

# AAKT: Enhancing Knowledge Tracing with Alternate Autoregressive Modeling

Hao Zhou, Wenge Rong, Jianfei Zhang, Qing Sun, Yuanxin Ouyang and Zhang Xiong

**Abstract**—Knowledge Tracing (KT) aims to predict students’ future performances based on their former exercises and additional information in educational settings. KT has received much attention since it provides personalized experiences in educational situations. The autoregressive modeling on the sequence of former exercises has been proven effective for this task. One of the main challenges for autoregressive modeling in KT lies in the different visibility of exercise information, which leads to sacrifice on effectiveness in previous methods. To address this issue, we propose the Alternate Autoregressive Knowledge Tracing (AAKT) framework by combining question and response sequences into a unique sequence in an alternate order and using sliding windows to cut subsequences for training and testing. The model embeds exercise information into vectors through embedding layers and an auxiliary task during the training process. Advanced autoregressive technologies from Natural Language Generation (NLG) are introduced to extract information from historical interactions. The output of the autoregressive model is transformed into the probability of answering correctly for the given question. We compare AAKT with several knowledge tracing baselines on four real-world datasets and the experimental studies demonstrate its promising potential. Ablation studies and visualizations are also conducted to validate the effects of several key components of AAKT.

**Index Terms**—Educational Technology, Knowledge Tracing, Autoregressive Models, Transformer

## I. INTRODUCTION

IT is crucial for intelligent tutoring systems and online learning platforms to model students’ knowledge status over time. To trace the mastery level of each student and generate predictions of the correctness of various questions based on students’ knowledge status, a fundamental task called Knowledge Tracing (KT) has been proposed [1]. It collects the history exercise sequences of students and calculates their mastery levels to predict whether the student will answer the next exercise and enables the educators to provide personalized learning materials, weakness suggestions, and exercise recommendations.

Conventional approaches in this domain mainly include Bayesian Knowledge Tracing (BKT) using Hidden Markov

Models [1], [2], [3], Deep Knowledge Tracing (DKT) using Deep Neural Networks [4] and its derivative models [5], [6]. Although original KT models (e.g., BKT) are easy to implement and have relatively higher interpretability, they fail to capture the dynamic of knowledge status because of simplified assumptions related to the learning process.

Original DKT models take advantage of recurrent neural networks (RNNs) and mainly utilize LSTM and GRU to model students’ knowledge status [4], [7], [8]. However, RNN-based models are more concentrated on the closer exercises of certain interactions, therefore negatively affecting the capability of modeling long-term dependency. Moreover, RNNs cannot function concurrently so model training and inferring will cost relatively more time. To solve the problem of the narrow scope of attention, Sequential Key-Value Memory Networks (SKVMN) [9] use LSTM with hops to enable RNNs to capture further dependencies in the exercise sequences. However, this improvement does not fully address the issue of limited attention scope.

Transformer [10] is a novel structure of neural networks and an effective model for sequence-to-sequence prediction tasks, leveraging the mechanism of attention. Some KT models follow the idea of attention and introduce self-attention mechanism into DKT [11], [12]. The major advantage of self-attention-based models is that they can predict each interaction by taking whole historical exercises in the sequence into consideration, thus avoiding the imbalance of concentration.

Despite encouraging improvements brought by the development of DNNs, current KT models are still insufficiently satisfactory. It can be attributed to the following three factors. Firstly, previous knowledge tracing models encode question sequences and response sequences separately, particularly KT models utilizing self-attention mechanism and transformer [11], [13], [12], [14]. Although they proposed various kinds of methods to model the question sequences and response sequences, the interaction between these two types of information is insufficient, causing a loss of bilateral information. Research has shown that the final performance of AKT (self-attention-based model) cannot beat that of DKT (RNN-based model) [12]. Self-attention-based models like AKT have more parameters than DKT, however demonstrating less capability in knowledge tracing. The main reason for this phenomenon is that current self-attention-based models fail to model the sequences in an autoregressive way and effectively combine the information of questions and responses, thus wasting the models’ ability to extract information from all prior contexts.

Secondly, some KT models that combine the information of questions and skills fail to concentrate on the detail of fusion

Manuscript received xx xxxx xxxx; revised xx xxxx xxxx and xx xxx xxxx; accepted xx xxxx xxxx. This work was supported in part by the National Natural Science Foundation of China under Grant 61977002. (Corresponding author: Wenge Rong.)

H. Zhou, W. Rong, J. Zhang, Q. Sun, and Y. Ouyang are with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China. (e-mail: h.zhou@buaa.edu.cn, w.rong@buaa.edu.cn, zhangjf@buaa.edu.cn, sunqing@buaa.edu.cn, oyyx@buaa.edu.cn).

Z. Xiong is with the School of Information Technology & Management, University of International Business and Economics, Beijing 100029, China. (e-mail: xiongz@uibe.edu.cn).

Digital Object Identifier 10.1109/TLT.xxxx.xxxxxxx

and simply concatenate or add the embedding vectors together [13], [15]. This fusion method leaves the task of information extraction to other parts of the model and is not a natural way to some extent because questions and skills attached to them are not completely equivalent in students' practicing process. Therefore, regarding them as the same is negligent of the hierarchical difference between them.

Lastly, current KT models fail to utilize datasets efficiently. Every question in a certain exercise sequence is used only once when training and testing [4], [7], [11], [12], [3], causing waste of information. Therefore, the potential of the model cannot be fully demonstrated when training the model. Moreover, many previous KT models merely utilize the information of questions or skills [4], [8], [11], [12]. Some other information (including anterior information which is known before answering and posterior information which can only be known after answering) in datasets is still useful for extracting the knowledge status of students and predicting future interactions.

Considering the aforementioned limitations of current KT models, in this paper, we propose the alternate autoregressive knowledge tracing model (AAKT). Our model is based on alternate sequence construction which arranges elements of every question sequence and its corresponding response sequence in an alternate order, and introduces sliding windows to expand the volume of datasets and improve the performance of prediction. The information of questions and skills is combined by conducting an auxiliary task and calculating auxiliary loss. Moreover, additional information (i.e., time spent on answering every question) is utilized in the embedding process.

We have demonstrated that the sequential causal relationship between various information in the knowledge tracing task, such as problem information, answer history information, and answer result information, conforms to the paradigm of autoregressive prediction. Based on this, every question sequence and its corresponding response sequence are rearranged and combined in a special way in which the question indices and responses appear in an alternate order, thus explicitly separating anterior and posterior information and facilitating the autoregressive prediction process. The rearranged sequences are cut into subsequences with overlaps by sliding windows and then combined with additional information (i.e., time spent on answering questions), generating embedding vectors. These vectors are then utilized for an auxiliary task to make sure that they contain information on corresponding skill(s) mapped from question label sequences extracted from subsequences. The modified vectors are sent into an autoregressive transformer (namely the GPT-J instance in this paper). The output of the autoregressive transformer, regarded as the hidden knowledge status of students, is then transformed into prediction results via feed-forward networks. Binary cross entropy loss is calculated from predictions and response label sequences extracted from subsequences and added with KL divergence generated from the auxiliary task as the total loss of the model.

The contribution of our paper can be summarized as follows:

- To our best knowledge, in knowledge tracing models, we are the first to leverage alternate sequences to combine question sequences and response sequences. This novel

method facilitates autoregressive modeling and automatically involves effective and sufficient interaction of these two types of information.

- We introduce skill-related information into question embeddings via an auxiliary task and this method is proven better than traditional addition or concatenation strategies.
- We utilize the sliding window strategy in the preprocessing of datasets to allow overlaps in data batches. Proper masks are considered in testing status so datasets are expanded without data leakage.
- Apart from information related to questions, skills, and responses, we also utilize information about the time spent on answering questions, which is crucial information in exercises.

## II. RELATED WORKS

By adopting the Hidden Markov Model (HMM), Corbett and Anderson proposed the first knowledge tracing model (namely Bayesian Knowledge Tracing, BKT) [1]. BKT utilizes binary variables to predict the knowledge status of a student (i.e., whether a skill in the skill set is mastered or not). Moreover, it uses some assumptions concerning the distribution of some variables including response accuracy and question difficulty [16], [17]. It mainly introduces four factors to achieve the goal: initial knowledge status, learning rate, guessing probability, and slipping probability, and it predicts knowledge status based on the immediate last response. Subsequently, other researchers expanded the original BKT model by introducing other factors such as problem difficulty or the individual ability of learning [18], [19]. Furthermore, Knowledge Tracing Machine (KTM) [20] takes advantage of factorization machine [21] to establish a logistic model and generate pairwise interactions of skills. These traditional knowledge tracing models aim to improve the performance and interpretability of the original BKT model. However, the performance of BKT models does not hide the flaw in the presented assumptions. These simplified assumptions have relatively high interpretability and are easier to understand but limit the capability of capturing the dynamic of knowledge status and fail to thoroughly explain the students' process of practicing, negatively influencing the overall performances of models.

