International Journal of Artificial Intelligence in Education
https://doi.org/10.1007/s40593-022-00297-z

ARTICLE

Interpreting Deep Learning Models for Knowledge Tracing

Yu Lu[1,2] · Deliang Wang[1,2] · Penghe Chen[1,2] · Qinggang Meng[1,2] · Shengquan Yu[1,2]

Accepted: 6 June 2022

Abstract
As a prominent aspect of modeling learners in the education domain, knowledge tracing attempts to model learner's cognitive process, and it has been studied for nearly 30 years. Driven by the rapid advancements in deep learning techniques, deep neural networks have been recently adopted for knowledge tracing and have exhibited unique advantages and capabilities. Due to the complex multilayer structure of deep neural networks and their "black box" operations, these deep learning based knowledge tracing (DLKT) models also suffer from non-transparent decision processes. The lack of interpretability has painfully impeded DLKT models' practical applications, as they require the user to trust in the model's output. To tackle such a critical issue for today's DLKT models, we present an interpreting method by leveraging explainable artificial intelligence (xAI) techniques. Specifically, the interpreting method focuses on understanding the DLKT model's predictions from the perspective of its sequential inputs. We conduct comprehensive evaluations to validate the feasibility and effectiveness of the proposed interpreting method at the skill-answer pair level. Moreover, the interpreting results also capture the skill-level semantic information, including the skill-specific difference, distance and inner relationships. This work is a solid step towards fully explainable and practical knowledge tracing models for intelligent education.

Keywords Artificial intelligence in education · Intelligent tutoring system · Educational data mining · Intelligent agent · Interpretability of deep learning

Introduction

In the education domain, knowledge tracing refers to modeling and predicting the knowledge states of learners by quantitatively diagnosing their mastery levels on skills (e.g., "integer" or "fraction" in algebra). The knowledge tracing models and

✉ Shengquan Yu
yusq@bnu.edu.cn

[1] Advanced Innovation Center for Future Education, Beijing Normal University, Beijing, China

[2] School of Educational Technology, Faculty of Education, Beijing Normal University, Beijing, China

Published online: 27 June 2022

the predicted knowledge states can be used to identify learner's current strengths and weaknesses at the individual skill level. Hence, they have served as a key component in today's intelligent tutoring system (ITS) (Yazdani, 1989) and massive open online course (MOOC) (Pappano, 2012) platforms to help provide scaffolding support and personalized recommendation of learning resources or courses (Pardos et al., 2013; Wang et al., 2016). Most traditional knowledge tracing models, such as the Bayesian knowledge tracing (BKT) model (Corbett & Anderson, 1994), consider the knowledge state as a latent variable and adopt the statistical methods to estimate the key parameters. Given a new learner's exercise performance on a sequence of questions (i.e., either correctly-answered or falsely-answered), the built knowledge tracing model could calculate the learner's skill mastery probabilities in a timely manner and subsequently make predictions on the learner's future performance.

Driven by rapid advancements in deep learning techniques and their powerful representation capabilities, a variety of deep learning based knowledge tracing (DLKT) models have been proposed (Piech et al., 2015; Zhang et al., 2017a). These DLKT models can capture the inherent information from the exercise data from large-scale learners and concurrently predict learner performance on multiple skills. Given the temporal characteristics of learners' exercise data, the recurrent neural network (RNN) (Schuster & Paliwal, 1997) is often adopted for building DLKT models (Piech et al., 2015; Zhang et al., 2017b; Yang & Cheung, 2018; Wang et al., 2017; Tong et al., 2020). As popular implementation variants of the RNN, the long short-term memory (LSTM) unit (Hochreiter & Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014) have achieved comparable or better performances than many other knowledge tracing models in practice (Xiong et al., 2016; Montero et al., 2018). On the other hand, the rapid development of ITS and MOOC platforms greatly facilitates the inexpensive collection of large-scale data from learners. The available rich information sources and the large number of online learners also promote the study and usage of DLKT models.

Similar to deep learning models in many other domains, many existing DLKT models also suffer from interpretability issues (Grégoire et al., 2018). In other words, it is principally hard to map a DLKT model's abstract decision (e.g., predicting the next question falsely answered) onto a target task that the end-users could easily make sense of (e.g., enabling ITS designers or learners to understand why giving such a prediction). The non-transparent decision process of many DLKT models requires providing users with convincing criteria to explain the obtained predictions. The interpretability issue has painfully impeded the practical applications of DLKT models for system designers, teachers and learners.

Fortunately, explainable artificial intelligence (xAI) can be an effective tool to tackle the interpretability issue. xAI is not a new topic in AI domain, and it mainly targets explaining complex AI models, especially for deep neural networks (Xu et al., 2019). We attempt to introduce xAI techniques to interpret RNN-based DLKT models and accordingly enable the users to easily understand and manage the DLKT models. More importantly, the interpreting results could help to increase the trust of users (e.g., teachers and learners) in the systems that have implemented the DLKT models. In particular, we present an interpreting

method specifically for the DLKT models, which analyzes the contributions of each model's input skill-answer pairs to explain the model's final predictions. The method is mainly based on a classic interpreting method, called layer-wise relevance propagation (LRP) (Bach et al., 2015). The LRP method is originally designed for the image classification models. It enables users to understand the model decisions by pixel-wise decomposition of the image classifiers. The LRP method is attractive, as its backward propagation rule can be implemented efficiently and modularly (Montavon et al., 2019; Lu et al., 2020).

The key contributions of this work can be summarized as follows:

- To tackle the interpretability issue faced by the existing knowledge tracing models, we advocate introducing xAI techniques into the knowledge tracing field. To the best of our knowledge, this is the first work explicitly utilizing xAI techniques to systematically interpret the DLKT models.
- We present a post-hoc interpreting approach based on the classic LRP method to explain the DLKT models' output predictions from their readily available input variables. The approach distributes the contributions of the model inputs to the individual skill-answer pairs, which is generally applicable to RNN-based DLKT models.
- We conduct comprehensive experiments to evaluate the interpreting method at both the pair level and skill level, and the results validate the effectiveness of the proposed interpreting method.

While the designed interpreting method is effective, we believe the key impact of this work is to highlight the possibility of utilizing xAI techniques to tackle the interpretability issues of learner modeling in education, and eventually enables learners to be modeled in a more transparent and accurate way.

The rest of the paper is organized as follows. "Related Work" summarizes related work, "Interpreting DLKT Model" presents how to interpret the DLKT models, and "Evaluation" shows the evaluation results. Finally, we present a discussion in "Discussion" and conclude in "Conclusion".

Related Work

Knowledge Tracing Models

Early studies adopted statistical models to conduct knowledge tracing, where BKT can be regarded as the most prominent knowledge tracing model. The BKT model mainly adopts a hidden Markov model (HMM) to estimate learner's mastery state with respect to individual skills, and subsequent studies improved the model by considering cognitive factors (Baker et al., 2008), learners' abilities (Yudelson et al., 2013; Liu & Koedinger, 2017), prior knowledge (Chen et al., 2017), the difficulty level of questions (Pardos & Heffernan, 2011) and the learning time (Baker et al., 2011). In addition to the BKT model, factor analysis (Pavlik et al., 2009; Cen et al., 2006) and matrix factorization-based (Vie & Kashima, 2019; Thai-Nghe et al.,

2012) models have been proposed and investigated. Moreover, researchers have recently adopted functional magnetic resonance imaging (fMRI) signals to improve knowledge tracing models (David & et al., 2018).

Deep learning techniques were recently introduced into the knowledge tracing domain, as they have sufficient capacity for automatically learning the inherent relationships and do not require explicit labels at the skill level. The deep knowledge tracing (DKT) model (Piech et al., 2015) that employs an RNN is the pioneering work. Subsequently, a number of RNN-based DLKT models have been proposed (Zhang et al., 2017b; Yang & Cheung, 2018; Wang et al., 2017; Tong et al., 2020). They normally operate as "black-box" and are difficult to explicitly capture the knowledge state variables inside their inner structure. In addition, educational relations (Chen et al., 2018), prediction-consistent regularization (Yeung & Yeung, 2018) and learning behaviors (Nagatani et al., 2019) have been considered and utilized to design RNN-based models. In short, RNN-based models currently play a crucial role and have the most variants among different DLKT models.

