Journal of Computers Vol. 36 No. 6, December 2025, pp. 61-71
https://doi.org/10.63367/19911599202512360600 5

Application of Self-attention with Relative Position Representation in Math Teaching Knowledge Tracking

Qi Cheng[1*] and Ting Zhu[2]

[1] School of Mathematics and Physics, Fu Zhou Preschool Education College,
Fuzhou 330201, Jiangxi Province, China
15179431677@163.com

[2] School of Foreign Languages, East China of University of Technology,
Nanchang 330001, Jiangxi Province, China
19917930003@163.com

Accepted 8 August 2024; Revised 9 August 2024; Accepted 10 August 2024

Abstract. The evolution of Intelligent Tutoring Systems (ITS) has profoundly shaped the landscape of students' engagement with and mastery of mathematical logic, thereby enhancing their overall learning efficacy. Integral to these systems is the imperative of accurately modeling learners' knowledge proficiency as a precursor to dispensing targeted feedback and guidance, rendering algorithms such as Knowledge Tracing pivotal in this context. In this study, we introduce the improved Self-Attention with Relative Position Representation Knowledge Tracing (SARPR-KT) framework. The proposed SARPR-KT model demonstrates superior performance, exhibiting a greater increase in Area Under the Curve (AUC), Recall value, compared to alternative models, as evidenced by experimentation on ASSISTments2009 data.

Keywords: intelligent tutoring systems, knowledge tracing, self-attention

1 Introduction

The Intelligent Tutoring System (ITS) stands as a computational paradigm designed for educational support, imparting timely feedback to learners with the aim of enhancing their learning efficiency [1]. In fulfilling this role, ITS necessitates a nuanced understanding of learners' proficiency in knowledge concepts and the interrelationships among diverse knowledge domains [2]. Consequently, the advent of Knowledge Tracing (KT) algorithms [3], integrated across various models, has been instrumental in this pursuit. Over the past decades, the evolution of ITS has been substantial, with contemporary applications assuming pivotal roles in the realm of online education [4]. Hence, a critical challenge resides in the enhancement of model accuracy within this context. Given a student's reply to different kinds of mathematical exercises R_1, R_2, …, R_{n-1}, the primary KT assignment involves predicting the probability, that the student will provide the correct answer to the next exercise R_n. From this context, the interactions between a student and tons of mathematical exercises are encapsulated in an independent sequence S, where each element is a tuple $S_i = (R_i, P_i)$. Here, R_i represents the exercise content for the ith exercise, and P_i represents users' response to exercise R_i. Within P_i, $p_i \in \{0,1\}$ indicates whether the student successfully completed exercise R_i. In essence, this paper aims to predict the probability:

$$Pr\left(p_i = 1 \middle| S_1, S_2, \ldots, S_{n-1}, R_n\right) \tag{1}$$

The Programming Teaching Assistant (PTA) serves as an instructional platform widely utilized by students across numerous universities in China, providing a conducive environment for programming practice. Students engage in autonomous programming exercises, while educators leverage the PTA platform to assign programming tasks, thereby enhancing students' programming proficiency. This utilization generates substantial datasets encapsulating students' interactive learning experiences. Leveraging these datasets for KT exposes the modeling

* Corresponding Author

of users' mastery levels concerning knowledge concepts, subsequently facilitating the implementation of an ITS on the PTA platform [5].

Diverging from conventional testing platforms, PTA uniquely permits users to iteratively provide the identical exercise till they achieve correctness, fostering a substantial increase in data interaction. College instructors commonly deploy PTA for ongoing assessments during the teaching process, wherein students progressively enhance their knowledge mastery through exercise completion. This dynamic leads to user interactions within PTA data exhibiting a distinctive "wrong before right" pattern on specific exercises. These idiosyncrasies pose challenges for conventional models such as Deep Knowledge Tracing (DKT) [6] or Self-attention Knowledge Tracing (SAKT) [7], underscoring the unique nature of programming problems compared to other domains. Acknowledging these disparities as valuable and necessary, we propose a novel method inspired by Deep Self-attention Knowledge Tracing (DSAKT) [8]. This approach extends the self-attention mechanism to adeptly incorporate representations of relative positions or distances between sequence elements. Consequently, when applied to PTA data, it generates superior results. However, Neural networks have revolutionized various fields of artificial intelligence, yet their inherent lack of interpretability poses challenges in understanding model decisions and behaviors, including feature visualization, attention mechanisms, saliency maps, layer-wise relevance propagation, and concept activation vectors.