To address the issues of traditional KT models, Piech et al. [4] was the first to introduce deep neural networks into knowledge tracing (i.e., the DKT model). DKT utilized long-short term memory recurrent neural networks (LSTM) [22] to model students' knowledge status and outperformed BKT and its extended variants. Considering the prediction consistency of exercise sequences, DKT+ [7] improved the DKT model by adding a regularization on the hidden state of LSTM to smooth the evolving of knowledge status. Following the pioneering work of DKT, more researchers explored the function of neural networks in knowledge tracing tasks. Zhang et al. [8] proposed a dynamic key-value memory network (DKVMN) which utilized external memory matrices to store the memory of knowledge and update the corresponding mastery level. Inspired by DKVMN, Abdelrahman and Wang [9] introduced

modified LSTM (i.e., hop-LSTM) on top of the dynamic key-value network in their model. Guo et al. [15] leveraged adversarial training in knowledge tracing (ATKT) to improve the robustness of the model and avoid overfitting in small datasets. Wang et al. [23] combine educational priors and RNN in KT model to enhance interpretability.

With the success of attention mechanism and transformer [10] in natural language processing and computer vision tasks, self-attention-based knowledge tracing models have been proposed. Pandey and Karypis [11] first introduced self-attention mechanism into KT models (i.e., SAKT). SAKT can capture the long-term dependency of history exercises. Pu et al. [14] proposed a KT model based on the vanilla transformer. Furthermore, Ghosh et al. [12] proposed a context-aware attentive KT model with monotonic attention and exponential decay mechanism. More recently, Cui et al. [24] designed multi-relational transformer to enable fine-grained interaction modeling between question-response pairs.

Despite these breakthroughs, self-attention-based KT models fail to efficiently combine the information of questions and responses. They encompass various design features trying to model these two types of information, but the interactive relation of them is partly neglected. In SAKT [11], history interactions (including questions and responses) and future questions are encoded to different sequences, causing inconsistency of semantic spaces. AKT [12] calculates the self-attention inside question sequences and response sequences respectively, then utilizes a knowledge retriever to partly combine two types of information. However, the information of responses does not participate in calculating attention weights in knowledge retrievers, causing insufficient information interaction. MRT-KT [24] defines multiple kinds of relation between different exercises according to their question skill sets and response results, but gives up including the response information as the encoded input for lack of effective autoregressive sequence construction method.

To further improve the competence of understanding the students' knowledge status and the performance of KT models, we believe that interactions between different types of information should be fully emphasized without losing causal relationships in sequences. Following this line of thought, we analyzed the causal relationship of in knowledge tracing task, and therefrom propose an alternate autoregressive modeling method based on a novel sequential representation of question information and response information.

### III. METHODOLOGY

In this section, we introduce the details of our framework AAKT. The overall architecture is shown in Figure 1. Firstly, every question sequence and its corresponding response sequence are combined in an alternate order. Moreover, the combined alternate sequence is cut into subsequences by sliding window. Response label sequences and question label sequences are extracted from these subsequences. Additional information is added together with the embedded subsequences generating final embedding vectors which will be sent into an autoregressive transformer. Then autoregressive

transformer structure handles the final embedding sequences and extracts the hidden knowledge status of students. Apart from this, an auxiliary task is conducted by utilizing a multi-hot skill matrix mapped from the question label sequence. Finally, an auxiliary loss (i.e., KL divergence) of the normalized skill matrix and predicted skill distribution (transformed from question embeddings extracted from embedding vectors) is calculated.

#### A. Problem Formulation

The notations needed to formulate the target of knowledge tracing task are listed in Table I.

TABLE I  
NOTATIONS USED FOR PROBLEM FORMULATION

Notation	Description
$\mathcal{Q} = \{q_1, q_2, \dots\}$	Question set
$\mathcal{T} = \{t_1, t_2, \dots\}$	Skill set
$q_i$	The $i$ -th question of $\mathcal{Q}$
$t_i$	The $i$ -th skill of $\mathcal{T}$
$t_i^*$	The skill set of the $i$ -th question
$\mathcal{I}$	The question sequence of a student
$\tilde{q}_i \in \mathcal{Q}$	The question in $\mathcal{I}$ at step $i$
$\tilde{t}_i \in \mathcal{T}$	The skill set in $\mathcal{I}$ at step $i$
$\tilde{a}_i \in \mathbb{R}$	The posterior information in $\mathcal{I}$ at step $i$
$\tilde{c}_i \in \{0, 1\}$	The correctness in $\mathcal{I}$ at step $i$
$\text{Ele}_j = (\tilde{q}_j, \tilde{t}_j, \tilde{a}_j, \tilde{c}_j) \in \mathcal{I}$	The element of $\mathcal{I}$ at step $j$

Given the exercise sequence  $\mathcal{I}$  of a certain student, knowledge tracing seeks to monitor the student's evolving knowledge status by predicting the correctness of future questions. Therefore, suppose that the model has already known the data of  $\mathcal{I} = \{(\tilde{q}_1, \tilde{t}_1, \tilde{a}_1, \tilde{c}_1), \dots, (\tilde{q}_{|\mathcal{I}|}, \tilde{t}_{|\mathcal{I}|}, \tilde{a}_{|\mathcal{I}|}, \tilde{c}_{|\mathcal{I}|})\}$ , it should then predict the probability of answering the next question correctly, namely  $\Pr\{\tilde{c}_{|\mathcal{I}|+1} = 1 \mid \mathcal{I}, \tilde{q}_{|\mathcal{I}|+1}, \tilde{t}_{|\mathcal{I}|+1}\}$  ( $\tilde{a}_{|\mathcal{I}|+1}$  is posterior information so we do not know it before the question is answered).

#### B. Preliminaries

In general, the autoregressive model describes a system whose status (dependent variable) depends linearly on its status in the past. The system can be mathematically described by a stochastic difference equation such as the following:

$$y_t = \beta_0 + \sum_{i=1}^p \beta_i y_{t-i} + \epsilon_t. \quad (1)$$

Here,  $\beta_i$  is constant and describes how status  $i$  steps ago will impact current values. Normally, one would expect  $\{\beta_i\}_{i=0}^p$  to decrease as  $i$  increases, that is, the events that happen further in the past have less impact on current events. Anything that happens earlier than  $p$  steps ago will have no impact and the model is then noted as  $AR(p)$ , where  $\epsilon_t$  denotes a "noise" term that describes some random events that affect the status of the system. The "noise" term is often required to be stationary to make lots of statistical estimators valid (least-square estimation, maximum-likelihood estimation, etc.).

In deep learning models, autoregressive models (AR models) are mainly used in natural language processing, such

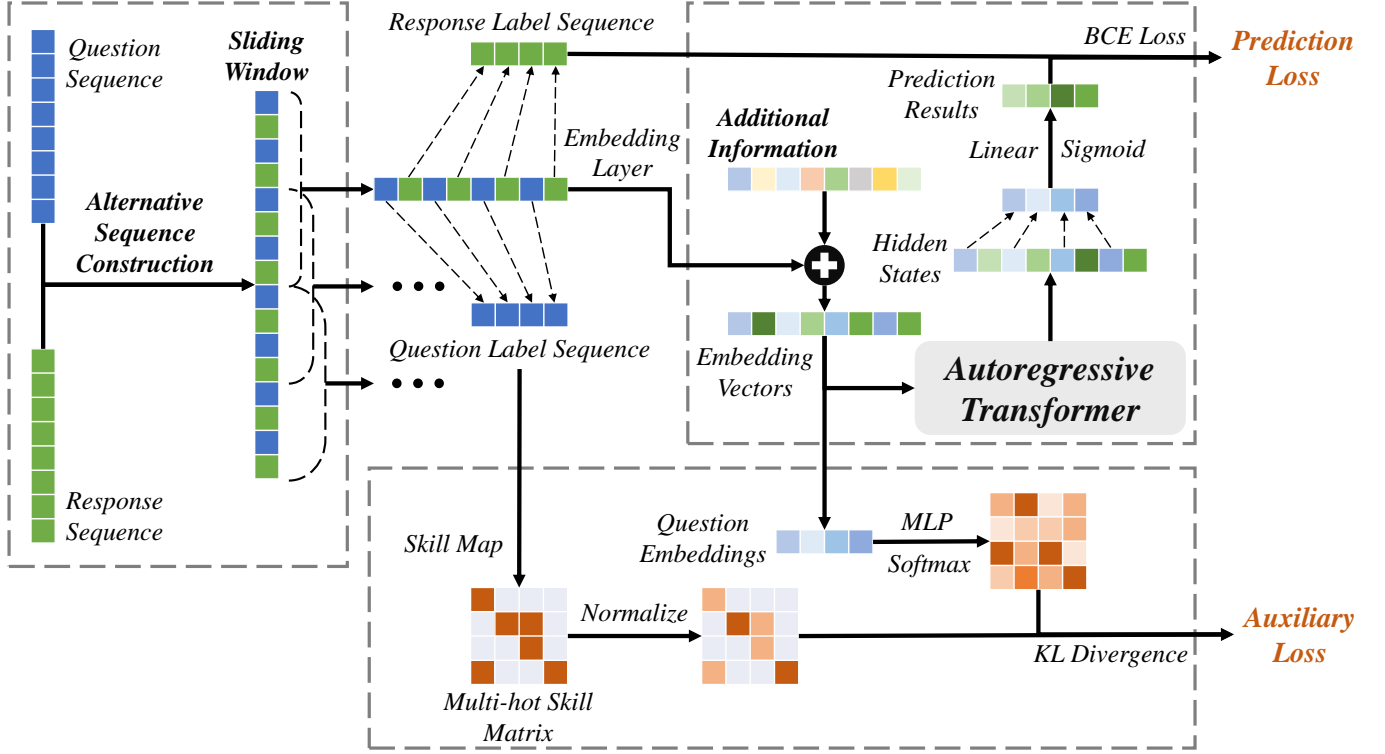


Fig. 1. Overall architecture of the proposed AAKT framework. Our framework can be generally divided into three parts. The first part includes **Alternate Sequence Construction** that combines the question sequence and the question and response sequences into a unique sequence, followed by **Sliding Window** which selects subsequences for training and prediction from the alternate sequence. The second part injects the **Additional Information** into the sequential embeddings, captures the dependencies between the sequence elements through an autoregressive transformer, and predicts the response on each question in sequence, where a Binary Cross Entropy (BCE) loss is calculated based on the prediction results and the response label sequences. The third part forces the question embeddings to learn the skill information through predicting the skill distribution for each question and minimizing its Kullback–Leibler divergence (KL divergence) to the actual skill distribution.

as GPT series [25], [26], [27]. Unlike original AR models, nonlinear transformation is introduced as a special extension (RNN, Transformer, etc.) to enable more flexible contextual interactions. AR models are unidirectional and can only see preceding information when processing specific sequences. Moreover,  $\{\beta_i\}_{i=0}^p$  in Equation 1 are not constant and they depend on inputs of the model. They learn from previous steps and take previous outputs as inputs to predict the next step recurrently. Therefore, AR models show excellent performance in prediction tasks or generative tasks.