To better trace learner's knowledge state, the dynamic key-value memory network (Zhang et al., 2017a) and its variants (Abdelrahman & Wang, 2019; Chaudhry et al., 2018; Yeung, 2019) were proposed, which are mainly based on memory-augmented neural networks (Santoro et al., 2016). Recently, attention mechanism (Vaswani et al., 2017) has also been introduced to handle sparse data of learners (Pandey & Karypis, 2019) and incorporate rich information (Ghosh et al., 2020; Su et al., 2018; Liu et al., 2019). The new structures of these DLKT models partially solve the interpretability issue by introducing additional components (e.g., external memory or attention module). However, there is still lack of the generally applicable method that could explicitly interpret the existing DLKT models, especially the RNN-based DLKT models.

On the other hand, the researchers compared the DKT model with the extended BKT model, and found that the two models could achieve the similar performance (Khajah et al., 2016). Furthermore, the researchers found that although DKT is powerful for modeling student learning, its gain does not come from the discovery of novel representations, i.e., the key advantage of deep learning (Wilson et al., 2016). Hence, it is crucial and significant to fully exploit the DLKT models by leveraging the methods from xAI domain.

Explainable Artificial Intelligence

An AI model's interpretability can be simply regarded an *ante-hoc* category and a *post-hoc* category. The *ante-hoc* category refers to training self-explainable machine learning models (Melis & Jaakkola, 2018), but it is usually constrained to the use of simple-structured models, such as linear regression (Strumbelj & Kononenko, 2010), decision tree (Deng, 2019) or naive Bayes models (Poulin et al., 2006). Hence, the *ante-hoc* category normally has difficulty handling complex deep learning based AI models.

The *post-hoc* category refers to utilizing specific methods to interpret models that have been trained, and generally it can be further divided into *global* and

local methods. Simply speaking, the *global* methods attempt to understand the inner working mechanism or fundamental logic of a trained model. The existing methods typically include rule extraction (Mashayekhi & Gras, 2015) and model distillation (Tan et al., 2018). While *global* interpretability looks attractive, it is not applicable to many deep learning models trained by complex machine learning algorithms and many domain-specific services. On the other hand, *local* methods mainly focus on understanding a model's decisions or predictions from the perspective of its readily available input variables. Specifically, given a model's decision (e.g., classification or regression results), the *local* methods usually analyze the contributions of the input variable's features to explain that decision. Typical *local* methods include backward propagation (Zeiler & Fergus, 2014), sensitivity analysis (Andrea et al., 2004) and simple Taylor decomposition (Lapuschkin et al., 2016). For example, the backward propagation method interprets a model's decision by explicitly using its deep neural network structure and the backpropagation mechanism (Rumelhart et al., 1988). The backward prorogation method is applicable and empirically scales to general deep learning models. The LRP method is a typical backward propagation method.

In short, research on interpreting complex and domain-specific machine learning models in the xAI domain is still in its infancy. In this work, we introduce the local method LRP into KT domain to accomplish post-hoc interpretability.

Interpreting DLKT Model

As mentioned earlier, KT models need to handle the sequential and temporal characteristics of learners' exercise data, and thus RNN is often adopted, especially for the DLKT models. In this section, we first briefly introduce the existing RNN-based KT models, and then present the method to conduct the interpreting task.

RNN-based DLKT Models

Briefly, RNN is a practical and effective implementation of the neuron architecture that is able to connect previous information to the present task, such as linking a learner's past exercise information to her performance on future questions. Hence, a number of DLKT models, such as DKT (Piech et al., 2015), adopt LSTM or similar architectures (e.g., GRU) to accomplish the KT task. As illustrated in Fig. 1, the model maps input sequence vectors $\{\mathbf{x}_0,...,\mathbf{x}_{t-1},\mathbf{x}_t,...\}$ to output sequence vectors $\{\mathbf{y}_0,...,\mathbf{y}_{t-1},\mathbf{y}_t,...\}$, where $\mathbf{x}_t$ represents the skill-answer pairs, and $\mathbf{y}_t$ refers to the predicted probability vectors for mastering the skills. The implementation of LSTM unit in DLKT models can be denoted as below:

$$\mathrm{f_t} = \sigma\left(\mathbf{W}_{\mathrm{fh}}\mathrm{h}_{\mathrm{t-1}} + \mathbf{W}_{\mathrm{fx}}\mathbf{x}_{\mathrm{t}} + \mathrm{b}_{\mathrm{f}}\right) \tag{1}$$

$$\mathrm{i_t} = \sigma\left(\mathbf{W}_{\mathrm{ih}}\mathrm{h}_{\mathrm{t-1}} + \mathbf{W}_{\mathrm{ix}}\mathbf{x}_{\mathrm{t}} + \mathrm{b}_{\mathrm{i}}\right) \tag{2}$$

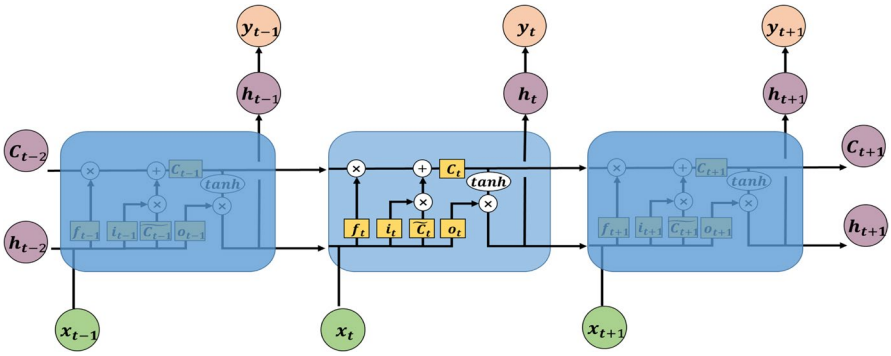

Fig. 1 The Architecture of a LSTM-based DLKT Model

$$\widetilde{\mathrm{C}}_{\mathrm{t}} = \tanh\left(\mathbf{W}_{\mathrm{ch}}\mathrm{h}_{\mathrm{t-1}} + \mathbf{W}_{\mathrm{cx}}\mathbf{x}_{\mathrm{t}} + \mathrm{b}_{\mathrm{c}}\right) \tag{3}$$

$$\mathrm{C}_{\mathrm{t}} = \mathrm{f}_{\mathrm{t}} \odot \mathrm{C}_{\mathrm{t-1}} + \mathrm{i}_{\mathrm{t}} \odot \widetilde{\mathrm{C}}_{\mathrm{t}} \tag{4}$$

$$\mathrm{o}_{\mathrm{t}} = \sigma\left(\mathbf{W}_{\mathrm{oh}}\mathrm{h}_{\mathrm{t-1}} + \mathbf{W}_{\mathrm{ox}}\mathbf{x}_{\mathrm{t}} + \mathrm{b}_{\mathrm{o}}\right) \tag{5}$$

$$\mathrm{h}_{\mathrm{t}} = \mathrm{o}_{\mathrm{t}} \odot \tanh\left(\mathrm{C}_{\mathrm{t}}\right). \tag{6}$$

After obtaining the output h_t, the DLKT models usually employ another layer to output the final prediction y_t as below:

$$\mathbf{y}_{\mathrm{t}} = \sigma\left(\mathbf{W}_{\mathrm{yh}}\mathrm{h}_{\mathrm{t}} + \mathrm{b}_{\mathrm{y}}\right) \tag{7}$$

The implementations above usually consist of two types of connections, namely ***weighted linear connections***, i.e., (1), (2), (3), (5), and (7), and ***multiplicative connections***, i.e., (4) and (6). These two connection types are handled by the interpreting method in different ways.

Interpreting Method Design

As mentioned earlier, the backward propagation method analyzes the contributions of input variable's features to explain a model's decision, where the model's inner structure and parameters are both considered. From the perspective of the knowledge tracing task, it is necessary to analyze and understand the relationship between the learners' exercise sequences and the DLKT model's final prediction results. We thus adopt the basic idea of the backward propagation method for interpreting DLKT models. Specifically, the interpreting task starts from the DLKT model's output, progressively and inversely mapping the model's prediction onto the lower layers until it reaches the input

learners' exercise sequences. In other words, given a built DLKT model and its current output, the share of the model's current output received by each neuron is properly redistributed by its predecessors to achieve relevance conservation, and the quantity being back propagated can be filtered to retain only the information that passes through certain neurons.

In practice, the proposed method firstly set the relevance of the output layer neuron to the model's output value for the target class, and simply ignores the output layer's other values and neurons. After that, it starts backpropagating the relevance score from the output layer to the input layer. Similar to the LRP method, the interpreting method separately addresses the weighted linear connection and multiplicative connection in the model's intermediate layer to accomplish the relevance distribution task.

