Context-Aware Attentive Knowledge Tracing

Aritra Ghosh
University of Massachusetts Amherst
Amherst, MA
arighosh@cs.umass.edu

Neil Heffernan
Worcester Polytechnic Institute
Worcester, MA
nth@wpi.edu

Andrew S. Lan
University of Massachusetts Amherst
Amherst, MA
andrewlan@cs.umass.edu

ABSTRACT

Knowledge tracing (KT) refers to the problem of predicting future learner performance given their past performance in educational applications. Recent developments in KT using flexible deep neural network-based models excel at this task. However, these models often offer limited interpretability, thus making them insufficient for personalized learning, which requires using interpretable feedback and actionable recommendations to help learners achieve better learning outcomes. In this paper, we propose attentive knowledge tracing (AKT), which couples flexible attention-based neural network models with a series of novel, interpretable model components inspired by cognitive and psychometric models. AKT uses a novel monotonic attention mechanism that relates a learner's future responses to assessment questions to their past responses; attention weights are computed using exponential decay and a context-aware relative distance measure, in addition to the similarity between questions. Moreover, we use the Rasch model to regularize the concept and question embeddings; these embeddings are able to capture individual differences among questions on the same concept without using an excessive number of parameters. We conduct experiments on several real-world benchmark datasets and show that AKT outperforms existing KT methods (by up to 6% in AUC in some cases) on predicting future learner responses. We also conduct several case studies and show that AKT exhibits excellent interpretability and thus has potential for automated feedback and personalization in real-world educational settings.

ACM Reference Format:
Aritra Ghosh, Neil Heffernan, and Andrew S. Lan. 2020. Context-Aware Attentive Knowledge Tracing. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining USB Stick (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403282

1 INTRODUCTION

Recent advances in data analytics and intelligent tutoring systems [32] have enabled the collection and analysis of large-scale learner data; these advances hint at the potential of *personalized* learning at large scale, by automatically providing personalized feedback [24] and learning activity recommendations [11] to each learner by analyzing data from their learning history.

KDD '20, August 23–27, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403282

A key problem in learner data analysis is to predict future learner performance (their responses to assessment questions), given their past performance, which is referred to as the knowledge tracing (KT) problem [3]. Over the last 30 years, numerous methods for solving the KT problem were developed based on two common assumptions: i) a learner's past performance can be summarized by a set of variables representing their current latent *knowledge level* on a set of concepts/skills/knowledge components, and ii) a learner's future performance can be predicted using their current latent concept knowledge levels. Concretely, let t denote a set of discrete time indices, we have the following generic model for a learner's knowledge and performance

$$r_t \sim f(h_t), \quad h_t \sim g(h_{t-1}),$$

where $r_t \in \{0, 1\}$ denotes the learner's graded response to an assessment question at time step t, which is usually binary-valued (1 corresponds to a correct response and 0 corresponds to an incorrect one) and is observed. The latent variable h_t denotes the learner's current knowledge level and is not observed. $f(\cdot)$ and $g(\cdot)$ are functions that characterize how learner knowledge dictate their responses and how it evolves; they are sometimes referred to as the response model and the knowledge evolution model, respectively.

Earlier developments in KT methods before 2010 can be divided into two classes. The first class centered around the Bayesian knowledge tracing (BKT) method [19, 35] where knowledge (h_t) is a binary-valued scalar that characterizes whether or not a learner masters the (single) concept covered by a question. Since the response (r_t) is also binary-valued, the response and knowledge evolution models are simply noisy binary channels, parameterized by the guessing, slipping, learning, and forgetting probabilities. The second class centered around item response theory (IRT) models [16] and use these models (especially sigmoidal link functions) as the response model $f(\cdot)$; learner knowledge level is then modeled as real-valued vectors ($\mathbf{h}_t$) for questions that cover multiple concepts. Among these methods, the SPARFA-Trace method [13] used a simple affine transformation model as the explicit knowledge evolution model $g(\cdot)$. Other methods, e.g., additive factor models [1], performance factor analysis [22], the difficulty, ability, and student history (DASH) model [15], and a few recent methods including knowledge factorization machines [30] and an extension to the DASH model, the DAS3H model [2], used *hand-crafted features* such as the number of previous attempts, successes, and failures on each concept in their knowledge evolution model. Methods in both classes rely on expert labels to associate questions to concepts, resulting in excellent interpretability since they can effectively estimate the knowledge level of each learner on expert-defined concepts.

Recent developments in KT centered around using more sophisticated and flexible models to fully exploit the information contained

in large-scale learner response datasets. The deep knowledge tracing (DKT) method [23] was the first method to explore the use of (possibly deep) neural networks for KT by using long short-term memory (LSTM) networks [7] as the knowledge evolution model $g(\cdot)$. Since LSTM units are nonlinear, complex functions, they are more flexible than affine transformations and more capable of capturing nuances in real data.

The dynamic key-value memory networks (DKVMN) method extended DKT by using an external memory matrix ($\mathbf{H}_t$) to characterize learner knowledge [36]. The matrix is separated into two parts: a static, "key" matrix that contains the fixed representation of each concept, and a dynamic, "value" matrix that contains the evolving knowledge levels of each learner on each concept. DKVMN also uses separate "read" and "write" processes on this external matrix for the response and knowledge evolution models; these processes make it even more flexible than DKT. DKT and KVMN reported state-of-the-art performance in predicting future learner performance [9] and have been the benchmark for new KT methods.

The self-attentive knowledge tracing (SAKT) method [18] is the first method to use attention mechanisms in the context of KT. Attention mechanisms are more flexible than recurrent and memory-based neural networks and have demonstrated superior performance in natural language processing tasks. The basic setup of SAKT has many similarities to the Transformer model [29], an effective model for many sequence-to-sequence prediction tasks. However, we observe that SAKT does not outperform DKT and DKVMN in our experiments; see Section 4 for details. Possible reasons for this include i) unlike in language tasks where strong long-distance dependencies between words are more common, the dependence of future learner performance on the past is likely restricted to a much shorter window, and ii) the sizes of learner response datasets are several magnitudes lower than natural language datasets and are less likely to benefit from highly flexible and large-scale attention models.

More importantly, no existing KT method truly excels at both future performance prediction and interpretability. Early KT methods exhibit excellent interpretability but do not provide state-of-the-art performance on future learner performance prediction. Recent KT methods based on deep learning excel at that but offer limited interpretability. Therefore, these KT methods do not fully meet the needs of personalized learning, which requires not only accurate performance prediction but also the ability to provide automated, interpretable feedback and actionable recommendations to help learners achieve better learning outcomes.

