Neural Computing and Applications
https://doi.org/10.1007/s00521-023-08686-8

S.I. : 2021 INDIA INTL. CONGRESS ON COMPUTATIONAL INTELLIGENCE

Massive open online course recommendation system based on a reinforcement learning algorithm

Jian-Wei Tzeng[1] · Nen-Fu Huang[2] · An-Chi Chuang[3] · Ting-Wei Huang[2] · Hong-Yi Chang[4]

Received: 17 August 2022 / Accepted: 10 May 2023

Abstract

Massive open online courses (MOOCs) are open online courses designed on the basis of the teaching progress. Videos and learning exercises are used as learning materials in these courses, which are open to numerous users. However, determining the prerequisite knowledge and learning progress of learners is difficult. On the basis of learners' online learning trajectory, we designed a set of practice questions for a recommendation system for MOOCs, provided suitable practice questions to students through the LINE chatbot (a type of social media software), and used mobile devices to encourage participation in MOOCs. Reinforcement learning, which involves reward function design and iterative solution improvement, was used to set task goals, including those related to course learning and practice question difficulty. The proposed system encouraged certain learning behaviors among students. Students who used the system exhibited an exercise completion rate of 89.97%, which was higher than that of students who did not use the system (47.23%). The system also increased the students' overall learning effectiveness. Students who used and did not use the proposed system exhibited average midterm scores of 64.73 and 58.21, respectively. We also collected 227 online questionnaires from students. The results of the questionnaires indicated that 90% of the students were satisfied with the system and hoped to continue using it.

Keywords Massive open online course (MOOC) · Deep reinforcement learning (RL) · Recommendation system · Learning analysis · Learning assistant

1 Introduction

Massive open online courses (MOOCs; e.g., Coursera, Udemy, and edX), which are flexible open-access learning resources, have gained popularity among learners. Learners can participate in these courses at their convenience through the Internet. The number of students enrolled in

Nen-Fu Huang, An-Chi Chuang, Ting-Wei Huang and Hong-Yi Chang have contributed equally to this work.

✉ Hong-Yi Chang
hychang@mail.ncyu.edu.tw

Jian-Wei Tzeng
tjw@nutc.edu.tw

Nen-Fu Huang
nfhuang@cs.nthu.edu.tw

An-Chi Chuang
anchichuang@gmail.com

Ting-Wei Huang
wei84914@gmail.com

[1] Department of Information Management, National Taichung University of Science and Technology, Sanmin Rd., Taichung City 40401, Taiwan, R.O.C.

[2] Department of Computer Science, National Tsing Hua University, Kuang-Fu Road, Hsinchu City 30013, Taiwan, R.O.C.

[3] Institute of Communications Engineering, National Tsing Hua University, Kuang-Fu Road, Hsinchu City 30013, Taiwan, R.O.C.

[4] Department of Management Information Systems, National Chiayi University, Sinmin Rd., Chiayi City 60054, Taiwan, R.O.C.

Published online: 17 June 2023

MOOCs is increasing every year. MOOCs enable large-scale interactive participation, and they provide various learning materials, such as videos, text, and exercises. The consistent increase in the number of MOOC users has enabled data on users to be collected [1]. Thus, the application of artificial intelligence to MOOCs to analyze a large quantity of learning data has drawn considerable attention. Researchers have described various problems associated with MOOCs. For example, although the self-regulated learning structures of MOOCs provide considerable flexibility, many learners still cannot complete the course because of their stress-free learning environment [2]. Several studies have developed recommendation systems for MOOCs. Many MOOC providers have attempted to encourage MOOC use among students by using recommendation systems. A recommendation system recommends customized resources to help students learn from various learning materials at their own pace [3, 4]. Consequently, research has increasingly focused on the integration of recommendation systems into MOOCs to provide personalized recommendations to learners [5].

Various methods, such as collaborative filtering (CF), have been proposed for constructing recommendation systems. In classical recommendation methods, learning exercises are recommended on the basis of similarities between students and their learning behaviors. Most studies have used strategies that involve recommending learning exercises in which students have previously provided wrong answers. Such strategies have several limitations. For example, they cannot be used to recommend exercises with the appropriate difficulty level. Whether students answer exercises correctly affects their level of engagement and satisfaction with a recommendation system.

In this study, we developed a recommendation system with a LINE bot (LINE is a social media tool commonly used in Taiwan). We proposed a learning exercise recommendation system based on a reinforcement learning (RL) algorithm. Our system accounts for "review" and "difficulty" objectives. Students generally follow in-class lessons and then complete exercises. The proposed system recommends exercises that students have not completed or have completed incorrectly. Because students learn knowledge gradually, the difficulty of the exercises cannot vary considerably [3]. Therefore, the system recommends personalized exercises with suitable difficulty levels and concepts. We selected "NTHU MOOCs," which was developed by National Tsing Hua University in Taiwan, as the research platform. The contributions of this study are as follows:

1. To the best of our knowledge, the system is the first to use the actor–critic framework of RL to recommend personalized exercises.
2. The system can encourage certain learning behaviors in students and increase learning effectiveness.

2 Related work

2.1 Deep Q learning-based recommendation systems

Deep Q learning is a type of RL in which the deep learning is used to learn actions in an environment. Two types of deep Q learning network (DQN) architectures are used. In one architecture, only the state space is used as the input, and the Q values of all actions are the output. This architecture cannot handle a large action space. In the other architecture, the state and action are fed as inputs; therefore, this architecture does not need to store all Q values in the memory and can handle a large action space. However, because this architecture must compute the Q values of all actions, it has high computational complexity. To utilize the advantages of both architectures, Peters proposed the actor–critic framework [6]. Artificial intelligence (AI) systems can use this framework to learn how to delay the fall of a pole on a cart in the Cart–Pole challenge. Such systems can also use this framework to learn how to swing a baseball perfectly. RL has been used in various fields, such as gaming [7] and robotics [8]. It has also been used in recommendation systems to account for users' feedback [9]. Peng employed a comparative method to analyze the correlation between recurrent neural networks and infer the coupling between recurrent neural networks, and they also used continuous attractors to evaluate the effectiveness of smart learning [10]. Nima and Ahmad [11] proposed a Q learning-based framework that uses Web data for recommendations. Hasan et al. developed a parking recommendation system based on Q learning that recommends nearby parking spots. DQNs are widely used in recommendation systems, and the actor–critic framework outperforms the other two architectures; therefore, we selected the actor–critic framework as the agent of our recommendation system.

