A Survey on Reinforcement Learning for Recommender Systems

Yuanguo Lin, Yong Liu, Member, IEEE, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, Senior Member, IEEE, and Chunyan Miao, Fellow, IEEE

Abstract—Recommender systems have been widely applied in different real-life scenarios to help us find useful information. In particular, reinforcement learning (RL)-based recommender systems have become an emerging research topic in recent years, owing to the interactive nature and autonomous learning ability. Empirical results show that RL-based recommendation methods often surpass supervised learning methods. Nevertheless, there are various challenges in applying RL in recommender systems. To understand the challenges and relevant solutions, there should be a reference for researchers and practitioners working on RL-based recommender systems. To this end, we first provide a thorough overview, comparisons, and summarization of RL approaches applied in four typical recommendation scenarios, including interactive recommendation, conversational recommendation, sequential recommendation, and explainable recommendation. Furthermore, we systematically analyze the challenges and relevant solutions on the basis of existing literature. Finally, under discussion for open issues of RL and its limitations of recommender systems, we highlight some potential research directions in this field.

Index Terms—Interactive recommendation, policy gradient, recommender systems, reinforcement learning (RL).

I. INTRODUCTION

PERSONALIZED recommender systems [1], [2], [3] are competent to provide interesting information that matches users’ preferences, thereby alleviating the information overload problem. Recommendation technologies usually make use of various information to provide potential items for users. To this end, the early recommendation research primarily focuses on developing content- and collaborative filtering-based methods [4], [5]. Recently, motivated by the quick developments of deep learning, various neural recommendation methods have been developed [6]. However, modeling various information is not enough. In real-world scenarios, the recommender system suggests items according to the user–item interaction history and then receives user feedback to make further recommendations [7], [8]. In other words, the recommender system aims to obtain users’ preferences from the interactions and recommend items that users may be interested in. Nevertheless, existing recommendation methods (e.g., supervised learning) usually ignore the interactions between a user and the recommendation model. They do not effectively capture the user’s timely feedback to update the recommendation model, thus leading to unsatisfactory recommendation results.

In general, the recommendation task could be modeled as an interactive process, i.e., the user is recommended an item and then provides feedback (e.g., skip, click, or purchase) for the recommendation model. In the next interaction, the recommendation model learns the optimal policy from the user’s explicit/implicit feedback and recommends a new item to the user. From the user’s point of view, an efficient interaction means helping users find their favorite items as soon as possible. The interactive recommendation approach has been applied in real-world recommendation tasks. However, it often suffers from some problems, e.g., cold-start [9], [10], data sparsity [11], interpretability [12], and safety [13].

As a machine learning method that focuses on how an intelligent agent interacts with its environment, reinforcement learning (RL) [14], [15] learns the policy by trial and error search, which is beneficial to sequential decision-making. Hence, it can provide potential solutions to model the interactions between the user and agent. In particular, deep RL (DRL) [16], the combination of traditional RL with deep learning methods, is competent to learn from historical data with enormous state and action spaces to address large-scale problems. It has powerful representation learning and function approximation properties to be applied across various fields [17], [18], e.g., games [19] and robotics [20]. Recently, the application of RL to solve recommendation problems has become a new research trend in recommender systems [21], [22], [23]. Specifically, RL enables the recommender agent to constantly recommend items to users for learning the optimal recommendation policies. Many experimental results have demonstrated that RL-based recommendation methods [24], [25] evidently outperform supervised learning methods. For
example, as shown in Table I, RL-based recommendation methods (i.e., PGPR [26], actor-critic [27], and ADAC [28]) consistently perform better than the supervised learning-based recommendation methods (i.e., DKN [29], BPR [30], and RippleNet [31]) on two Amazon datasets in terms of hit ratio (HR) and normalized discounted cumulative gain (NDCG), especially with significant margin ($p$-value < 0.01) on the Clothing dataset. In practice, RL-based recommender systems have been applied to many specific scenarios [32], [33], [34], [35], [36], [37], such as e-commerce [38], [39], [40], e-learning [41], [42], and health care [43], [44], [45].

A. Data Collection Methodology

There are a growing number of studies of RL-based recommender systems. To search for relevant articles for the analysis, we adopted the following collection rules to include or exclude papers.

1) Search Terms: Our survey involves two keywords: RL$^1$ and recommender systems. Accordingly, we mainly adopted the following search terms: “Reinforcement Learning” and “Recommender Systems.” To find more research papers, we also used related RL algorithms as the search terms (e.g., “Q-Learning,” “Policy Gradient,” and “Actor-Critic”) instead of “Reinforcement Learning,” along with “Recommender Systems.” Similarly, we used the search term “Recommendation” instead of “Recommender Systems,” along with “Reinforcement Learning.”

2) Search Sources: We first used Google Scholar to find research papers with these search terms. We then increased the collection of relevant articles through the following academic databases: Science Direct, ACM Portal, Springer Link, and IEEE Xplore. Finally, we selected 98 related papers to include in this survey.

3) Publication Type: Only high-level publications on RL-based recommendations from international conferences and top journals were included in our survey.

The distribution of collected research papers over the years is shown in Fig. 1. There are a few publications on traditional RL-based recommendation methods from 2005 to 2017, with an increase in the number of papers on DRL-based recommendation methods since 2018. The main reason is that DRL algorithms have proved to be satisfactory solutions to some recommendation issues, and they have attracted much attention from the research community.

B. Related Work

To facilitate the research about RL-based recommender systems, Afsar et al. [46] provide a review of the RL- and DRL-based algorithms developed for recommendations and present several research directions in the top-$K$ recommendation, application architecture, and evaluation. Besides, Chen et al. [47] provide an overview of DRL-based recommender systems mainly according to model-based and model-free algorithms, and discuss the benefits and drawbacks of DRL-based recommendations. Nevertheless, it is necessary to make a more comprehensive overview and analysis of (D)RL-based recommender systems.

C. Our Contribution

Different from [46] and [47], the main contributions made in this work are given as follows.

1) Comprehensive Review: We summarize existing (D)RL algorithms applied in four typical recommendation scenarios, i.e., interactive recommendation, conversational recommendation, sequential recommendation, and explainable recommendation. It could be helpful for readers to understand how (D)RL algorithms are applied in different recommendation systems. Moreover, from the RL perspective, the comprehensive survey of RL-based recommender systems follows three classes of RL algorithms: value function, policy search, and actor-critic. This taxonomy of the literature is made based on the fact that these three types of methods have been widely applied in existing RL-based recommender systems.

2) Systematical Analysis: We systematically analyze the challenges of applying (D)RL in recommender systems and relevant solutions, including environment construction (e.g., the state representation and negative sampling), prior knowledge, reward function definition, learning bias (e.g., the data or policy bias), and task structuring (i.e., the task of RL can be decomposed into basic components or a sequence of subtasks).

3) Open Directions: To facilitate future progress in this field, we put forward open issues of RL, analyze the practical challenges of this field, and suggest possible future directions for the research and application of recommender systems. The open issues and emerging topics contain sampling efficiency, reproducibility, generalization, evaluation, biases, interpretability, safety, and privacy.

The remainder of this article is organized as follows. Section II introduces the background of RL, defines related concepts, and lists commonly used approaches. Section III presents a standard definition of the RL-based recommendation methods. Section IV provides a comprehensive review of the RL algorithms developed for recommender systems. Then, Section V discusses the challenges and relevant solutions of applying RL in recommender systems. Next, Section VI discusses various limitations and potential research directions.

---

$^1$Note that this survey focuses on recommender systems using full RL. Therefore, we did not include bandits, which are different from full RL.
of RL-based recommender systems. Finally, Section VII concludes this study.

II. OVERVIEW OF REINFORCEMENT LEARNING

Different from supervised and unsupervised learning, RL [48] focuses on goal-directed learning that maximizes the total reward achieved by an agent when interacting with its environment. Trial-and-error and delayed rewards are the two most important characteristics that distinguish RL from the other types of machine learning methods.

In RL, the agent learns the optimal policy from its interactions with the environment to maximize the total reward for sequential decision-making. As shown in Fig. 2(a), the learning process of RL can be formulated as a Markov decision process (MDP) [49]. Formally, the MDP is defined as a 5-tuple $(S, A, \mathbb{P}, R, \gamma)$, where $S$ denotes a finite set of states, $A$ denotes a finite set of actions, $\mathbb{P}$ is the state transition probabilities, $R$ denotes the reward function, and $\gamma \in [0, 1]$ is a discount factor of the reward. At each time step $t$, the agent receives an environment state $S_t \in S$ and selects the corresponding action $A_t \in A$. Then, the agent receives a numerical reward $R_{t+1} \in \mathbb{R}$ and makes itself into a new state $S_{t+1}$. Thus, the MDP forms a sequence $\tau$ as follows:

$$\tau = \{S_0, A_0, R_1, S_1, A_2, \ldots, S_T\}$$

where $T$ is the maximum time step in a finite MDP. To maximize the return in the long run, the agent tries to select actions so that the cumulative reward that it receives over the future is maximized. In this case, we introduce the concept of discounting. In general, the agent selects an action $A_t$ to maximize the discounted return $G_t$ [49]

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \sum_{i=t+1}^{T} \gamma^{i-t-1} R_i$$

The discount factor $\gamma$ affects the return. If $\gamma = 0$, the agent only maximizes immediate rewards that reduce the return in the long run. As $\gamma$ approaches 1, the agent takes future rewards into account more strongly.

Based on whether to use models and planning for solving RL problems, RL algorithms can be classified into two main groups, i.e., model-free algorithms and model-based algorithms. The model-free algorithms directly learn the policy without any model of the transition function, whereas the model-based algorithms employ a learned or predetermined model to learn the policy. On the other hand, according to the way of action conducted by the agent, RL algorithms fall into the following major groups, i.e., value-function methods, policy search methods, and actor-critic methods.

A. Value-Function Approaches

Many traditional RL methods generally achieve a global optimum return by obtaining the maximal value in terms of the best action. These methods are called value-function approaches, which utilize the maximal value to learn the optimal policy indirectly. Intuitively, the maximal value is generated by the best action $a^*$ following the optimal policy $\pi^*$, that is, the state-value under an optimal policy equals the expected return for the best action from the state. This is the Bellman equation for the optimal state-value function $v^*_\pi(s)$, or called the Bellman optimality equation defined as

$$v^*_\pi(s) = \max_{a} \mathbb{E}[R_{t+1} + \gamma v^*_\pi(S_{t+1}) | S_t = s, A_t = a]$$

$$= \max_{a} \sum_{s', r} p(s', r | s, a) \left[r + \gamma v^*_\pi(s')\right]$$

where $\mathbb{E}[]$ denotes the expected value of a random variable given by following the optimal policy $\pi^*$.

Similarly, the Bellman optimality equation for the action-value function $q^*_\pi(s, a)$ is defined by

$$q^*_\pi(s, a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q^*_\pi(S_{t+1}, a') | S_t = s, A_t = a\right]$$

$$= \sum_{s', a'} p(s', r | s, a) \left[r + \gamma \max_{a'} q^*_\pi(s', a')\right].$$

Different from the optimal state-value function $v^*_\pi(s)$, $q^*_\pi(s, a)$ explicitly reflects the effects of the best action, which is more prone to be adopted by many algorithms.