1.1 Contributions

For the task of predicting the learner's response to the current question, we propose the attentive knowledge tracing (AKT) method, which uses a series of attention networks to draw connections between this question and every question the learner has responded to in the past. We summarize our key innovations below:

(1) Contrary to existing attention methods that use raw question and response embeddings, we put raw embeddings into context and use *context-aware representations* of past questions and responses by taking a learner's entire practice history into account.
(2) Inspired by cognitive science findings on the mechanisms of forgetting, we propose a novel *monotonic attention mechanism* that uses an exponential decay curve to down weight the importance of questions in the distant past. We also develop a context-aware measure to characterize the time distance between questions a learner has responded to in the past.
(3) Leveraging the Rasch model, a simple and interpretable IRT model, we use a series of *Rasch model-based embeddings* to capture individual differences among questions without introducing an excessive amount of model parameters.

We conduct a series of experiments on several benchmark real-world educational datasets comparing AKT to state-of-the-art KT methods. Our results show that AKT (sometimes significantly) outperforms other KT methods in predicting future learner performance. Further, we perform ablation studies on each of the key AKT model components to demonstrate their value. We also perform several case studies to show that AKT exhibits excellent interpretability and has the potential for automated feedback and practice question recommendation, both key requirements of personalized learning[1].

2 KNOWLEDGE TRACING PROBLEM SETUP

Each learner's performance record consists of a sequence of questions and responses at each discrete time step. For learner i at time step t, we denote the combination of the question that they answered, the concept this question covers, and their graded response as a tuple, (q_t^i, c_t^i, r_t^i), where $q_t^i \in \mathbb{N}^+$ is the question index, $c_t^i \in \mathbb{N}^+$ is the concept index, and $r_t^i \in \{0, 1\}$ is the response. Under this notation, $(q_t^i, c_t^i, 1)$ means learner i responded to question q_t^i on concept c_t^i correctly at time t. We note that this setup is different from some prior works on deep knowledge tracing that often ignore the question index and summarize learner performance as (c_t^i, r_t^i). This choice was made to avoid overparameterization; see Section 3.3 for a detailed analysis. In the following discussions, we omit the superscript i as we discuss how to predict future performance for a single learner. Given their past history up to time $t-1$ as $\{(q_1, c_1, r_1), \ldots, (q_{t-1}, c_{t-1}, r_{t-1})\}$, our goal is to predict their response r_t to question q_t on concept c_t at the current time step, t.

2.1 Question and Response Embeddings

Following previous work [36], we use real-valued embedding vectors $\mathbf{x}_t \in \mathbb{R}^D$ and $\mathbf{y}_t \in \mathbb{R}^D$ to represent each question and each question-response pair (q_t, r_t), respectively. $\mathbf{x}_t$ characterizes information about questions, and $\mathbf{y}_t$ characterizes the knowledge learners acquire by responding to questions, with two separate embeddings for correct and incorrect responses, respectively. D denotes the dimension of these embeddings. Therefore, let Q denote the number of questions, there are a total of Q question embedding vectors and $2Q$ question-response embedding vectors. In most real-world educational settings, the question bank is considerably larger than the set of concepts and many questions are assigned to very few learners. Therefore, the majority of existing KT methods use concepts to index questions to avoid overparameterization; all questions covering the same concept are treated as a single question. In this case, $q_t = c_t$ and $Q = C$.

[1] Source code and datasets will be available at https://github.com/arghosh/AKT

3 THE AKT METHOD

The AKT method consists of four components: two self-attentive encoders, one for questions and one for knowledge acquisition, a single attention-based knowledge retriever, and a feed-forward response prediction model; Figure 1 visualizes the AKT method and its connected components.

We use the two self-attentive encoders to learn context-aware representations of the questions and responses. We refer to the first encoder as the *question encoder*, which produces modified, contextualized representations of each question, given the sequence of questions the learner has previously practiced on. Similarly, we refer to the second encoder as the *knowledge encoder*, which produces modified, contextualized representations of the knowledge the learner acquired while responding to past questions. Alternatively, we could use raw embeddings of questions and responses similar to prior work. We found that the context-aware representation performs better in most datasets. We refer to the knowledge evolution model as the *knowledge retriever*, which retrieves knowledge acquired in the past that is relevant to the current question using an attention mechanism. Finally, the response prediction model predicts the learner's response to the current question using the retrieved knowledge. The AKT method is motivated by three intuitions rooted in cognitive science and psychometrics; we will detail these intuitions in what follows.

3.1 Context-aware Representations and The Knowledge Retriever

As introduced above, we use two encoders in our model. The question encoder takes raw question embeddings $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ as input and outputs a sequence of context-aware question embeddings $\{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_t\}$ using a monotonic attention mechanism (detailed in the next subsection). The context-aware embedding of each question depends on both itself and the past questions, i.e., $\hat{\mathbf{x}}_t = f_{\text{enc}_1}(\mathbf{x}_1, \ldots, \mathbf{x}_t)$. Similarly, the knowledge encoder takes raw question-response embeddings $\{\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}\}$ as input and outputs a sequence of actual knowledge acquired $\{\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{t-1}\}$ using the same monotonic attention mechanism. The context-aware embedding of acquired knowledge depend on the learner's response to both the current question and past questions, i.e., $\hat{\mathbf{y}}_{t-1} = f_{\text{enc}_2}(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})$.

The choice of using context-aware embeddings rather than raw embeddings reflects our first intuition: *the way a learner comprehends and learns while responding to a question depends on the learner.* These modified representations reflect each learner's actual comprehension of the question and the knowledge they actually acquire, given their personal response history. This model choice is motivated by the intuition that for two learners with different past response sequences, the way they understand the same question and the knowledge they acquire from practicing on it can differ.

The knowledge retriever takes the context-aware question and question-response pair embeddings $\hat{\mathbf{x}}_{1:t}$ and $\hat{\mathbf{y}}_{1:t-1}$ as input and outputs a retrieved knowledge state $\mathbf{h}_t$ for the current question. We note that in AKT, the learner's current knowledge state is also context-aware since it depends on the current question they are responding to; this model choice is different from that in most existing methods, including DKT. We also note that the knowledge retriever can only use information on the past questions, the learner's responses to them, and the representation of the current question, but not the learner's response to the current question, i.e., $\mathbf{h}_t = f_{\text{kr}}(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_t, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{t-1})$. The response prediction model uses the retrieved knowledge to predict the current response.