In this paper we solved the problems of large amount of training calculations and large memory consumption and proposed a more optimized model named Self-Attention with Relative Position Representation Knowledge Tracing (SARPR-KT) to identify the excavation and connection of different knowledge concepts by samples, quickly and automatically. In addition, we highlight the significance of interpretability in fostering trust, accountability, and adoption of neural network models in critical applications by adding advanced Capsule modules. In the end, the proposed SARPR-KT method gains the higher AUC, Recall value of ASSISTments2009 dataset.

2 Related Works

In China, a substantial cohort of candidates annually engages in diverse examinations, notably those pertaining to the entry qualifications of junior high schools and high schools. Against the backdrop of the transitional phase induced by the epidemic, the pervasive integration of online classes and e-learning tools, intelligent tutoring system has facilitated a notable augmentation in users enhancing their proficiency in academic subjects. Fig. 1 shows the architecture of intelligent tutoring system. The discernible facilitation in the establishment of a mathematical knowledge framework germane to diverse examinations is evident. Consequently, an increasing number of educational institutions are allocating considerable financial and human resources towards the investigation of knowledge system construction and the development of effective online assistance mechanisms for users.

The methodology of knowledge tracing, derived from educational data mining, has exhibited significant efficacy within the domain of mathematics education. This approach entails the ongoing evaluation of students' mastery levels and learning progression by analyzing their responses to diverse educational tasks. Within the landscape of mathematics education, knowledge tracing has substantiated its effectiveness in discerning individual students' strengths and weaknesses. Consequently, it facilitates the implementation of personalized and targeted interventions designed to address specific educational needs. This application underscores the adaptability and utility of knowledge tracing as a valuable tool in optimizing educational outcomes within the field of mathematics. In recent years, the application of neural networks in the field of Knowledge Tracing are springing into educational field, reshaping the landscape of adaptive learning and educational technology. Knowledge Tracing involves the modeling of students' learning processes to predict their future performance and tailor instructional content accordingly. Neural networks, with their ability to capture complex patterns and relationships in data, have emerged as powerful tools to enhance the accuracy and efficiency of Knowledge Tracing models, Fig. 2 shows the basic concept of performance prediction in mathematical KT-question, where T_k means each exercise.

The renowned Bayesian Knowledge Tracing algorithm and collaborative filtering have been extensively researched over time, gradually yielding ground to methodologies employing neural networks. Although there are numerous studies focusing on traditional machine learning models such as support vector machines, random forests, or logistic regression, this paper exclusively discusses deep learning and improved models without further literature search. Deep Knowledge Tracing (DKT) stands out as a pivotal algorithm, originating from researchers at Stanford University and experiencing rapid development over the past several years [9-11]. An impactful advancement in this domain involves the incorporation of recurrent neural networks (RNNs) into Knowledge Tracing. RNNs, tailored for sequential data, adeptly model the temporal dependencies intrinsic to students'

learning trajectories. By capturing the sequential nature of student interactions with educational content, RNN-based Knowledge Tracing models offer nuanced insights into knowledge acquisition dynamics [12], facilitating improved predictions and personalized interventions. Despite the utilization of Memory Augmented Neural Network in Dynamic Key-value memory network (DKVMN) for Knowledge Tracing to enhance interpretability [13], it encounters similar challenges as DKT. DKT has emerged as a significant and mainstream knowledge tracing method, undergoing continuous refinement [14]. Recent progress, building upon the original DKT model and its variants, has yielded valuable achievements [15]. Moreover, Self-Attentive Knowledge Tracing (SAKT) introduces attention mechanisms, enabling the model to concentrate on specific segments of the input sequence, prioritizing relevant information while disregarding noise. This attention-driven approach enhances model interpretability, elucidating the specific steps and concepts contributing to a student's learning progress—critical for educators and educational technologists seeking actionable insights to guide pedagogical decisions. Given the unique characteristics of the ASSIST dataset, we replicated the SAKT method and innovatively enhanced its positioning plan. Subsequently, we implemented a novel algorithm grounded in Self-Attention with Relative Position Representation. This creative adaptation aims to address the distinctive aspects of the ASSIST dataset, further contributing to the evolution and refinement of knowledge tracing methodologies.

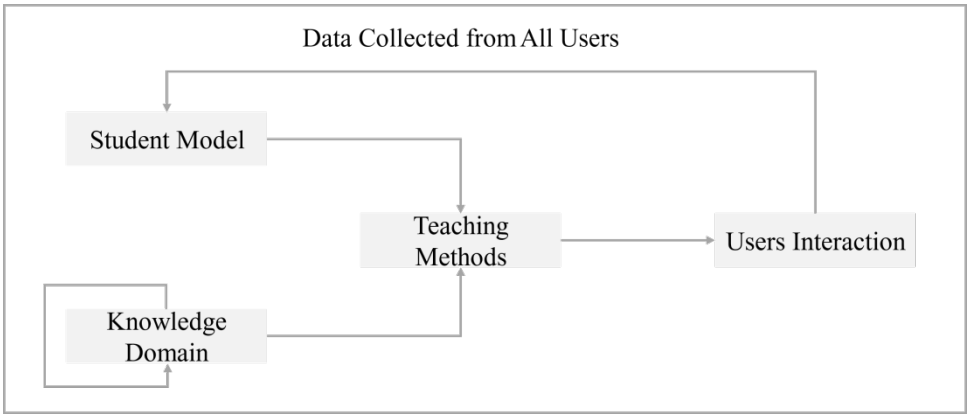

Fig. 1. Basic architecture of Intelligent Tutoring System (ITS)

