Personalized Education with Ranking Alignment Recommendation

Haipeng Liu[1], Yuxuan Liu[2], and Ting Long[3] ✉

[1] School of Artificial Intelligence, Jilin University
liuhp22@mails.jlu.edu.cn
[2] School of Artificial Intelligence, Jilin University
liuyuxuan23@mails.jlu.edu.cn
[3] School of Artificial Intelligence, Jilin University
longting@jlu.edu.cn

Abstract. Personalized question recommendation aims to guide individual students through questions to enhance their mastery of learning targets. Most previous methods model this task as a Markov Decision Process and use reinforcement learning to solve, but they struggle with efficient exploration, failing to identify the best questions for each student during training. To address this, we propose Ranking Alignment Recommendation (RAR), which incorporates collaborative ideas into the exploration mechanism, enabling more efficient exploration within limited training episodes. Experiments show that RAR effectively improves recommendation performance, and our framework can be applied to any RL-based question recommender. Our code is available in https://github.com/wuming29/RAR.git.

Keywords: question recommendation · online education · reinforcement learning · ranking.

1 Introduction

Question recommendation provides personalized educational services by recommending different questions to help students improve their mastery of knowledge concepts [10,3,9]. By accessing a student's historical records and learning targets (i.e., the specific knowledge concepts they aim to master), the recommender selects a question from a predefined question set to aid in mastering these targets. After the student answers, the recommender updates the historical record and proceeds with the next question. For example, as shown in Figure 1, Lily aims to learn *Basic Arithmetic* and *Multiples & Factors*. Upon realizing that Lily had not yet mastered *Counting*, the recommender initially recommended questions on *Counting* to help her understand foundational concepts. Then, it recommended questions on *Basic Arithmetic* and *Multiples & Factors* to further support her in mastering her learning target.

To build an efficient recommender and provide personalized educational services, most previous methods [10,3,9] model question recommendation as a

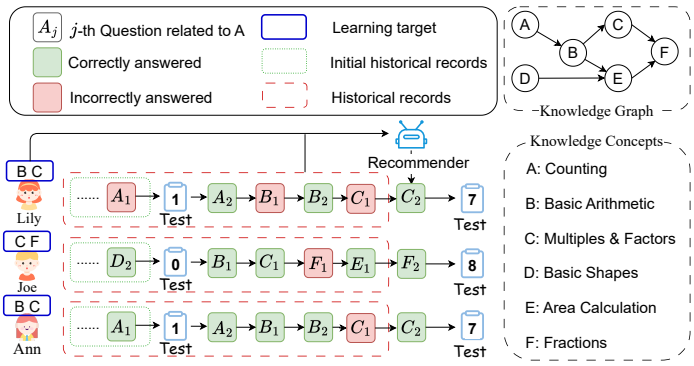

Fig. 1: Illustration of the question recommendation task. The knowledge graph indicates the prerequisite relationships between different knowledge concepts.

Markov Decision Process (MDP) and use reinforcement learning (RL) to maximize students' mastery of learning targets. Specifically, at each training iteration, these methods sequentially recommend questions, collect student feedback, and use this real-time data to optimize the recommendation strategy. By repeatedly performing *recommend questions to collect training data - train the recommender* until the model converges, the recommender can then be used to provide personalized questions for students.

To train a recommender effectively, it is essential to encourage exploration of how different questions impact students' knowledge mastery. Previous studies, such as [10], have used classical reinforcement learning exploration strategies like the ϵ-greedy algorithm, where the recommender randomly selects a question with probability ϵ. These strategies focus on individual students and perform well with small student bases and question sets. However, on large-scale platforms with millions of users and questions, such exploration methods struggle to identify suitable questions within limited training iterations, reducing the performance of the recommender.

To address this issue, we propose incorporating collaborative information into the exploration mechanism to improve efficiency in RL-based recommenders. The collaborative principle suggests that similar users should receive similar recommendations, while different users should get distinct ones. As shown in Figure 1, Lily and Ann, with the same learning targets, should receive the same questions, while Lily and Joe, with different targets, should receive different questions.

