Journal of Computers Vol. 36 No. 1, February 2025, pp. 113-126
https://doi.org/10.63367/19911599202502360100 8

Multi-agent Deep Reinforcement Learning Recommendation Algorithm Based on User Positive and Negative Feedback

Fan-Lin Hu and Gao-Peng Wang*

School of Software Engineering, Chongqing University of Post and Telecommunications, Chongqing, China
s221231021@stu.cqupt.edu.cn, wanggp@cqupt.edu.cn

Received 8 May 2024; Revised 11 September 2024; Accepted 27 September 2024

Abstract. Recommendation systems play an increasingly crucial role amidst the prevalence of information overload. However, most existing recommendation systems perceive the recommendation process as static, thus overlooking the implicit information value in interactions between users and systems. Additionally, the majority of current research efforts are focused on processing positive user feedback, while neglecting negative feedback. However, negative feedback also contains valuable user preference information and has the potential to significantly improve recommendation tasks. Therefore, this paper introduces a novel recommendation model based on Deep Reinforcement Learning (DRL), referred to as MDRR-att. In the proposed framework, we presents a specially designed state generation module that incorporates an attention mechanism to extract real-time preference information from users' historical interaction records. Furthermore, a multi-agent Actor-Critic algorithm is employed to simulate the real-time recommendation process, enabling the dynamic collection of user preference information while effectively utilizing negative feedback data. Experiments were conducted on two publicly available datasets. The results suggest that our proposed recommendation framework offers significant advantages in utilizing user negative feedback information.

Keywords: recommendation system, deep reinforcement learning, negative feedback, multi-agent, attention mechanism

1 Introduction

In the digital age, the primary objective of recommendation systems is to deliver information that aligns with users' interests and requirements. However, users' interests are inherently diverse and subject to rapid fluctuations, which poses a significant challenge for traditional recommendation systems in adapting to these swift changes, particularly in relation to users' positive and negative feedback. To improve the efficacy of recommendation systems, this study introduces a novel recommendation model—MDRR-att. This model integrates the benefits of users' positive and negative feedback, attention mechanisms, and multi-agent systems to offer more precise and personalized recommendations.

In contemporary recommendation systems, collaborative filtering methods have demonstrated considerable success, particularly through techniques such as matrix factorization, as introduced by Koren et al. [1]. However, these methods frequently encounter challenges in addressing the cold start problem associated with new users or items, and they often exhibit insufficient responsiveness to rapid changes in user interests. Furthermore, traditional collaborative filtering approaches tend to overlook users' negative feedback, as highlighted in [2], thereby constraining the system's capacity to adapt to user dissatisfaction. Content-based methods, exemplified by the content recommendation model proposed in [3], primarily focus on aligning item features with user preferences. Nevertheless, these approaches frequently neglect the dynamic nature of user preferences, resulting in recommendations that may lack both timeliness and diversity. Typically, these methods rely on static user profiles that are not updated in real-time to reflect shifts in user interests. Hybrid recommendation systems, as suggested in [4], aim to integrate collaborative filtering and content-based techniques to enhance the accuracy and coverage of recommendations. Despite this integration, such systems continue to grapple with efficiency challenges when

* Corresponding Author

processing large-scale data and real-time feedback, often employing single-agent decision-making that inadequately simulates the complexities of user behavior. Deep learning models, as proposed in [5], enhance recommendation accuracy through the automatic extraction of features. However, these models generally necessitate substantial data and computational resources, and they may lack adaptability in the face of abrupt changes in user behavior. Deep reinforcement learning (DRL) recommendation systems offer the potential to interact with the environment to learn optimal recommendation strategies, showcasing their capability to address complex decision-making challenges. Nonetheless, many DRL models encounter practical application issues, including learning efficiency, stability, and scalability. Additionally, numerous DRL models do not fully leverage users' negative feedback, as discussed in [6], which further constrains their ability to respond effectively to user dissatisfaction.

In the existing literature on recommendation systems, the effective management of both positive and negative feedback is crucial for enhancing the quality of recommendations. For example, Zhao et al. [7] examined the significance of user feedback in recommendation systems in their comprehensive review, highlighting the necessity of differentiating between positive and negative feedback. Attention mechanisms, recognized as a potent tool, have been shown to improve the performance of recommendation systems. Cheng et al. [8] illustrated in their research how attention mechanisms can be employed to better capture user interests within recommendation systems. Additionally, multi-agent systems [9] exhibit considerable potential in simulating the interactive effects present in users' decision-making processes.

The MDRR-att model proposed in this study seeks to overcome the limitations of current recommendation systems by integrating the benefits of both positive and negative user feedback, attention mechanisms, and multi-agent systems, thereby offering more accurate and personalized recommendations. The design of the model thoroughly considers the dynamic and multi-dimensional characteristics of user behavior, facilitating a comprehensive understanding of user preferences through innovative state generation modules and multi-agent architectures. Experimental results indicate that the MDRR-att model surpasses existing methods across various evaluation metrics, underscoring its potential for application and research significance within the domain of recommendation systems. Future research will further investigate the optimization and extension of the model, building upon this foundation to accommodate more complex recommendation scenarios and diverse user needs.

The primary contributions of this paper are as follows: (1) The MDRR-att model integrates both positive and negative feedback from users into the recommendation process, thereby offering a more nuanced understanding of user preferences through the construction of distinct representations for positive and negative states. (2) By incorporating an attention layer, the model is capable of assigning varying weights to users' preferences for items, which enhances the relevance and accuracy of the recommendations provided. (3) The model employs a multi-agent architecture to separately process and learn from positive and negative states, thereby improving the model's adaptability and flexibility in response to user behavior. (4) The model has been validated in real-world recommendation scenarios, demonstrating the efficacy of the MDRR-att model and its superiority over existing technologies across multiple dimensions.

The structure of this paper is organized as follows: Section 2 presents a comprehensive analysis of current recommendation algorithms and their inherent limitations. Section 3 explores the modeling and application of deep reinforcement learning within recommendation systems. Section 4 elaborates on the architecture, principles, and implementation methodologies of the MDRR-att model, including the training process. Section 5 illustrates the performance of the MDRR-att model and highlights its advantages over alternative recommendation systems through comparative experiments. Finally, Section 6 concludes the paper and offers insights into potential future research directions.

