Long-Term Recommendation Model for Online Education Systems: A Deep Reinforcement Learning Approach

Wei Wang*
Xianyang Normal University, Xianyang Shaanxi, 712000 Shaanxi, China

***Abstract*—Intelligent tutoring systems serve as tools capable of providing personalized learning experiences, with their efficacy significantly contingent upon the performance of recommendation models. For long-term instructional plans, these systems necessitate the provision of highly accurate, enduring recommendations. However, numerous existing recommendation models adopt a static perspective, disregarding the sequential decision-making nature of recommendations, rendering them often incapable of adapting to novel contexts. While some recent studies have delved into sequential recommendations, their emphasis predominantly centers on short-term predictions, neglecting the objectives of long-term recommendations. To surmount these challenges, this paper introduces a novel recommendation approach based on deep reinforcement learning. We conceptualize the recommendation process as a Markov Decision Process, employing recurrent neural networks to simulate the interaction between the recommender system and the students. Test results demonstrate that our model not only significantly surpasses traditional Top-N methods in hit rate and NDCG concerning the enhancement of long-term recommendations but also adeptly addresses scenarios involving cold starts. Thus, this model presents a new avenue for enhancing the performance of intelligent tutoring systems.**

***Keywords*—Deep reinforcement learning; long-term recommendation; intelligent tutoring system; Markov Decision Process; recurrent neural network**

I. INTRODUCTION

In the current educational landscape, personalized learning is increasingly gaining prominence. Nevertheless, traditional educational approaches often employ static teaching paradigms, overlooking the dynamic and sequential nature of students' learning progress and personalized needs. This oversight may lead to uneven allocation of educational resources, as certain students, regardless of their learning trajectories, might receive similar instructional resources and methods. Additionally, these methods are susceptible to the impact of students' aptitude issues, wherein newly enrolled students may struggle to receive precise personalized recommendations due to a lack of historical learning records [1].

Indeed, education should be construed as a sequential decision-making process, and adaptability is crucial for intelligent tutoring systems, given that students' learning progress and needs invariably evolve over time. To integrate the capability for sequential processing into intelligent tutoring systems, recurrent neural networks (RNNs) have recently been introduced into educational systems. However, the majority of existing sequential learning methods are applicable only to short-term predictions, disregarding predictions for long-term learning. Furthermore, these RNN-based sequence models entirely overlook the interaction between intelligent tutoring systems and students, a pivotal component of interactive reinforcement learning [2].

To address the aforementioned issues, this paper proposes a novel model based on deep reinforcement learning (DRL) for long-term learning prediction. Specifically, the model employs RNN to adaptively evolve the student's learning state to simulate the sequential interaction between the student and the intelligent tutoring system. The model is applicable to cold start scenarios and utilizes an additional gated neural network to balance the influence between the state of the RNN and the historical state derived from the student's learning records [3].

To maximize the expected long-term learning outcomes and optimize model parameters, we present an effective learning approach based on the popular policy gradient algorithm REINFORCE. Extensive experiments conducted on two real-world datasets demonstrate the commendable performance of our proposed model in cold start scenarios, surpassing the current state-of-the-art methods in various learning performance metrics [4].

The remaining part of the paper is organized as follows. Section II provides a literature review on recommender systems in intelligent tutoring systems and deep reinforcement learning, Section III presents the overall framework and detailed mechanisms of the proposed long-term recommendation model based on deep reinforcement learning, Section IV describes the environment and interaction processes within the model, Section V discusses the recommendation agent and its training methodologies, Model training is given in Section VI. Section VII reports on experiments that validate the performance of the proposed model in long-term recommendations, and Section VIII concludes the paper by summarizing the findings and discussing the implications of the deep reinforcement learning approach for intelligent tutoring systems.

II. RELATED WORK

A. Recommender Systems

In Intelligent Tutoring Systems (ITS), the role of recommender systems is to furnish learners with personalized recommendations, aiding them in selecting suitable learning resources based on their learning progress and comprehension.

Traditional recommender systems, such as those based on Collaborative Filtering (CF) methods [6], Matrix Factorization (MF) methods [5], and neural network-based approaches [7], prove highly effective when recommending static content (e.g., textbooks, videos, exercises). However, these methods often rely on a substantial volume of historical interaction data and frequently grapple with the cold start problem when confronted with new students or content. Moreover, these approaches tend to overlook the dynamism and sequential nature of the learning process, wherein the learner's knowledge state gradually evolves over the course of learning.

In handling sequential data, recommender systems based on Recurrent Neural Networks (RNNs) have made strides. Hidasi et al. [8] employed RNNs for predicting students' learning paths, and other researchers have proposed several RNN-based enhancements [9]. Nevertheless, these methods primarily focus on short-term predictions, neglecting the long-term learning trajectories of students.

In Interactive Recommender Systems (IRS), student feedback is incorporated into the model. Such models can iteratively construct and optimize representations of students and content, thereby holding an advantage in addressing the cold start problem. Some researchers have utilized Multi-Arm Bandit (MAB) methods to build IRS, yet these methods primarily address the exploration-exploitation dilemma and do not explicitly optimize for long-term returns.

B. *Deep Reinforcement Learning*

Deep Reinforcement Learning (DRL) has made significant breakthroughs in many interactive systems, such as Atari games [10] and Go [11], but its application in recommender systems remains relatively limited. Wang et al. [12] proposed a DRL approach based on the Actor-Critic (AC) framework for page recommendations, a hybrid RL method that integrates value-based and policy-based modules. Li et al. [13] introduced a method based on Deep Q Network (DQN) for keyword prompt recommendations. Through this approach, they achieved effective recommendations in the absence of explicit student feedback. Sewak et al [14] presented a DRL-based interactive recommender system for news recommendations. This method, incorporating a memory network, better handles historical interaction data. Hernandez-Leal et al. [15] proposed a DRL method for personalized recommendations. Their approach learns the latent representation of students through deep neural networks and employs reinforcement learning for recommendation decisions. Ausin et al. [16] introduced a hybrid recommender system combining deep learning and reinforcement learning. Their method dynamically updates after each interaction and can make recommendations without student historical information. Abdelshiheed et al. [17] proposed a DRL-based recommender system to address multi-objective recommendation problems. Their method considers personalized student needs while taking into account business objectives. Koroveshi et al. [18] presented a DRL-based sequential recommendation method that predicts students' future behavior while considering both long-term and short-term interests. Jung et al. [19] introduced a DRL-based interactive recommender system to address cold start problems. Their method effectively recommends in situations where students lack historical interaction data.

