Journal of Network Intelligence ©2025 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **10**, Number **3**, August **2025**

Machine Learning-based Online Learning Resource Recommendation System Under Distributed Computing Paradigm

Jiezhi Chen[1,*], Wenbiao Li[2]

[1]Guangdong Lingnan Institute of Technology, Guangdong, Guangzhou 510663, China
cjz@lnc.edu.cn

[2]College of Information Technology and Engineering, St. Paul University Philippines
Tuguegarao City, Cagayan Valley 3500, Philippines
liwenbiao@spup.edu.ph

*Corresponding author: Jiezhi Chen

Received December 19, 2023, revised November 12, 2024, accepted May 27, 2025.

Abstract. *The rapid development of internet technologies and the exponential growth in data scale have led to the ubiquitous challenge of information overload. In response, recommendation systems (RSs) have emerged as effective solutions. However, with the emergence of big data, traditional RSs face challenges in storage, processing, and the escalating demand for personalized user services. Addressing these issues, this paper proposes an RS based on the Hadoop cloud platform, integrating machine learning-based online learning resource recommendations (OLRR) into a cloud computing environment. This innovative system not only amplifies the storage and processing capacities for large-scale data within the RS but also adeptly addresses the evolving requirements for personalized services. A hierarchical reinforcement learning (RL) model based on clustering is introduced for personalized OLRR. This model aggregates basic recommendation models, clustering models, and hierarchical RL models to mitigate data noise influence and improve the performance of RS in the face of data sparsity. Nevertheless, precision challenges arise in low-level tasks when modifying sequences within the clustering-based hierarchical RL model. To overcome this, Q-learning is employed to optimize the process of removing elements in low-level tasks. The user's historical interactions are stored in a memory network, enabling a more detailed categorization of the user's current behavioral patterns into short-term, long-term, as well as global preferences. An attention mechanism is introduced to generate corresponding user memory vectors. Experimental data indicates that the incorporation of clustering in the hierarchical RL model enhances the handling of sparse data, and the addition of Q-learning in this model effectively improves accuracy. Furthermore, the implementation of RL algorithms on the Hadoop cloud platform demonstrates that distributed computing can enhance the performance of RS in handling large-scale data. We believe that the proposed RS system has real-world application potential by enhancing learners' learning effectiveness and experiences through personalized and accurate recommendations, catering to diverse learning needs, and promoting the intelligent development of the education and training field.*

Keywords: Recommendation System; Clustering; Online Learning Resource Recommendation; Attention Mechanism; Hadoop; Distributed Computing

1. **Introduction.** The contemporary proliferation of internet technologies has led to an exponential surge in the quantity and diversity of online learning resources (OLRs) [1]. Schools offer resources such as library collections and online courses, The former includes

textbooks, reference books, journals, papers, etc., designed to support students' academic research and subject learning, while the latter encompass online courses, lectures, instructional videos, etc., offering a flexible learning approach, enabling students to study according to their own schedules. The internet provides massive open online courses (MOOCs), digital libraries, and instructional videos. In addtion, learning videos on video platforms involve learning-related videos on platforms like YouTube, Coursera, covering course lectures, experimental demonstrations, and more. The advent of the big data era, characterized by the daily generation of massive amounts of information, has given rise to the challenge of information overload (IO) [2]. In this context, users face increasing difficulty in navigating the vast array of OLRs to find materials tailored to their specific needs. While solutions like classification directories and search engines have mitigated IO issue, the demand for personalized services has rendered finding suitable OLRs increasingly challenging [3]. The digital library, serving as a pivotal means to facilitate users in acquiring specialized knowledge and elevating their professional proficiency, has garnered considerable attention across numerous universities. Despite digital libraries being more user-friendly and manageable than traditional counterparts, the formidable challenge for users lies in extracting relevant OLRs from extensive collections [4].

The emergence of recommendation systems (RSs) addresses the limitations of traditional approaches, presenting a more sophisticated solution that intelligently and proactively helps users efficiently discover information aligned with their interests and preferences [5]. RSs analyze user preferences to provide personalized suggestions, matching user preferences with item features to recommend the most likely selections. An effective RS saves users time in extracting valuable information from an abundance of data by leveraging historical interaction records to extract user preferences, such as major and interests.

Traditional RSs employ methods such as content-based, collaborative filtering (CF), association rule-based, utility-based, and knowledge-based recommendations. Yet, these approaches demonstrate rigid recommendation processes, disregarding the fluid evolution of user preferences. Moreover, the majority of RSs fixate on optimizing immediate benefits, dismissing prospective long-term advantages, and tend to suggest popular items, lacking the ability to explore less frequented or niche options [6]. Deep reinforcement learning (RL)-based RSs attracted increasing attention due to their flexible recommendation strategies and consideration of users' future long-term interaction experiences. These systems utilize a Markov decision process (MDP) to simulate the interaction between the RS and users. The environment's state is defined as user preferences, actions represent items in the RS, and the reward value is determined by user feedback. The RS suggests items based on user preferences, user feedback induces changes in preferences, and the recommendation agent adjusts suggestions accordingly [7]. The advantage of RL in learning resource recommendation lies in its flexible recommendation strategies, consideration of users' future long-term interaction experiences, and modeling of dynamic changes in user preferences, garnering widespread attention from researchers. By simulating the interaction process between the RSs and users, it can dynamically adjust recommendations in real-time based on user feedback, providing more personalized and long-term beneficial learning resource recommendations. Despite existing works on DRL-based RSs, two major challenges persist. First, current approaches struggle to capture users' transient preferences, often represented by atypical interactions with low frequency in historical records. Second, existing methods oversimplify the capture of user sequence preferences, treating long-term and short-term preferences equally and overlooking potential differences in their contributions during user-item interactions.

The next generation of RSs must efficiently provide high-quality recommendations even in the face of vast data. The Hadoop cloud computing (CC) framework emerges as a dependable, streamlined, and expandable open-sourced distributed framework primarily used for the storage and distributed computing processing of massive big data [8]. Leveraging the capabilities of R language for data analysis, this paper proposes the establishment of the Hadoop CC platform, combining the strengths of Hadoop and R, to study and design RSs. Utilizing Hadoop as the implementation platform for RSs addresses storage challenges through HDFS's dynamic storage capabilities and enhances computation speed through MapReduce parallel computing, thereby significantly contributing to overcoming the limitations of existing recommendation technologies.

1.1. **Related work.** The research panorama on RSs encompasses early CF-based techniques and other traditional recommendation methods, transitioning to recent advancements employing machine learning (ML). On account of the simplicity and high accuracy of CF, numerous studies have sought to enhance its principles. Konstas et al. [9] proposed a CF-based RS adept at adapting to individual user profile needs, employing a generic framework with restart-enhanced random walks to offer a representation of social networks that is both more organic and impactful. Choi et al. [10] designed a hybrid online-product RS, merging rating-based CF algorithms with sequential pattern analysis. Yang et al. [11] introduced a book RS based on the analysis of book inquiry history, proposing a framework for analyzing book inquiry history and providing recommendations for book acquisitions. Sohail et al. [12] devised an OWA-based book recommendation ranking method, using ordered weighted aggregation to ease the complexity of providing tailored recommendations to a large user community by aggregating rankings from several top universities. While these traditional RSs have proven effective on certain datasets, they exhibit limitations, such as a tendency to recommend popular items and poor exploratory capabilities. Additionally, they struggle in scenarios of data sparsity. Hence, with the widespread adoption of ML models, an increasing number of ML-based approaches are being applied to recommendation systems.