Based on this idea, we introduce Ranking Alignment Recommendation (RAR), a novel exploration mechanism that measure differences between students and differences between recommendations, and align the rankings of these differences. This approach encourages exploration with differentiated recommendations for different students while reduces exploration complexity for similar students. RAR consists of two modules: one is the recommendation module, which

can be an arbitrary RL-based recommender responsible for recommending questions to students; the other is the ranking alignment module, which helps the recommendation module explore efficiently by calculating and aligning the differences between students and the differences between recommendations.

In summary, the contributions of our paper are:

- We propose a framework named Ranking Alignment Recommender (RAR), designed to help any RL-based question recommender achieve efficient exploration and enhance performance.
- To our knowledge, we are the first to propose an exploration mechanism based on collaborative information. This new exploration mechanism is expected to benefit any RL-based question recommenders.
- Our extensive experiments across five settings demonstrate the outstanding performance of our method.

2 Preliminaries

2.1 Terminologies

Suppose there is a student $u(u \in \mathcal{U})$ who is learning on the online education platform. We define their **historical records** at time t as $\mathcal{H}_t^u = \{(q_1, y_1), (q_2, y_2), ..., (q_{t-1}, y_{t-1})\}$, where q_i denote the question student u answer at step i, and $q_i \in \mathcal{Q}$. y_i denotes the correctness of the student's response to the question q_i. The **learning target** of student u is the set of questions that the student aims to master, denoted as $\mathcal{T}_u \subseteq \mathcal{Q}$. When the student intends to learn specific concepts, the learning target can also be represented as $\mathcal{T}_u = \bigcup_i \mathbb{C}_i$, where $\mathbb{C}_i$ denotes the set of questions related to target concept i. Following previous works [10,3,9], we define the **learning effect** to evaluate the recommendation performance as:

$$\Delta_u = \frac{m_e - m_b}{m_{sup} - m_b}, \tag{1}$$

where m_e and m_b denote the mastery of the learning target at the beginning and end of the recommendation respectively, and m_{sup} denotes the maximum mastery of the learning target.

2.2 Problem Formulation

In this paper, we aim to design a recommender that, given the historical records $\mathcal{H}_t^u$ and the learning target $\mathcal{T}_u$ of a student u at step t, selects a question q_t from the set of questions $\mathcal{Q}$ to recommend to the student. After the student answers the question q_t, the historical records $\mathcal{H}_t^u$ will be updated based on the student's response y_i, resulting in $\mathcal{H}_{t+1}^u$, which will be used by the recommender for the next step's recommendation. After n rounds of recommendations, the recommendation session ends, and the student's learning effect Δ_u will be calculated. Our goal is to maximize the students' learning effects Δ_u.

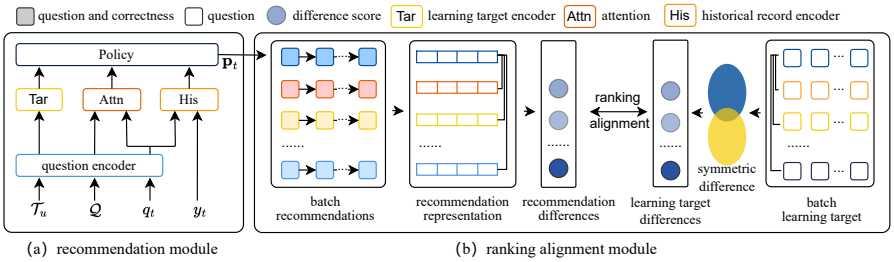

Fig. 2: The pipeline of our method.

This problem is typically formalized as a Markov Decision Process (MDP) [10,9], where the state s_t is generated by the student's historical records $\mathcal{H}_t^u$ and learning target $\mathcal{T}_u$. The action a_t is the question recommended by the recommender based on s_t. The reward r_t represents the reward for taking action a_t in state s_t, which is typically determined based on the learning effect Δ_u.

3 Methodology

As shown in the Figure 2, our method includes a recommendation module and a ranking alignment module. The recommendation module is responsible for selecting the recommended questions, while the ranking alignment module is responsible for enhancing the exploration efficiency.

3.1 Recommendation Module

The recommendation module receives the students' historical records $\mathcal{H}_t^u$ and learning targets $\mathcal{T}_u$ and outputs a recommended question q_t to the students.