2 Related Works

In the contemporary digital landscape characterized by an abundance of information, recommendation systems are essential for improving user experience. These systems analyze user behavior and preferences to deliver personalized content recommendations. With ongoing technological advancements, recommendation systems have progressed from conventional collaborative filtering techniques to sophisticated models that incorporate deep learning and reinforcement learning methodologies. This section will critically review pertinent research from recent years to highlight the innovative features and potential benefits of the MDRR-att model in comparison to existing technologies.

Traditional recommendation systems predominantly rely on collaborative filtering techniques, which encompass user-based collaborative filtering (User-based CF) and item-based collaborative filtering (Item-based

CF). User-based CF generates recommendations by identifying groups of users exhibiting similar rating behaviors [10]. Conversely, Item-based CF recommends products by assessing the similarity between items [11]. Nonetheless, these methodologies encounter several challenges, including the cold start problem, scalability issues, and data sparsity, particularly in scenarios characterized by a substantial increase in the number of users and items [12, 13]. To mitigate these challenges, matrix factorization (MF) techniques have been integrated into collaborative filtering, facilitating the identification of latent factors through the decomposition of the user-item rating matrix [14]. A notable example of enhancing recommendation quality through matrix factorization is Netflix's recommendation algorithm [15]. However, it is important to acknowledge that MF methods possess limitations in effectively capturing the dynamics of evolving user interests over time.

The advancement of deep learning technologies has led to substantial enhancements in recommendation systems. By employing deep neural networks, these systems are capable of learning more abstract and intricate patterns of user-item interactions, as well as extracting deep-level features from auxiliary information [16]. For instance, convolutional neural networks (CNNs) are utilized to analyze movie posters and product images, thereby improving the feature representation of items within recommendation systems [17]. The implementation of the BERT model has significantly augmented the capacity of text-based recommendation systems to discern nuanced differences in content [18]. Furthermore, recurrent neural networks (RNNs), particularly long short-term memory networks (LSTMs), are proficient in tracking sequences of user behavior, offering a novel perspective on the temporal dynamics of user interactions [17].

Deep reinforcement learning (DRL) presents a novel approach to recommendation systems by conceptualizing the recommendation process as a continuous decision-making framework. DRL methodologies are capable of learning from user feedback in real-time, thereby enabling the dynamic adjustment of recommendation strategies to deliver personalized suggestions [19]. For instance, the Q-learning algorithm has been utilized in video recommendation systems to enhance the relevance of video suggestions by continuously updating the value function [20]. Additionally, policy gradient methods, including Actor-Critic architectures, have been employed in music recommendation systems; these approaches assess recommendations and modify strategies to accommodate fluctuations in user preferences [21]. Nevertheless, these algorithms predominantly concentrate on positive feedback, often overlooking the potential insights derived from negative feedback, which constrains their capacity to comprehensively understand user preferences.

In recent years, research in recommendation systems has advanced significantly, resulting in the development of numerous innovative models and technologies. For example, recommendation systems based on graph neural networks effectively capture intricate social relationships and contextual information by constructing interaction graphs that represent the relationships between users and items [22]. However, these models continue to encounter challenges when addressing large-scale data and real-time recommendation scenarios. Additionally, multimodal learning within recommendation systems has made notable progress by integrating diverse types of data, including text, images, and videos, thereby enhancing the richness and diversity of these systems [23]. Nonetheless, these approaches often necessitate considerable computational resources for the integration of various modalities, and there remains potential for improvement in the management of negative feedback.

Revised Text: This study presents the MDRR-att model, which synthesizes user feedback, attention mechanisms, and multi-agent systems to overcome the limitations of current recommendation systems in accommodating the dynamic fluctuations in user interests and real-time feedback. The primary research question addressed by the MDRR-att model is how to effectively incorporate both positive and negative user feedback to optimize recommendation strategies, as well as how to enhance the adaptability and flexibility of the recommendation process through the implementation of multi-agent systems. Comprehensive experiments conducted on various public datasets demonstrate that the MDRR-att model surpasses existing models in terms of accuracy, diversity, and user satisfaction, thereby underscoring its potential applications and research significance within the domain of recommendation systems.

3 The Overall Framework of Reinforcement Learning Recommendation System

In this paper, we examine the interaction process between the recommendation system and users as illustrated in Fig. 1. The recommendation system aims to infer user preferences based on the history of users' interactions with the system (i.e., the state s_t), and recommends items (i.e., action a_t) in line with users' preferences. Users will then provide feedback (i.e., rewards r_t) on the recommended items, and the recommendation system will assess the accuracy of these recommendations based on user feedback. The goal of the recommendation system is to

suggest items that align with the user's preferences, aiming to maximize user satisfaction. Therefore, this process is modeled as a Markov Decision Process (MDP) in this paper. An MDP can be defined as follows:

State space S:A state $s_t \in S$ is defined as the collection of items with which users have interacted with the system prior to time t and the related information of users registered in the system (if registered).

Action space A: An action $a_t \in A$ is chosen by the system based on the user's state at time t.

State transition function P: P is a probability function that indicates the probability of the state transitioning from s_t to s_{t+1} after taking action a_t.

Reward function R: The recommendation system will suggest a set of items to the user based on the action a_t, and the user will provide feedback on the recommended items. Different feedback taken by the user on the recommended items will result in different rewards (i.e., rewards r_t) for the agent.

Discount factor γ : This factor is used to measure the present value of long-term rewards. When $\gamma = 0$, it signifies that the agent considers only immediate rewards and disregards future ones. When $\gamma = 1$, both short-term and long-term rewards are deemed equally important.

Within this system, at each time step t, the recommendation agent decides on an action $a_t \in A$ based on the current user state $s_t \in S$. Subsequently, it makes recommendations based on this action a_t and receives a reward $r(s_t, a_t)$. Subsequently, the user's state is updated from s_t to s_{t+1} according to the state transition function $P(s_{t+1}, s_t, a_t)$. The objective of the recommendation agent is to discover an optimal policy π_θ that maximizes the expected cumulative reward of the recommendation system over time.

In the recommendation model proposed in this paper, the action is represented by a continuous parameter vector rather than by one or a set of items. This vector is then dot-multiplied with the feature vector of items, and the item with the highest dot product score is recommended to the user. The user receives the system's recommendation and provides feedback to the recommendation system, which then updates the user's state and receives a reward based on the feedback.