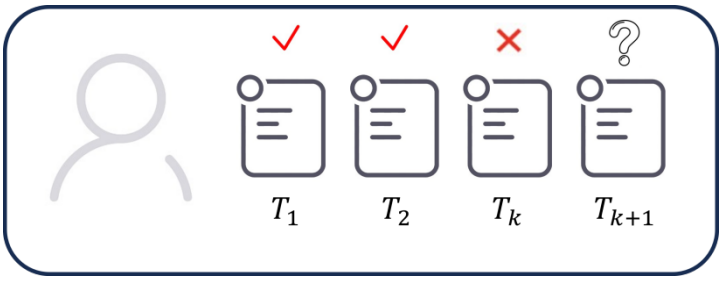

Fig. 2. User mathematic performance prediction

3 Proposed Model

3.1 Input Representation

Prior to constructing mathematical models, data preprocessing is essential, encompassing data cleansing, format conversion, feature engineering, data splitting, data balancing, and data normalization operations. The preprocessing procedures in this segment align with those employed in analogous studies within the field.

Proposed SARPR-KT model processes a sequence of interactions $S_k = (R_k, P_k)$ with the length of k, taking the next exercise T_k as input, and predicts the probability that the user will correctly solve T_{k+1}. To streamline subsequent data processing, we represent the interaction using the following equation. Here, δ signifies the response bias, typically the total number of exercises, $\dot{T}$ depicts the set of all exercise. Therefore, the final model is:

$$\begin{cases} S_k = p_k * \delta + T_k \\ S = [S_1, S_2, \ldots, S_k] \\ \dot{T} = [T_2, T_3, \ldots, T_{k+1}] \end{cases} \tag{2}$$

3.2 Embedding Layer

Given the embedding layer in SARPR-KT, which comprises three embedding matrices, $\dot{T} \in R^{\delta * d}$, $\dot{P} \in R^{k * d}$, $\dot{S} \in R^{2\delta * d}$ where δ denotes the gross amounts of exercises, the parameter k represents the length of the sequence, while d depicts the latent dimension, Fig. 3 shows complete flow on KT prediction. And the reasons why we prefer embedding layer are as follows:

Firstly, embeddings can provide a dense representation of high-dimensional discrete features such as words or categories, reducing dimensionality and capturing semantic relationships more effectively. As for ASSIST dataset, it can reduce the dimensional issues.

Secondly, embeddings capture semantic information by mapping similar features to nearby vectors in the embedding space. This enables models to better understand the underlying semantics of the data. It reduces computational complexity by decreasing the number of parameters in the model, particularly beneficial for large-scale datasets.

Thirdly, embeddings generalize well to unseen data, as they learn continuous representations that encode rich structural and pattern information from the data, enables end-to-end learning, allowing models to learn rich feature representations flexibly and comprehensively from data, leading to improved performance and interpretability.

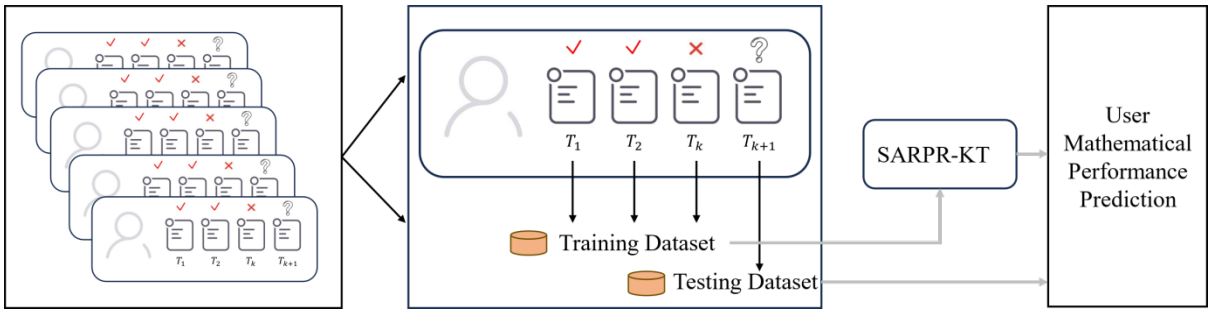

Fig. 3. User mathematic performance prediction flow

1) Position Encoding. Previously mentioned $\dot{P} \in R^{k * d}$ corresponds to a static and unchanging coordinate embedding. This embedding is acquired through the computation of the sine position embedding vector:

$$\dot{P}_{i,j} = \begin{cases} \sin\left(i / 10000^{j/d}\right) j \bmod 2 = 0 \\ \cos\left(i / 10000^{(j-1)/d}\right) j \bmod 2 = 1 \end{cases} \tag{3}$$

