Zhao TY, Zeng M, Feng JH. An exercise collection auto-assembling framework with knowledge tracing and reinforcement learning. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(5): 1105–1117 Sept. 2022. DOI 10.1007/s11390-022-2412-2

An Exercise Collection Auto-Assembling Framework with Knowledge Tracing and Reinforcement Learning

Tian-Yu Zhao[1] (赵天宇), Man Zeng[2] (曾　曼), and Jian-Hua Feng[1] (冯建华)

[1] *Department of Computer Science and Technology, Tsinghua University, Beijing 100190, China*
[2] *Sanfan Chaoyang School Attached to Beijing Normal University, Beijing 100190, China*

E-mail: zhaoty17@mails.tsinghua.edu.cn; 201622090091@mail.bnu.edu.cn; fengjh@tsinghua.edu.cn

Received April 14, 2022; accepted September 23, 2022.

Abstract In educational practice, teachers often need to manually assemble an exercise collection as a class quiz or a homework assignment. A well-assembled exercise collection needs to have the proper difficulty index and discrimination index so that it can better develop students' abilities. In this paper, we propose an exercise collection auto-assembling framework, in which a teacher provides the target values of difficulty and discrimination indices and a qualified exercise collection is automatically assembled. The framework consists of two stages. At the answer prediction stage, a knowledge tracing model is utilized to predict the students' answers to unseen exercises based on their history interaction records. In addition, to better represent the exercises in the model, we propose exercise embeddings and design a pre-training approach. At the collection assembling stage, we propose a deep reinforcement learning model to assemble the required exercise collection effectively. Since the knowledge tracing model in the first stage has different confidences in the predicted answers, it is also taken into account in the objective. Experimental results show the effectiveness and efficiency of the proposed framework.

Keywords exercise collection, knowledge tracing, reinforcement learning

1 Introduction

Currently, the online education system has been widely applied in educational practice. In such platforms, students are free to take exercises to reinforce the learned knowledge and develop their skills. Meanwhile, teachers need to assign homework or organize quizzes to students. A homework assignment or a quiz paper consists of multiple exercises, and thus we refer to it as an exercise collection in the paper.

In order to assemble an exercise collection, teachers usually have to manually select several exercises from the ones that students have not done. Additionally, teachers also need to ensure that the assembled exercise collection is neither too easy nor too difficult for students, and it can distinguish students of different levels well. Normally, the task of exercise collection assembling is essential for teachers on most subjects for primary and secondary education (e.g., Chinese, mathematics, English, and physics). Traditionally, this task is highly dependent on the educational experience of teachers and may take a great amount of time.

Therefore, this paper aims to propose a framework for exercise collection auto-assembling. By using this framework, teachers only need to provide a few parameters (including the number of exercises, the target difficulty index, and the discrimination index) to automatically obtain a required exercise collection, which will save teachers a lot of time.

Because the exercises to be selected have not been done by the students, it is first necessary to predict the correctness of students' answers to unencountered exercises given their history interaction records in the system, which can be done by a knowledge tracing model. However, most of the existing models [1–4] only take the knowledge concepts in the exercise and/or the exercise text as input, which will cause other features of the ex-

Regular Paper
Special Section on Scalable Data Science
This work was supported by the National Natural Science Foundation of China under Grant Nos. 62072261 and 61925205, and Huawei and TAL education.

ercise (such as the difficulty) not to be modeled. Therefore, how to efficiently represent these extra features of the exercise and feed them into the knowledge tracing model is a challenge.

Afterwards, we need to design an algorithm to select exercises from the exercise set based on the prediction of the students' answers, making the collection in line with the teacher's requirements. However, finding the optimal exercise collection by simply traversing all the candidates may cause an impractical time overhead. Therefore, how to effectively and efficiently assemble the exercise collection is another challenge of this task.

To address the above challenges, we propose an exercise collection auto-assembling framework that utilizes students' history interactions in the system to predict their answering results of unseen exercises, and thus automatically assembles an exercise collection that meets the teacher's requirement. First, we embed each exercise into a vector as the additional input of the knowledge tracing model to represent the features of the exercise. We also design a pre-training approach for exercise embeddings to accelerate the training process. In addition, we propose a deep reinforcement learning model to select exercises and assemble an approximate optimal exercise collection.

To summarize, this paper mainly makes the following contributions.

• We propose a framework to automatically assemble required exercise collections for teachers. To the best of our knowledge, this paper is the first study to propose the exercise collection auto-assembling task and present an efficient solution.

• We introduce exercise embeddings into the knowledge tracing models, which will model the similarities and differences of exercises based on history interaction records of students. We also propose a pre-training method for exercise embeddings to speed up the training process.

• We propose a reinforcement learning based model to assemble an exercise collection efficiently and effectively.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 formally defines the exercise collection auto-assembling problem. Section 4 gives an overview of the framework. Section 5 proposes the exercise embeddings for knowledge tracing models. Section 6 proposes the reinforcement learning based model to assemble the exercise collection. Section 7 conducts experiments on our framework. Section 8 concludes the paper.

2 Related Work

2.1 Knowledge Tracing

Knowledge tracing is a task that models a student's knowledge level given his/her history exercises and predicts whether the student can correctly answer future exercises. Also, there exist some crowd-based studies [5–8] that leverage workers' historical experience to deduce the answers of tasks accurately. We mainly focus on the exercise scenario.

Statistical Knowledge Tracing Model. Bayesian Knowledge Tracing (BKT) [9] is a classical method in knowledge tracing, which models the knowledge state of the student as a set of latent variables in a hidden Markov model, and updates the variables by the result of doing exercises. However, BKT treats every knowledge concept as independent, thus ignoring the correlation between knowledge concepts.

Deep Learning Based Knowledge Tracing Model. Deep Knowledge Tracing (DKT) [1] utilizes a recurrent neural network to model the knowledge state of the student. DKT takes the history interaction records of the student as input and gives predictions for the next exercise. DKVMN (Dynamic Key-Value Memory Networks) [2] applies a memory module to represent the knowledge concepts and discover the correlation between them. Exercise-aware Knowledge Tracing (EKT) [3,4] leverages the exercise text to learn the semantic information of each exercise. Recently, there has also been some new development on graph-based knowledge tracing models [10–13], which utilize graph-based models (such as graph neural networks, graph convolutional networks) to organize the relationship among the knowledge concepts.

However, these existing knowledge tracing models only consider the knowledge concepts and the text information from the exercise, and thus some features of the exercise (e.g., the complexity of the exercise) are lost.

2.2 Reinforcement Learning

Reinforcement learning is an area of machine learning, which describes how an agent learns strategies to maximize the total reward during the process of interaction with the environment.

In the field of reinforcement learning, Q-learning [14] is a representative value-based method, in which each state has a Q-value representing the maximum reward

that it can obtain in the future. Q-learning uses a Q-table to store the Q-values of all possible states and applies dynamic programming to decide which action to take to finally reach the state with maximum reward. However, when the number of states is quite large, it needs long time and large memory to compute and store the values in Q-table. Therefore, the DQN (Deep Q-Network) model further improves Q-learning by using a neural network to estimate the Q-value of a state, which will take less time and memory than Q-learning. There exist some studies focusing on improving the ML model [15–17] or system performance [18, 19] using reinforcement learning. In this paper, we use a DQN model (in Subsection 6.2) to select k exercises from the candidate exercise set to ensure the difficulty index and the discrimination index of the exercise collection meet the teacher's requirements.

