applied sciences

MDPI

Article

A Multi-Task Knowledge-Tracking Model with a Novel Representative Approach to Problem Difficulty and Student Ability

Wei Zhang, Kaiyuan Qu *, Zhaobin Kang and Sen Hu

Faculty of Artificial Intelligence Education, Central China Normal University, Wuhan 430079, China
* Correspondence: qu2020123592@mails.ccnu.edu.cn

Abstract: Question difficulty and student ability are important factors that affect students' correct answers. Because existing knowledge-tracking models fail to consider these factors, they cannot accurately predict the results of students' answers. In order to explore question difficulty and student ability information more accurately and to improve the accuracy of model prediction, this paper proposes a multi-task knowledge-tracking model (MTLKT) with a novel representative approach to question difficulty and student ability. The model first used the idea of multi-task learning to share underlying information and parameters, to jointly train, and to obtain an information difficulty representation vector consisting of skill difficulty and question difficulty. Then, combined with student learning process, a performance bias function was introduced to improve the attention mechanism and obtain a vector for student current knowledge state and a vector for question-solving performance, thus obtaining a vector for student ability information representation. Finally, the above vectors were concatenated and input into the model as a new representative embedding vector. The experimental results of three real-world data sets showed that our model had great improvement in the evaluation criteria of AUC and ACC and had a better predictive performance than the existing advanced knowledge-tracking models.

Keywords: knowledge tracking; multi-task learning; attention mechanism; education data mining; deep learning

Citation: Zhang, W.; Qu, K.; Kang, Z.; Hu, S. A Multi-Task Knowledge-Tracking Model with a Novel Representative Approach to Problem Difficulty and Student Ability. *Appl. Sci.* **2023**, *13*, 4226. https://doi.org/10.3390/app13074226

Academic Editor: Yu-Dong Zhang

Received: 3 March 2023
Revised: 20 March 2023
Accepted: 22 March 2023
Published: 27 March 2023

1. Introduction

Knowledge tracking is a technology that models students according to their answers in the past to obtain their current knowledge mastery. Its goal is to retrieve potential learning rules from students' learning trajectories by simulating student modeling. In recent years, knowledge-tracking technology has been widely used in online education systems to track the changes in students' knowledge mastery, learning behavior characteristics, and individual differences in ability levels to provide personalized learning guidance for different students and improve learning efficiency.

The key to knowledge-tracking technology is to accurately grasp the situation of students in the process of answering questions, and the difficulty of questions and student ability are two important factors that affect whether students answer questions correctly. On the one hand, different skills have different difficulties, and different questions containing the same skill also have different difficulties. The difficult information contained in questions greatly influences whether students can give correct answers [1,2]. On the other hand, with the dynamic change in students' answering processes, the specific performance of a student in answering is an important reflection of their ability. It is difficult to accurately evaluate students' ability levels without considering the impact of students' specific performances on their current ability.

Existing knowledge-tracking methods often ignore the above factors or do not consider them in a comprehensive enough manner in the modeling process. Considering any of the

above influencing factors in isolation or insufficiently makes it difficult to model students accurately. For example, the pre-trained PEBG model [3] extracts high-level information, such as the relationship between questions and skills, the relationship between skills and skills, and the difficulty of questions, but does not consider the change in student ability. The CKT [4] model based on convolutional neural networks takes into account the differences in students' a priori knowledge, as well as learning ability, and takes a student's answer performance over a period of time as his knowledge mastery level. However, the model ignores the differences caused by the different difficulties of different questions. The context-aware AKT knowledge-tracking model [5] uses a monotonic attention mechanism to calculate the attentional weights between questions and obtains a student's knowledge state through weighted aggregation, but the modeling process does not take into account the impact of answer performance on student ability and is not sufficiently considered. Although these models perform well in prediction, they ignore certain factors and have great potential for improvement in prediction accuracy.

It is a major challenge for current research to consider the above factors in the modeling process, to dig deeper into the difficulty information of questions, and to extract information about student ability accurately. Therefore, this paper proposes a multi-task knowledge-tracking model (MTLKT) based on a novel question difficulty and student ability representation method. We specifically build a question information difficulty extraction module, which uses the idea of multi-task learning to share the underlying information and parameters, to jointly train, and to obtain an information difficulty representation vector composed of skill difficulty and question difficulty. We build a student ability extraction module. With the introduction of a performance bias function to enhance the weight of questions with good performance and weaken the weight of questions with poor performance, our study reflects the influence of students' specific response performances on student current state and ability to improve the attention mechanism and obtain a student current knowledge state vector and a question-solving performance vector. Once summed, we can obtain a vector representing student ability information.

In summary, the contributions of this paper are as follows:

- A new information difficulty representation method is proposed, which includes both question difficulty information and skill difficulty information, uses multi-task learning to fully exploit the potential correlation information between the two, and provides a more comprehensive representation of information difficulty.
- A new representation of student ability is proposed, which includes information on student knowledge state, as well as information on student performance in solving questions. The attention mechanism is improved by combining the student learning process, and student response performance is considered when calculating the attention weights to model student ability more accurately.
- A new embedding vector representation method is designed, which splices the question difficulty information vector and the student ability information vector to form a new embedding vector that incorporates more abundant feature information and alleviates the underfitting problem of the model due to the sparse interaction records.
- Experiments on several publicly available data sets show that our model outperforms the comparison model. The ablation experiment proves the effectiveness of the model components.

2. Related Works

At present, knowledge-tracking models can be divided into two types: knowledge-tracking models based on probability graphs and knowledge-tracking models based on neural networks. These are respectively introduced in the following.

Regarding knowledge-tracking models based on probability graphs, ZA Pardos proposed Bayesian knowledge-tracking using a machine-learning algorithm [6], which had good interpretability but required manual skill annotation by human experts and was costly.

With regard to knowledge-tracking models based on neural networks, in recent years with the development of deep neural network technology, C Piech input students' interactive records into LSTM [7] or RNN models and used hidden states of neural networks to represent students' knowledge states, proposing deep knowledge tracking (DKT) [8]. Compared with the previous Bayesian knowledge-tracking model, the improvement in AUC evaluation metrics was close to 20%, which also has allowed a wide range of scholars to see the huge development potential of deep knowledge tracking. Yeung et al. [9] improved DKT by adding regularization terms into the loss function, which alleviated the problem of large fluctuation in student state prediction in DKT. Zhang proposed the DKVMN (dynamic key-value memory network) [10] model in 2016, which used a static matrix to store skills and a dynamic matrix to update students' mastery of the skills after answering each question. The DKVMN model solved the drawback of the DKT model, which could not extract students' mastery of each skill well. Liu S in [11] used a hierarchical memory network to simulate long-term and short-term memory and restore the human memory mechanism in a knowledge tracking task. Pandey S in [12] proposed a relationship-aware self-attentive mechanism by which the forgetting behavior of students was simulated. Nakagawa and several other scholars in [13–15] applied graph neural networks to the knowledge tracking domain to enhance the prediction by using the powerful expressive power of graph neural networks. In general, deep-learning-based knowledge tracing and its improved models show significant performance improvements compared to the Bayesian knowledge-tracking model but suffer from the problem of not being able to fully learn effective features when interaction records are too sparse due to limitations of their structure.