For $1 \le i \le k$, where i denotes the position, and $1 \le j \le d$, representing the dimension, each dimension of the position embedd1ing in the aforementioned equation corresponds to a sine curve [16].

The matrices $\dot{P}$ and $\dot{S}$ serve to transform the acquired input sequence S into a vector within the latent space. This transformation involves embedding the interaction and position separately, followed by the addition of the resulting embedding vectors. To embed T_{k+1} into a vector within the latent space of the same shape, information from the last $(k-1)$ exercises in the interaction is utilized. This information is then concatenated with T_{k+1} and subjected to transformation through $\dot{T}$ and $\dot{P}$.

2) Projection. Following the embedding process, the projection matrices $Z^S \in R^{d*d}$, $Z^T \in R^{d*d}$ signify the linear projection of the respective vectors into distinct spaces. The output of embedding layer comprises embedded vectors $\hat{S} \in R^{k*d}$, $\hat{T} \in R^{k*d}$. Among this operation, the projection layer serves to enhance feature space by incorporating additional feature transformations or attention mechanisms, allowing the model to better focus on and utilize crucial information within the input data, thus improving model performance and robustness. In tasks involving multi-modal data processing, the projection layer can be employed for aligning and integrating features from different modalities, thereby enhancing the model's ability to handle multi-modal data. In the model described in the paper, its greatest advantage lies in the re-representation of feature space, which aids the model in better comprehending and leveraging the structure and patterns inherent in the data.

3.3 Deep Self-attentive Encoder-Decoder

Our proposed SARPR-KT model, built upon the deep multi-head self-attention architecture, features an enhanced encoder-decoder mechanism. The encoder accepts the input of the embedded vector $\hat{S}$ and provides the processed output $\hat{O}$ to the decoder. Subsequently, the decoder takes $\hat{O}$ and another sequential input, $\hat{T}$ representing the exercise embedding vector, producing the predicted response $p_i = [p_1, p_2, \ldots, p_k]$.

1) Multi-head Attention. The cornerstone of SARPR-KT lies in its multi-head self-attention mechanism. Self-attention sublayers can sublayers can utilize individually selected attention heads. The sublayer output is obtained by concatenating the outcomes from each head and applying a parameterized linear transformation. The Scaled Dot-Product describes the attention as:

$$Attention(Q,K,V) = Soft\max\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{4}$$

where Q, K, V denotes the queries, keys, and values respectively.

In the KT framework, our model should take into account only the initial interactions when predicting the outcome of $(k+1)$th exercise. Each attention head generate a new sequence with same length of input sequence.

Each output element O is denoted as summary of weights:

$$O_i = \sum_{j=1}^{k} \alpha_{ij}\left(x_j W^V\right), x_j \in \mathrm{InputSeq}\left[x_1, x_2, \ldots x_n\right] \tag{5}$$

Where α_{ij} depicts weight coefficient, is calculated as a soft function. And δ_{ij} adopts a compatibility function that compares two input elements:

$$\alpha_{ij} = \frac{\exp \delta_{ij}}{\Sigma_{k=1}^{n} \exp \delta_{ik}}, \delta_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right)^T}{\sqrt{d_z}} \tag{6}$$

The compatibility function was selected as the scaled dot product, facilitating efficient computation. Linear transformations of the inputs enhance expressive power adequately.

2) Relation-aware Self-Attention. We modify O_i to disseminate edge information to the sublayer output:

$$O_i = \sum_{j=1}^{k} \alpha_{ij} (x_j W^V + \sigma_{ij}{}^V) \tag{7}$$

This extension is likely crucial for tasks in which information regarding the selected edge types by a particular attention head proves valuable to downstream encoder or decoder layers. In addition,

$$\delta_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K + \sigma_{ij}{}^K\right)^T}{\sqrt{d_z}} \tag{8}$$

The principal rationale for employing a straightforward addition to integrate edge representations is to facilitate an efficient implementation. When accounting for relative positions, the representations vary for different pairs of positions. It prevents the need to compute all δ_{ij} for every pair of positions through a single matrix multiplication:

$$\delta_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right) + x_i W^Q \left(\sigma_{ij}{}^K\right)^T}{\sqrt{d_z}} \tag{9}$$