3.2 The Monotonic Attention Mechanism

We use a modified, monotonic version of the scaled dot-product attention mechanism for the encoders and the knowledge retriever. We start by briefly summarizing the original scaled dot-product attention mechanism. Under this framework, each encoder and the knowledge retriever has a key, query, and value embedding layer that maps the input into output queries, keys, and values of dimension $D_q = D_k$, D_k, and D_v, respectively. Let $\mathbf{q}_t \in \mathbb{R}^{D_k \times 1}$ denote the query corresponding to the question the learner responds to at time t, the scaled dot-product attention values $\alpha_{t,\tau}$ are computed using the softmax function [5] as

$$\alpha_{t,\tau} = \text{Softmax}(\frac{\mathbf{q}_t^\intercal \mathbf{k}_\tau}{\sqrt{D_k}}) = \frac{\exp(\frac{\mathbf{q}_t^\intercal \mathbf{k}_\tau}{\sqrt{D_k}})}{\sum_{\tau'} \exp(\frac{\mathbf{q}_t^\intercal \mathbf{k}_\tau}{\sqrt{D_k}})} \in [0, 1].$$

The output of the scaled dot-product attention mechanism is then given by $\sum_\tau \alpha_{t,\tau} \mathbf{v}_\tau \in \mathbb{R}^{D_v \times 1}$. $\mathbf{k}_\tau \in \mathbb{R}^{D_k \times 1}$ and $\mathbf{v}_\tau \in \mathbb{R}^{D_v \times 1}$ denote the key and value for the question at time step τ, respectively. Depending on the specific component, the output depends either on both the past and the current ($\tau \leq t$ for the question and knowledge encoders) or only the past ($\tau < t$ for the knowledge retriever).

Both encoders employ the self-attention mechanism, i.e., $\mathbf{q}_t$, $\mathbf{k}_t$, and $\mathbf{v}_t$ are computed using the same input; the question encoder uses $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ while the knowledge encoder uses $\{\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}\}$. The knowledge retriever, on the other hand, does not use self-attention. As shown in Fig. 1, at time step t, it uses $\hat{\mathbf{x}}_t$ (the modified embedding of the current question), $\{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_{t-1}\}$ (the context-aware embeddings of past questions), and $\{\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{t-1}\}$ (the context-aware embeddings of past question-response pairs) as input to generate the query, keys, and values, respectively. We note that SAKT uses question embeddings for mapping queries whereas response embeddings for key and value mapping. In our experiments, we have found that using question embeddings for mapping both queries and keys is much more effective.

However, this basic scaled dot-product attention mechanism is not likely going to be sufficient for KT. The reason is that learning is temporal and memories decay [21]; a learner's performance in the distant past is not as informative as recent performance when we are predicting their response to the current question. Therefore, we develop a new monotonic attention mechanism that reflects our second intuition: *when a learner faces a new question, past experiences i) on unrelated concepts and ii) that are from too long ago are not likely to be highly relevant.* Specifically, we add a multiplicative exponential decay term to the attention scores as:

$$\alpha_{t,\tau} = \frac{\exp(s_{t,\tau})}{\sum_{\tau'} \exp(s_{t,\tau'})},$$

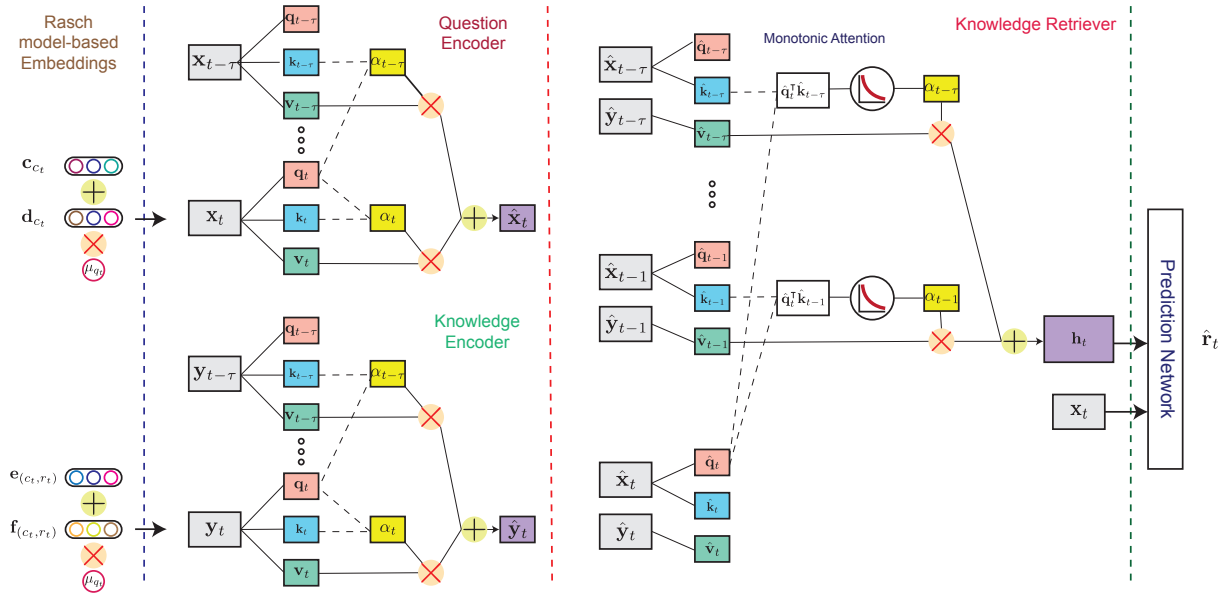

Figure 1: Overview of the AKT method. We use the Rasch model-based embeddings as raw embeddings for questions and responses. The question and knowledge encoders compute the context-aware representations of questions and responses pairs. The knowledge retriever uses these representations as input and computes the knowledge state of the learner. For simplicity, we do not show the monotonic attention mechanism in the encoders. We do not show sublayers either.

with

$$s_{t,\tau} = \frac{\exp\left(-\theta \cdot d(t,\tau)\right) \cdot \mathbf{q}_t^\intercal \mathbf{k}_\tau}{\sqrt{D_k}}, \tag{1}$$

where $\theta > 0$ is a learnable decay rate parameter and $d(t,\tau)$ is temporal distance measure between time steps t and τ. In other words, the attention weights for the current question on a past question depends not only on the similarity between the corresponding query and key, but also on the relative number of time steps between them. In summary, our monotonic attention mechanism takes the basic form of an exponential decay curve over time with possible spikes at time steps when the past question is highly similar to the current question. We note that we apply exponential decay to the attention weights rather than latent knowledge, which is the common approach in existing learner models (see e.g., [17, 26]).

We note that there are many other possible ways to characterize the temporal dynamics of attention. First, in language tasks that attention networks excel at, the temporal dynamics can be modeled using additive positional embeddings or learnable embeddings [29]. Second, in our monotonic attention mechanism, we can also parameterize the exponential decay as $s_{t,\tau} = \frac{\mathbf{q}_t^\intercal \mathbf{k}_\tau}{\sqrt{D_k}} - \theta \cdot d(t,\tau)$. However, neither of these changes lead to comparable performance to our chosen model setup; we will compare AKT against its variants using positional encoding rather than monotonic attention in our experiments.