3 Problem Definition

Definition 1 (Exercise). *An exercise* $e_j = \{w_1^{(j)}, w_2^{(j)}, \cdots, w_M^{(j)}\}$ *is formed with a sequence of words as its text. In addition, each exercise is tagged with a set of knowledge concept labels, denoted as* $KC_j = \{c_1, c_2, \cdots, c_k\}$*, which is provided by experts and each label* $c \in KC_j$ *represents a knowledge concept involved in exercise* e_j*.*

Example 1. We consider the following math exercise. "x and y are both complex numbers. It is known that x and y are mutually conjugate. Also, $x-y=6, xy=10$. Find the sum of x and y." The knowledge concept set of this exercise could be {"Complex Number", "Complex Conjugate", "Arithmetic of Complex Numbers", "Equations with Complex Numbers"}.

Definition 2 (Student History Interaction Record). *In the online educational system, students will do exercises (either self-determined or assigned by the teacher) to develop their skills. The system records the exercising history of each student* s_i*, denoted as a list* $H(s_i) = \left(\left(e_{k_1}, a^i_{k_1}\right), \left(e_{k_2}, a^i_{k_2}\right), \cdots, \left(e_{k_L}, a^i_{k_L}\right)\right)$*, where each item represents that the exercise has been done by the student and the score is* a^i_j*. Generally, if* s_i *answers* e_j *right,* $a^i_j = 1$*; otherwise,* $a^i_j = 0$*.*

Definition 3 (Exercise Collection). *An exercise collection* $C = \left\{e_1, e_2, \cdots, e_{|C|}\right\}$ *consists of a set of exercises, which is usually collected by a teacher as a homework assignment or a quiz paper.*

Apparently, in teaching practice, the exercises in the collection cannot all be extremely too difficult or too easy for students to solve; otherwise the collection will have little effect on developing students' skills. In order to evaluate the quality of the exercise collection, we adopt two widely-used metrics in education: difficulty index and discrimination index [20, 21].

Definition 4 (Difficulty Index). *Given a student set* S *and an exercise collection* C*, the difficulty index is defined as the proportion of the average score to the overall score. Formally,*

$$P(C,S) = \frac{\sum\limits_{s_i \in S} \sum\limits_{e_j \in C} a^i_j}{|C| \times |S|}.$$

The difficulty index measures the difficulty level of the whole collection. The possible value varies from 0 to 1, where a higher value represents an easier collection.

Definition 5 (Discrimination Index). *Given a student set* S *and an exercise collection* C*, a high score group and a low score group are collected in equal amounts from the student set. Then the discrimination index is defined as the difference of the difficulty index between the high score group and the low score group. Formally,*

$$\begin{aligned} D(C,S) &= P(C,S_H) - P(C,S_L) \\ &= \frac{\sum\limits_{e_j \in C} \left(\sum\limits_{s_i \in S_H} a^i_j - \sum\limits_{s_i \in S_L} a^i_j \right)}{|C| \times |S_H|}, \end{aligned}$$

where S_H *and* S_L *denote the high and the low score groups respectively, such that*

1) $S_H, S_L \subset S$;

2) $|S_H| = |S_L| = \lceil \tau |S| \rceil$*, where* τ *is set to* 27% *in practice*;

3) $\forall s \in S_H, \forall s' \in S \backslash S_H, P(C,\{s\}) \geqslant P(C,\{s'\})$;

4) $\forall s \in S_L, \forall s' \in S \backslash S_L, P(C,\{s\}) \leqslant P(C,\{s'\})$.

The discrimination index measures the degree to which the collection discriminates among students. The value ranges from 0 to 1, where a higher value indicates more discrimination of the exercise collection.

The difficulty index and the discrimination index are widely used by teachers to assess whether a quiz paper (i.e., exercise collection) is qualified (i.e., with reasonable difficulty and good discrimination ability). Therefore, when assembling an exercise collection, teachers will expect a result that meets the target values of difficulty p and discrimination index d. For example, a teacher may want an exercise collection that has a difficulty index of 0.55 and a discrimination index of 0.6. Therefore, given a student subset (i.e., students in the teacher's class) and an exercise subset where none of the exercises has been done by these students, the

framework needs to select several exercises and assemble an exercise collection that meets the teacher's requirements on the difficulty index and the discrimination index.

Next, we formally define the exercise collection auto-assembling problem.

Exercise Collection Auto-Assembling Problem. In an online educational platform, there is an exercise set E and a student set S with their interaction history $H = \{H(s_i)|s_i \in S\}$. Given a student subset $S' \subset S$, an exercise subset $E' \subset E$ (where $\forall e \in E', \forall s \in S', e \notin H(s)$), a target collection size k, a target difficulty index p and a target discrimination index d as query parameters, the system aims to collect the optimal exercise collection $C^* \subset E'$ containing k exercises that has the smallest difference from p in the difficulty index and d in the discrimination index. Formally,

$$C^* = \underset{\{C|C\subset E' \wedge |C|=k\}}{\operatorname{argmin}} (P(C,S') - p)^2 + (D(C,S') - d)^2. \quad (1)$$

Example 2. Considering an exercise subset $E' = \{e_1, e_2, e_3\}$, and $k = 2, p = 0.55, d = 0.6$, there will be three candidate collections: $C_1 = \{e_1, e_2\}, C_2 = \{e_1, e_3\}$, and $C_3 = \{e_2, e_3\}$. Suppose the result shows that $P(C_1, S') = 0.6$ and $D(C_1, S') = 0.6$; $P(C_2, S') = 0.6$ and $D(C_2, S') = 0.55$; $P(C_3, S') = 0.58$ and $D(C_3, S') = 0.61$. Then we should report $C^* = C_3$ as the optimal exercise collection.

4 Overview

At a high level, the exercise collection auto-assembling framework can be divided into two stages, as shown in Fig.1.

4.1 Answer Prediction

In the first stage (i.e., the answer prediction stage), the main task is to predict the correctness of the students' answer on each candidate exercise in E' based on the history interaction records of students in S'. This stage can be accomplished by training a knowledge tracing model. By using the well-trained knowledge tracing model, we can predict the probability of each student $s \in S'$ correctly answering each exercise $e \in E'$.

In addition, since the knowledge tracing model focuses on modeling the knowledge state of students and lacks the representation of the features of the exercises (only the knowledge concepts and the text of the exercises are included), we propose to introduce exercise embeddings into the model to improve the prediction accuracy of the model (Subsection 5.1). In order to accelerate the convergence of the knowledge tracking model, a pre-training model of the exercise embeddings is also designed in Subsection 5.2.

4.2 Collection Assembling

In the second stage (i.e., the collection assembling stage), the main task is to select a certain number of exercises from the candidate exercise set E' according to the predicted students' answers in the previous stage to form an exercise collection that satisfies the teacher's requirements on the difficulty index and the discrimination index.