In general, the optimal value function is estimated by dynamic programming (DP), Monte Carlo methods, or temporal-difference (TD) learning, such as Sarsa [49], Q-Learning [50], and deep Q-networks (DQNs) [51]. Compared to policy-search methods, the value-function approaches are often of vastly reduced variance. Nevertheless, they are not suitable for complex application scenarios due to the slow convergence.

B. Policy Search Methods

In contrast to value-function approaches, policy-search methods directly optimize the policies, which are parameterized by a set of policy parameters $\theta_i$. These policy parameters

| Dataset    | Beauty | HR  | NDCG | p-value | Cloth | HR  | NDCG | p-value |
|------------|--------|-----|------|---------|-------|-----|------|---------|
| DKN        | 8.67±0.058 | 1.872±0.049 | 0.021 | 1.203±0.088 | 0.300±0.024 | 1.45E-05 |
| BPR        | 9.021±0.068 | 2.744±0.045 | 0.036 | 1.820±0.061 | 0.609±0.027 | 6.95E-05 |
| RippleNet  | 9.294±0.027 | 2.401±0.036 | 0.040 | 1.882±0.041 | 0.624±0.025 | 8.07E-05 |
| PGPR       | 14.559±0.051 | 5.513±0.042 | -     | 7.003±0.032 | 2.843±0.030 | -     |
| Actor-Critic | 14.821±0.043 | 5.730±0.051 | 0.912 | 6.924±0.044 | 2.796±0.031 | 0.949 |
| ADAC       | 15.856±0.053 | 5.863±0.048 | 0.718 | 7.501±0.022 | 3.010±0.058 | 0.748 |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
can be updated to maximize the expected return with either gradient-free or gradient-based optimization methods [16]. As one of the most popular RL algorithms, the gradient-based optimization method learns the policy parameters with the gradient of some performance measure $J(\theta)$. Formally, the updates approximate gradient ascent in $J(\theta)$ by

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

(5)

where $\nabla J(\theta_t)$ denotes a stochastic estimate that approximates the gradient of $J(\theta_t)$ in terms of $\theta_t$ and $\alpha$ is the step size that influences the learning rate [49]. Existing policy gradient methods generally follow the gradient updating strategy in (5).

The policy gradient methods provide an appropriate equation proportional to the policy gradient, which may need a Monte Carlo-based method of sampling their expectation that approximates the equation. Thus, the REINFORCE algorithm [52] that adopts the Monte Carlo policy gradient method can be established by

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \nabla \log \pi(a|s, \theta) q(s, a)$$

$$\propto \mathbb{E}_\pi \left[ \sum_a \nabla \log \pi(a|s_t, \theta) q(s_t, a) \right]$$

(6)

where the symbol $\propto$ refers to “proportional to,” the distribution $\mu(s)$ is the on-policy distribution under the policy $\pi$, and the gradients are column vectors of partial derivatives with respect to the policy parameter $\theta$.

Some advanced algorithms have been proposed to address the shortcomings of policy-search methods. For example, policy gradient methods with function approximation [53] ensure the stability of the algorithms. By adjusting superparameters artificially or adaptively, proximal policy optimization (PPO) algorithms [54] and trust region policy optimization (TRPO) [55] speed up the convergence of the algorithms. Moreover, guided policy search (GPS) [56] utilizes the path optimization algorithm to guide the training process of the policy gradient method and, thereby, improves its efficiency.

C. Actor-Critic Algorithms

There are a set of algorithms that incorporate the advantages of value-function approaches and policy search methods. They attempt to estimate a value function and, meanwhile, adopt the policy gradient to search in the policy space. Actor-critic [57] is one of the most representative algorithms. It combines the policy-based method (i.e., the actor) with the value-based approach (i.e., the critic) to learn the policy and value function together. The actor trains the policy according to the value function of the critic’s feedback, while the critic trains the value function and uses the TD method to update it in one step. The one-step actor-critic algorithm replaces the full return with the one-step return as

$$\theta_{t+1} = \theta_t + \alpha (R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)) \nabla_\theta \pi(A_t|S_t, \theta_t)$$

(7)

where $w$ is the state-value weight vector learned by the actor-critic algorithm [49]. In (7), $\hat{v}(S_t, w)$ is a learned state value function that is used as the baseline. In recent years, many improved actor-critic algorithms have been developed, such as asynchronous advantage actor-critic (A3C) [58], soft actor-critic (SAC) [59], deterministic policy gradient (DPG) [60], and its variation deep DPG (DDPG) [27]. Actor-critic algorithms may alleviate the problem of sampling efficiency by the experience replay [61]. However, due to the coupling of value evaluation and policy updates, the stability of these algorithms is unsatisfactory.

III. FORMULATION OF RECOMMENDATION PROBLEM

In a typical recommender system, suppose that there are a set of users $U$ and a set of items $I$, with $R \in \mathbb{R}^{X \times Y}$ denotes the user–item interaction matrix, where $X$ and $Y$ denote the number of users and items, respectively. Let $s_t$ denote the interaction behavior between user $u$ and item $i$ at time step $t$. The recommender system aims to generate a predicted score $\hat{r}_{ui}^t$, which describes the user’s preference for the item $i$.

Generally, we can formulate the recommendation task as a finite MDP [62], [63], as shown in Fig. 2(b). At each time step, the recommender agent interacts with the environment (i.e., the user and/or logged data) by recommending an item to the user in the current state. In the next time step, the recommender agent receives feedback from the environment and recommends a new item to the user in a new state. The user’s feedback may contain explicit feedback (e.g., purchase and rating) or implicit feedback (e.g., user’s browsing record from logged data). The recommender agent aims at learning an optimal policy with the policy network to maximize the
cumulative reward. More precisely, the MDP in the recommendation scenario is a 5-tuple \( \langle S, A, P, R, \gamma \rangle \), which can be defined as follows.

1) **States** \( S \): The finite state space describes the environment states in the fixed length history trajectories, in which \( S_t = \{i_1, i_2, \ldots, i_t\} \) is an observed state from the sequence of interacted items at time step \( t \).

2) **Actions** \( A \): The discrete action space contains the whole set of recommended candidate items. An action \( A_t \) is to recommend an item \( i \) at time step \( t \). In logged data, the action can be taken from the user–item interactions.

3) **Transition Probability** \( P \): It is the state transition probability matrix. The state transits from \( s \) to \( s' \) according to the probability \( p(s', r|s, a) \) after the recommender agent receives the user’s feedback (i.e., the reward \( r \)).

4) **Reward Function** \( R \): Once the recommender agent takes action \( a \) at state \( s \), it obtains the reward \( r(s, a) \) in accordance with the user’s feedback.

5) **Discount Factor** \( \gamma \): It is the discount-rate parameter for future rewards.

We assume that, in an online RL-based recommendation environment, the recommender agent recommends an item \( i_t \) to a user \( u \), while the user provides feedback \( f_t \) for the recommender agent at the \( t \)th interaction. The recommender agent obtains the reward \( r(S_t, A_t) \) associated with the feedback \( f_t \) and recommends a new item \( i_{t+1} \) to the user \( u \) at the next interaction. Given the observation on the multinurn of interactions, the recommender system generates a recommendation list. The recommender agent aims to learn a target policy \( \pi^*_0 \) to maximize the cumulative reward of the sampled sequence [62]

\[
\max_{\pi_0} \mathbb{E}_{r \sim p_\pi}[R_T] \tag{8}
\]

where \( \theta \) refers to policy parameters and \( R_T = \sum_{t=0}^{\tau} \gamma^t r(S_t, A_t) \) denotes the cumulative reward with respect to the sampled sequence \( \tau = \{S_0, A_0, \ldots, S_T\} \).

The recommender agent often suffers from the high cost to learn the target policy by interacting with real users online. An alternative is to employ offline learning, which learns a behavior policy \( \pi_b \) from the logged data. We should solve the policy bias to learn an optimal policy \( \pi^* \) when using offline learning since there is a noticeable difference between the target policy \( \pi_0 \) and the behavior policy \( \pi_b \).

IV. RECOMMENDER SYSTEMS BY REINFORCEMENT LEARNING

In this section, we summarize specific RL algorithms applied in four typical recommendation scenarios (i.e., interactive recommendation, conversational recommendation, sequential recommendation, and explainable recommendation) following value function, policy search, and actor-critic, respectively. An overview of related literature is shown in Table II. Note that some models/frameworks have an offline evaluation strategy in the offline experimental environment,

\[\text{Note that the action space may include other kinds of actions in different recommendation scenarios, e.g., the selection of query attributes in conversational recommender systems and the outgoing edges of entities in KG-based explainable recommender systems.}\]
for all target users. To model user-specific preferences explicitly, a biased User-specific Deep Q-learning Network (UDQN) is proposed by adding a bias parameter to capture the difference in the $Q$ values of different target users.

2) Policy Search Methods: Existing RL-based algorithms are often developed for the short-term recommendation, whereas Huang et al. [9] employ DRL and recurrent neural network (RNN) to improve the accuracy of long-term recommendation. Specifically, RNN is performed to simulate the sequential interactions between the environment (the user) and the recommender agent by evolving user states adaptively. This strategy can help tackle the cold-start issue in recommender systems. On the other hand, the interaction process is split into subepisodes and reboots the accumulated reward of each subepisode, which significantly improves the effectiveness of policy learning.