3) Feed-Forward Networks. The output vector derived from the multi-head layer remains a linear combination of its inputs. Consequently, feed-forward networks, comprising two linear transformations with a sigmoid activation, are employed to introduce non-linearity to the model.

$$FFN(x_i) = MAX\left(0, x_i W_1 + b_1\right) W_1 + b_2 \tag{10}$$

3.4 Residual Connection & Layer Normalization & Capsule Neural Network

The proposed residual model is employed to propagate lower-level features to higher levels, thereby facilitating the model's ability to capture low-level relations. Normalization is applied in our architecture, which stabilizes and accelerates the neural architecture [17]. Following the advanced research on artificial intelligent. Capsule networks (CapsNets) have emerged as a promising alternative to traditional convolutional neural networks, including Convolutional Neural Networks (CNNs), Multilayer Perceptions, offering several inherent advantages in modeling complex data structures. CapsNets offer enhanced interpretability compared to CNNs, thanks to their explicit instantiation parameters. Each capsule in a CapsNet architecture not only activates based on learned features but also encapsulates instantiation parameters such as pose and deformation. This enables CapsNets to provide interpretable representations of input data, making them invaluable for applications requiring transparency and explainability, such as medical diagnosis and autonomous systems. It exhibits greater resilience to adversarial attacks compared to CNNs, primarily due to their ability to encode spatial relationships and detect anomalies in input data. Adversarial attacks often exploit vulnerabilities in deep learning models by introducing imperceptible perturbations to input samples. Inherent capacity to capture spatial hierarchies makes them less susceptible to such attacks, enhancing their reliability and security in critical applications such as cybersecurity and fraud detection. The capsule module comprises neurons designed to represent different relations, capturing the intricate relationships between various parts of an entity during the dynamic routing of the capsule [18]. We apply those three structures after each sub-layer. Thus, the output of each sub-layer is actually:

$$\text{LayerNorm}\left(x_i + \text{Sublayer}\left(x_i\right) + CapsuleNN\left(x_i\right)\right) \tag{11}$$

1) Encoder. The encoder comprises a multi-head attention layer with an upper triangular mask, followed by a feedforward network. Formally, the encoder can be expressed as follows:

$$\begin{cases} y_i = \text{LayerNorm}\left(x_i + \text{Multihead}\left(x_i, x_i, x_i\right)\right) \\ z_i = \text{LayerNorm}\left(y_i + \text{FFN}\left(y_i\right)\right) \\ \text{Encoder}\left(x_i\right) = z_i \end{cases} \tag{12}$$

2) Decoder. The decoder is similar to the encoder in that it contains a multi-head attention layer and a feed-forward network. Different from original transformer model, the attention layer in our SARPR-KT model will be calculated twice, sharing equal weight. For the second iteration, the keys and values correspond to the vectors obtained from the encoder. The computation of the decoder layer can be summarized as follows:

$$\begin{cases} y_i = \text{LayerNorm}\left(x_i + \text{Multihead}\left(x_i, x_i, x_i\right)\right) \\ z_i = \text{LayerNorm}\left(y_i + \text{Multihead}\left(y_i, x_{i+1}, x_{i+1}\right)\right) \\ w_i = \text{LayerNorm}\left(z_i + \text{FFN}\left(z_i\right)\right) \\ \text{Decoder}\left(x_i, x_{i+1}\right) = w_i \end{cases} \tag{13}$$

3) Prediction Layer. Lastly, each row of the vector $\tilde{\varphi}$ obtained above undergoes a linear transformation layer with sigmoid activation to derive the predicted response value for the exercise.

$$\tilde{\varphi} = \text{Sigmoid}\left(\tilde{\varphi}W + b\right) \tag{14}$$

3.5 Network Training

The objective of the training is to minimize the negative logarithm likelihood of the observed user response sequence within the model. The core idea of the cross-entropy loss function lies in comparing the probability distributions of the actual labels with those predicted by the model and computing their cross-entropy. This process is closely related to the probability distribution of events. Moreover, this classification method is extensively employed in similar studies due to its greater persuasiveness and uniformity. The parameters are acquired through minimizing the binary cross-entropy loss between $\tilde{\varphi}$ and φ, which denotes as follows:

$$Loss = -\sum_{i}^{k} \varphi_i \log\left(\tilde{\varphi}_i\right) + \left(1 - \varphi_i\right) \log\left(1 - \tilde{\varphi}_i\right) \tag{15}$$

4 Experiments

4.1 Dataset

The educational data mining field has been actively engaged in constructing student models to accommodate student data and forecast academic performance over an extended period. We trained the SARPR-KT model based on the following dataset: ASSISTments2009. Numerous investigations have utilized ASSISTment2009 dataset to prognosticate student achievements. The comprehensive dataset is bifurcated into two distinct files: one comprises all skill builder data, while the other encompasses non-skill builder data. Skill builder data, interchangeably known as mastery learning data, originates from problem sets emphasizing skill acquisition. In this context, a student attains mastery of a skill upon satisfying specific criteria, typically defined as correctly answering three consecutive questions, after which no further questions are presented. We designed a muti-experiment comparing

to other Knowledge Tracing methods. ASSISTment 2009 exhibits sparsity, with a density of 0.005, as illustrated in Table 1.