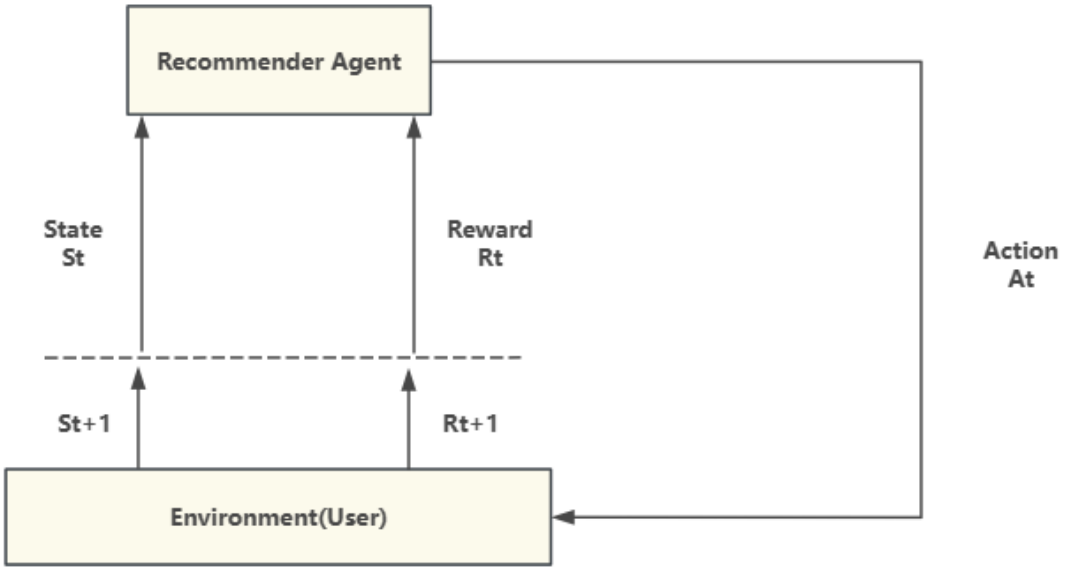

Fig. 1. Recommended deep reinforcement learning

4 MDRR-att Model Framework

In this paper, a reinforcement learning recommendation algorithm (MDRR-att) based on positive and negative user feedback is developed. The algorithm is based on a dual-Actor architecture designed to handle both positive feedback (i.e., features of items that users like) and negative feedback (i.e., features of items that users dislike)

from users. This method not only offers personalized recommendations but also improves the accuracy of the model.

4.1 State Generation Module

The state generation module is responsible for producing the current state of the user by utilizing the user's feature information alongside historical interaction data with the system. An effective state generation network is capable of not only generating high-quality state information but also improving the overall performance of the recommendation system.

As illustrated in Fig. 2, the user's feature vector, along with the feature vectors of the top N items that received positive and negative feedback from the user prior to time t, are utilized as inputs to the state representation module. The output of this module corresponds to the user's current state. The state representation module is bifurcated into two segments: the left segment is dedicated to the generation of the positive state, while the right segment is responsible for the generation of the negative state. The process of generating the positive feedback state is further subdivided into three components: (1) the positive item feedback from the user, in conjunction with the user's feature vector on the right, is processed through an attention network and an average pooling layer to produce a weighted feature vector that encapsulates the user's positive feedback items; (2) the user's feature vector on the right, which conveys the user's feature information; and (3) the resultant vector derived from the matrix multiplication operation located in the central section. The weights of the weighted feature vectors are computed using equations (1), (2), and (3). This component is primarily designed to distinguish the varying degrees of preference that different users exhibit towards the same items.

$$a'_{uv} = \mathrm{Re}\,LU\left(\left(\left[u, i_v\right]W_2\right) + b_2\right)W_1 + b_1 . \tag{1}$$

$$a_{uv} = \frac{\exp(a'_{uv})}{\sum_{i=1}^{n} \exp(a'_{uv})} . \tag{2}$$

$$avgpooling\left(i\right) = avg\left(a_{uv} i_v\right) \mid v = 1, \cdots, n . \tag{3}$$

In this context, u represents the user feature vector, while i_v denotes the feature vector associated with item v. The parameters W_1, W_2 and b_1, b_2 correspond to the weights and biases of the attention network, respectively. The variable a_{uv} indicates the preference weight of user u for item v. The function $avg\_pooling(\cdot)$ refers to the average pooling layer, and i signifies the weighted feature vector.

The generation of the negative feedback state is similarly categorized into the three previously mentioned components. The primary distinction lies in the fact that the item vector input into the attention network transitions from the user's positive feedback items to the user's negative feedback items (referred to as negative items), while all other operations remain consistent. As a result, this model incorporates not only the user's characteristic information but also extracts historical interaction data between the user and the recommendation system. It simultaneously accounts for both positive and negative feedback provided by the user concerning the system's recommendations. The state generation module can be mathematically represented by the following equations:

$$s_t = [s_t^+, s_t^-] . \tag{4}$$

$$s_t^+ = \left[i_k^+, u \otimes i_k^+, u \right] . \tag{5}$$

$$s_t^- = \left[u, u \otimes i_k^-, i_k^- \right] . \tag{6}$$

Herein, $\otimes$ denotes the element-wise product, u represents the feature vector of the user, i_k^+ and i_k^- denote the weighted eigenvectors corresponding to positive and negative feedback items, respectively. s_t^+ denotes the user's

positive feedback state, s_t^- signifies the user's negative feedback state, and s_t represents the comprehensive state of the user. The dimensionality of all users and items is r, the dimensions of s_t^+ and s_t^- are $3r$, and the dimension of s_t is $6r$.

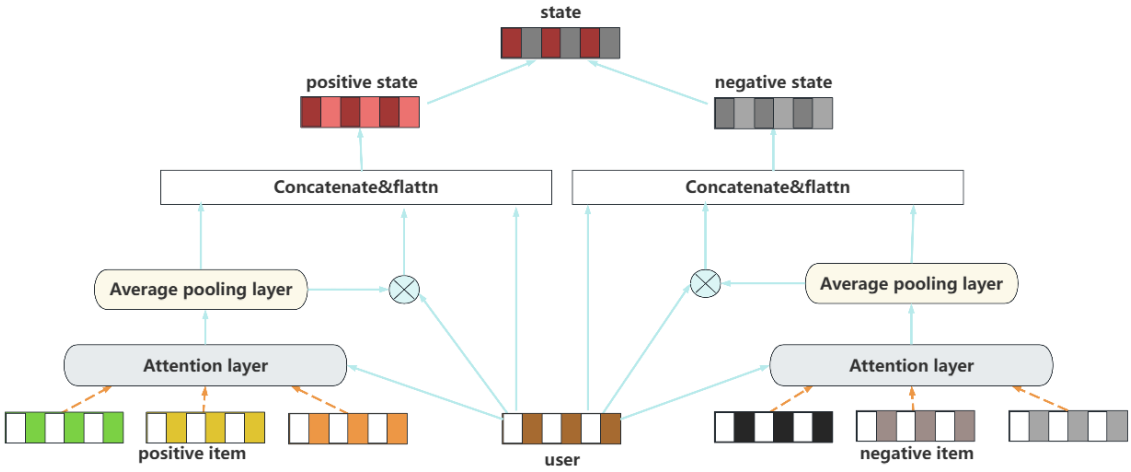

Fig. 2. State generation module

4.2 MDRR-att Module Framework

As illustrated in Fig. 3, this study introduces a recommendation system model constructed on the basis of a dual-actor architectural design. The model employs two mutually independent yet collaborative actor functions to process both positive and negative feedback from users, thereby producing more accurate and personalised recommendation. outcomes.