To address the shortcomings of deep-knowledge-tracking models, many scholars have made improvements:

Xia Sun considered the characteristics of students' behavior and learning ability in DKVMN-LA [16], dynamically simulated student ability, and improved the accuracy of prediction, but failed to conduct an in-depth exploration of question information. The EKT [17] model Liu Qi proposed was based on the DKT model and used the text information of the question to enhance the representation of the question, alleviating the model underfitting problem caused by sparse interactive records, but the model lacked research on student ability. In addition, there have also been attempts to incorporate an attention mechanism [18] into knowledge-tracking models to improve the performances of the models, such as SAKT and [19] SAINT [20] using a Transformer [21] model based on a self-attention mechanism to assign weights to previous responses to extract key information, as well as Shi et al. in [22], who improved the attention calculation of a Transformer model by considering the time interval between answers, thus simulating the forgetting behavior of students and optimizing student modeling.

Overall, all of the above models have made good progress in the area of knowledge tracking, but there is much room for improvement in prediction performance due to a lack of adequate consideration of question difficulty, as well as student ability.

3. The Proposed Method

In this section, we describe the MTLKT model in detail, which consisted of three modules: an information difficulty extraction module, a student ability extraction module, and a prediction module. In the information difficulty extraction module, in order to have a more comprehensive and abundant embedding representation of information difficulty, more fine-grained question information, such as response time, number of hints, etc., was input into the module, and then an information difficulty representation vector consisting of skill difficulty and question difficulty was obtained using the idea of multi-task learning to share the underlying information and parameters for joint training.

In the student ability extraction module, the attention mechanism was improved by introducing a performance bias function to increase the weight of questions with good performance and decrease the weight of questions with poor performance to reflect the

influence of a student's specific performance on student ability. We also used the Phi coefficient, which measures the relationship between two variables, as the correlation coefficient between skills and used the obtained correlation coefficients and attention weights to weight the aggregated skill interaction records and question interaction records, respectively, to obtain a vector for student current knowledge state and a vector for question-solving ability. We then added the two vectors to obtain a vector for student ability.

The input of the prediction module was the output vector of the above two modules and the predicted question identification, and the output was the predicted response result, which was divided into 1 and 0, representing correct and incorrect responses, respectively.

The model structure is shown in Figure 1.

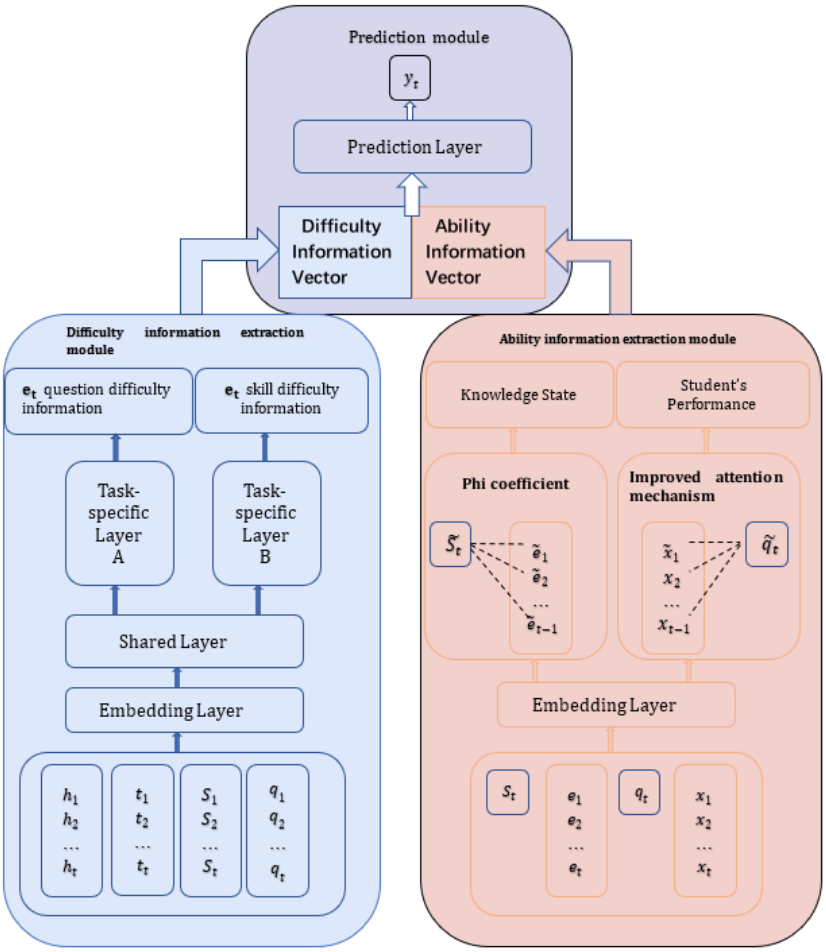

Figure 1. Model structure diagram.

3.1. Information Difficulty Extraction Module

A multi-task learning approach was used in this module to obtain an information difficulty representation vector. Multi-task learning is an inductive migration method that leverages domain-specific information implicit in the training signals of multiple related tasks to improve the predictive performance and generalization of models [23]. It is an effective method for improving the performance of natural language-processing tasks and has been widely used in several natural language-processing tasks [24,25]. Therefore, we used this method to enrich the representation of the information difficulty vector.

The model was trained in two stages: first on a shared, multi-task coding layer and then on supervised, multi-task learning training using the labeled data of each task and the corresponding loss function.

3.1.1. Task Selection

The key to improving the effectiveness of multi-task learning is to find suitable multiple tasks for co-training. The stronger the correlation between various tasks, the more influential the co-training.

In this section, multiple attribute features of the question were predicted as multiple tasks, and the difficulty of the question was predicted as the main task. We needed to find the secondary task that had the strongest correlation with the main task first. The correlation between two tasks in multi-task learning could be expressed by the Pearson coefficient between the labels of the two tasks, and the larger the correlation coefficient, the stronger the correlation between the tasks.

The Pearson coefficient is a statistic used to measure the strength of a correlation between two variables, and the Pearson coefficient is calculated as follows:

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E\left(X^2\right) - E^2(X)}\sqrt{E(Y^2) - E^2\left(Y^2\right)}} \quad (1)$$

where X represents the question difficulty label, and Y represents other labels, including skill difficulty, number of hints, and number of attempts. In Table 1, we also show the meanings of these different features. Since the original data set did not contain the two features of question difficulty and skill difficulty, we obtained these two features by analyzing and processing the data set, which were calculated as follows:

$$question_{dif_i} = \frac{question\ i = 1}{N_i} \quad (2)$$

$$skill_{dif_j} = \frac{Skill\ j = 1}{N_j} \quad (3)$$

Table 1. Feature Introduction.

Name	Description
question_dif	Difficulty of a question, defined as the percentage of questions answered correctly out of the total number of responses to a question
skill_dif	Difficulty of a skill, defined as the percentage of questions containing a skill that are answered correctly
attempt_count	Number of student attempts on a specific question
hint_total	Total number of hints contained in a question
overlap_time	Amount of time a student spends on a question

The difficulty of question i is calculated as the number of correct answers to question i divided by the total number of answers to question i. The difficulty of skill j is calculated as the number of correct answers to questions containing skill j divided by the total number of answers to questions containing skill j.

