in Eq. (3) is modified to the following estimator for top-K recommendation under two assumptions:

$$\sum_{\tau \sim \beta} \left[\sum_{t=0}^{|r|} \frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)} \frac{\partial \alpha(a_t|s_t)}{\partial \pi(a_t|s_t)} R_t \Delta_\theta \log \pi_\theta(a_t|s_t) \right. \quad (4)$$

where α is the likelihood that an item α appears in the final non-repetitive set A (top-K items).

Finally, in (Huizhi, 2020), the environment is represented as a heterogeneous information network (graph) composed of persons, items, and other information sources such as content, tags, reviews, friends, and so on. The goal is to find a path in the graph between a user and an unobserved item. As shown below, the article employs a multi-iteration training procedure. A meta-path base (similar to a knowledge base) stores the meta-path computed at each iteration. The meta-path base is initially empty and is filled with meta-paths provided by experts. The meta-paths tried by the RL agent in each iteration are then added to the meta-path base. At the next iteration, the revised meta-path base is used to train the RL agent. This procedure is continued until no new information can be gathered or the maximum number of iterations is reached. The nearest neighbor algorithm is utilized for the top-K recommendation.

6.2.4 Compound Methods

Authors in (Su et al., 2018) utilize RL to recommend learning activities in a smart class, which is an unusual but intriguing use. Specifically, a cyber-physical-social system is constructed that monitors students' learning state by gathering multi-modal data such as test results, heartbeat, and facial expression, and then offers a learning activity that is appropriate for them.

Ref. (Floris et al., 2019) proposes and applies a task-oriented dialogue management system to various recommendation tasks. For dialogue management, two approaches are proposed: segmentation-based and state-based. The former separated the user population depending on the context, such as demographics and buying history, and each category has its own policy. The latter strategy is based on the concatenation of agent beliefs about past dialogues, user intents, and context. This belief vector is then loaded into a unique policy for all users. The work is evaluated using a benchmark (Inigo et al., 2017) in the field, which consists of a variety of recommendation tasks, such as recommending eateries in Cambridge or San Francisco.

7 ALGORITHMS OVERVIEW

7.1 RL-based Methods Recap

This section's RL approaches can be separated into tabular and approximation methods. DP approaches are typically impracticable among tabular methods due to their high computing cost and the requirement

for a perfect understanding of the environment. While the number of states in these algorithms is polynomial, implementing even one iteration of policy or value iteration methods is frequently infeasible (Andrew, 1995). DP is only used by two RS approaches in the RLRS literature (Guy et al., 2005; Changxi et al., 2019). To make it more practical, (Guy et al., 2005) employs a few features in their state space and employs some approximations. Similarly, (Tariq and Francesco, 2007) limits the number of policy iterations that can be conducted. Unlike DP, MC approaches do not require a perfect understanding (or model) of the environment. However, MC approaches have several disadvantages, such as the fact that they do not bootstrap. TD approaches, on the other hand, have been quite popular in the RS community (Thorsten et al., 1997; Anongnari and Pisit, 2005) The fundamental reason for their appeal is their simplicity; they are online, model-free, require little processing, and can be stated using a single equation (Richard and Andrew, 2017). In general, while tabular approaches may find the exact answer, i.e., the optimal value function and policy, they suffer from the curse of dimensionality as the state and action spaces grow, rendering them ineffective in learning. RLRSs that use DP and TD approaches attempt to address this issue by limiting the state space as small as possible. Methods based on MCTS must likewise preserve only the information from a sampling event, not the entire environment.

On the other hand, aside from the SARSA (λ) approach employed by (Mircea and Dan, 2005), the sole sort of approximate method used by RL-based RSs is the fitted Q method, which is a flexible framework that can fit any approximation architecture to the Q-function (Philip et al., 2015). As a result, any batch-mode supervised regression algorithm that can scale effectively to high dimensional spaces can be utilized to approximate the Q-function (Richard and Andrew, 2017). However, when the number of four-tuples ((x_t, u_t, r_t, x_{t+1}), where x_t represents 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) increases (Damien et al., 2005), the computational and memory cost may grow. Several RLRSs have used this algorithm (Yufan et al., 2011; Susan et al., 2011; Georgios et al., 2015).