The aforementioned research endeavors underscore the potential of deep reinforcement learning [22] in recommender systems, addressing a spectrum of challenges from history-based recommendations to tackling cold start problems and resolving multi-objective recommendation issues.

C. *Research Gaps and Motivation*

In summary, the primary challenges faced by recommender system research in intelligent tutoring systems include an overreliance on historical interaction data, neglecting the issue of students' long-term learning paths, and the difficulty in handling large action spaces. In the next section, we will introduce how we address these issues by proposing a new DRL-based recommender system framework to provide more effective personalized learning recommendations.

III. Overall Framework

Typically, reinforcement learning-based systems involve interaction between the environment and an intelligent agent [20]. During the training process, the parameters within the intelligent agent are optimized based on rewards generated from the continuous interaction between the environment and the agent [21]. More specifically, this interaction comprises two consecutive steps: 1) the agent selects and executes an action based on the environment's state; 2) the environment responds to the action performed by the agent and returns feedback and a reward.

In the recommender system scenario considered in this study, the environment consists of various students, and the intelligent agent is a recommendation model based on RNN. Actions correspond to generating Top-N recommendation lists for specific students, and feedback indicates whether the student accepts this recommendation list. The entire recommendation process is illustrated in Fig. 1. It can be observed that for each individual student, there is a corresponding recommendation agent. Notably, all recommendation agents share the same network parameters. This allocation strategy for recommendation agents has its advantages as it can prevent mutual interference from different students.

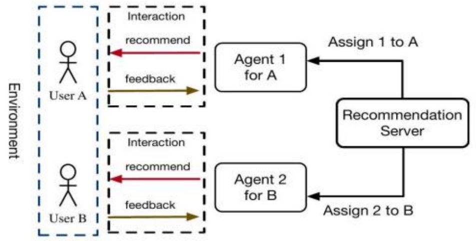

Fig. 1. The entire recommendation process.

IV. Environment and Interaction

The overall environment in this paper is constructed through offline datasets, such as the Secondary_school_curriculum, as online environments are not always feasible. Typically, offline datasets consist of a student-course rating matrix $\tilde{R} \in \mathbb{R}^{U*M}$, where U represents the students and M denotes the courses. The elements $\tilde{R}_{u,i}$ in $\tilde{R}$ signify the rating given by student u to course i. Specifically, the explicit rating matrix $\tilde{R}$ can be transformed into an implicit feedback matrix F, where the element $F_{u,i}$ indicates whether student u is associated with course i.

$$F_{u,i} = \begin{cases} 1 & \tilde{R}_{u,i} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Based on the implicit feedback matrix F, it is straightforward to derive the set of courses I_u that interest student u. The courses in the set are sorted by timestamp, and for each course $i \in I_u$, the value of $F_{u,i}$ must be 1.

To obtain feedback $f_{u,t}$, the student response function $\mathbb{V}(P_{u,t}^N, I_u)$ used in Eq. (1) can be defined as Eq. (2).

$$f_{u,t} = \mathbb{F}(P_{u,t}^N, I_u) = \begin{cases} 1 & P_{u,t}^N \cap I_u \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Here, $f_{u,t}$ being 1 indicates positive feedback, and 0 denotes negative feedback. $P_{u,t}^N \cap H_u \neq \emptyset$ signifies that $P_{u,t}^N$ successfully includes at least one course liked by the student. In practice, positive feedback can refer to student actions such as clicks or purchases. Similarly, negative feedback refers to students ignoring the recommended list or clicking on courses outside the recommended list. The system automatically provides feedback when the student performs any of these actions, without requiring the student to provide real-time feedback.

During the testing phase, the long-term recommendation performance of a student is the average result obtained over all steps in the corresponding interaction sequence. More intuitively, the overall performance of the recommender system can be evaluated using recall-based metrics, such as hit rate, and precision-based metrics, such as NDCG. These can be calculated using Eq. (3) and Eq. (7), respectively.

$$hit@N = \frac{\Sigma_u \frac{1}{|I_u|} \Sigma_{t=1}^{|I_u|} f_{u,t}}{\#user} \tag{3}$$

$$NDCG@N = \frac{\Sigma_u \frac{1}{|I_u|} \Sigma_{t=1}^{|I_u|} \frac{DCG@N(P_{u,t}^N)}{iDCG@N}}{\#user}. \tag{4}$$

$$DCG@N(P_{u,t}^N) = \sum_{i=1}^{|K|} \frac{h_i}{\log_2 1 + i}. \tag{5}$$

$$h_i = \begin{cases} 1 & P_{u,t,i}^N \text{ inI} \\ 0 & \text{otherwise.} \end{cases}. \tag{6}$$

$$iDCG@N = DCG@N\binom{N}{u,t} \tag{7}$$

where ${}_{u,t}^{N}$ represents the recommended sequence, which includes courses of interest to the student that have not been recommended previously.

V. Recommendation Agent

Within the recommendation agent, the recommendation process is viewed as a Markov Decision Process (MDP), providing a more suitable framework for recommender systems due to its consideration of the long-term impact of each recommendation and the corresponding expected values. Fig. 2 shows the diagram of the model.