2.2 Recommendation systems for online learning

Recommendation systems developed into an independent research field in the 1990s [12]. With increase in the number of options available to users, the importance of recommendation systems that facilitate decision-making has increased. Several methods have been proposed for the recommendation of learning exercises. Recommendation systems for MOOCs intelligently provide actions to

learners [13]. Collaborating filtering (CF) is a traditional technique used in recommendation systems. It involves filtering out items that a user (student) might require on the basis of the learning processes of similar users. A common method of learning exercise recommendation involves identifying users with similar answering processes. Imran et al. [14] developed a framework for recommending learning materials on the basis of content similarity. To design their recommendation system, Rama et al. used the deep autocoder of feature learning; specifically, they embedded the features of the autocoder into a novel discriminative model of a deep neural network [15].

In content-based filtering (CBF), previously used content is analyzed, and a mechanism of the features based on these items is constructed. Sivaramakrishnan et al. proposed a hybrid Bayesian stacked auto-denoising encoder, which used interest analysis and matrix factorization to achieve collaborative filtering and provided high-quality recommendations through a personalized method [16]. Huang et al. [17] used CBF and natural language processing to filter out similar courses.

DQNs are used in MOOC recommendation systems. Huang et al. [18] proposed a DQN-based system for recommending learning exercises to students. In this system, states and actions are used as inputs. States are learning exercises that students have solved previously, and actions are the exercises recommended to students. RL models learn from feedback; however, obtaining feedback from the real world or a simulated environment is difficult. Therefore, Huang et al. used students' exercise logs as feedback. They tested their model on a math course and Java course and compared it with other models. The DQN-based model outperformed the other models. In the math course, the model of Qi et al. achieved NDCG@10 and NDCG@15 values of 0.6114 and 0.7813, respectively. Moreover, in the Java course, this model achieved NDCG@10 and NDCG@15 values of 0.4538 and 0.5907, respectively. However, when states and actions are used as inputs, a considerable amount of time is required to compute the Q values of all actions. Therefore, we used the actor–critic architecture to construct a recommendation model based on course logs from prior years. The model sends recommendations through a LINE chatbot to new students so that feedback can be collected.

3 System architecture

This section introduces the architecture of the MOOC exercise recommendation system (MOOCERS) and describes the problem statement of this system. Figure 1 illustrates the architecture of the MOOCERS, which contains an exercise module and a recommendation module.

3.1 NTHU MOOCs platform

The NTHU MOOCs platform [19] has been the main MOOC platform of National Tsing Hua University since 2019. This platform provides not only an efficient learning and teaching environment for online lecturers and learners but also valuable supplementary resources, such as self-study resources, a performance visualization function, and knowledge maps.

3.2 Exercise module

The exercise module determines the knowledge concept encoding, exercise type, and exercise difficulty and then concatenates these three vectors. Exercises can be single, multiple, or filling exercises and are transformed into vectors through one-hot encoding [20]. The knowledge concept encoding is also transformed using this method. With regard to exercise difficulty, students are divided into a high- and low-scoring group on the basis of their accuracy during exercises. The accuracy in the exercise is then calculated using the accuracy from the two groups. Finally, the exercise difficulty is computed using the final accuracy. The accuracy for an exercise is calculated as Formula (1):

$$R_{\text{correct}} = \frac{\sum_{i=1}^{N} C_i}{\sum_{i=1}^{N} A_i} \tag{1}$$

where R_{correct} is the accuracy of an exercise, C is the number of correct answers provided by student i in an exercise, and A is the total number of answers provided by student i in the exercise. For example, if a student that provides five correct answers out of a total of 13 answers has an accuracy of 5/13 = 0.38 (i.e., 38%). If the difficulty level is directly calculated using each student's accuracy, problems can emerge. When the percentages of correct answers for two questions are the same, the group of students with the higher number of correct answers cannot be determined.

Thus, high- and low-scoring groups are considered to calculate the degree of difficulty instead of the percentage of correct answers. The high-scoring group, low-scoring group, and exercise difficulty are defined as follows:

- High-scoring group: students with the top 27% accuracy.
- Low-scoring group: students with the bottom 27% accuracy.
- Exercise difficulty: (average accuracy of the high-scoring group + average accuracy of the low-scoring group)/2.

On the basis of Kelley's derivation for a normal distribution, the top and bottom 27% of accuracies are used for