The utilization of ML methods in RSs has addressed numerous shortcomings associated with traditional approaches. For instance, employing feature classification methods from ML to categorize books can alleviate the issue of recommending overly popular items prevalent in CF-based methods. The current landscape of DRL-based recommendation algorithms can be categorized into three types based on the training strategy: value-based, policy-based, and Actor-Critic structured algorithms. Zhou et al. [13] utilized a knowledge graph-enhanced DRL approach to obtain item embeddings and merge item representations from a user's historical records, training the recommendation strategy through value-based DRL framework. Zou et al. [14] presented a value-based DRL model for multi-behavior sequence recommendation tasks, incorporating multiple recurrent neural networks (RNNs) to encode various user behavior sequences and constructing a simulator to simulate user feedback on non-interacted items. Gao et al. [15] utilized convolutional neural networks (CNNs) to encode positive feedback items, generating embeddings, and employed a Generative Adversarial Network (GAN) to obtain negatively correlated feedback vectors. The final user state is derived from the fusion of positive and negative feedback vectors. Policy-based DRL recommendation algorithms incorporate a policy network as the recommendation strategy. Chen et al. [16] utilized a policy network with RNN to fuse user browsing history to output a probability distribution. The model was trained using an off-policy DRL algorithm. Chen et al. [17] addressed the challenge of large action spaces through a balanced hierarchical clustering tree. Non-leaf nodes represent policy networks, and leaves represent items, utilizing policy-based

DRL to train the policy networks. Actor-Critic structured DRL algorithms combine both value-based and policy-based approaches. Chen et al. [18] proposed a Knowledge-Guided DRL framework, KGRL, which integrates DRL and knowledge graphs in interactive recommendation. An actor-critic framework was employed with a local knowledge network and attention mechanism to excel in dynamic recommendation scenarios. Chen et al. [19] proposed a Keywords-enhanced DRL model KDRL to address challenges in tourism recommendation. By incorporating informative keywords and dynamic user intentions, it outperformed static methods, enhancing travel recommendation effectiveness. Hou et al. [20] proposed DRR-Max, a novel recommendation model based on DRL, to Address issues of static systems. it incorporates a state generation module for long-term preferences and uses the Actor-Critical algorithm for real-time recommendations. The above models demonstrated the diverse applications of DRL in RSs, spanning different architectures and methodologies. These models not only optimize user engagement but also provide interpretable and personalized recommendations in various domains, showcasing the versatility and effectiveness of ML-based approaches in enhancing RS performance.

1.2. **Contribution.** This paper builds upon existing DRL methodologies, specifically exploring the integration of DRL into RSs to enhance optimization outcomes. A novel approach is proposed, introducing a hierarchical DRL method tailored to user short and long-term interests. The methodology begins by training historical user sequences using a foundational model to extract features representing the characteristics of items and users. Subsequently, these features undergo clustering through a clustering algorithm, strategically aimed at mitigating the sparsity of item-related data for improved handling by the hierarchical DRL model. The introduced Q-learning mechanism within the clustering-based DRL model enhances the overall performance of the RS through iterative Q-learning processes. Lastly, Hadoop is employed as the implementation platform. This choice serves a dual purpose: firstly, dynamic storage space capabilities provided by HDFS address storage challenges posed by vast datasets, and secondly, the MapReduce model facilitates distributed parallel computation, augmenting the system's processing capabilities. This research contributes to the advancement of RSs by integrating hierarchical DRL with a focus on user temporal preferences, resulting in a more refined and effective model.

2. **Research background.**

2.1. **Hadoop.** Hadoop, an open-source parallel software framework, operates seamlessly on large-scale distributed clusters [21]. This framework efficiently organizes computing resources, establishing a distributed computing platform that harnesses the storage and processing capabilities of extensive clusters for parallelized distributed handling of big data. Comprising various sub-projects, such as HBase, Hive, and Sqoop, among others, collectively known as the Hadoop ecosystem. The framework's compositional structure is depicted in Figure 1.

Within Figure 1, Hadoop Common stands as the foundational and bottommost module in the Hadoop system, providing diverse tools for other sub-projects. HDFS, another integral component, not only exhibits excellent fault tolerance but also boasts significant data access throughput. These attributes make it particularly advantageous for deployment on large, cost-effective clusters. MapReduce, another core element of Hadoop, represents an efficient distributed computing model employing a "divide and conquer" approach to process extensive datasets. This involves the distributed processing of data through Map and Reduce functions. HBase, an open-source database within the Hadoop system, is distinguished by its distributed and column-oriented nature. In HBase, large-scale data

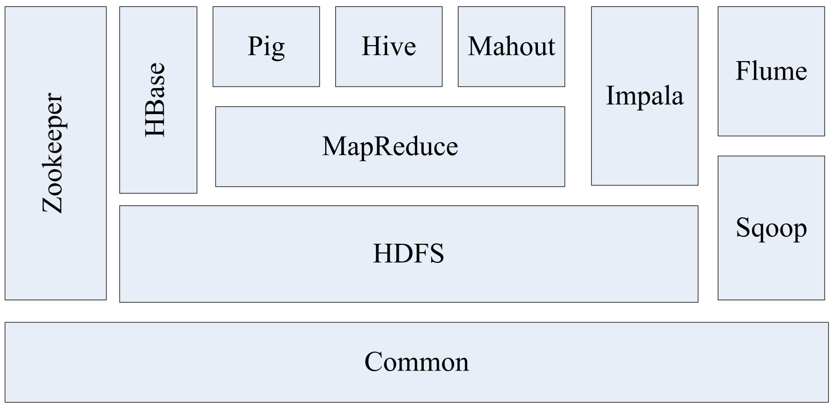

Figure 1. Architectural Composition of the Hadoop Ecosystem

reading and writing operations are more real-time and random, emphasizing its suitability for specific applications. Hive, a data warehousing tool built on the Hadoop system, facilitates users with a simplified SQL-like query language, HQL, for data retrieval, making it particularly suitable for statistical analysis. Beyond the aforementioned sub-projects, each component within the Hadoop ecosystem possesses unique functional characteristics, collectively contributing to the robust and feature-rich Hadoop ecosystem.

2.2. **Basic recommendation algorithms.** In the realm of learning-based collaborative RS models, notable approaches include Sparse Linear Methods (SLIM), Factored Item Similarity Models (FISM), and Neural Attentive Item Similarity Model (NAIS). Among these, SLIM [22] stands out by employing a predictive similarity matrix to model item-item relationships, demonstrating superior recommendation performance compared to conventional item-based CF methods. The fundamental premise of SLIM involves learning a similarity matrix between items from the historical interaction records of users. Subsequently, user interest in a target item is determined by evaluating the correlation between the target item and items with which there has been past interaction. This approach represents an enhancement over traditional item-based CF methods, showcasing improved recommendation efficacy. The representations and mechanisms of SLIM can be expressed as:

$$\hat{y}_{u,i} = \sum_{j \in I^+ \backslash \{i\}} s_{i,j} \tag{1}$$

where $\hat{y}_{u,i}$ represents the predicted value. I^+ represents the set of items associated with user u. $\backslash\{i\}$ signifies the set excluding item i itself, thereby avoiding the computation of similarity between item i and itself. $s_{i,j}$ in the item similarity matrix S indicates the element in the i-th row and j-th column, representing the similarity between target item i and associated item j. A higher similarity indicates a greater probability that user u is interested in the target item i.