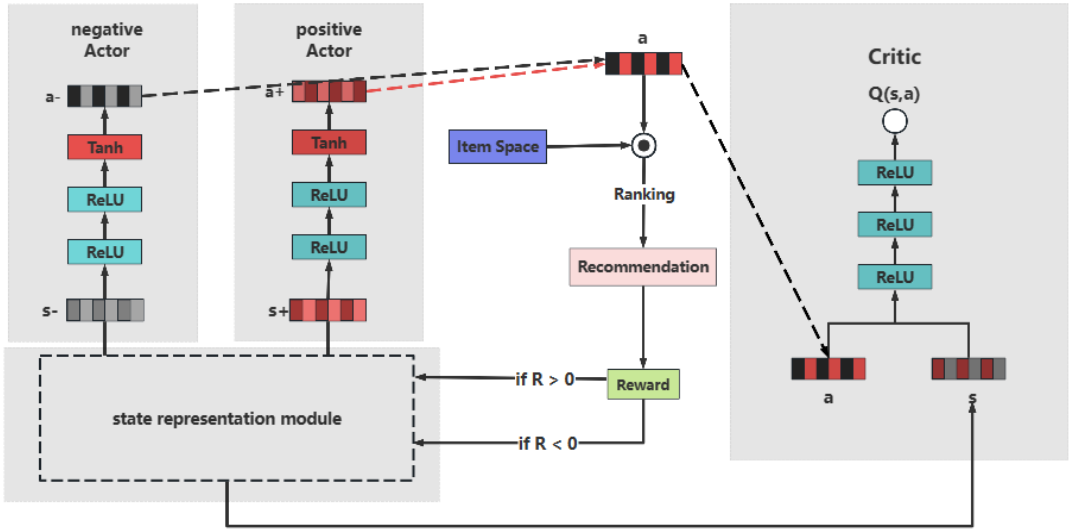

Fig. 3. MDRR-att module framework

Actor Network. The Actor network, also referred to as the policy network, is utilised to facilitate the generation of actions based on the user's current state. As illustrated in the upper left section of Fig. 3, at time t, the positive and negative feedback states s_t^+ and s_t^- generated by the state generation module are inputted into the positive Actor network $\pi_{\theta^+}^+$, which processes positive feedback information, and the negative Actor network $\pi_{\theta^-}^-$, which handles negative feedback information, respectively. Subsequently, they are transformed into actions through two layers of ReLU linear rectifier functions and one layer of Tanh hyperbolic tangent function to be transformed into actions $a_t^+ = \pi_{\theta^+}^+(s_t^+)$, and $a_t^- = \pi_{\theta^-}^-(s_t^-)$. Subsequently, based on equation (7):

$$a_t = sum\left(a_t^+, -a_t^-\right). \tag{7}$$

The two actions are combined into a composite action utilising the $sum(\cdot)$ function. As previously established, the previous section, the dimensions of s_t^+ and s_t^- are *3r*, the dimension of s_t is *6r*, and the action $a_t \in R^{1\times r}$. The action entails a matrix multiplication operation with each item in the item space, resulting in a similarity score for each item, as illustrated in Equation (8).

$$score_I = i_k a_t^T. \tag{8}$$

In this context, the subscript k denotes the k-th item in the item space, represented by the variable i_k. Subsequently, the items in the item space are ordered in descending order of similarity score, with the top N ranked items recommended to the user. In this process, an ε-greedy strategy is employed.

Critic Network. The Critic network, also referred to as the value function, is depicted on the right side of Fig. 3 and is utilised to evaluate the expected return of the actions a_t taken by our model under the state s_t. The Critic network is a Deep Q-Network, which approximates the true action-value function $Q_\pi(s_t, a_t)$, that is to say, the q-value function - with a deep neural network parameterised as $Q_\omega(s_t, a_t)$. The q-value is a measure of the actions a_t taken by the Actor network in state s_t. In particular, the input to the Critic network receives as input the user state s_t generated by the user state generation module and the action a_t generated by the policy network Actor. the output is the q-value, which is a scalar. Based on the q-value, the parameters of the Actor network are a manner that improves the performance of action a_t, thereby enhancing $Q_\omega(s_t, a_t)$. In the field of reinforcement learning, a considerable number of applications have been developed with the objective of identifying the optimal action-value function, represented by $Q^*(s_t, a_t)$. This function seeks to identify the maximum expected return through the optimal policy, as described by the Bellman equation:

$$Q^*\left(s_t, a_t\right) = E_{s_{t+1}}\left[r_t + \gamma \max_{a_{t+1}} Q^*\left(s_{t+1}, a_{t+1}\right) \mid s_t, a_t\right]. \tag{9}$$

In practical situations, it is not feasible to calculate the complete action space *A* in order to select the optimal action a_t using equation (9), given that the action space in real-world scenarios is typically vast. Accordingly, the proposed model employs a deterministic action *a*, generated by the Actor network, thereby circumventing the computational burden associated with calculating the entire action space *A* as per equation (9). Accordingly, equation (9) is modified as follows:

$$Q_\omega\left(s_t, a_t\right) = E_{s_{t+1}}\left[r_t + \gamma Q_\omega\left(s_{t+1}, a_{t+1}\right) \mid s_t, a_t\right]. \tag{10}$$