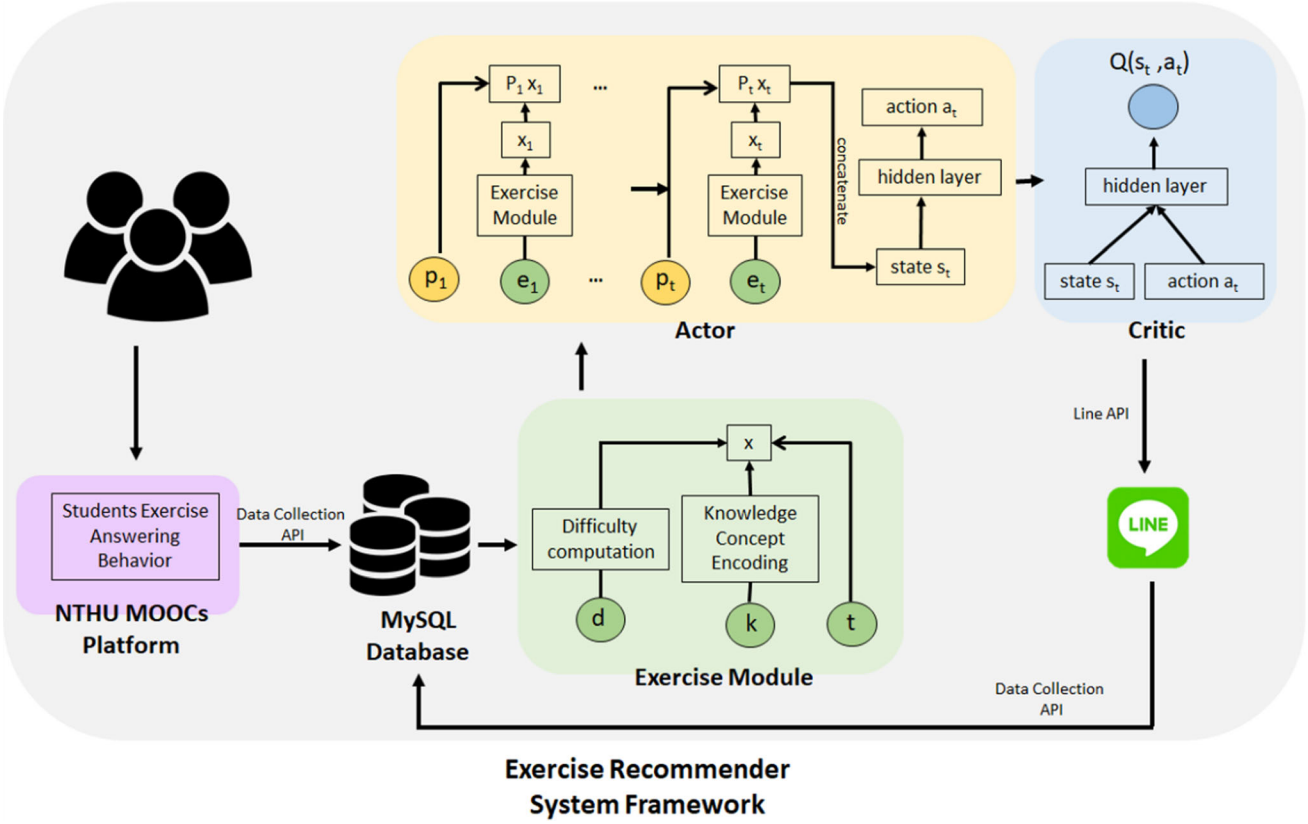

Fig. 1 Architecture of the MOOCERS

categorizing students [21]. The total accuracy in an exercise represents the overall accuracy when the average accuracies of the high- and low-scoring students are considered. After exercise difficulty is determined, the exercise difficulty is concatenated with the exercise type and exercise knowledge concept.

3.3 Recommendation module

3.3.1 Problem statement

In a MOOC environment, several students (U) and exercises (N) exist. Information on students' exercise-answering process is obtained using the data collection application programming interface (API) introduced in Sect. 3.2. The exercise-answering process of a student is given by $u = \{(e_1, p_1), (e_2, p_2), \ldots, (e_S, p_S)\}$, where $u \in \mathbf{U}$ and $e_n \in \mathbf{N}$. Parameter P_t represents a student's answer in exercise t. If the student provides the correct answer, then P_t is equal to 1; otherwise, P_t is equal to 0. A course has a total of N exercises, and each exercise is described using the tuple $e = \{d, k, t\}$, where d denotes to the exercise difficulty, k denotes the corresponding knowledge concept of the exercise, and t denotes the exercise type.

The Markov decision process (MDP) is used in the recommendation process. The MDP involves states, actions, and rewards and represented using the tuple (S, A, R, P), which

Each S_i^t represents an individual response or interaction the student had with a particular exercise up to time t.

- **State Space S**: Parameter S represents the exercise-answering process of a student. State $S_t = \{S_t^1, \ldots S_t^P\} \in \mathbf{S}$ represents the exercise-answering process at time t. Each element in S_t is the concatenation of $e = \{d, k, t\}$ and P_t.

A_i^t : Each action A_i^t represents a specific exercise in the pool of available exercises.

- **Action Space A**: Parameter A represents all the exercises of the course. Action $A_t = \{A_t^1, \ldots A_t^N\} \in \mathbf{A}$ contains all the exercises. Making an exercise recommendation at time t is equivalent to taking an action A_t^1.
- **Reward R**: When the recommender agent takes action A_t^1 at time t, the student answers or clicks. The recommender agent then receives reward (S_t, A_t) on the basis of the student's feedback.
- **Transition P**: Parameter P $(S_{t+1}S_t, A_t)$ refers to the probability of transitioning from state S_t to state S_{t+1} after taking action A_t.

Rewards can be maximized by using recommendation policy π: $\boldsymbol{S}\ \boldsymbol{A}$.

refers to how likely it is for a student to move from one state St (their current learning process) to the next state St+1 after the system recommends an exercise At

the system goal is to maximize the rewards. The way the system decides which exercises to recommend to achieve this is called a Recommendation policy, denoted as /pi

3.4 Reward function design

This section describes the design of the reward function, which plays a crucial role in the training of the proposed framework. Traditional recommendation systems follow a single rule, such as CF [17] or CBF [18], to recommend learning exercises. However, such systems cannot recommend suitable exercises for students, and students are not fully satisfied with them. Difficulty and knowledge concept coverage are crucial attributes of exercises [22]. Therefore, we combined three objectives, namely the review, difficulty, and learning objectives, in our reward function design [20].

(1) Design for the review objective. In Eq. (2), γ_1 is for review. If a student provides the wrong answer in an exercise at a certain time but the recommender agent recommends an exercise with a completely different knowledge concept, it is assigned a negative reward.

$$R_1 = \begin{cases} \gamma_1, & \text{if } P_t = 0 \text{ and } K_{t+1} \cap K_t = \emptyset \\ \gamma_2, & \text{otherwise} \end{cases} \quad (2)$$

Parameter γ_1 denotes a negative reward, and we set γ_1 to -2 in our training scenario. Parameter γ_2 denotes a positive reward, and we set γ_2 to 1 in our training scenario.

(2) Design for the difficulty objective. If a student only learns the definition of a limit in a calculus course but the recommender agent recommends the student an exercise about integration by parts, the student would find the exercise difficult. Therefore, the following squared loss function shown in formula (3) is used to meet the difficulty objective []:

$$R_2 = -(d_t - d_{t+1})^2 \quad (3)$$

where d_t is the difficul… $_{+1}$ is the difficulty of exercise B at time $t+1$.

designed to maintain an appropriate difficulty progression when recommending exercises. If the difficulty between consecutive exercises is too large, the system gets a penalty, which encourages it to recommend exercises that provide a smoother learning curve

(3) Design for the learning objective: The recommender agent recommends exercises to students through LINE. In formula (4), if a student clicks on an exercise, a positive reward is generated; otherwise, a negative reward is generated.

$$R_3 = \begin{cases} \gamma_3, & \text{if user clicks and answers} \\ \gamma_4, & \text{otherwise} \end{cases} \quad (4)$$

Parameter γ_3 denotes a positive reward, whereas γ_4 denotes a negative reward.