Despite SLIM achieving good recommendation results, it faces the challenge of featuring a large parameter count. In practice, the number of users and items is enormous, and there are sparse interaction records between users and most items, meaning interactions between users and items are highly sparse. FISM [23] addresses this issue by decomposing the item similarity matrix S in SLIM, consequently decreasing the parameter count. The basic idea is to decompose S into the product of two independent low-rank matrices:

$$S = PQ^T \tag{2}$$

where P and Q are matrices of dimensions $|I| \times k$, and $k \leq |I|$. By performing further normalization, the expression for FISM can be yielded as follows [24]:

$$\hat{y}_{u,i} = \frac{1}{|I^+ \setminus \{i\}|^\alpha} \sum_{j \in I^+ \setminus \{i\}} \langle p_j, q_i \rangle \tag{3}$$

where α is the hyperparameter controlling the effect of normalization. p_j and q_i represent the latent vectors for historical item j and target item i, respectively. In comparison to SLIM, FISM addresses the issue of parameterization on sparse datasets, enhancing recommendation performance. However, it assumes that all historical interacted items contribute equally to predictions. In practical recommendation scenarios, different historical interacted items may have varying impacts on predictions. Therefore, when modeling long-term user preferences, the equal treatment of all historical interacted items can constrain its recommendation capabilities.

NAIS [25], building upon FISM, introduces a joint attentional mechanism to distinguish the contribution levels of past interacted items to predictions, achieving improved effectiveness. When contrasting NAIS with FISM, NAIS exhibits advantages in the following aspects:

(1) **Model Structure:** NAIS adopts a neural network structure, providing enhanced nonlinear modeling capabilities. With the incorporation of an attention mechanism, it dynamically learns the relationships between users and items, making the model more adaptable to diverse shifts in user interests. In comparison, FISM utilizes traditional factorization methods, which may exhibit inferior performance in handling complex user behavior patterns.

(2) **Attention Mechanism:** NAIS introduces an attention mechanism, allowing the model to focus more on items within the user's historical behavior sequence that have significant impact, effectively capturing the evolution of user interests. This enables NAIS to better cater to individualized user preferences, enhancing recommendation accuracy.

(3) **Sequence Modeling:** NAIS, when considering user interests, places greater emphasis on modeling the user's behavior sequence rather than solely focusing on static features. This enables NAIS to adeptly handle the temporal and dynamic aspects of user behavior, making it suitable for recommendation scenarios that require consideration of user historical behavior sequences.

The model expression for NAIS is as follows:

$$\hat{y}_{u,i} = \sum_{j \in I^+ \setminus \{i\}} \langle a_{i,j} p_j, q_i \rangle \tag{4}$$

where $\hat{y}_{u,i}$ represents the predicted value. I^+ denotes the set of items interacted by user u. q_i and p_j represent the latent vectors for the target item i and historical item j, respectively. Additionally, $a_{i,j}$ signifies the attention weight contributed by historical item j when predicting user preference for the target item i. This weight is calculated by a multilayer perceptron (MLP) and processed through the softmax function.

3. Reinforcement learning-based online learning resource recommendation model.

3.1. Overall model design.

Conventional RSs often exhibit limited diversity in learning resources recommendations. To address this issue, this paper introduces an novel DRL-based RS framework, as illustrated in Figure 2. The system involves the analysis and preprocessing of user data to derive feature data, initial rating collection from users, and the analysis of user behavior and interests. Subsequently, an algorithmic model based on deep Q-learning is established and reinforced through training to acquire the feature data. The model training encompasses both offline and online aspects. Offline model training is conducted through a hybrid DRL model and exploration strategy, and the trained model is employed for online recommendation, generating recommendations that are presented to users via an interactive interface. Upon receiving recommendations, users provide feedback actions. If a user engages in resouce learning or provides ratings, the system collects this behavioral data, transfers it to the data layer, and initiates a new iteration. User-generated behavioral data serves as new samples for model training, facilitating dynamic model updates and online learning objectives.

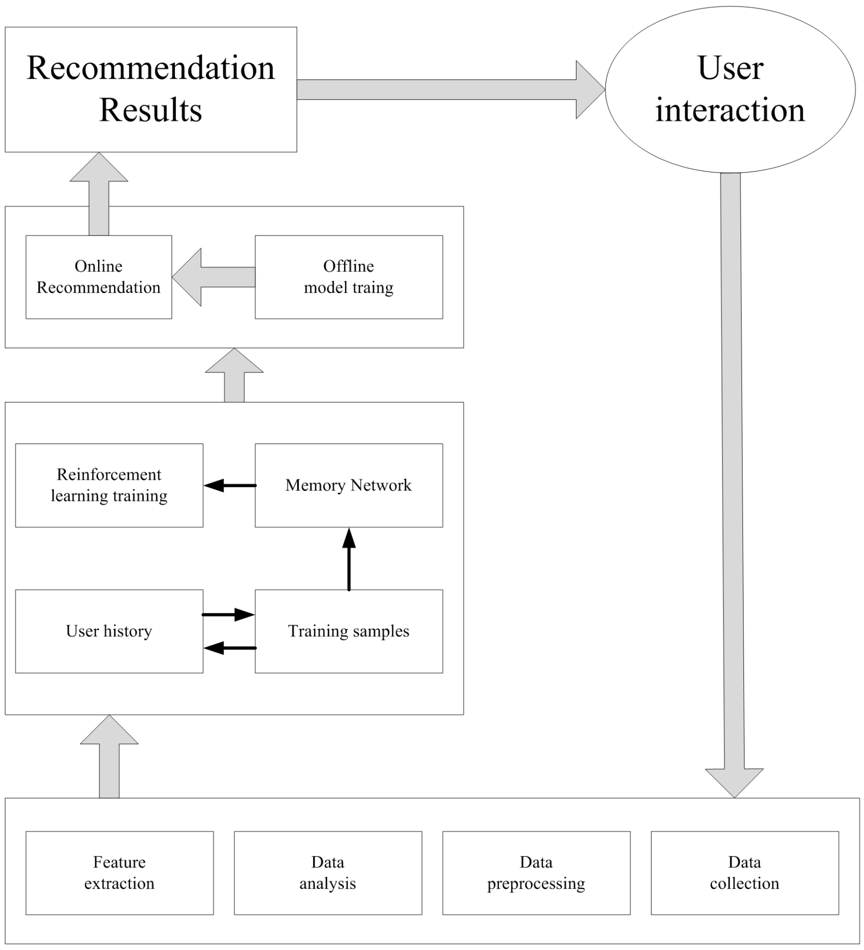

Figure 2. Architecture of the learning resouce commendation model

3.2. **OLR recommendation algorithm incorporating user Preferences.** A ORL recommendation algorithm based on hierarchical DRL networks and Q-learning is introduced. The fundamental concept revolves around leveraging the capabilities of clustering-based DRL techniques to eliminate noise that might mislead interactive RSs. The integration of Q-learning with low-level tasks in the hierarchical DRL model aims to enhance the accuracy of low-level tasks within the hierarchical DRL model. Figure 3 illustrates the basic architecture of the model, consisting of three components: the foundational recommendation model, sequence modification model, and clustering model. Specifically, the foundational recommendation model aims to provide the basic framework for OLR recommendations. This model utilizes attention-based neural networks to model user and item preferences. The sequence modification model serves to enhance the filtering process, eliminating potential noise that could misguide the basic RS. The clustering model, by clustering features of all resources, mitigates data sparsity by grouping learned resource features from the foundational model, thereby reducing the impact of data sparsity on the hierarchical reinforcement learning model.