First, by binarizing the predicted probabilities (that

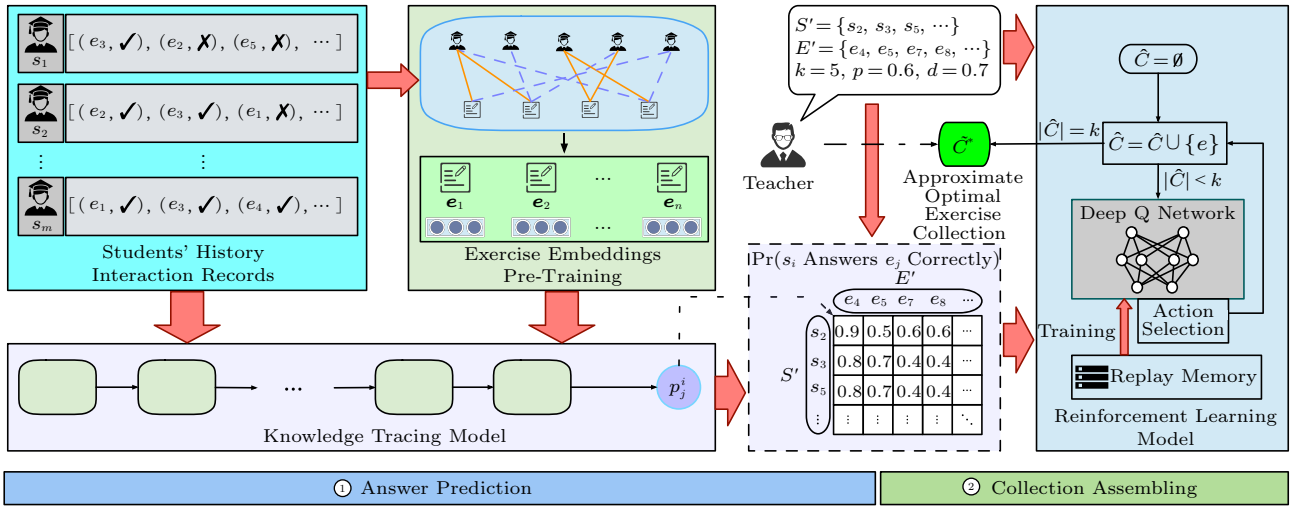

Fig.1. Overview of the exercise collection auto-assembling framework.

is, the probability greater than 0.5 is converted to 1; otherwise it is converted to 0), the predicted values of the knowledge tracing model can be used as an estimate of the students' answers.

However, considering that the predictions are not 100% accurate, it is necessary to minimize the influence of the wrong predictions on exercise collection assembling. Therefore, we introduce the concept of answer prediction confidence in this stage (Subsection 6.1). By selecting exercises with predictions of high confidence, the influence of wrong predictions can be reduced.

In addition, to assemble the "optimal" exercise collection that best meets the teacher's requirements needs to traverse all the candidate exercise collections, which will result in huge time overhead and thus is unacceptable when the number of candidate exercises (i.e., $|E'|$) is large. Therefore, we propose to apply a reinforcement learning model in this stage, which iteratively selects k exercises from the candidate exercise set $|E'|$. The reinforcement learning model can ensure that the assembled exercise collection meets the teacher's requirements through self-supervised learning while the time to assemble an exercise collection is acceptable. The details of the reinforcement learning model will be described in Subsection 6.2.

5 Answer Prediction Using Knowledge Tracing Model

In this section, we introduce the structure of the knowledge tracing model used to predict the correctness of a student's answer to an exercise (Subsection 5.1). We also propose the exercise embeddings technique to enhance the model and provide a pre-training method for exercise embeddings (Subsection 5.2).

5.1 Knowledge Tracing Model with Exercise Embeddings

Knowledge Tracing Model. Given the history interaction records of a student, the knowledge tracing model can predict the correctness of the student's answer to the next exercise. With the development of deep learning and neural network, many studies on deep learning based knowledge tracing models [1–4] are proposed and show better performance than traditional methods (e.g., Bayesian knowledge tracing). Although existing deep learning based knowledge tracing models use different techniques in detail, most of them have similar model structures and almost the same input and output.

Fig.2 shows the common structure of the knowledge tracing model. For a student s_i, the knowledge tracing model will intercept a sequence from his/her history interaction records, taking the features of each exercise (i.e., the knowledge concepts KC_j and/or the exercise text e_j) and the student's answer a_j^i as model inputs. The model then will learn a sequence of hidden states of the student $\boldsymbol{h}_j$, representing the student's knowledge state after exercise e_j is done. After the last exercise e_T is inputted into the model, it uses the student's latest hidden state $\boldsymbol{h}_T$ and features of the next exercise e_{T+1} to predict whether the exercise will be answered correctly.

Answer Prediction. The knowledge tracing model will be trained on all the history interaction records of all students in S. After the knowledge tracing model is trained, given a student subset S' and an exercise subset E' by the teacher, the model can predict the probability of student $s_i \in S$ correctly answering the exercise $e_j \in E'$, (denoted as p_j^i) by inputting the history records $H(s_i)$ and the information of exercise e_j

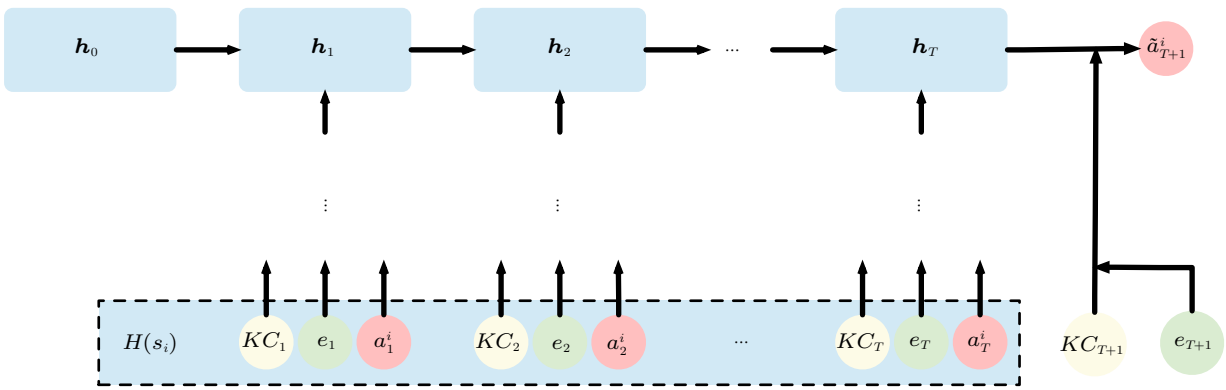

Fig.2. Simplified structure of knowledge tracing models.

into the model. For convenience, all probabilities are represented by matrix $\boldsymbol{P} \in \mathbb{R}^{|S'|\times|E'|}$, where the element of the i-th row and the j-th column is p_j^i.

However, we argue that these models miss some features of exercises. Since most of them only take knowledge concepts of the exercise as input, the model cannot effectively distinguish different exercises with the same knowledge concepts. Although EKT [4] introduces the exercise text into the model as additional model input, it still cannot recognize these features of the exercise, for instance, whether the exercise contains "traps" for students, the complexity of the exercise calculation, and required reasoning skills.

Exercise Embeddings. In order to address these limitations, we propose a simple but effective modification, in which an embedding vector is created for each exercise to embed the extra features of the exercise. The exercise embeddings can carry the similarities and differences among exercises. Formally, Given an exercise set E, we embed each exercise $e_j \in E$ into a d-dimensional vector space as a vector $\boldsymbol{e}_j$. The embedding matrix is denoted as $\boldsymbol{M}_E = \left(\boldsymbol{e}_1; \boldsymbol{e}_2; \cdots; \boldsymbol{e}_{|E|}\right)$. When an exercise needs to be inputted into the knowledge tracing model, the corresponding embedding vector will be concatenated with the representation of knowledge concepts and exercise text to learn the knowledge state of the student. The exercise embeddings will also be treated as parameters of the knowledge tracing model, which will be updated during the training process.