Assuming $\pi(i \mid s_{u,t})$ represents the probability of recommending course i given the student's state $s_{u,t}$, the generation function $\mathbb{G}$ can be defined as follows:

$$P_{u,t}^N = \mathbb{G}(s_{u,t}, I) = \text{Top}_{i \in I}\, N(\pi(i \mid s_{u,t}) \times m_{u,i}^t) \tag{8}$$

where, $\pi(i \mid s_{u,t})$ is the recommendation probability of course i at time t, and $m_{u,i}^t$ is the element of course i in the masking vector m_u^t. The value of $m_{u,i}^t$ is 0 or 1, indicating whether the student has previously selected that course.

At time t, the recommendation probability $\pi(i \mid s_{u,t})$ for course i can be obtained as:

$$\pi(i \mid s_{u,t}) = \text{Softmax}(o_{u,t}^i) \tag{9}$$

where, $s_{u,t}$ is a obtained l-dimensional vector (which will be discussed later), and $o_{u,t}^i$ is the i-th element in the vector $o_{u,t}$ defined as:

$$o_{u,t} = \hat{\sigma}(W_s s_{u,t} + b_s). \tag{10}$$

where $\hat{\sigma}$ is the ReLU activation function, $W_s \in \mathbb{R}^{M \times l}$ and $b_s \in \mathbb{R}^M$ are the parameter matrix and bias, respectively. The softmax function is defined as:

$$\text{Softmax}(o_{u,t}^i) = \frac{\exp^{o_{u,t}^i}}{\sum_{k \in I} \exp^{o_{u,t}^k}} \tag{11}$$

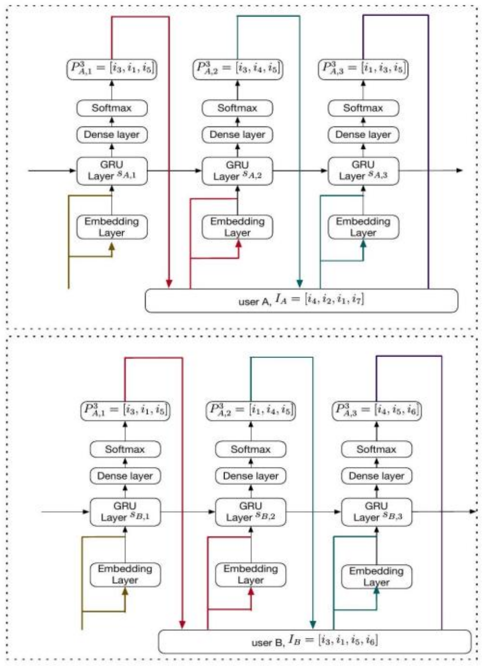
(a) Recommendations for A
$P_{A,1}^3 = [i_3, i_1, i_5]$
$P_{A,2}^3 = [i_3, i_4, i_5]$
$P_{A,3}^3 = [i_1, i_3, i_5]$
Softmax
Dense layer
GRU Layer $s_{A,1}$
GRU Layer $s_{A,2}$
GRU Layer $s_{A,3}$
Embedding Layer
user A, $I_A = [i_4, i_2, i_1, i_7]$
(b) Recommendations for B
$P_{A,1}^3 = [i_3, i_1, i_5]$
$P_{A,2}^3 = [i_1, i_4, i_5]$
$P_{A,3}^3 = [i_4, i_5, i_6]$
GRU Layer $s_{B,1}$
GRU Layer $s_{B,2}$
GRU Layer $s_{B,3}$
user B, $I_B = [i_3, i_1, i_5, i_6]$

Fig. 2. Model.

In the formula, $s_{u,t}$ is the state of student *u* at time *t*, obtained through the following state transition process:

$$s_{u,t} = \mathbb{T}\left(s_{u,t-1}, f_{u,t-1}, P_{u,t-1}^{N}\right) = S\left(s_{u,t-1}, \hat{a}_{u,t-1}\right) \tag{12}$$

where, $P_{u,t-1}^{N}$ is the recommended list, $f_{u,t-1}$ is the student's feedback, *S* is the internal state transition process, and $\hat{a}_{u,t-1}$ is the action used to represent $P_{u,t-1}^{N}$ and $f_{u,t-1}$ defined as:

$$\hat{a}_{u,t-1} = \underset{a \in A_{u,t-1}}{\operatorname{argmax}}\left(\pi\left(a \mid s_{u,t-1}\right) \times m_{u,i}^{t-1}\right). \tag{13}$$

where, $A_{u,t-1}$ is the auxiliary set for feedback in different situations, defined as:

$$A_{u,t-1} = \begin{cases} P_{u,t-1}^{N} \cap I_u, & f_{u,t-1} > 0 \\ P_{u,t-1}^{N}, & \text{otherwise.} \end{cases} \tag{14}$$

The internal state transition process $s_{u,t} = S\left(s_{u,t-1}, \hat{a}_{u,t-1}\right)$ is obtained through an RNN with Gated Recurrent Unit (GRU).

Firstly, $\hat{a}_{u,t-1}$ should be transformed into the input of RNN$x_{u,t}$ at time *t*. To more effectively incorporate feedback into RNN, the input $x_{u,t}$ is obtained through the following formula:

$$x_{u,t} = E\left(\hat{a}_{u,t-1}\right) = \begin{cases} \hat{e}\left(\hat{a}_{u,t}\right), & f_{u,t} > 0 \\ -\hat{e}\left(\hat{a}_{u,t}\right), & \text{otherwise.} \end{cases} \tag{15}$$

where, $\hat{e}\left(\hat{a}_{u,t}\right) \in \mathbb{R}^{\hat{l}}$ is the *l*-dimensional embedding of course $\hat{a}_{u,t}$ also a part that needs to be learned in the proposed model. Using *E*, positive and negative feedback can be clearly distinguished for a given $\hat{a}_{u,t}$.