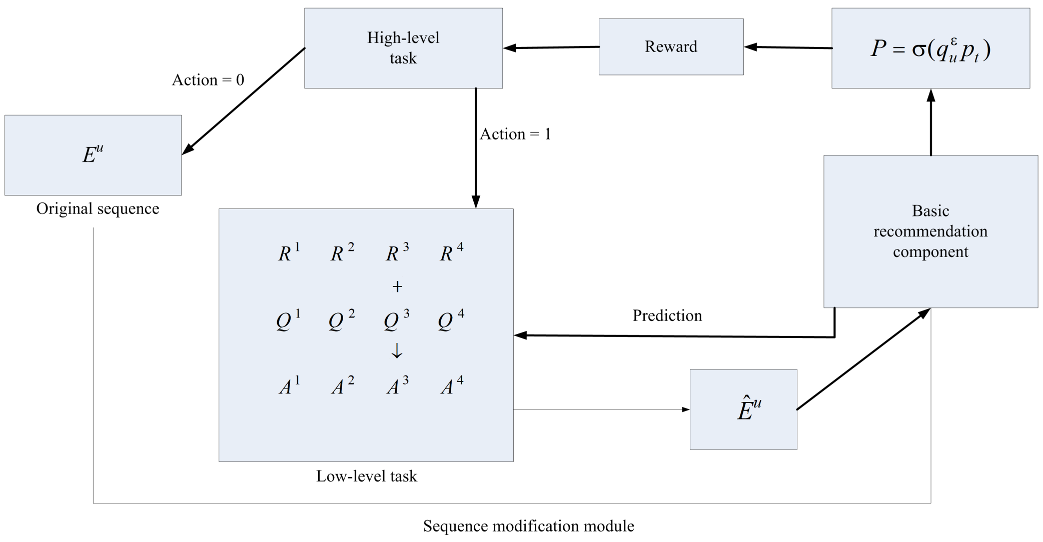

Figure 3. Architecture of the DRL Algorithm

3.2.1. *Problem Definition.* Let $U = \{u_1, u_2, ..., u_n\}$ denotes the set of users, and $T = \{t_1, t_2, ..., t_m\}$ represents the set of OLR items, n is the quantity of users and m is the quantity of items. For each u, given their historical sequence $E_u = (t_u^1, ..., t_u^j, ..., t_u^\epsilon)$, where ϵ denotes the time of the item in the historical record, the goal is to predict subsequent item $t_u^{\epsilon+1}$ that the user is most interested in.

3.2.2. *Components of the basic recommendation model.* The NAIS algorithm is employed as the basic recommendation model in the proposed framework. The NAIS algorithm adeptly tackles the learning problem through item-wise CF. It portrays each item by constructing a feature vector, capturing the similarity between two items through the inner product of their respective feature vectors. Consequently, a sequence can be articulated as a collection of these feature vectors, and the prediction of outcomes becomes achievable by leveraging the features associated with each individual item. Expanding on this approach, NAIS incorporates an attention network to discern the varying significance of items within the sequence. This enhances the algorithm's ability to dynamically learn and adapt to the varying importance of items as they appear in the sequence.

To illustrate user tendencies based on their learning sequence E_u, each item viewed by a user is represented as a real-valued low-dimensional feature vector p_u. Consequently, the borrowing sequence of each user can be represented as $p_u^1, ..., p_u^\epsilon$. The feature representation of the target item b_i is given as p_i. Let the user's viewing records as q_u, the probability of recommending an item to the user can be calculated as:

$$P = \sigma(q_\epsilon p) \tag{5}$$

3.2.3. *Memory network.* The user's historical interaction information is stored in an external component called the Memory Network. The specific construction of the Memory Network is outlined below:

Let $S = \{s_{11}, s_{12}, ..., s_{ij}\}$ represents the interaction matrix between users and items, where s_{ij} indicates the rating given by user u_i to item t_j, with different scores reflecting the user's preference for the respective item. If a user has not rated a particular item, the score is 0. For each user, there exists a set $T_u = \{t_u^1, t_u^2, ..., t_u^{k-1}, t_u^k\}$, where k is the quantity of items the user has interacted with, and this set is arranged chronologically based on the time of user ratings.

At each time point ϵ, the historical interactions T_u of each user u before this moment are stored in their corresponding Memory Network. Formally, $M_u = \{m_u^1, m_u^2, ..., m_u^{\epsilon-2}, m_u^{\epsilon-1}\}$ represents the user's behavior, categorized into short-term, long-term, and global preferences.

1) Short-term preferences: When user behavior is influenced by the last interaction, i.e., short-term preferences, attention weights are computed for all items in the user interaction set with the most recent interaction $m_u^{\epsilon-1}$. The calculation of the weights is depicted as:

$$\begin{cases} w_n = m_{\varepsilon-1}^u \times (m_n^u)^{\mathrm{T}} \\ z_n = \dfrac{\exp(w_n)}{\sum_j \exp(w_j)} \end{cases} \tag{6}$$

where $n, j \in [1, \epsilon - 2]$. w_n represents the product of the vector of the most recent interaction item and each vector in the Memory Network, resulting in a one-dimensional value. The term exp() denotes the exponential function with e as the base. z_n signifies the weight of the n-th item vector in this Memory Network. After obtaining the weight values, the calculation of the weight vector in the current state is determined. The computation of the weight vector through the attention mechanism is illustrated as:

$$A_\epsilon = \sum_j (z_j \cdot m_u^j) \tag{7}$$

where A_ϵ represents the attention vector for the preceding $(\epsilon - 2)$-th items, denoting the user's memory vector.

2) Long-term preferences: In cases where user behavior is influenced by a specific previous behavior, i.e., long-term preferences, the process involves first to identify the item m_u with the highest attention weight concerning the most recent behavior. Then, the attention weight is computed using m_u and the user's historical interaction vector. Finally, the user's memory vector is calculated.

3) Global preferences: When user behavior is not directly related to historical interactions (indicating a global preference), attention weight calculation involves both the user vector and the vector of historical interactions. The user's memory vector is then computed.

For writing into the user's memory network, a first-in-first-out (FIFO) strategy is employed for updates. Each user's memory network is set to a fixed length, with the initial two interactions stored. Training begins from the third interaction onward. If the number of memories for a user is less than the network's capacity, the most recent interaction is directly written into the memory network. Otherwise, the earliest user interaction memory is removed, and the most recent interaction is added.

3.2.4. *Clustering Component.* Given the sparsity issue inherent in digital OLR data, the way features are distributed in the data is notably dispersed. The hierarchical tasks of DRL often eliminate extensive data, rendering the foundational recommendation component incapable of providing accurate recommendations. Consequently, following the pre-training of the basic recommendation component, a clustering component is introduced to enhance the stability of the features utilized in the sequence modification component.

Upon training the foundational model, the model acquires features for each item. Given that the model's objective is to facilitate the sequence modification component in accurately identifying noise within the OLR data rather than discarding all elements from the history sequence, it becomes imperative to cluster the features of the items. Subsequent to clustering, the feature p for each item transforms into the feature p_c corresponding to the centroid of that cluster. In other words, post-clustering, the feature of each item becomes the centroid feature p_c for its respective cluster. The clustered history sequence E_u is denoted as E'_u. The sequence modification component will be trained using these clustered features, effectively eliminating noise from the history sequence.