The exercise embeddings can provide more side information to the knowledge tracing model, which will enhance the model's fitting ability, and it can be easily applied to most of the existing deep learning based knowledge tracing models (e.g., DKT, DKVMN, and EKT).

However, if the exercise embeddings are initialized randomly, it may take a long time for the model to learn the similarities and differences among these exercises. Therefore, we also propose a pre-training method in Subsection 5.2 to capture the feature of the exercise ahead and accerlate the training process.

5.2 Exercise Embeddings Pre-Training

As discussed above, exercise embeddings are used to represent the similarity between exercises. In order to accelerate the training process, we design a method to pre-train the exercise embeddings.

To pre-train the exercise embeddings, we consider the relationship between exercises and students. Given all the history interaction records, there are two possible types of connection between a student and an exercise: right answer and wrong answer. Thus, for each exercise e_j, we maintain two student sets $S_R(e_j)$ and $S_W(e_j)$, where $S_R(\cdot)$ contains all students who answered this exercise right and $S_W(\cdot)$ contains all students who answered this exercise wrong.

Exercise Similarity. Inspired by the idea of the second-order proximity in LINE [22], we evaluate the similarity of the exercises by the similarity of the connected student sets. Specifically, for two exercises, if there are students who answered both of them right or wrong, the two exercises should be considered with a larger similarity; but if there are students who answered one exercise right and the other wrong, they should be considered with a smaller similarity.

As a consequence, the similarity of exercises e_i and e_j is defined as follows.

$$r_{ij} = \frac{|S_R(e_i) \cap S_R(e_j)| + |S_W(e_i) \cap S_W(e_j)|}{|S|} - \frac{|S_R(e_i) \cap S_W(e_j)| + |S_W(e_i) \cap S_R(e_j)|}{|S|}, \quad (2)$$

where $|S|$ is the normalizing factor, mapping the value range of r_{ij} into $[-1, 1]$.

Next, we illustrate the process of computing the exercise similarity using an example.

Example 3. There are three exercises and four students in the example dataset, as shown in Fig.3. In the figure, a solid line between an exercise and a student represents the student answering this exercise right in the history interaction records while a dotted line represents a wrong answer. Then we compute $S_R(e_1) = \{s_1, s_2\}, S_W(e_1) = \{s_3\}; S_R(e_2) = \{s_2, s_4\}, S_W(e_2) = \{s_3\}; S_R(e_3) = \{s_3\}, S_W(e_3) = \{s_4\}$. Therefore the similarity between e_1 and e_2 can be further computed as $r_{1,2} = (1 + 1 - 0 - 0)/4 = 0.5$ since they both have s_2 in the S_R set and s_3 in the S_W set but no common students in their opposite sets. Similarly, $r_{1,3} = (0 + 0 - 1 - 0)/4 = -0.25$ and $r_{2,3} = (0 + 0 - 1 - 1)/4 = -0.5$.

Pre-Training Model. After the similarity between exercises is defined, we use inner products of the corresponding exercise embeddings to estimate the similarity in the embedding space,

$$\hat{r}_{ij} = \tanh\left(\boldsymbol{e}_i \cdot \boldsymbol{e}_j\right),$$

where $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is the hyperbolic tangent function. Note that we do not use the Sigmoid function like LINE because the value range of the tanh function is $(-1, 1)$, which matches the range of r_{ij}.

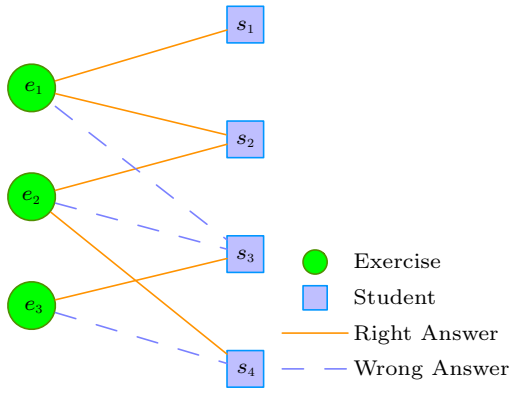

Fig.3. Relationship between exercises and students.

We then use a modified cross-entropy loss function to enforce the model $\hat{r}_{ij}$ to be close to the defined similarity r_{ij},

$$\mathcal{L}(E, \boldsymbol{M}_E) = -\frac{1}{2}\sum_{i=1}^{|E|}\sum_{j=1}^{|E|}((1-r_{ij})\log(1-\hat{r}_{ij}) + (1+r_{ij})\log(1+\hat{r}_{ij})).$$

We apply the loss function to the pre-training model and use the stochastic gradient descent algorithm to continuously update the exercise embeddings. Algorithm 1 illustrates the pre-training process of the exercise embeddings. After the pre-training, we can feed the embedding vector into the knowledge tracing model as discussed in Subsection 5.1.

Algorithm 1. Pre-Training of Exercise Embeddings

Input: E: the exercise set; S: the student set;
d: dimension of the exercise embedding vector;
$iter$: number of training iterations (epochs);
α: learning rate
Output:
$\boldsymbol{M}_E = (\boldsymbol{e}_1; \boldsymbol{e}_2; \cdots; \boldsymbol{e}_{|E|})$: the pre-trained embedding matrix

Randomly initialize the embedding matrix $\boldsymbol{M}_E$ of size $|E| \times d$
foreach $e_j \in E$ **do**
  $S_W(e_j) = S_R(e_j) = \emptyset$
foreach $s_i \in S$ **do**
  foreach $(e_j, a_j^i) \in H(s_i)$ **do**
    if $a_j^i == 1$ **then**
      $S_R(e_j)$.add(s_i)
    else
      $S_W(e_j)$.add(s_i)
for epoch $= 1 \to iter$ **do**
  foreach $e_i, e_j \in E$ **do**
    Compute r_{ij} by (2)
    $\hat{r}_{ij} = \tanh(\boldsymbol{e}_i \cdot \boldsymbol{e}_j)$
    $\mathcal{L} = -\frac{1}{2}((1-r_{ij})\log(1-\hat{r}_{ij}) + (1+r_{ij})\log(1+\hat{r}_{ij}))$
    $\boldsymbol{M}_E = \boldsymbol{M}_E - \alpha\frac{\partial \mathcal{L}}{\partial \boldsymbol{M}_E}$
return $\boldsymbol{M}_E$

6 Exercises Collection Assembling Using Deep Reinforcement Learning

In this section, we first discuss the criteria for assessing how well an exercise collection meets the requirements when only the probabilities of each student correctly answering each exercise are known in Subsection 6.1. Then we propose the deep reinforcement model to assemble the exercise collection in Subsection 6.2.

6.1 Estimated Optimal Exercise Collection

As proposed in Subsection 5.1, given the student subset S' and the exercise subset E', we can use the knowledge tracing model to predict the probability matrix $\boldsymbol{P}$, where each element p_j^i denotes the probability of student $s_i \in S'$ answering exercise $e_j \in E'$ correctly.

Then we can predict the student answer $\tilde{a}_j^i$ based on the corresponding probability p_j^i by a simple transformation: if $p_j^i > 0.5$, $\tilde{a}_j^i = 1$; otherwise $\tilde{a}_j^i = 0$. All the predicted answers form as a matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{|S'|\times|E'|}$, where the element of the i-th row and the j-th column is $\tilde{a}_j^i$.