A context-aware distance measure. The exponential decay function decides the rate at which the attention weights decay as the distance between the current time index and the previous time indices increase. A straightforward way to define the distance between two time indices is their absolute value difference, i.e., $d(t,\tau) = |t-\tau|$. However, this distance is not context-aware and ignores the practice history of each learner. For example, consider the two following sequences of concepts a learner practiced on:

$$\text{Venn Diagram (VD)}_1, \text{VD}_2, \cdots, \text{VD}_8, \text{Prime Numbers (PN)}_9, \text{PN}_{10}$$

and

$$\text{PN}_1, \text{VD}_2, \text{VD}_3, \cdots, \text{VD}_9, \text{PN}_{10},$$

where the notation "VD_2" means that the learner practiced the concept of Venn Diagram at time step 2. In this example, the learner answers a question on Prime Numbers at $t = 10$, i.e., the current time index, in both of these sequences, but the most recent past practice on Prime Numbers comes at different time indices. Since the concepts of Venn Diagram and Prime Numbers are not closely related, the learner's previous practices on Prime Numbers is more relevant to us when predicting their answer to the current practice question than recent practices on Venn Diagram. In this case, with the straightforward absolute value difference, an exponential decay curve will significantly reduce the attention weight assigned to the practice on Prime Numbers at $t = 1$.

Therefore, we propose the following context-aware distance measure between time steps $d(t,\tau)$ with $\tau \leq t$ for the exponential decay mechanism (in the encoders):

$$d(t,\tau) = |t-\tau| \cdot \sum_{t'=\tau+1}^{t} \gamma_{t,t'},$$

$$\gamma_{t,t'} = \frac{\exp\left(\frac{\mathbf{q}_t^\intercal \mathbf{k}_{t'}}{\sqrt{D_k}}\right)}{\sum_{1 \leq \tau' \leq t} \exp\left(\frac{\mathbf{q}_t^\intercal \mathbf{k}_{\tau'}}{\sqrt{D_k}}\right)}, \forall t' \leq t.$$

For the knowledge retriever, we replace $\tau' \leq t$ with $\tau < t$ and $t' \leq t$ with $t' < t$ correspondingly. In other words, this context-aware distance measure uses another softmax function to adjust the distance between consecutive time indices according to how the concept practiced in the past is related to the current concept. In practice, in each iteration during model training, we use the current AKT model parameters to compute the modified distance measure and fix it; we do not pass gradients through the distance measure.

Multi-head attention and sub-layers. We also incorporate multi-head attention which is effective in attending to past positions at multiple time scales [29]. Therefore, we use H independent attention heads where every head has its own decay rate θ, concatenate the final output into a $(D_v \cdot H)\times 1$ vector and pass it to the next layer. This model design enables AKT to summarize past learner performance at multiple time scales, which bears some similarities to the multiple time windows in the multiscale context, DASH, and DAS3H models [2, 15, 21]. We also use several sub-layers, including one for layer normalization [14], one for dropout [27], a fully-connected feedforward layer, and a residual connection layer [6] in each encoder and the knowledge retriever.

3.3 Response Prediction

The last component of the AKT method predicts the learner's response to the current question. The input to the prediction model is a vector that concatenates both the retrieved knowledge (the knowledge retriever output $\mathbf{h}_t$) and the current question embedding $\mathbf{x}_t$; this input goes through another fully-connected network before finally going through the sigmoid function [5] to generate the predicted probability $\hat{r}_t \in [0, 1]$ that the learner answers the current question correctly. All learnable parameters in the entire AKT method are trained in end-to-end fashion by minimizing the binary cross-entropy loss of all learner responses, i.e.,

$$\ell = \sum_i \sum_t -(r_t^i \log \hat{r}_t^i + (1 - r_t^i) \log(1 - \hat{r}_t^i)).$$

3.4 Rasch Model-Based Embeddings

As we discussed above, existing KT methods use concepts to index questions, i.e., setting $q_t = c_t$. This setup is necessary due to data sparsity. Let Q denote the total number of questions and L denote the number of learners. In most real-world learner response datasets, the number of learner responses is comparable to CL and much less than QL since many questions are assigned to few learners. Therefore, using concepts to index questions is effective in avoiding overparameterization and overfitting. However, this basic setup ignores the individual differences among questions covering the same concept, thus limiting the flexibility of KT methods and their potential for personalization.

We use a classic yet powerful model in psychometrics, the Rasch model (which is also known as the 1PL IRT model) [16, 25], to construct raw question and knowledge embeddings. The Rasch model characterizes the probability that a learner answers a question correctly using two scalars: the question's difficulty, and the learner's ability. Despite its simplicity, it has shown to achieve comparable performance to more sophisticated models on learner performance prediction in formal assessments when knowledge is static [12, 31]. Specifically, we construct the embedding of the question q_t from concept c_t at time step t as

$$\mathbf{x}_t = \mathbf{c}_{c_t} + \mu_{q_t} \cdot \mathbf{d}_{c_t},$$

where $\mathbf{c}_{c_t} \in \mathbb{R}^D$ is the embedding of the concept this question covers, and $\mathbf{d}_{c_t} \in \mathbb{R}^D$ is a vector that summarizes the variation in questions covering this concept, and $\mu_{q_t} \in \mathbb{R}$ is a scalar *difficulty* parameter that controls how far this question deviates from the concept it covers. The question-response pairs (q_t, r_t) from concept c_t are extended similarly using the scalar difficulty parameter for each pair:

$$\mathbf{y}_t = \mathbf{e}_{(c_t, r_t)} + \mu_{q_t} \cdot \mathbf{f}_{(c_t, r_t)},$$

where $\mathbf{e}_{(c_t, r_t)} \in \mathbb{R}^D$ and $\mathbf{f}_{(c_t, r_t)} \in \mathbb{R}^D$ are concept-response embedding and variation vectors. This model choice reflects our third intuition: *questions labeled as covering the same concept are closely related but have important individual differences that should not be ignored.* This model choice is partly inspired by another work in fusing KT and IRT models [8].