3.2.5. *Deep Q-learning.* The Q-Learning algorithm is a highly representative model-independent algorithm in DRL [26]. As a model-agnostic DRL algorithm, Q-Learning determines an optimal policy π^* by learning the optimal action-value function through policy iteration. Without acquiring knowledge of the environmental model, Q-Learning directly optimizes an iteratively computable Q-function. Each state-action pair corresponds to a relevant Q-value, and the learning of the Q-function is accomplished through the iteration of Q-values. The initial Q-values can be arbitrarily assigned depending on the environment. The updating rule for Q-values can be calculated as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{8}$$

where s represents the current environmental state, a represents the action in that state, and $Q(s,a)$ is the value function for the state-action pair. α $(0 \le \alpha < 1)$ represents learning rate. γ $(0 \le \gamma < 1)$ denotes discount factor. r is the reward for the action in the given state. And s', a' represent the new state and the action in the new state, respectively [27].

After pre-training the base model and clustering features with the clustering component, the model obtains the base recommendation model G and the user's historical sequence $x_{1:\epsilon}$, where all items are represented as a set of features. As the base recommendation model has already transformed the input sequence into features, the proposed model directly employs the sequence features from the base model as the state s_ϵ in the DRL of Q-learning. On this basis, the Q-values for the sequence are computed. A fully connected layer is utilized as the solution for calculating Q-values:

$$Q(s_\epsilon, a_\epsilon) = \delta(s_\epsilon h^T + b) \tag{9}$$

where δ denotes activation function. h_ϵ and b are trainable parameters in Q-learning.

After obtaining Q value, the loss for Q-learning can be defined based on the Temporal Difference Error (TDE) of the DRL as:

$$L_{TDE} = \Big(r(s_\epsilon, a_\epsilon) + \gamma \max_{a'} Q(s_{\epsilon+1}, a') - Q(s_\epsilon, a_\epsilon) \Big) \tag{10}$$

3.2.6. *Sequence Modification Component.* The process of modifying a user's history sequence is treated as a hierarchical MDP, denoted as M. MDP is a mathematical framework used to describe sequential decision problems involving states, decisions, and rewards. In this process, current decisions depend only on the current state, unaffected by previous decisions, aiming to maximize long-term rewards. This process is bifurcated into two distinct steps: a high-level task denoted as M_h and a low-level task denoted as M_l. M_h is responsible for determining whether the complete sequence necessitates modification. Should modification be deemed essential, the process advances to M_l. The M_l, in turn, is tasked with deciding whether each individual historical record in the sequence should be deleted. Subsequent to the modification of the sequence, the DRL agent receives delayed rewards based on the interactions with the environment and the adjusted sequence.

There are several key definitions used in the sequence modification process:

Environment. It can be regarded as set of items, along with pre-learned base recommendation model.

State. In high-leveled assignment, it can be regarded as cosine similarity of existing history sequence with the feature vector of the target item. In low-leveled task, the state is characterized by the mean cosine similarity and mean element-to-element product between the feature vectors of every history record in the sequence and the feature vector of the targeted item.

Actions and Decisions: In high-leveled task, the action a_h is specified as a boolean $a_h \in \{0, 1\}$, indicating whether to proceed with the low-leveled task and make alterations to the historical records. The actions in the low-leveled task are a set of booleans $a_l \in \{0, 1\}$, representing whether to delete each sequence element. The goal is to determine which action the agent should take, and the model uses probabilities to decide the appropriate action. The model executes the following low-level actions:

$$h_l = \text{Relu}(W_l s_l + b_l) \tag{11}$$

$$\pi(s_l, a_l) = P(a_l|s_l, \psi_l) = a_l\sigma(W_{l1}h_l) + (1 - a_l)(1 - \sigma(W_{l2}h_l)) \tag{12}$$

where W_{l1}, W_{l2}, b_l are learnable parameters, h_l represents the characteristics of the input condition. The parameters to be acquired are denoted as $\psi_l = \{W_{l1}, W_{l2}, b_l\}$. σ is the sigmoid function transforming inputs into probabilities.

Rewards: The reward represents the rationality of the executed actions. For low-leveled task, presuming that each action within the low-leveled task sequence entails a deferred reward in relation to the final action within the sequence, the reward can be represented based on the accuracy difference between the modified sequence and the original sequence. During the execution of the low-leveled task, the agent retains the option to expunge all elements. In such circumstances, the model arbitrarily designates a singular element to compose the revised sequence. In the execution of high-leveled task, should the decision be made to alter the sequence, the reward aligns with the corresponding reward for low-leveled task. Opting against sequence modification results in a reward of 0. Furthermore, the model introduces an intrinsic reward, denoted as G, within low-leveled task. This serves the purpose of encouraging the agent to favor items most pertinent to the desired item. The model computes the average cosine similarity of every element pairs before and after modification in relation to the target item. The disparity serves as the internal reward G.

Objective Function. The goal is to identify the most effective parameters for the policy function in order to optimize the reward:

$$\psi^* = \arg\max_{\psi} \sum_{\tau} P_\psi(\tau; \psi) R(\tau) \tag{13}$$

where ψ represents either ψ_h or ψ_l. τ represents the series of sampling maneuvers and transitional states in the process. $P_\psi(\tau; \psi)$ denotes the corresponding sampling probability. $R(\tau)$ indicates the reward for sampled sequence τ.

Through sequence modification component, modified sequence of user's learning history can be obtained. Since the task is to adjust the basic recommendation model using these modified interactions to obtain more precise results, these modified sequences are transmitted to the fundamental recommendation model.

3.3. **Implementation of DRL Algorithm on the Hadoop Platform.** The main steps of implementing the DRL algorithm on the Hadoop cloud platform are as follows:

1. Establishing rewards and penalties for the state-action matrix.

2. Establishing the Q-function value-optimal policy matrix.
3. Generating recommendation results based on the optimal policy.

The algorithm is implemented on Hadoop by MapReduce, decomposing all algorithm operations into tasks handled by Map and Reduce.

1. Establishment of rewards and penalties for the state-action matrix involves grouping different users' environmental states and actions at different times, finding the rewards and penalties at that time, and then conducting individual counting and pairwise counting.
2. Establishing the Q-function value-optimal policy matrix involves defining MapReduce jobs to build the matrix with the optimal Q-function values.
3. In the part of generating recommendations, results lists are obtained based on the optimal policy.

In the proposed framework, Hadoop cloud computing platform with R language are adopted to implement the RS under distributed computing paradigm. This approach not only addresses the storage issues of massive data information through the dynamic storage space capability provided by the Hadoop platform but also improves the efficiency of the recommendation algorithm through the distributed parallel computing and data analysis capabilities. This better realizes personalized services in the RS, meeting users' individualized needs.

The implementation of this system relies on user registration information, various operation records, and log files in the RS. The RS utilizes user registration information, routine behavior records, and log files to generate models that align with users' preferences, enabling personalized recommendations. To meet user expectations, recommendation results need to be produced within a tolerable timeframe and in real-time. The recommendation system adopts the Spring MVC architecture, providing modularity to the system. This not only enhances the scalability of the RS but also improves its stability. The logical architecture and constituent modules of the RS are illustrated in Figure 4.

4. **Experiment.** This paper offers empirical findings of the envisaged framework alongside comparative methodologies, assessed across two real-world datasets. Subsequent to this, a detailed examination of the outcomes will be provided. All experiments were evaluated on a PC running Ubuntu 18.04, configured with Intel(R) Core i7-7800 3.5GHz CPU, 128GB RAM, and NVIDIA GeForce GTX 1080. In Q-learning, the reward for correct items was set to 1, and the reward for incorrect items was set to 0. As for the training parameters, The training epoch for the basic recommendation model is set to 40, with a learning rate of 0.02 and 18 features. The pre-training epoch is set to 50, with a pre-training learning rate of 0.05, and a joint learning rate of 0.05. Additionally, the delay coefficient for joint training is set to 0.0003.