Answer Prediction Confidence. Although the predicted student answers are obtained, it is not appropriate to use them as the sole basis for exercise collection assembling. The reason is that the knowledge tracing model does not have accurate predictions on all answers and the probability p_j^i outputted by the model actually reflects the model's confidence in the prediction of this answer. For instance, supposing the knowledge tracing model predicts $p_1^1 = 0.9$ and $p_2^1 = 0.55$, although both of these answers will be predicted as correct (i.e., $\tilde{a}_1^1 = \tilde{a}_2^1 = 1$), the model is more confident about the success on the former prediction.

We utilize the entropy to represent the answer prediction confidence. Formally, given a student answer prediction $\tilde{a}_j^i$ with a probability p_j^i, the confidence of the prediction is defined as:

$$Conf(\tilde{a}_j^i) = 1 - p_j^i\log(p_j^i) - (1-p_j^i)\log(1-p_j^i),$$

where a higher value indicates more confidence on the prediction. Further, given the student subset S' and an exercise collection $C \subset E'$, the confidence of the prediction on the whole answer matrix $\tilde{\boldsymbol{A}}$ is defined as follows.

$$Conf(C, S') = \frac{\sum_{s_i \in S'}\sum_{e_j \in C} Conf(\tilde{a}_j^i)}{|C| \times |S'|}.$$

Estimated Optimal Exercise Collection. In order to evaluate whether an exercise collection matches the teacher's requirements on the difficulty index and the discrimination index, we can predict the difficulty index and the discrimination index of the exercise collection by (3) and (4) respectively.

$$\tilde{P}(C,S)=\frac{\sum_{s_i\in S}\sum_{e_j\in C}\tilde{a}_j^i}{|C|\times|S|}. \tag{3}$$

$$\tilde{D}(C,S)=\tilde{P}(C,S_H)-\tilde{P}(C,S_L). \tag{4}$$

Meanwhile, since the above indices are computed based on the predicted answers of the knowledge tracing, it is reasonable to also consider the prediction confidence when evaluating an exercise collection. Thus, we use (5) to evaluate an exercise collection $C\subset E'$.

$$\begin{aligned}\mathcal{R}(C|S',E',\boldsymbol{P},p,d)=&(\tilde{P}(C,S')-p)^2+\\&(\tilde{D}(C,S')-d)^2+\\&\lambda\times Conf(C,S'),\end{aligned} \tag{5}$$

where λ is a coefficient controlling the trade-off between the difference on two indices and the prediction confidence. A smaller value of $\mathcal{R}(C|S,E,\boldsymbol{P},p,d)$ indicates an exercise collection that meets the requirements better.

Therefore, under the answer prediction of the knowledge tracing model, given the collection size k, we can use the following formula to select the estimated optimal exercise collection $\tilde{C}^*$.

$$\tilde{C}^*=\text{argmin}_{\{C|C\subset E'\bigwedge|C|=k\}}\mathcal{R}(C|S',E',\boldsymbol{P},p,d). \tag{6}$$

However, if we directly use (6) to retrieve the estimated optimal solution, we will have to traverse all $C^k_{|E'|}$ candidates (e.g., over 10^{35} when $|E'|=500$ and $k=20$), which may take an impractical long time to process. Therefore, in Subsection 6.2, we propose a deep reinforcement learning based approach to select appropriate exercises and return the collection.

6.2 Exercise Collection Assembling Model

Naïvely assembling the estimated optimal exercise collection is rather expensive in terms of time. However, we can model the exercise collection assembling as the following process. 1) Initially, the exercise collection is created as an empty set and the exercise candidate set is created as the exercise subset E'. 2) At each step, one exercise is selected by the model and then removed from the candidate set, and added into the collection. 3) Finally, when the exercise collection contains k exercises, it will be reported as output.

The above process can be treated as a Markov decision process, in which the state at the current moment is only influenced by the previous state and the chosen action. Note that when the model selects an exercise, it needs to consider the long-term reward brought by the exercise, that is, the effect of the exercise in the final exercise collection. Therefore, we use a DQN [23] to model the process and generate an approximate optimal solution.

States. In this problem, a state is an exercise collection (that may not be fully collected), denoted as $\hat{C}=\{e_{j_1},e_{j_2},\cdots,e_{j_{|\hat{C}|}}\}$ (s.t. $\hat{C}\leqslant k$), and it is represented by a matrix $\boldsymbol{P}_{\hat{C}}$ of size $|S|\times k$, where the value of each element is as follows.

$$(\boldsymbol{P}_{\hat{C}})_{i,j}=\begin{cases}p_j^i, & \text{if } j\leqslant|\hat{C}|,\\ 0, & \text{otherwise.}\end{cases} \tag{7}$$

To explain (7) in detail, in the matrix $\boldsymbol{P}_{\hat{C}}$, the first $|\hat{C}|$ columns correspond to all exercises in $\hat{C}$ and each column stores the probability of being answered correctly by students (i.e., the corresponding column in the probability matrix $\boldsymbol{P}$); the rest of the columns are filled with zeros, which are equivalent to $k-|\hat{C}|$ hypothetical exercises that cannot be correctly answered by students and will be replaced by other exercises in the future. Specially, the initial state is an empty exercise collection $\hat{C}=\emptyset$, and the state matrix $\boldsymbol{P}_{\hat{C}}=\boldsymbol{0}^{|S'|\times k}$. In order to be further inputted into the DQN model, the representation of the state is vectorized into $vec(\boldsymbol{P}_C)$.

Actions. In this problem, an action is considered as adding an exercise $e_j\in E'\backslash\hat{C}$ into the exercise collection $\hat{C}$. By performing an action, the agent will transition from the current state $\hat{C}$ into another state $\hat{C}'=\hat{C}\cup\{e_j\}$.

Q-Value. In Q-learning [14], the Q-value of a state indicates the "goodness" of the state. In this problem, we define the Q-value (i.e., $Q(\hat{C})$) as the expected smallest value of (5) of all the possible exercise collections. Formally, the Q-value can be computed as:

$$Q(\hat{C})=\begin{cases}\mathcal{R}(\hat{C}|S',E',\boldsymbol{P},p,d), & \text{if } |\hat{C}|=k,\\ \min\limits_{e\in E'\backslash\hat{C}}Q(\hat{C}\cup\{e\}), & \text{if } |\hat{C}|<k.\end{cases}$$

However, directly computing the Q-values of all the states is impractical, because the number of the states is rather large. Therefore, the DQN (Deep Q-Network) model is utilized to estimate the Q-value of a state.

DQN Model. The DQN model uses a forward-feed neural network (named Q-network) to estimate the Q-value of a state. The Q-network takes the representation of a state $\hat{C}$ (i.e., vec($\hat{C}$)) and the target values p and d as input, and outputs the estimated Q-value of the state, denoting as $\tilde{Q}(\hat{C};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the parameters of the Q-network.

DQN Model Training. In reinforcement learning, an episode is a complete process of the agent from the initial state (i.e., an empty exercise collection) to a terminating state (i.e., a collection with k exercises). The $\mathcal{R}$-value of the terminating state (computed by (5)) is denoted as r. For a state $\hat{C}$ in the episode, the loss function is defined as (8), indicating the squared error between the target Q-value and the predicted Q-value.

$$\mathcal{L}(\hat{C};\boldsymbol{\theta}) = (\min(r, \min_{e\in E'\setminus\hat{C}} \tilde{Q}(\hat{C}\cup\{e\};\boldsymbol{\theta})) - \tilde{Q}(\hat{C};\boldsymbol{\theta}))^2. \quad (8)$$

The DQN model can automatically acquire the training data by observing many episodes. Then all the observed training data (i.e., transitions) will be stored in an experience replay memory [23]. Batches of the training data are randomly sampled from the replay memory to avoid data correlation, and then fed into the Q-network. The stochastic gradient descent algorithm is used on the loss function to update the DQN parameters. The model training process is shown in Fig.4.