In Figure 2, we show the magnitude of the Pearson coefficients between these features and the question difficulty features (0.8–1 is a very strong correlation, 0.6–0.8 is a strong correlation, and 0.2–0.4 is a weak correlation). Finally, we selected the skill difficulty feature that had the strongest correlation with the question difficulty feature and used predicting the skill difficulty contained in a question as an auxiliary task.

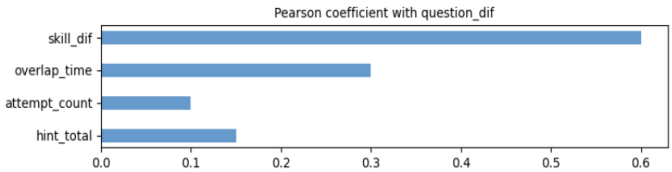

Figure 2. Correlations between different characteristics.

3.1.2. Module Inputs

We took the question identification q_i and the skill identification s_i contained in the question, as well as the time sequence of answering the questions t_i and the number of hints h_i, as the input of this module. The above elements then passed through the embedding layer and were mapped to a vector in the potential space, yielding the question embedding vector $\widetilde{\mathrm{Q}} \in R^d$, the skill embedding vector $\widetilde{\mathrm{S}} \in R^d$, the time vector for answering questions $\widetilde{\mathrm{T}} \in R^d$, and the embedding vector for the number of hints $\widetilde{\mathrm{H}} \in R^d$, where d is the dimensionality of the mapping.

3.1.3. Position Encoding

Sequence processing in the attention mechanism relied on location encoding and the location information of the contained sequence elements. The position encoding of the model in this paper used an absolute position-encoding method, that is, the position encoding of the sine and cosine functions [21]. For a length d at a position pos in the sequence, the value of dimension i in the position-encoding vector is as follows:

$$\mathrm{PE}_{\mathrm{pos,2i}} = \sin\left(\frac{\mathrm{pos}}{10,000^{\frac{2\mathrm{i}}{\mathrm{d}}}}\right) \tag{4}$$

$$\mathrm{PE}_{\mathrm{pos,2i+1}} = \cos\left(\frac{\mathrm{pos}}{10,000^{\frac{2\mathrm{i}}{\mathrm{d}}}}\right) \tag{5}$$

where $\mathrm{i} \in (0, 1, 2 \ldots \mathrm{d}/2)$. After generating the location code, it was combined with the embedded representation vector in an additive manner.

3.1.4. Shared Layer

The shared layer consisted of a multi-head attention layer, a normalization layer, and a forward feedback layer. The input was obtained by summing each embedding vector with the position encoding $\mathrm{P_i}$:

$$\mathrm{O} = \widetilde{\mathrm{Q}} + \widetilde{\mathrm{S}} + \widetilde{\mathrm{T}} + \widetilde{\mathrm{H}} + \mathrm{P_i} \tag{6}$$

where $\mathrm{O} \in R^d$.

Multi-head self-attentive layer. In the self-attentive mechanism, Q is usually used to denote the query, K denotes the key, and V denotes the value. The attention weights were calculated by first dotting Q and K. After that, to prevent the result from being too large to affect the subsequent gradient propagation, the dotted product result was divided by $\sqrt{d}$ and, finally, input to the SoftMax function for normalization. The attention weights are calculated as follows:

$$\mathrm{Attention}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \mathrm{softmax}\left(\frac{\mathrm{QK^T}}{\sqrt{\mathrm{d}}}\right)\mathrm{V} \tag{7}$$

When we predicted the information related to question t, we should only consider the interaction information of the first $\mathrm{t}-1$ questions, so we used the upper triangle mask to mask the information of the subsequent positions.

$$\mathrm{head_i} = \mathrm{Softmax}\left(\mathrm{Mask}\left(\frac{\mathrm{Q_iK_i^T}}{\sqrt{\mathrm{d}}}\right)\right)\mathrm{V_i} \tag{8}$$

$$\begin{array}{rcl} Q_i & = & \left[q_1^i, \cdots, q_t^i\right] = \mathbf{O}W_i^Q \\ K_i & = & \left[k_1^i, \cdots, k_t^i\right] = \mathbf{O}W_i^K \\ V_i & = & \left[v_1^i, \cdots, v_t^i\right] = \mathbf{O}W_i^V \end{array} \tag{9}$$

The multi-head attention layer projected Q, K, and V into different potential spaces through different matrices W^Q, W^K, and W^V to capture rich feature information to form different heads and stitched these information heads together.

$$\text{Multihead(Q, K, V)} = \text{Concat}(\text{head}_1, \text{head}_2 \ldots \text{head}_i W^O \Big) \tag{10}$$

Feed forward layer. Since the output of the input vector after the multi-headed self-attentive layer was still a linear combination, we used a feed forward layer to convert the linear combination to a nonlinear combination. The feed forward layer consisted of two linear transformations activated by a ReLU function to add nonlinearity to the model.

$$\text{FFN(x)} = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{11}$$

where x = Multihead(Q, K, V), and W_1, W_2, b_1, b_2 are the weight matrices and bias vectors, which are continuously updated in training.

Residual connect. He introduced the concept of residual connect in [26], where the primary role of residual linking was to propagate low-level features to higher levels, which made it easier for the model to utilize low-level information. Residual connection is a common deep-learning technique that builds deeper networks in a more stable manner.

Normalization layer. The input information was normalized after passing through the multi-head attentive layer and the feed forward layer. The research of Ba J L in [27] showed that a neural network could be more stable and fast by normalizing the input, and gradient disappearance and explosion could be prevented simultaneously.

3.1.5. Task-Specific Layer

The information was input to different task-specific layers after sharing the underlying parameters through the shared layer, which consisted of a feedforward layer and a linear layer. In this paper, we predicted the question difficulty and the skill difficulty to obtain the information difficulty representation vector of the prediction question. First, we preprocessed the original data and eliminated the questions and skills whose interactive records were less than 10 times. Then, we took the difficulty of the question d_q and difficulty of the skill d_s as the label of a task. The error L_{dq} between $\widetilde{d_q}$ and d_q and the error L_{ds} between $\widetilde{d_s}$ and d_s were calculated by the mean square loss function MSEloss, and the parameters of the model were continuously optimized by backpropagation.

$$L_{dq} = {}^1/_t \sum_{i=1}^{t} \left(d_q - \widetilde{d_q} \right)^2 \tag{12}$$

$$L_{dq} = {}^1/_t \sum_{i=1}^{t} \left(d_s - \widetilde{d_s} \right)^2 \tag{13}$$

Usually, the easiest way to calculate the total loss of multi-task learning is to add up the losses of multiple tasks or manually set the weight parameter w_i:

$$\text{loss}(t) = \sum_i w_i L_i(t) \tag{14}$$

However, this method is not only time-consuming but also cannot solve the problem effectively. For example, if the gradient of task A is large with a small weight w_A, w_A continues limiting the model from better learning task A until the end of the training, so the weight w_i needs to be continuously changed during the training process:

$$\text{loss}(t) = \textstyle\sum_i w_i(t) L_i(t) \tag{15}$$

During the training process, the different convergence rates of different tasks lead to an inability to achieve the best results when training together at the same time, and this imbal-

ance caused by different convergence rates is mainly manifested as the gradient imbalance during backpropagation. Therefore, we used the optimization method of GradNorm [28] to improve the above problem. GradNorm enables different tasks to be trained at similar rates by dynamically adjusting the magnitude of the backpropagation gradients of different tasks. Table 2 illustrates some of the symbolic concepts in the GradNorm method.