These Rasch model-based embeddings strike the right balance between modeling individual question differences and avoiding overparameterization. For the question embeddings, the total number of embedding parameters in this model is $2CD + Q$, which is slightly more than that in a model that uses concepts to index questions (CD), but much less than that in a model where each question is parameterized individually (QD), since $C \ll Q$ and $D \gg 1$. We further define the concept-response embeddings as $\mathbf{e}_{(c_t, r_t)} = \mathbf{c}_{c_t} + \mathbf{g}_{r_t}$, where $\mathbf{g}_1$ and $\mathbf{g}_0$ denote the embeddings for correct and incorrect responses (regardless of the concept), respectively. Therefore, we only introduce a total of $(C + 2)D + Q$ new embedding parameters instead of $2CD + Q$ new parameters for the concept-response embeddings. We note that our question and question-response embeddings share a group of parameters ($\mathbf{c}_{c_t}$); this setting is different from existing neural network-based KT methods where the two are independent of each other. These compact embedding representations significantly reduce the number of parameters in not only AKT but also some other KT methods, leading to improved performance on future learner performance prediction; see Table 5 for details.

Dataset	Learners	Concepts	Questions	Responses
Statics2011	333	1, 223	-	189, 297
ASSISTments2009	4, 151	110	16, 891	325, 637
ASSISTments2015	19, 840	100	-	683, 801
ASSISTments2017	1, 709	102	3, 162	942, 816

Table 1: Dataset statistics.

4 EXPERIMENTAL RESULTS

In this section, we detail a series of experiments we conducted to test on several real-world datasets. We evaluate AKT both quantitatively through predicting future learner responses and qualitatively through a series of visualizations and case studies.

4.1 Experimental Setup

Datasets. We evaluate the performance of AKT and several baselines on predicting future learner responses using four benchmark

Dataset	BKT+	DKT	DKT+	DKVMN	SAKT	AKT-NR	AKT-R
Statics2011	∼ 0.75	0.8233 ± 0.0039	**0.8301 ± 0.0039**	0.8195 ± 0.0041	0.8029 ± 0.0032	*0.8265 ± 0.0049*	
ASSISTments2009	∼ 0.69	*0.817 ± 0.0043*	0.8024 ± 0.0045	0.8093 ± 0.0044	0.752 ± 0.004	0.8169 ± 0.0045	**0.8346 ± 0.0036**
ASSISTments2015		0.731 ± 0.0018	*0.7313 ± 0.0018*	0.7276 ± 0.0017	0.7212 ± 0.002	**0.7828 ± 0.0019**	
ASSISTments2017		0.7263 ± 0.0054	0.7124 ± 0.0041	0.7073 ± 0.0044	0.6569 ± 0.0027	*0.7282 ± 0.0037*	**0.7702 ± 0.0026**

Table 2: Performance of all KT methods on all datasets in predicting future learner responses. AKT (sometimes significantly) outperforms all baseline methods on all datasets. Best models are bold, second best models are italic.

datasets: ASSISTments2009, ASSISTments2015, ASSISTments2017[2], and Statics2011.[3] The ASSISTments datasets were collected from an online tutoring platform; in particular, the ASSISTments2009 dataset has been the standard benchmark for KT methods over the last decade. The Statics2011 dataset was collected from a college-level engineering course on statics. On all these datasets, we follow a series of standard pre-processing steps in the literature. For the ASSISTments2009 dataset, we remove all interactions that are not associated to a named concept. For the ASSISTments2015 dataset, we remove all interactions where the "isCorrect" field is not 0 or 1. We list the numbers of learners, concepts, questions, and question-response pairs in Table 1. Out of these datasets, only the ASSISTments2009 and ASSISTments2017 datasets contain question IDs; therefore, the Rasch model-based embeddings are only applicable to these two datasets.

Baseline methods and evaluation metric. We compare AKT against several baseline KT methods, including BKT+ [35], DKT, DKT+ (which is an improved version of DKT with regularization on prediction consistency [34]), DKVMN [36], and the recently proposed self-attentive KT (SAKT) method [18], which uses an attention mechanism that can be viewed as a special case of AKT without context-aware representations of questions and responses and the monotonic attention mechanism. We use the area under the receiver operating characteristics curve (AUC) as the metric to evaluate the performance of all KT methods on predicting binary-valued future learner responses to questions.

Training and testing. For evaluation purposes, we perform standard k-fold cross-validation (with $k = 5$) for all models and all datasets. Thus, for each fold, 20% learners are used as the test set, 20% are used as the validation set, and 60% are used as the training set. For each fold, we use the validation set to perform early stopping and tune the parameters for every KT method.

We truncate learner response sequences that are longer than 200, following [23, 36], for computational efficiency reasons. If a learner has more than 200 responses, we break up their entire sequence into multiple shorter sequences. We use the Adam optimizer to train all models [10] with a batch size of 24 learners to ensure that an entire batch can fit into the memory of our machine (equipped with one NVIDIA Titan X GPU). We implement all versions of AKT in `PyTorch`; We also re-implement DKT, DKT+, and SAKT, since including question IDs requires new dataset partitions and leads to new experimental results. We use the Xavier parameter initialization method [4] for AKT, DKT, DKT+, and SAKT; for DKVMN, we follow their work and use samples from normal distributions to initialize the parameters [36]. We do not re-implement BKT+; its performance on various datasets is taken from [36]. For most datasets and most algorithms, one training epoch takes less than 10 seconds. We set the maximum number of epochs to 300.

[2]The ASSISTments datasets are retrieved from https://sites.google.com/site/assistmentsdata/home and https://sites.google.com/view/assistmentsdatamining/.
[3]The Statics2011 dataset is retrieved from https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507.

Dataset	AKT$^{\text{raw}}$-NR	AKT-NR	AKT$^{\text{raw}}$-R	AKT-R
Statics2011	0.8253	**0.8265**		
ASSISTments2009	0.8082	**0.8169**	0.8267	**0.8346**
ASSISTments2015	0.7332	**0.7828**		
ASSISTments2017	0.7066	**0.7282**	0.7552	**0.7702**

Table 3: AKT outperforms its variants that do not use contextual-aware question and response representations.

Dataset	SAKT	AKT-NR$^{\text{pos}}$	AKT-NR$^{\text{fixed}}$	AKT-NR
Statics2011	0.8029	0.8196	0.8196	**0.8265**
ASSISTments2009	0.752	0.7706	0.7708	**0.8169**
ASSISTments2015	0.7212	0.7271	0.7272	**0.7828**
ASSISTments2017	0.6569	0.672	0.6722	**0.7282**

Table 4: AKT significantly outperforms its variants that do not use monotonic attention.