In text-based interactive recommender systems, user feedback with natural language usually causes undesired issues. For instance, the recommender system may violate user’s preferences since it ignores the previous interactions and, thus, recommends similar items. To handle these issues, a reward-constrained recommendation (RCR) model [72] is proposed to incorporate user preferences sequentially. More specifically, the text-based interactive recommendation is formulated as a constraint-augmented RL problem, where the user feedback is taken as a constraint. To generalize from the constraints, a discriminator parameterized as the constraint function is developed to detect the violation of user’s preferences. The policy is optimized by the policy gradient with baseline (i.e., a general constraint) to learn constraints on violations of user’s preferences. Based on the user’s feedback on the recommended items, the recommender system utilizes constraints from this feedback to prevent undesired text generation.

| Scenario          | Model     | RL Algorithm   | RL Environment | Evaluation Strategy | Dataset                  |
|-------------------|-----------|----------------|----------------|---------------------|--------------------------|
| Interactive       |           |                |                |                     |                          |
|                    | UWE [64]  | Q-Learning     | Offline        | N/A                 |                          |
|                    | Multi-w.  | DP             | model-free     | Online              | ACM                      |
|                    | RL [65]   | DP             | model-based    | Offline             |                          |
|                    | ARA [66]  | DP             | model-based    | Online & Offline    | N/A                      |
|                    | DRCGR     | Value-function | DQN            | Offline             | N/A                      |
|                    | [67]      | DQN            | model-free     | Offline             |                          |
|                    | UDQN [68] | DQN            | model-free     | Offline             | ML100K, ML1IM, YMusic    |
|                    | FeedRec [69]| DQN         | model-free     | Offline             | JD                       |
|                    | PDL-Q     | Q-Learning     | model-based    | Offline             | Taobao, Retailroket      |
|                    | KGQI [70] | DQN            | model-based    | Offline             | Book-Crossing, ML20M     |
|                    | STL-EL [9] | REINFORCE      | REINFORCE      | ML100K, ML1IM, Steam|
|                    | RCR [72]  | Policy Search  | REINFORCE      | Yelp, UT-Zappos50K  |                          |
|                    | TPGR [73] | REINFORCE      | REINFORCE      | ML104M, Netflix      |                          |
|                    | FairRec [74]| DPG            | model-free     | Offline             | ML100K, Kiva             |
|                    | Attacks&Detection [13]| AC          | model-based    | Offline             | Amazon                   |
|                    | SDAC [75] | AC             | model-based    | Offline             | RecSys, Kaggle           |
|                    | AAMRL [76] | AC             | model-based    | Offline             | UT-Zappos50K             |
|                    | DRR [77]  | AC             | model-free     | Offline & Online    | ML100K, Yahoo! Music,    |
|                    |           |                |                |                     | ML100K, Jester           |
| Conversational    |           |                |                |                     |                          |
|                    | EMC, BTD, EHL [78]| Monte Carlo, | model-free     | Online & Offline    | TRAVEL, FC,              |
|                    | ISRA [79] | TD learning    |                |                     | CAMERA, CAR              |
|                    | EGE [80]  | Value-function | DQN            | Offline             | N/A                      |
|                    | MenN2N2 [81]| Q-learning    | model-based    | Online              | Shoes, FashionIQ Dress   |
|                    | SCPR [82] | DQN            | model-free     | Offline             | Personalized Dialog      |
|                    | UNICORN [83]| DQN          | model-free     | Offline             | Yelp, LastFM             |
|                    | CRM [84]  | Policy Search  | REINFORCE      | Yelp, LastFM        |                          |
|                    | EAR [85]  | Policy Search  | REINFORCE      | Yelp, LastFM        |                          |
|                    | CRSA [86] | Actor-Critic   | A3C            | model-free          | DSTC2, CamRes7076,      |
|                    | RelmCo [87]| AC             | model-free     | Offline             | MultiWOZ 2.1             |
|                    | SOQ [62]  | AC             | model-free     | Offline             | OpenDialKG, REDIAL       |
|                    | DEERS [88]| Q-Learning     | model-free     | Offline & Online    | RC15, RetailRocket       |
|                    | NRRS [89] | DQN            | model-based    | Online & Offline    | JD                       |
|                    | NLR [90]  | Value-function | Monte Carlo tree | model-based | Miliorsong               |
|                    | Kerc [91] | Policy Search  | REINFORCE      | Online & Offline    | Amazon                   |
|                    | HRL-NAS [41]| FG           | model-free     | Online & Offline    | XuantongX                |
|                    | DARL [92] | REINFORCE      | REINFORCE      | N/A                 |                          |
|                    | SAC [62]  | AC             | model-free     | Offline & Online    | RC15, RetailRocket       |
|                    | DeepChain [93]| AC           | model-based    | Online & Offline    | JD                       |
|                    | SAR [94]  | AC             | model-free     | Offline             | Steam, Electronics,      |
|                    |           |                |                |                     | ML104M, Kindle           |
| Sequential        |           |                |                |                     |                          |
|                    | MT Learning [95]| REINFORCE   | model-free     | Offline             | Amazon, Yelp              |
|                    | RL-Explanation [12]| REINFORCE | model-free     | Offline             | Amazon                   |
|                    | MKLNRN [96]| REINFORCE     | model-free     | Offline             | Amazon                   |
|                    | SAKG+SAFL [97]| REINFORCE   | model-free     | Offline             | Amazon                   |
|                    | PGR [26]  | REINFORCE      | REINFORCE      | Amazon              |                          |
|                    | AD-AC [28]| Actor-Critic   | AC             | model-free          | Amazon                   |
|                    | AnchorKG [98]| AC           | model-free     | Offline             | MIND, Bing News          |
Moreover, most existing RL methods fail to alleviate the issue of large discrete action space in interactive recommender systems since there are a large number of items to be recommended. To solve this problem, Chen et al. [73] propose a Tree-structured Policy Gradient Recommendation (TPGR) framework to achieve high effectiveness and efficiency for large-scale interactive recommendations. To maximize long-run cumulative rewards, the REINFORCE algorithm is utilized to learn the strategy for making recommendation decisions.

3) Actor-Critic Algorithms: Actor-critic algorithms have also been adopted extensively for interactive recommender systems in recent studies [77]. For instance, an RL-based framework (i.e., FairRec) [74] is proposed to dynamically achieve a fairness-accuracy balance, in which the fairness status of the system and user’s preferences combine to form the state representation. The FairRec framework contains two parts: an actor network and a critic network. The actor network generates the recommendation according to the fairness-aware state representation. The critic network is trained from the critic network and updated by the DPG algorithm. Then, the critic network estimates the value of the actor network outputs. The critic network is updated via TD learning.

Cao et al. [13] address adversarial attacks in RL-based interactive recommender systems. They propose a general framework that consists of two models. In the adversarial attack model, the agent crafts adversarial examples following the actor-critic or REINFORCE algorithm. In the encoder-classification detection model, the agent detects potential attacks based on adversarial examples, employing a classifier designed by attention networks. Following [26], the authors demonstrate the effectiveness of the proposed framework by judging whether the recommender system is attacked.

Online interactions in RL for interactive recommendation may hurt user experiences. An alternative is to adopt logged feedback to perform offline learning. However, it usually suffers from some challenges, such as unknown logging policies and extrapolation errors. To deal with these challenges, Xiao and Wang [75] propose a stochastic actor-critic method based on a probabilistic formulation and present some regularization approaches to reduce the extrapolation error.

In interactive recommender systems, utilizing multimodal data can enrich user feedback. To this end, Yu et al. [76] propose a vision-language recommendation approach that enables effective interactions with the user by providing natural language feedback. In addition, an attribute augmented RL is introduced to model explicit multimodal matching. More specifically, the multimodal data $A_t$ (i.e., the action of recommending items at time $t$) and $x_t$ (i.e., natural language feedback from users at time $t$) are leveraged in the proposed approach. Then, a recommendation tracker, consisting of a feature extractor and a multimodal history critic, is designed to enhance the grounding of natural language to visual items. The recommendation tracker may track the user’s preferences based on a history of multimodal matching rewards. The policy is updated via the actor-critic algorithm for recommending the items with desired attributes to the user.

B. Conversational Recommendation

Contrary to interactive recommender systems where users receive information passively (i.e., the recommendation system is dominant), conversational recommender systems interact with users actively. They explicitly acquire users’ active feedback and make recommendations that users really like. To achieve this objective, different from interactive recommender systems that recommend items from each interaction, conversational recommender systems [103] usually recommend items after communicating with users by real-time multturn interactive conversations based on natural language understanding and generation. In this case, there is a critical issue of tradeoffs between exploration and exploitation for conversation and recommendation strategies [102]. Conversational recommender systems explore the items unseen by a user to capture the user’s preferences through multturn interactions. However, compared to exploiting the related items that have already been captured, exploring the items that may be unrelated will harm the user experience. RL provides potential solutions to address this challenge. As shown in Fig. 3, the policy network stimulates the reward centers in conversational recommender systems, integrating exploration with exploitation in multturn interactions.

1) Value-Function Approaches: At each conversation turn of conversational recommendation, policy learning usually aims to decide what to ask, which to recommend, and when to ask or recommend. Deng et al. [83] introduce the UNICORN (i.e., UNified COnsertional RecommeNd) model that employs an adaptive DQN method to cast these decision-making processes as a unified policy learning task. To adapt the conversational strategy, Mahmood and Ricci [79] adopt the DP method to yield better policies that assist user behaviors more efficiently. Besides, Tsumita and Takagi [81] propose an RL-based dialog strategy that utilizes recommendation results based on the user’s utterances, whose intention is estimated by a long short-term memory (LSTM) network.

It is interesting that conversational recommender systems can use incremental critiquing as a powerful type of user feedback to retrieve the items in line with the user’s soft preferences at each turn. From this point of view, it needs a related quality measure for recommendation efficiency. To achieve this objective, a novel approach is proposed to improve the quality measure by combining a compatibility score with a similarity score [78]. To evaluate the compatibility of user critiques, exponential reward functions are presented by Monte Carlo and TD methods based on user specialization. Moreover, a global weighting of user’s preferences is proposed to enhance the critiquing quality, which brings about faster convergence of the similarity. By using the combination of these two scores, the conversational recommender system improves its robustness against noisy user data.

However, it is difficult to capture users’ preferences over time since the recommender system usually only obtains partial observations of the users’ preferences. In this case, we can formalize the observed interactions as a partially observable MDP (POMDP). Afterward, the estimator-generator-evaluator (EGE) [80] trains the POMDP with Q-learning to track users’
preferences and generates the next recommendations at each iteration.

The aforementioned conversational recommender systems usually utilize user feedback in implicit ways. Instead, to make full use of user-preferred attributes on items in an explicit way, a simple conversational path reasoning (SCPR) framework [82] is proposed to conduct interactive path reasoning for conversational recommendation on a graph. The SCPR obtains user-preferred attributes more easily, by pruning off irrelevant candidate attributes following user feedback based on a policy network. The policy network inputs the state vector and outputs the action value $Q(s, a)$, referring to the estimated reward for asking action $a_{ask}$ or recommending action $a_{rec}$. The policy is optimized by the standard DQN to achieve its convergence. Different from EAR [85], SCPR uses the adjacent attribute constraint on the graph to reduce the search space of attributes such that the decision-making efficiency can be improved.

2) Policy Search Methods: To model the user’s current intention and long-term preferences for personalized recommendations, Sun and Zhang [84] utilize deep learning technologies to train a conversational recommender system. Specifically, a deep belief tracker extracts a user intention by analyzing the user’s current utterance. Meanwhile, a deep policy network guides the dialog management by the user’s current utterance and long-term preferences learned by a recommender module. The framework of the proposed conversational recommender model (CRM) is composed of three modules: a natural language understanding with the belief tracker, dialog management based on the policy network, and a recommender designed based on a factorization machine [104]. More specifically, at first, the belief tracker converts the utterance (i.e., the input vector $z_t$) into a learned vector representation (i.e., $S_t$) by the LSTM network. Afterward, to maximize the long-term return, they use the deep policy network to select a reasonable action from the dialog state at each turn. The REINFORCE algorithm is adopted to optimize the policy parameter. Finally, the factorization machine is utilized to train the recommendation module, which generates a list of personalized items for the corresponding user.

Distinct from previous studies that ignore how to adapt the recommended items for user feedback, Lei et al. [85] propose a unified framework called estimation action reflection (EAR), in which a conversation component (CC) intensively interacts with a recommender component (RC) in the three-stage process. Specifically, the framework starts from the estimation stage, and the RC guides the action of CC by ranking candidate items. Afterward, at the action stage, the CC decides which questions to ask in terms of item features and makes a recommendation. The conversation action is performed by a policy network, which is optimized via the REINFORCE algorithm. When a user rejects the recommended items that are made at the action stage, the framework moves to the reflection stage for adjusting its estimations. Extensive experiments demonstrate the proposed EAR outperforms CRM [84] according to not only fewer conversation turns but also better recommendations, mainly because the candidate items adapt to user feedback in the interaction between the RC and CC.