7.2 DRL-based Methods Recap

The establishment of DRL marked a watershed moment in the history of RLRSs. This trend is clearly depicted in Fig. 1. DRL's unique ability to handle high-dimensional spaces makes it ideal for RSs with huge state and action spaces. DQN has been the most widely used DRL algorithm by RLRSs (Peter et al., 2015; Shamim et al., 2016; Su et al., 2018; Floris et al., 2019). DQN modified the original Q-learning algorithm in three ways, according to (Richard and Andrew, 2017): 1) It employs experience replay, a mechanism first introduced in (Long-Ji, 1992) that stores agents' experiences over various time steps in a replay memory and uses them to adjust weights throughout the training phase. 2) To simplify the complexity of updating weights, the current updated weights are fixed and fed into a second (duplicate) network, the outputs of which are utilized as Q-learning objectives. 3) To minimize the scale of error derivatives, the reward function is trimmed to be 1 for positive rewards, -1 for negative rewards, and 0 for no rewards. All of these changes proved to increase DQN's stability. However, as previously stated, DQN has certain issues. first, following the Q-learning method, DQN overestimates action values in some cases, making learning inefficient and perhaps leading to suboptimal policies (Sebastian and Anton, 1993). To address this issue, DDQN was proposed and is used by numerous RLRSs (Aniruddh et al., 2017). Second, DQN chooses events to replay at random, regardless of their significance, making the learning process slow and inefficient. Only four RLRS algorithms use an upgraded version of DQN's original experience replay mechanism, while the majority of DQN-based RLRSs use the original method. References (Aniruddh et al., 2017) employ prioritized experience replay (Jiahuan et al., 2019), ref. (Shi-Yong et al. 2018) uses stratified sampling rather than uniform sampling, and ref. (Tong et al., 2019) leverages cross-entropy of user interest to prioritize experiences. Third, DQN cannot handle continuous domains because it requires an iterative optimization process at each step, which is computationally prohibitively expensive. To address this issue, DDPG, which combines DQN and DPG, has been proposed.

Policy gradient methods, as opposed to action-value methods such as DQN, learn a parameterized policy without the use of a value function. When compared to action-value techniques, policy-based systems have three advantages (Richard and Andrew, 2017). 1) Policy approximate approaches can approach determinism; 2) Policy approximation may be simpler than value function approximation; and 3) Policy approximation methods can find stochastic optimal policies, whereas value-based methods cannot. REINFORCE and actor-critic approaches are two major policy gradient methods utilized by RSs.

REINFORCE is a Monte Carlo (MC) stochastic gradient approach that directly changes policy weights. The high variance and delayed learning of the REINFORCE algorithm is a key issue. These issues stem from REINFORCE's MC nature, as it takes samples at random. To address the high variance problem in the REINFORCE-based RLRSs reviewed, various techniques were used, including a neural network-based baseline (Yikun et al., 2019), first-order approximation (Satinder et al., 2000), REINFORCE with baseline algorithm (Inigo et al., 2017), weight capping. However, it is unclear how other REINFORCE-based RLRSs, such as (Huizhi, 2020), deals with this issue. Instead of a baseline, the actor-critic algorithm employs a critic to address the difficulties of REINFORCE. To be more specific, the critic is used to criticize the policy established by the actor; that is, it computes the value of the state-action pair provided by the actor and provides feedback on how good the action chosen is. The policy gradient approach now includes bootstrapping. While this creates a tolerable bias, it minimizes variance and speeds up learning (Richard and Andrew, 2017). As previously stated, DDPG is a well-known DRL technique that employs the actor-critic algorithm to handle continuous spaces. It is worth noting that among RLRSs, actor-critic is the second most common RL algorithm (Nima et al., 2007).