Table 2. Notations.

Notations	Description
W	Neural network parameters of the shared layer.
$G_w^{(i)}(t)$	L2 norm of the gradient of the band weight loss $w_i(t)L_i(t)$ for task i on W.
$\overline{G}_w(t)$	Average gradient norm across all tasks at training time t.
$\widetilde{L}_i(t)$	Loss ratio for task i at time t. $\widetilde{L}_i(t)$ is a measure of the inverse training rate of task i.
$r_i(t)$	Relative inverse training rate of task i.

To dynamically adjust the weight $w_i(t)$, we needed to put the gradient norms of different tasks under the same metric to obtain their relative gradient size. $\overline{G}_W(t)$ was used to measure the norm of the average gradient at time t and, thus, determine the relative gradient, and $r_i(t)$ was used to balance the gradient. If $r_i(t)$ was larger, the gradient of task i was larger to speed up task training. Therefore, the gradient norm objective of task i was $\overline{G}_W(t) \times r_i(t)$, and the loss function ***Grad Loss*** of the weight function $w_i(t)$ was further obtained:

$$\text{Grad Loss} = \sum_i \left| G_W^i(t) - \overline{G}_W(t) \times r_i(t) \right| \tag{16}$$

$$\widetilde{L}_i(t) = \frac{\widetilde{L}_i(t)}{\text{AVG}\left[\sum \widetilde{L}_i(t)\right]} \tag{17}$$

$$\text{Grad Loss} = \sum_i \left| G_W^i(t) - \overline{G}_W(t) \times r_i(t) \right| \tag{18}$$

Therefore, there were two kinds of losses in the multi-task learning module: 1. The loss of tags was used to update the predicted results. 2. The loss of weight gradient was used to update the weight of each task.

3.2. Student Ability Extraction Module

In this paper, the abstract concept of student ability was embodied as their knowledge state and their ability to solve questions.

Student knowledge state and question-solving ability change dynamically with student problem-solving process. Therefore, we obtained the weight of the correlation between the current question and the historical questions by exploring the correlation between the current question and the historical questions, as well as the correlation between the skill and the historical skills. Finally, the weight was used to aggregate the embedded vector of the question and the embedded vector of the skill, and the two vectors were added to obtain the ability vector for students.

3.2.1. Module Input

In this paper, we fused the skill identification s_i and the question identification q_i with the answer y_i, and then we could obtain the e_i and x_i. Through the embedding layer, we could obtain a skill interaction embedding vector $\widetilde{e} \in R^{2d}$ and a question interaction embedding vector $\widetilde{x} \in R^{2d}$.

$$e_i = s_i + y_i * E_s \tag{19}$$

$$X_i = q_i + y_i * E_q \tag{20}$$

E_s and E_q respectively represent the total numbers of skills and questions.

3.2.2. Knowledge State Vector Acquisition

Because different students have different understandings and applications of skills, the relationship between skills in different data sets was not fixed, so it was more practical to calculate the correlation between them through the interaction of skills in specific data sets. Specifically, we used the Phi coefficient ϕ to calculate the correlation between two skills. First, we built the following contingency Table 3:

Table 3. A contingency table for two skills: i and j.

		Skill j Incorrect	Correct	**Total**
Skill i	Incorrect	n_{00}	n_{01}	n_{0*}
	Correct	n_{10}	n_{11}	n_{1*}
	Total	n_{*0}	n_{*1}	n

Where Skill i appears before Skill j and only considers the latest right and wrong situation, $n_{00}, n_{01}, n_{10}, n_{11}$ represent, respectively, that i and j are both wrong, i is wrong and j is right, i is right and j is wrong, and i and j are both right. The Φ coefficient is calculated as follows:

$$\phi_{i,j} = \frac{n_{11}n_{00} - n_{01}n_{10}}{\sqrt{n_{1*}n_{0*}n_{*1}n_{*0}}} \tag{21}$$

The value of the Φ coefficient was between −1 and 1, with higher values representing a more significant influence of exercise i on exercise j. After obtaining the skill relevance coefficient, we multiplied it with the skill embedding vector $\widetilde{e_i}$ and accumulated it to obtain the student knowledge state vector $K \in R^{2d}$:

$$K = \sum_{i=1}^{t-1} \phi_{i,t}\widetilde{e_i} \tag{22}$$

3.2.3. Question-Solving Ability Vector Acquisition

In this paper, we calculated the relevance weights of the current question and historical questions through an attention mechanism and mapped the question embedding $\boldsymbol{q}_t$ to be predicted as a query vector and the historical interaction records as a key-and-value vector, and we made a dot product calculation. Considering that students' behavioral characteristics (e.g., time spent on the problem or number of attempts) reflect their ability to a certain extent, the shorter the time spent on a problem (or the fewer attempts), the better the students' mastery of the question, and the longer the time spent on a problem (or the more attempts), the less the students' mastery of a question. Therefore, we improved the formula for calculating attention weights by multiplying the performance bias function f (x) with the original formula: the attention weights were calculated by strengthening the weights of questions that were well-mastered and decreasing the weights of questions that were poorly mastered. The specific calculation method is as follows:

$$att_{i,t} = \text{softmax}(\boldsymbol{\alpha}_{i,t}) = \frac{\exp(\boldsymbol{\alpha}_{i,t})}{\sum_{i=1}^{t-1}\exp(\boldsymbol{\alpha}_{i,t})} \tag{23}$$

$$\boldsymbol{\alpha}_{i,t} = \frac{\left(\boldsymbol{W}^{Q}\boldsymbol{e}_t\right)^{\mathrm{T}} \cdot \boldsymbol{W}^{K}x_i}{\sqrt{d}} f(b-\theta) \tag{24}$$

$$f(b-\theta) = \frac{1}{1+e^{(b-\theta)}} \tag{25}$$

where $\boldsymbol{W}^{Q}$ and $\boldsymbol{W}^{K}$ are the mapping matrices of the query and key, respectively. $f(b-\theta)$ denotes the performance bias function, in which θ denotes the average response time for a question and b denotes the actual response time of a student on a question. The

function took the value of 1/2 when a student's actual performance equaled the expected performance. In contrast, the value of the function tended toward 1 (0) when a student's performance was much higher (lower) than the expected performance. After obtaining the weight att, we multiplied and aggregated the weight with the historical question interaction embedded vector $\widetilde{x}_i$ to obtain the performance vector $\mathrm{L} \in R^{2d}$ of the student in the current time step:

$$\mathrm{L} = \sum_{i=1}^{t-1} att_{i,t} \widetilde{x}_i \tag{26}$$

3.2.4. Prediction Module

After the information difficulty extraction module, we obtained the skill difficulty representation vector $d_s \in R^d$ and the question difficulty representation vector $d_q \in R^d$. After the student ability extraction module, we obtained the knowledge state vector $K \in R^{2d}$ and the question-solving ability vector $L \in R^{2d}$. Finally, by concatenating the above vectors with q_t (current predicted question embedding vector), we obtained the embedding vector $\mathrm{Z} \in R^{4d}$:

$$\mathrm{Z} = \left(d_s + d_q\right) \oplus (K + L) \oplus q_t \tag{27}$$

Then, the vector Z was input into the multi-layer perceptron (MLP) to predict the questions to be answered by the current student.

$$layer_t^l = \mathrm{ReLU}\left(\boldsymbol{W}^l layer_t^{l-1} + \boldsymbol{b}^l\right) \tag{28}$$

$$y_t = \mathrm{Sigmoid}\left(\boldsymbol{W}^l layer_t^{L-1} + \boldsymbol{b}^l\right) \tag{29}$$

where l denotes the layer of the multilayer perceptron, $layer_t^0 = \mathrm{Z}$, and $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ denote the weight matrix and bias vector of the lth layer, respectively. The error between the predicted value y_t and the true value $\widetilde{y}_t$ was calculated using a binary cross-entropy loss function, and the model parameters were optimized with the Adam optimizer. The loss values are calculated as follows:

$$L = -\sum_{i=1}^{t} (\widetilde{y}_i \log y_i + (1 - \widetilde{y}_i)) \log(1 - y_i)) \tag{30}$$

4. Experiment and Analysis

In this section, we conduct sufficient experiments to verify the following three questions:

(RQ1) How does the MTLKT model perform compared to current, mainstream knowledge-tracking models?