According to the state transition of GRU, $s_{u,t} = S\left(s_{u,t-1}, \hat{a}_{u,t-1}\right)$ can be obtained as follows:

$$\begin{aligned} s_{u,t} = S\left(s_{u,t-1}, \hat{a}_{u,t-1}\right) &= \left(1 - Z\left(x_{u,t}, s_{u,t-1}\right)\right) \odot s_{u,t-1} \\ &+ Z\left(x_{u,t}, s_{u,t-1}\right) \odot \tilde{S}\left(x_{u,t}, s_{u,t-1}\right). \end{aligned} \tag{16}$$

where, $\odot$ represents the element-wise product, *Z* is the function of the update gate, $\tilde{S}$ is the function generating candidate states, obtained through the following formulas:

$$Z\left(x_{u,t}, s_{u,t-1}\right) = \sigma\left(W_z x_{u,t} + U_z s_{u,t-1}\right) \tag{17}$$

$$\tilde{S}\left(x_{u,t}, s_{u,t-1}\right) = \tanh\left(W x_{u,t} + U\left(j_{u,t} \odot s_{u,t-1}\right)\right) \tag{18}$$

where, σ is the sigmoid activation function, $W_z \in \mathbb{R}^{l \times \hat{l}}, U_z \in \mathbb{R}^{l \times l}, W \in \mathbb{R}^{l \times \hat{l}}$ and $U \in \mathbb{R}^{l \times l}$ are parameter matrices, and $j_{u,t}$ is the reset gate, obtained through the following formula:

$$j_{u,t} = \sigma\left(W_j x_{u,t} + U_j s_{u,t-1}\right) \tag{19}$$

where, $W_j \in \mathbb{R}^{l \times \hat{l}}$ and $U_j \in \mathbb{R}^{l \times l}$ are parameter matrices.

In the warm-start model, there is an additional component used to merge student historical information.

Assuming $\hat{I}_u$ is the set containing all historical courses related to student *u* (it should be noted that, $\hat{I}_u \cap I_u = \emptyset$, and if $i \in \hat{I}_u$, then $m_{u,i}^{0} = 0$), i.e., courses from the student's history cannot be selected during the interaction.

In particular, the latent vector h_u representing student *u*'s historical items is obtained through the following formula:

$$h_u = \tanh\left(\sum_{i \in \hat{I}_u} e(i)\right) \tag{20}$$

where, $e(i) \in \mathbb{R}^{\hat{l}}$ is another l-dimensional embedding defining course i, which also needs to be learned and is different from $\hat{e}_i$.

Additionally, to adaptively balance the effects of h_u and $s_{u,*}$ an External Memory Gated Recurrent Unit (EMGRU) is proposed, with detailed descriptions as follows.

Firstly, the state $s_{u,t}$, is obtained according to the formula, then, a new integrated state $\hat{s}_{u,t}$ used to generate course selection probabilities is obtained through the formula:

$$\hat{s}_{u,t} = d_{u,t} \odot s_{u,t} + \left(1 - d_{u,t}\right) \odot h_u \tag{21}$$

where, $d_{u,t}$ is the balance gate, which can control the impact of static h_u and dynamic $s_{u,t}$.

The balance gate $d_{u,t}$ can be obtained through the following formula:

$$d_{u,t} = \sigma\left(W_d h_u + U_d s_{u,t}\right) \tag{22}$$

where, $W_d \in \mathbb{R}^{l \times \hat{l}}$ and $U_d \in \mathbb{R}^{l \times l}$ are parameter matrices.

It is noteworthy that h_u or $\hat{s}_{u,t}$ does not affect the transition process $\mathbb{T}$ of state $s_{u,t}$ and h_u only affects the generation of course selection probabilities. In other words, the static and dynamic branches are independent of each other. In summary, the specific structure of the EMGRU unit is shown in Fig. 3.

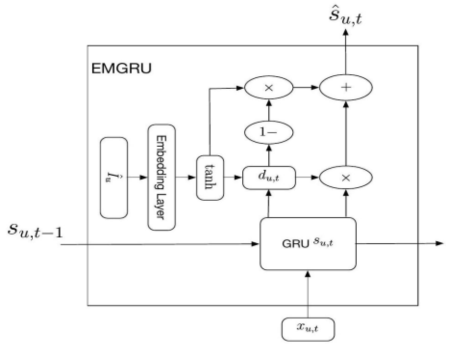

Fig. 3. EMGRU unit.

Finally, the selection probability $\pi\left(i \mid \hat{s}_{u,t}\right)$ is obtained through $\hat{s}_{u,t}$ rather than the state, $s_{u,t}$ and thus the course selection probability is defined as follows:

$$\pi\left(i \mid \hat{s}_{u,t}\right) = \operatorname{Softmax}\left(\hat{o}_{u,t}^{i}\right) \tag{23}$$

where, $\hat{o}_{u,t} = \hat{\sigma}\left(W_s \hat{s}_{u,t} + b_s\right)$.

The parameter set of the warm-start model is $\hat{\theta} = \tilde{\theta} \cup \{e(*), W_d, U_d\}$. The overall architecture of the warm-start model is shown in Fig. 4.

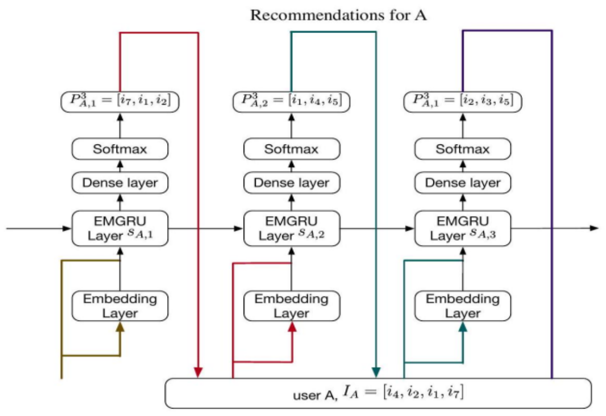
Recommendations for A
$P^3_{A,1} = [i_7, i_1, i_2]$
$P^3_{A,2} = [i_1, i_4, i_5]$
$P^3_{A,1} = [i_2, i_3, i_5]$
Softmax
Dense layer
EMGRU Layer $s_{A,1}$
EMGRU Layer $s_{A,2}$
EMGRU Layer $s_{A,3}$
Embedding Layer
user A, $I_A = [i_4, i_2, i_1, i_7]$

Fig. 4. The overall architecture of the warm-start model.