When processing sequential information, AR models predict the next step by analyzing previous words and estimating the probability distribution of the whole corpus provided. Formally, if the corpus  $\mathcal{C}$  and a sequence  $\mathbf{x} = \{x_1, x_2, \dots, x_n\} (\forall i \in \{1, \dots, n\}, x_i \in \mathcal{C})$  is given, the target of AR models can be described as:

$$\begin{aligned} & \max_{\Theta} \mathcal{L}(\mathbf{x}) \\ &= \sum_{i=1}^n \log \Pr(x_i | \mathbf{x}_{<i}; \Theta) \\ &= \sum_{i=1}^n \log \frac{\exp(\text{hidden}_{\Theta}(\mathbf{x}_{<i})^{\top} \cdot \text{emb}(x_i))}{\sum_{x \in \mathcal{C}} \exp(\text{hidden}_{\Theta}(\mathbf{x}_{<i})^{\top} \cdot \text{emb}(x))}, \end{aligned} \quad (2)$$

where  $\text{hidden}_{\Theta}(\cdot)$  and  $\text{emb}(\cdot)$  represent hidden states of model and embedding vector respectively.

Similarly, interaction sequences in knowledge tracing tasks include causal relationships where the answering result current question is dependent on previous questions and responses. Therefore, we propose that autoregressive models can be utilized in knowledge tracing tasks.

### C. Alternate Autoregressive Modeling

1) *Alternate Sequence Construction*: Some current KT models modify the raw datasets into two sequences, namely history exercise sequence and query sequence [11], [15], [28]. According to the notations listed in Table I, the exercise sequence  $\mathcal{I}$  is split into:

$$\begin{aligned} \mathcal{I}_h &= \{h_1, \dots, h_{|\mathcal{I}|-1}\} \\ &= \{\mathbf{f}_h(\tilde{q}_1, \tilde{t}_1, \tilde{c}_1), \dots, \mathbf{f}_h(\tilde{q}_{|\mathcal{I}|-1}, \tilde{t}_{|\mathcal{I}|-1}, \tilde{c}_{|\mathcal{I}|-1})\}, \\ \mathcal{I}_q &= \{\text{qry}_2, \dots, \text{qry}_{|\mathcal{I}|}\} \\ &= \{\mathbf{f}_{\text{qry}}(\tilde{q}_2, \tilde{t}_2), \dots, \mathbf{f}_{\text{qry}}(\tilde{q}_{|\mathcal{I}|}, \tilde{t}_{|\mathcal{I}|})\}. \end{aligned} \quad (3)$$

here  $\mathbf{f}_h(\cdot)$  and  $\mathbf{f}_{\text{qry}}(\cdot)$  denote functions that embed arguments into a vector (but not all arguments are necessarily used). These models learn from students' history exercise sequences and transfer them into hidden knowledge status of every step first. Then query sequences that have a shift of one step are combined with history sequences, generating predictive results of future exercises:

$$\hat{c}_i = \text{Predict}(\mathcal{I}_{h,<i}, \text{qry}_i) \quad (4)$$

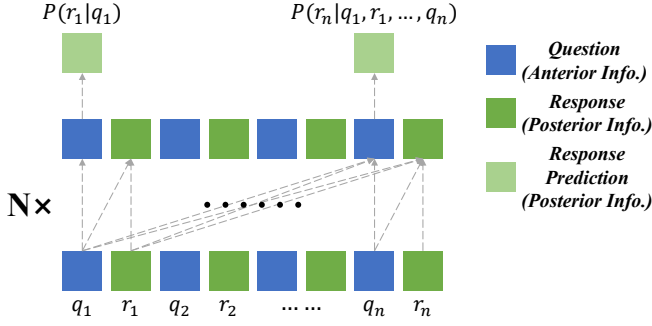


Fig. 2. In autoregressive models, each element in the sequence can attend to all preceding elements (as well as itself) at each layer. With our proposed Alternating Sequence Construction,  $q_n$  can attend to  $\{(q_i, r_i) | i < n\}$  and  $r_n$  can further attend to  $q_n$  (as well as themselves), which maximizes the scope of information acquisition for each element under the condition of following the causal relationship between all elements.

where  $2 \leq i \leq |\mathcal{I}|$  and  $\text{Predict}(\cdot)$  denotes prediction layers of models. Two embedding sequences with overlap of the same information imply that two different feature spaces are used. Therefore, the same information in the original sequence is represented in two different forms using two encoders, thus causing inconsistency and limiting the performance of models. Moreover, insufficient interaction of questions and responses causes limited ability to encode representation.

Taking information consistency and interaction into consideration, we borrow the idea of autoregressive models and modify the original exercise sequences in a novel way. Similar to the representation shown in Figure 1, the input sequence is defined as follows (notations are borrowed from Table I):

$$\mathcal{I}_i^{\text{in}} = \begin{cases} \mathbf{f}_r(\tilde{q}_k, \tilde{t}_k) & , i = 2k - 1 \\ \mathbf{f}_q(\tilde{c}_k, \tilde{a}_k) & , i = 2k \end{cases} \quad (1 \leq k \leq \frac{|\mathcal{I}|}{2}), \quad (5)$$

where  $\mathbf{f}_q(\cdot)$  and  $\mathbf{f}_r(\cdot)$  denote embedding functions of questions (information of exercises and skills) and responses (correctness and posterior information) respectively. From Equation 5, it is clear that every element in the original sequences is embedded only once. Therefore, the input sequence uses a united embedding space and improves information consistency. **Detailed explanation of  $\mathbf{f}_q(\cdot)$  and  $\mathbf{f}_r(\cdot)$  is demonstrated in Subsection III-D and III-E.**

Moreover, this kind of representation guarantees the full interaction between questions and responses in the processing of autoregressive models (see Figure 2). At step  $i = 2k - 1$  (where  $\mathcal{I}_i^{\text{in}} = \mathbf{f}_r(\tilde{q}_k, \tilde{t}_k)$ ), the autoregressive transformer will see every previous steps (namely  $\mathcal{I}_{<2k-1}^{\text{in}}$  including the current step), therefore generating the prediction of hidden knowledge status at step  $i + 1 = 2k$  (where  $\mathcal{I}_{i+1}^{\text{in}} = \mathbf{f}_r(\tilde{c}_k, \tilde{a}_k)$ ) which is transformed into prediction of correctness afterward. However, the prediction of step  $2k + 1$  (where  $\mathcal{I}_{2k+1}^{\text{in}} = \mathbf{f}_r(\tilde{q}_{k+1}, \tilde{t}_{k+1})$ ) is useless because it is meaningless for the KT model to predict the next question provided by online exercise platforms or other unrelated sources. Therefore, these useless prediction

results are excluded from the final output:

$$\begin{aligned} \mathcal{I}_i^{\text{out}} &= \hat{c}_i \\ &= \Pr(\tilde{c}_i = 1 | \mathcal{I}_{\leq 2i-1}^{\text{in}}). \end{aligned} \quad (6)$$

To be more specific, one step of hidden knowledge status (a vector with more dimensions) is transformed into a 2D vector  $\mathbf{v} = (v_{\text{correct}}, v_{\text{incorrect}})$  by a feed-forward network. We interpret the first and second dimension of  $\mathbf{v}$  as the 'raw weight' of answering correctly and incorrectly respectively. Then the probability is calculated as follows:

$$\begin{aligned} \Pr(\tilde{c}_k = 1 | \mathcal{I}_{\leq 2k-1}^{\text{in}}) &= \text{Sigmoid}(v_{\text{correct}} - v_{\text{incorrect}}) \\ &= \frac{1}{1 + \exp(v_{\text{incorrect}} - v_{\text{correct}})}. \end{aligned} \quad (7)$$

Finally, we use binary cross-entropy loss as the prediction loss:

$$\mathcal{L}_{\text{pred}} = -\frac{1}{|\mathcal{I}|} \sum_{i=1}^{|\mathcal{I}|} \hat{c}_i \cdot \log(\hat{c}_i) + (1 - \hat{c}_i) \cdot \log(1 - \hat{c}_i). \quad (8)$$

**2) Sliding Window Technique:** To make the most of the information in the datasets, we leverage the sliding window technique in preprocessing. It allows overlaps both in the training dataset and testing dataset. However, due to the consideration of validity in the testing process and metrics calculation, the concrete steps are slightly different in training and testing. The overall sliding window technique is demonstrated in Figure 3. As shown in the figure, we set two hyperparameters: max sequence length and overlap ratio. Max sequence length denotes the maximum length in a data batch and overlap ratio denotes how much the next sliding window covers the current one.