4.2 Results and Discussion

Table 2 lists the performance of all KT methods across all datasets on predicting future learner responses; we report the averages as well as the standard deviations across five test folds. AKT-R and AKT-NR represent variants of the AKT model with and without the Rasch model-based embeddings, respectively. We see that AKT (sometimes significantly) outperforms other KT methods on the ASSISTments datasets while DKT+ marginally outperforms AKT on the smallest Statics2011 dataset. In general, AKT performs better on larger datasets; this result suggests that attention mechanisms are more flexible than recurrent neural networks and are thus more capable of capturing the rich information contained in large-scale real-world learner response datasets. On the ASSISTments2015 and ASSISTments2017 datasets, AKT-NR improves the AUC by 6% and 1% over the closest baseline. It performs on-par with the best-performing baseline on the Statics2011 and ASSISTments2009 datasets. More importantly, on the ASSISTments2009 and 2017 datasets with question IDs, AKT-R significantly outperforms other KT methods, by 2% and 6% over the closest baseline, respectively. We note that DKT outperforms the more advanced DKVMN method in our implementation. While we are able to replicate the performance of DKVMN using the same experimental setting [36], we found that

Dataset	DKT	DKT-R	DKT+	DKT+-R	DKVMN	DKVMN-R	SAKT	SAKT-R	AKT-NR	AKT-R
ASSISTments2009	0.817	0.8179	0.8024	0.8033	0.8093	0.8235	0.752	0.7784	0.8169	**0.8346**
ASSISTments2017	0.7263	0.7543	0.7124	0.7382	0.7073	0.7628	0.6569	0.7137	0.7282	**0.7702**

Table 5: The Rasch model-based embeddings (sometimes significantly) improve the performance of KT methods.

DKT performs much better than previously reported in that work. DKT+ performs on-par with DKT, with minor improvements on the Statics2011 dataset. We also observe that the RNN-based model, DKT, outperforms SAKT on all datasets.

Ablation study. In order to justify the three key innovations in the AKT method, context-aware representations of questions and responses, the monotonic attention mechanism, and the Rasch model-based embeddings, we perform three additional ablation experiments comparing several variants of the AKT method. The first experiment compares AKT-NR and AKT-R using context-aware question and response representations (with the question and knowledge encoders) with two variants AKT$^{\text{raw}}$-NR and AKT$^{\text{raw}}$-R; In these variants, we use raw question and response embeddings as their representations instead of the context-aware representations (i.e., without passing them through the encoders). The second experiment compares AKT-NR against several variants without the monotonic attention mechanism. These variants include AKT-NR$^{\text{pos}}$, which uses (learnable) positional encoding to capture temporal dependencies in learner response data and AKT-NR$^{\text{fixed}}$, which uses (fixed) positional encoding using different frequencies of sine and cosine functions [29]. The third experiment compares AKT-R with AKT-NR, DKT, DKT-R, DKT+, DKT+-R, DKVMN, DKVMN-R, SAKT, and SAKT-R on the ASSISTments2009 and 2017 datasets where question IDs are available; DKT-R, DKT+-R, DKVMN-R, and SAKT-R refer to the DKT, DKT+, DKVMN, and SAKT methods augmented with the Rasch model-based embeddings as input, respectively.

Table 3 shows the results (only averages and not standard deviations across test folds, due to spatial constraints) of the first ablation experiment for the context-aware representations (i.e., the question and knowledge encoders). On all datasets AKT-R and AKT-NR outperform their counterparts, AKT$^{\text{raw}}$-NR and AKT$^{\text{raw}}$-R, which use only a single self-attention mechanism with exponential decay (i.e., the knowledge retriever). These results suggest that our context-aware representations of questions and responses are effective at summarizing each learner's practice history.

Table 4 shows the results of the second ablation experiment for the monotonic attention mechanism. We see that AKT-NR significantly outperforms other attention mechanisms using positional embeddings, including SAKT, by about 1% to 6% on all datasets. We postulate that the reason for this result is that unlike in language tasks where strong long-distance dependencies between words are more common, the dependence of future learner performance on the past is restricted to a much shorter time window. Therefore, using multi-head attention with different exponential decay rates in the attention weights can effectively capture short-term dependencies on the past at different time scales.

Table 5 shows the results of the third ablation experiment for the Rasch model-based embeddings on the two ASSISTments datasets where question IDs are available. All baseline KT methods with the added Rasch model-based embeddings outperform their regular versions, especially on the ASSISTments2017 dataset. These results confirm our intuition that treating all questions covering the same concept as a single question is problematic; individual differences among these questions should not be ignored as long as overparameterization can be avoided.

DKT	DKT+	DKVMN	SAKT	AKT-NR	AKT-R
0.7616	0.7552	0.7556	0.7432	*0.7627*	**0.7866**

Table 6: AKT still outperforms other KT methods on the ASSISTments2009 dataset under an alternative experimental setting for questions tagged with multiple concepts.

Remark. Our standard experimental setting follows that used in [23, 36]. In this setting, for questions tagged with multiple concepts (in the ASSISTments2009 dataset), a single learner response is repeated multiple times, one for each concept. Other works used different experimental settings for these questions; In [31], the authors removed such questions and as a result, DKT's performance dropped to 0.71. In [33], the authors built new concepts for each combination of co-occurring single concepts and as a result, DKT's performance dropped to 0.73. Therefore, we also use an alternative experimental setting on the ASSISTments2009 dataset. For a question tagged with multiple concepts, we average the corresponding concept embeddings and use them as both input embeddings and for response prediction. Table 6 lists the performance of all KT methods on the ASSISTments2009 dataset under this setting. DKT's performance dropped to 0.76 using average embeddings, faring better than settings under [31, 33]. We observe similar performance drops compared to our standard experimental setting for all KT methods, while AKT-R still comfortably outperforms all baselines.

4.3 Visualizing Learned AKT Parameters

Monotonic attention. Figure 2 shows the interpretability offered by AKT's monotonic attention mechanism using the ASSISTments2009 dataset. Figure 2(a) visualizes the attention weights in the knowledge retriever for one learner as an example; we plot the attention weights used to predict their performance on 20 consecutive practice questions across three attention heads. We see that each attention head operates on its own time scale: they all have attention windows of different widths. For example, the second head is capable of attending to the entire past, up to 20 time steps (in this example); on the contrary, the third head can only attend to the immediate past and focuses primarily on the last 3-5 time steps. This observation suggests that some questions and responses in the past contain information that is highly predictive of the learner's response to the current question; this information can be effectively captured by multiple attention heads with different decay rates.