VI. Model Training

In this section, we will describe how to train the proposed model in the sequential interaction between the recommendation agent and the environment.

A. Reinforcement Learning

The goal of the learning algorithm in this section is to maximize the expected long-term recommendation reward, where θ is learned through the interaction process E_u for each student u. Specifically, E_u represents the complete interaction process obtained by the recommendation agent for student u under the current parameters.

Generally, an E_u interaction process includes the immediate reward $V_{u,t}$ at time t, state $s_{u,t}$, and action $\hat{a}_{u,t}$, defined as:

$$E_u = [s_{u,1}, \hat{a}_{u,1}, f_{u,1}, V_{u,1}, \ldots, s_{u,M}, \hat{a}_{u,M}, f_{u,M}, V_{u,M}] \tag{24}$$

where, $s_{u,t}$ is generated by Equation (19) $\hat{a}_{u,t}$ is obtained by Equation (16) and $f_{u,t}$. Thus, $V_{u,t}$ can be computed as:

$$V_{u,t} = \begin{cases} 1.0, & f_{u,t} > 0 \\ -0.2, & \text{otherwise.} \end{cases} \tag{25}$$

where 1.0 and -0.2 are values determined based on experience.

To maximize the expected cost J, each action $\hat{a}_{u,t}$ corresponds not only to an immediate reward $V_{u,t}$, but also to a long-term reward $R_{u,t}$, computed as follows:

$$R_{u,t} = \Sigma_{k=0}^{M-k} \gamma^k V_{u,t+k} \tag{26}$$

where $\gamma \in [0,1]$ is the discount factor. The objective function J is defined as:

$$J = \mathbb{E}_{s_{u,1}, a_{u,1}, \ldots}[R_{u,t}] \tag{27}$$

The parameter θ of the recommendation agent can be optimized using the gradient ascent method:

$$\theta = \theta + \eta \nabla_\theta J \tag{28}$$

where η is the learning rate, and the gradient $\nabla_\theta J(\theta)$ is given by:

$$\nabla_\theta J = \sum_{t=1}^{M} \gamma^{t-1} R_{u,t} \nabla_\theta \log \pi(\hat{a}_t \mid s_{u,t}) \tag{29}$$

In the standard practice of applying REINFORCE, the interaction process should be a complete E_u, which means the parameters should be updated after completing the interaction process for student u. However, the lengths of student interactions $I*$ can vary significantly. For example, $|I_A| = 20$, but $|I_B| = 200$. This causes large variances infor different students at the same time t. Moreover, due to the accumulation of excessive negative rewards, long-term interaction processes can hide positive results, making it challenging for the recommendation agent to obtain positive training samples. Therefore, a recommendation agent trained using traditional REINFORCE learning cannot achieve satisfactory recommendation performance.

To address this issue, this paper proposes to divide the original E into I_u/B sub-interaction processes and restart the reward accumulation at the beginning of each sub-interaction process. The learning process for both $\hat{\theta}$ and θ remains the same. The detailed processes of the learning process and the sub-interaction process generation are shown in Algorithm (1) and Algorithm (2), respectively. It should be noted that in the warm-start scenario, to ensure sufficient training data, the total length of the interaction process for each student u in the training phase is equal to $|I_u \cup \hat{I}_u|$. At the same time, the recommendation agent can choose courses from $\hat{I}_u$. However, the length of the interaction process is still equal to $|I_u|$, and during the testing phase, the recommendation agent cannot select courses from $\hat{I}_u$ for each student u.

B. Supervised Learning

Another approach to training the recommendation agent is to optimize it using supervised learning in a short-term prediction scenario and then apply it to a long-term testing environment.

To facilitate the transition of the recommendation agent from short-term to long-term prediction scenarios, the neural network architecture for the recommendation agent under supervised learning and reinforcement learning should be consistent. The only difference is that explicit labels need to be provided for supervised learning, and these labels are the actual course selections made by students at each time step.

Let I_u denote the actual sequence of courses chosen by student u over time. To maximize the accuracy of short-term predictions, this paper uses cross-entropy $\hat{J}$, defined as the cost function for supervised learning:

$$\hat{J} = -\Sigma_{i=1}^{B} \log\left(\pi(I_{u,i} \mid s_{u,i})\right) \tag{30}$$

where θ can be updated as follows:

$$\theta = \theta - \eta \nabla_\theta \hat{J} \tag{31}$$

Compared to the reinforcement learning approach, the supervised learning method is closer to traditional sessionbased RNNs, where each recommended step has a specific corresponding label for the training signal.

Furthermore, before applying reinforcement learning, fine-tuning the recommendation agent through supervised learning can be performed. This can help the reinforcement learning-based recommendation agent start from a relatively good policy rather than a random one, thus accelerating the convergence speed of reinforcement learning.

VII. Experiments

In this section, a substantial number of experiments were conducted to demonstrate the advantages of the proposed method in long-term recommendations and showcase the effectiveness of the core components of the proposed model.

A. Evaluation Datasets and Experimental Settings

Two offline real-world benchmarks, Secondary_school_curriculum 100K and Secondary_school_curriculum 1M, were used to evaluate the proposed model. Secondary_school_curriculum 100K contains 100,000 rating records about 943 students and 1682 courses. Secondary_school_curriculum 1M includes one million rating records about 6040 students and 3900 courses.

The proposed model was evaluated using the previously mentioned interaction environment to assess the performance of the proposed method and other methods. In the experiments, a 1-layer RNN and GRU with a hidden layer size of 100, matching the embedding size, are used. The size of B is set to 20, and γ is set to 0.9. For experiments on the 100K and 1M datasets, the proposed model is optimized using Adam with a learning rate of 0.005.

B. Benchmark

Pop: This algorithm always recommends the most popular items in the training set. While simple, it often serves as a powerful baseline.