3) Actor-Critic Algorithms: To address the training issue of task-oriented dialog systems, a conversational recommender system with adversarial learning (CRSAL) [86] is proposed to fully enable two-way communications. The process of CRSAL is divided into three stages. In the information extraction stage, a dialog state tracker first infers the current dialog belief state $b_t$ from the user’s utterances. Afterward, a neural intent encoder extracts and encodes the user’s utterance intention $z_t$. Finally, a neural recommender network derives recommendations from item features and dialog states. In the conversational response generation stage, a neural policy agent (i.e., the actor) generates human-like action in the current dialog state, and a natural language generator updates conversational responses from the critic network by the selected action. In the RL stage, the decision procedure of dialog actions can be formulated as a POMDP. The agent selects the best action in each conversation round under the long-term policy. To this end, an adversarial learning mechanism based on the A3C algorithm is developed to fine-tune the actions generated by the agent, which employs the discriminator to train the optimal parameters of the proposed model.

In conversational recommender systems, valuable information from user’s utterances is often conducive to retrieval performance. For example, Montazeralghaem and Allan [87] propose an RL-based model to extract relevant information from the context of the conversation. The model introduces
two actors: a selector-actor finds the most relevant words for the target of the conversation, and an arrangement-actor returns the related order of words based on the user’s utterances. Both actors are trained by the actor-critic algorithm.

C. Sequential Recommendation

Unlike interactive recommendation methods that generate recommendations based on the user’s feedback via constant interactions, sequential recommender systems predict the user’s future preference and recommend the next item given a sequence of historical interactions. Let $i_1^n = i_1 \rightarrow i_2 \rightarrow \cdots \rightarrow i_n$ denote the user–item interaction sequence, where $n$ is the sequence length. The sequential recommender system aims to recommend the next item $i$ that is not in the historical interactions. Some attempts have revealed that RL algorithms deal well with sequential recommendation problems since such problems can be naturally formulated as an MDP to predict the user’s long-term preferences. In this case, the recommender agent easily performs a sequence of ranking, which usually learns the optimal policy from the logged data with off-policy methods.

1) Value-Function Approaches: In sequential recommendation, it is essential to fuse long-term user engagement and user–item interactions (e.g., clicks and purchases) into the recommendation model training. Learning the recommendation policy from logged implicit feedback by RL is a promising direction. However, there exist challenges in implementation due to a lack of negative feedback and the pure off-policy setting. Xin et al. [62] present a novel self-supervised RL (SRL) method to address the challenges. More precisely, the next item recommendation problem is modeled as an MDP, where the recommender agent sequentially recommends items to related users to maximize the cumulative reward. To optimize the recommendation model, they define a flexible reward function that contains purchase interactions, long-term user engagement, and item novelty. Based on this method, the authors develop a self-supervised Q-learning model to train two layers with the logged implicit feedback. Similarly, Moling et al. [90] leverage R-Learning [105] to develop a music recommender system named RLradio. It exploits both explicitly revealed channel preferences and user’s implicit feedback, i.e., the user actually listens to a music track that is played in the recommended channel. Hong et al. [89] also leverage the wireless sensing and RL algorithm to improve the user experience in music recommender systems, in which user’s preferences are explored by the Monte Carlo tree search method.

Moreover, in [88], a novel DQN is built for the proposed framework, where the gated recurrent unit (GRU) captures the user’s sequential behaviors as positive state $s_+$, and the negative state $s_-$ is obtained by a similar way. The positive and negative signals are fed into the input layer separately, which assists the new DQN to distinguish contributions of the positive and negative feedbacks in recommendations.

2) Policy Search Methods: It is a nontrivial problem to capture the user’s long-term preferences in sequential recommender systems since the user–item interactions may be sparse or limited. Thus, it is unreliable for RL algorithms to learn user interests by using random exploration strategies. To overcome these issues, Wang et al. [91] propose a Knowledge-guided RL (KERL) framework that adopts an RL model to make recommendations over KG. In the MDP modeled by KERL, the environment contains the information of interaction data and KG, which are useful for sequential recommendation. In this case, the agent selects an action $a$ in state $S_i$ for recommending an item $i_{t+1}$ to a user $u$. During each recommendation process, the agent obtains an intermediate reward, which is defined by integrating sequence- and knowledge-level rewards.

Moreover, Zhang et al. [41] analyze users’ sequential learning behaviors and point out that the attention-based recommender systems perform poorly when the users enroll in diverse historical courses because the effects of the contributing items are diluted by many different items. To remove noisy items and recommend the most relevant items the next time, a profile reviser with two-level MDPs is designed. In this profile reviser, a high-level task decides whether to revise the user profile, and a low-level task decides which item should be removed. The agent in the proposed hierarchical RL (HRL) framework performs hierarchical tasks under the revising policy, which is updated by the REINFORCE algorithm. In addition, for capturing users’ dynamic preferences in sequential learning behaviors, Lin et al. [92] propose a dynamic attention mechanism, which combines with the HRL-based profile reviser to distinguish the effects of contributing courses in each interaction. As a result, the proposed model achieves more accurate recommendations than HRL [41].

3) Actor-Critic Algorithms: As mentioned before, Xin et al. [62] propose a novel self-SRL method to learn the recommendation policy from logged implicit feedback in the sequential recommendation. To optimize the recommendation model, a self-supervised actor-critic (SAC) model treats the self-supervised layer as an actor to perform ranking and takes the RL layer as a critic to estimate the state-action value.

However, both SQN and SAC [62] utilize a fixed length of interaction sequences as input to train the models, which affects the recommendation accuracy, because users usually have various sequential patterns. To address this issue, Antaris and Rafailidis [94] propose a sequence adaptation model via DRL (SAR) to adjust the length of interaction sequences. In particular, the RL agent performs the selection of a sequence length (i.e., action) in a personalized manner in the actor network. Finally, a joint loss function is optimized to align the cumulative rewards of the critic network with the recommendation accuracy.

D. Explainable Recommendation

The objective of explainable recommendation is to solve the problem of interpretability in recommender systems. The explainable recommender systems not only provide high-quality recommendations but also generate relevant explanations for recommendation results [106]. In particular, the visual explanation seems more intelligent when a recommender system conducts path reasoning over KG because KG contains rich relationships between users and items for intuitive explanations. Fig. 4 illustrates the common framework of RL with KG for the explainable recommendation. The KG is
cast as a part of the MDP environment, and the agent performs path reasoning for recommendations [26], [28].

In the survey on explainable recommendation [107], the explainable recommendation methods are divided into five classes: explanation with relevant items or users, feature-based explanation, social explanation, textual sentence explanation, and visual explanation. The emphasis of this section is to survey RL applied to the explainable recommendation, where the approaches focus on textual sentence explanation and visual explanation.

1) Policy Search Methods: In terms of textual sentence explanation, Lu et al. [95] propose a multitask learning framework that simultaneously makes rating predictions and generates explanations from users’ reviews. The rating prediction is performed by a context-aware matrix factorization model, which learns latent factor vectors of users and items. The recommendation explanation is employed by an adversarial sequence-to-sequence model, in which GRU generates personalized reviews from observed review documents, and adversarial training for the review generation is optimized by the REINFORCE algorithm.

To control the explanation quality flexibly, Wang et al. [12] design an RL framework to generate sentence explanations. In the proposed framework, there are a couple of agents instantiated with attention-based neural networks. The task of one agent is to generate explanations, while the other agent is responsible for predicting the recommendation ratings based on the explanations. The environment mainly consists of users, items, and prior knowledge. In this work, the recommendation model is treated as a black box. The agents extract the interpretable components from the environment to generate effective explanations.

Furthermore, to leverage the image information of entities, rather than focusing on rich semantic relationships in a heterogeneous KG, Tao et al. [96] present a multimodal knowledge-aware RL network (MKRLN), where the representation of recommended path consists of both structural and visual information of entities. The recommender agent starts from a user and searches for suitable items along hierarchical attention paths over the multimodal KG. These attention paths can improve the recommendation accuracy and explicitly explain the reason for recommendations.

The real-world KG is generally enormous, with a focus on finding more reasonable paths in the graph. From this aspect, Xian et al. [26] propose a reinforcement KG reasoning approach called policy-guided path reasoning (PGPR), which performs recommendations and explanations by providing actual paths with the REINFORCE algorithm over the KG. Specifically, the recommendation problem is treated as a deterministic MDP over the KG. In the training stage, following a soft reward and user-conditional action pruning strategy, the agent starts from a given user and learns to reach the correct items of interest. In the inference stage, the agent samples diverse reasoning paths for recommendation by a policy-guided search algorithm and generates genuine explanations to answer why the items are recommended to the user.

Distinct from previous explainable recommendation methods that use KGs, Park et al. [97] focus on sentiment on relations in the KG. To obtain more convincing explanations with sentiment analysis, a sentiment-aware KG (SAKG) is constructed by analyzing users’ reviews and ratings on items. Moreover, a sentiment-aware policy learning (SAPL) method is introduced to make recommendations and guide the reasoning over the SAKG. Experimental results demonstrate that the proposed framework outperforms state-of-the-art baselines (e.g., PGPR [26]) in terms of both accuracy and explainability.

2) Actor-Critic Algorithms: Differing from PGPR [26], the ADversarial Actor-Critic (ADAC) model [28] is proposed to identify interpretable reasoning paths. By learning the path-finding policy, the actor obtains its search states over the KG and potential actions. The actor obtains the reward $R_{e,t}$ if the path-finding policy from the current state fits the observed interactions. To integrate the expert path demonstrations, they design an adversarial imitation learning module based on two discriminators (i.e., metapath and path discriminators), which are trained to distinguish the expert paths from the paths generated by the actor. When its paths are similar to the expert paths in the metapath or path discriminator, the actor obtains the reward $(R_{m,t}, R_{p,t})$ from the imitation learning module. The critic merges these three rewards (i.e., $R_{e,t}$, $R_{m,t}$, and $R_{p,t}$) to estimate each action value accurately. Later, the actor is trained with an unbiased estimate of the reward gradient through the learned action values. Finally, the major modules of ADAC are jointly optimized to find the demonstration-guided path, which brings accurate recommendations.

Although existing works (e.g., PGPR [26] and ADAC [28]) combine KG and RL to enhance recommendation reasoning,

---

**Fig. 4.** RL algorithms with KG for explainable recommendation [26], [28]. The models aim to learn a path-finding policy over KG. The learned policy is adopted for the interpretable reasoning paths that make accurate recommendations to the given user.
they are not suitable for news recommendation, where a news article usually contains multiple entities. To address this challenge, a recommendation reasoning paradigm, named AnchorKG, is proposed to employ anchor KG path to make news recommendations [98]. The AnchorKG model consists of two parts. An anchor graph generator captures the latent knowledge information of the article, which leverages k-hop neighbors of article entities to learn high-quality reasoning paths. On the other hand, an AC-based framework is developed to train the anchor graph generator. Finally, the model performs a multitask learning process to optimize both the anchor graph generator and the recommendation task jointly.