8 DISCUSSION AND FURURE WORKS

To begin, RL algorithms were designed to select one action from a set of possible actions. However, in the RS field, it is highly common to recommend a list of products. This is also known as slate, top-K, or list-wise recommendation. Except for a few (Georgios et al., 2015; Peter et al., 2015; Wacharawan et al., 2018; Eugene et al., 2019; Minmin et al., 2019), the vast majority of the algorithms examined to consider the problem of single item recommendation. Only references (Minmin et al., 2019; Georgios et al., 2015) properly study this topic and adapt their RL technique to deal with a number of issues. The problem of recommending a list of objects should be investigated more in the future when the RL agent confronts a wide combinatorial action space. Nonetheless, there is no apparent justification for employing a specific RL algorithm in an RS application. As a result, finding a relationship between the RL algorithm and the RS application is an important research direction for the future.

Explainable recommendation refers to an RS's capacity to not only make a recommendation but also to explain why that recommendation was made (Yongfeng and Xu, 2018). Explanation of recommendations made may improve user experience, increase trust in the system, and assist users in making better selections (Dan et al., 2003; Li and Pearl, 2005; Nava and Judith, 2007). Explainable approaches can be classified into two categories: model-intrinsic and model-agnostic (Zachary, 2018). In the former, an explanation is offered as part of the recommendation process, but in the latter, the explanation is supplied after the suggestion has been made. The method we discussed before (Yikun et al., 2019) could be an inherent explanatory method. In contrast, as a model-agnostic example (Xiting et al., 2018), RL is used to explain various recommendation approaches. The method employs a pair of agents, one of which is in charge of generating explanations and the other predicts whether the given explanation is satisfactory to the user. Debugging the unsuccessful RS (Xiting et al., 2018) is one intriguing application of explainable suggestion. That is, with the explanations supplied, we can follow the cause of problems in our system and determine which sections are malfunctioning. Only ref. (Yikun et al., 2019) supports an explainable recommendation among the RLRSs assessed in this survey, indicating that there is a gap in this area and that further attention is needed in the future. In conclusion, we feel that explainable recommendations are essential in the future generation of RSs, and that RL may be effectively used to generate better explanations.

Finally, RLRS evaluation should be enhanced. To learn what to do, an RL agent must directly interact with the environment. This is analogous to an online study for an RS; that is, the RS algorithm creates recommendations and receives user responses in real-time. Nonetheless, with the exception of a few approaches discussed (Eugene et al., 2019; Minminet al., 2019), the majority of the works use an offline study for evaluation. This is especially because of the high price of online research, as well as the high risk of implementing an RLRS to optimize its recommendation strategy for most organizations. As a result, offline evaluation, utilizing accessible datasets or simulation, is critical for RLRSs evaluation.

The future research could focus on different areas. For instance, dealing with one of the most significant issues is keeping track of all users' global and local states. Proposing and implementing RLRSs capable of managing large state space is a research subject that is rarely explored.

9 CONCLUSIONS

We offered a complete and up-to-date survey of RLRSs in this work. We emphasized the importance of DRL in changing the research direction in the RLRS field and, as a result, categorized the algorithms into two broad categories: RL- and DRL-based approaches. Following that, each broad group was subdivided into sub-categories based on the RL algorithm employed, such as Q-learning and actor-critic. We feel that research on RLRSs is in its infancy and that significant progress is needed. Both RL and RSs are active research topics that are of particular interest to large corporations and industries. As a consequence, we may anticipate new and intriguing models to emerge each year. Finally, we believe that our survey will help researchers understand crucial concepts and progress in the field in the future.

REFERENCES

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, Senthil Yogamani (2017). Deep reinforcement learning framework autonomous driving. Electronic Imaging, 2017(19):70-76.

Arthur Guez, Robert D Vincent, Massimo Avoli, Joelle Pineau (2008). Adaptive treatment of epilepsy via batch-mode reinforcement learning. In AAAI, pages 16711678.

Aleksandra Klasnja-Milicevic, Mirjana Ivanovic (2015), Alexandros Nanopoulos. Recommender systems in e learning environments: a survey of the state of the art and possible extensions. Artficial Intelligence Review, 44(4):571-604.