Encode questions. We first encode q_i recommended at time i with one-hot encoding based on the question ID and feed the one-hot vector to an embedding layer. Then, we feed the question embeddings into a multi-head attention layer to capture the relationships between questions, following previous works [8,3]:

$$\boldsymbol{e}_i = f_e(\text{one-hot}(q_i)), \tag{2}$$

$$\boldsymbol{r}_i = Attn(f_r(\boldsymbol{e}_i), \boldsymbol{E}, \boldsymbol{E}) \oplus f_r(\boldsymbol{e}_i), \tag{3}$$

where f_e, f_r are linear layers. $e_i \in \mathbb{R}^{d_e}$ denotes the embedding of q_i. $E \in \mathbb{R}^{|\mathcal{Q}| \times d_e}$ denote the embedding matrix of all the questions in the question set $\mathcal{Q}$. $\oplus$ denotes concatenation, $\boldsymbol{r}_i \in \mathbb{R}^{d_r}$ represents the attentive representation of question q_i. $Attn(\cdot, \cdot, \cdot)$ denotes the multi-head attention [15], where key and value are $\boldsymbol{E}$ and query is $f_r(\boldsymbol{e}_i)$.

Encode Historical records. We encode the pair (q_i, y_i) from the students' historical record and input it into a sequence model to obtain a representation of the historical record:

$$\boldsymbol{v}_i = f_v(\boldsymbol{e}_i \oplus y_i), \tag{4}$$

$$\boldsymbol{h}_t = \mathit{SEM}(\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_{t-1}\}), \tag{5}$$

where f_v is a linear layer, $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$ is the representation vector of (q_i, y_i). *SEM(·)* denotes a sequential model like Transformer [15]. $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$ denotes the representation of historical record $\mathcal{H}_t^u$.

Encode learning targets. We use attention representations of the questions r_i included in the learning target to obtain the learning target representation:

$$\boldsymbol{a}_t = f_t(\boldsymbol{T}), \boldsymbol{T} = [\boldsymbol{r}_i | q_i \in \mathcal{T}_u], \tag{6}$$

where $\boldsymbol{T} \in \mathbb{R}^{|\mathcal{T}_u| \times d_r}$ is the target matrix. f_t denotes the learning target encoder, which could implemented by average pooling or multi-head attention [15].

Generate actions. Finally, we extract the most recent question from the historical records as a supplementary item, combine the historical record representation and the learning target representation to generate the state s_t, and feed it to the policy network to generate the action a_t:

$$\boldsymbol{s}_t = f_1(\boldsymbol{r}_{t-1}) + f_2(\boldsymbol{a}_t) + f_3(\boldsymbol{h}_t), \tag{7}$$

$$\boldsymbol{p}_t = \mathit{Softmax}(f_p(s_t)) \tag{8}$$

$$q_t \sim \boldsymbol{p}_t, \tag{9}$$

where f_1, f_2, f_3, f_p are linear layers, $\boldsymbol{r}_{t-1}$ is the attentive representation of q_{t-1}. $\boldsymbol{s}_t \in \mathbb{R}^{d_s}$ is the state we encode. $\boldsymbol{p}_t \in \mathbb{R}^{|\mathcal{Q}|}$ is the probability of recommending the corresponding question. $q_t \sim \boldsymbol{p}_t$ represent sample q_t according to $\boldsymbol{p}_t$.

3.2 Ranking Alignment Module

The Ranking Alignment Module helps the recommender achieve efficient exploration by calculating the differences between students and between recommendations, and aligning their rankings.

Student differences metrics We believe that differences between students are primarily reflected in their learning targets. Therefore, we construct the student differences metric based on the differences in learning targets. Specifically, since a learning target is represented as a set of questions (refer Section 2.1), we measure

the learning target difference for each pair of students using the cardinality of the symmetric difference:

$$\begin{aligned} \mathcal{D}_{uv} &= \mathcal{T}_u \cup \mathcal{T}_v - \mathcal{T}_u \cap \mathcal{T}_v \\ d^t_{uv} &= |\mathcal{D}_{uv}| \end{aligned} \tag{10}$$

where $\mathcal{T}_u$, $\mathcal{T}_v$ denote the learning target of student u, v respectively. $\mathcal{D}_{uv}$ is the symmetric difference of $\mathcal{T}_u$ and $\mathcal{T}_v$. d^t_{uv} denotes the distance of the learning target between u and v.