The rewards merge into total rewards with balance coefficients α_1, α_2, and α_3 as follows:

$$R = \alpha_1 \times R_1 + \alpha_2 \times R_2 + \alpha_3 \times R_3, \quad \alpha_1, \alpha_2, \alpha_3 \in [0, 1] \quad (5)$$

In Eq. (5), the values of α_1, α_2, and α_3 are between 0 and 1. This equation provides a flexible method for adjusting our recommendation system. Thus, different balance coefficients can be used to achieve different goals. For instance, to focus on difficulty, R_1 and R_3 can be set to 0. Alternatively, to focus on knowledge concepts, R_2 and R_3 can be set to 0.

3.5 LINE platform (API)

An API can provide a series of functions, such as APP and WEB. LINE is a popular instant messaging application with a comprehensive public API [23]. It can be installed on various platforms, such as smartphones, iPads, and personal computers. LINE provides a messaging API, which enabled us to develop a two-way communication service between a LINE chatbot and LINE users.

3.6 Actor–critic framework

The actor–critic framework combines the advantages of the

1.Actor: The actor looks at the student's current state (what they know, how they've performed so far) and recommends an exercise. It outputs parameters (scores) to suggest which exercise is best.

2.Critic: After the actor recommends an exercise, the critic checks if that recommendation was appropriate by calculating a value (Q-value). The Q-value is a score that tells the system whether the recommendation was a good fit for the student at that moment.

3.Learning and updating:

•If the critic determines that the recommendation was good, the system will continue making similar recommendations.

•If it wasn't a good recommendation, the actor learns from the critic's judgment and updates its policy to make better choices next time.

Why use this framework?
•Large action spaces: When there are many possible exercises to recommend (large action space), it's important to efficiently evaluate which actions (exercises) are most suitable. The actor-critic framework helps handle this by breaking the decision-making process into two parts: recommendation (actor) and evaluation (critic).

•Continuous improvement: The actor-critic method allows the system to continuously learn and improve the quality of its recommendations based on the critic's feedback.

Formula (6):

$$Q^*(S_t, A_t) = E_{S_{t+1}}[r_t + \gamma \max_{A_{t+1}} Q^*(S_{t+1}, A_{t+1}) S_t, A_t] \quad (6)$$

In a real-world scenario, the state and action spaces are large, and computing the state–action pairs is highly difficult. In addition, not all state–action pairs appear in real-world scenarios. The MOOC environment considered in this study contains numerous students. However, some students might only complete a few exercises. Therefore, updating the relevant state–action pairs and calculating the transition probability for when students do not complete corresponding exercises are difficult. The action-value function is nonlinear, and deep neural networks provide accurate approximations of nonlinear functions. In accordance with the method in [27], we used a deep neural network to address this problem. The approximate function Q (S, A) Q (S, A; θ). Our training procedure minimizes loss function ($\mathbf{L}$ (θ_μ)) as formula (7):

It's called nonlinear because small changes in what the system does (actions) don't always lead to simple, predictable changes in the results (outcomes).

Recommending a slightly harder exercise might help one student, but might confuse another, depending on their learning history. This makes the problem nonlinear.

$$L(\theta_\mu) = E_{S_t,A_t,R_t,S_{t+1}}[(Y_t - Q(S_t, A_t; \theta_\mu))^2],$$
$$\text{where} \quad Y_t = E_{S_{t+1}}[(R_t + \gamma Q'(S_{t+1}, A_{t+1}; \theta_\mu)), A_t] \tag{7}$$

4 Implementation

This section describes the algorithms of the proposed MOOCERS.

4.1 Data collection

By developing RESTful APIs, a connection was established between the databases and the course website. We conducted experiments with a dataset collected from a real MOOC environment to evaluate the performance of the proposed deep RL framework. The experimental dataset was collected from the NTHU MOOCs platform (Table 1). We collected data on a calculus course implemented in 2020 as training data to construct a recommendation model. We then used the trained recommendation model to send recommendations to students in a calculus course taught in 2021. To send these recommendations to the students, we asked them to connect with our LINE bot; approximately 700 students (approximately 60% of all the students) chose to connect with our bot. This course lasts for 12 weeks, and its topics range from limits to L'Hopital's rule and improper integrals. The course comprises 143 exercises, each of which corresponds to a specific concept. The total number of concepts is 108.

The exercise logs were randomly split into a training set and testing set at a ratio of 8:2, as expressed in Eq. (8).

$$D_{training} : D_{testing} = 0.8 : 0.2 \tag{8}$$

D_{training} represents the data used for model training, and D_{testing} represents the data used for model testing.

Each student in the calculus course can use the exercise recommender agent, but its use is optional. The binding ratio for the "2021 Calculus (I)" course was approximately 60%.

Table 2 presents the data recorded for each exercise. Every user, course, chapter, and exercise were assigned a unique ID (i.e., "userId," "courseId," "chapterId," and "exerId," respectively). When a user completed a question, we recorded their answer, the correct answer, and their score, using "True" and "False" to denote correct and incorrect answers, respectively. The time spent answering each question was also recorded.

4.2 Personalized recommendations

In RL methods, data are collected by constructing an agent in the environment. For example, classical applications, such as robotics, might be time consuming and costly [28]. Therefore, simulation is a suitable alternative for RL. However, simulated data cannot be used to obtain accurate recommendations in our system because it is more complex than robotics and gaming systems. Data for the system can only be obtained from real-world environments, such as MOOC platforms. In addition, the system can obtain rewards only from an explored state space. To obtain a reward from an unexplored space, the system must recommend an exercise to a student and obtain their feedback. To solve this problem, we used students' exercise logs. The procedure is presented in Algorithm 1.

(1) First, memory space $\boldsymbol{M} = \{M_1, M_2 \ldots\}$ is constructed to store a student's state–action–reward pair $((S_t, A_t), R_t)$.

(2) The current state (line 4), current actions (exercises), and rewards are observed and stored in the memory (line 7).

(3) The state space is updated by adding the action to the end of the state space. For instance, if the recommender agent recommends $\{A_1, A_2, A_3\}$ to a student, when the student clicks the A_1 exercise, the state space is then updated to $\{S_1, S_2, \ldots, A_1\}$ (Fig. 2).

4.2.1 Training procedure

This section describes the training procedures of the algorithm and the parameter update process. The deep deterministic policy gradient algorithm was used for model training. This algorithm is presented in Algorithm 2 [20].