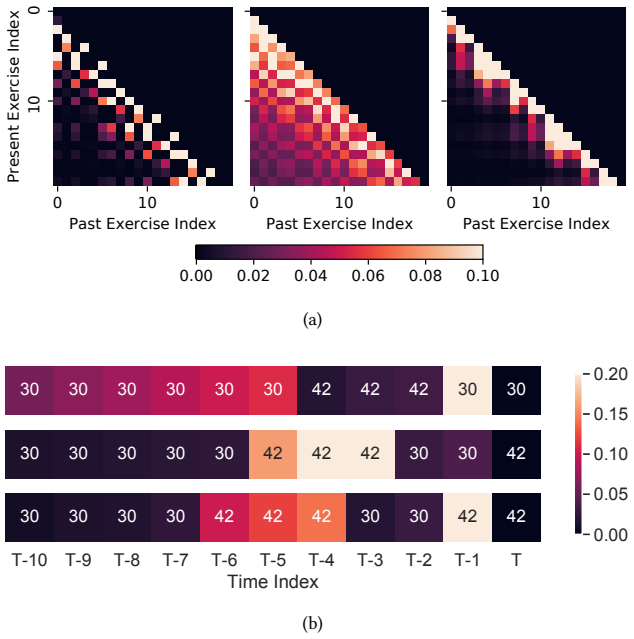

Figure 2: Visualizations of (a) attention weights in the decoder of AKT for three attention heads and (b) attention weights for three consecutive practice questions for a learner. Concept similarity and recency are key factor that control the attention weights.

Figure 2(b) visualizes the normalized attention weights in the knowledge retriever for a single learner for three consecutive time steps. In the top row, the learner is responding to a question on Concept 30 at time T after practicing this concept from $T-10$ to $T-5$, then taking a break to practice on Concept 42, before coming back to Concept 30 at time $T-1$. We see that AKT predicts their response to the current question by focusing more on previous practices on this concept (both in the immediate past and further back) than practices on another concept also in the immediate past. In the middle row, the learner switches to practicing on Concept 42 again. Again, AKT learns to focus on past practices on the same concept rather than the immediate past on a different concept at times $T-2$ and $T-1$. In the bottom row, the learner practices on Concept 42 for the second consecutive time, and AKT shows a similar focus pattern to that in the top row, with the roles of Concepts 30 and 42 swapped. These observations suggest that AKT's monotonic attention mechanism has the potential to provide feedback to teachers by linking a learner's current response to their responses in the past; this information may enable teachers to select certain questions that they have practiced for them to re-practice and clear misconceptions before moving on. We also note that AKT, using a data-driven approach, learns these attention patterns that match hand-crafted features in existing KT methods (e.g., the number of total attempts and correct attempts on this concept) [15, 22].

Rasch model-based embeddings. Figure 3 visualizes the learned Rasch model-based question embeddings for several concepts using t-SNE [28] using the ASSISTments2009 dataset, together with their empirical difficulties for selected questions (portions of correct responses across learners). We also highlight the hardest

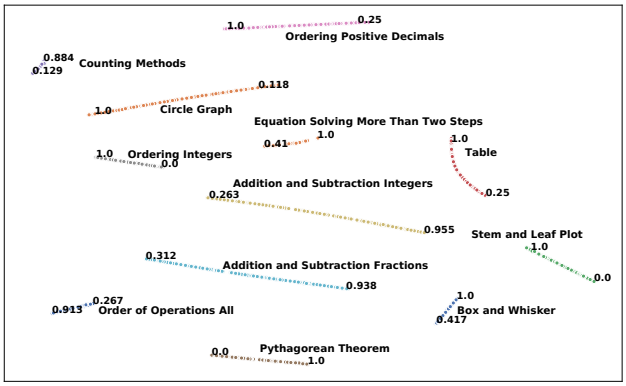

Figure 3: Visualization of learned question embeddings with fraction of correct responses among learners for selected concepts.

and easiest question for each concept based on their empirical difficulties. We see that questions on the same concept form a curve and are ordered by their difficulty levels: for the majority of concepts, questions on one end of the line segment are easy while questions on the other end are hard. This result confirms our intuition that questions from the same concept are not identical but closely related to each other; this relationship can be well-captured by the Rasch model using its difficulty parameter.

Concept	Question	μ_q
Ordering Positive Decimals	From the list of numbers below, which number is the largest? 6.7, 6.4, 3.4, 5.1	-0.0397
	Which of following decimals is the smallest? 0.107, 0.1889, 0.12, 0.11582	0.0090
	Arrange these numbers from least to greatest. -1/4, 12.1, -1.4, -4/4	0.0279
Probability of a Single Event	Steve has a marble jar, that he likes to randomly select marbles from it to play with. The jar has 5 orange marbles and 5 purple marbles. What is the probability that Steve gets an orange marble from the jar?	−0.0515
	A bag contains 8 red, 5 green, and 7 blue popsicles. John is going to draw out a popsicle without looking in the bag. What is the probability that he will draw either a green or a blue popsicle?	0.0088
	A card is selected at random from a standard deck of 52 cards. Find the probability of choosing a club or an ace card. Enter your answer as a fraction.	0.0548
Conversion of Fractions	Convert 7/7 into a percent.	-0.0540
	Convert 8/4 into a percent.	0.0038
	Convert 9/8 into a percent.	0.0529

Table 7: Question text and learned difficulty parameters (μ_q) for selected questions on three concepts. Learned difficulty levels match our intuition on the difficulty of these questions.

Table 7 lists sample questions each for three different concepts, "Ordering Positive Decimals", "Probability of a Single Event", and

"Conversion of Fractions to Percents", and their learned difficulty parameters. We show three questions for each concept: an easy one, an average one, and a hard one. Using the "Probability of a Single Event" concept as an example, the learned difficulty parameter values (μ_q) are -0.0515 for the easy one, 0.0088 for the average one, and 0.0548 for the hard one. These learned difficulty levels match our understanding about the difficulty levels of these questions.

These results suggest that AKT has the potential to be applied in real-world educational settings. Using the estimated difficulty parameters, a computerized learning platform can either i) automatically select questions with appropriate difficulty levels for each learner given their past responses, or ii) support teachers to adjust course plans by providing them feedback on question difficulty levels learned from real data. Therefore, AKT improves over existing KT methods by not only providing state-of-the-art predictive performance but also exhibiting interpretability and potential for personalized learning.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed attentive knowledge tracing, a new method for knowledge tracing that relies fully on attention networks. Our method improves upon existing knowledge tracing methods by building context-aware representations of questions and responses, using a monotonic attention mechanism to summarize past learner performance in the right time scale, and by using the Rasch model to capture individual differences among questions covering the same concept. Experimental results on a series of benchmark real-world learner response datasets show that our method outperforms state-of-the-art KT methods and exhibit excellent interpretability. Avenues of future work include i) incorporating question text to further enhance the interpretability of question and concept embeddings and ii) testing whether our method can improve prediction performance on language learning datasets where memory decay occurs [26].

REFERENCES

[1] Hao Cen, Kenneth Koedinger, and Brian Junker. 2006. Learning factors analysis–A general method for cognitive model evaluation and improvement. In *Proc. International Conference on Intelligent Tutoring Systems*. 164–175.

[2] Benoît Choffin, Fabrice Popineau, Yolaine Bourda, and Jill-Jênn Vie. 2019. DAS3H: Modeling student learning and forgetting for pptimally scheduling distributed practice of skills. In *Proc. International Conference on Educational Data Mining*. 29–38.