Recommendation differences metric We calculate the recommendation difference based on the sequence of questions recommended by the recommender. Specifically, we first obtain the representation of the recommendation sequence, and then compute the recommendation difference based on this representation.

To make this metric differentiable, we apply the sequence of recommendation probabilities $\boldsymbol{p}_t$ in Eq. 8 to compute the recommendation difference. We tested two methods for representing the recommendation sequence. The first method adopts a sequential model to encode the sequence of recommendation probabilities for an arbitrary student u:

$$\boldsymbol{b}^s_u = f_s(\{\boldsymbol{p}_1, \boldsymbol{p}_2, ...\}), \tag{11}$$

where $f_s(\cdot)$ is a sequential model. $\boldsymbol{p}_i$ denotes the probabilities of recommendation at step i, which is computed in Eq.(8). $\boldsymbol{b}^s_u \in \mathbb{R}^{d_s}$ denotes the **sequential representation** of the recommendation sequence for student u. We denote the model that encodes the sequence representation using this method as **RAR-S**.

The second method is to directly sum up the recommended probabilities at each step:

$$\boldsymbol{b}^a_u = \sum_{i=1}^{n} \boldsymbol{p}_i, \tag{12}$$

where n is the length of recommendation path, and $\boldsymbol{b}^a_u \in \mathbb{R}^{|\mathcal{Q}|}$ denotes the **addition representation** of the recommendation sequence for student u. We denote the model that encodes the sequence representation using this method as **RAR-A**.

Then, for a pair of students u and v, we feed the representation of recommendation sequence to compute the recommendation difference between two students by:

$$d^p_{uv} = dist(\boldsymbol{b}_u, \boldsymbol{b}_v), \tag{13}$$

where $\boldsymbol{b}_u$ can be $\boldsymbol{b}^s_u$ or $\boldsymbol{b}^a_u$, and $\boldsymbol{b}_u$ and $\boldsymbol{b}_v$ come from the same encoding method. $dist(\cdot)$ denotes the distance between vectors, we implement it with L2 distance. d^p_{uv} denotes the recommendation sequence distance between u and v.

Ranking Alignment In each training iteration, for each student u in the batch, we select m students from the current batch to form a student set $\mathcal{U}_u$. Then, we apply the user differences d^t_{uv} and the recommendation difference d^p_{uv} between student u and $v \in \mathcal{U}_u$ to construct the rank loss:

$$\mathcal{L}_r = \sum_{u \in \mathcal{U}_b} \sum_{v \in \mathcal{U}_u} Clip(\psi d^t_{uv} - d^p_{uv}, 0, \omega), \tag{14}$$

where $Clip(x, a, b)$ is the function that clips the value of x within a and b. ψ and ω are hyperparameter.

3.3 Optimization

We help the recommendation module achieve efficient exploration by adding the rank loss from Eq. 14 to the total loss function. For our recommendation module, it will be optimized by three losses:

The first loss is policy gradient loss [14]:

$$\mathcal{L}_p = -\sum_{t=1}^{T} \hat{r}(s_t, q_t) \log p_t, \tag{15}$$

where p_t is the probability of recommending question q_t, which is obtained from $\boldsymbol{p}_t$ in Eq.(8). $\hat{r}(\boldsymbol{s}_t, q_t)$ denotes the accumulative reward of recommendation at step t, which is computed by $\hat{r}(\boldsymbol{s}_t, q_t) = r(\boldsymbol{s}_t, q_t) + \gamma \hat{r}(\boldsymbol{s}_t, q_t)$, $r(\boldsymbol{s}_t, q_t)$ is the instant reward of recommendation (refer section 2.2).