DQN Model Inference. In the inference stage, the initial state is created as an empty exercise collection. Then, the state will be iteratively updated based on the following policy function:

$$\pi(\hat{C}) = \operatorname*{argmin}_{e\in E\setminus\hat{C}} Q(\hat{C}\cup\{e\}).$$

Each updating operation selects an exercise $\pi(\hat{C})$ and adds it into the current state $\hat{C}$. After k updates, the model will report the terminating state (containing k exercises) as the output.

7 Experiments

7.1 Dataset and Experimental Settings

Dataset. The dataset is supplied by TAL Education Group① and collected from XueErSi②, which is a widely-used online learning system in China. The details of the dataset are listed in Table 1.

Experimental Settings. The experiments in the paper are implemented using PyTorch 1.11.0 with

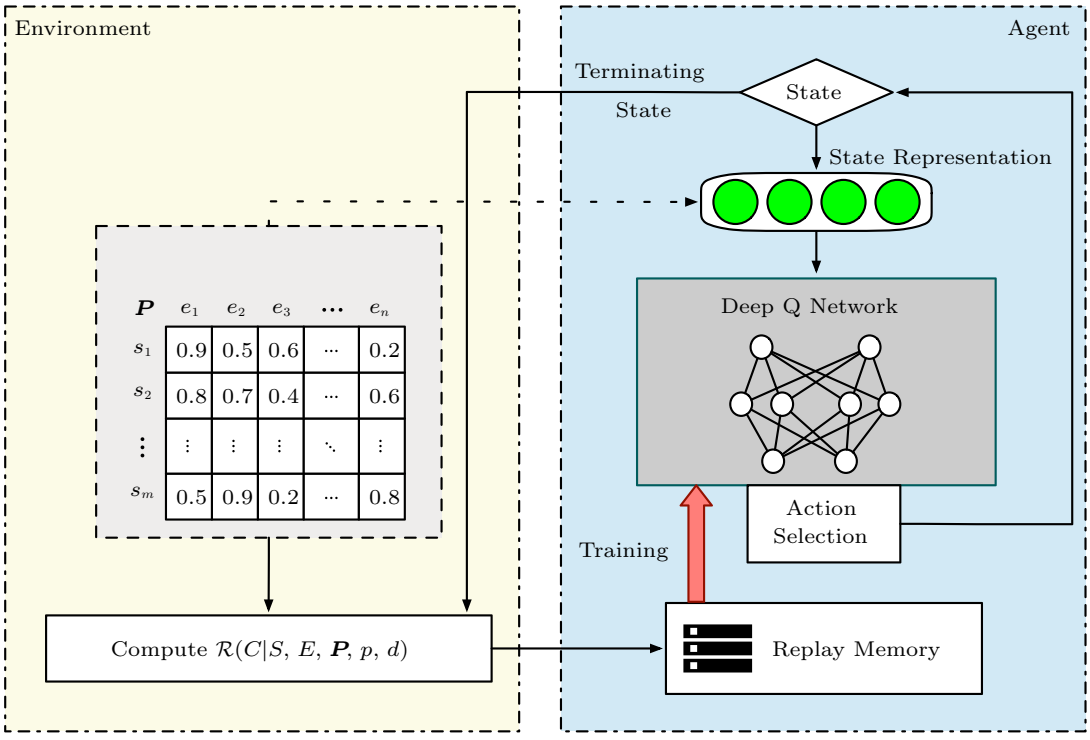

Fig.4. DQN model training.

①https://en.100tal.com/, Aug. 2022.
②https://www.xueersi.com/, Aug. 2022.

Table 1. Summary of the Dataset

Dataset	#Students	#Exercises	#Knowledge Concepts	#Interactions	#Average Exercises per Student
Training	4 918	1 948	86	1 130 651	229.9
Test	374	674	86	252 076	674.0

Note: # means "number of".

Python 3.9. All the experiments are conducted in a machine with 3.10 GHz Intel Xeon CPU 6242R, 256 GB RAM, and an NVIDIA RTX 3090 GPU, running Ubuntu 20.04.

Implementation Details. The exercise embeddings dimension is set to 256. DQN is implemented by a 3-layer neural network, where the sizes of the hidden layers are 512 and 256 respectively. All the parameters of the models are initialized with Gaussian distributions. An Adam [24] optimizer is used to train the exercise pre-training model and the knowledge tracing model, where the settings are: initial learning rate $\alpha = 0.005$ and momentum parameters $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The learning rate is decayed by $\gamma = 0.5$ during training. An RMSProp optimizer is used to train the deep Q-network. The number of exercises in an exercise collection is set as $k = 20$. The coefficient of the answer prediction confidence is set as $\lambda = 0.05$.

7.2 Evaluating Answer Prediction

In this subsection, we evaluate the effectiveness of the exercise embeddings and the pre-training model proposed in this paper.

Comparison Methods. In the experiments, we implement the following different knowledge tracing models.

• *DKT* [1]. It only takes the knowledge concepts of exercises as the model input and an RNN model is used to represent the student's state and predict the answers.

• *DKVMN* [2]. It also only takes the knowledge concepts of exercises as the model input. It applies a memory module to store the knowledge concepts and the mastery of the student.

• *EKT* [4]. It takes the knowledge concepts and the text of exercises as the model input. A BiLSTM model is used to encode the exercise text. Like DKVMN, it also uses a memory module to encode the knowledge concepts.

For each model, we create three variants: 1) Vanilla Model: the original model structure; 2) Model+Embeddings: the proposed exercise embeddings are also inputted into the knowledge tracing model and the embeddings are randomly initialized; 3) Model+Embeddings+Pre-Train: the exercise embeddings are first pre-trained before the training of the knowledge tracing model. We evaluate the performance of these models by measuring the AUC (area under curve) and the accuracy of answer prediction. All the models are trained for 50 epochs.

Effectiveness on Answer Prediction. The experimental results are shown in Fig.5. We make the following observations. 1) EKT performs better than DKT and DKVMN (about 3%–6% on AUC and accuracy). That is because EKT takes the extra exercise text information into the model. 2) After applying exercise embeddings, all these three models have a slight improvement in AUC and accuracy (about 1%–3%). The reason is that the exercise embeddings learn to rep-