Multiplicative Connection

A LTSM unit (or GRU) typically has "gate" structures in the form of multiplicative connections, as denoted in (4) and (6), where the neuron output (ranging between 0 to 1) can be called the "gate" neuron, and the remaining one can be regarded as the "source" neuron. This type of connection can be written in a general form:

$$a_j^{(l+1)} = a_g^{(l)} \odot a_c^{(l)}, \tag{8}$$

where $a_g^{(l)}$ and $a_c^{(l)}$ are the messages received in the feedforward direction by "gate" neuron g and "source" neuron c, respectively, in layer l. In the feedforward direction, the "gate" neuron and "source" neuron jointly decide how much of the information should be retained and eventually contributed to the model's decision. For the interpreting process, the previous study simply gives the full credit to the "source" neuron (Arras et al., 2017). We advocate both "gate" neuron and "source" neuron should receive the credits from the upper layer. It is because the "gate" neuron and "source" neuron jointly decide the upper layer's output during the feedforward process (i.e., the prediction process) as given in (8), and thus both neuron's contribution should not be ignored during the backpropagating process (i.e., the interpreting process). After attempting different operations, we select logarithm operation to help accomplish the backpropagating task.

Specifically, given $R_{g\leftarrow j}^{(l)}$ is the relevance received by neuron g in layer l from neuron j in layer $l+1$, the relevance can be distributed as below:

$$R_{g\leftarrow j}^{(l)} = \begin{cases} \frac{\ln\left|a_g^{(l)}\right|}{\ln\left|a_j^{(l+1)}\right| + \varepsilon * \text{sign}\left(\ln\left|a_j^{(l+1)}\right|\right)} R_j^{(l+1)} & , a_j^{(l+1)} \neq 0 \\ 0 & , a_j^{(l+1)} = 0 \end{cases} \tag{9}$$

where $R_j^{(l+1)}$ is the total relevance received by neuron j in the upper layer $l+1$, and the term $\varepsilon * sign\left(\ln\left|a_j^{(l+1)}\right|\right)$ is a stabilizer to prevent $R_{g\leftarrow j}^{(l)}$ becoming an unbounded value.

By using Equation (9), the interpreting results could jointly consider the contributions from both "gate" and "source" neurons rather than simply giving the full credit to one neuron. Accordingly, we have:

$$R_{c\leftarrow j}^{(l)} = R_j^{(l+1)} - R_{g\leftarrow j}^{(l)} \tag{10}$$

where $R_{c\leftarrow j}^{(l)}$ is the relevance received by neuron c in layer l from neuron j in layer $l+1$.

Weighted Linear Connection

As illustrated in (1), (2), (3), (5), and (7), this type of connection can be written in a general form:

$$a = f(W\mathrm{h} + W\mathbf{x} + \mathrm{b}), \tag{11}$$

where $f()$ is an activation function commonly used in deep learning models. Assuming that activation functions do not change the relevance distribution, the weighted connection can be further denoted as:

$$a_j^{(l+1)} = \sum_i w_{ij}\, a_i^{(l)} + b_j, \tag{12}$$

where $a_j^{(l+1)}$ is the message received in the feedforward direction at neuron j in layer $l+1$, w_{ij} and b_j are the corresponding connection weight and bias. Given $R_{i\leftarrow j}^{(l)}$ is the relevance received by neuron i in layer l from neuron j in layer $l+1$, we have

$$R_{i\leftarrow j}^{(l)} = \frac{w_{ij}a_i^{(l)}}{a_j^{(l+1)} + \varepsilon * sign\left(a_j^{(l+1)}\right)} R_j^{(l+1)}, \tag{13}$$

where $R_j^{(l+1)}$ is the total relevance received by neuron j in the upper layer layer $l+1$, and the item $\varepsilon * sign\left(a_j^{(l+1)}\right)$ is a stabilizer to prevent $R_{i\leftarrow j}^{(l)}$ becoming an unbounded value. In practice, ε can be a small positive value, and $sign\left(a_j^{(l+1)}\right)$ is set to 1 or -1, given $a_j^{(l+1)}$ is positive or negative respectively. Note that we do not consider the bias b_j to conserve more relevance for the lower-level neurons.

Normally, neuron i is connected to multiple neurons in upper layer (not only neuron j), and thus its total relevance can be computed as:

$$R_i^{(l)} = \sum_j R_{i\leftarrow j}^{(l)}, \tag{14}$$

where all the neurons j are located in upper layer $l+1$.

In short, the designed interpreting method can properly handle the different connections in DLKT models and accordingly compute the relevance. We can then conduct the task of interpreting the DLKT models.

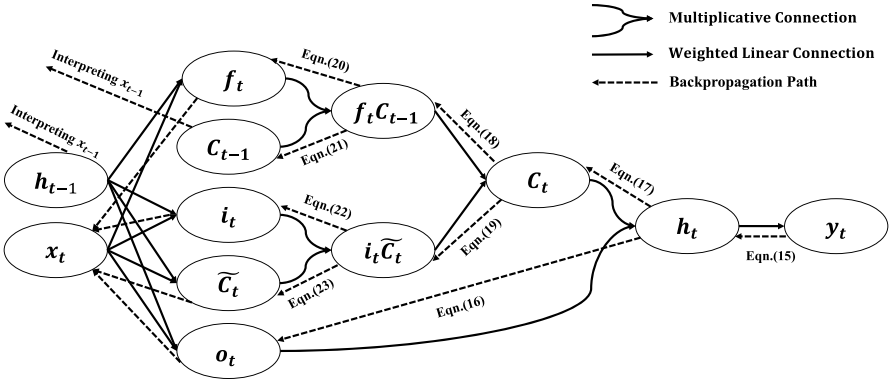

Fig. 2 The Feed Forward Prediction Path with the Two Types of Connections in LSTM and the Back Propagation Path for Interpreting its Prediction Results

Interpreting DLKT Models

The interpreting process starts from the model's output, namely the prediction results y_t, and then progressively maps the model's prediction onto the lower layers until it reaches the model input x_t. Figure 2 illustrates both the feedforward prediction path and the backpropagation path inside the RNN-based DLKT model. The two types of connections, i.e., multiplicative connection and weighted linear connection, are marked using different symbols as well.

Specifically, given $R^d_{y_t}$ is the value of the d^{th} dimension of the prediction output y_t, the first step is to calculate the relevance received by the connected neuron h_t, namely $R_{\mathrm{h_t}}$. The connection between h_t and y_t is the weighted linear connection, we thus adopt (13) to compute the relevance:

$$R_{\mathrm{h_t}} = \frac{W_{yh}\mathrm{h}_t}{W_{yh}\mathrm{h}_t + b_y + \varepsilon * sign(W_{yh}\mathrm{h}_t + b_y)} * R^d_{y_t} \tag{15}$$

After obtaining the relevance of neuron h_t, the next step is to backpropagate to the previous layer and compute the relevance of the connected neurons o_t and C_t. The connection between the two neurons and h_t is the multiplicative connection (i.e., gate structure), we thus adopt (9) and (10) to compute their relevances respectively:

$$R_{o_t} = \frac{\ln |o_t|}{\ln |h_t| + \varepsilon * \mathrm{sign}(\ln |h_t|)} * R_{h_t} \tag{16}$$

$$R_{C_t} = R_{h_t} - R_{o_t} \tag{17}$$

Based on the computed relevance of neuron C_t, we could further backpropagate and compute the relevance of the connected neurons $f_t C_{t-1}$ and $i_t \tilde{C}_t$. Both connections are the weighted linear connections, we thus adopt Equation (13) to compute their relevances respectively:

$$R_{f_t C_{t-1}} = \frac{f_t C_{t-1}}{C_t + \epsilon * sign(C_t)} * R_{C_t} \tag{18}$$

$$R_{i_t \tilde{C}_t} = \frac{i_t \tilde{C}_t}{C_t + \epsilon * sign(C_t)} * R_{C_t} \tag{19}$$