Then, following [3], we integrate an auxiliary KT module into the historical record encoder and construct a prediction loss:

$$\begin{aligned} \hat{y}_t &= f_k(\boldsymbol{h}_t), \\ \mathcal{L}_k &= -\sum_{t=1}^{T} (y_t \ln \hat{y}_t + (1 - y_t) \ln (1 - \hat{y}_t)), \end{aligned} \tag{16}$$

where f_k is the KT module, which is implemented by a linear function, $\hat{y}_t$ denotes the corresponding predicted correctness of the question q_t.

Finally, with the rank loss in Eq. (14), the final loss of the recommender is:

$$\mathcal{L} = \mathcal{L}_p + \alpha \mathcal{L}_k + \beta \mathcal{L}_r, \tag{17}$$

where α and β are hyperparameters.

4 Experiment

4.1 Simulators and datasets

Training and evaluating a question recommender relies on students' learning effects, which require interaction with students. However, obtaining these effects

Table 1: Comparison of learning effects Δ_u of different question recommenders in five environments when t=10, 30 and 200. Bold indicates the best performance among all methods. Underline indicates the second-best performance.

	KSS		IEKTJU			DKTJU			IEKTA09			DKTA09		
t	10	30	10	30	200	10	30	200	10	30	200	10	30	200
FMLP	0.002	0.002	-0.053	-0.074	-0.087	-0.002	-0.003	-0.003	-0.008	-0.050	-0.103	-0.054	-0.073	-0.074
GPT-3.5	0.022	0.057	-0.109	-0.044	-0.102	-0.003	-0.001	-0.000	-0.065	0.093	0.517	0.007	-0.007	-0.011
DQN	0.062	0.113	0.169	0.293	0.525	0.004	0.010	0.007	0.241	0.424	0.645	0.019	0.016	0.005
SAC	0.031	0.049	-0.020	0.007	0.137	0.002	0.001	-0.001	0.238	0.303	0.517	0.011	0.010	0.020
CSEAL	0.148	0.476	-0.014	0.019	0.256	-0.000	-0.000	0.002	-0.029	-0.070	0.099	-0.038	-0.041	-0.009
SRC	0.110	0.251	0.036	0.048	0.185	0.001	0.001	0.003	0.073	0.132	0.311	-0.001	0.000	0.002
GEHRL	0.149	0.323	0.020	0.058	0.233	-0.000	0.001	0.005	0.030	0.121	0.313	-0.006	0.010	0.004
RAR-S	**0.285**	**0.616**	0.265	**0.505**	**0.716**	**0.019**	**0.019**	**0.019**	**0.516**	**0.576**	0.677	0.058	0.058	0.055
RAR-A	0.232	0.614	**0.283**	0.442	0.702	0.013	0.012	0.016	0.384	0.523	**0.712**	**0.058**	**0.059**	**0.059**

is time-consuming, and low-quality recommendations before model convergence may harm students' learning, raising ethical concerns. Therefore, building on previous works [10,7,3], we evaluate our method in simulated environments.

We divide the simulation environments we use into two groups. The first group is constructed using manually designed rules, which includes the KSS [10] simulator. The second group consists of simulators based on deep knowledge tracing (KT) models. We use two real-world datasets, ASSIST09 [4] [4] and Junyi [5] [2], to train two deep KT models, DKT [13] and IEKT [11], respectively, resulting in four deep KT simulators: DKTA09, IEKTA09, DKTJU, and IEKTJU.

4.2 Baselines

We compare our method with three groups of methods. The first group consists of traditional commercial recommendation methods, including FMLP [16] and GPT-3.5 [1]. The second group consists of classic reinforcement learning algorithms, including DQN [12] and SAC [5]. The third group is recent question recommendation methods, including CSEAL [10], SRC [3], GEHRL [9].

4.3 Experiment Setting

For our proposed method, we set $d_e = 48$, $d_r = 128$, $d_h = 128$, $d_v = 128$, $d_s = 128$. The head of attention is one for all the cases. We select the α, β from $\{0, 0.1, 0.5, 1\}$. We use the Adam [6] to optimize our model. The learning rate is selected from $\{1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}\}$. We use the learning effect in Eq. 1 as the evaluation metric for the performance of different models.

[4] https://sites.google.com/site/assistmentsdata/home/2009-2010-assistment-data

[5] https://www.kaggle.com/datasets/junyiacademy/learning-activity-public-dataset-by-junyi-academy

Table 2: The performance of introducing ranking alignment modules to baselines.