Subsequently, the Actor is updated by sampling the policy gradient in accordance with the Deterministic Policy Gradient theorem, the update formula for which is as follows:

$$\nabla_\theta J\left(\pi_\theta\right) \approx \frac{1}{N}\sum\nolimits_t \nabla_a Q_\omega\left(s,a\right)\big|_{s=s_t, a=\pi_\theta(s_t)} \nabla_\theta \pi_\theta\left(s\right)\big|_{s=s_t}. \tag{11}$$

where $J(\pi_\theta)$, is the expectation of all possible q-values under policy π_θ , and a mini-batch strategy is used here, with N representing the batch size. Since the proposed model of this paper is a dual-actor network, the above equation (11) needs to be rewritten as follows:

$$\nabla_{\theta_+} J\left(\pi_{\theta_+}\right) \approx \frac{1}{N} \sum_t \nabla_a Q_\omega(s,a)\big|_{s=s_t, a=\pi_\theta(s_t)} \nabla_{\theta_+} \pi_{\theta_+}\left(s^+\right)\big|_{s^+=s_t^+} . \quad (12)$$

$$\nabla_{\theta_-} J\left(\pi_{\theta_-}\right) \approx \frac{1}{N} \sum_t \nabla_a Q_\omega(s,a)\big|_{s=s_t, a=\pi_\theta(s_t)} \nabla_{\theta_-} \pi_{\theta_-}\left(s^-\right)\big|_{s^-=s_t^-} . \quad (13)$$

where θ_+ and θ_- are the parameters for the positive feedback Actor $\pi_{\theta^+}^+$ and negative feedback Actor $\pi_{\theta^-}^-$, respectively. During the updating process of the Critic network, the temporal difference learning method is employed:

$$L = \frac{1}{N} \sum_t \left(y_t - Q_\omega\left(s_t, a_t\right)\right)^2 . \quad (14)$$

where $y_t = r_t + \gamma Q_{\omega'}(s_{t+1}, \pi_{\theta'}(s_{t+1}))$, with $r_t = R(s_t, a_t)$, representing the immediate reward and γ being the discount factor. The MDRR-att framework also adopts the technique of target networks, where ω' and θ' are the parameters of the Critic and Actor networks, respectively.

4.3 Model Train

The training process of the MDRR-att algorithm, as detailed in Algorithm 1, encompasses three phases: action generation, evaluation generation, and model parameter updating.

The training process is initiated with the random initialization of network weights and the buffer D (lines 1-3). The algorithm's core is illustrated in lines 4-28. The action generation phase, delineated on lines 7-9, commences with the calculation of scores for each historical interaction item derived from the user's historic evaluations. This is achieved through the application of the formula $p_{score} = t_i * r_i$, wherein t_i represents the evaluation time, and r_i signifies the user's rating score for item i. In instances where the evaluation is negative, the score is calculated using the formula $p_{score} = t_i * (r_{max} - r_i)$, and subsequently ordered, with r_{max} denoting the maximum rating score. Subsequently, the top n positively rated item matrix $p_t = \{ i_1^+, \ldots, i_n^+ \}$, (where i_k^+ are the feature vectors of positively rated items) and the top n negatively rated item matrix $n_t = \{ i_1^-, \ldots, i_n^- \}$, (where i_k^- are the feature vectors of negatively rated items) are then selected. This selection method considers both temporal factors and user evaluations, given that user preferences are subject to change over time. Subsequently, the state generation module produces three states, designated as s_t^+ , s_t^- , s_t (line 7). The states s_t^+ , s_t^- are then input into the positive Actor network π_{θ_+} and negative Actor network π_{θ_-} , respectively, in order to yield the current actions a_t^+ , a_t^- , and a_t (line 8). Subsequently, an item i_t is then selected from the item space for recommendation to the user, based on the formula (8) (line 9).

The second phase (lines 10-20) involves the utilisation of the Critic network for the computation of the optimal action-value function, $Q_\omega(s_t , a_t)$. Firstly, the reward represented by the function $r_t = R(s_t , a_t)$, is calculated through user feedback. Then, based on the action and state s_t , the optimal action-value function $Q_\omega(s_t , a_t)$, is computed via the Critic network. The user's next state is updated based on the rewards to s_{t+1}^+ , s_{t+1}^- , $s_{t+1} = f(p_{t+1}, n_{t+1})$, where $f(\cdot)$ denotes the state generation module, and p_{t+1} and n_{t+1} respectively represent the list of user positive and negative feedback items at the next time step. If r_t is positive, the user's positive feedback state is updated, such that $p_{t+1}=U^+(p_t , i_t)$, where $U^+(\cdot)$ denotes the function that calculates the scores for the items in p_t and i_t using the formula $p_{score} = t_i * r_i$, and removes the item with the lowest score. Conversely, if r_t is negative, the user's negative feedback state is updated to $n_{t+1}=U^-(n_t , i_t)$, where $U^-(\cdot)$, denotes the function that calculates the scores for the items in n_t and i_t using the formula $p_{score} = t_i * (r_{max} - r_i)$, removing the item with the lowest score.

The trajectory (s_t^+ , s_t^- , s_t , a_t^+ , a_t^- , a_t, r_t, s_{t+1}^+ , s_{t+1}^-), is subsequently stored in the buffer D.

The third phase (lines 21-28), the model update, begins with the target iteration value being obtained through formula (14). This is followed by the Actor and Critic networks being updated based on the respective update formulas. In conclusion, the parameters θ and ω of the target networks are updated.

Algorithm 1. The training process of the MDRR-att algorithm

Input: Actor learning rate ηa, Critic learning rate ηc, discount factor λ, sample size N, user vector space U, item vector space I

Output: θ_+ , θ_- and ω

1: Initialization of Actor $\pi_{\theta+}$, Actor $\pi_{\theta-}$, and Critic Q_ω with

Parameters θ_+ , θ_- and ω;

2: Initialize the target networks π'_+ , π'_- , and Q' with weights set as

$\theta'_+ \leftarrow \theta_+$, $\theta'_- \leftarrow \theta_-$, and $\omega' \leftarrow \omega$;

3: Initialize the replay buffer D

4: **for** session = 1, M **do for**

5: Observe the initial state s_0 based on the historical interactions:

6: **for** t = 1, T **do**

7: Observe the states s_t^+ , s_t^- and $s_t = f(p_t, n_t)$, where $P_t = \{ i_1^+ , i_n^+ \}$, $n_t = \{ i_1^- , i_n^- \}$;

8: Based on the ε − greedy policy $\pi_{\theta+}$ and $\pi_{\theta-}$, obtain the actions a_t^+ , a_t^- and a_t ;

9: Based on the action a_t, recommend item i_t ;

10: Calculate the reward $r_t = R(s_t, a_t)$ based on user feedback;

11: Calculate $Q_\omega(s_t , a_t)$ through the Critic network;