Given the multiplicative connection and the computed relevance $R_{f_t C_{t-1}}$, we then further backpropagate and obtain the relevances for neurons f_t and C_{t-1} using (9) and (10):

$$R_{f_t} = \frac{\ln |f_t|}{\ln |f_t C_{t-1}| + \epsilon * \text{sign}(\ln |f_t C_{t-1}|)} * R_{f_t C_{t-1}} \tag{20}$$

$$R_{C_{t-1}} = R_{f_t C_{t-1}} - R_{f_t} \tag{21}$$

Similarly, based on the multiplicative connection and the computed relevance $R_{i_t \tilde{C}_t}$, we could also obtain the relevances for neurons i_t and $\tilde{C}_t$ using (9) and (10):

$$R_{i_t} = \frac{\ln |i_t|}{\ln |i_t \tilde{C}_t| + \epsilon * \text{sign}(\ln |i_t \tilde{C}_t|)} * R_{i_t \tilde{C}_t} \tag{22}$$

$$R_{\tilde{C}_t} = R_{i_t \tilde{C}_t} - R_{i_t} \tag{23}$$

As illustrated in Fig. 2, the input x_t directly connects to the four neurons, namely f_t, i_t, $\tilde{C}_t$ and o_t, and all the connections are the weighted linear connections. Hence. the next step is to backpropagate to the input layer and compute the relevances from the four connected neurons respectively:

$$R_{x_t \leftarrow f_t} = \frac{W_{fx} x_t}{W_{fh} h_{t-1} + W_{fx} x_t + b_f + \epsilon * \text{sign}(W_{fh} h_{t-1} + W_{fx} x_t + b_f)} * R_{f_t} \tag{24}$$

$$R_{x_t \leftarrow i_t} = \frac{W_{ix} x_t}{W_{ih} h_{t-1} + W_{ix} x_t + b_i + \epsilon * \text{sign}(W_{ih} h_{t-1} + W_{ix} x_t + b_i)} * R_{i_t} \tag{25}$$

$$R_{x_t \leftarrow \tilde{C}_t} = \frac{W_{cx} x_t}{W_{ch} h_{t-1} + W_{cx} x_t + b_c + \epsilon * \text{sign}(W_{ch} h_{t-1} + W_{cx} x_t + b_c)} * R_{\tilde{C}_t} \tag{26}$$

$$R_{x_t \leftarrow o_t} = \frac{W_{ox} x_t}{W_{oh} h_{t-1} + W_{ox} x_t + b_o + \epsilon * \text{sign}(W_{oh} h_{t-1} + W_{ox} x_t + b_o)} * R_{ot} \tag{27}$$

The final step is to add the four relevance values above and obtain the relevance of input x_t to the output y_t at the d^{th} dimension:

$$R_{x_t}^d = R_{x_t \leftarrow \tilde{C}_t} + R_{x_t \leftarrow o_t} + R_{x_t \leftarrow f_t} + R_{x_t \leftarrow i_t} \tag{28}$$

In short, the entire backpropagation process of the interpreting method, i.e., starting from the output layer to the input layer, has been illustrated in Fig. 2, where the key steps are marked with the corresponding equations.

Table 1 A Toy Case Illustrating the Interpreting Results

Question #	Skill	Correct	Relevance
1	Addition Numbers	Y	0.15
2	Addition Numbers	Y	0.17
3	Area Rectangle	N	-0.05
4	Area Rectangle	Y	0.09
5	Subtraction Numbers	N	-0.02
6	Subtraction Numbers	Y	0.50
7	Subtraction Numbers	Y	0.58

Evaluation

We first provide a toy case to demonstrate the interpreting results computed by the given method. After that, we conduct the comprehensive evaluations to systematically understand and validate the interpreting method. Specifically, the first part of the evaluation mainly investigates the pair-level (i.e., the input skill-answer pairs) interpreting results. The second part of the evaluation further examines the skill-level interpreting results, which validates whether the skill-specific semantic and relationship information can be captured by the calculated relevance values.

The toy case is drawn from real data to demonstrate the interpreting results. Given a trained DLKT model for math and a learner's exercise sequence as the input, the input sequence consists of seven consecutive skill-answer pairs, where the questions test three different skills. Table 1 shows the question ID, the corresponding skills and whether the learner correctly answers the questions. Assuming that the 8^{th} question is testing on the skill *subtraction numbers*, the built DLKT model predicts that the leaner would correctly answer this question. By performing the proposed interpreting method, the computed relevance value for each input pair, i.e., from the 1^{st} to the 7^{th} pair is given in the last column of Table 1. We see that the computed relevance values can be either positive or negative with a wide range from -0.05 to 0.58. More importantly, these values are closely related to the correctness of the answers and the tested skills. For example, the 7^{th} question tests the same skill (i.e., *subtraction numbers*) and was correctly answered, the corresponding relevance value 0.58 is positive and large. The 3^{rd} question tests a different skill (i.e., *Area Rectangle*) and was falsely answered, the corresponding relevance value -0.05 is negative and small. This toy case exhibits the possibility and necessity to fully understand the interpreting results, and we thus conduct the comprehensive evaluations on them.

Data and Model Preparation

We build a DLKT model on the public educational dataset ASSISTment 2009[1](Feng et al., 2009), which has been commonly used for building knowledge tracing models. Specifically, we employ its "skill builder" dataset for math, and filter out all

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

the duplicate exercise sequences and those without labelling skills. Eventually, the dataset used for training and testing the DLKT model consists of 325,637 answer records on 26,688 questions associated with 110 skills from 4,151 students.

The DLKT model adopts a LSTM structure with a hidden dimensionality of 256. During the training process, the mini-batch size and dropout are set to 20 and 0.5 respectively. The Adam optimization algorithm is utilized for model training, where the iteration number and initial learning rate are set to 500 and 0.01, respectively. The DLKT model is built and evaluated with the five-fold cross validation, and its AUC, ACC and F1-score are 0.75, 0.72 and 0.79 respectively. When executing the designed interpreting algorithm on multiplicative connection, the parameter ϵ in the stabilizer can be set slightly larger to keep the relevance received by the "gate" neuron lower than that of the "source" neuron.

For the interpreting tasks, a total of 48,673 exercise sequences are used, where each sequence consists of 15 individual skill-answer pairs. For each sequence, we take its first 14 skill-answer pairs as the input for the built DLKT model, and the last pair as the ground truth to validate the model's prediction on the 15th question. The DLKT model correctly predicts the last question for 34,292 sequences, where the numbers of positive and negative predictions are 25,015 and 9,277, respectively. We then conduct evaluations at both the pair-level and the skill-level.

Pair-Level Evaluation

Consistency Experiment

Intuitively, a correctly-answered question would provide a positive contribution to the prediction results, and accordingly the calculated relevance should be a positive value. In contrast, a falsely-answered question should be assigned a negative relevance value, as it provides a negative contribution to the prediction results. We conduct the first experiment to validate it. Specifically, we define *consistent question* among the input exercise questions given that it is a "correctly-answered question with a positive relevance value" or "falsely-answered question with a negative relevance value". Accordingly, we compute the percentage of the consistent questions in the exercise sequence, and name it as the *consistent rate* of the sequence. Obviously, a high consistent rate reflects that most correctly-answered questions are assigned positive relevance values and meanwhile most falsely-answered questions are assigned negative relevance values.

Using all 34,292 correctly-predicted sequences, we adopt the proposed interpreting method to calculate the relevance values of their skill-answer pairs, and then calculate the consistent rate for each sequence. Figure 3 shows the histogram of the consistent rates for the input sequences of skill-answer pairs. Clearly, we see that the majority of the sequences achieve 90 percent (or above) consistent rate, while only a few sequences with a low consistent rate exist. Hence, the experimental results validate that the sign of the obtained relevance value is highly correlated to whether the question has been correctly answered. From Table 1, we see an exemplary but concrete example that all the correctly-answered questions receive a positive relevance value, while the two falsely-answered questions obtain a negative relevance value.