4.1. **Dataset description.** The first dataset used in the experiment is the Goodbooks-10k dataset, comprising 912,705 borrowing records in the realm of book recommendations [28]. After data processing, a total of 753,267 borrowing records were obtained, comprising 9,800 books and sequences from 35,426 users. Additionally, a real-world MOOC dataset was collected, containing 435,624 viewing records processed to remove records with too few views. These records involve 91,224 users and 2,038 courses. Compared to the first dataset, the MOOC data has more users but fewer item categories, indicating weaker sparsity and shorter sequence lengths. Possible reasons include the following: 1) Greater user diversity—MOOCs typically attract a large number of users from various backgrounds and fields of interest. Due to the wide-ranging differences in user interests and academic domains, interactions among users for common items may be more dispersed, diminishing

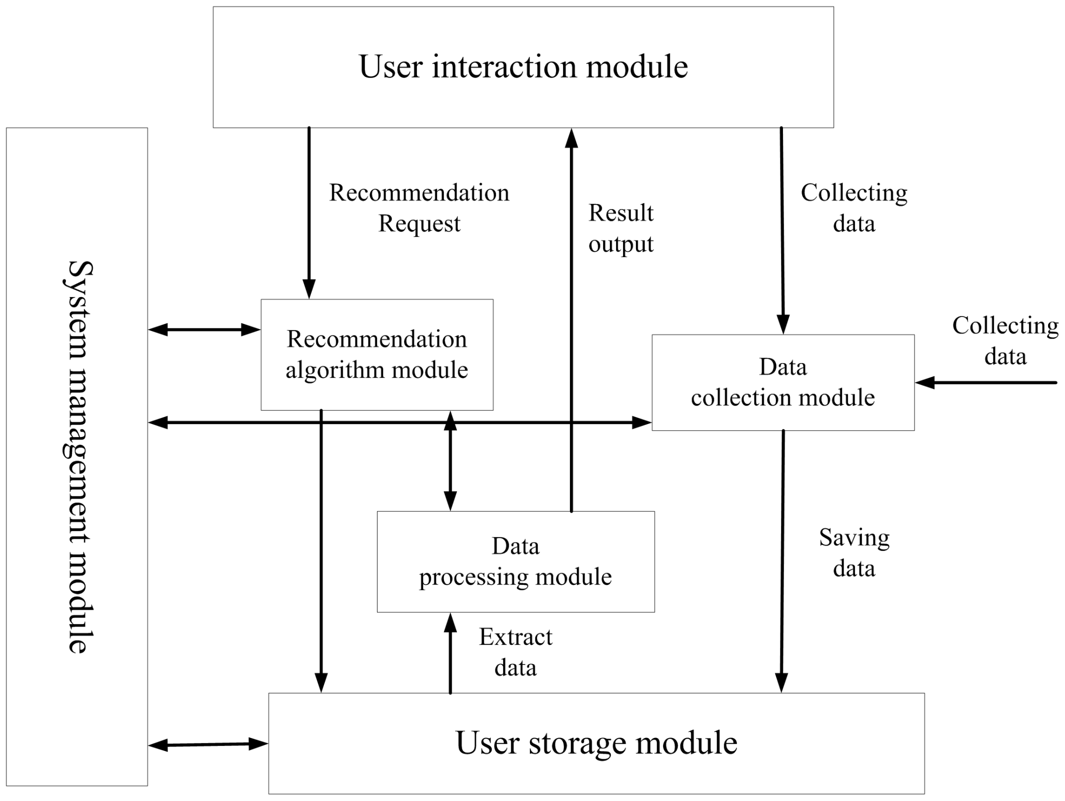

Figure 4. Distributed RS Implementation Process Diagram

the dataset's density. 2) Subject-specific nature—MOOCs often focus on specific subjects or fields, resulting in a relatively smaller number of item categories compared to datasets covering more diverse topics. This reduction in item categories affects dataset sparsity. 3) Short-term learning behavior—MOOCs primarily offer short-term courses, where users complete learning tasks within a limited timeframe. In contrast to long-term learning behaviors, short-term learning activities may lead to shorter sequence lengths, reducing the temporal information in the dataset. Therefore, shorter sequences in the MOOC data were excluded from testing for more accurate comparison results. The detailed information of the two datasets are given in Table 1.

Table 1. Statistical Information of the Experimental Dataset

	Goodbooks-10k	MOOC
Records	753,267	435,624
Users	35,426	91,224
Categories	9,800	2,038
Training data	677,940	392,061
Testing data	75,327	43,563

4.2. **Evaluation metrics.** To evaluate the recommendation effectiveness of the proposed model and comparative methods, HR@k, Recall@k, and NDCG@k were adopted as evaluation metrics. HR and Recall measure the proportion of target items correctly ranked in the top-k list, while NDCG assesses the ranking of target items in the top-k list. Recall@k focuses on the comprehensiveness of the RS in capturing user interests, aiming to minimize the omission of items that users find interesting. NDCG@k takes into account the position of items in the recommendation list and their relevance, providing a more comprehensive evaluation of the quality of the recommendation results. NDCG@k pays attention to the ranking of items in the recommendation list and the level of user interest in these items, offering a more detailed performance assessment. Given $\{v_u^1, ..., v_u^{N_u}\}$ as the item set for user u, where N_u is the number of items interacted by user u. For each item, a top-k list S_u is generated. The quantity of items interacted by user u in the recommendation list is defined as follows:

$$S_u = \sum_{i=1}^{N_u} \mathbb{I}(\text{rank}(v_u^i, l_u^i) \leq k) \tag{14}$$

Where $\text{rank}(v, l)$ is the ranking of item v in list l, and $\mathbb{I}(x)$ is an indicator function. $\mathbb{I}(x)$ returns 1 when x is true; otherwise, it returns 0. The calculation for HR, Recall and NDCG are as follows [29]:

$$\text{HR@}k = \frac{1}{|U|} \sum_{u \in U} \mathbb{I}(S_u \geq 1) \tag{15}$$

$$\text{Recall@}k = \frac{1}{|U|} \sum_{u \in U} \frac{S_u}{N_u} \tag{16}$$

$$\text{NDCG@}k = \frac{1}{|U|} \sum_{u \in U} \frac{1}{N_u} \sum_{i=1}^{N_u} \frac{\mathbb{I}(\text{rank}(v_u^i, l_u^i) \leq k)}{\log_2(1 + \text{rank}(v_u^i, l_u^i))} \tag{17}$$

4.3. **Comparison results.** Tables 2 and 3 present the results of the proposed method and comparative approaches on two experimental datasets. Notably, references [11] and [12] introduce traditional RSs based on enhanced CF algorithm. These methods, characterized by their intuitive conceptualization, are readily comprehensible and implementable. By computing similarity between users or items, rapid generation of recommendation lists is facilitated. However, such methods face challenges in effectively addressing sparsity issues, where the majority of users interact with only a limited set of items, making it difficult to identify sufficiently similar users or items. Moreover, their understanding of user interests is constrained by historical behavior, limiting the capability to capture latent interests and variations. Consequently, their performance is suboptimal. Reference [14] and [15] are value-based DRL approaches, which focus on estimating the value of taking different actions in a given state. They are known for stability in training and can handle large action spaces. However, These methods might struggle with exploring diverse recommendations due to the reliance on the learned value function. They might face challenges in handling the continuous action space and can be sensitive to hyperparameter choices. References [16] and [17] are policy-based DRL methods, directly learn the policy to map states to actions. They exhibit better exploration capabilities and can handle continuous action spaces effectively. However, they might suffer from high variance during training. References [18] and [19] are actor-critic DRL approaches, which leverage a value function (critic) to guide the learning of the policy (actor). This can lead to more stable training and improved sample efficiency. However, designing a well-balanced actor-critic network and ensuring the stability of the training process can be challenging. In comparison to the aforementioned approaches, the proposed method employs clustering for data categorization, effectively addressing data sparsity problem. Furthermore, utilizing hierarchical reinforcement learning as the sequence modification component successfully mitigates data noise issues. Unlike traditional DRL-based recommendation models encoding static user attribute information into long-term preferences, the proposed method incorporates an attention mechanism to dynamically learn user preferences from extended historical interactions. Unlike conventional DRL algorithms that amalgamate user long-term and short-term preferences indiscriminately, our approach utilizes an attention mechanism to discern the contributions of long and short-term preferences separately. Consequently, the recommended outcomes of the proposed method surpass those of other comparative methods on both datasets.