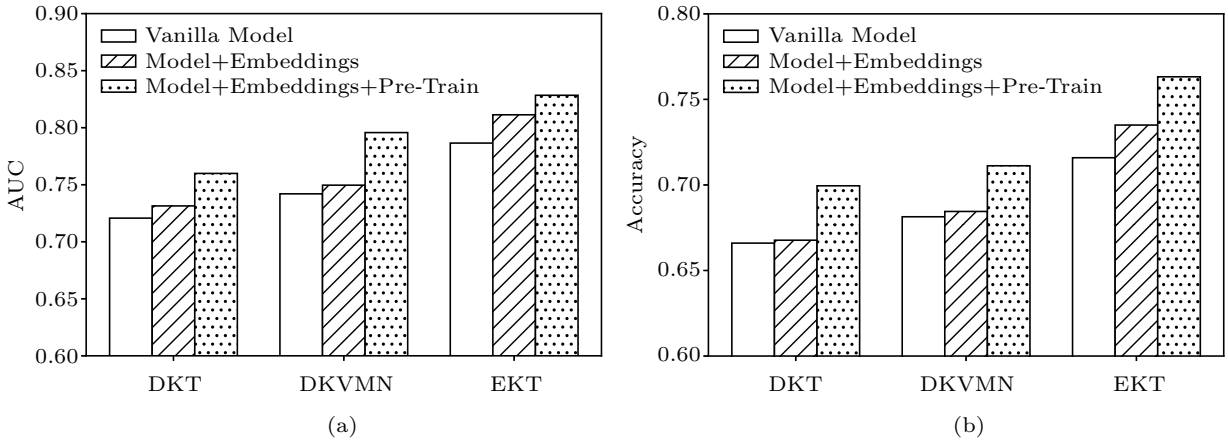

Fig.5. Experimental results on answer prediction. (a) AUC. (b) Accuracy.

resent other features of the exercise other than knowledge concepts and text. 3) Models with pre-trained exercise embeddings have a higher AUC and a higher accuracy than those with randomly initialized exercise embeddings. That is because the proposed pre-training model for exercise embeddings can well capture the relationship among exercises, thus reducing the burden of training the exercise embeddings for the knowledge tracing model.

7.3 Evaluating Exercise Collection Assembling

In this subsection, we evaluate the effectiveness and the efficiency of the exercise collection assembling model proposed in this paper.

Comparison Method. In the experiments, we implement a heuristic method HEU as the baseline and compare it with the reinforcement learning based exercise collection assembling model proposed in the paper.

- *DQN.* The model is proposed in Subsection 6.2.
- *HEU.* Initially, k exercises are randomly selected from the set E' and assembled into the initial exercise collection C. Then, we update the collection by removing an exercise from the collection and adding an exercise from outside the collection. To decide which exercise to remove and which exercise to add, among all cases (i.e., exercise pairs), we choose the one that gives the greatest descent in $\mathcal{R}$-value (computed by 5) of the updated collection. The exercise collection is iteratively updated until the $\mathcal{R}$-value stops descending. Finally, the updated exercise collection is reported as output.
- *DQN w/o Conf.* The DQN model without the answer prediction confidence is proposed in Subsection 6.1.
- *HEU w/o Conf.* It is the HEU model without the answer prediction confidence.

Effectiveness on Exercise Collection Assembling. The average errors on the difficulty index and the discrimination index are shown in Fig.6 and the overall errors (average sum of squares for two errors) are shown in Fig.7. We make the following observation. 1) DQN has smaller errors than HEU on the difficulty index and the discrimination index. The reason is that the DQN model can automatically learn from the experience and try learning to assemble a global optimal exercise collection, but the heuristic baseline can only obtain a local optimal exercise collection from a random initialization solution. 2) DQN and HEU have much smaller errors than DQN w/o conf and HEU w/o conf correspondingly. That is because the proposed answer prediction confidence can be relatively efficient to capture the difference between confident predictions of the model and unconfident ones. Without it, the model is likely to select exercises with many mispredictions on students' answers into the exercise collection, thus resulting in a large error on the difficulty index and the discrimination index.

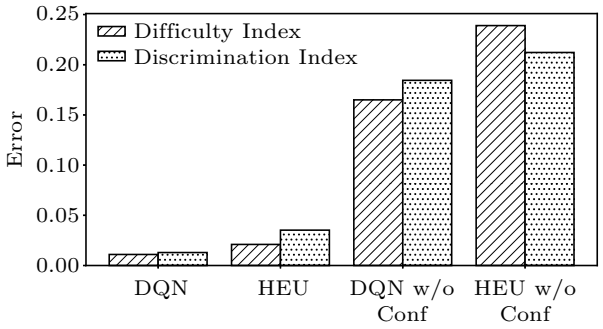

Fig.6. Average error on the difficulty index and the discrimination index.

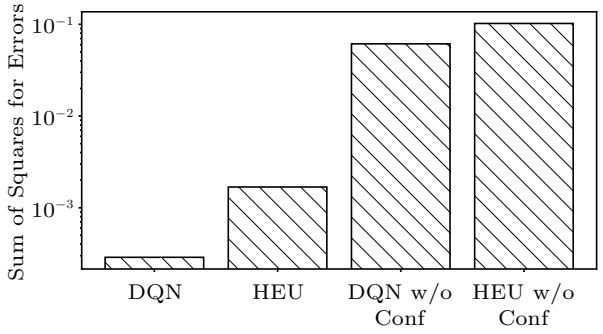

Fig.7. Average sum of squares for errors.

Effect of λ. We study the effect of the coefficient of the answer prediction confidence λ and the overall errors of the assembled exercise collection with different values of λ are shown in Fig.8. We make the following observation. 1) First, as the value of λ gets larger (i.e., from 0.005 to 0.05), the error gets smaller. The reason is that a small λ indicates the answer prediction confidence has not been given enough attention when assembling the exercise collection and thus some of the selected exercises may still contain many wrong predictions. 2) As λ continues to increase (i.e., from 0.05 to 5), the errors get larger. The reason is that a large λ indicates the model pays too much attention to selecting exercises with few wrong predictions, and ignoring the difference of the difficulty index and the discrimination index of the exercise collection from the target values.

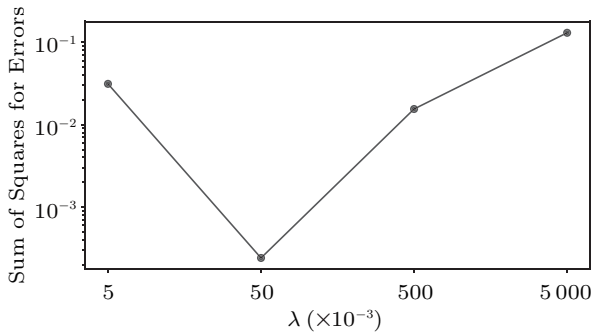

Fig.8. Effect of λ.

Efficiency on Exercise Collection Assembling. Fig.9 shows the average running time of DQN and HEU with a varying number of exercises in E'. We make the following observation. 1) DQN takes much less running time (about 0.01x) than HEU. The reason is that i) HEU may take many iterative steps to reach the local optimal but DQN only takes k steps to select exercises and assemble collection; and ii) as a deep learning based model, DQN can utilize the parallel computing power of GPU to speed up the model. 2) Both DQN and HEU require longer running time to assemble the exercise collection as the number of exercises in E' increases. That is because both methods will have to select exercises from more candidates, which increases the time cost of each decision step in both DQN and HEU. 3) As the number of candidate exercises increases from 100 to 300, the increasing trend of HEU running time (about 130%) is faster than that of DQN (about 39%). The reason is that with more candidate exercises in E', HEU usually has to iterate over much more steps to converge while DQN only takes fixed k selection actions to accomplish the assembling process.

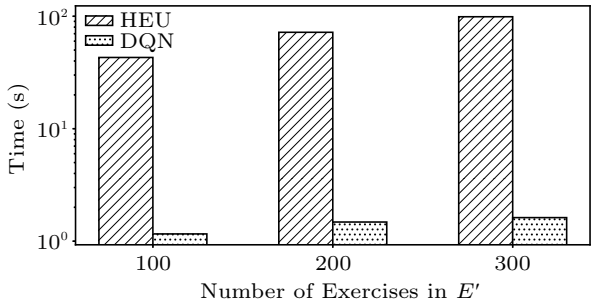

Fig.9. Efficiency on exercise collection assembling.