	KSS	DKTA09	IEKTA09	DKTJu	IEKTJu
DQN	0.1128	0.0050	**0.6445**	**0.0070**	0.5248
DQN+RAM	**0.1310**	**0.0351**	0.3736	0.0057	**0.5463**
SAC	0.0485	0.0204	0.5170	-0.0011	0.1370
SAC+RAM	**0.0799**	**0.0691**	**0.7574**	**0.0085**	**0.4217**

4.4 Experiment Result

We measure the performance of different methods after recommending t steps, where $t \in [10, 30, 200]$. Since the maximum steps of recommendation in KSS is 30, we only test the learning effect in $t = 10$ and $t = 30$ for KSS. The results are presented in Table 1. From Table 1, we can observe: (1) RAR-S and RAR-A achieves the best or second-best performance in all the cases. That demonstrates the effectiveness of our method. (2) The second and third groups of baselines, which are based on reinforcement learning, perform better. This indicates that the question recommendation task is more suited to RL-based recommenders, and therefore, our method can benefit most question recommenders.

4.5 Compatibility Study

To demonstrate the compatibility of our rank alignment module with different recommendation models, we integrated the module into DQN and SAC and compared their performance. The results, shown in Table 2, indicate that the rank alignment module improves performance in most cases. For the few instances where performance decreased with DQN, we suspect this is because DQN is unable to effectively learn suitable questions for students from the exploration data.

5 Conclusion

In this paper, we propose a framework called Ranking Alignment Recommendation (RAR), which leverages collaborative ideas to effectively enhance the exploration efficiency of RL-based question recommenders. The RAR framework consists of a recommendation module and a rank alignment module, the latter of which can be integrated into any RL-based recommendation module. Through extensive experiments, we demonstrate the effectiveness and compatibility of our framework and highlight the importance of exploration ability for the performance of RL-based models.

Acknowledgement

This work is partially supported by the Ministry of Science and Technology of China (2023YFF0905400) and the National Natural Science Foundation of China (62307020, U2341229).

References

1. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
2. Chang, H.S., Hsu, H.J., Chen, K.T.: Modeling exercise relationships in e-learning: A unified approach. In: EDM. pp. 532–535 (2015)
3. Chen, X., Shen, J., Xia, W., Jin, J., Song, Y., Zhang, W., Liu, W., Zhu, M., Tang, R., Dong, K., et al.: Set-to-sequence ranking-based concept-aware learning path recommendation. arXiv preprint arXiv:2306.04234 (2023)
4. Feng, M., Heffernan, N., Koedinger, K.: Addressing the assessment challenge with an online system that tutors as it assesses. User modeling and user-adapted interaction **19**, 243–266 (2009)
5. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. pp. 1861–1870. PMLR (2018)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Kubotani, Y., Fukuhara, Y., Morishima, S.: Rltutor: Reinforcement learning based adaptive tutoring system by modeling virtual student with fewer interactions. arXiv preprint arXiv:2108.00268 (2021)
8. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: International conference on machine learning. pp. 3744–3753. PMLR (2019)
9. Li, Q., Xia, W., Yin, L., Shen, J., Rui, R., Zhang, W., Chen, X., Tang, R., Yu, Y.: Graph enhanced hierarchical reinforcement learning for goal-oriented learning path recommendation. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. pp. 1318–1327 (2023)
10. Liu, Q., Tong, S., Liu, C., Zhao, H., Chen, E., Ma, H., Wang, S.: Exploiting cognitive structure for adaptive learning. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 627–635 (2019)
11. Long, T., Liu, Y., Shen, J., Zhang, W., Yu, Y.: Tracing knowledge state with individual cognition and acquisition estimation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 173–182 (2021)
12. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
13. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. Advances in neural information processing systems **28** (2015)
14. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: International conference on machine learning. pp. 387–395. Pmlr (2014)
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
16. Zhou, K., Yu, H., Zhao, W.X., Wen, J.R.: Filter-enhanced mlp is all you need for sequential recommendation. In: Proceedings of the ACM web conference 2022. pp. 2388–2399 (2022)