12: Observe the next state s_{t+1}^+ , s_{t+1}^- , $s_{t+1} = f(p_{t+1}, n_{t+1})$

13: **if** $r_t > 0$:

14: $p_{t+1} = U^+(p_t, i_t)$;

15: $n_{t+1} = n_t$;

16: **else if** $r_t < 0$:

17: $n_{t+1} = U^-(n_t, i_t)$;

18: $p_{t+1} = p_t$;

19: **end if**

20: Store the trajectory (s_t^+ , s_t^- , s_t , a_t^+ , a_t^- , a_t, r_t, s_{t+1}^+ , s_{t+1}^-), into the replay buffer D;

21: Sample a mini-batch of trajectories (s_i^+ , s_i^- , s_i , a_i^+ , a_i^- , a_i, r_i, s_{i+1}^+ , s_{i+1}^-), from the replay buffer ***D*** using the technique of prioritized experience replay;

22: Set $y_i = r_i + \gamma Q_{\omega'}(s_{i+1}, \pi_{\theta'}(s_{i+1}))$;

23: Update the Critic network by minimizing the loss:

$$L = \frac{1}{N}\sum_i (y_i - Q_\omega(s_i, a_i))^2;$$

24: Update Actor $\pi_{\theta+}$ and Actor $\pi_{\theta-}$ using sampled policy gradients:

$$\nabla_{\theta_+} J(\pi_{\theta_+}) \approx \frac{1}{N}\sum_t \nabla_a Q_\omega(s,a)\Big|_{s=s_t, a=\pi_\theta(s_t)} \nabla_{\theta_+} \pi_{\theta_+}(s^+)\Big|_{s^+=s_t^+}$$

$$\nabla_{\theta_-} J(\pi_{\theta_-}) \approx \frac{1}{N}\sum_t \nabla_a Q_\omega(s,a)\Big|_{s=s_t, a=\pi_\theta(s_t)} \nabla_{\theta_-} \pi_{\theta_-}(s^-)\Big|_{s^-=s_t^-}$$

25: Update the target network parameters:

$$\theta'_+ \leftarrow \tau\theta_+ + (1-\tau)\theta'_+$$

$$\theta'_- \leftarrow \tau\theta_- + (1-\tau)\theta'_-$$

$\omega' \leftarrow \tau\omega + (1-\tau)\omega'$;

26: **end for**

27: **end for**

28: **return** θ_+, θ_- and ω;

5 Experiments

5.1 Data Sets

To validate the effectiveness of the MDRR-att model, experiments were conducted on the following two real-world datasets:

MovieLens-100k. A benchmark dataset comprising 100,000 ratings given by 943 users on 1,682 recommended movies on the MovieLens website.

MovieLens-1M. A benchmark dataset containing 1 million ratings from 6,040 users for 3,952 movies on the MovieLens website.

Table 1 presents the statistical information of the datasets:

Table 1. Dataset description

	MovieLens-100k	MovieLens-1M
# user	943	6,040
# item	1,682	3,952
# ratings	100,000	1,000,209

5.2 Evaluation Indicators

The primary goal of this recommendation system is to generate top-N recommendation items for users. Therefore, this paper employs Precision@N and NDCG@K to evaluate the quality of the recommendations produced by the recommendation system. In the proposed model, the model will recommend *N* items to a user at a given moment, and these two metrics are defined as follows:

$$Precision@N = \frac{TP}{TP + FP}. \tag{15}$$

$$NDCG@K = \frac{DCG@K}{IDCG@K}. \tag{16}$$

In the above formula, TP represents the number of recommended items that received positive evaluations from users, and FP denotes the number of items evaluated as non-positive by users. Therefore, TP+FP represents the total number of items evaluated by users. DCG@K considers the positional relevance of items, where K is the considered length of the recommendation list. IDCG@K is the maximum possible DCG for a given recommendation list length. NDCG@K is used to assess the accuracy of the ranking results.

5.3 Experiment Details Settings

For the two datasets referred to above, the paper randomly splits them into two parts: 80% of the ratings are used for training, and the remaining 20% are utilized for evaluation. For these datasets, the reward was set at $r = (ratings\text{-}3)/2$, where ratings of 4 and 5 are defined as positive evaluations, and ratings of 1 and 2 are defined as neg-

ative evaluations. In each episode, the model does not recommend duplicate items. The learning rates of the two Actor networks were set to 0.0001, while the learning rate of the Critic network was set to 0.001. Batch sampling size was set to 64, and the discount rate γ was set to 0.9. To avoid overfitting, the Adam optimizer was utilized to optimize all RL methods based on L_2 norm regularization in this paper. For the evaluation of the model, we utilized the evaluation method in [7]. For a given session, the agent only recommends items that appear in this session, rather than items from the entire item space. This is because in the recorded off-line logs, only the basic real feedback for existing items in the current session is available. Therefore, this evaluation process can be seen as a re-ranking procedure for existing items within the current session. For a fair comparison, all baseline methods were meticulously adjusted in this study.

5.4 Experimental Results and Analysis

To evaluate the performance of the proposed model, we conducted a comparative analysis against several baseline methods utilizing two widely recognized datasets, MovieLens-100k and MovieLens-1M. The evaluation metrics employed in this analysis were Precision@20 and NDCG@20. Furthermore, we examined the influence of various components on the model's performance.

(1) Comparison experiment with baseline model

This study utilizes conventional representative methods as baseline models, which include Popularity [24], PMF [25], and SVD++ [26]. Furthermore, it incorporates deep learning-based approaches such as DeepFM [27] and AFM [28]. Additionally, we conducted a comparison of methods grounded in reinforcement learning, specifically DDPG [29], DQN [30], and Deep Reinforcement Learning for Recommendation Systems DEERS [7]. The experimental results are summarized in Table 2.

Table 2. Performance of each algorithm on the Top-20 models on the MovieLens dataset (The best performing ones are in bold.)

Model	MovieLens-100k		MovieLens-1M	
	Precision@20	NDCG@20	Precision@20	NDCG@20
Popularity	0.5685	0.8720	0.5094	0.8727
PMF	0.5845	0.8849	0.5213	0.8734
SVD++	0.5876	0.8866	0.5183	0.8785
AFM	0.6325	0.8914	0.5714	0.8825
DeepFM	0.6362	0.8941	0.5750	0.8841
DQN	0.6076	0.8815	0.5377	0.8759
DDPG	0.6052	0.8870	0.5364	0.8776
DEERS	0.6481	0.8933	0.5963	0.8878
MDRR-att	**0.6525**	**0.9013**	**0.6247**	**0.9078**

Popularity: Recommend the most popular items, that is, the items with the highest ratings.