Table 2. Comparative results on Goodbooks-10k dataset

Methods	HR@20	HR@50	Recall@20	Recall@50	NDCG@20	NDCG@50
[14]	0.2017	0.3815	0.0285	0.0624	0.0138	0.0241
[15]	0.3315	0.5815	0.0492	0.1077	0.0199	0.0381
[16]	0.3577	0.6032	0.0511	0.0722	0.0184	0.0288
[17]	0.2433	0.5411	0.0525	0.0788	0.0166	0.0244
[18]	0.3733	0.5769	0.0514	0.0733	0.0202	0.0286
[19]	0.4578	0.6762	0.0546	0.1375	0.0216	0.0318
Proposed method	0.4872	0.7345	0.0682	0.1733	0.0307	0.0582

Table 3. Comparative results on MOOC dataset

Methods	HR@20	HR@50	Recall@20	Recall@50	NDCG@20	NDCG@50
[14]	0.1538	0.2697	0.0199	0.0628	0.0085	0.0137
[15]	0.1945	0.3037	0.0202	0.0670	0.0109	0.0161
[16]	0.2346	0.3536	0.0397	0.0688	0.0127	0.0189
[17]	0.2617	0.3618	0.0383	0.0695	0.0131	0.0223
[18]	0.2538	0.4147	0.0412	0.0705	0.0133	0.0201
[19]	0.2629	0.4222	0.0415	0.0712	0.0141	0.0203
Proposed method	0.2766	0.4358	0.0428	0.0892	0.0153	0.0271

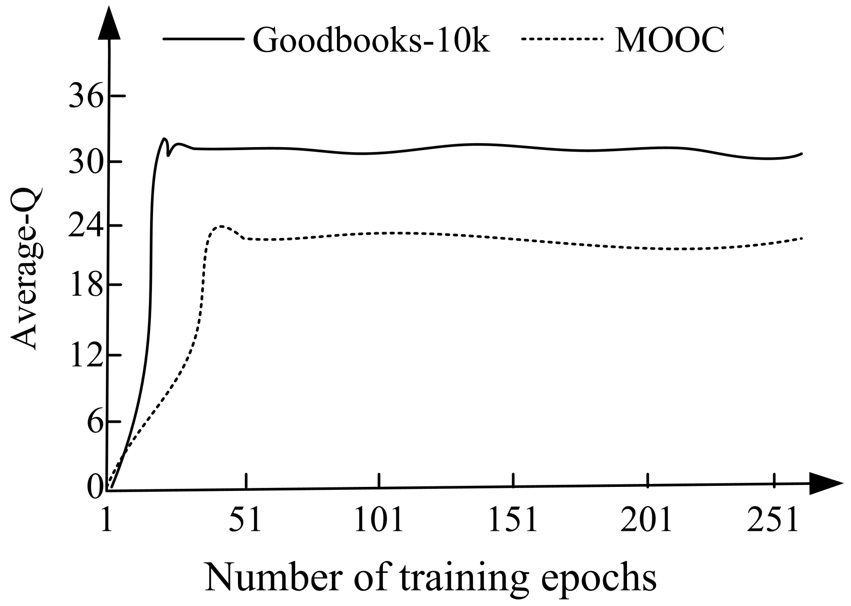

Figure 5. Training process of the proposed method

The hyperparameter of the clustering component is the quantity of clusters. If the quantity of clusters has been established at a conservative limit, many unrelated items may be grouped together, making it challenging for the hierarchical DRL model to distinguish these items and affecting the final results. If the quantity of clusters has been fixed at an overly ambitious level, the clustering model's discriminating power weakens, and the hierarchical DRL model may delete too many projects, leading to suboptimal results. Numerous parameter experiments were conducted, and the results on Goodbooks-10k dataset is given in Table 4. The results demonstrate that the best results were obtained with 500 clusters.

Table 4. Results on Goodbooks-10k with different category numbers

Categories	HR@20	HR@50	Recall@20	Recall@50	NDCG@20	NDCG@50
10,000	0.2872	0.4017	0.0307	0.0975	0.0192	0.0358
5,000	0.3544	0.6374	0.0498	0.1345	0.0285	0.0473
1,000	0.4666	0.7129	0.0679	0.1648	0.0301	0.0560
500	0.4872	0.7345	0.0682	0.1733	0.0307	0.0582

To test the algorithm's performance conveniently, synthetic datasets in specific formats required for DRL algorithm were manually generated. The dataset, formatted as UserID, State, ItemID, Action, and Reward, corresponds to different users, user states, items, user actions, and action rewards, respectively. Three synthetic datasets were created with different data sizes: one thousand rows, ten thousand rows, and one hundred thousand rows. A comparison was made between the proposed algorithm and a single-machine DRL algorithm for online calculation execution time. Additionally, the classic ItemCF algorithm executed on the Hadoop cloud platform was compared. The results are presented in Table 5. It can be observed that when the data volume is small, the proposed DRL algorithm's efficiency on a single machine is slightly better than that on the Hadoop cloud platform. However, as the data scale increases, the advantages of the Hadoop platform become apparent, and the efficiency advantage becomes more pronounced with larger data volumes. Therefore, compared to a single-machine environment, implementing the DRL algorithm on the Hadoop platform provides a good solution for handling big data processing challenges. A comparison with the online calculation time of the classical ItemCF algorithm showed that the proposed algorithm is more suitable for online calculation, providing significant help in addressing real-time online issues.

Table 5. Online execution time of DRL algorithm (/s)

Data Volume	Hadoop platform		Single machine
	Classic ItemCF	Proposed method	Proposed method
1000	0.6	0.4	0.37
10, 000	0.8	0.5	5.2
100, 000	10.7	8.3	31

5. **Conclusion.** This paper starting from the requirements and design of the recommendation system, implemented an RS based on the Hadoop distributed computing platform. Existing RSs for learning resources often overlook the impact of noise data and data sparsity. Therefore, this paper proposes a hierarchical DRL model based on clustering for recommending learning resources. By aggregating the basic recommendation model, clustering model, and hierarchical DRL model, the proposed model addresses noise data and improves the impact of data sparsity on the RS, achieving better results. To enhance the training speed and accuracy of low-level tasks, a method using Q-learning to strengthen

hierarchical DRL is proposed. The results from testing different recommendation algorithms demonstrate the effectiveness of the proposed model. In the times ahead, as a myriad of recommendation algorithms, including network and graph-based ones, continue to surface, contemplating their integration into the Hadoop platform promises to inject a rich tapestry of diversity into our recommendation algorithms. Moreover, for enhanced real-time precision in recommendation services, delving into the incorporation of open-source online real-time streaming computing systems like Spark and Storm presents an avenue. This approach facilitates comprehensive decision-making rooted in diverse user information, yielding immediate and refined recommendation outcomes. Additionally, the prospect of fortifying the Hadoop cloud platform's operational security could be explored through the inclusion of monitoring software like CloudWatch and ScaleIO.