E. Discussion on RL Aspect

As mentioned above, the comprehensive survey of RL-based recommender systems follows three classes of RL algorithms: value function, policy search, and actor-critic. From the RL perspective, we make a comparison of these three types of methods when they are applied in the recommender systems.

1) **Value-function approaches** depend heavily on the samples to learn the optimal policy. Thus, these approaches are suitable for small discrete action spaces and are often applied in small-scale recommender systems, such as traditional interactive recommendation and conversational recommendation. However, real-world recommender systems usually contain large action spaces, which leads to the slow convergence of RL when they utilize value-function approaches to make recommendations.

2) **Policy search methods** directly optimize the policy without relying on the value function. They are only suitable for continuous action spaces due to the policy gradient, which results in high variances. However, benefiting from fast convergence properties, policy search methods are competent in large-scale recommender systems, such as sequential recommendation and explainable recommendation, especially KG-based recommendation.

3) **Actor-critic algorithms** incorporate the advantages of value-function approaches and policy search methods. Nevertheless, actor-critic algorithms usually cause potential information loss since they map discrete action spaces into continuous action spaces by the policy network [47], which ensures that the policy is differentiable with respect to its parameters. Hence, actor-critic algorithms are rarely applied in recommender systems that focus on convincing recommendations, e.g., conversational recommendations and explainable recommendations.

RL-based recommendation models can also be divided into model-based and model-free algorithms. As shown in Table II, a few recommendation models adopt model-based algorithms, while most existing recommendation models utilize model-free algorithms.

1) **Model-based algorithms** (e.g., DP and heuristic search) require a model to represent the environment of the recommender system; thus, the agent relies on planning and guarantees sample efficiency. Nevertheless, such algorithms often result in biased estimations since the environment of recommender systems dynamically changes. Moreover, the transition probability is deterministic in model-based algorithms. Therefore, these algorithms are not applicable to real-world recommender systems.

2) **Model-free algorithms** (e.g., TD, DQN, and REINFORCE) generally achieve better recommendation performances because the agent mainly relies on learning from previous experiences. The drawback of such algorithms is sample inefficiency, i.e., they require a large number of samples to ensure the convergence of the algorithms.

V. CHALLENGES IN REINFORCEMENT LEARNING-BASED RECOMMENDATION APPROACHES

As noted above, RL aims to maximize long-run cumulative rewards by learning the optimal policy from the interactions between the agent and its environment. Thus, it relies not only on the environment but also on prior knowledge. In recommendation applications, RL approaches often suffer from various challenges. To this end, many researchers put forward relevant solutions to different issues. In the following, we summarize relevant studies from five aspects: environment construction, prior knowledge, reward function definition, learning bias, and task structuring.

A. Environment Construction

In RL, the agent observes states from the environment, then conducts relevant actions under the policy, and receives a reward from the environment. Applied to recommender systems, policy training in the environment is often confronted with many unpredictable situations due to the need for exploration. In this case, environment construction is critical to learn the optimal recommendation policy.

1) **State Representation:** The state representation plays an important role in RL to capture the dynamic information during the interactions between users and items since the environment state is a key component of MDP. However, most current RL methods focus on policy learning to optimize recommendation performance. To effectively model the state representation, Liu et al. [77] propose a DRL-based recommendation framework termed DRR, where four state representation schemes are designed to learn the recommendation policy (the actor network) and the value function (the critic network). Specifically, the DRR-p scheme employs the elementwise product operator to learn the pairwise local dependency between items. The DRR-u scheme incorporates the pairwise interactions between users and items into the DRR-p scheme. The DRR-ave scheme concatenates the user embedding, the average pooling result of items, and the user–item interactions into a whole vector to describe the state representation. In the DRR-att scheme, a weighted average pooling is conducted by attention networks. The actor network conducts a ranking action \( a = \pi_r(s) \) according to the state representation \( s \) and generates a corresponding \( Q \)-value by the action-value function \( Q_\pi(s, a) \). The critic network leverages
a DQN parameterized as \( Q_w(s, a) \) to approximate \( Q^*(s, a) \). Based on DPG [60], the actor network can be updated by the policy gradient via

\[
\nabla_q J(\pi_\theta) \approx \frac{1}{N} \sum_i \nabla_a Q_w(s, a)_{|s=S_i, a=\pi_\theta(S_i)} \nabla_\theta \pi_\theta(s)_{|s=S_i} \tag{10}
\]

where \( J(\pi_\theta) \) denotes the expectation of all possible \( Q \)-values following the policy \( \pi_\theta \) and \( N \) is the batch size.

Moreover, Shang et al. [108] introduce the DEconfounded Multiagent Environment Reconstruction (DEMER) framework, which assumes that the environment reconstruction from the historical data is powerful in RL-based recommender systems. DEMER randomly samples one trajectory from the observed data and then forms its first state as the initial observation. It finally generates a policy-generated trajectory by the TRPO algorithm, considering the confounder embedded policy as a role of hidden confounders in the environment.

It may be practical to formulate a simulation environment of recommendation as an RL gym. To achieve this objective, PyRecGym [109] is designed for RL-based recommender systems to support standard test datasets and common input types. In the PyRecGym, the states of the environment refer to user profiles, items, and interactive features with contextual information. The RL agent interacts with a gym engine in the current state and obtains related feedback from the gym engine according to the reward. PyRecGym contains three major functions: the initialization function that initializes the initial state of the environment for the user, the step function that reacts to the agent’s action and returns the next state.

2) Knowledge Graph Leverage: Recent advances in KG have attracted increasing attention in RL-based recommender systems. The KG-enhanced interaction fashion enriches user–item relations by the structural knowledge to describe the MDP environment. Besides, the multihop paths in a KG contribute to the reasoning process for explainable recommendations.

The KERL framework [91] incorporates KG into RL-based recommender systems, with the aim of predicting future user’s preferences and addressing the sparsity. To learn the user’s preferences from sparse user feedback, Zhou et al. [71] propose a KG-enhanced Q-learning model for interactive recommender systems. Instead of learning the policy by trial-and-error search, the model utilizes the knowledge of correlations among items learned from KG to enrich the state representation of both items and users. Thus, it spreads the user’s preferences among the correlated items over KG.

Xian et al. [26] made the first attempt to leverage KG and RL for the explainable recommendation. They develop a unified framework (i.e., PGPR) to provide actual recommendation paths guided by the policy network in a KG. To further improve the convergence of the PGPR approach, Zhao et al. [28] propose a demonstration-based KG reasoning framework named ADAC, in which imperfect path demonstrations are extracted to guide path-finding. The ADAC model aims to identify interpretable reasoning paths for accurate recommendations.

Another interesting work is the negative sampling by the KG policy network [110]. It is incorporated into the recommendation framework to explore informative negatives over KG. The recommender (i.e., matrix factorization) and the sampler (i.e., KG policy network) are jointly trained by iteration optimization. The recommender parameters are updated by the stochastic gradient descent (SGD) method, and the sampler parameters are updated via the REINFORCE algorithm.

3) Negative Sampling: Most existing studies extract negative sampling from the unobserved data to assist the training of the recommendation model. However, they often fail to yield high-quality negative samples to reflect the user’s needs, which provides an important clue on the environment construction for RL. To discover informative negative feedback from the missing data, Wang et al. [110] propose a KG policy network for knowledge-aware negative sampling, which employs an RL agent to search high-quality negative instances with multihop exploring paths over the KG. Similarly, to learn the sampling strategy from the missing negative signals, the supervised negative Q-learning method [111] trains the RL algorithm with a supervised sequential learning method to sample negative items.

Moreover, the user exposure data, which records the history of interactions based on implicit feedback, are also beneficial to train negative samples. From this point of view, Ding et al. [112] introduce a recommender-sampler framework, where the sampler samples candidate negative items as the output, and the recommender is optimized by the SGD method to learn the pairwise ranking relation between a ground-truth item and a generated negative item. After obtaining the multiple rewards, the sampler is optimized by the REINFORCE algorithm to generate both real and hard negative items. Moreover, Zhao et al. [113] use a CF-based pretraining method to sample negative items for RL-based recommender systems.

4) Social Relation: Traditional recommender systems often suffer from two main challenges, i.e., cold start and data sparsity. A promising way for alleviating these issues is to utilize users’ social relationships to model users’ preferences efficiently. Thus, the agent can sample users’ social relationships and deliver them to the environment.

Applied to RL-based recommendation, Lei et al. [114] integrate social networks into the estimation of action-values. More specifically, a social matrix factorization method is proposed to describe the high-level state/action representations. To learn more relevant hidden representations from personal preferences and social influence, an enhanced SADQN model is developed to utilize additional neural layers to summarize potential features from the hidden representations and then predict the final action value with the summarized features. Moreover, Lu et al. [115] propose a Social Recommendation framework based on RL (SRRL) to identify reliable social relationships for the target user. In particular, SRRL adaptively samples the social friends to improve the recommendation quality with user feedback since the reward is always real time.

B. Prior Knowledge

The combination of imitation learning with RL can be named apprenticeship learning [116], which utilizes
demonstrations to initialize the RL. In particular, the inverse RL (IRL) algorithm [117] often assumes that the expert acts to maximize the reward and derives the optimal policy from the learned reward function. In RL-based recommender systems, the reward signals are usually unknown since users scarcely offer feedback. On the other hand, IRL algorithms are good at reconstructing the reward function for the optimal policy from users’ observed trajectories.

To improve the novelty of the next-point-of-interests (POI) recommendation that boosts users’ interests, David and Ricci [118] adopt a novel IRL algorithm termed maximum log-likelihood (MLIRL) [119] to model the unknown user’s preferences based on the state features. This method exploits the knowledge of the user’s preferences to estimate an initial reward function that justifies the observed trajectories and optimizes the user’s behavior by a gradient ascent method.

To mine users’ interests in online communities efficiently, Liang [120] designs a reinforced user profiling for recommendations by employing both data-specific and expert knowledge, where the agent employs random search to find data-specific paths in the environment. These metaphats contain expert knowledge and semantic meanings for searching useful nodes.

Distinct from the top-K recommendation with the target of ranking optimization, the novel exact-K recommendation focuses on combinatorial optimization. It may be more suitable to address the recommendation problems in application scenarios. Toward this end, Gong et al. [121] design an encoder–decoder framework named graph attention networks (GAttNs). The proposed framework learns the joint distribution of the $K$ items and outputs an optimal card that contains these $K$ items. To train the GAttN efficiently, an RL from demonstrations method integrates behavior cloning [122] into RL.

C. Reward Function Definition

The agent behavior in RL indirectly relies on the reward function, while, in practice, it is a key challenge to define a promising reward function. Beyond the need to learn the policy by intermediate rewards, the reward definition based on the specific demands in different recommendation scenarios is necessary. For instance, Xin et al. [62] define a flexible reward function that contains purchase interactions, long-term user engagement, and item novelty for the relevant recommendation tasks. Gao et al. [123] define discrete rewards for different user behaviors. Moreover, a one-step reward in [124] is distributed to the online recommendations during visitor interactions.