In summary, the exercise collection auto-assembling framework proposed in the paper can output exercise collections that are close to the teacher's requirements on the difficulty index and the discrimination index, and the proposed framework is proved to be effective.

8 Conclusions

In this paper, we studied the problem of exercise collection auto-assembling, where the difficulty index and the discrimination index of the assembled exercise collection are close to teachers' requirements. We proposed a two-stage framework to solve the problem. First, a knowledge tracing model is utilized to predict students' answers to exercises based on their history interaction records. We also introduced exercise embeddings and a pre-training approach to enhance the model. This approach can improve the accuracy on answer prediction by about 3%–5%. Then, we proposed a deep reinforcement learning model to select exercises and assemble a collection based on the predicted students' answers, which enables the framework to assemble the exercise collection in a short time (about 100 times faster than a heuristic baseline) and with a small error (about 0.02) on the difficulty index and the discrimination index. In conclusion, the proposed framework can auto-assemble exercise collections effectively and efficiently while satisfying the teacher's requirements on the difficulty index and the discrimination index.

References

[1] Piech C, Bassen J, Huang J, Ganguli S, Sahami M, Guibas L J, Sohl-Dickstein J. Deep knowledge tracing. In *Proc. the Annual Conference on Neural Information Processing Systems*, Dec. 2015, pp.505-513.

[2] Zhang J, Shi X, King I, Yeung D. Dynamic key-value memory networks for knowledge tracing. In *Proc. the 26th International Conference on World Wide Web*, Apr.2017, pp.765-774. DOI: 10.1145/3038912.3052580.

[3] Su Y, Liu Q, Liu Q, Huang Z, Yin Y, Chen E, Ding C H Q, Wei S, Hu G. Exercise-enhanced sequential modeling for student performance prediction. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, Feb. 2018, pp.2435-2443. DOI: 10.1609/aaai.v32i1.11864.

[4] Liu Q, Huang Z, Yin Y, Chen E, Xiong H, Su Y, Hu G. EKT: Exercise-aware knowledge tracing for student performance prediction. *IEEE Trans. Knowl. Data Eng.*, 2021, 33(1): 100-115. DOI: 10.1109/TKDE.2019.2924374.

[5] Chai C, Li G, Li J, Deng D, Feng J. Cost-effective crowdsourced entity resolution: A partial-order approach. In *Proc. the 2016 ACM International Conference on Management of Data*, June 26-July 1, 2016, pp.969-984. DOI: 10.1145/2882903.2915252.

[6] Li G, Chai C, Fan J, Weng X, Li J, Zheng Y, Li Y, Yu X, Zhang X, Yuan H. CDB: Optimizing queries with crowd-based selections and joins. In *Proc. the 2017 ACM International Conference on Management of Data*, May 2017, pp.1463-1478. DOI: 10.1145/3035918.3064036.

[7] Chai C, Fan J, Li G. Incentive-based entity collection using crowdsourcing. In *Proc. the 34th IEEE International Conference on Data Engineering*, Apr. 2018, pp.341-352. DOI: 10.1109/ICDE.2018.00039.

[8] Chai C, Cao L, Li G, Li J, Luo Y, Madden S. Human-in-the-loop outlier detection. In *Proc. the 2020 ACM SIGMOD International Conference on Management of Data*, Jun. 2019, pp.19-33. DOI: 10.1145/3318464.3389772.

[9] Corbett A T, Anderson J R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 1994, 4(4): 253-278. DOI: 10.1007/BF01099821.

[10] Nakagawa H, Iwasawa Y, Matsuo Y. Graph-based knowledge tracing: Modeling student proficiency using graph neural network. In *Proc. the 2019 IEEE/WIC/ACM International Conference on Web Intelligence*, Oct. 2019, pp.156-163. DOI: 10.1145/3350546.3352513.

[11] Song X, Li J, Lei Q, Zhao We, Chen Y, Mian A. Bi-CLKT: Bi-graph contrastive learning based knowledge tracing. *Knowl. Based Syst.*, 2022, 241: Article No. 108274. DOI: 10.1016/j.knosys.2022.108274.

[12] Song X, Li J, Tang Y, Zhao T, Chen Y, Guan Z. JKT: A joint graph convolutional network based deep knowledge tracing. *Information Sciences*, 2021, 580: 510-523. DOI: 10.1016/j.ins.2021.08.100.

[13] Xue G, Zhong M, Li J, Chen J, Zhai C, Kong R. Dynamic network embedding survey. *Neurocomputing*, 2022, 472: 212-223. DOI: 10.1016/j.neucom.2021.03.138.

[14] Watkins C J C H, Dayan P. Q-learning. *Machine Learning*, 1992, 8(3): 279-292. DOI: 10.1007/BF00992698.

[15] Chai C, Wang J, Luo Y, Niu Z, Li G. Data management for machine learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*. DOI: 10.1109/TKDE.2022.3148237.

[16] Chai C, Liu J, Tang N, Li G, Luo Y. Selective data acquisition in the wild for model charging. *Proceedings of the VLDB Endowment*, 2022, 15(7): 1466-1478. DOI: 10.14778/3523210.3523223.

[17] Liu J, Chai C, Luo Y, Lou Y, Feng J, Tang N. Feature augmentation with reinforcement learning. In *Proc. the 38th IEEE International Conference on Data Engineering*, May 2022, pp.3360-3372. DOI: 10.1109/ICDE53745.2022.00317.

[18] Zhou X, Chai C, Li G, Sun J. Database meets artificial intelligence: A survey. *IEEE Trans. Knowl. Data Eng.*, 2022, 34(3): 1096-1116. DOI: 10.1109/TKDE.2020.2994641.

[19] Yu X, Li G, Chai C, Tang N. Reinforcement learning with Tree-LSTM for join order selection. In *Proc. the 36th IEEE International Conference on Data Engineering*, Apr. 2020, pp.1297-1308. DOI: 10.1109/ICDE48307.2020.00116.

[20] Aggarwal Y P. Statistical Methods: Concepts, Application and Computation. Stosius Incorporated, 1986.

[21] Boopathiraj C, Chellamani K. Analysis of test items on difficulty level and discrimination index in the test for research in education. *International Journal of Social Science & Interdisciplinary Research*, 2013, 2(2): 189-193.

[22] Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. LINE: Large-scale information network embedding. In *Proc. the 24th International Conference on World Wide Web*, May 2015, pp.1067-1077. DOI: 10.1145/2736277.2741093.

[23] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. Playing Atari with deep reinforcement learning. arXiv:1312.5602, 2013. http://arxiv.org/abs/1312.5602, Aug. 2022.

[24] Kingma D P, Ba J. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014. http://arxiv.org/abs/1412.6980, Jul. 2022.

Tian-Yu Zhao received his B.E. degree in software engineering from Beihang University, Beijing, in 2017. He is currently working toward his Ph.D. degree in the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests mainly include intelligent education and spatial databases.

Man Zeng received her M.Ed. degree in Chinese international education from Beijing Normal University, Beijing, in 2019. Her research interests include intelligent education and data visualization.

Jian-Hua Feng received his B.S., M.S., and Ph.D. degrees in computer science from Tsinghua University, Beijing, in 1991, 1993 and 2006 respectively. He is currently working as a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing. His main research interests include large-scale data management.