[3] Albert Corbett and John Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-adapted Interaction* 4, 4 (Dec. 1994), 253–278.

[4] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. International Conference on Artificial Intelligence and Statistics*. 249–256.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780.

[8] MM Khajah, Y Huang, JP González-Brenes, MC Mozer, and P Brusilovsky. 2014. Integrating knowledge tracing and item response theory: A tale of two frameworks. In *Proc. International Workshop on Personalization Approaches in Learning Environments*, Vol. 1181. 7–15.

[9] Mohammad Khajah, Robert Lindsey, and Michael Mozer. 2016. How deep is knowledge tracing?. In *Proc. International Conference on Educational Data Mining*. 94–101.

[10] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations*.

[11] Andrew Lan and Richard Baraniuk. 2016. A Contextual Bandits Framework for Personalized Learning Action Selection. In *Proc. International Conference on Educational Data Mining*. 424–429.

[12] Andrew Lan, Tom Goldstein, Richard Baraniuk, and Christoph Studer. 2016. Dealbreaker: A nonlinear latent variable model for educational data. In *Proc. International Conference on Machine Learning*. 266–275.

[13] Andrew Lan, Christoph Studer, and Richard Baraniuk. 2014. Time-varying learning and content analytics via sparse factor analysis. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 452–461.

[14] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (July 2016).

[15] Robert Lindsey, Jeffery Shroyer, Harold Pashler, and Michael Mozer. 2014. Improving students' long-term knowledge retention through personalized review. *Psychological Science* 25, 3 (Jan. 2014), 639–647.

[16] Frederick Lord. 1980. *Applications of Item Response Theory to Practical Testing Problems*. Erlbaum Associates.

[17] Michael Mozer, Denis Kazakov, and Robert Lindsey. 2017. Discrete-event continuous-time recurrent nets. *arXiv preprint arXiv:1710.04110* (Oct. 2017).

[18] Shalini Pandey and George Karypis. 2019. A self attentive model for knowledge tracing. In *Proc. International Conference on Educational Data Mining*. 384–389.

[19] Zachary Pardos and Neil Heffernan. 2010. Modeling individualization in a Bayesian networks implementation of knowledge tracing. In *Proc. International Conference on User Modeling, Adaptation, and Personalization*. 255–266.

[20] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proc. International Conference on Machine Learning*. 1310–1318.

[21] Harold Pashler, Nicholas Cepeda, Robert Lindsey, Ed Vul, and Michael Mozer. 2009. Predicting the optimal spacing of study: A multiscale context model of memory. In *Proc. Conference on Advances in Neural Information Processing Systems*. 1321–1329.

[22] Philip Pavlik Jr, Hao Cen, and Kenneth Koedinger. 2009. Performance factors analysis–A new alternative to knowledge tracing. In *Proc. International Conference on Artificial Intelligence in Education*. 531–538.

[23] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Proc. Conference on Advances in Neural Information Processing Systems*. 505–513.

[24] Chris Piech, Jonathan Huang, Andy Nguyen, Mike Phulsuksombati, Mehran Sahami, and Leonidas Guibas. 2015. Learning Program Embeddings to Propagate Feedback on Student Code. In *Proc. International Conference on Machine Learning*. 1093–1102.

[25] Georg Rasch. 1993. *Probabilistic Models for Some Intelligence and Attainment Tests*. MESA Press.

[26] Siddharth Reddy, Igor Labutov, Siddhartha Banerjee, and Thorsten Joachims. 2016. Unbounded human learning: Optimal scheduling for spaced repetition. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1815–1824.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (June 2014), 1929–1958.

[28] Laurens Van Der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research* 15, 1 (2014), 3221–3245.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. Conference on Advances in Neural Information Processing Systems*. 5998–6008.

[30] Jill-Jênn Vie and Hisashi Kashima. 2019. Knowledge tracing machines: Factorization machines for knowledge tracing. In *Proc. AAAI Conference on Artificial Intelligence*, Vol. 33. 750–757.

[31] Kevin Wilson, Yan Karklin, Bojian Han, and Chaitanya Ekanadham. 2016. Back to the basics: Bayesian extensions of IRT outperform neural networks for proficiency estimation. In *Proc. International Conference on Educational Data Mining*. 539–544.

[32] Beverly Park Woolf. 2010. *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-learning*. Morgan Kaufmann.

[33] Xiaolu Xiong, Siyuan Zhao, Eric G Van Inwegen, and Joseph E Beck. 2016. Going deeper with deep knowledge tracing. *International Educational Data Mining Society* (2016).

[34] Chun-Kit Yeung and Dit-Yan Yeung. 2018. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proc. ACM Conference on Learning at Scale*. ACM, 5.

[35] Michael Yudelson, Kenneth Koedinger, and Geoffrey Gordon. 2013. Individualized Bayesian knowledge tracing models. In *Proc. International Conference on Artificial Intelligence in Education*. 171–180.

[36] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. 2017. Dynamic key-value memory networks for knowledge tracing. In *Proc. International Conference on World Wide Web*. 765–774.

Supplementary Material

Parameter tuning. For all methods, we use a two-layer, fully-connected network for the response prediction model, with 512 and 256 hidden units in the first and second layers, respectively. For AKT, we fix the number of attention heads to $H = 8$ and use the same dimension for the queries, keys, and values for each head in the encoders and the knowledge retriever, i.e., $D_k = D_v = D/8$ where D is the input embedding dimension. For AKT, we share the query and key embedding layer in the attention mechanism. We do not perform grid search over the values of these parameters since the performance of KT methods is insensitive to these parameters. For all methods, we use $\{256, 512\}$, $\{256, 512\}$, and $\{0, 0.05, 0.1, 0.15, 0.2, 0.25\}$ as values of the input embedding dimension, the hidden state dimension, and the dropout rate for the feedforward network, respectively. We also use $\{1, 10, \infty\}$ as values of the maximum gradient norm for clipping [20] where ∞ means no gradient norm clipping. For AKT, SAKT, and their variants, we use $\{5\times10^{-6}, 10^{-5}, 10^{-4}\}$ as values of the learning rate in the Adam optimizer. For DKT, DKT+, and DKVMN, we use $\{10^{-4}, 10^{-3}\}$ as values of the learning rate. Additionally, for DKVMN, we use $\{20, 50\}$ as values of the memory size parameter, following [36]. For DKT+, we use $\{0.1, 0.2\}$, $\{0.3, 1\}$, and $\{3, 30\}$ as values of the reconstruction regularization, the ℓ_1-norm penalty, and the ℓ_2-norm penalty parameters, respectively, following [34].