A creative work is reported in [125], which leverages GAN to model the dynamics of user behaviors and learn relevant reward functions, which depends not only on the user’s historical behaviors but also on the selected item. The learned reward allows the agent to recommend items in a principled way, instead of relying on the manual design. Based on the simulation environment using the user behavior model, a cascading DQN is proposed to learn the combinatorial recommendation policy. Similarly, Chen et al. [126] introduce a generative IRL approach to avoid defining a reward function manually. The recommendation problem is regarded as automatic policy learning. Thus, this approach first generates a policy based on the user’s preference. Later, it uses a discriminative actor-critic network to evaluate the learned policy based on the reward function defined by

$$R(s, a) = \log D(s, a) - \log(\max(\epsilon, 1 - \log D(s, a))) + r$$

(11)

where $\log D(s, a)$ is the negative reward generated by the discriminator, $r \in [0, 1]$ represents the user’s feedback to prevent the agent from taking an extra step to update itself when the user clicks each recommendation, and $\epsilon$ is the maximum percentage of change that is updated at a time.

For commercial recommendation systems, online advertising is frequently inserted into personalized recommendations to maximize profit. To this aim, a value-aware recommendation model [127] based on RL is designed to optimize the economic value of candidate items to make recommendations. In the value-aware model, measured by the gross merchandise volume, the total reward is defined as the expected profit that is converted from all types of user actions (e.g., click and pay). Moreover, Zhao et al. [128] present an advertising strategy for DQN-based recommendation, where the advertising agent simultaneously maximizes the income of advertising and minimizes the negative influence of advertising on the user experience. Thus, the reward contains both the income of advertising and the influence of advertising on the user experience.

In practice, users have interest in exploring novel items. To this end, Zou et al. [129] propose a fast Monte Carlo tree search method for diversifying recommendations, where the reward function is designed with the diversity and accuracy gain derived from recommending items at corresponding positions.

D. Learning Bias

RL algorithms can be classified into on-policy and off-policy methods [49]. On-policy methods often sweep through all states to learn the policy and result in expensive costs; hence, on-policy methods are not applicable to large-scale recommender systems. On the other hand, learning the recommendation policy from logged data (e.g., logged user feedback) is a more practical solution because it alleviates complex state space and high interaction cost with off-policy [130], [131]. Nevertheless, due to the difference between the target policy and the behavior policy, the off-policy methods usually result in data biases or policy biases.

1) Data Biases: There are lot of logged implicit feedback, such as user clicks and dwell time, available for learning users’ preferences. However, the learning methods tend to easily suffer from biases caused by only observing partial feedback on previous recommendations. To deal with such data biases, Chen et al. [132] propose a recommender system with the REINFORCE algorithm, where an off-policy correction approach is utilized to learn the recommendation policy from the logged implicit feedback and incorporates the learned model of the behavior policy to adjust the data biases.
Fully cooperative tasks in MARL-based recommender systems have the same goal (i.e., maximizing the same cumulative return) achieved by all the agents [93]. For instance, in the DEMER approach [108], the policy agent cooperates with the environment agent to learn the policy of a hidden confounder. Moreover, to capture the general preference of users and their temporary interests, Du et al. [138] introduce a Temporary Interest Aware Recommendation (TIARRec) model with MARL. Particularly, an auxiliary classifier agent can judge whether each interaction is atypical or not. The classifier agent and the recommender agent are jointly trained to maximize the cumulative return of the recommendations.

In contrast, fully competitive tasks typically adopt the minimax principle: each agent maximizes its own benefit under the assumption that the opponents keep endeavoring to minimize it. In a dynamic collaboration recommendation method for recommending collaborators to scholars [65], competition should be characterized as a latent factor since scholars hope to compete for better candidates. To this end, the proposed method uses competitive MARL to model scholarly competition, i.e., multiple agents (authors) compete with each other by learning the optimized recommendation trajectories. Besides that, an improved market-based recommendation model [139] urges all agents to classify their recommendations into various internal quality levels and employs the Boltzmann exploration strategy to conduct these tasks by the recommender agents.

Both cooperative and competitive tasks also exist in MARL-based recommender systems [140]. For example, Zhang et al. [141] aim to recommend public accessible charging stations intelligently. Each charging station is regarded as an individual agent. Subsequently, a centralized attentive critic module is developed to stimulate multiple agents to learn cooperative policies, while a delayed access strategy is proposed to exploit future charging competition information during model training. Besides, to address the suboptimal policy of ranking due to the competition between independent recommender modules, He et al. [38] promote the cooperation of different modules by generating signals for these modules. Each agent acts on the basis of its signal without mutual communication. For the \( i \)th agent with the signal vector \( \phi^i \), given a \( Q \) value function \( Q^i_j(S_i, A_i) \) and a shared signal network \( \Phi^i(S_i) \), the objective function can be defined as follows:

\[
J_{\phi}(\xi) = \frac{1}{N} \sum_i \left[ \mathbb{E}_{S_i, A_i \sim D, \phi^i \sim \Phi} \left[ -Q^i_j(S_i, A_i; \phi^i_j) + \alpha \log \Phi^i(\phi^i_j|S_i) \right] \right]
\]  

where \( \xi \) and \( \theta \) are the network parameters, \( D \) denotes the distribution of samples, and \( N \) is the batch size. The objective function for each agent is optimized by the SAC algorithm.

2) Hierarchical RL: In HRL, an RL problem can be decomposed into a hierarchy of subproblems or subtasks, which reduces the computational complexity. There exist some studies for HRL, which solve the recommendation problems well. For example, using hierarchies of abstract machines (HAMs) [41], [142] formalizes the overall task of profile reviser as an MDP and decomposes the task MDP into two abstract subtasks. If the agent decides to revise the user profile
(i.e., a high-level task), it allows the high-level task to call a low-level task to remove noisy courses. To improve the recommendation adaptivity, a dynamic attention and HRL (DARL) framework [92] is developed to automatically track the changes in the user’s preferences in each interaction. These two methods adopt the REINFORCE algorithm to optimize both the high- and low-level policy functions.

HRL with the MaxQ approach [143] is a task-hierarchy that restricts subtasks to different subsets of states, actions, and policies of the task MDP without importing extra states. For recommender systems, a multigoal abstraction-based HRL [11] is designed to learn the user’s hierarchical interests. In addition, Xie et al. [144] propose a novel HRL model for the integrated recommendation (i.e., simultaneously recommending the heterogeneous items from different sources). In the proposed model, the task of integrated recommendation is divided into two subtasks (i.e., sequentially recommending channels and items). The low-level agent works as a channel selector, which provides personalized channel lists in terms of user’s preferences. On the other hand, the high-level agent is regarded as an item recommender, which recommends heterogeneous items based on channel constraints.

Another prevailing approach is the options framework (i.e., closed-loop policies for taking action over a period of time) [145], which generalizes the primitive actions to include a temporally extended navigation of actions. For example, Wang et al. [146] design a subgoal-conditioned hierarchical imitation learning (SHIL) framework for dynamic treatment recommendation. In the SHIL framework, the high-level policy sequentially selects a subgoal for each subtask. Based on each subgoal, the low-level policy produces the low-level action (i.e., effective medication) in the corresponding state for each subtask.

3) Supervised Reinforcement Learning: In practice, the recommendation problems can be more easily solved by utilizing a combination of RL and supervised learning rather than only using RL, as supervised learning and RL can handle corresponding tasks according to their advantages, respectively. For example, Xin et al. [62] propose a self-SRL model to learn the recommendation policy from users’ logged feedback in sequential recommender systems. The model has two output layers. One is the self-supervised layer trained with a cross-entropy loss function to perform ranking. The other is trained with RL based on a flexible reward function, which performs as a regularizer for the supervised layer. Studies in [147] and [148] focus on optimizing the user’s life-time value in personalized ad recommender systems. To achieve this goal, they propose an RL-based recommendation model, where mapping from features to actions is learned by a random forest algorithm.

SRL is often applied to satisfy the adaptability of recommendation strategies. For instance, to address the top-aware drawback (i.e., the performance on the top positions) that may reduce user experiences, a supervised DRL model [149] jointly utilizes the supervision signals and RL signal to learn the recommendation policy in a complementary way. Different from the top-aware recommender distillation framework [150] that utilizes DQN to reinforce the rank of recommendation lists, the supervised DRL model contains two styles of supervision signals. It adopts the cross-entropy loss for the classification-based supervision signal and employs pairwise ranking-based loss for the ranking-based supervision signal. The suitable supervision signals adaptively balance the long-term reward and immediate reward.

Moreover, Wang et al. [43] put forward SRL with RNN for dynamic treatment recommender systems. By combining the supervised learning signal (e.g., the indicator of doctor prescriptions) with the RL signal (e.g., maximizing the cumulative reward from survival rates), this approach learns the prescription policy to refrain from unacceptable risks and provides the optimal treatment. Huang et al. [9] also propose a novel SRL with RNN for long-term recommendation. More precisely, RNN is employed to adaptively evolve user states for simulating the sequential interactions between the recommender system and users.

To tackle the training compatibility among the components of embedding, state representation, and policy in RL-based recommender systems, Liu et al. [63] propose an End-to-end DRL-based Recommendation framework namely EDRR, in which a supervised learning signal is designed as a classifier of the user’s feedback on the recommendation results. Moreover, DQN and DDPG are employed to elaborate on how embedding component works in the proposed framework.

F. Summary

An overview of challenges in RL-based recommendation approaches is presented in Table III. Overall, a number of studies have attempted to address the challenges of applying RL in recommender systems. On the one hand, many studies focus on adapting RL algorithms to different recommendation scenarios. For instance, a promising reward function that is skillfully designed can meet the specific requirements of the recommender system. On the other hand, some studies leverage related techniques to solve the issues of RL-based recommender systems. For example, the KG not only enriches user-item relations by the structural knowledge to describe the MDP environment but also contributes to explainable recommendations by the multihop paths. Nonetheless, there are some other possible limitations of RL-based recommender systems, as well as challenges in complex application scenarios. For these limitations and challenges, we provide detailed discussions in Section VI.

VI. DISCUSSION

In this section, we first discuss the central issues in applying RL algorithms for recommender systems. Afterward, we analyze practical challenges and provide several potential insights into successful techniques for RL-based recommender systems.

A. Open Issues

In practice, RL algorithms are not applicable to real-time recommender systems due to their trial-and-error mechanism. A key step in the application of RL algorithms is to improve the recommendation effectiveness and realize an intelligent design. Hence, the following issues need to be solved for making much progress in this field.
1) Sampling Efficiency: Sampling plays a substantial role in RL, especially in DRL-based recommender systems. Importance sampling has empirically demonstrated its application feasibility in both on-policy and off-policy. Some works [110], [112] have highlighted the superiority of negative sampling in recommender systems. Nevertheless, it is necessary to focus on the improvement in sampling efficiency since the user feedback available to train recommendation models is scarce. We can leverage auxiliary tasks to improve sampling efficiency. For example, Chen et al. [151] develop a user response model to predict the user’s positive or negative responses toward the recommendation results. Thus, the state and action representations can be enhanced with these responses. Moreover, transfer learning may be qualified for the sampling task. For example, we can use the transfer of knowledge between temporal tasks [152] to perform RL on an extended state space and concretize the similarity between the source task and the target task by logical transferability. In addition, the transfer of experience samples [153] can estimate the relevance of source samples. Both transfer learning methods may be able to improve the sampling efficiency in RL-based recommender systems.