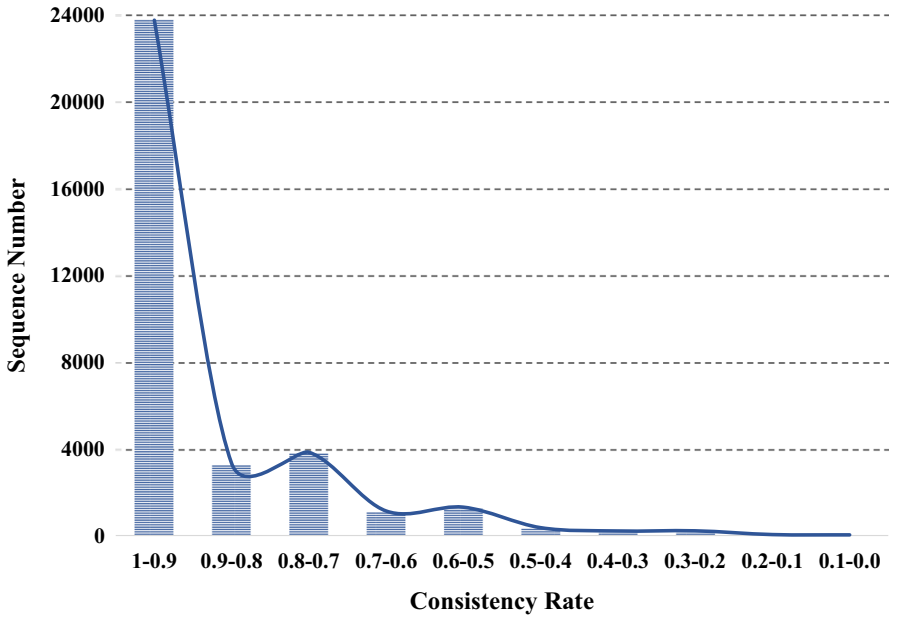

Fig. 3 Histogram of the Consistency Rates for the Correctly-Predicted Sequences

Deletion Experiment

Table 1 also indicates that the quantity of the relevance value might reflect the corresponding skill-answer pair's importance to the model's prediction. We thus design another experiment to observe the changes in the DLKT model's prediction performance when deleting specific skill-answer pairs. As mentioned earlier, 34,292 sequences were correctly predicted in the interpreting dataset, where the positive and negative predictions are 25,015 and 9,277, respectively. For the positive prediction group (i.e., the 25,015 sequences whose last questions were correctly answered), we rank the skill-answer pairs in each sequence according to their relevance values and delete the pairs in decreasing order (i.e., deleting the largest positive one first). Similarly, we rank each sequence in the negative prediction group (i.e., the 9,277 sequences whose last questions were falsely answered) according to their relevance values, and delete the pairs in increasing order (i.e., deleting the largest negative one first). Furthermore, we perform random pair deletion, and then compare the model's performances. Figure 4 illustrates the comparison results for both groups by tracking the model's accuracy over the number of pairs deleted. We clearly see that for both groups, deleting skill-answer pairs using the relevance-based method significantly decreases the model's accuracy in comparison to that obtained with the random method. In other words, the skill-answer pairs with large relevance values normally have a stronger impact on the model's final predictions.

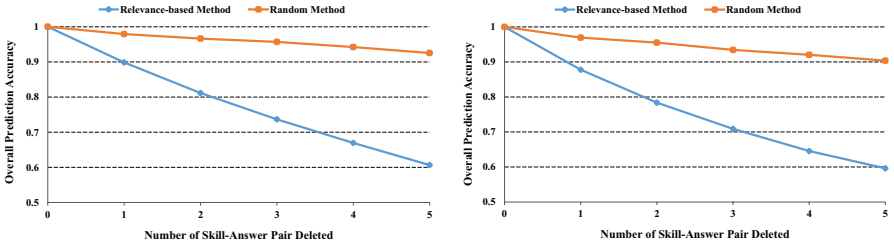

Fig. 4 Pair Deletion Results for the Correctly Predicted Sequences. **a** Positive Prediction Group. **b** Negative Prediction group

We further take the remaining 14,381 sequences that were falsely predicted in the interpreting dataset, including 7,591 positive and 6,790 negative predictions. For the positive prediction group, i.e., the last questions in their sequences were falsely answered but the model predicted them being correctly answered, we rank the pairs in each sequence according to their relevance and delete them in decreasing order, and then we compare it with the results obtained with those of the random method. For the negative prediction group, we conduct a similar experiment but delete the pairs in each sequence in increasing order. Figure 5 illustrates the comparison results for both groups by tracking the model's accuracy over the number of pairs deleted. We clearly see that for both groups, deleting skill-answer pairs using the ranked relevance values significantly increases the model's accuracy in comparison to that obtained with the random method. Hence, all the deletion experiments validate that the relevance value indicates the corresponding pair's importance to the model's predictions.

In short, the consistency experiment and the deletion experiment verify that the calculated relevance values qualitatively and quantitatively capture the pair-level contributions to the model's predictions. Simply speaking, the sign of the relevance determines whether the corresponding skill-answer pair supports the model's positive prediction (i.e., correctly answer next question). If the sign is positive, the corresponding skill-answer pair tends to increase the probability of model's positive prediction. If the sign is negative, the corresponding skill-answer pair tends to decrease the probability of model's positive prediction. Meanwhile, the absolute value of the relevance reflects the amount of the corresponding skill-answer pair increase or decrease the probability of the positive prediction. Therefore, the proposed method is feasible and effective for interpreting DLKT models.

Skill-Level Evaluation

Semantic Information

Based on the previous findings, we further examine whether the relevance values capture the semantic information from the model inputs. As mentioned earlier, each skill-answer pair usually tests one specific skill (e.g., "ordering integers" or "adding

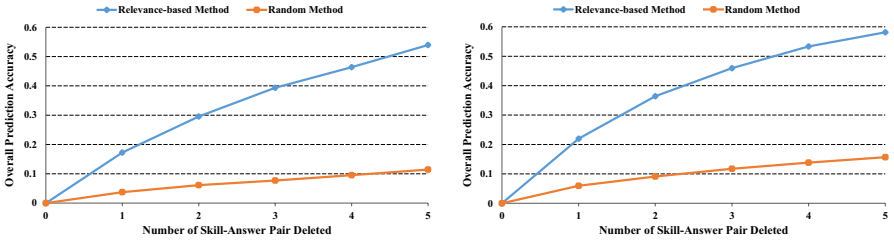

Fig. 5 Pair Deletion Results for the Falsely Predicted Sequences. **a** Positive Prediction Group. **b** Negative Prediction group

whole numbers"), and in each sequence, different pairs might test the same or different skills. We define the skill tested by the last pair, i.e., the 15^{th} skill-answer pair, as the predicted skill. Among the first 14 pairs in each sequence, we count the number of pairs that test for the predicted skill in the sequence. We still use the 34,292 correctly-predicted sequences from the previous experiment and Fig. 6 illustrates the distribution of the counting results, where the positive predictions and negative predictions are shown separately. We see that for both the positive and negative prediction groups, the sequences might have different numbers (ranging from 1 to 13) of skill-answer pairs on the predicted skill. For example, more than 2000 sequences in the positive prediction group have only one pair on the predicted skill, while only approximately 550 sequences have 13 pairs on the predicted skill. In other words, most individual sequences consist of multiple skills. Intuitively, learners' previous performances regarding the predicted skill are important, and such skill-answer pairs could provide more contributions to the final predictions than the other pairs in the sequence. We thus conduct an experiment to investigate whether the interpreting results can capture this phenomenon.

For each sequence, we take the calculated relevance values of all the pairs on the predicted skill, and simply compute the means of their absolute values. All the computed means from different sequences are put into the *same skill group* (SKG). Similarly, we compute the means of all the other pairs in each sequence, and put all the computed means into the *different skill group* (DKG). We then perform a two-tailed t-test, and Table 2 summarizes the results. For the positive predictions, the mean of the SKG group is significantly higher than that of the DKG group ($t = 149.914$, $p < .001$). For the negative predictions, the mean of the SKG group is also significantly higher than that of the DKG group ($t = 40.538$, $p < .001$). These results validate that the relevance values can reflect the importance of the pairs on the predicted skills. This finding also indicates that the interpreting results could partially capture the semantics of skill-answer pairs given a skill-specific prediction task. Note that in this experiment, we remove the sequences with 0 and 14 pairs on the predicted skills, as it is infeasible to compute the mean values for SKG and DKG groups.