Table 1. Dataset information

Dataset	User	Skill	Interaction (K)	Density
ASSISTments2009	4219	26688	525	0.005

1) Training Details and Evaluation Methodology. We compare our model with the well-known KT models, DKT with RNN, LSTM and SAKT. We implemented SARPR-KT with PyTorch and the Adam optimizer. SARPR-KT has trained with 80% data and tested on the remaining data. The parameter setting of SARPR-KT is consistent with that of DSAKT and SAKT. We trained comparative test under the same condition with a NVIDIA RTX 2070 MaxQ GPU. Batch size: 64, step length: 50, questions in our datasets are 124, hidden layers are 128, learning rate sets 0.001, training epochs set 10.

2) Metrics: The prediction task is considered in a binary classification setting i.e., answering an exercise correctly or not. Hence, Under the dataset we measured area under the curve (AUC), *F*1 *score*, *Recall*. *TP* denotes True Positive, *FN* denotes False Negative, *TN* represents True Negative, *FP* depicts False Positive.

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{18}$$

Few papers choose Precision as one of the evaluation metrics, so we only provide the detailed calculation process, no longer provide a separate Precision comparison.

3) Model: The KT-RNN model represents a pioneering approach in employing Recurrent Neural Networks (RNNs) for the resolution of Knowledge Tracing. We apply two different types of RNNs: a vanilla RNN model with sigmoid units, Fig. 4 shows the basic flow of RNN, and a LSTM model, which is depicted in Fig. 5, address the challenge of forecasting users responses to exercises grounded in their interactions. The interrelationship of variables is delineated through a straightforward network as defined by the equations:

$$h_t = \tanh\left(W_{hx}x_t + W_{hh}h_{t-1}\right) + b_h \tag{19}$$

$$y_t = \sigma\left(W_{yh}h_t + b_y\right) \tag{20}$$

As shown in equation, both tanh and the sigmoid function $\sigma(\cdot)$ are used in elementwise operations. The proposed model is parameterized by an input weight matrix W_{hx} firstly, recurrent weight matrix W_{hh}, initial state h_0, and readout weight matrix W_{yh}. Biases for latent and read out units are given by b_h and b_y. LSTMs models are based on traditional RNNs, which has been proved more effective on NLP and time-serials problems. LSTMs latent units preserve their values until explicitly cleared by a forget-gate operation. Those process commonly hold with valid feed-backs for many time-steps, which has been proved easier to train in actual training and checking. In addition, those hidden units are constantly updated by applying multiplicative interactions, so as to carry out more complex transformations for the same number of latent units. The update equations for LSTMs are significantly more complicated than for an RNN.