2) Reproducibility: In many application scenarios, including recommendation, it is often difficult to reproduce the results of RL algorithms because of various factors, such as instability of RL algorithms, lacking open-source codes & account of hyperparameters, and different simulation environments (e.g., experimental setup and implementation details), although most existing studies have performed empirical experiments on public datasets, as shown in Table II. Especially for DRL methods, nondeterminism in policy networks and intrinsic variance of these methods leads to a reproducibility crisis. Besides policy evaluation methods, there are several future directions for reproducibility investigation with statistical analysis. We can use significance testing according to related metrics or hyperparameters, such as batch size and reward scale [154]. For example, the reward scale needs to check its rationality and robustness in specific recommendation scenarios. The average returns should be evaluated to verify whether it is proportional to the relevant performance. Moreover, due to the sensitivity of RL algorithms changed in their environments, random seeds, and definition of the reward function, we should ensure reproducible results with fair comparisons. To achieve this objective, we need to run the same experiment trials for baseline algorithms and take each evaluation with the same preset epoch. Moreover, all experiments should adopt the same random seeds [154].

3) MDP Formulation: In principle, MDP formulation is essential to guarantee the performance of RL algorithms. However, the state and action spaces in recommender systems often suffer from the curse of dimensionality. Besides, the reward definition is sensitive to the external environment since users’ demands may change constantly, while interactions between users and items are random or uncertain. To address these issues, we can employ task-specific inductive biases to learn the representations of selection-specific action. As a result, the agent relies on better action structures to learn RL policies when the recommendation problem is formulated [155]. Another direction is to use causal graph and probabilistic graph models [156] to enable the MDP formulation, where the causal graph describes the causal-effect relations among user–item interactions, and the probabilistic graph reasons the recommendation paths. This joint process can improve the efficiency of agent search and tracks users’ interests over time. However, how to design MDPs for recommendation problems in a verifiable way remains an open issue.

4) Generalization: Model generalization ability is almost the pursuit of all applications. Limited by the shortcomings of different RL algorithms, it is difficult to develop a general framework to meet various specific requirements of recommendation. On the one hand, model-based algorithms are only applicable to the specified recommendation problem and fail to solve the problem that cannot be modeled. On the other hand, model-free algorithms are insufficient in dealing with different tasks in complex environments. Fortunately, meta-RL, such as meta-strategy [157] and meta-Q-learning [158],
first learns a large number of RL tasks to obtain enough prior knowledge and later can be quickly adapted to the new environment in the face of new tasks. In this case, it may enable the generalization of RL-based recommender systems. Besides transfer learning based on meta-RL, we can use multitask learning [159] to handle related tasks (e.g., construction of user profiles, recommendation, and causal reasoning) in parallel. These tasks complement each other to improve the generalization performance via shared representation of domain information, such as parameter-based sharing and joint feature-based sharing.

5) Autonomy: It seems easy to utilize RL for autonomous control [48]. However, in practice, it is difficult to achieve this objective in online recommendation scenarios since there are complex interactions between users and items, while it is not feasible to capture users’ dynamic intentions by existing RL algorithms. To this end, we can achieve the feedback control of recommender systems by combining RL with LSTM or GRU, which enables the RL agent to be possessed powerful memory that preserves the sequences of state and action, as well as different kinds of reward. Later, the historical information is encoded and transmitted to the policy network, thereby improving the autonomous navigation ability of the RL agent. Another interesting research direction is to improve the autonomous learning ability of RL from the fields of biology, neuroscience, and cybernetics. Thus, it is straightforward to use the learned knowledge to make better recommendations.

B. Practical Challenges

The existing recommendation models based on RL have demonstrated superior recommendation performance. Nevertheless, there are many challenges and opportunities in this field. We summarize five potential directions that deserve more research efforts from the practical aspect.

1) Computational Complexity: RL-based recommender systems often suffer from huge computations due to the exploration–exploitation tradeoff and the curse of dimensionality [160]. Apart from task structuring techniques, the IRL algorithm is an efficient solution to reduce the computation cost. Initializing RL with demonstrations can supervise the agent taking action correctly, particularly for some specific recommendation tasks. For example, apprenticeship learning via IRL algorithm [116] highlights the need for learning from an expert, which maximizes a reward based on a linear combination of known features. The hierarchical DQNs [161] alleviate the curse of dimensionality in large-scale recommender systems. Besides, a promising method of driving route recommendation based on RL is to employ behavior cloning [122], which makes expert trajectories available and quickens the learning speed. Another feasible scheme is to improve the efficiency of agent exploration. For example, we can adopt NoisyNet [162] that adds parametric noise to its network weights, which aids efficient exploration according to the stochasticity of the policy for the recommender agent.

2) Evaluation: Most existing RL-based recommender systems focus on the single goal of recommendation accuracy, without considering recommendation novelty and diversity [163] based on the user experience. Beyond the need for new evaluation measures for multiobjective goals of recommendation (e.g., popularity rate [164]), we should design standard metrics for other nonstandardized evaluation measures (e.g., diversity, novelty, explainability, and safety). In general, these kinds of evaluation measures can be considered to be combinatorial optimization problems and, naturally, maybe well achieved by multiobjective evolutionary algorithms [165]. Recently, some works have developed Pareto efficiency models for multiobjective recommendation [166], [167]. For example, in the personalized approximate Pareto-efficient recommender system [168], a Pareto-oriented RL module learns personalized objective weights on multiple objectives for the target user. Nevertheless, it remains a challenge to reconcile different evaluation measures for the recommendation since these evaluation measures are usually relevant and even conflict.

3) Biases: In recommender systems, item popularity often changes over time due to the user engagement and recommendation strategy [169]; thus, long-tailed items are rarely recommended to users. The selection bias may hurt user satisfaction [170]. Therefore, it is necessary to concentrate on fairness work. Toward this end, we can combine anthropology to analyze the differences in human behavior and cultural characteristics or explore the user’s intentions in user–recommender interactions. Besides, heterogeneous data of user behavior often exist in online recommendation platforms, whereas most recommendation approaches are trained with a single type of data. Due to the information asymmetry between the user behavior and training model, the recommender system suffers from learning bias. Undoubtedly, the previous feature extraction and representation learning are crucial to deal with such bias. In addition, MARL can be used to encourage different agents to execute multidimensional data simultaneously and share parameters for a unified recommendation goal.

4) Interpretability: Due to the complexity of RL, the post-hoc explanation may be easier to achieve than intrinsic interpretability [171]. In fact, the same is true in RL-based recommendation systems. How to explore intrinsic interpretable methods for RL to provide more convincing recommendations is a promising line. Another research direction toward explainable recommendation is to provide formal guidance of recommendation reasoning process, rather than being concerned with a multihop reasoning process [172]. Tai et al. [173] propose a user-centric path reasoning framework that adopts a multiview structure to combine sequence reasoning information with subsequent user decision-making. However, they only focus on the user’s demand and do not provide theoretical proof of the rationality of reasoning. We should avoid plausible explanations in the reasoning process. To this end, we can employ multitask learning to perform the recommendation reasoning process among multiple related tasks, e.g., representation of interaction, recommended path generation, and Bayesian inference for policy networks. These related tasks jointly provide credible recommendation explanations by sharing the representation of internal correlation and causality.

5) Safety and Privacy: System security and user privacy are important issues, which are ignored by most existing studies. For example, personal privacy is easy to be leaked when using RL and KG to perform the explainable recommendation because the relationships among users and items are exposed.
Differential privacy is widely applied to protect user privacy, and DRL can be employed to choose the privacy budget against inference attacks [174]. Besides, recent studies have found that deep neural networks (DNNs) are vulnerable to attacks, such as adversarial attacks and data poisoning. For example, in DNN-based recommender systems, users with fake profiles may be generated to promote selected items. To address this issue, a novel black-box attacking framework [175] adopts policy gradient networks to refine real user profiles from a source domain and copy them into the target domain. Notwithstanding, we need to make more efforts on the research topic. For example, the safe RL [176] can be utilized to guarantee the security of recommender systems (e.g., detecting error states and preventing abnormal agent actions). We may also leverage federated learning [177], [178] to achieve privacy-preserving data analysis for MARL-based recommender systems, where each agent is localized on the distributed device and updates a local model based on the user data stored in the corresponding user device.

VII. CONCLUSION

Recommender systems serve as a powerful technique to address the information overload issue on the Internet. There has been increasing interest in extending RL approaches for recommendations in recent years. RL-based recommendation methods autonomously learn the optimal recommendation policies from user–item interactions, and thus, they recommend better items to users compared with other recommendation methods. In this survey, we first conduct a comprehensive review of RL-based recommender systems using three major categories of RL (i.e., value function, policy search, and actor-critic) to cover four typical recommendation scenarios. We also restructure the general frameworks for some specific scenarios, such as interactive recommendation, conversational recommendation, and explainable recommendation based on KG. Furthermore, the challenges of applying RL in recommender systems are systematically analyzed, including environment construction, prior knowledge, the definition of the reward function, learning bias, and task structuring. To facilitate future progress in this field, we discuss the theoretical issues of RL, analyze the limitations of existing approaches, and finally put forward some possible future directions.