Furthermore, we also examine whether the calculated relevance value could be aware of the distance between the corresponding pair and the last question used

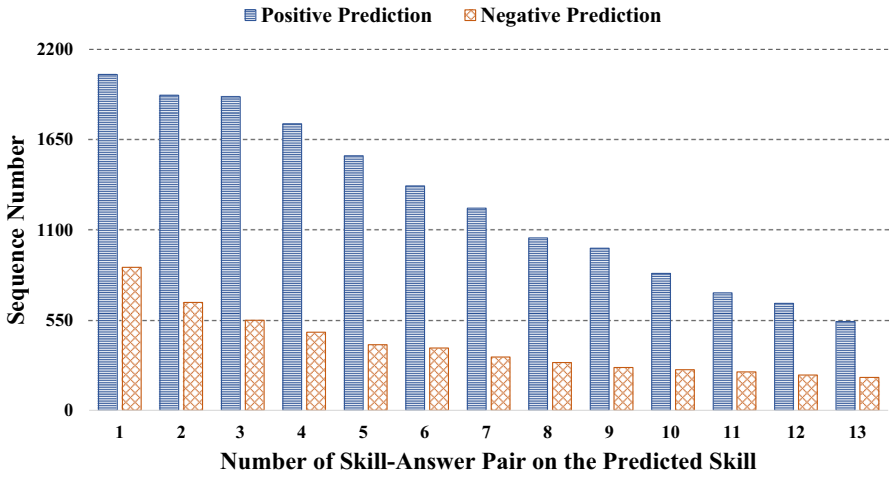

Fig. 6 Distribution of Skill-Answer Pairs among the Predicted Skill

Table 2 T-test between the Same Skill Group and Different Skill Group

	Group	N	Mean	S.D.	T-test
Positive Predictions	**SKG**	16,556	0.265	0.162	149.914***
	DKG	16,556	0.065	0.059	
Negative Predictions	**SKG**	5,108	0.140	0.102	40.538***
	DKG	5,108	0.060	0.098	

***$p < 0.001$

for prediction. Specifically, we simply regard the first 7 pairs in each sequence as the far pairs and the last 7 pairs as the close pairs. For each sequence, we take all the relevance values of the far pairs that test the predicted skill, and compute the mean of their absolute values. The computed mean values from all the correctly-predicted sequences are put into the *far group* (FG). Similarly, we form the *close group* (CG) from the last 7 pairs in each sequence. We then perform a two-tailed t-test, and Table 3 summarizes the results. We see that for the positive predictions, the mean of the CG is significantly higher than that of the FG (t = -263.472, p < .001). For the negative predictions, the mean of the CG is also significantly higher than that of the FG (t = -80.401, p < .001). The results show that the skill-answer pair closer to the last question for prediction tends to receive a larger relevance value, which is intuitively reasonable. Note that in this experiment, we remove the sequences that do not have any pairs on the predicted skills either in the first 7 pairs or last 7 pairs.

Table 4 demonstrates a concrete case for illustration purposes. We see that the predicted skill is *proportion* (i.e., the 15^{th} pair with skill ID 47), which is correctly answered and predicted. The first 14 pairs include 10 different skills (e.g., addition or multiplication whole numbers), where only the 7^{th} and 13^{th} pairs directly test the

Table 3 T-test between the Far Group and Close Group

	Group	N	Mean	S.D.	T-test
Positive Predictions	**FG**	11,171	0.051	0.032	-263.472***
	CG	11,171	0.179	0.054	
Negative Predictions	**FG**	4,224	0.052	0.043	-80.401***
	CG	4,224	0.122	0.051	

***$p < 0.001$

Table 4 A Concrete Case Illustrating the Semantics of Pair Relevance

Seq. #	Skill ID	Correct	Relevance	Ranking
1	11	Y	0.033	8
2	10	Y	0.021	10
3	29	Y	0.034	7
4	28	Y	0.048	3
5	29	Y	0.038	6
6	33	Y	0.028	9
7	**47**	**Y**	**0.062**	**2**
8	37	Y	0.042	5
9	36	N	-0.111	12
10	29	Y	0.044	4
11	41	N	-0.075	11
12	43	N	-0.115	13
13	**47**	**Y**	**0.357**	**1**
14	43	N	-0.121	14
15	47	Y	N.A	N.A

10: Mode; 11: Range; 28: Calculations with Similar Figures; 29: Conversion of Fraction Decimals Percents; 33: Ordering Integers; 36: Addition Whole Numbers; 37: Division Fractions; 41: Multiplication Fractions; 43: Percent Of; 47: Proportion

skill *proportion*. Given that both pairs are correctly answered, their relevance values, namely 0.357 for the 13^{th} pair and 0.062 for the 7^{th} pair, are both positive, and ranked as the largest among all the relevance values. Furthermore, the relevance value of the 13^{th} pair is obviously larger than that of the 7^{th} pair, meaning that it provides the largest contributions to the final prediction made by the DLKT model.

Relationship Information

Based on the previous findings, we further investigate whether the interpreting results can retain the inner relationships of the skills and capture the meaningful information, especially for the diagnosis purposes. In practice, for some skills A and B, learners find it difficult to understand skill A if they cannot understand skill B. In that case, the two skills are often closely related, and possibly the prerequisite or

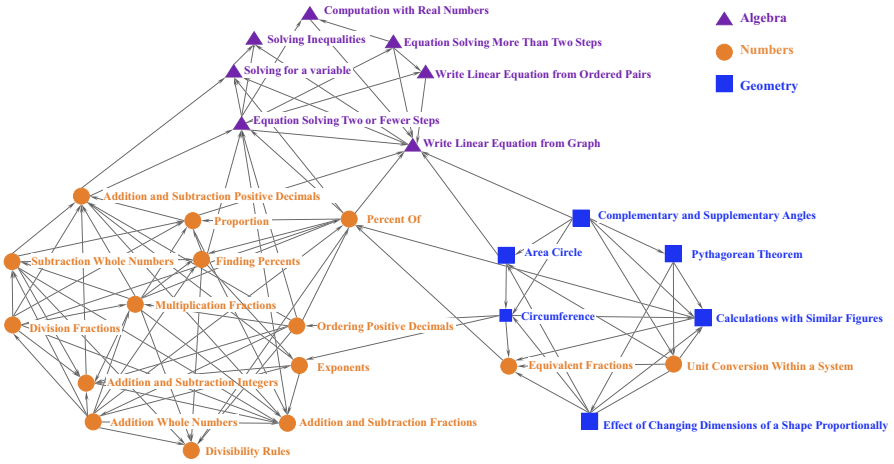

Fig. 7 A Directed Diagnosis Graph Derived from a Subset of Skills

inclusion relation might exist between them. Hence, we design a simple approach to identify such relations among the skills by leveraging on the previous interpreting results. Specifically, among the first 14 skill-answer pairs in each correctly-predicted sequence, we choose all the falsely-answered pairs on the skills that are different from the predicted skill. We then group these pairs by skill and find the skill with the largest negative value. Hence, for each predicted skill, we obtain another skill, named the *adjacent skill*. For each predicted skill, it is possible to have multiple adjacent skills derived from different sequences, and one predicted skill might be the adjacent skill of another predicted skill. The predicted and adjacent skills together with their directed linkages can be further formulated as a directed skill-specific diagnosis graph, where the skills are nodes, and each directed link starts at an adjacent skill and ends at its predicted skill. Using the most frequently-linked skills, Fig. 7 presents a graph automatically generated by a social network analysis tool called Ucient (Borgatti et al., 2002).

Clearly, we see that the skills are generally grouped into three clusters, and each cluster roughly maps one higher-level category of math, namely *numbers*, *algebra* and *geometry*. While several skills from *numbers* category are mixed with the skills in *geometry*, the intralinks inside each cluster are apparently more than the interlinks across clusters. Table 5 summarizes the *network densities* (Knoke & Yang, 2019) of different categories. As a commonly used metric in social network analysis to depict the connectedness of a network, *network density* is defined as the proportion of existing links to all probable links in a network (Borgatti & Cross, 2003). We see that the overall network density is 0.131, while the density of each category is much larger. For example, the category of *geometry* and the category of *algebra* achieve large densities of 0.367 and 0.333, respectively.

Furthermore, we see many meaningful connections among the skills in the graph. For example, multiple skills are directed to the skill *addition and subtraction fraction* in *numbers* cluster. Additionally, we see that the skills *percent of* and *write linear*

Table 5 The Statistics of the Skill Categories and their Network Densities

Category	Node #	Link #	Density
Overall	28	99	0.131
Numbers	15	52	0.248
Algebra	7	14	0.333
Geometry	6	11	0.367
Numbers + Algebra	22	75	0.162
Numbers + Geometry	21	72	0.171
Algebra + Geometry	13	27	0.173

equation from graph stand at the central position of the three clusters, which indicates that they might serve as the bridging roles in learning these skills. In addition, each predicted skill on the graph has a number of adjacent skills. For example, two adjacent skills, namely *equation solving more than two steps* and *equation solving two or fewer steps*, are the adjacent skills of the predicted skill *write linear equation from ordered pairs*. From the original skill-answer pair sequences, we can also find the skills with a high error rate. We thus have two skill sets for each predicted skill, namely, *adjacent skill set* and *error skill set*. We then define the coverage ratio as the number of skills in both skill sets divided by the number of skills in the error skill set. For all the predicted skills on the graph, the averaged coverage ratio is approximately 0.79, which indicates that the graph can capture most of the error-prone skills that are relevant to the predicted skill. Note that the experiment is running on the skill builder dataset of ASSISTment 2009, where the questions have been grouped by higher-level categories in some sequences. Given the interpreting method favors the recent skills and we only selects the most relevant one (i.e., the skill with the largest negative relevance value), this feature of the dataset might reduce the possibility of identifying the cross-category relations.