For the training dataset, the raw sequence (see Equation 5) of one student is cut into many subsequences. The window starts sliding from the beginning of raw sequence and goes all the way towards the end, each step moving  $\text{msl} \cdot (1 - r_{\text{overlap}})$  elements where  $\text{msl}$  and  $r_{\text{overlap}}$  denote max sequence length and overlap ratio respectively. Also,  $\text{int}(\text{msl} \cdot (1 - r_{\text{overlap}}))$  must be even because one full interaction includes two elements in the sequence (question and answer). If the range of the last window exceeds the raw sequence, the residue should be masked, thus not taken into consideration when calculating loss. This technique successfully expanded the volume of datasets by introducing overlaps:

$$|\text{Dataset}'| \approx \frac{1}{1 - r_{\text{overlap}}} |\text{Dataset}|. \quad (9)$$

Handling testing dataset and calculating metrics are a little bit more difficult. All the overlaps in the raw sequence for testing should be masked at corresponding positions (see Figure 3(b)) because we do not want the same interaction to be counted twice or more when calculating metrics, causing inconsistency and inaccuracy. The sliding window technique in testing is proven successful in improving the performance of prediction (see Section IV).

#### D. Auxiliary Task in Embedding

To combine the information of both questions and related skills, we propose a method of auxiliary task in our model. We

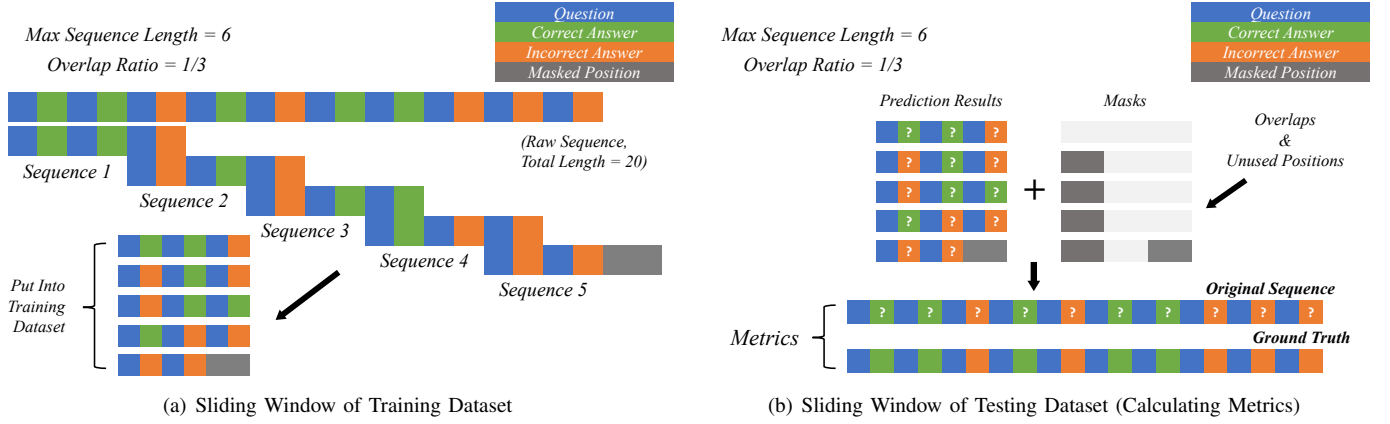


Fig. 3. Overall illustration of sliding window technique in our model. The methods for the training dataset and testing dataset are slightly different.

	Discrete	Continuous
Anterior	Related Skills Concrete Subject Student's Gender Difficulty Level (Labeled in Advance)	Timestamp of Starting
Posterior	Count of Hints/Attempts Used Student's Confidence Level	Timestamp of Finishing Total Time Spent Average Correct Rate

Fig. 4. Four categories of interaction-related information and examples.

assume that questions and skills have hierarchical differences and should be dealt with differently. Moreover, the information of questions is more concrete than that of skills and should be predominant.

We simply transform question indices into vectors using an embedding layer:

$$W^q \in \mathbb{R}^{|\mathcal{Q}|} \times \mathbb{R}^{dim}$$

$$\text{emb}(\tilde{q}_i) = W_{\tilde{q}_i}^q \in \mathbb{R}^{dim}, \quad (10)$$

where  $dim$  denotes the dimension of embedding vectors. After that, a simple feed-forward network is used to transfer the question embedding vectors into another space, preparing for the auxiliary task. Then the skill sets are embedded into multi-hot vectors as ground truth. Finally, the transferred question embedding vectors are compared with multi-hot skill vectors, generating auxiliary loss which is KL divergence in our model. The full process can be described as follows (some notations

are borrowed from Table I):

$$\begin{aligned} \text{emb}'(\tilde{q}_i) &= \text{FNN}(\text{emb}(\tilde{q}_i)) \in \mathbb{R}^{|\mathcal{T}|} \\ W^{sk} &\in \mathbb{R}^{|\mathcal{Q}|} \times \mathbb{R}^{|\mathcal{T}|} \\ W_{ij}^{sk} &= \begin{cases} \frac{1}{|t_i^*|} & , t_j \in t_i^* \\ 0 & , t_j \notin t_i^* \end{cases} \\ \text{emb}(\tilde{t}_i) &= W_{\tilde{t}_i}^{sk} \in \mathbb{R}^{|\mathcal{T}|} \\ \text{GT} &= \{\text{emb}(\tilde{t}_1), \dots, \text{emb}(\tilde{t}_{|\mathcal{I}|})\} \\ \text{Prediction} &= \{\text{emb}'(\tilde{q}_1), \dots, \text{emb}'(\tilde{q}_{|\mathcal{I}|})\} \\ \mathcal{L}_{aux} &= \text{KLDivergence}(\text{GT} || \text{Prediction}). \end{aligned} \quad (11)$$

Therefore,  $\mathbf{f}_r(\tilde{q}_k, \tilde{t}_k)$  in Equation 5 is explained clearly. It is important and obvious that  $\forall i \in \{1, \dots, |\mathcal{Q}|\}$ ,  $t_i^* \neq \emptyset$ , therefore limiting this method to datasets where every question has its related skill set.

The reason why the auxiliary task can enable the information of skills flows into the embedding vectors of question is that the backpropagation process will bring the auxiliary loss back to the embedding weights of questions. Considering the hierarchical features, we make a pathway for information on skills rather than adding it directly to the embedding vectors of questions. This method is an improvement both conceptually and empirically (see Section IV).

Similar to the prediction loss mentioned in Subsection III-C, the auxiliary loss is regarded as one part of the total loss of the model. Therefore, to jointly learn all parameters in our model, the following loss is optimized:

$$\mathcal{L} = \mathcal{L}_{pred} + \mathcal{L}_{aux}. \quad (12)$$

#### E. Processing of Additional Information

In online education platforms, an interaction record contains various kinds of attributes including timestamps, gender of student, etc. Taking these attributes into consideration in knowledge tracing tasks can be helpful because it is common sense that mere question indices and skill sets are insufficient to fully describe the characteristics of the exercises provided. Besides these attributes, some KT models even utilize semantic information such as the textual description of questions [29], [30]. Apart from semantic information, interaction-related

information can be divided into four categories which are displayed in Figure 4.

Anterior and posterior information are easy to deal with respectively. As shown in Figure 2, a full description of specific interaction includes two elements in the input sequence. Anterior information should be combined with  $q_i$  and posterior information should be combined with  $r_i$  at step  $i$ .

The method of information combination is more important than the position. For discrete information, a common way to deal with it is to transfer it into embedding vectors. However, it is more difficult to handle continuous information and our method is described as follows.

In our model, we only utilized one category: posterior continuous information (i.e., ‘total time spent’) because of the importance of time spent on exercises in online education and the characteristics of datasets. The values of total time spent range from  $0ms$  to about  $4 \times 10^5ms$ . After clipping all the values to the range of  $0ms$  to  $2 \times 10^5ms$ , we set a constant called **Time Factor** ( $\tau = 60000ms$ ) to partly normalize them:

$$\text{time}_i^{norm} = \frac{\text{time}_i}{\tau}. \quad (13)$$

After normalization, these values become dimensionless and can be fed into the model (here  $\tilde{\text{time}}_i^{norm} \in \tilde{a}_i$ ). To be more specific, the normalized time values are multiplied by a constant vector  $\mathbf{v}_{time}$  and then added to the embedding vectors of  $r_i$  (please refer to Equation 5):

$$\begin{aligned} \tilde{a}_i &= \text{time}_i^{norm}, \\ \mathbf{f}_q(\tilde{a}_i, \tilde{c}_i) &= \text{emb}(\tilde{a}_i) + \text{emb}(\tilde{c}_i) \\ &= \begin{cases} \tilde{a}_i \cdot \mathbf{v}_{time} + \text{emb}_{right} & , \tilde{c}_i = 1 \\ \tilde{a}_i \cdot \mathbf{v}_{time} + \text{emb}_{wrong} & , \tilde{c}_i = 0 \end{cases} \end{aligned} \quad (14)$$

#### IV. EXPERIMENTS

In this section, we conduct a series of experiments on real-world datasets to evaluate the performance of our model<sup>1</sup>. Moreover, we show some ablation studies and visualizations to further validate the key components in AAKT architecture.