(RQ2) How do the various modules in the MTLKT model affect the model prediction results?

These questions are answered in the following section after some basic experimental setups are described.

4.1. Data Sets

This paper chose three data sets commonly used in the field of knowledge tracking: Assistments09, Assistments12, and Assistments17. These three data sets are collected from the real data of the ASSISTments online learning platform, The complete information is shown in the Table 4, and there are some examples of datasets are shown in Table 5. In order to obtain higher quality and clean data, we first cleaned the data set: we deleted items containing null values and duplicates, we deleted skill identifications and question identifications with less than five interactive records (to ensure adequate records to calculate difficulty coefficients), we deleted items that took too long to answer, and we randomly

selected 5000 students from the Assistment12 data set with too many students. The processed data set information is given in the following.

Table 4. Data set statistics.

Data Set	Assistments09	Assistments12	Assistments17
Number of students	4151	5000	1709
Number of skills	123	260	102
Number of questions	15,680	38,052	3162
Number of records	325,600	1,000,964	942,816
Number of records/Number of skills	2647	3850	9067
Number of records/Number of questions	21	26	298

Table 5. Some of the questions in the data set.

Question	Question Text	Skill
e_1	Solve x:$2\|x+6\|+7=4x+6+4$	Linear Equations
e_2	Solve x:$x^2+3x-28=0$	Quadratic Equations
e_3	Evaluate $\sqrt{28}$	Square roots
e_4	What are the factors of 5	Division

4.2. Evaluation Metrics

The evaluation indices used in this experiment were Area Under the Curve (AUC) and Accuracy (ACC). AUC is the area enclosed by coordinates and axes under the ROC curve. Receiver operating characteristic curves are in the vertical and horizontal coordinates of the true positive rate (True Positive Rate) and the false positive rate (False Positive Rate), respectively. The area under the curve usually ranges from 0.5 to 1, and the higher the value range, the better the performance of the model.

4.3. Compared Models

DKT-Q [4]. The most classical model of deep knowledge tracking uses a long short-term memory network (LSTM) or a recurrent neural network (RNN) to predict. Unlike the original DKT model with knowledge skill identification as the input, this paper chose to use the question identification as the input for DKT.

DKVMN-Q [16]. A dynamic key-value memory network is used to update student knowledge status at each time step.

CKT [4]. Student prior knowledge and learning rate are considered in modeling the student state, which enriches the student characteristics.

SAKT [8] is a knowledge-tracking model based on an attention mechanism.

SAINT [10] is a knowledge-tracking model based on a Transformer, but the question identifications, as well as the responses, are separated and entered into an encoder and decoder in the Transformer, respectively.

PEBG [3] is a pre-trained model for mining the relationship between questions and skills. In this paper, we also used a multilayer perceptron as the prediction layer of PEBG.

4.4. Experimental Environment and Parameter Setting

The experiment in this paper was conducted in the following environment: a Windows 10 operating system and an Intel (R) Core (TM) i5-9300H CPU. The model in this paper was implemented using Pytorch framework, the maximum sequence length was set to 100 in the Assistments data set, and if there were fewer questions than the maximum, the padding token was 0. The Adam [25] optimizer was used, the learning rate was set to 1×10^{-3}, the embedding of the matrix representation vector dimension was set to 128, the feed-forward layer dimension was 200, the number of multi-headed attention layer heads was 8, the number of layers of the multilayer perceptron was 2, the intermediate layer dimension was

128 dimensions, the number of samples for each batch of training was 128, and the training was iterated 50 times.

The compared models used the publicly available codes in their respective papers and the hyperparameter settings.

4.5. Comparsion and Analysis of Experimental Results

4.5.1. Display of Experimental Results (RQ1)

The experimental results are shown in Table 6.

Table 6. Performance of each model on different data sets.

Data Sets	Assistments09		Assistments12		Assistments17	
	AUC	ACC	AUC	ACC	AUC	ACC
DKT-Q	0.7141	0.6985	0.7012	0.6834	0.7314	0.7210
DKVMN-Q	0.6961	0.6543	0.7000	0.6710	0.7211	0.7059
CKT	0.7340	0.7155	0.7180	0.7150	0.7403	0.7004
SAKT-Q	0.7322	0.6953	0.7472	0.7083	0.7507	0.7322
SAINT-Q	0.7432	0.7231	0.7501	0.7138	0.7533	0.7026
PEBG	0.7649	0.7360	0.7710	0.7423	0.7789	0.7521
MTLKT	0.7703	0.7659	0.7780	0.7542	0.7860	0.7653

From the table, we concluded that:

(1) Compared with the six comparison models, the experimental results of the model in this paper on the three data sets were optimal, and the AUC values of the 09, 12, and 17 data sets were 0.7703, 0.7780, and 0.7860, respectively. In the experiment, the performance of the model on the 17 data set was better than other data sets because the interaction records of the Assistment17 data set were relatively dense, which was more conducive to model learning and training.

(2) The model in this paper improved 7.68% in AUC and 7.08% in ACC (in the Assistments12 data set) compared to the classical DKT model. It was also observed in Figure 3 that the attention-based knowledge-tracking models SAKT and SAINT outperformed the recurrent-neural-network-based DKT model in terms of results because the recurrent neural network suffered from information loss when the interaction sequence was too long, which affected the prediction results. The SAINT model outperformed SAKT because it used a deeper level of attention network, which was better for the model to capture the complex relationship between questions and responses. The model outperformed the SAINT model on all three data sets because we considered both question difficulty and student ability and designed an embedding vector containing both factors into the prediction layer so that the model could learn more effective features for prediction, thus alleviating the problem of model underfitting due to sparse data sets and, ultimately, improving the accuracy of prediction. We also improved the calculation of attention weights based on student answer performance, which made the calculation of attention weights more reasonable.

(3) As shown in Figure 4, the results of this model also outperformed the CKT model based on convolutional neural networks, with increases in the AUC values of 3.63%, 6%, and 4.57% for the three data sets, as well as of 5.04%, 3.92%, and 6.49% for ACC, respectively. The CKT attempts to consider students' prior knowledge and the effect of learning rate on students' answers in the model. However, in considering these factors, only the answer result is used as an evaluation indicator, i.e., the author believed that the student could fully master the questions answered correctly. This approach, which considers only the effect of answer results on student ability, is not accurate and objective because it ignores the difficulty levels of different questions. Therefore, the model in this paper, which fully considered the information on the difficulty of the questions, worked better.