In every iteration (line 4), the recommender agent recommends a list of actions (exercises) $A_t = \{A_t^1, \ldots A_t^K\}$ based on the current state S_t (line 5). The recommender agent then observes reward list $R_t = \{R_t^1, \ldots R_t^K\}$ (line 6)

Table 1 Courses

Course Name	2020 Calculus (I)	2021 Calculus (I)
Duration	2020/05/01–2020/08/31	2021/05/01–2021/08/31
Number of students	1167	1200
Number of exercises	143	143
Number of knowledge concepts	108	108
Number of records of exercise answering	122,040	25,428 (2021/05/01–06/30)
Avg. records per student	104	21

Table 2 Exercise data

Key	Value	Example
userId	The ID of the user	3002
courseId	The ID of the course	10900MATH0001
chapterId	The ID of the chapter	10900MATH0001ch80
exerId	The ID of the exercise	10900MATH0001ch80e1
score	The student gets right answer or not	False
timeCost	The time cost the student spends on the exercise	5
userAns	The answer of the student	[1,2,3]
correctAns	The correct answer of the exercise	[1,4]
update at	Time when the log was established	2021-05-10T12:45:08

and obtains new state S_{t+1} (line 7) by using Algorithm 1. The transition probability is stored in memory M (line 8), and state S_t is updated to S_{t+1} (line 9). Next, the recommender agent executes the parameter update procedure by sampling a minibatch of transitions (S, A, R, S') from M (-line 10) and updating the parameters of the actor–critic network (lines 11–17). The network sizes of the actor and critic are presented in Table 3. For the actor, because the state of a student at time t comprises a series of exercise-answering processes, a GRU is used to process the series and predict the exercise that should be recommended; specifically, 105 nodes are used to represent each exercise, and the node with the highest score represents the exercise that should be recommended. For the critic, a final dense layer with a single node is used to predict the reward. The learning rate of the actor is 0.0001, and the learning rate of the critic is 0.001. When updating the network, only 0.1% of the actor and critic are updated (Fig. 3).

Table 3 Network sizes of the actor and critic

Actor			Critic		
Layer	Size (cells)	Activation	Layer	Size (cells)	Activation
GRU	6	Relu	Dense	32	Relu
Dense	105	None	Dense	16	Relu
			Dense	1	None

4.2.2 Offline evaluation

A successful recommendation can be made in two situations depending on the reward setting: one in which the correct answer is obtained for e_t and the other in which an incorrect answer is obtained for e_t [20].

(1) Correct answer for e_t.

When a student answers correctly in an exercise that covers a certain knowledge concept, the recommender agent can recommend more difficult questions with similar knowledge concepts. If the overlapping knowledge

Fig. 2 Online memory algorithm

Algorithm 1 Online Memory

Input: Students' historical episodes E and recommendation list length R

Output: Memory M

1: for *episode* = 1 to E do
2: Set $S_0 = \{S_0^1, \ldots S_0^P\}$ as initial state
3: for *exercise* $l = 1$ to L;R do
4: current state $S = \{S^1, \ldots S^P\}$
5: current action $A = \{A_1, \ldots A_{l+R-1}\}$
6: current reward $R = \{R_1, \ldots R_{l+R-1}\}$
7: $Add((s,a) \rightarrow r)$ *to* M
8: for $r = 0$ to *R-1* do
9: if $r_{l+k} > 0$ then
10: Remove first triple in S
11: Add A_{l+R} to the bottom of S

Algorithm 2 Deep Deterministic Policy Gradient (DDPG)

Input: State Space S, Action Space A, Discount Rate γ
Output: Q(s,a)

Randomly initialize critic network $Q(s,a|\theta^{Q})$and actor network $\mu(s|\theta^{\mu})$with random weights
Initialize target network $Q^{'}$ and $\mu^{'}$ with weights $\theta^{Q^{'}} \leftarrow \theta^{Q}$, $\theta^{\mu^{'}} \leftarrow \theta^{\mu}$
Initialize Memory M

for *episode* = 1 to M do
  Receive initial state s_0 from previous sessions
  for t =1 to T do
    Select an action $A_t = \{A_{t^1},\ldots A_{t^k}\}$ according to algorithm 1
    Execute A_t and observe reward R_t
    New state S_{t+1}
    Store transition (S_t, A_t, R_t, S_{t+1}) to Memory M
    Updates state $S_t = S_{t+1}$
    Sample a minibatch of N transitions $(S, A, R, S^{'})$from M
    Set $y = r + \gamma Q^{'}(S^{'}, \mu^{'}(s^{'}|\theta^{\mu^{'}})|\theta^{Q^{'}})$
    Update critic by minimizing $(y - Q(S,A;Q^{\mu}))^2$
    Update the actor policy using the sample policy gradient:
      $\nabla_{\theta^{\mu}} U \approx \frac{1}{N}\sum_{i}^{a} \nabla_{a} Q(S, A|\theta^{\mu}) \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})$
    Update the Target network:
      $\theta^{Q^{'}} \leftarrow \tau\theta^{Q} + (1\text{-}\ \tau)\theta^{Q^{'}}$
      $\theta^{\mu^{'}} \leftarrow \tau\theta^{\mu} + (1\text{-}\ \tau)\theta^{\mu^{'}}$

Fig. 3 Deep deterministic policy gradient algorithm

concepts between e_t and e_{t+1} are at least half and the difficulty of e_{t+1} is higher than that of e_t, the recommendation is successful.

(2) Incorrect answer for e_t.

When a student answers incorrectly in an exercise, the student must continue with exercises on similar concepts and levels of difficulty. If the difference in difficulty between recommended exercises (e_{t+1}) and e_t is lower than 0.2 and the number of overlapping concepts is at least half e_t, the recommendation is successful.

4.2.3 Exercise completion rate

The exercise completion rate (ECR) represents the ratio of exercises solved by a student to the total number of exercises. The ECR of each student is as formula (9):

$$\text{Completion\_Rate}_e = \frac{\text{Number of solved exercises}}{\text{Total number of exercises}} \quad (9)$$

4.2.4 Hit rate

In typical settings for a recommender, the hit rate is the proportion of users for which the correct answer is included in the recommendation list. The hit rate is as Formula (10):

$$\text{Hit\_Rate}_{\text{exercise}} = \frac{\text{Number of clicks}}{\text{Total pushed messages}} \quad (10)$$

Number of clicks represent the total number of students clicking on the recommended exercise, and total pushed messages represent the total number of exercises pushed by the recommender agent.

4.2.5 Recommendation through LINE

We examined the hit rate, the ECR, and students' satisfaction with the system. Students were divided into a control group and an experimental group. The recommender agent recommended exercises to the experimental group, and the control group was recommended exercises randomly. Screenshots of pushed messages on LINE are presented in Fig. 4. For example, the recommender agent determined that exercises W1 Ex01 Q1 and W2 Ex23 25

Q1 were the most suitable exercises for students A and B, respectively. Students were recommended exercises through LINE, and they completed them after clicking on the links. After the exercises, they received their results.