##### A. Datasets

Four real-world datasets with information on total time spent on every question are used to evaluate the performance of AAKT. The statistics of four datasets are shown in Table II and 5. We randomly select 80% of students for training and the remaining 20% for testing.

1) *EdNet*: This dataset<sup>2</sup> is collected and introduced by [31]. Because the whole dataset is too large, we only choose EdNet-KT1 dataset which consists of students’ practice history logs, and selected the records of 5,000 students. There are in total 188 skills and 12,276 questions. The largest number of concepts underlying one question is 6.

2) *ASSISTments2009*: This dataset is one of the most popular benchmark datasets in KT field which is provided by the ASSISTments platform [32]. The version in our paper is conducted using the latest updated version ‘Skill-builder’. There are 4,162 students, 17,751 questions, and 112 concepts. We obtain 337,285 observed responses in total.

3) *ASSISTments2017*: The dataset<sup>3</sup> is obtained from the ASSISTments data mining competition 2017. We do the same preprocessing as ASSISTments2009.

4) *Junyi*: **JunyiAcademy**<sup>4</sup> [33] collects data on JunyiAcademy platform from 2018/08 to 2019/09. After preprocessing, the dataset has a total of 721 exercises and 25,925,983 interactions, with 247,605 users.

##### B. Baselines and Evaluation Metrics

We compare AAKT with the following advanced KT models.

- DKT [4] is the first knowledge tracing model that utilizes deep neural networks. It uses long-short term memory recurrent neural network (i.e., LSTM) to generate sequential prediction of students’ performance. The hidden state of the LSTM can be extracted to infer the student’s mastery level of different skills.
- DKT+ [7] is an improvement of DKT. It adds a regularization term on the hidden states of students, smoothing the evolving of knowledge status.
- DKVMN [8] extends DKT by introducing extra memory-augmented neural networks. Two matrices (namely key matrix and value matrix) are used to better trace students’ hidden knowledge status. Moreover, DKVMN utilizes the ‘read’ and ‘write’ processes to emulate the process of learning.
- SAKT [11] is the first KT model using self-attention mechanism. Its overall structure resembles that of the transformer.
- AKT [12] adopts a novel monotonic attention mechanism to model the influence of interaction from the distant past on students’ knowledge status. It also introduces the difficulty coefficient when encoding skills.
- KQN [28] introduces a novel concept called probabilistic skill similarity that relates the pairwise cosine and Euclidean distances between skill vectors to the odds ratios of the corresponding skills.
- ATK [15] adopts adversarial training with attentive LSTM to model students’ knowledge status. The training process of the model contains perturbation to improve robustness.

In this paper, the metric used to evaluate the performance of models is **Area Under ROC Curve** (AUC) which is widely used in knowledge tracing tasks [4], [8], [12]. A higher value of AUC indicates better performance in predicting students’ responses, and the value of 0.5 represents the performance of making predictions by random guessing.

<sup>1</sup>The code is available at <https://github.com/alxzzhou/AAKT>

<sup>2</sup>This dataset is available at: <https://github.com/riiid/ednet>

<sup>3</sup>This dataset is available at: <https://sites.google.com/view/assistmentsdatamining/data-mining-competition-2017>

<sup>4</sup><https://www.junyiacademy.org/>



TABLE II  
DATASETS STATISTICS

Features/Datasets	EdNet-KT1	ASSISTments2009	ASSISTments2017	Junyi
#Students	5,000	4,162	1,709	247,605
#Questions	12,276	17,751	3,162	721
#Skills	188	112	99	40
#Records	2,208,254	337,285	942,540	25,925,983
#Avg. Questions per Skill	65.30	158.49	31.94	18.02
#Avg. Skills per Question	2.14	1.00	1.00	1.00
%Correct Answering Records	69.88%	65.31%	37.27%	82.79%
%Incorrect Answering Records	30.12%	34.69%	62.73%	17.21%
%Sequences within Length (0,10)	0.00%	52.86%	0.00%	49.56%
%Sequences within Length [10,100)	23.75%	31.69%	3.63%	33.50%
%Sequences within Length [100,1000)	53.03%	14.20%	82.67%	14.80%
%Sequences within Length [1000,10000)	22.06%	1.25%	13.70%	2.13%
%Sequences within Length [10000,+∞)	1.16%	0.00%	0.00%	0.01%

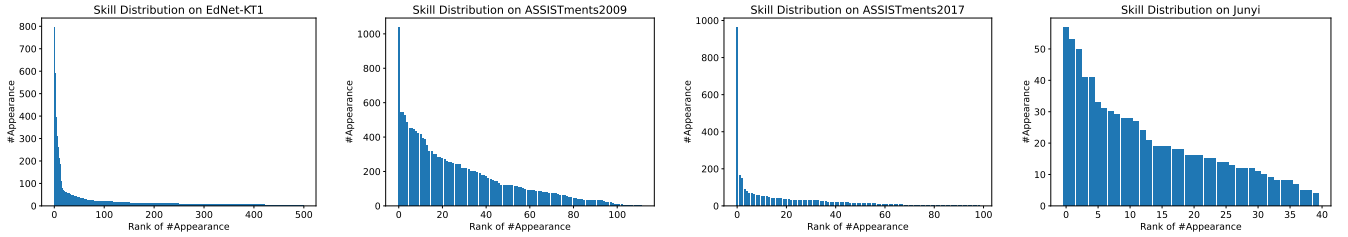


Fig. 5. Skill distribution on every dataset. For EdNet-KT1, we define that two questions are the same in terms of skill if and only if their skill sets are identical. Moreover, considering the extremely long and thin tail of skill distribution on EdNet-KT1, we truncate the list to 500 records.

### C. Implementation

1) *Details of Autoregressive Transformer:* In our model, we utilize GPT-J [34] instance (without pre-trained weights) as the autoregressive transformer. The design of GPT-J closely follows GPT-3 Curie [27]. However, two minor architectural improvements have been made:

- Rotary Positional Embedding (RoPE) [35] is adopted for better performance.
- The attention layer and the feedforward layer are placed in parallel for decreased communication.

2) *Other Details:* Our AAKT model is implemented with PyTorch and we use Adam optimizer to train our model, setting learning rate and batch size to 0.001 and 64 for all datasets. All experiments are conducted on a server with one NVIDIA A5000 (24GB) graphic card. Balancing time cost and performance, we set the overlap ratio  $r_{overlap}$  to 0.5 both in training and testing. Moreover, the implementations of baselines are based on open-source code provided by corresponding authors and pyKT [36], a comprehensive python based benchmark platform for knowledge tracing tasks.

Details of other hyperparameters are listed below:

- Dimension of embedding is chosen from {64, 96, 128, 256}.
- Max sequence length is chosen from {20, 50, 100, 300, 500, 1000, 2000} according to the distribution of sequences in each dataset.
- The number of heads in multi-head attention is set to 8.
- The number of GPT-J blocks is set to 2.
- The dimension of rotary position embedding is set to  $\frac{d_{emb}}{2n_{head}}$  where  $d_{emb}$  and  $n_{head}$  denotes embedding di-

mension and number of heads in multi-head attention respectively.

### D. Results and Discussions

In this section, we list the average results of 5 trials in Table III ( $\times$  means that the model does not fit the dataset), where the highest results are displayed in bold and the second highest results are underlined. Our model AAKT achieves the highest on all four datasets. From the data shown in the table, the following observations can be made:

- AAKT outperforms all the state-of-the-art models on four datasets (i.e., EdNet-KT1, ASSISTments2009, ASSISTments2017, and Junyi), validating its effectiveness in knowledge tracing tasks.
- DKT+ achieves the second highest metric on EdNet-KT1 and ASSISTments2017, outperforming models with more complicated structures like DKVMN. DKT and DKT+ are very simple models which merely utilize LSTM structure. This implies that more skillful techniques should be developed in modeling the learning sequences of students for knowledge tracing and complexity does not necessarily lead to accuracy.
- AKT achieves the second highest metric on ASSISTments2009 and Junyi. The introduction of transformers enables AKT to outperform SAKT which utilizes vanilla self-attention mechanism.
- ATKT has a relatively good performance on smaller datasets (e.g., ASSISTments2009). This can partly be attributed to the adversarial design of ATKT. The adversarial examples from the adversarial attack create noise



TABLE III  
THE PERFORMANCE OF AAKT AND ALL BASELINES ON 4 DATASETS

	EdNet-KT1	ASSISTments2009	ASSISTments2017	Junyi
DKT	0.76082	0.75248	0.76140	0.80296
DKT+	0.76397	0.75463	0.77579	0.80252
DKVMN	0.75863	0.73532	0.73163	0.80029
SAKT	0.76164	0.71030	0.73212	0.79951
AKT	0.74235	0.79333	0.76950	0.80565
KQN	×	0.73383	0.70652	0.75425
ATKT	×	0.75356	0.71175	0.75408
AAKT (Ours)	<b>0.77683</b>	<b>0.80524</b>	<b>0.80168</b>	<b>0.81358</b>

TABLE IV  
THE PERFORMANCE OF ORIGINAL AAKT MODEL AND ITS VARIANTS

	EdNet-KT1	ASSISTments2009	ASSISTments2017	Junyi	ASSISTments2009 (Filtered)	Junyi (Filtered)
AAKT	<b>0.77683</b>	<b>0.80524</b>	<b>0.80168</b>	<b>0.81358</b>	<b>0.78216</b>	<b>0.80974</b>
AAKT w/o Sliding	0.76910	0.78727	0.78273	0.79976	0.77232	0.78873
AAKT w/o Time	0.77134	0.79167	0.79742	0.80863	0.77054	0.79548
AAKT w/o Skill	0.77256	0.79852	0.78459	0.80804	0.77410	0.79256
AAKT w/o Time&Skill	0.76896	0.79071	0.78191	0.80365	0.76203	0.78925
AAKT w/o Auxiliary Task	0.76794	0.79794	0.78231	0.80516	0.76801	0.79487

to confuse the model during training, making the model more robust on small datasets.