Linear-UCB (L-UCB): A linear bandit algorithm, a widely used and mature multi-armed bandit algorithm. In this study, a context-independent bandit algorithm was used since content information was not considered. Course embeddings were obtained through matrix factorization (MF) methods.

ε-greedy: Similar to Linear UCB, but the balance between exploration and exploitation is adjusted by tuning the ε parameter. Course embeddings were also obtained through MF.

DQN (Deep Q-Network): A deep reinforcement learning algorithm based on value functions.

SARSA (State-Action-Reward-State-Action): A classic policy-based reinforcement learning algorithm commonly used as a benchmark.

Actor-Critic: A deep reinforcement learning algorithm that combines value function and policy methods. The architecture proposed in was used for comparison.

PPO (Proximal Policy Optimization): An advanced reinforcement learning algorithm that improves training stability by adding an "agent" constraint during policy updates.

Among these algorithms, Pop, Linear-UCB, and ε-greedy are traditional recommender systems or multi-armed bandit algorithms, while DQN, SARSA, Actor-Critic, and PPO are classical or advanced algorithms in the field of reinforcement learning.

C. Warm-start Model Comparative Experiment

In the experiment, a comparative study of warm-start models was conducted on the 100K, and the results are shown in Table I. In this table, $p = 10\%$ means that 10% of courses for each student u are retained as the warm-start historical set $\hat{I}_u$, and five different p values are considered in the experiment. Specifically, the historical data $\hat{I}_u$ for each student u in the test set was also used to train static models like BPR and NeuCF because these static models need to obtain corresponding student representations and are evaluated without using $\hat{I}_u$ during testing. However, the historical data of test students was not used to train other models.

TABLE I. NDCG@10 Comparison of the Warm-Start Models on 100K Dataset

	p=10%	p=30%	p=50%	p=70%	p=90%
Pop	2.90%	2.16%	1.65%	1.31%	1.48%
BPR	3.41%	3.25%	2.89%	2.74%	3.23%
NeuCF	3.51%	3.33%	3.02%	2.86%	3.35%
sRNN	8.54%	6.92%	5.45%	4.05%	3.66%
sl-cold	8.97%	6.76%	5.47%	3.96%	3.40%
rl-cold	14.93%	12.11%	8.88%	1.24%	3.69%
sl+rl-cold	15.54%	12.53%	9.38%	6.32%	3.71%
sl-warm	8.65%	6.79%	5.37%	4.73%	4.42%
rl-warm	13.81%	11.53%	8.37%	6.76%	4.44%
sl+rl-warm	14.34%	12.07%	9.64%	7.90%	6.18%

The proposed method's large-scale warm-start model significantly outperforms baseline models. With an increase in p, the warm-start model demonstrates significant improvements in HR and NDCG. These results suggest that incorporating historical data can enhance the model's performance if the historical data is sufficiently rich. It is noted that experiments with larger p values are more challenging than those with smaller p values because the total number of $I_u \cup \hat{I}_u$ is fixed. Specifically, larger p values result in a smaller correct candidate set I_u and a shorter reasoning process. Therefore, the proposed model's performance is relatively better at smaller p values. Additionally, training the recommendation agent through supervised learning generally improves HR and NDCG performance. Furthermore, this is particularly effective for the larger 1M dataset. Without supervised pre-training, the rl-warm model cannot even outperform sRNN at p=90%. Conversely, the gap between sRNN and sl+rl-warm is significant.

D. Long-Term Prediction Performance Comparison

To validate the performance of the proposed model in long-term recommendations, Fig. 5 presents the recommendation results of various comparative methods at different stages. The experimental results are obtained from selected students in the respective test sets. Specifically, the size of the selected student set I_u is above average because students with longer histories can more clearly reveal the performance of long-term recommendations. Additionally, the results for student u are divided into five different stages: [0%,20%), [20%,40%), [40%,60%), [60%,80%), and [80%,100%], where $[s\%, e\%)$ denotes the steps between $(s\% \times I_u)$ and $(e\% \times I_u)$ during the entire interaction period. Furthermore, e is used to represent the range [s%,e%) and provides the average results within that range.

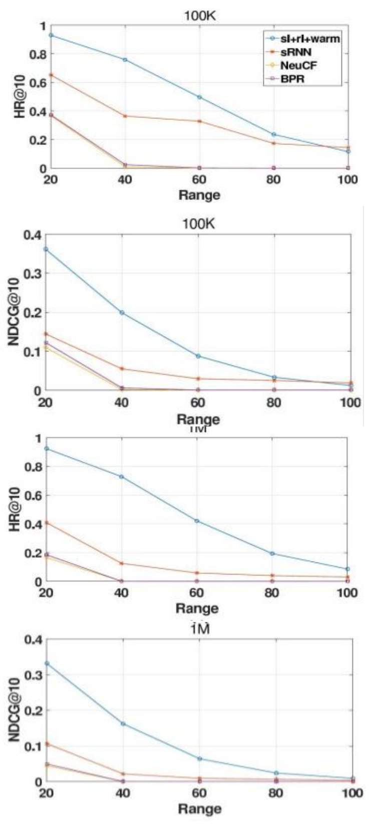

Fig. 5. The Recommendation performance at different stages. (a) HR@10 on 100 K; (b) NDCG@10 on 100 K; (c) HR@10 on 1M; (d)NDCG@10 on 1M.

As shown in Fig. 5, static models like NeuCF and BPR still achieve good hit rates in the first range [0%,20%). However, due to the almost unchanged recommendation results obtained by static methods, their performance sharply drops to 0% in the subsequent stages. Consequently, their overall HR and NDCG results are poor, as shown in Table II. On the other hand, sequential methods like sl+rl-warm and sRNN can achieve hits in all ranges, adapting their respective recommendation results. Compared to sRNN, the proposed sl+rl-warm model significantly surpasses sRNN in the ranges [0%, 20%) to [60%, 80%) because the proposed model already obtains a sufficient hit rate in the early stages. Since I_u has a fixed size, fewer courses can be hit in the last range [80%, 100%]. Therefore, the performance of sl+rl-warm is almost equivalent to that of RNN in the last range [80%, 100%]. All these results indicate that the proposed method can effectively adopt recommendation transfer in long-term recommendations.