Discussion

Summary of the Key Findings

Given the non-transparent decision process of the existing DLKT models, we adopt xAI techniques to interpret the model predictions. The designed post-hoc interpreting method is generally applicable to a wide range of DLKT models. We perform the proposed interpreting method on a DLKT model using LSTM structure, and conduct comprehensive experiments to validate the interpreting results. From the pair-level perspective, we find that most of the input sequences achieve high consistent rates, which means that most correctly-answered questions are assigned positive relevance values and most falsely-answered questions are assigned negative relevance values. Furthermore, the relevance value indicates the corresponding skill-answer pair's importance to the model's final prediction. From the skill-level perspective, we find that the interpreting results can capture the semantic information from the inputs,

including the skill-specific difference, distance and inner relationships. Accordingly, the interpreting results could help to identify the error-prone skills that are relevant to the predicted skill. In summary, it is reasonable to confirm the effectiveness of the proposed interpreting method.

Potential Application

The obtained interpreting results have great potential to support a variety of intelligent services and applications in education. We simply provide an exemplary usage case that utilizes the interpreting results for learner diagnosis. Given a DLKT model built on a category of skills in one subject (e.g., geometry) and an exercise sequence from a new learner, where the last question on skill K is falsely answered, the calculated relevance values and the generated skill-specific diagnosis graph can be directly used to find the possible weak skills of the current new learner. Figure 8 illustrates a simplified diagnosis workflow. The diagnosis service could first take the entire sequence except the last pair as the DLKT model's input, and accordingly obtain the correctness probability of the last question. If the model's prediction conflicts with the ground truth, it might require more exercise information from the current learner or a more accurate DLKT model. Otherwise, the service could start performing the interpreting method to generate the relevance values for all input pairs. Among the input skill-answer pairs, the pairs with large negative relevance values are checked to determine whether at least one of them tests the same skill K. If so, the system could add skill K to the set of suspicious weak skills. Furthermore, if any other pairs with large negative relevance values can be found on the diagnosis graph and directed to skill K, they would also be added into the set of suspicious weak skills. These suspicious weak skills can be provided to learners or teachers directly, or used to build recommendation engine for ITS or MOOC platforms. They can also be utilized to analyze the possible root causes of the weakness on skill K by introducing expert experiences or educational relation information, such as prerequisite or inclusion relations. This example also demonstrates that the interpreting results could be used to provide the skill mastery information on individual learners (e.g., suspicious skills), while most DLKT models cannot directly output the skill mastery information but only the predicted correctness of next question.

Conclusion

In this work, we designed a post-hoc interpreting method for knowledge tracing that can be applied to the general RNN-based DLKT models. By progressively mapping the model outputs onto the input layer using the backward propagation mechanism, the interpreting method assigns a relevance value to each individual input skill-answer pair. We conducted comprehensive experiments to evaluate the proposed method at both pair-level and skill-level. The evaluation results showed that the assigned relevance values qualitatively and quantitatively captured the pair-level

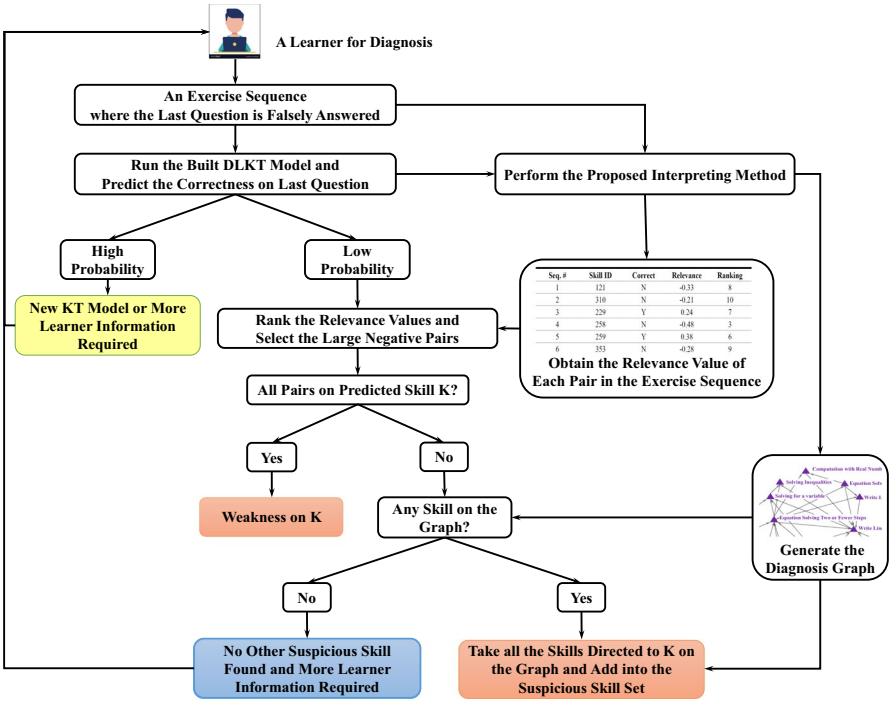

Fig. 8 An Exemplary Work Flow for Diagnosis

contributions to the model's predictions. Moreover, they could also capture the skill-level semantic information from the model inputs, including the skill-specific difference, distance and inner relationships.

The proposed interpreting method demonstrates the feasibility of using xAI techniques to improve the DLKT models. We believe this work could inspire many new interpreting methods, including both post-hoc and ante-hoc methods, using a similar design philosophy and validation methodology. In addition, the interpreting results can be used as the criteria to validate the new DLKT models before their deployment. Hence, it could be a solid step towards building fully interpretable and effective DLKT models to eventually solve the knowledge tracing problem in education.

On a broader canvas, the ultimate objective of interpreting DLKT models is to increase their usability and utility for ordinary users. For intelligent tutoring systems, the system designers could take the DLKT model's outputs and its interpreting results as inputs to user-friendly modules. The user interface of the designed modules should enable the system users to easily perceive and understand the key information of the system operation, such as how the system evaluates the learner's current knowledge status and how his or her weak skills are identified. From the perspective of teachers and learners, it could directly help to increase their trust on the intelligent services supported by the latest complex models.

Acknowledgements This research is partially supported by National Natural Science Foundation of China (No. 62077006 and No.62177009), Open Project of the State Key Laboratory of Cognitive Intelligence (No. iED2021-M007) and the Fundamental Research Funds for the Central Universities.

Author Contributions All authors contributed to the system design and implementation. Algorithm design, data collection and analysis were performed by all authors. The first draft of the manuscript was written by Yu Lu and Deliang Wang, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Availability of Data, Material and Code Data and materials created for this research are available upon request. Please direct all inquiries to the corresponding author.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

Abdelrahman, G., & Wang, Q. (2019). Knowledge tracing with sequential key-value memory networks. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval* (pp. 175–184).

Andrea, S., Stefano, T., Francesca, C., & Ratto, M. (2004). *Sensitivity analysis in practice: a guide to assessing scientific models*. Hoboken: Wiley.

Arras, L., Montavon, G., Müller, K.R., & Samek, W. (2017). Explaining recurrent neural network predictions in sentiment analysis. *EMNLP*, *2017*, 159.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Muller, K., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *Plos One*, *10*(7), 0130140.

Baker, R.S., Corbett, A.T., & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International conference on intelligent tutoring systems* (pp. 406–415). Springer.

Baker, R.S., Goldstein, A.B., & Heffernan, N.T. (2011). Detecting learning moment-by-moment. *International Journal of Artificial Intelligence in Education*, *21*(1-2), 5–25.

Borgatti, S.P., & Cross, R. (2003). A relational view of information seeking and learning in social networks. *Management Science*, *49*(4), 432–445.

Borgatti, S.P., Everett, M.G., & Freeman, L.C. (2002). *Ucinet for windows: Software for social network analysis*, (p. 6). Harvard: Analytic Technologies.