As shown in Table II, about half of sequences in ASSISTments2009 and Junyi are shorter than 10 steps. Moreover, about 20% of students in ASSISTments2009 only answered one question. The distribution of sequence length in these two datasets may cause bias in model prediction because extremely short sequences are predominant and some models are sensitive to short sequences. To eliminate the influence of extremely short sequences, we remove sequences shorter than 10 steps in ASSISTments2009 and Junyi and compare the performances of different models again on them (see Table V). This operation is also seen in EKT [37]. These processed datasets are marked as ‘**Filtered**’.

TABLE V  
THE PERFORMANCE OF AAKT AND ALL BASELINES ON FILTERED ASSISTMENTS2009 AND JUNYI

	ASSISTments2009 (Filtered)	Junyi (Filtered)
DKT	0.71723	0.80298
DKT+	0.72352	0.80213
DKVMN	0.69318	0.79352
SAKT	0.71529	0.80196
AKT	0.76277	0.80631
KQN	0.75006	0.74695
ATKT	0.75657	0.75013
AAKT (Ours)	<b>0.78216</b>	<b>0.80974</b>

Generally, the performances on filtered ASSISTments2009 and Junyi show a little bit decrease. Our model ranks first on these two datasets. Moreover, AKT still ranks second compared to all baselines and our model, indicating its sufficient competence.

#### E. Ablation Study

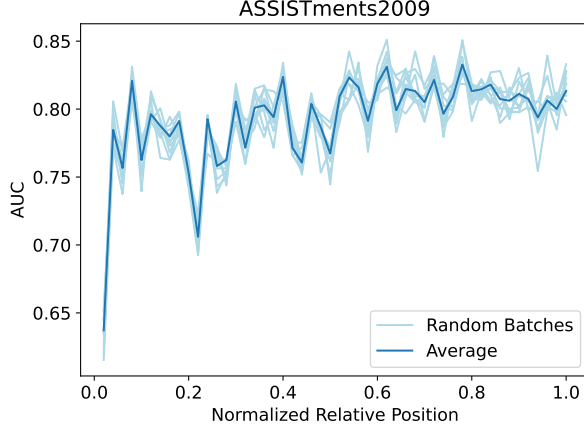
To further investigate the contribution of key components in AAKT, we conducted a series of ablation studies. Variants

of AAKT are defined which denote one or more missing components in our model. For precision, we also conducted ablation studies on filtered ASSISTments and Junyi. Results are listed in Table IV.

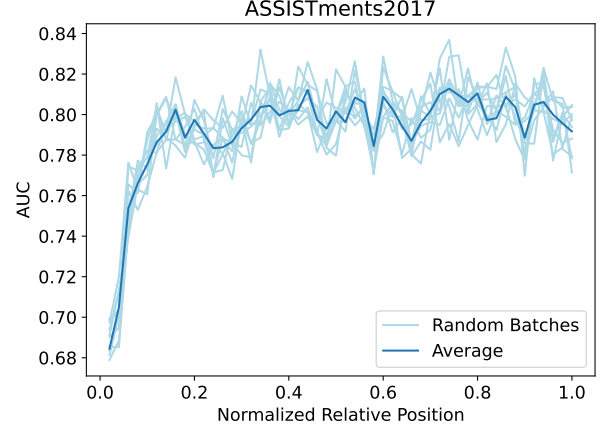
- **AAKT w/o Sliding** removes sliding window technique. That is to say, the  $r_{overlap}$  is set to 0 both in training and testing.
- **AAKT w/o Time** removes the attribute ‘total time spent’ for every question in the dataset.
- **AAKT w/o Skill** removes the information related to skills.
- **AAKT w/o Time&Skill** removes both ‘total time spent’ attributes and the information related to skills.
- **AAKT w/o Auxiliary Task** removes the auxiliary task. Specifically, it directly adds the embedding vectors of questions and skills together. For questions with more than one skill, we take the average of skill embedding vectors.

From Table IV, the following observations can be made:

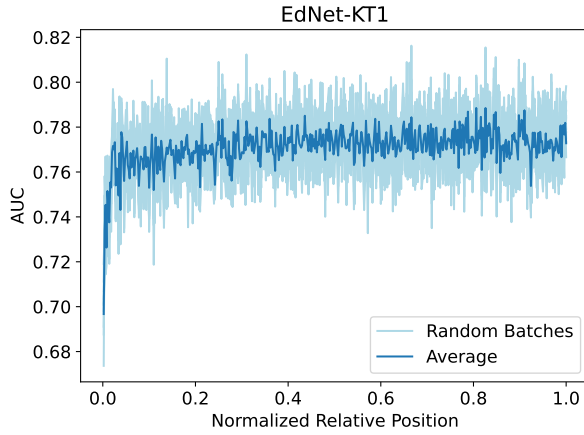
- The metrics will decrease on all four datasets if any key components (i.e., Sliding Window Technique, Additional Information, and Auxiliary Task) are removed. Therefore, the effectiveness of key components in our model is validated.
- On EdNet-KT1 and ASSISTments2009, the cost of removing additional information (i.e., total time spent on each question) is larger than removing skill embeddings. This can be partly attributed to the different weights of attributes related to every question. Not all KT datasets are homogeneous.
- The performance of AAKT w/o Time&Skill is lower than both AAKT w/o Time and AAKT w/o Skill. Therefore, it is reasonable to assume that more additional information leads to better performance.
- On all datasets, AAKT w/o Skill performs better than



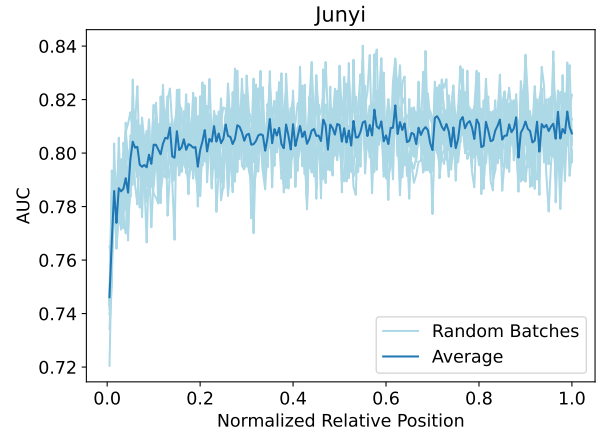
(a) AUC Distribution of Random Batches on ASSISTment2009



(b) AUC Distribution of Random Batches on ASSISTment2017



(c) AUC Distribution of Random Batches on EdNet-KT1



(d) AUC Distribution of Random Batches on Junyi

Fig. 6. AUC distribution on random batches in different datasets. The x-axis denotes the relative position of each question in the sequence. Light blue lines are results of 10 randomly chosen batches in each figure and the dark blue line is their average.

AAKT w/o Auxiliary Task, indicating the loss brought by bad embedding strategy. Merely adding to embedding vectors of questions and skills together does not function well in our model. This result further indicates that a better and more specific fusion method should be developed in knowledge tracing.

## F. Visualization

In this subsection, we conduct some visualizations to get deeper insights into some key components of AAKT.

1) *Case Study*: In this subsection, a simple case study is conducted to further validate the effectiveness of alternate sequence construction and autoregressive modeling. We select a modified subsequence from a certain sequence in EdNet-KT1 in which our model predicted the first 8 answers correctly. Moreover, the attention weights of the first causal self-attention layer in our model are extracted when it is calculating the prediction results. A heat map is plotted according to the attention weights (see Figure 7).

Our model has different preferences on history interactions when predicting the answer to the next question. More specif-

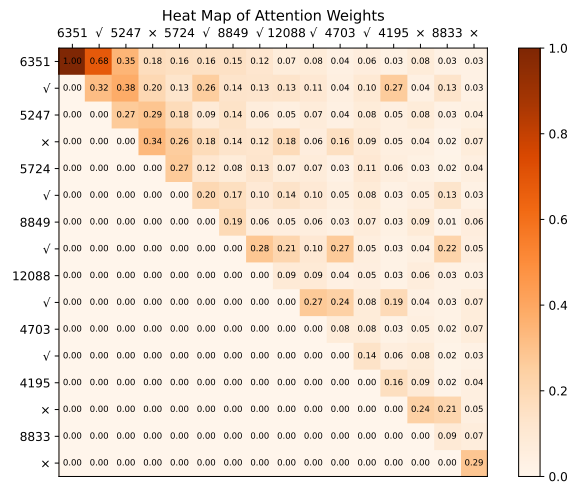


Fig. 7. Heat map of attention weights of first causal self-attention layer in our model. The X-axis and y-axis denote the same interaction subsequence from a certain sequence in EdNet-KT1 and the elements denote the specific attention weights (with softmax). Our model predicts all the answers to this subsequence correctly. Therefore, attention weights can demonstrate the preference of our model on the history of interactions when predicting.