5 Results and discussion

5.1 Accuracy of the proposed model

A recommendation system usually outputs a series of lists. Therefore, instead of proxy metrics such as the mean squared error, other suitable evaluation metrics should be used to determine the ranking quality. For recommendation systems, the commonly used evaluation metric is top@K ranking metrics [29, 30], including NDCG@k [31] and Hit Rate@K. The order of recommendations is crucial in the evaluation of a recommendation series, and DCG can be used to assign a penalty for recommendations that appear later in a list, as expressed in Eq. (11). Although DCG considers the order of a recommendation series, it does not indicate whether the series is appropriate. Therefore, NDCG, which is the ratio of the actual DCG to the ideal DCG, is calculated [Eq. (12)]. If the recommendation series is closer to the ideal series, NDCG is closer to 100%. Huang et al. [18] used a normal DQN framework to recommend exercises to students and achieved NDCG@10 scores of 0.6114 and 0.4538 in a math course and Java course, respectively, in offline testing. Because we considered offline training as a ranking task, we used NDCG@K for offline performance evaluation and hit rate for online performance evaluation. In Table 4, the NDCG@5 value of the deep RL model was 0.65; its NDCG@8 value was 0.665; and its NDCG@10 value was 0.684. NDCG@k indicated that the model selects the top k exercises on the basis of their Q values. It then calculates the scores with different reward settings. When the correct answer is obtained, if the number of overlapping concepts between e_t and $e_t + 1$ is at least half and the difficulty of $e_t + 1$ is higher than that of e_t, the recommendation is considered successful. When an incorrect answer is obtained, if the difference in exercise difficulty between e_t and $e_t + 1$ is lower than 0.2, the recommendation is considered successful.

We will continue to collect feedback from users on the chatbot and hope to encourage students to continue completing the exercises.

$$\mathrm{DCG}_k = \sum_{i=1}^{k} \frac{r_i}{\log_2(i+1)} \tag{11}$$

$$\mathrm{NDCG}_k = \frac{\mathrm{DCG}_k}{\mathrm{IDCK}_k} \tag{12}$$

5.2 ECR

Exercises are a crucial learning resource in MOOCs. Learning using MOOCs is flexible, and few students proactively complete exercises. For this reason, one goal of developing the system was to increase ECR. The ECR of the students who used our system (89.97%) was higher than that of the students who did not use our system (47.23%; Table 4; Fig. 5). We divided the students who

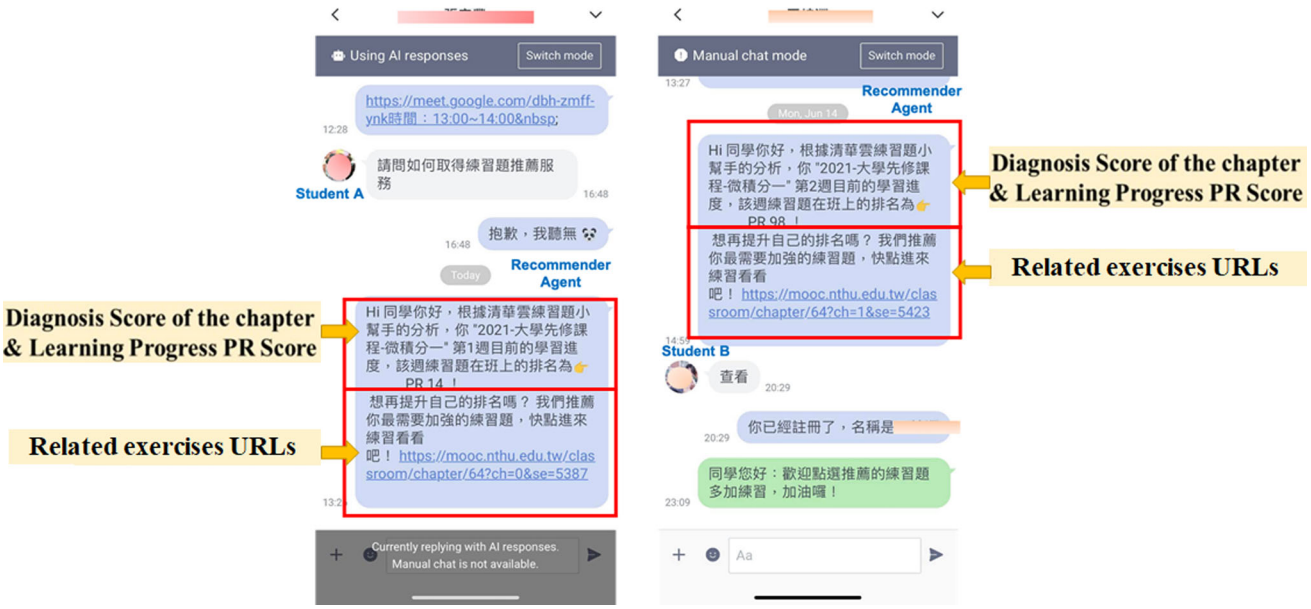

Fig. 4 Recommendation message received through a chatbot

Table 4 NDCG values obtained in this study

NDCG@5	NDCG@8	NDCG@10
0.65	0.665	0.684

used our system into the following four groups on the basis of their ECRs:

1. Very low group: ECR $< 1\%$
2. Low group: $1\% \leq$ ECR $< 25\%$
3. Middle group: $25\% \leq$ ECR $< 50\%$
4. High group: $50\% \leq$ ECR.

The ECR of students in the high group increased from 68.69% to 94.72% (Fig. 6). The ECRs of the very low group and low group increased from 0.1% to 90.07% and from 17.46% to 87.07%, respectively. The ECRs of the students in the middle group increased from 56.7% to 89.26%. Therefore, our system considerably increased ECR, especially that of the very low group, low group, and middle group.

5.3 Hit rate

In our experiment, we distributed messages four times, and 520, 640, 694, and 738 students accessed them through LINE each time, respectively. For the fourth message (Table 5), 40 (5.4%), 69 (9.3%), 88 (11.9%), and 105 (14.2%) students completed the recommended exercises within 4 h of receiving the message, on the day of receiving the message, on the day after receiving the message, and 2 days after receiving the message.