TABLE II. TAB.6 HR@10 COMPARISON OF THE WARM-START MODELS ON 1M DATASET

	p=10%	p=30%	p=50%	p=70%	p=90%
Pop	6.83%	5.73%	5.04%	4.52%	4.10%
BPR	4.73%	4.73%	4.95%	4.99%	5.15%
NeuCF	6.27%	6.56%	6.65%	6.99%	7.78%
sRNN	16.55%	17.16%	17.27%	17.67%	18.52%
sl-cold	31.31%	26.99%	22.92%	17.82%	9.63%
rl-cold	41.65%	36.93%	32.07%	26.07%	14.73%
sl+rl-cold	45.15%	40.98%	33.24%	27.28%	15.15%
sl-warm	23.39%	23.23%	22.07%	16.89%	8.59%
rl-warm	42.23%	35.32%	29.53%	26.51%	16.09%
sl+rl-warm	45.06%	43.88%	37.14%	31.58%	21.20%

E. Impact of EMGRU

EMGRU is crucial in the warm-start model as it can adaptively adjust the dynamic RNN state and static historical representation to generate $\hat{s}_{u,t}$. To study the impact of this adaptive balance, the experiment considered four different settings combining RNN states and historical states: 1) using only RNN states (s); 2) using only historical representations (h); 3) the combination of historical representations and RNN states ($s + h$); 4) the combination of historical representations and RNN states with a balanced gate ($s + h$w/).

As shown in the Table III, the method with only h performs poorly because h_u is unchangeable throughout all interaction processes, and the fixed h_u cannot generate different recommendation results at different times. On the other hand, when the minimum setting is $p = 10\%$, the method with only s surpasses the combination methods $s + h$ and $s + h$w/ gate in terms of HR because the data volume of h_u is not sufficient. However, as p increases, the combination methods can outperform the method with only s. Moreover, adapting the combination of h and s with a gate generally has a better effect than mixing h and s with an equal constant. These results indicate that the effects between $s_{u,t}$ and h_u are non-fixed and should be adjusted according to the current situation.

TABLE III. NDCG@10 COMPARISON OF THE WARM-START MODELS ON 1M DATASET

	p=10%	p=30%	p=50%	p=70%	p=90%
Pop	2.03%	1.50%	1.18%	0.92%	0.81%
BPR	1.14%	1.05%	0.96%	0.84%	0.92%
NeuCF	1.74%	1.62%	1.37%	1.26%	1.53%
sRNN	3.26%	3.55%	3.37%	3.30%	3.79%
sl-cold	5.63%	4.37%	3.37%	2.53%	1.31%
rl-cold	8.74%	7.09%	5.41%	3.92%	2.28%
sl+rl-cold	11.30%	9.31%	6.78%	4.96%	2.50%
sl-warm	3.97%	3.51%	2.98%	2.08%	1.11%
rl-warm	9.02%	7.47%	5.44%	4.19%	2.71%
sl+rl-warm	11.62%	9.91%	7.02%	5.09%	3.61%

F. *Convergence Analysis*

To demonstrate the effectiveness of the settings in the REINFORCE algorithm, Fig. 6 illustrates the performance curves of different optimization algorithms in the warm-start (p=50%) scenario on the Secondary_school_curriculum 100K dataset.

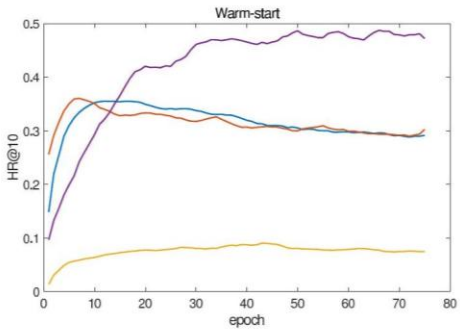

Fig. 6. The Learning curves of different algorithms in the warm-start.

Clearly, the performance of supervised learning methods with complete interaction processes (blue line) or the corresponding separated sub-interaction processes (red line) is similar, indicating that dividing the entire interaction process into several sub-interaction processes does not improve the accuracy of recommendations. Similarly, the performance of the basic REINFORCE algorithm (yellow line) is inferior and even significantly different from methods based on supervised learning. However, the performance of the special REINFORCE method (purple line) shows a significant improvement, as separating and restarting the reward accumulation for overly long interaction processes can obtain more useful self-generated training labels for reinforcement learning. Although this approach is simple, it is highly effective.

G. *Recommendation Behavior Analysis*

To analyze the recommendation behavior of the recommendation agent, the experiment provides the average results of the relevant popularity and Hit@10 for all test students on the Secondary_school_curriculum 100K dataset at each step t in the warm-start scenario, as shown in Fig. 7.

Thus, the recommendation agent in this study gradually evolves from widespread recommendations to personalized recommendations and can accurately hit courses within the specified range.

H. *Dynamic Recommendation Analysis*

To evaluate the effectiveness of the dynamic recommendation process, the experiment randomly selected several cases and provided the recommendation results of the proposed model on the dataset. For each case, a sequence of sequentially recommended courses is displayed for a specific student. It is important to note that, for ease of presentation, only one course is displayed in the Top-10 ranking list, i.e. $\hat{a}u, t$. Specifically, for each case from (a) to (c), the results from step 0 to step 14 are always shown. On the other hand, for each case from (d) to (e), the results of a randomly selected consecutive 15 steps within the specified range are displayed.