PMF: Treat matrix decomposition as singular value decomposition, which only considers non-zero elements.

SVD++: Mixes the advantages of latent models and neighborhood models.

DeepFM: Explicitly model low-order and high-order feature interactions for recommendation.

AFM: Leveraging attention networks to explicitly model the importance of feature interactions for recommendations.

DQN: Use Deep Q-network to generate q-values to evaluate all possible actions in the current state.

DDPG: Simply splicing positive feedback item embeddings to represent user status, this paper uses this method as a baseline to evaluate the effectiveness of the proposed model.

DEERS: A state with positive and negative feedback generated by a Recurrent Neural Network (RNN) within the Deep Q-Network (DQN) framework.

From Table 2, it is evident that the MDRR-att model exhibits superior performance across both datasets, as evidenced by its Precision@20 and NDCG@20 metrics, which significantly surpass those of other models. This indicates that the MDRR-att model possesses a distinct advantage in delivering personalized recommendations.

The MDRR-att model demonstrates a substantial improvement in performance when compared to traditional recommendation models. Unlike the Popularity model, which relies on static data, the MDRR-att model analyzes users' historical behaviors by incorporating both positive and negative feedback to develop a more nuanced user

preference profile, thus providing more personalized recommendations. Furthermore, the deep reinforcement learning framework employed by MDRR-att enables the model to dynamically adjust its recommendation strategy in response to real-time fluctuations in user preferences, a feature that the traditional Popularity model lacks. In comparison to matrix factorization-based models such as PMF and SVD++, the MDRR-att model effectively captures nonlinear and higher-order user-item relationships through deep learning techniques, whereas conventional matrix factorization models are often constrained by linear assumptions and are unable to fully encapsulate this complexity. Additionally, the MDRR-att model can integrate user feedback in real-time to continuously refine the recommendation strategy, whereas traditional matrix factorization models typically necessitate retraining to update their parameters.

In comparison to deep learning-based models such as DeepFM and AFM, the MDRR-att model considers both positive and negative user feedback, thereby facilitating the extraction of a more comprehensive array of user preferences. This approach enhances the relevance and accuracy of recommendations. Additionally, MDRR-att incorporates multi-dimensional features of users and items through its multi-agent architecture, whereas traditional deep learning models typically concentrate on a singular data representation.

In comparison to deep reinforcement learning recommendation models, specifically when evaluating DQN and DDPG models, the MDRR-att framework generates more complex and enriched state representations through its attention mechanism and the incorporation of user negative feedback. This approach facilitates a more comprehensive understanding of user needs and preferences. While the DEERS model, which operates within the DQN framework and employs recurrent neural networks (RNN), generates states based on both positive and negative feedback and demonstrates the capacity to accommodate users' dynamic preferences, MDRR-att consistently outperforms DEERS in delivering personalized recommendations. This superior performance can be attributed to MDRR-att's ability to produce more sophisticated state representations, as well as its multi-agent architecture, which enables independent processing and collaborative optimization of both positive and negative feedback. Such a design provides greater flexibility for strategy adjustments and enhances the model's adaptability to the dynamic changes in user preferences.

(2) Performance testing of models with varying components

To enhance the validation of the contributions of various components of the MDRR-att model to its overall performance, comparative analyses were conducted with the MDRR-n, MDRR-p, and basic MDRR models utilizing the MovieLens-1M dataset. The MDRR-p model emphasizes positive user feedback, whereas the MDRR-n model concentrates on negative feedback. In contrast, the basic MDRR model takes into account both positive and negative feedback but does not integrate an attention mechanism. The results of these experiments are illustrated in Fig. 4.

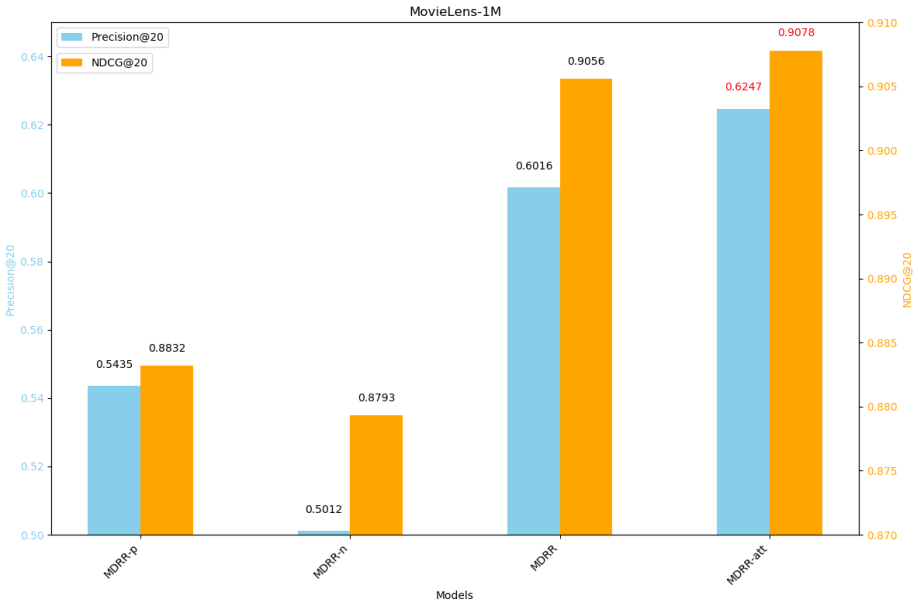

Fig. 4. Performance of each algorithm on movieLens-1M

In the context of the MovieLens-1M dataset, the MDRR-att model demonstrates superior performance compared to the MDRR-p, MDRR-n, and MDRR models, as evidenced by its higher scores in both Precision@20 and NDCG@20 metrics. Notably, the MDRR-p model exhibits enhanced performance relative to the MDRR-n model, indicating that positive user feedback exerts a more favorable influence on the efficacy of the recommendation system within this dataset. This phenomenon may be attributed to users' propensity to actively articulate their preferences for content they favor, while providing comparatively less feedback regarding content they dislike. Furthermore, the basic MDRR model, which integrates both positive and negative feedback, outperforms the MDRR-p and MDRR-n models that consider only a singular type of feedback. This finding underscores the advantage of incorporating both positive and negative feedback in a recommendation system, as it enhances the quality of recommendations. The MDRR model's ability to balance these two types of feedback allows for a more comprehensive understanding of user preferences, whereas the MDRR-p and MDRR-n models, by focusing exclusively on one dimension of user feedback, may hinder the model's capacity to fully grasp user needs. Additionally, the MDRR-att model significantly surpasses the basic MDRR model in performance, attributable to the integration of an attention mechanism. This mechanism enables the MDRR-att model to assign varying weights to identical items for different users, thereby facilitating a more precise capture of individualized user requirements. The comparative analysis of the MDRR-p, MDRR-n, and MDRR models further substantiates the critical roles of positive feedback, negative feedback, and the attention mechanism within the overall model framework.