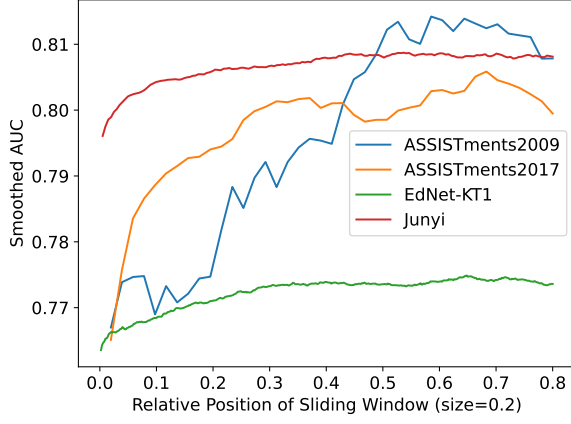


Fig. 8. Smoothed AUC results in Figure 6 using a sliding window with the length of  $0.2 \times \max\_sequence\_length$ . The x-axis denotes the normalized relative position of the starting point of the sliding window.

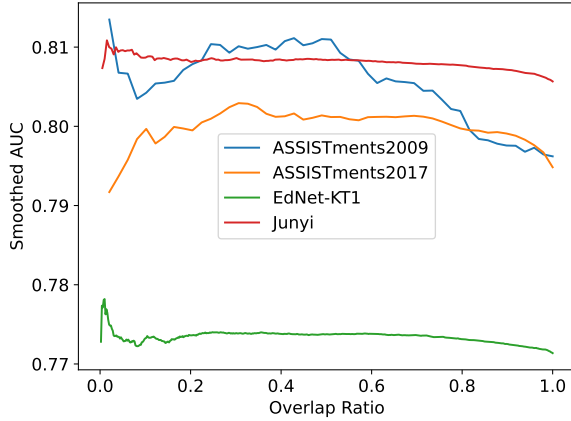


Fig. 9. The relationship between overlap ratio and the mean AUC score in the corresponding sliding window.

ically, it emphasizes the result of answering question 5247 when predicting the answer of 5724. Similar emphasis can be observed from the prediction of 12088, 4703, 4195, and 8833. Our model utilizes the information of history responses to predict the answer to the current question, successfully and efficiently combining the information of questions and responses. This phenomenon can be credited to alternative autoregressive modeling of original sequences.

2) *The Effectiveness of Sliding Window*: To validate the effectiveness of the sliding window technique, we should see raw datasets in another way. The sliding window is about the concrete sequence, not just packing all the parallel sequences and dealing with them together. Therefore, we should horizontally observe the datasets.

Firstly, showing the overall AUC distribution is necessary because sliding windows leave merely the segments at the sequences' back end for prediction. For demonstration, we set the value of  $r_{overlap}$  to 0.0 and choose 10 random batches from the testing process (the hyperparameters may not be optimal here) on every dataset and calculate AUC in terms of the position in the sequence. These batches with

shape  $(batch\_size, \max\_sequence\_length)$  are transposed to  $(\max\_sequence\_length, batch\_size)$ . Then for every position  $1 \leq i \leq batch\_size$ , there is a 1D array recording every prediction in this batch. After removing the masked and useless positions, AUC is calculated on these 1D arrays. Figure 6 shows the visualizations of this experiment (the settings of  $\max\_sequence\_length$  may be different on each dataset). The AUC of front 10% prediction results are relatively lower, then increasing quickly. Afterward, the AUC fluctuates dramatically around a fixed value, showing considerable randomness.

To make the most of Figure 6, we smoothed all these results using a sliding window with the length of  $0.2 \times \max\_sequence\_length$ . Figure 8 demonstrates the result of smoothing, from which we can observe that there are some tendencies beneath the randomness. Generally, the mean AUC scores are increasing while the sliding window moves forward on random batches chosen from EdNet-KT1, Junyi, and ASSISTments2017. For ASSISTments2009, there are more intense fluctuations but the general tendency remains.

As explained in Subsection III-C2, assuming there is a batch of data prepared for testing and calculating metrics, only the part at back end of each sequence is not masked. Therefore, as  $r_{overlap}$  decreases from 1.0, the window will grow from the last element of every sequence. Although the specific tendency is hard to distinguish from Figure 6, we can still get some information from it. Figure 9 demonstrates the relationship between  $r_{overlap}$  and the mean AUC in the corresponding sliding window. We still use the same randomly selected batches of data described above here. The line of ASSISTments2009 shows a small peak around 0.5 while others are relatively stable. Considering information shown in Figure 9, the fixed value 0.5 of  $r_{overlap}$  in our model is proven reasonable and close to optimal choice.

3) *The Effectiveness of Auxiliary Task*: To demonstrate whether the auxiliary task truly introduces the information of skills into embedding vectors of questions, we conduct an experiment to plot the overall distribution on a 2D plane using UMAP [38], an effective dimension reduction algorithm. The results are shown in Figure 10. It is obvious that the auxiliary task successfully achieves its goal by separating embedding vectors of questions by their respective skill(s) in UMAP maps.

Detailed analysis of UMAP maps on every dataset are listed as follows:

- **EdNet-KT1**: The majority of embedding vectors are correctly grouped. However, there is a cluster that contains various types of points. This can be attributed to the relatively extreme distribution of skills on the question set (see Figure 5). The distribution of EdNet-KT1 has an exceedingly long and thin tail and some skill sets just appeared once in the overall question set, causing insufficiency of training. Therefore, the UMAP algorithm doesn't know how to classify these skill sets.
- **ASSISTments2009**: Although the embedding vectors are generally grouped, the distribution inside these groups is messy. The reason for this phenomenon includes two factors. Firstly, the volume of this dataset is small compared to the other three datasets (see Table II). Secondly, the

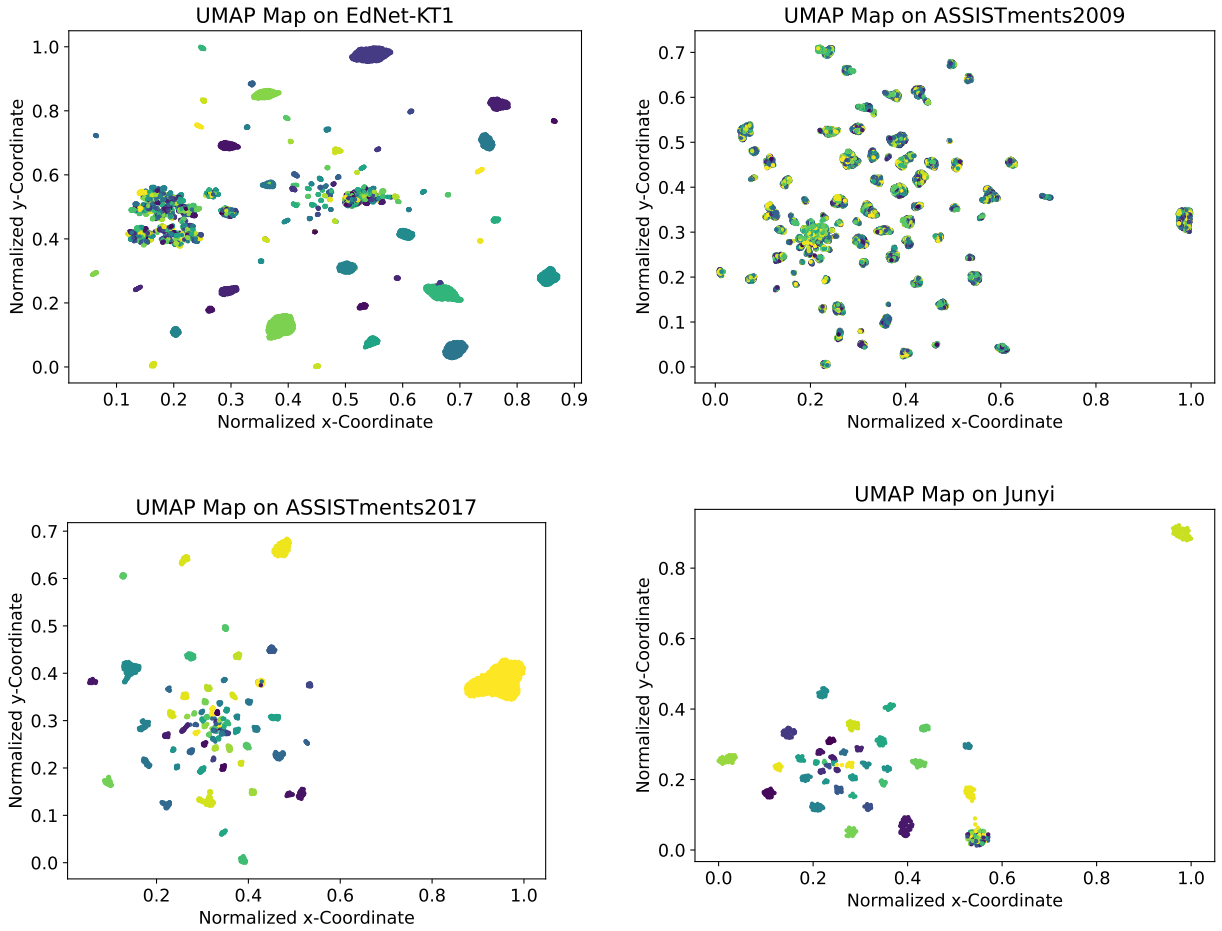


Fig. 10. UMAP maps of embedding vectors after auxiliary task. Different colors in one subfigure denote different skills or skill sets. The dimensions of embedding vectors are reduced by UMAP, thus facilitating the visualization on a 2D plane. Distance between clusters is not important, although UMAP keeps the original spatial distribution of raw data partly.

majority of answering sequences have a relatively short length. 50% of sequences are shorter than 40 steps.