(4) As shown in Figure 5, Compared with the pre-trained PEBG model, MTLKT performed better when using the same prediction layer, increasing by 0.54%, 0.7%, and 0.71%, respectively, for AUC and 2.99%, 1.19%, and 1.32%, respectively, for ACC. A PEBG is dedicated to discovering advanced information between questions and skills, and it considers the cross-relationships between questions and skills, where a question contains multiple skills and a skill is related to multiple questions. A PEBG solves the sparsity problem by constructing a question–skill bipartite graph to extract the explicit and implicit relationships between topics and skills. However, in this paper, we only considered single-skill questions, which means that a question contained only one skill. Therefore, the method of constructing a dipartite graph in a PEBG does not capture enough information. In addition, although a PEBG also considers information difficulty, it only considers the difficulty of the questions themselves, ignoring the difficulty of the skills included in the questions, and thus, it does not provide a comprehensive representation of information difficulty. Finally, a PEBG extracts information only between questions and skills and ignores information about student ability, whereas question difficulty and student ability are the two most important factors that must be considered to predict whether students can answer correctly. By combining the above information, it can be concluded that the proposed embedding vector incorporating information difficulty and student ability information was more suitable for the knowledge-tracking task.

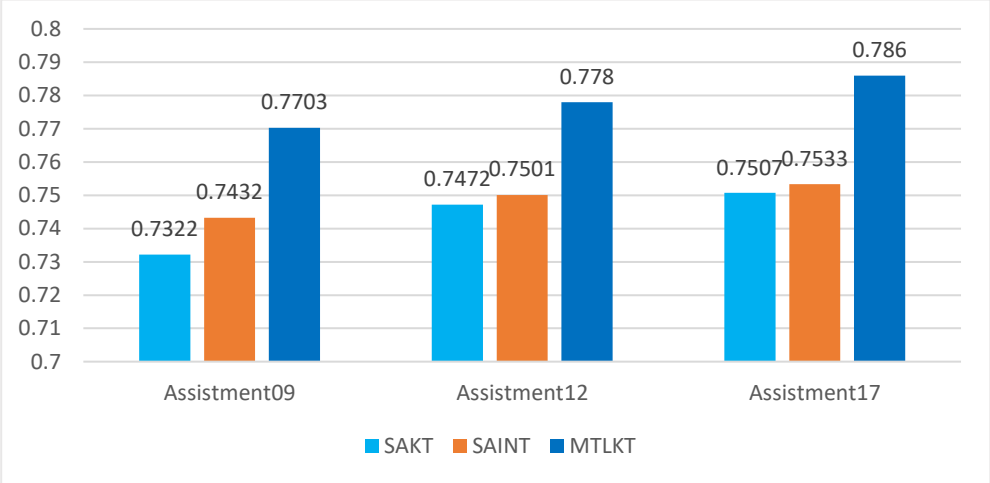

Figure 3. Comparison with attention mechanism model.

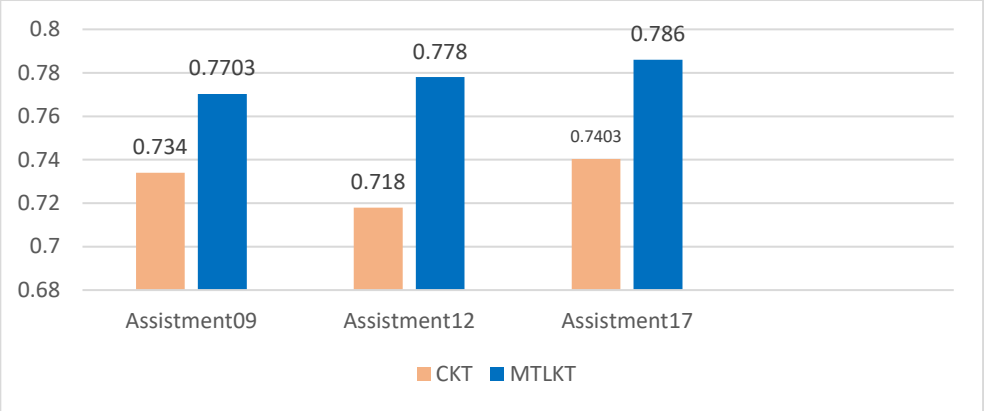

Figure 4. Comparison with convolutional-based model.

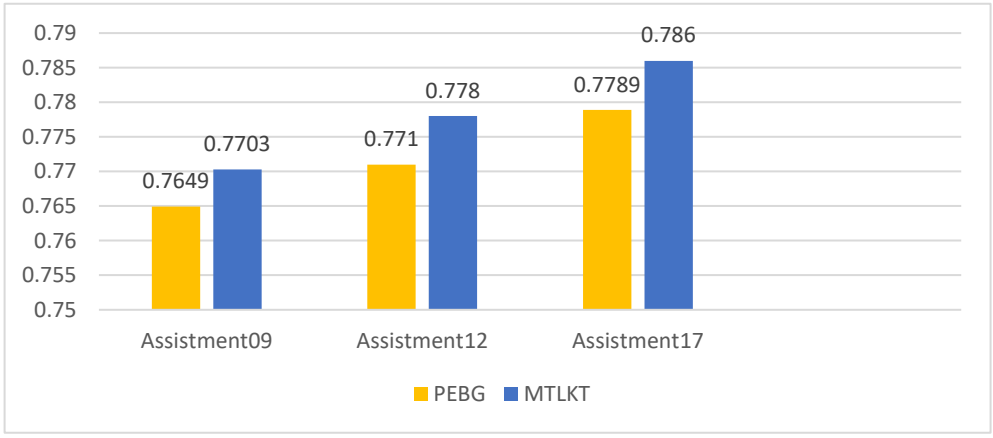

Figure 5. Comparison with pre-trained model.

4.5.2. Ablation Experiment and Visualization

In order to verify and analyze the effectiveness of the components in the model of this paper, as well as the optimal parameter settings, ablation experiments were conducted in this paper.

Information Difficulty Extraction Module Ablation

In this section, we replace the multi-task learning part of the information difficulty extraction module in this model using two parallel Transformers (without sharing parameters and inputs) to obtain the difficulty embedding vector of the predicted questions and the difficulty embedding vector of the skills, respectively, while the rest of the model remains the same as the original model. The results of the ablation experiments are shown in Table 7.

Table 7. Information difficulty extraction module ablation results.

Data Sets	Assistments09		Assistments12		Assistments17	
	AUC	ACC	AUC	ACC	AUC	ACC
Compared Model	0.7550	0.7324	0.7669	0.7438	0.7800	0.7575
MTLKT	0.7703	0.7659	0.7780	0.7542	0.7860	0.7653
Elevation	1.53%	3.35%	1.11%	1.04%	0.6%	0.78%

From the results in the table, it can be concluded that the model with the information difficulty extraction module had a better performance, and the improvement was especially obvious for the Assistments09 data set. This may be due to the fact that the question-based interaction records in the Assistments09 data set were more sparse (refer to Table 1), and the multi-task learning allowed the model to learn more and richer features by sharing inputs and underlying parameters, thus improving the prediction performance of the model.