6 Conclusion

In this paper, we propose a recommendation model that utilizes multi-agent deep reinforcement learning, referred to as MDRR-att. The model incorporates a specially designed state generation module that effectively captures both positive and negative feedback derived from the user's historical interactions with items. This module generates the user's current state information by integrating an attention mechanism. Furthermore, the dual Actor network is capable of simultaneously processing the user's positive and negative feedback, thereby facilitating accurate recommendations based on the user's state. The performance of the proposed MDRR-att model was evaluated using two publicly available datasets, MovieLens-100k and MovieLens-1M. The results indicate that the MDRR-att model outperforms other models in terms of Precision@20 and NDCG@20 metrics.

Future research should concentrate on extracting relational network information among users. This information can be employed to mitigate the cold start problem associated with new users. In instances where there is a lack of evaluative information available for a user during a session, preferences may be inferred by integrating the relationships among users, thereby facilitating more accurate recommendations.

References

[1] Y. Koren, R. Bell, C. Volinsky, Matrix Factorization Techniques for Recommender Systems, Computer 42(8)(2009) 30-37.
[2] G. Adomavicius, B. Mobasher, F. Ricci, A. Tuzhilin, Context-Aware Recommender Systems, AI magazine 32(3)(2011) 67-80.
[3] M.J. Pazzani, D. Billsus, Content-Based Recommendation Systems, The Adaptive Web: Methods and Strategies of Web Personalization 17(2007) 325-341.
[4] R. Burke, Hybrid Recommender Systems: Survey and Experiments. User Model User-Adap Interaction 12(4)(2002) 331-370.
[5] R. Zheng, L. Qu, B. Cui, Y. Shi, H. Yin, AutoML for Deep Recommender Systems: A Survey, ACM Transactions on Information Systems 41(4)(2023) 1-38.
[6] C. Li, I. Ishak, H. Ibrahim, M. Zolkepli, F. Sidi,C. Li, Deep Learning-Based Recommendation System: Systematic Review and Classification, IEEE Access 11(2023) 113790-113835.
[7] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, D. Yin, Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning, in: Proc. 2018 KDD'18: The 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018.
[8] J. Chen, X. Wang, S. Zhao, F. Qian, Y. Zhang, Deep attention user-based collaborative filtering for recommendatio, Neurocomput 383(C)(2020) 57-68.

[9] S. Gronauer, K. Diepold, Multi-agent deep reinforcement learning: a survey, Artif Intelligence Review 55(2)(2022) 895-943.
[10] M.D. Ekstrand, J.T. Riedl, J.A. Konstan, Collaborative Filtering Recommender Systems , Foundations and Trends in Human-Computer Interaction 4(2)(2011) 81-173.
[11] R. Marappan, Movie Recommendation System Using an Item-based Collaborative Filtering, International Journal of Mathematical Engineering Biological and Applied Computing 1(1)(2022) 42-43.
[12] L. Ji, Q. Qin, B. Han, H. Yang, Reinforcement learning to optimize lifetime value in cold-start recommendation, in: Proc. 2021 CIKM'21: The 30th ACM International Conference on Information and Knowledge Management, 2021.
[13] T.M. Al-Hasan, A.N. Sayed, F. Bensaali, Y. Himeur, I. Varlamis, G. Dimitrakopoulos, From Traditional Recommender Systems to GPT-Based Chatbots: A Survey of Recent Developments and Future Directions, Big Data and Cognitive Computing 8(4)(2024) 36.
[14] X. Su, T.M. Khoshgoftaar, Advances in Collaborative Filtering, Advances in Artificial Intelligence 2009(4)(2009) 2.
[15] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N.J. Yuan, X. Xie, Z. Li, DRN: A Deep Reinforcement Learning Framework for News Recommendation, in: Proc. 2018 International World Wide Web Conference Committee, 2018.
[16] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep Learning Based Recommender System: A Survey and New Perspectives, ACM Computing Surveys(CSUR) 52(1)(2020) 1-38.
[17] K. Sanjay, N. Pervin, Integration of short and long-term interests: A preference aware session-based recommender, Neurocomputing 583(C)(2024) 127558.
[18] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proc. 2019 Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019.
[19] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based Recommendations with Recurrent Neural Networks. <https://arxiv.org/abs/1511.06939> , 2015 (accessed 29.03.2016).
[20] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, J. Tang, Deep reinforcement learning for page-wise recommendations, in: Proc. 2018 RecSys 2018 - 12th ACM Conference on Recommender Systems, 2018.
[21] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Gou, Y. Zhang, X. He, State representation modeling for deep reinforcement learning based recommendation, Knowledge-Based Systems 205(2020) 106170.
[22] S. Wu, F. Sun, W. Zhang, X. Xie, B. Cui, Graph Neural Networks in Recommender Systems: A Survey, ACM Computing Surveys 55(5)(2022) 1-37.
[23] L. Qidong, J. Hu, Y. Xiao, X. Zhao, J. Gao, W. Wang, Q. Li, L. Tang, Multimodal Recommender Systems: A Survey. <https://arxiv.oorg/abs/2302.03883> , 2023 (accessed 08.02.2023).
[24] R. Cañamares, P. Castells, Should I Follow the Crowd? A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems, in: Proc. 2018 SIGIR '18: The 41st International ACM SIGIR conference on research and development in Information Retrieval, 2018.
[25] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Proc. 2007 Proceedings of the 20th International Conference on Neural Information Processing Systems, 2007.
[26] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proc. 2008 The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008.
[27] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: A factorization-machine based neural network for CTR prediction, in: Proc. 2017 Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017.
[28] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T.S. Chua, Attentional factorization machines: learning the weight of feature interactions via attention networks, in: Proc. 2017 Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017.
[29] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: Proc. 2016 4th International Conference on Learning Representations, ICLR 2016, 2016.
[30] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518(7540)(2015) 529-533.