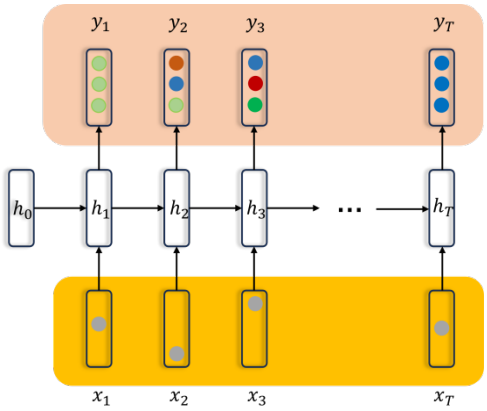

Fig. 4. Basic flow of KT-RNN

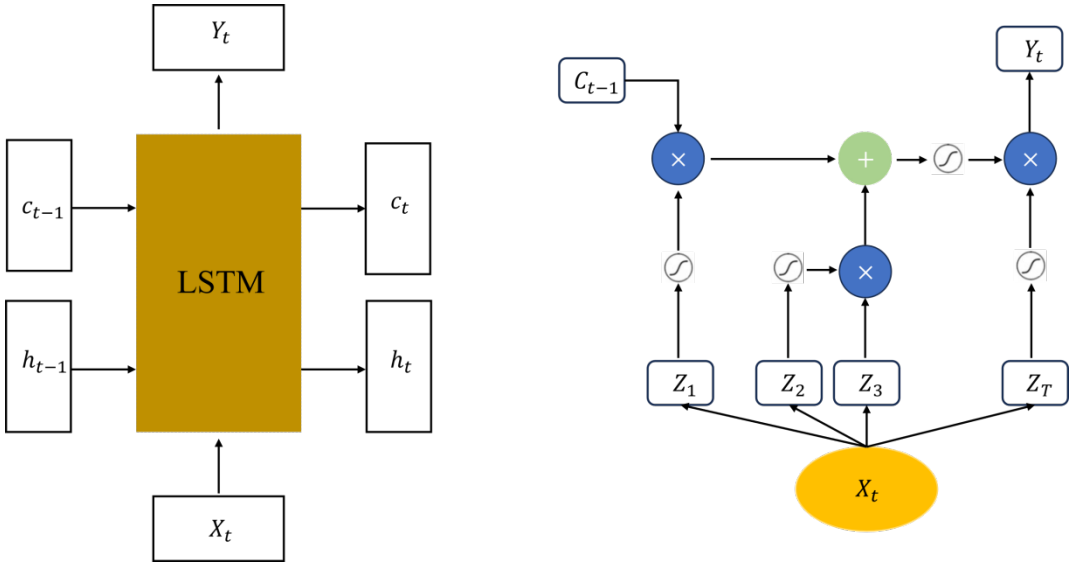

Fig. 5. Basic flow of KT-RNN

Table 2. User performance comparative prediction (AUC)

Dataset	KT-RNN	KT-LSTM	SAKT	SARPR-KT	Fluctuation
ASSISments2009	0.845	0.849	0.851	0.856	0.52%

4) Results: Student Performance Prediction: Table 2 depicts the comparative performance of SARPR-KT with the state-of-the-art models. On the ASSISTments2009 dataset, SARPR-KT extends greater than the competing methods, accomplishing an AUC of 0.856 compared to 0.893 by SAKT, Table 3 shows that *F*1-*score* does not achieve an obvious improvement, as showed in Table 4, our proposed model performs a strong improvement of 8% over the current best knowledge tracing model, which means it can assist students users to draw more attention in same knowledge field, less possibility in further calculation. In the final column of the experimental data, we augmented the content with fluctuations, representing the averaged values of 10 data points, also comprising the top 10 rankings of optimal data values. Unlike other studies that solely relied on the AUC as an evaluation metric, we also provided data for several other evaluation metrics, facilitating data references for fellow researchers to reproduce the experimental results outlined in the paper.

Table 3. User performance comparative prediction (F1-score)

Dataset	KT-RNN	KT-LSTM	SAKT	SARPR-KT	Fluctuation
ASSISments2009	0.855	0.857	0.893	0.893	0.12%

Table 4. User performance comparative prediction (Recall)

Dataset	KT-RNN	KT-LSTM	SAKT	SARPR-KT	Fluctuation
ASSISments2009	0.906	0.915	0.924	0.932	5.22%

4.2 Experiment Analysis

Assessing student users' learning abilities dynamically involves categorizing them based on their prior performance records at the conclusion of each time segment, prior to the commencement of the subsequent segment.

1) Definition of Time Segment: A time segment denotes a discrete section encompassing numerous endeavors made by a student to respond to queries within the educational framework. Herein, a time tick signifies the initial attempt by a student to address a question or exercise.

2) Segmentation of Student Response Sequences: Segmenting each student's response sequence into distinct time intervals serves twofold objectives: Firstly, it alleviates computational strain and optimizes memory allocation during learning sessions marked by extended sequences. Secondly, it facilitates the reassessment of a student's learning proficiency following each time segment, dynamically allocating them to the appropriate group for forthcoming segments.

The ASSISTment2009 dataset aggregates extensive behavioral data from students engaged in mathematical problem-solving activities, encompassing response patterns, solution times, and correctness rates. These data constitute valuable resources for analyzing students' learning capabilities and their temporal evolution. Examining students' behavior patterns during problem-solving tasks, such as response times and correctness rates, provides insights into their proficiency levels. This analysis offers deeper insights into students' learning behaviors, thereby informing personalized learning strategies. In addition, Employing clustering techniques on student cohort data allows for the delineation of distinct groups, each representing unique learning patterns or proficiency levels. This analytical approach facilitates the identification of learning characteristics within student cohorts across various time intervals, guiding the formulation of tailored instructional strategies.