Student Ability Extraction Module Ablation

In this section, we show how the improved attention mechanism in the student competence module affected the performance of the model and how different behavioral characteristics in the performance bias function affected the status of students.

MTLKT-NA: Represents calculating weights between vectors without using the attention mechanism in the model, instead simply adding up the historical interaction embedded vectors to obtain the current knowledge state of the students.

MTLKT-A: Indicates that attention mechanisms were used but performance bias functions were not.

MTLKT-attempt: Represents the use of a student's "number of attempts" as a parameter in the bias function to improve the attention mechanism.

From the experimental results in Table 8, we could conclude that the use of the attention mechanism improved the accuracy of the model, and the introduction of the performance bias function with appropriate improvements to the attention mechanism could better simulate the impact of student behavioral performance on knowledge state during the question-answering process, thus improving the accuracy of the prediction. MTLKT used "response time "as a parameter in the performance bias function, the performances for all three data sets were higher than that of MTLKT-attempt, and the "response time" was more representative of student performance than their "number of attempts". It had a greater impact on student knowledge state.

Table 8. Student ability extraction module ablation results.

Data Sets	Assistments09		Assistments12		Assistments17	
	AUC	ACC	AUC	ACC	AUC	ACC
MTLKT-NA	0.7503	0.7322	0.7630	0.7318	0.7739	0.7524
MTLKT-A	0.7596	0.7406	0.7685	0.7355	0.7704	0.7555
MTLKT-attempt	0.7658	0.7528	0.7710	0.7423	0.7765	0.7605
MTLKT	0.7703	0.7659	0.7780	0.7542	0.7860	0.7653

Visualization of attention weights

To better visualize the effects of the performance bias function, we randomly selected five students (S1–S5) from the test set and visualized their attention weights between the 11th question and first 10 questions (title number indicates order, not specific title) as a heatmap (Figures 6 and 7). At the same time, the performances of these students in answering these questions are given in Table 9. The contents of the table are the differences between the students' answer times and the average time. Negative numbers indicate that the time spent on the questions was lower than the average time. Positive numbers indicate that the time spent on the questions was higher than the average time (for display purposes, the sizes of the data are divided by 1000).

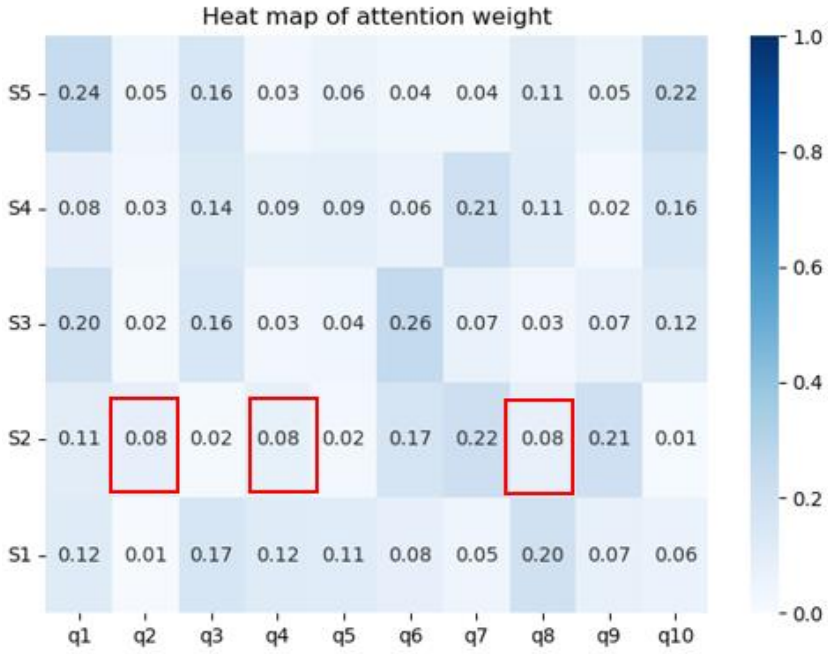

Figure 6. Initial attention weights.

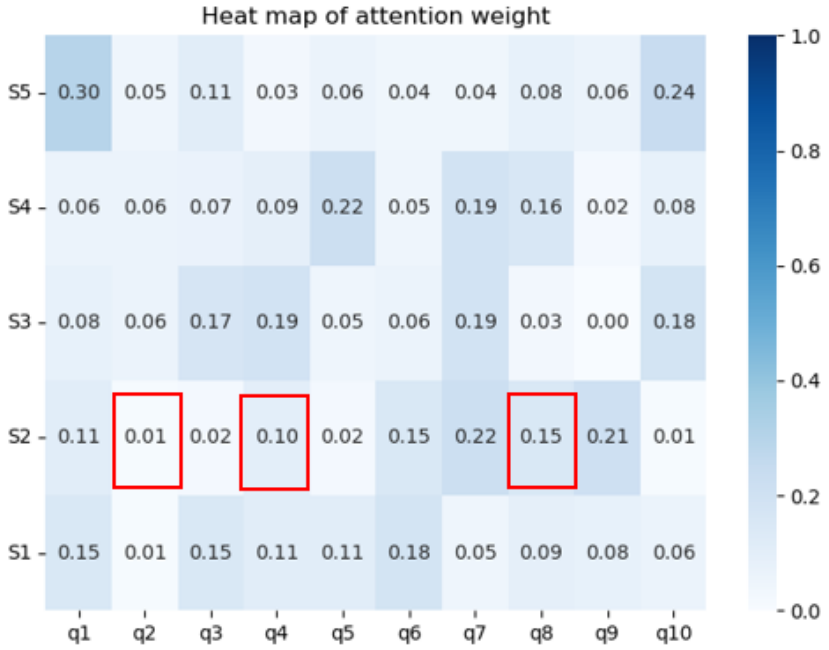

Figure 7. Attention weights after using performance bias function.

Table 9. The difference between student performance and average performance.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
S5	−10.5	−1.5	+8	−1	+0.1	+1	+0.5	+6	−0.6	−2
S4	+7.4	−8	+10.4	−0.4	−14.6	+3	−3.8	−6	0	+12
S3	+16.5	−3.6	−4.6	−12.2	+1.4	−3	+0.5	+1	+12.4	−8.4
S2	−5.2	+30	+1.8	−10.2	−5.4	+17.2	−7	−20	−6.2	+5
S1	−22	+13	+18	+16	+15	−60.5	+14	+29	−5	−17.5

It can be seen from the three red boxes in Figure 6 that, without using the performance bias function, the three questions of Q2, Q4, and Q8 in the answer record of student S2 had the same attention weight of 0.08. From Figure 7, we can see the specific performance of student S2 on these three questions. Among them, the performance on Q2 was lower than the average performance, while the performances on Q4 and Q8 were better than the average performance. Therefore, after using the performance bias function, we can see that the attention weight of Q2 decreased by 0.07, while the attention weights of Q4 and Q8 increased by 0.02 and 0.07, respectively.

The weights of other questions were also adjusted according to the students' performances shown in Table 9. The experimental results show that the improved attention mechanism after introducing the performance bias function could more accurately mine the relationship between the students' questions, as well as enhance the interpretability.