Alex Krizhevsky, Ilya Sutskever, and Georey E Hinton (2012). Image net classification with deep convolutional neural networks. Neural information processing systems, pages 1097-1105.

Andrew G Barto (1995). Reinforcement learning and dynamic programming. In Analysis, Design and Evaluation of Man Machine Systems, pages 407-412. Elsevier.

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

Anongnart Srivihok and Pisit Sukonmanee (2005). E-commerce intelligent agent: personalization travel support agent using q learning.7th international conference on Electronic commerce, pages 287-292.

Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava(2000). Automatic personalization based on web usage mining. ACM, 43(8):142-151.

Bellogin, I. Cantador, F. Diez, P. Castells, E. Chavarriaga (2013), An empirical comparison of social, collaborative filtering, and hybrid recommenders, ACM Transactions on Intelligent Systems and Technology (TIST), 4 1-29.

Burke (2002), Hybrid recommender systems: survey and experiments, User Model User-Adap Inter, 12 331-370.

Ben Schafer, Joseph Konstan, John Riedl (1999). Recommender systems in e-commerce. ACM conference on Electronic commerce, pages 158-166.

Billsus, M. Pazzani (2000), User modeling for adaptive news access, User Model User-Adap Inter, 10 147-180.

Burke (2007), Hybrid web recommender systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.) The Adaptive Web, Springer-Verlag, Berlin Heidelberg, pp. 377-408.

Changxi You, Jianbo Lu, Dimitar Filev, Panagiotis Tsiotras (2019). Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. Robotics and Autonomous Systems, 114:118.

Christopher John Cornish Hella by Watkins(1989). Learning from delayed rewards.

Claudio Greco, Alessandro Suglia, Pierpaolo Basile, and Giovanni Semeraro (2017). Converseet-impera: Exploiting deep learning and hierarchical reinforcement learning for conversational recommender systems. Italian Association for Artificial Intelligence, pages 372-386. Springer.

Damien Ernst, Pierre Geurts, and Louis Wehenkel (2005). Tree-based batch mode reinforcement learning. Journal of Machine Learning Research, 6(Apr):503-56.

Dan Cosley, Shyong K Lam, Istvan Albert, Joseph A Konstan,, John Riedl (2003). Is seeing believing how recommender system interfaces a ect users' opinions. conference on Human factors in computing systems, pages 585-592.

David Goldberg, David Nichols,Douglas Terry, Brian M Oki (1992).Using collaborative filtering toweave an information tapestry. ACM, 35(12):61-70.

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

Deshpande, G. Karypis (2004), Item-based top-N recommendation algorithms, ACM Transactions on Information Systems (TOIS), 22 143-177.

Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone (2014). Dj-mc: A reinforcement earning agent for music playlist recommendation. arXiv

Emre Sezgin, Sevgi Ozkan (2013). A systematic literature review on health recommendersystems. E-Health and Bioengineering Conference (EHB), pages 1-4. IEEE.

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

Floris Den Hengst, Mark Hoogendoorn, Frank Van Harmelen, and Joost Bosman (2019). Reinforcement learning for personalized dialogue management. International Conference on Web Intelligence.

Francesco Ricci, Lior Rokach, Bracha Shapira (2011). Introduction to recommender systems handbook. In Recommender systems handbook, pages 1-35.

Gabriel Dulac-Arnold, Richard Evans, Peter Sunehag, Hado

van Hasselt, Timothy Lillicrap, Timothy Mann, Jonathan Hunt, Theophane Weber, Ben Coppin,

Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh (2015). Personalized ad recommendation systems for lifetime value optimization with guarantees. In Twenty-Fourth International Joint Conference on Artificial Intelligence.

Guy Shani, David Heckerman, Ronen I Brafman (2005). An mdp based recommender system. Machine Learning ResearchJournal , 6 (Sep) : 1265-1295.

Huizhi Liang. Drpro ling (2020): deep reinforcement user pro ling for recommendations in heterogenous information networks. IEEE on Knowledge and Data Engineering.