Acknowledgment. This work is supported by the Scientific Research Project of the Fourth Council of Guangdong Society of Technical and Vocational Education for 2023-2024 (No. 202212G078).

REFERENCES

[1] C. Greenhow, C. R. Graham, and M. J. Koehler, "Foundations of online learning: Challenges and opportunities," Educational Psychologist, vol. 57, no. 3, pp. 131-147, 2022.

[2] N. H. Al-Kumaim, S. H. Hassan, M. S. Shabbir, A. A. Almazroi, and H. M. Abu Al-Rejal, "Exploring the Inescapable Suffering Among Postgraduate Researchers," International Journal of Information and Communication Technology Education, vol. 17, no. 1, pp. 19-41, 2021.

[3] S. S. Khanal, P. W. C. Prasad, A. Alsadoon, and A. Maag, "A systematic review: machine learning based recommendation systems for e-learning," Education and Information Technologies, vol. 25, no. 4, pp. 2635-2664, 2019.

[4] M. Rafiq, S. H. Batool, A. F. Ali, and M. Ullah, "University libraries response to COVID-19 pandemic: A developing country perspective," The Journal of Academic Librarianship, vol. 47, no. 1, pp. 102280, 2021.

[5] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and Debias in Recommender System: A Survey and Future Directions," ACM Transactions on Information Systems, vol. 41, no. 3, pp. 1-39, 2023.

[6] T. Silveira, M. Zhang, X. Lin, Y. Liu, and S. Ma, "How good your recommender system is? A survey on evaluations in recommendation," International Journal of Machine Learning and Cybernetics, vol. 10, no. 5, pp. 813-831, 2017.

[7] X. Chen, L. Yao, J. McAuley, G. Zhou, and X. Wang, "Deep reinforcement learning in recommender systems: A survey and new perspectives," Knowledge-Based Systems, vol. 264, pp. 110335, 2023.

[8] T.-Y. Wu, F. Kong, Q. Meng, S. Kumari, and C.-M. Chen, "Rotating behind security: an enhanced authentication protocol for IoT-enabled devices in distributed cloud computing architecture," EURASIP Journal on Wireless Communications and Networking, vol. 2023, 36, 2023.

[9] I. Konstas, V. Stathopoulos, and J. M. Jose, "On social networks and collaborative recommendation," in Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, IEEE, 2009, pp. 195-202.

[10] K. Choi, D. Yoo, G. Kim, and Y. Suh, "A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis," Electronic Commerce Research and Applications, vol. 11, no. 4, pp. 309-317, 2012.

[11] S.-T. Yang and M.-C. Hung, "A model for book inquiry history analysis and book-acquisition recommendation of libraries," Library Collections, Acquisitions, and Technical Services, vol. 36, no. 3-4, pp. 127-142, 2012.

[12] S. S. Sohail, J. Siddiqui, and R. Ali, "An OWA-Based Ranking Approach for University Books Recommendation," International Journal of Intelligent Systems, vol. 33, no. 2, pp. 396-416, 2017.

[13] S. Zhou, X. Dai, H. Chen, W. Zhang, K. Ren, R. Tang, X. He, Xiuqiang and Y. Yu, "Interactive Recommender System via Knowledge Graph-enhanced Reinforcement Learning," in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, IEEE, 2020, pp. 179-188.

[14] L. Zou, L. Xia, Z. Ding, J. Song, W. Liu, and D. Yin, "Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, IEEE, 2019, pp. 2810-2818.

[15] R. Gao, H. Xia, J. Li, D. Liu, S. Chen, and G. Chun, "DRCGR: Deep Reinforcement Learning Framework Incorporating CNN and GAN-Based for Interactive Recommendation," in 2019 IEEE International Conference on Data Mining (ICDM), IEEE, 2019, pp. 1048-1053.

[16] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi, "Top-K Off-Policy Correction for a REINFORCE Recommender System," in Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, IEEE, 2019, pp. 456-464.

[17] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R.Tang, and Y. Yu, "Large-Scale Interactive Recommendation with Tree-Structured Policy Gradient," in Proceedings of the AAAI Conference on Artificial Intelligence, IEEE, 2019, vol. 33, no. 01, pp. 3312–3320.

[18] X. Chen, C. Huang, L. Yao, X. Wang, W. Liu, and W. Zhang, "Knowledge-guided Deep Reinforcement Learning for Interactive Recommendation," in 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1-8.

[19] L. Chen, J. Cao, W. Liang, J. Wu, and Q. Ye, "Keywords-enhanced Deep Reinforcement Learning Model for Travel Recommendation," ACM Transactions on the Web, vol. 17, no. 1, pp. 1–21, 2022.

[20] Y. Hou, W. Gu, W. Dong, and L. Dang, "A Deep Reinforcement Learning Real-Time Recommendation Model Based on Long and Short-Term Preference," International Journal of Computational Intelligence Systems, vol. 16, no. 1, pp. 4-17, 2023.

[21] J. Zhang and M. Lin, "A comprehensive bibliometric analysis of Apache Hadoop from 2008 to 2020," International Journal of Intelligent Computing and Cybernetics, vol. 16, no. 1, pp. 99-120, 2022.

[22] S. Kataria and U. Batra, "Co-clustering neighborhood—based collaborative filtering framework using formal concept analysis," International Journal of Information Technology, vol. 14, no. 4, pp. 1725–1731, 2022.

[23] Y. Ni, X. Chen, W. Pan, Z. Chen, and Z. Ming, "Factored heterogeneous similarity model for recommendation with implicit feedback," Neurocomputing, vol. 455, pp. 59–67, 2021.

[24] K. Bi, J. Liu, Q. Zhao, Y. Chen, B. Xing, and B. Guo, "A crowd-sourcing recommendation algorithm OPCA-CF using outer-product co-attention mechanism," Nondestructive Testing and Evaluation, pp. 1–23, 2023.

[25] Z. Liang, L. Mu, J. Chen, and Q. Xie, "Graph path fusion and reinforcement reasoning for recommendation in MOOCs," Education and Information Technologies, vol. 28, no. 1, pp. 525–545, 2022.

[26] T.-Y. Wu, Q. Meng, L. Yang, X. Guo, and S. Kumari, "A provably secure lightweight authentication protocol in mobile edge computing environments," The Journal of Supercomputing, 2022. [Online]. Available: https://doi.org/10.1007/s11227-022-04411-9.

[27] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," Information Sciences, vol. 512, pp. 1170-1191, 2020.

[28] S. Qassimi, E. H. Abdelwahed, M. Hafidi, and A. Qazdar, "Towards a folksonomy graph-based context-aware recommender system of annotated books," Journal of Big Data, vol. 8, no. 1, pp. 1-17, 2021.

[29] J. Ma, T. Sun, and X. Zhang, "Time Highlighted Multi-Interest Network for Sequential Recommendation," Computers, Materials & Continua, vol. 76, no. 3, pp. 3569–3584, 2023.

[30] D. Liu and C. Yang, "A Deep Reinforcement Learning Approach to Proactive Content Pushing and Recommendation for Mobile Users," IEEE Access, vol. 7, pp. 83120–83136, 2019.