- **ASSISTments2017**: All the embedding vectors are perfectly categorized.
- **Junyi**: The majority of embedding vectors are correctly categorized.

## V. CONCLUSION

In this paper, we propose AAKT, a novel knowledge tracing model based on the autoregressive transformer. In particular, we utilize GPT-J instances (without pre-trained weights) as the backbone. To feed the model with suitable sequences, we modify the input form of datasets and transform raw data into sequences resembling natural language. We also introduce the sliding window technique to make the most of limited datasets and improve the performance of prediction. To better combine the information of questions and skills, we introduce an auxiliary task to enable the information of skills to flow into the embedding vectors of questions. Moreover, additional information is utilized to further boost the ability of our model. We conducted comparative experiments with seven previous KT models on four real-world datasets and verified the effectiveness of AAKT in knowledge tracing tasks. We also

demonstrated that the key components of AAKT indeed benefit the overall performance of it. At last, visualization experiments were conducted to further explain the foundation of our model.

Possible directions of future work include 1) designing a more advanced method of combining information of questions and skills, 2) exploring more efficient autoregressive transformer structures to fit knowledge tracing tasks, and 3) utilizing limited datasets in training and testing in a more efficient way.

## REFERENCES

- [1] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Modeling and User-Adapted Interaction*, vol. 4, no. 4, pp. 253–278, 1994.
- [2] M. Yudelson, K. R. Koedinger, and G. J. Gordon, "Individualized Bayesian knowledge tracing models," in *Proceedings of the 16th International Conference on Artificial Intelligence in Education*, 2013, pp. 171–180.
- [3] R. Pelánek, "Bayesian knowledge tracing, logistic models, and beyond: An overview of learner modeling techniques," *User Modeling and User-Adapted Interaction*, vol. 27, no. 3-5, pp. 313–350, 2017.
- [4] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, 2015, pp. 505–513.

- [5] G. Abdelrahman, Q. Wang, and B. P. Nunes, "Knowledge tracing: A survey," *ACM Computing Surveys*, vol. 55, no. 11, pp. 224:1–224:37, 2023.
- [6] X. Song, J. Li, T. Cai, S. Yang, T. Yang, and C. Liu, "A survey on deep learning based knowledge tracing," *Knowledge-Based Systems*, vol. 258, p. 110036, 2022.
- [7] C. Yeung and D. Yeung, "Addressing two problems in deep knowledge tracing via prediction-consistent regularization," in *Proceedings of the 5th Annual ACM Conference on Learning at Scale*, 2018, pp. 5:1–5:10.
- [8] J. Zhang, X. Shi, I. King, and D. Yeung, "Dynamic key-value memory networks for knowledge tracing," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 765–774.
- [9] G. Abdelrahman and Q. Wang, "Knowledge tracing with sequential key-value memory networks," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 175–184.
- [10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, 2017, pp. 5998–6008.
- [11] S. Pandey and G. Karypis, "A self attentive model for knowledge tracing," in *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.
- [12] A. Ghosh, N. T. Heffernan, and A. S. Lan, "Context-aware attentive knowledge tracing," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 2330–2339.
- [13] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi, "SAINT+: integrating temporal features for ednet correctness prediction," in *Proceedings of the 11th International Learning Analytics and Knowledge Conference*, 2021, pp. 490–496.
- [14] S. Pu, M. Yudelson, L. Ou, and Y. Huang, "Deep knowledge tracing with transformers," in *Proceedings of the 21st International Conference on Artificial Intelligence in Education*, 2020, pp. 252–256.
- [15] X. Guo, Z. Huang, J. Gao, M. Shang, M. Shu, and J. Sun, "Enhancing knowledge tracing via adversarial training," in *Proceedings of the 29th ACM Multimedia Conference*, 2021, pp. 367–375.
- [16] R. S. J. de Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," in *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, 2008, pp. 406–415.
- [17] Z. A. Pardos and N. T. Heffernan, "KT-IDEM: Introducing item difficulty to the knowledge tracing model," in *Proceedings of the 19th International Conference on User Modeling, Adaption and Personalization*, 2011, pp. 243–254.
- [18] M. Khajah, R. V. Lindsey, and M. Mozer, "How deep is knowledge tracing?" in *Proceedings of the 9th International Conference on Educational Data Mining*, 2016.
- [19] S. Shen, Z. Huang, Q. Liu, Y. Su, S. Wang, and E. Chen, "Assessing student's dynamic knowledge state by exploring the question difficulty effect," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 427–437.
- [20] J. Vie and H. Kashima, "Knowledge tracing machines: Factorization machines for knowledge tracing," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, the 31st Innovative Applications of Artificial Intelligence Conference, the 9th AAAI Symposium on Educational Advances in Artificial Intelligence*, 2019, pp. 750–757.
- [21] S. Rendle, "Factorization machines," in *Proceedings of the 10th IEEE International Conference on Data Mining*, 2010, pp. 995–1000.
- [22] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, 2015, pp. 802–810.
- [23] F. Wang, Z. Huang, Q. Liu, E. Chen, Y. Yin, J. Ma, and S. Wang, "Dynamic cognitive diagnosis: An educational priors-enhanced deep knowledge tracing perspective," *IEEE Transactions on Learning Technologies*, vol. 16, no. 3, pp. 306–323, 2023.
- [24] J. Cui, Z. Chen, A. Zhou, J. Wang, and W. Zhang, "Fine-grained interaction modeling with multi-relational transformer for knowledge tracing," *ACM Transactions on Information Systems*, vol. 41, no. 4, pp. 104:1–104:26, 2023.
- [25] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," <https://www.mikecaptain.com/resources/pdf/GPT-1.pdf>, 2018.
- [26] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
- [27] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, 2020.
- [28] J. Lee and D. Yeung, "Knowledge query network for knowledge tracing: How knowledge interacts with skills," in *Proceedings of the 9th International Conference on Learning Analytics and Knowledge*, 2019, pp. 491–500.
- [29] S. Sonkar, A. S. Lan, A. E. Waters, P. Grimaldi, and R. G. Baraniuk, "qDKT: Question-centric deep knowledge tracing," in *Proceedings of the 13th International Conference on Educational Data Mining*, 2020.
- [30] W. Wang, H. Ma, Y. Zhao, F. Yang, and L. Chang, "SEEP: Semantic-enhanced question embeddings pre-training for improving knowledge tracing," *Information Sciences*, vol. 614, pp. 153–169, 2022.
- [31] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim, and J. Heo, "EdNet: A large-scale hierarchical dataset in education," in *Proceedings of the 21st International Conference on Artificial Intelligence in Education*, 2020, pp. 69–73.
- [32] H. Ebbinghaus, "Memory: A contribution to experimental psychology," *Annals of Neurosciences*, vol. 20, no. 4, p. 155, 2013.
- [33] C. Pojen, H. Mingen, and T. Tzuyang, "Junyi academy online learning activity dataset: A large-scale public online learning activity dataset from elementary to senior high school students." 2020.
- [34] B. Wang, "Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX," <https://github.com/kingoflolz/mesh-transformer-jax>, May 2021.
- [35] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu, "RoFormer: Enhanced transformer with rotary position embedding," *CoRR*, vol. abs/2104.09864, 2021.
- [36] Z. Liu, Q. Liu, J. Chen, S. Huang, J. Tang, and W. Luo, "pyKT: A python library to benchmark deep learning based knowledge tracing models," in *Proceedings of the 36th Annual Conference on Neural Information Processing Systems*, 2022.
- [37] Q. Liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu, "EKT: Exercise-aware knowledge tracing for student performance prediction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 1, pp. 100–115, 2021.
- [38] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.



**Hao Zhou** is a BSc student in the School of Computer Science and Engineering, Beihang University. His research interests include natural language processing and machine learning.



**Wenge Rong** is professor in School of Computer Science and Engineering, Beihang University, China. He received his PhD from University of Reading, UK, in 2010; MSc from Queen Mary College, University of London, UK, in 2003; and BSc from Nanjing University of Science and Technology, China, in 1996. He has many years of working experience as a senior software engineer in numerous research projects and commercial software products. His area of research covers machine learning, natural language processing, and information management.





**Jianfei Zhang** received his MSc from School of Software, Beijing University of Technology, China, in 2020; BSc from College of Software, Taiyuan University of Technology, China, in 2017. He is currently pursuing his PhD in School of Computer Science and Engineering at Beihang University. His research interests include natural language processing and machine learning.



**Qing Sun** is associate professor in School of Computer Science and Engineering, Beihang University. She received her Ph.D. from Department of Information System of Beihang University in 2017; MSc and BSc from School of Information Science of Beijing Normal University in 2008 and 2005, respectively. Her research interests include smart education, data mining and knowledge engineering.



**Yuanxin Ouyang** received her BSc and PhD degrees from Beihang University, in 1997 and 2005, respectively. She is a professor with Beihang University, China. Her area of research covers recommender system, data mining, and social networks.



**Zhang Xiong** is professor in School of Information Technology & Management, University of International Business and Economics, and director of the Advanced Computer Application Research Engineering Center of National Educational Ministry of China. He has published over 250 referred papers in international journals and conference proceedings and won a National Science and Technology Progress Award. His research interests span from smart cities, knowledge management, and information systems.