Inigo Casanueva, Pawe l Budzianowski, Pei-Hao Su, Nikola Mrk si c, Tsung-Hsien Wen, Stefan Ultes, Lina Rojas-Barahona, Steve Young, and Milica Ga si c (2017). A benchmarking environment for reinforcement learning based task oriented dialogue management. arXiv preprint arXiv:1711.11023.

Jens Kober, J Andrew Bagnell, Jan Peters (2013). Reinforcement learning in robotics: A survey. Journal of Robotics Research, 32(11) : 1238 - 1274.

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

Joel Tetreault and Diane Litman (2006). Using reinforcementlearning to build a better model ofdialogue state. European Chapter ofthe Association for Computational Linguistics.

Li Chen, Pearl Pu (2005). Trust building in recommender agents Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces at the 2nd International Conference on E-Business s, pages 135-145. Citeseer.

Lixin Zou, Long Xia, Zhuoye Ding, Dawei Yin, Jiaxing Song, and Weidong Liu (2019). Reinforcement learning to diversify top-n recommendation. International Conference onDatabase Systems for Advanced Applications, pages 104-120. Springer.

Michael J Pazzani and Daniel Billsus (2007). Content-based recommendation systems. In The adaptive web, pages 325341. Springer.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, Ed H Chi (2019). Top-k offpolicy correction for a reinforce recommender system. ACM International Conference on Web Search and DataMining, pages 456464.

Mircea Preda and Dan Popescu (2005). 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.

Mobasher, X. Jin, Y. Zhou (2004), Semantically enhanced collaborative filtering on the web, in: B. Berendt, A. Hotho, D.Mladenic, M. Someren,˘ M. Spiliopoulou, G. Stumme (Eds.) From Web to Semantic Web, Springer, pp. 57-76.

Mozhgan Karimi, Dietmar Jannach, Michael Jugovac(2018). News recommender systems survey and roads ahead. Information Processing and Management, 54(6):1203-1227.

Nava Tintarev, Judith Mastho (2007): Exective explanations of recommendations: usercentered design.ACM conference on Recommender systems, pages 153-156.

Nima Taghipour, Ahmad Kardan, Saeed Shiry Ghidary (2007). Usage based web recommendations:a reinforcement learning approach. ACM conference on Recommender systems, pages 113-120.

Paul Resnick and Hal R Varian (1997). Recommender systems. ACM, 40(3):56-58.

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

Pazzani (1999), A framework for collaborative, content based and demographic filtering, Artificial Intelligence Review, 13 393-408.

Pazzani, D. Billsus (2007), Content-based recommendation systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.) The Adaptive Web, Springer Berlin Heidelberg, pp. 325-341.

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

Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh (2015). High-con dence off-policy evaluation. In Twenty-Ninth AAAI Conference on Artficial Intelligence.

Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl (1994). GroupLens : an open architecture forcollaborative filtering of netnews, ACM Conference on Computer Supported Cooperative Work, ACM, Chapel Hill, North Carolina, USA, , pp. 175-186.

Richard S Sutton and Andrew G Barto (2017). Introduction to reinforcement learning, volume 2. MIT press Cambridge.

Richard Meyes, Hasan Tercan, Simon Roggendorf, Thomas Thiele, Christian Buscher, Markus Obdenbusch, Christian Brecher, Sabina Jeschke, and Tobias Meisen (2017). Motionplanning for industrial robots using reinforcement learning. Procedia CIRP, 63:107-112.

Ronald J Willams (1992). Simple statistical gradient following algorihms for connectionist reinforcement learning. Machine learning, 8(3-4):229-256

Roderick JA Little and Donald B Rubin (2019). Statistical analysis with missing data, volume 793. John Wiley

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

Satinder P Singh, Michael J Kearns, Diane J Litman, Marilyn A Walker (2000). Reinforcement learning for spoken dialogue systems. Neural Information Processing Systems, pages 956-962.