5 Conclusion

This study introduces a new Knowledge Tracing model called Deep Self-Attention with Relative Position Representation Knowledge Tracing (SARPR-KT). The model demonstrated strong performance on PTA data, supporting the development of a tutoring system for students practicing mathematics on PTA and providing valuable guidance. In the field of education data mining, a significant concern is knowledge tracing. However, the SARPR-KT model has limitations. Specifically, the model only considers the relationship between exercises and knowledge concepts, ignoring the complex interaction among students, exercises, and skills. A secondary constraint of this approach is its exclusive focus on hierarchical levels within knowledge concepts, omitting nuanced aspects such as relevance or belonging. Therefore, one of our further types of research is necessary to address these deficiencies in future endeavors. In addition, the principle of ASSIST data processing involves natural language process transforming raw text data into feature representations that are manageable by the model. Continuous training and refinement ensure that our proposed model performs well across various mathematical tasks and scenarios. another further research is transferring our model and methodology to other educational datasets, consists of algebra 2005 and 2006, ASSIST 2017, aims to enhance the generalization capability of the model.

Acknowledgments

The research was supported by the Science and Technology Project of Jiangxi Province (NO. GJJ213805).

References

[1] A.C. Graesser, P. Chipman, B.C. Haynes, A. Olney, AutoTutor: an intelligent tutoring system with mixed-initiative dialogue, IEEE Transactions on Education 48(4)(2005) 612-618.
https://doi.org/10.1109/TE.2005.856149

[2] D. Auguste, Book review: Intelligent Tutoring Systems: D. Sleeman and J.S. Brown, (Academic Press, New York, 1982); 345 pages, $39.50, Artificial Intelligence 26(2)(1985) 233-238.
https://doi.org/10.1016/0004-3702(85)90033-5

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

[4] R. Baker, A.T. Corbett, V. Aleven, More Accurate Student Modeling through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing, in: Proc. International Conference on Intelligent Tutoring Systems, 2008.
https://doi.org/10.1007/978-3-540-69132-7_44

[5] M. Hafidi, T. Bensebaa, Architecture for an adaptive and intelligent tutoring system that considers the learner's multiple intelligences, International Journal of Distance Education Technologies 13(1)(2015) 1-21.
https://doi.org/10.4018/ijdet.2015010101

[6] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L.J. Guibas, J.N. Sohl-Dickstein, Deep knowledge tracing, in: Proc. NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015.
https://dl.acm.org/doi/abs/10.5555/2969239.2969296

[7] A. Vaswani, N.M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017.
https://dl.acm.org/doi/10.5555/3295222.3295349

[8] J. Zeng, Q. Zhang, N. Xie, B. Yang, Application of Deep Self-Attention in Knowledge Tracing. <https://arxiv.org/abs/2105.07909>, 2021 (accessed 17.04.2023).

[9] I. Szita, V. Gyenes, A. Lőrincz, Reinforcement Learning with Echo State Networks, in: Proc. 16th International Conference on Artificial neural networks -- ICANN 2006, 2006.
https://doi.org/10.1007/11840817_86

[10] W. Zhang, Y. Zhang, S. Liu, X. Shang, Online Deep Knowledge Tracing, in: Proc. 2022 IEEE International Conference on Data Mining Workshops (ICDMW), 2022.
https://doi.org/10.1109/ICDMW58026.2022.00047

[11] T. Long, J. Qin, J. Shen, W. Zhang, W. Xia, R. Tang, X. He, Y. Yu, Improving Knowledge Tracing with Collaborative Information, in: Proc. the Fifteenth ACM International Conference on Web Search and Data Mining, 2022.
https://doi.org/10.1145/3488560.3498374

[12] C. Yeung, D.Y. Yeung, Addressing two problems in deep knowledge tracing via prediction-consistent regularization, in: Proc. the Fifth Annual ACM Conference on Learning at Scale, 2018.
https://doi.org/10.1145/3231644.3231647

[13] X. Sun, X. Zhao, B. Li, Y. Ma, R. Sutcliffe, J. Feng, Dynamic key-value memory networks with rich features for knowledge tracing, IEEE Transactions on Cybernetics 52(8)(2022) 8239-8245.
https://doi.org/10.1109/TCYB.2021.3051028

[14] S. Ruan, W. Wei, J. Landay, Variational deep knowledge tracing for language learning, in: Proc. LAK21: 11th International Learning Analytics and Knowledge Conference, 2021.
https://doi.org/10.1145/3448139.3448170

[15] S. Pandey, G. Karypis, A Self-Attentive model for Knowledge Tracing. <https://arxiv.org/abs/1907.06837>, 2019 (accessed 17.04.2023).

[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
https://doi.org/10.1109/CVPR.2016.90

[17] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer Normalization. <https://arxiv.org/abs/1607.06450>, 2016 (accessed 21.6.2023).

[18] B. Gagana, H.A.U. Athri, S. Natarajan, Activation Function Optimizations for Capsule Networks, in: Proc. 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018.
https://doi.org/10.1109/ICACCI.2018.8554604