REFERENCES

[1] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, “On deep learning for trust-aware recommendations in social networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 5, pp. 1164–1177, May 2017.
[2] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” Knowl.-Based Syst., vol. 46, pp. 109–132, Jul. 2013.
[3] Z. Huang, X. Xu, H. Zhu, and M. Zhou, “An efficient group recommendation model with multitattension-based neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 11, pp. 4461–4474, Nov. 2020.
[4] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, Jun. 2005.
[5] Y. Shi, M. Larson, and A. Hanjalic, “Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges,” ACM Comput. Surveys, vol. 47, no. 1, pp. 1–45, May 2014.
[6] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” ACM Comput. Surveys, vol. 52, no. 1, pp. 1–38, Feb. 2019.
[7] W. Zhao et al., “Leveraging long and short-term information in content-aware movie recommendation via adversarial training,” IEEE Trans. Cybern., vol. 50, no. 11, pp. 4680–4693, Nov. 2020.
[8] F. Pan, Q. Cai, P. Tang, F. Zhuang, and Q. He, “Policy gradients for contextual recommendations,” in Proc. World Wide Web Conf., May 2019, pp. 1421–1431.
[9] L. Huang, M. Fu, F. Li, H. Qu, Y. Liu, and W. Chen, “A deep reinforcement learning based long-term recommender system,” Knowl.-Based Syst., vol. 213, Feb. 2021, Art. no. 106706.
[10] L. Ji, Q. Qin, B. Han, and H. Yang, “Reinforcement learning to optimize lifetime value in cold-start recommendation,” in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage., Oct. 2021, pp. 782–791.
[11] D. Zhao, L. Zhang, B. Zhang, L. Zheng, Y. Bao, and W. Yan, “MaHRL: Multi-goals abstraction based deep hierarchical reinforcement learning for recommendations,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2020, pp. 871–880.
[12] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, and X. Xie, “A reinforcement learning framework for explainable recommendation,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2018, pp. 587–596.
[13] Y. Cao, X. Chen, L. Yao, X. Wang, and W. E. Zhang, “Adversarial attacks and detection on reinforcement learning-based interactive recommender systems,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2020, pp. 1669–1672.
[14] E. O. Nefci and B. B. Averbeck, “Reinforcement learning in artificial and biological systems,” Nature Mach. Intell., vol. 1, no. 3, pp. 133–143, Mar. 2019.
[15] H. Li, D. Liu, and D. Wang, “Manifold regularized reinforcement learning,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 4, pp. 932–943, Apr. 2018.
[16] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26–38, Nov. 2017.
[17] G. Zheng et al., “DRN: A deep reinforcement learning framework for news recommendation,” in Proc. World Wide Web Conf. World Wide Web (WWW), 2018, pp. 167–176.
[18] D. Zha et al., “AutoShard: Automated embedding table sharing for recommender systems,” in Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining, Aug. 2022, pp. 4461–4471.
[19] M. Jaderberg et al., “Human-level performance in 3D multiplayer games with population-based reinforcement learning,” Science, vol. 364, no. 6443, pp. 859–865, May 2019.
[20] J. Koher, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” Artif. Intell., vol. 32, no. 11, pp. 1238–1274, Sep. 2013.
[21] L. Zou et al., “Neural interactive collaborative filtering,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2020, pp. 749–758.
[22] Q. Liu et al., “Exploiting cognitive structure for adaptive learning,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 627–635.
[23] S. Ji, Z. Wang, T. Li, and Y. Zheng, “Spatio-temporal feature fusion for dynamic taxi route recommendation via deep reinforcement learning,” Knowl.-Based Syst., vol. 205, Oct. 2020, Art. no. 106302.
[24] H. Lee, D. Huang, K. Min, and J. Choo, “Towards validating long-term user feedbacks in interactive recommendation systems,” in Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2022, pp. 2607–2611.
[25] K. Wang et al., “Reinforcement learning with a disentangled universal value function for item recommendation,” in Proc. AAAI, 2021, pp. 4427–4435.
[26] Y. Xian, Z. Fu, S. Muthukrishnan, G. de Melo, and Y. Zhang, “Reinforcement knowledge graph reasoning for explainable recommendation,” in Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2019, pp. 285–294.
[27] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” in Proc. ICLR, 2016, pp. 1–14.
[28] K. Zhao et al., “Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2020, pp. 239–248.
[29] H. Wang, F. Zhang, X. Xie, and M. Guo, “DKN: Deep knowledge-aware network for news recommendation,” in Proc. World Wide Web Conf. World Wide Web (WWW), 2018, pp. 1835–1844.
[30] S. Rendle, C. Freudenthaler, S. Gantner, and L. Schmidt-Thieme, “BPR: Bayesian personalized ranking from implicit feedback,” in Proc. UAI, 2009, pp. 452–461.
Y. Sun, F. Zhang, H. Zhu, Q. He, and H. Xiong, “Cost-effective and interpretable job skill recommendation with deep reinforcement learning,” in Proc. Web Conf., Apr. 2021, pp. 3827–3838.

Y. Wang, “A hybrid recommendation for music based on reinforcement learning,” in Proc. PAKDD, 2020, pp. 91–103.

P. Wei, S. Xia, R. Chen, J. Qian, C. Li, and X. Jiang, “A deep-reinforcement-learning-based recommender system for occupant-driven energy optimization in commercial buildings,” IEEE Internet Things J., vol. 7, no. 7, pp. 6402–6413, Jul. 2020.

X. He et al., “Learning to collaborate in multi-module recommendation via multi-agent reinforcement learning without communication,” in Proc. SALT Conf. Recommender Syst., Sep. 2020, pp. 210–219.

G. Ke, H.-L. Du, and Y.-C. Chen, “Cross-platform dynamic goods recommendation system based on reinforcement learning and social networks,” Appl. Soft Comput., vol. 104, Jun. 2021, Art. no. 107213.

O. Jangmin, J. Lee, J. W. Lee, and B.-T. Zhang, “Adaptive stock trading with dynamic asset allocation using reinforcement learning,” Inf. Sci., vol. 176, no. 15, pp. 2121–2147, Aug. 2006.

J. Zhang, B. Hao, B. Chen, C. Li, H. Chen, and J. Sun, “Hierarchical reinforcement learning for course recommendation in MOOCs,” in Proc. AAAI, 2019, pp. 435–442.

Y. Lin et al., “Hierarchical reinforcement learning with dynamic recurrent mechanism for course recommendation,” Knowl.-Based Syst., vol. 244, May 2022, Art. no. 108546.

L. Wang, W. Zhang, X. He, and H. Zha, “Supervised reinforcement learning with recurrent neural network for dynamic recommendation,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 2447–2456.

Z. Zheng et al., “Interaction-aware drug package recommendation via policy gradient,” ACM Trans. Inf. Syst., vol. 41, no. 1, pp. 1–32, Feb. 2022.

S. M. Shortreed, E. Laber, D. J. Lizotte, J. S. Stroup, J. Pineau, and S. A. Murphy, “Informing sequential clinical decision-making through reinforcement learning: An empirical study,” Mach. Learn., vol. 84, no. 1–2, pp. 109–136, Jul. 2011.

M. M. Afsar, T. Crump, and B. H. Far, “Reinforcement learning based recommender systems: A survey,” ACM Comput. Surv., vol. 55, no. 7, pp. 1–37, Jun. 2022.

X. Chen, L. Yao, J. McAuley, G. Zhou, and X. Wang, “A survey of deep reinforcement learning in recommender systems: A systematic review and future directions,” 2021, arXiv:2109.03540.

B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, “Adversarial point-of-interest recommendation,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 6, pp. 2042–2062, Jun. 2018.

R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT, 2018.

C. J. C. H. Watkins and P. Dayan, “Technical note Q-learning,” Mach. Learn., vol. 8, no. 3, pp. 279–292, 1992.

V. Mnih et al., “Playing Atari with deep reinforcement learning,” 2013, arXiv:1312.5062.

R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” Mach. Learn., vol. 8, nos. 3–4, pp. 229–256, May 1992.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in Proc. NIPS, 2000, pp. 1057–1065.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” in Proc. ICML, 2015, pp. 1889–1897.

S. Levine and V. Koltun, “Guided policy search,” in Proc. ICML, 2013, pp. 1–9.

V. R. Konda and J. N. Tsitsiklis, “On actor-critic algorithms,” SIAM J. Control Optim., vol. 42, no. 4, pp. 1143–1166, 2003.

V. Mnih et al., “Asynchronous methods for deep reinforcement learning,” in Proc. IJCAI, 2016, pp. 1928–1937.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in Proc. ICLR, 2018, pp. 1856–1865.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in Proc. ICML, 2014, pp. 387–395.

S. Adam, L. Busoniu, and R. Babuska, “Experience replay for real-time reinforcement learning control,” IEEE Trans. Syst., Man, Cybern., C Appl. Rev., vol. 42, no. 2, pp. 201–212, Mar. 2012.

X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose, “Self-supervised reinforcement learning for recommender systems,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2020, pp. 931–940.

F. Liu, H. Guo, X. Li, R. Tang, Y. Ye, and X. He, “End-to-end deep reinforcement learning based recommendation with supervised embedding,” in Proc. 13th Int. Conf. Web Search Data Mining, Jan. 2020, pp. 384–392.

N. Taghipour, A. Kardan, and S. S. Ghidary, “Usage-based web recommendations: A reinforcement learning approach,” in Proc. ACM Conf. Recommender Syst., Oct. 2007, pp. 113–120.

Y. Zhang, L. C. Zhang, and X. Liu, “Dynamic scholarly collaborator recommendation via competitive multi-agent reinforcement learning,” in Proc. 11th ACM Conf. Recommender Syst., Aug. 2017, pp. 331–335.

T. Mahmood and F. Ricci, “Learning and adaptivity in interactive recommender systems,” in Proc. 9th Int. Conf. Electron. Commerce, Aug. 2007, pp. 75–84.

R. Gao, H. Xia, J. Li, D. Liu, S. Chen, and G. Chun, “DRCGR: Deep reinforcement learning framework incorporating CNN and GAN-based for interactive recommendation,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2019, pp. 1048–1053.

Y. Lei and W. Li, “Interactive recommendation with user-specific deep reinforcement learning,” ACM Trans. Knowl. Discovery Data, vol. 13, no. 6, p. 61, Oct. 2019.

L. Zou, L. Xia, Z. Ding, J. Song, W. Liu, and D. Yin, “Reinforcement learning to optimize long-term user engagement in recommender systems,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 2810–2818.

L. Zou et al., “Pseudo Dyna-Q: A reinforcement learning framework for interactive recommendation,” in Proc. 15th Int. Conf. Web Search Data Mining, Jan. 2020, pp. 816–824.

S. Zhou et al., “Interactive recommender system via knowledge graph-enhanced reinforcement learning,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2020, pp. 179–188.

R. Zhang, T. Yu, Y. Shen, H. Jin, and C. Chen, “Text-based interactive recommendation via constraint-augmented reinforcement learning,” in Proc. NIPS, 2019, pp. 15214–15224.

H. Chen et al., “Large-scale interactive recommendation with tree-structured policy gradient,” in Proc. AAAI, 2019, pp. 3312–3320.

W. Liu, F. Liu, R. Tang, B. Liao, G. Chen, and P. A. Heng, “Balancing between accuracy and fairness for interactive recommendation with reinforcement learning,” in Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining, 2020, pp. 354–365.

T. Xiao and D. Wang, “A general offline reinforcement learning framework for interactive recommendation,” in Proc. AAAI, 2021, pp. 1–11.

T. Yu, Y. Shen, R. Zhang, X. Zeng, and H. Jin, “Vision-language recommendation via attribute augmented multimodal reinforcement learning,” in Proc. 27th ACM Int. Conf. Multimedia, Oct. 2019, pp. 39–47.

F. Liu et al., “State representation modeling for deep reinforcement learning based recommendation,” Knowl.-Based Syst., vol. 205, Art. no. 106710, 2020.

M. S. Llorente and S. Escalera Guerrero, “Increasing retrieval quality in conversational recommenders,” IEEE Trans. Knowl. Data Eng., vol. 32, no. 10, pp. 1876–1888, Oct. 2020.

T. Mahmood and F. Ricci, “Improving recommender systems with adaptive conversational strategies,” in Proc. 20th ACM Conf. Hypertext Hypermedia, Jun. 2009, pp. 73–82.

Y. Wu, C. Macdonald, and I. Ounis, “Partially observable reinforcement learning for dialog-based interactive recommendation,” in Proc. 15th ACM Conf. Recommender Syst., Sep. 2021, pp. 241–251.