Cen, H., Koedinger, K.R., & Junker, B.W. (2006). Learning factors analysis – a general method for cognitive model evaluation and improvement. In *Proceedings of international conference on intelligent tutoring systems* (pp. 164–175).

Chaudhry, R., Singh, H., Dogga, P., & Saini, SK (2018). Modeling hint-taking behavior and knowledge state of students with multi-task learning. In *Proceedings of educational data mining*.

Chen, P., Lu, Y., Zheng, V.W., & Pian, Y. (2018). Prerequisite-driven deep knowledge tracing. In *2018 IEEE international conference on data Mining (ICDM)* (pp. 39–48). IEEE.

Chen, Y., Liu, Q., Huang, Z., Wu, L., Chen, E., Wu, R., Su, Y., & Hu, G. (2017). Tracking knowledge proficiency of students with educational priors. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 989–998). ACM.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv:14091259.

Corbett, A.T., & Anderson, J.R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, *4*(4), 253–278.

David, H., et al. (2018). Knowledge tracing using the brain. In *Proceedings of the educational data mining (EDM)*.

Deng, H. (2019). Interpreting tree ensembles with intrees. *International Journal of Data Science and Analytics*, *7*, 277–287.

Feng, M., Heffernan, N., & Koedinger, K. (2009). Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*, *19*(3), 243–266.

Ghosh, A., Heffernan, N., & Lan, A.S. (2020). Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2330–2339).

Grégoire, M., Wojciech, S., & Klaus-Robert, M. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, *73*, 1–15.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Khajah, M., Lindsey, R.V., & Mozer, M.C. (2016). How deep is knowledge tracing? arXiv:160402416.

Knoke, D., & Yang, S. (2019). *Social network analysis* Vol. 154. Thousand Oaks: Sage Publications.

Lapuschkin, S., Binder, A., Montavon, G., Muller, K., & Samek, W. (2016). Analyzing classifiers: Fisher vectors and deep neural networks. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2912–2920).

Liu, Q., Huang, Z., Yin, Y., Chen, E., Xiong, H., Su, Y., & Hu, G. (2019). Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, *33* (1), 100–115.

Liu, R., & Koedinger, K.R. (2017). Towards reliable and valid measurement of individualized student parameters. In *Proceedings of the 10th international conference on educational data mining* (pp. 135–142).

Lu, Y., Wang, D., Meng, Q., & Chen, P. (2020). Towards interpretable deep learning models for knowledge tracing. In *International conference on artificial intelligence in education* (pp. 185–190). Springer.

Mashayekhi, M., & Gras, R. (2015). Rule extraction from random forest: the rf+ hc methods. In *Proceedings of canadian conference on artificial intelligence* (pp. 223–237).

Melis, D.A., & Jaakkola, T.S. (2018). Towards robust interpretability with self-explaining neural networks. In *Proceedings of advances in neural information processing systems(NIPS)* (pp. 7786–7795).

Montavon, G., Binder, A., Lapuschkin, S., Samek, W., & Müller, K.R. (2019). Layer-wise relevance propagation: an overview. In *Explainable AI: interpreting, explaining and visualizing deep learning* (pp. 193–209).

Montero, S., Arora, A., Kelly, S., Milne, B., & Mozer, M. (2018). Does deep knowledge tracing model interactions among skills?. In *Proceedings of the 11th international conference on educational data mining*.

Nagatani, K., Zhang, Q., Sato, M., Chen, Y.Y., Chen, F., & Ohkuma, T. (2019). Augmenting knowledge tracing by considering forgetting behavior. In *The world wide web conference* (pp. 3101–3107).

Pandey, S., & Karypis, G. (2019). A self-attentive model for knowledge tracing. arXiv:190706837.

Pappano, L. (2012). The year of the mooc. *The New York Times*, *2*(12), 2012.

Pardos, Z.A., & Heffernan, N.T. (2011). Kt-idem: introducing item difficulty to the knowledge tracing model. In *International conference on user modeling, adaptation, and personalization* (pp. 243–254). Berlin: Springer.

Pardos, Z.A., Bergner, Y., Seaton, D.T., & Pritchard, D.E. (2013). Adapting bayesian knowledge tracing to a massive open online course in edx. *EDM*, *13*, 137–144.

Pavlik, Jr, P.I., Cen, H., & Koedinger, K.R. (2009). Performance factors analysis–a new alternative to knowledge tracing. In *Proceedings of international conference on artificial intelligence in education*.

Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. In *Advances in neural information processing systems* (pp. 505–513).

Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D.S., Fyshe, A., Pearcy, B., Macdonell, C., & Anvik, J. (2006). Visual explanation of evidence in additive classifiers. In *Proceedings of national conference on artificial intelligence* (pp. 1822–1829).

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1988). Learning representations by back-propagating errors. *Nature*, *323*(6088), 696–699.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., & Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *Proceedings of international conference on machine learning* (pp. 1842–1850).

Schuster, M., & Paliwal, K.K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Strumbelj, E., & Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, *11*, 1–18.

Su, Y., Liu, Q., Liu, Q., Huang, Z., Yin, Y., Chen, E., Ding, C., Wei, S., & Hu, G. (2018). Exercise-enhanced sequential modeling for student performance prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tan, S., Caruana, R., Hooker, G., & Lou, Y. (2018). Distill-and-compare: Auditing black-box models using transparent model distillation. In *Proceedings of AAAI/ACM conference on AI, ethics, and society* (pp. 303–310).

Thai-Nghe, N., Drumond, L., Horváth, T, Krohn-Grimberghe, A., Nanopoulos, A., & Schmidt-thieme, L. (2012). Factorization techniques for predicting student performance. In *Educational recommender systems and technologies: Practices and challenges, IGI Global* (pp. 129–153).

Tong, H., Zhou, Y., & Wang, Z. (2020). Exercise hierarchical feature enhanced knowledge tracing. In *International conference on artificial intelligence in education* (pp. 324–328). Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv:170603762.

Vie, J., & Kashima, H. (2019). Knowledge tracing machines: Factorization machines for knowledge tracing. In *Proceedings of AAAI conference on artificial intelligence*, (Vol. 33 pp. 750–757).

Wang, L., Sy, A., Liu, L., & Piech, C. (2017). Deep knowledge tracing on programming exercises. In *Proceedings of the fourth (2017) ACM conference on learning@ scale* (pp. 201–204).

Wang, Z., Zhu, J., Li, X., Hu, Z., & Zhang, M. (2016). Structured knowledge tracing models for student assessment on coursera. In *Proceedings of the third (2016) ACM conference on learning@ scale* (pp. 209–212).

Wilson, K.H., Xiong, X., Khajah, M., Lindsey, R.V., Zhao, S., Karklin, Y., Van Inwegen, E.G., Han, B., Ekanadham, C., Beck, J.E., & et al. (2016). Estimating student proficiency: Deep learning is not the panacea. In *In Neural information processing systems, workshop on machine learning for education*, Vol. 3.

Xiong, X., Zhao, S., Van Inwegen, E., & Beck, J. (2016). Going deeper with deep knowledge tracing. In *EDM* (pp. 545–550).

Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., & Zhu, J. (2019). Explainable ai: A brief survey on history, research areas, approaches and challenges. In *CCF international conference on natural language processing and Chinese computing* (pp. 563–574). Springer.

Yang, H., & Cheung, L.P. (2018). Implicit heterogeneous features embedding in deep knowledge tracing. *Cognitive Computation*, *10*(1), 3–14.

Yazdani, M. (1989). Intelligent tutoring systems survey. *Artificial Intelligence Review*, *1*(1), 43–52.

Yeung, C. (2019). Deep-irt: Make deep learning based knowledge tracing explainable using item response theory. In *Proceedings of educational data mining*.

Yeung, C.K., & Yeung, D.Y. (2018). Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the fifth annual ACM conference on learning at scale*. 5: ACM.

Yudelson, M.V., Koedinger, K.R., & Gordon, G.J. (2013). Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education* (pp. 171–180). Springer.

Zeiler, M.D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of european conference on computer vision* (pp. 818–833).

Zhang, J., Shi, X., King, I., & Yeung, D.Y. (2017a). Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on world wide web* (pp. 765–774).

Zhang, L., Xiong, X., Zhao, S., Botelho, A., & Heffernan, N.T. (2017b). Incorporating rich features into deep knowledge tracing. In *Proceedings of the fourth (2017) ACM conference on learning@ scale* (pp. 169–172).