5.4 Midterm score

When the students exhibited higher learning effectiveness, they spent more time on the NTHU MOOCs platform to complete exercises. Therefore, learning behavior is correlated with learning effectiveness, and students with a higher ECR tended to receive higher midterm scores (Fig. 7). The students who used and did not use the system exhibited average midterm scores of 64.7 and 58.2, respectively.

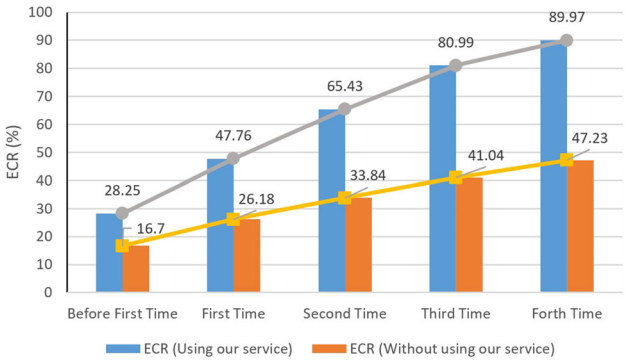

Fig. 5 ECRs of students who used and did not use the system by number of uses

5.5 Students' assessment of the system

We distributed an online questionnaire to the students who used the system to obtain their feedback. A total of 227 valid questionnaires were collected. The questionnaire was based on the self-adjusting learning strategy proposed by Zimmerman [32, 33]. The students responded to the questions by using a Likert scale [34], which is the most widely used scale in survey research. The scale ranges from 1 to 5, with 1 representing "strongly disagree" and 5 representing "strongly agree." The questionnaire contained 20 questions on usability, usefulness, attitude toward learning, perceived value, and metacognition.

Approximately 85% of the students provided a score of more than 4 points for usability, which indicates that they found the system easy to use and clear. Approximately 50% of the students provided a score of more than 4 points for usefulness, which indicates that the system recommended helpful exercises and increased the students' willingness to complete exercises. Approximately 76% of the students provided a score of more than 4 points for attitude toward learning, which indicates that the system motivated the students to learn and enabled them to answer exercises more efficiently. Approximately 90% of the students provided a score of more than 4 points for perceived value, which indicates that they were satisfied with the system. Approximately 82% of the students provided a scored of more than 4 points for metacognition, which indicates that the system helped them understand key concepts and consider feasible learning methods.

6 Conclusion

To the best of our knowledge, the system is the first to use the actor–critic framework to provide personalized exercise recommendations on a MOOC platform. Our system recommends exercises with suitable difficulty levels and concepts for each student. The NDCG@10 value of the proposed model is 0.684. In contrast to other systems, our system was integrated into LINE to provide personalized exercise recommendations to each student; thus, the students did not need to log in to the system and could receive recommendations from LINE in the form of text messages. Evaluating the effectiveness of recommendations in the real world is difficult. We examined the hit rate of each

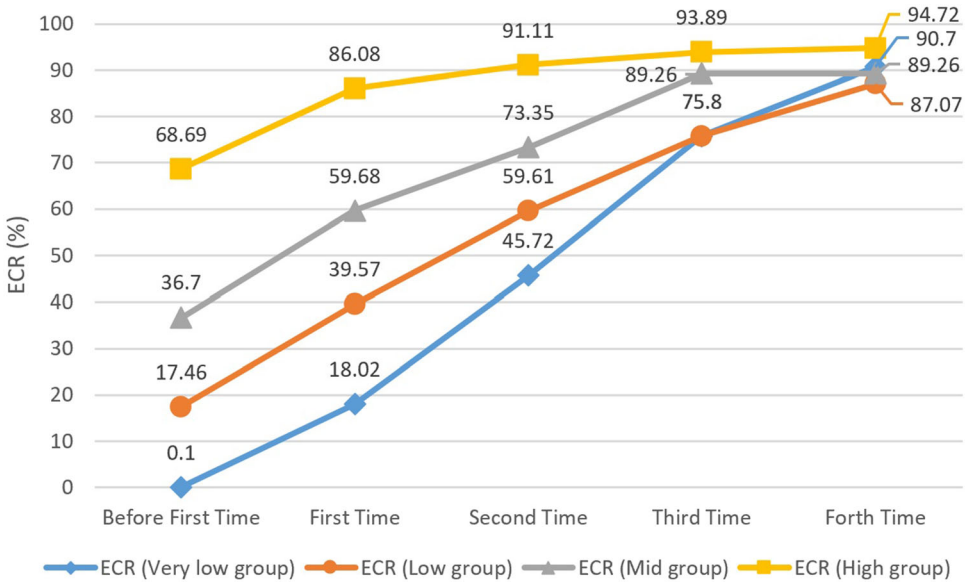

Fig. 6 ECRs of groups by number of uses

Table 5 Number of students in each group who completed the recommended exercises

	First	Second	Third	Forth
Number of students combined Line	520	640	694	738
Number of students answered in 4 h	43 (8.2%)	51 (7.9%)	43 (6.1%)	40 (5.4%)
Number of students answered on the same day	46 (8.8%)	59 (9.2%)	60 (8.6%)	69 (9.3%)
Number of students answered on the next day	53 (10.2%)	95 (14.8%)	97 (13.9%)	88 (11.9%)
Number of students answered on the day after tomorrow	54 (10.4%)	118 (18.4%)	109 (15.7%)	105 (14.2%)

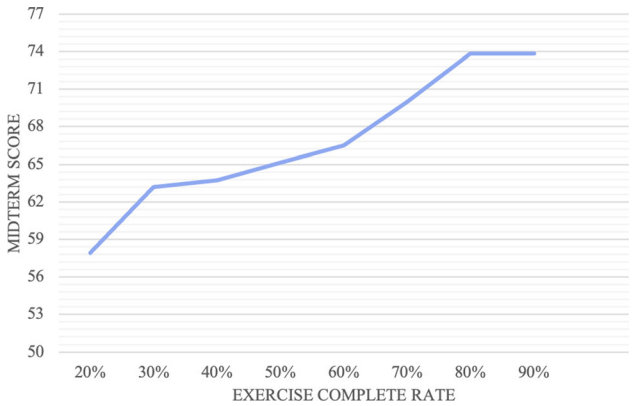

Fig. 7 Midterm scores by ECR

recommendation provided by our system and found that 40% of the participating students answered the exercises in the final recommendation. The system can encourage certain learning behaviors in students. Students who used and did not use the system exhibited ECRs of 89.97% and 47.23%, respectively. The system can also increase learning effectiveness for students. The students who used and did not use the system exhibited average midterm scores of 64.73 and 58.21, respectively. The questionnaire revealed that 88.2% of the students who used the system intended to use it again and that 89.5% of the students were satisfied with it. In the future, we intend to combine our system with the natural language processing method to make our LINE bot fully interactive.