Sarwar, G. Karypis, J. Konstan, J. Riedl (2001), Item-based collaborative filtering recommendation algorithms, 10th International Conference on World Wide Web, ACM, pp. 285-295.

Sebastian Thrun and Anton Schwartz (1993). Issues in using function approximation for reinforcementlearning. Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum.

Shambour, J. Lu (2011), A hybrid trust-enhanced collaborative filtering recommendation approach for personalized government-to-business eservices, International Journal of Intelligent Systems, 26 814843.

Shamim Nemati, Mohammad M Ghassemi, and Gari D Cli ord (2016). Optimal medication dosing fromsuboptimal clinical examples:A deep reinforcementlearning approach. Engineering in Medicine and Biology Society, pages 2978-2981. IEEE.

Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang (2018). 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.

Shuai Zhang, Lina Yao, Aixin Sun, Yi Tay (2019). Deep learningbased recommender system :A survey and new perspectives. Computing Surveys (CSUR), 52(1):1-38,

Smyth, P. Cotter (2000), A personalised TV listings service for the digital TV age, Knowledge-Based Systems.

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

Tariq Mahmood and Francesco Ricci (2007). Learning and adaptivity in interactive recommender systems. Conference on Electronic commerce, pages 75-84.

Thomas Degris (2015). Deep reinforcement learning in large discrete action spaces. arXiv : 1512.07679.

Thorsten Bohnenberger and Anthony Jameson (2001). When policies are better than plans: Decision theoretic planning of recommendation sequences. International Conference on intelligent user interfaces, pages 21-24.

Thorsten Joachims, Dayne Freitag, Tom Mitchell(1997). Webwatcher: A tour guide for the world wide web. In IJCAI (1), pages 770{777. Citeseer.

Timothy P Lillicrap, Alexander Pritzel, Jonathan J Hunt, Nicolas Heess, Yuval Tassa, Tom Erez, David Silver, Daan Wierstra (2015). Continuous control with deep reinforcement learning. arXiv.

Tong Yu, Yilin Shen, Ruiyi Zhang, Xiangyu Zeng, and Hongxia Jin (2019). Vision-language recommendation via attribute augmented multimodal reinforcement learning. ACM International Conference on Multimedia, pages 39-47.

Vladimir Vapnik (2013). The nature of statistical learning theory. Springer science & business media.

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

Yufan Zhao, Donglin Zeng, Mark A Socinski, and Michael R Kosorok (2011). Reinforcement learning strategies forclinical trials in nonsmall cell lung cancer.

Nima Taghipour, Ahmad Kardan, Saeed Shiry Ghidary (2007). Usage based web recommendations: a reinforcement learning approach. In Proceedings of the 2007 ACM conferenceon Recommender systems.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. Neural information processing systems, pages 3675-3683.

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

Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, Yongfeng Zhang (2019). Reinforcement knowledge graph reasoning for explainable recommendation. ACM SIGIR Conference on Research and Development in Information Retrieval, pages 285-294.

Wilson, B. Smyth, D. O'Sullivan (2003), Sparsity reduction in collaborative recommendation: A case-based approach, Journal of Pattern Recognition andArtificial Intelligence, 17863-884.

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

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

Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, Xing Xie (2018). A reinforcement learning framework for explainable recommendation. Conference on Data Mining, pages 587-596. IEEE.

Yongfeng Zhang, Xu Chen (2018). Explainable recommendation: A survey and new perspectives. arXiv:1804.11192.

YuWang (2020). A hybrid recommendation for music based on reinforcement learning. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 91-103. Springer,

Zachary C Lipton (2018). The mythos of model interpretability. Queue, 16(3):31-57

Zhengyao Jiang, Dixing Xu, Jinjun Liang (2017). A deep reinforcement learning framework for the nancial portfolio management problem. arXiv.

Zimdars, D. M. Chickering, C. Meek (2001). Using temporal data for making recommendations. In 17th Conference in Uncertainty in Articial Intelligence.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, Nando Freitas (2016). Dueling network architectures for deep reinforcement learning. In International conference on machine learning.