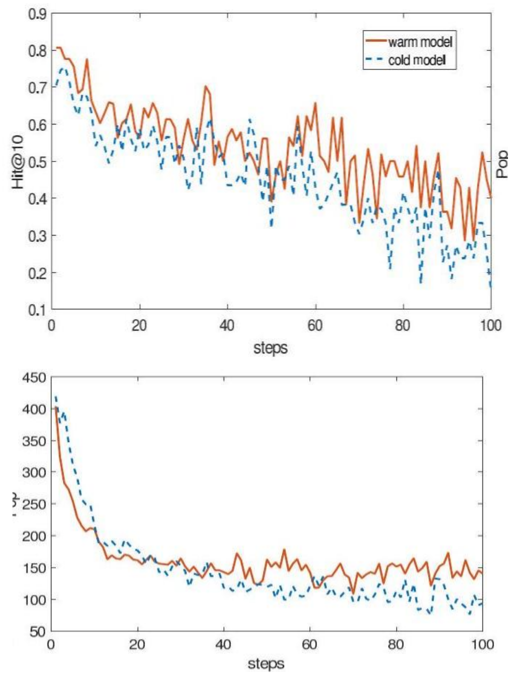

Fig. 7. Characteristics at different time steps in the warm-start scenario. (a) Hit@10; (b) The average results of the relevant popularity.

It can be observed that the proposed method can adaptively adjust recommendations based on past unsuccessful experiences. For example, in case (a), the first two recommendations are incorrect, but the subsequent four recommendations are correct. These results indicate that the proposed method can effectively change recommended courses based on previous feedback from students. Generally, the proposed model can dynamically update student states and modify recommendation results based on corresponding feedback.

In contrast, static methods cannot automatically adjust recommendation results and only make positive predictions at the beginning by correctly predicting recommendations, but they always make negative predictions. Therefore, compared to static methods, the recommendation approach in the proposed model is more effective.

VIII. Conclusion

In this paper, a new Top-N deep reinforcement learning recommender system is proposed to address the problem of long-term recommendations. In the proposed model, the recommendation process is considered as a Markov decision process. Thus, an RNN is used to simulate the sequential interaction between the agent (recommender system) and the environment (students). Moreover, the proposed model can be applied to warm-start scenarios. Additionally, the proposed model does not depend on any content information but only relies on the interaction between the environment and the agent, meaning it can effectively be applied in environments without sufficient content information. Experimental results show that, compared to traditional Top-N recommendation methods, the proposed method has better recommendation performance.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g." Avoid the stilted expression, "One of us (R. B. G.) thanks . . ." Instead, try "R. B. G. thanks."

REFERENCES

[1] Shen X, Liu S, Zhang C, et al. Intelligent material distribution and optimization in the assembly process of large offshore crane lifting equipment[J]. Computers & Industrial Engineering, 2021, 159: 107496.

[2] Xu M, Liu S, Shen H, et al. Process-oriented unstable state monitoring and strategy recommendation for burr suppression of weak rigid drilling system driven by digital twin[J]. The International Journal of Advanced Manufacturing Technology, 2022: 1-17.

[3] Fu T, Liu S, Li P. Intelligent smelting process, management system: Efficient and intelligent management strategy by incorporating large language model[J]. Frontiers of Engineering Management, 2024, 11(3): 396-412.

[4] Zheng H, Liu S, Zhang H, et al. Visual-triggered contextual guidance for lithium battery disassembly: A multi-modal event knowledge graph approach[J]. Journal of Engineering Design, 2024: 1-26.

[5] Nwana, H. S. (1990). Intelligent tutoring systems: An overview. Artificial Intelligence Review, 4(4), 251-277.

[6] Yazdani, M. (1986). Intelligent tutoring systems: An overview. Expert Systems, 3(3), 154-163.

[7] Alhabbash, M. I., Mahdi, A. O., & Naser, S. S. A. (2016). An intelligent tutoring system for teaching English grammar tenses.

[8] Sarrafzadeh, A., Alexander, S., Dadgostar, F., et al. (2008). "How do you know that I don't understand?" A look at the future of intelligent tutoring systems. Computers in Human Behavior, 24(4), 1342-1363.

[9] Hamed, M. A., & Naser, S. S. A. (2017). An intelligent tutoring system for teaching the seven characteristics of living things.

[10] Al-Bastami, B. G., & Naser, S. S. A. (2017). Design and development of an intelligent tutoring system for C#.

[11] Garnier, P., Viquerat, J., Rabault, J., et al. (2021). A review on deep reinforcement learning for fluid mechanics. Computers & Fluids, 225, 104973.

[12] Wang, H., Liu, N., Zhang, Y., et al. (2020). Deep reinforcement learning: A survey. Frontiers of Information Technology & Electronic Engineering, 21(12), 1726-1744.

[13] Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.

[14] Sewak, M. (2019). Deep reinforcement learning. Singapore: Springer Singapore.

[15] Hernandez-Leal, P., Kartal, B., & Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems, 33(6), 750-797.

[16] Ausin, M. S. (2019). Leveraging deep reinforcement learning for pedagogical policy induction in an intelligent tutoring system. In Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019).

[17] Abdelshiheed, M., Hostetter, J. W., Barnes, T., et al. (2023). Leveraging deep reinforcement learning for metacognitive interventions across intelligent tutoring systems. In International Conference on Artificial Intelligence in Education (pp. 291-303). Cham: Springer Nature Switzerland.

[18] Koroveshi, J., & Ktona, A. (2021). Training an intelligent tutoring system using reinforcement learning. International Journal of Computer Science and Information Security (IJCSIS), 19(3).

[19] Jung, G. (2023). Exploring batch deep reinforcement learning and multi-task learning across intelligent tutoring systems: Lessons learned.

[20] Paduraru, C., Paduraru, M., & Iordache, S. (2022). Using deep reinforcement learning to build intelligent tutoring systems.

[21] Milani, S., Fan, Z., Gulati, S., et al. (2020). Intelligent tutoring strategies for students with autism spectrum disorder: A reinforcement learning approach. In The 2020 CMU Symposium on Artificial Intelligence and Social Good.

[22] Subramanian, J., & Mostow, J. (2021). Deep reinforcement learning to simulate, train, and evaluate instructional sequencing policies. Spotlight presentation at the Reinforcement Learning for Education workshop at the Educational Data Mining 2021 conference.