5. Conclusions

In order to extract the information difficulty of questions and student ability information accurately and to use the above information to simulate student modeling, predict student answer results, and grasp knowledge mastery level, this paper proposed a multi-task knowledge-tracking model (MTLKT) with a novel representative approach to question difficulty and student ability. We used the idea of multi-task learning to share the underlying parameters and information, fully explore the potential relationship between question difficulty and skill difficulty, and enrich the expression of question information difficulty. At the same time, considering that student answer performance is an important reflection of their ability, we improved the attention mechanism by combining student answer performance and optimizing the calculation of attention weights so that the modeling

of student ability was more in line with the law of learning. We conducted comparative experiments on three data sets, and the experimental results showed that the performance of MTLKT was better than those of the comparison models. Meanwhile, we also conducted ablation experiments on each module of the model, and the experimental results proved the effectiveness of each module.

In future work, we will try to introduce knowledge mapping, graphical neural networks, and other techniques to explore the correlations between skills and questions in order to improve the accuracy of model prediction and provide personalized teaching tutoring and educational resource recommendation for students.

https://github.com/Shehan29/FRC-2014.

Author Contributions: Conceptualization, W.Z. and K.Q.; methodology and implementation, Z.K. and S.H.; writing—original draft preparation, K.Q.; writing—review and editing, W.Z., K.Q., Z.K. and S.H. All authors have read and agreed to the published version of the manuscript.

Funding: Research on the intelligent comprehensive assessment of computational thinking for the key competence Supported by National Natural Science Foundation of China (NO. 61977031).

Institutional Review Board Statement: The study did not require ethical approval.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The Assistments09 data set supporting this study was obtained from https://sites.google.com/site/assistmentsdata/home/assistment2009-2010-data/skill-builder-data-2009-2010 (accessed on 15 September 2022); the Assistments12 data set was obtained from https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-with-affect (accessed on the 15 September 2022); the Assistments17 data set was obtained from https://github.com/Shehan29/FRC-2014 (accessed on the 15 September 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Minn, S.; Zhu, F.; Desmarais, M.C. Improving Knowledge Tracing Model by Integrating Problem Difficulty. In Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, 17–20 November 2018; IEEE: Piscataway, NJ, USA, 2018.
2. Fang, J.; Zhao, W.; Jia, D. Exercise Difficulty Prediction in Online Education Systems. In Proceedings of the 2019 International Conference on Data Mining Workshops (ICDMW), Beijing, China, 8–11 November 2019.
3. Liu, Y.; Yang, Y.; Chen, X.; Shen, J.; Zhang, H.; Yu, Y. Improving Knowledge Tracing via Pre-training Question Embeddings. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence IJCAI-PRICAI-20, Yokohama, Japan, 7–15 January 2021.
4. Shen, S.; Liu, Q.; Chen, E.; Wu, H.; Huang, Z.; Zhao, W.; Su, Y.; Ma, H.; Wang, S. Convolutional Knowledge Tracing: Modeling Individualization in Student Learning Process. In Proceedings of the SIGIR'20: The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual, 25–30 July 2020; ACM: New York, NY, USA, 2020.
5. Ghosh, A.; Heffernan, N.; Lan, A.S. Context-Aware Attentive Knowledge Tracing. *arXiv* **2020**, arXiv:2007.12324.
6. Pardos, Z.A.; Heffernan, N.T.T. Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. In Proceedings of the User Modeling, Adaptation, & Personalization, International Conference, Umap, Big Island, HI, USA, 20–24 June 2010; Springer: Berlin/Heidelberg, Germany, 2010.
7. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 8–11 December 2014; MIT Press: Cambridge, MA, USA, 2014.
8. Piech, C.; Spencer, J.; Huang, J.; Ganguli, S.; Sahami, M.; Guibas, L.; Sohl-Dickstein, J. Deep Knowledge Tracing. *Comput. Sci.* **2015**, *3*, 19–23.
9. Yeung, C.K.; Yeung, D.Y. Addressing Two Problems in Deep Knowledge Tracing via Prediction-Consistent Regularization. In *Proceedings of the Artificial Intelligence*; ACM: New York, NY, USA, 2018.
10. Zhang, J.; Shi, X.; King, I.; Yeung, D.-Y. Dynamic Key-Value Memory Networks for Knowledge Tracing. In Proceedings of the Web Conference, Perth, Australia, 3–7 April 2017; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2017.
11. Liu, S.; Zou, R.; Sun, J.; Zhang, K.; Jiang, L.; Zhou, D.; Yang, J. A Hierarchical Memory Network for Knowledge Tracing. *Expert Syst. Appl.* **2021**, *177*, 114935. [CrossRef]
12. Pandey, S.; Srivastava, J. RKT: Relation-Aware Self-Attention for Knowledge Tracing. *arXiv* **2020**, arXiv:2008.12736.

13. Nakagawa, H.; Iwasawa, Y.; Matsuo, Y. Graph-based Knowledge Tracing: Modeling Student Proficiency Using Graph Neural Network. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Thessaloniki, Greece, 14–17 October 2019; ACM: New York, NY, USA, 2019.
14. Song, X.; Li, J.; Lei, Q.; Zhao, W.; Chen, Y.; Mian, A. Bi-CLKT: Bi-Graph Contrastive Learning based Knowledge. *Tracing. Knowl.-Based Syst.* **2022**, *241*, 108274. [CrossRef]
15. Yang, Y.; Shen, J.; Qu, Y.; Liu, Y.; Wang, K.; Zhu, Y.; Zhang, W.; Yu, Y. GIKT: A Graph-based Interaction Model for Knowledge Tracing. *arXiv* **2020**, arXiv:2009.05991.
16. Sun, X.; Zhao, X.; Li, B.; Ma, Y.; Sutcliffe, R.; Feng, J. Dynamic Key-Value Memory Networks With Rich Features for Knowledge Tracing. *IEEE Trans. Cybern.* **2021**, *52*, 8235–8245. [CrossRef] [PubMed]
17. 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*, 100–115. [CrossRef]
18. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**, arXiv:1409.0473.
19. Pandey, S.; Karypis, G. A Self-Attentive model for Knowledge Tracing. *arXiv* **2019**, arXiv:1907.06837.
20. Choi, Y.; Lee, Y.; Cho, J.; Baek, J.; Kim, B.; Cha, Y.; Shin, D.; Bae, C.; Heo, J. Towards an Appropriate Query, Key, and Value Computation for Knowledge Tracing. *arXiv* **2020**, arXiv:2002.07033.
21. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
22. Pu, S.; Yudelson, M.; Ou, L.; Huang, Y. Deep Knowledge Tracing with Transformers. In *Artificial Intelligence in Education*; Springer: Cham, Switzerland, 2020; pp. 252–256.
23. Caruana, R. Multitask Learning. *Mach. Learn.* **1997**, *28*, 41–75. [CrossRef]
24. Chaudhry, R.; Singh, H.; Dogga, P.; Saini, S.K. Modeling Hint-Taking Behavior and Knowledge State of Students with Multi-Task Learning. In Proceedings of the International Conference on Educational Data Mining (EDM), Raleigh, NC, USA, 16–20 July 2018; pp. 21–31.
25. Thung, K.H.; Wee, C.Y. A brief review on multi-task learning. *Multimed. Tools Appl.* **2018**, *77*, 29705–29725. [CrossRef]
26. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA, 2016.
27. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450.
28. Chen, Z.; Badrinarayanan, V.; Lee, C.Y.; Rabinovich, A. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. *arXiv* **2017**, arXiv:1711.02257.