Acknowledgements This study was funded by the National Science and Technology Council of Taiwan. The projects ID was NSTC 111-2410-H-025-039. It was also funded by the National Taichung University of Science and Technology.

Data availability The datasets generated during and/or analyzed during the current study are not publicly available due to other research cases are in progress but are available from the corresponding author on reasonable request.

Declarations

Conflict of interest There is no conflict of interest in this study.

References

1. Wang M, Peng J, Cheng B, Zhou H, Liu J (2011) Knowledge visualization for self-regulated learning. Educ Technol Soc 14(3):28–42
2. Ruiperez-Valiente JA, Merino P, Delgado-Kloos C, Niemann K, Scheffel M, Wolpers M (2016) Analyzing the impact of using optional activities in self-regulated learning. IEEE Trans Learn Technol 9:1–1
3. Zhao WX, Zhang W, He Y, Xie X, Wen J-R (2018) Automatically learning topics and difficulty levels of problems in online judge systems. ACM Trans Inf Syst 36(3):1–33
4. Jing Li ZY (2020) Course recommendations in online education based on collaborative filtering recommendation algorithm, pp 1076–2787
5. Micarelli A, Stamper J, Panourgia K (eds) (2016) Springer, Cham, pp 267–272
6. Peters J, Schaal S (2008) Natural actor-critic. Neurocomputing 71(7):1180–1190
7. Patel PG, Carver N, Rahimi S (2011) Tuning computer gaming agents using q-learning. In: 2011 federated conference on computer science and information systems (FedCSIS). IEEE, pp 581–588
8. Kober J, Bagnell JA, Peters J (2013) Reinforcement learning in robotics: a survey. Int J Robot Res 32(11):1238–1274
9. Wang P, Fan Y, Xia L, Zhao WX, Niu S, Huang J (2020) Kerl: a knowledge-guided reinforcement learning model for sequential recommendation. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, ser. SIGIR'20. Association for Computing Machinery, New York, pp 209–218
10. Peng N (2022) Research on the effectiveness of English online learning based on neural network. Neural Comput Appl 34:2543–2554
11. Taghipour N, Kardan A (2008) A hybrid web recommender system based on q-learning. In: Proceedings of the 2008 ACM symposium on applied computing, pp 1164–1168
12. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
13. Beel J, Langer S, Genzmehr M, Gipp B, Breitinger C, Nurnberger A (2013) Research paper recommender system evaluation: a quantitative literature survey. In: Proceedings of the international workshop on reproducibility and replication in recommender systems evaluation, ser. RepSys '13. Association for Computing Machinery, New York, pp 15–22
14. Imran K, Abdullah N (2009) Building an e-learning recommender system using vector space model and good learners average rating, pp 194–196
15. Rama K, Kumar P, Bhasker B (2021) Deep autoencoders for feature learning with embeddings for recommendations: a novel recommender system solution. Neural Comput Appl 33:14167–14177
16. Sivaramakrishnan N, Subramaniyaswamy V, Viloria A, Vijayakumar V, Senthilselvan N (2021) A deep learning-based hybrid model for recommendation generation and ranking. Neural Comput Appl 33:10719–10736
17. Huang R, Lu R (2018) Research on content-based Mooc recommender model. In: 2018 5th international conference on systems and informatics (ICSAI), pp 676–681
18. Huang Z, Liu Q, Zhai C, Yin Y, Chen E, Gao W, Hu G (2019) Exploring multi-objective exercise recommendations in online education systems. In: Proceedings of the 28th ACM international conference on information and knowledge management, ser. CIKM '19. Association for Computing Machinery, New York, pp 1261–1270
19. Nthu mooc (2019) https://mooc.nthu.edu.tw/
20. Chuang AC, Huang NF, Tzeng JW, Lee CA, Huang YX, Huang HH (2021) MOOCERS: Exercise recommender system in MOOCs based on reinforcement learning algorithm. In: 8th International Conference on Soft Computing & Machine Intelligence (ISCMI), Cario, Egypt, 2021, pp 186-190
21. Kelley TL (1939) The selection of upper and lower groups for the validation of test items. J Educ Psychol 30(1):17
22. Wu Z, Li M, Tang Y, Liang Q (2020) Exercise recommendation based on knowledge concept prediction. Knowl Based Syst 210:106481
23. LM Corporation. Line messenger. https://line.me/en/
24. Witten IH (1977) An adaptive optimal controller for discrete-time markov environments. Inf Control 34(4):286–295
25. Konda V, Tsitsiklis J (2000) Actor-critic algorithms. In: Solla S, Leen T, Muller K (eds) Advances in neural information processing systems, vol 12. MIT Press, Boca Raton
26. Bellman R (2013) Dynamic programming. Dover Publications.
27. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller MA (2013) Playing atari with deep reinforcement learning. CoRR abs/1312.5602
28. Rao K, Harris C, Irpan A, Levine S, Ibarz J, Khansari M (2020) Rl-cyclegan: reinforcement learning aware simulation-to-real. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
29. Park C, Kim D, Oh J, Yu H (2016) Improving top-k recommendation with truster and trustee relationship in user trust network. Inf Sci 374:100–114
30. Schroeder G, Thiele M, Lehner W (2011) Setting Goals and Choosing Metrics for Recommender System Evaluations, UCERSTI2 Workshop at the 5th ACM Conference on Recommender Systems, Chicago, USA, 23 October 2011
31. Wang Y, Wang L, Li Y, He D, Chen W, Liu T-Y (2013) A theoretical analysis of NDCG ranking measures. In: Proceedings of the 26th annual conference on learning theory (COLT 2013), vol 8. Citeseer, p 6
32. Wei X, Saab N, Admiraal W (2023) Do learners share the same perceived learning outcomes in MOOCs? Identifying the role of motivation, perceived learning support, learning engagement, and self-regulated learning strategies. Internet Higher Educ 56:100880
33. Onah DFO, Pang ELL, Sinclair JE (2022) An investigation of self-regulated learning in a novel MOOC platform. J Comput Higher Educ 25:1–34
34. Nemoto T, Beglar D (2014) Likert-scale questionnaires. In: JALT 2013 conference proceedings, pp 1–8