Does chronology matter? Sequential vs contextual approaches to knowledge tracing

Yueqi Wang
University of California, Berkeley
yueqi@berkeley.edu

Zachary Pardos
University of California, Berkeley
pardos@berkeley.edu

ABSTRACT

Deep learning architectures such as RNN and pure-attention based models have shown state-of-the-art performance in modeling student performance, yet the sources of the predictive power of such models remain an open question. In this paper, we investigate the predictive power of aspects of LSTM and pure attention-based architectures that model sequentiality. We design a knowledge tracing model based on a general transformer encoder architecture to explore the predictive power of sequentiality for attention-based models. For the LSTM-based Deep Knowledge Tracing model, we manipulate the state transition coefficient matrix to turn sequential modeling on and off. All models are evaluated on four public tutoring datasets from ASSISTments and Cognitive Tutor. Experimental results show that DKT and pure-attention based models are overall insensitive towards removing major sequential signals by disabling their sequential modeling parts but with the attention-based model about four times more sensitive. Lastly, we shed light on benefits and challenges of sequential modeling in student performance prediction.

Keywords

Knowledge tracing, DKT, BKT, Cognitive Tutor, ASSISTments, KDD Cup, Contextual embedding, Self-attention

1. INTRODUCTION

Bayesian and Deep Knowledge Tracing models use machine learning architectures with explicit representation of time slices that capture sequentiality. A new generation of models have emerged based on self-attention that have outperformed both BKT and DKT on the knowledge tracing task [9, 11, 19, 17, 7]. Interestingly, the transformer-based architecture that these models rely on has relatively weak modeling of the concept of a time-slice and subsequently of sequence. This prompts us to ask the question of what role sequentiality plays in the predictive performance of the latest generations of neural knowledge tracing models. Some of the better performing self-attention models adopt a combination of transformer and LSTM architectures, raising the question of if the LSTM is superior for capturing sequence-based signals. To investigate the role of sequentiality in the predictive power of the RNN-based DKT and transformer-based self-attention, or "contextual" models, we design experiments that compare two variants of each model; the original, a version in which time-components of the architecture are systematically disabled. Using four public datasets, we report each model's sensitivity to removing signals of sequence. Our major contributions in this work are:

1. A simple, yet theoretically effective way of disabling attention-based models' sequential modeling components.
2. A novel way of insulating DKT's sequential modeling components to study it's sequential modeling ability.
3. We find that DKT and BertKT are exceptionally robust to losing sequential modeling components in terms of performance.

2. RELATED WORK

For studying sequence effects for student performance prediction, Ding and Larson [4] showed that relative ordering of skills have negative effects on the performance of DKT. Kim et al. [10] studied DKT's behavior by conducting a series of perturbation experiments on Monotonicity, Robustness and Convergence etc. Work prior to DKT has also studied sequencing effects, Pardos and Heffernan [13] used a modification of the BKT model to capture learning rates of ordered item pairs, finding that while statistically separable learning rates could be fit, this extra modeling of order did not significantly improve performance prediction accuracy.

Pure-attention based models such as Transformer [18] and Bidirectional Encoder Representations from Transformers (BERT) [3] were first introduced in the NLP field and achieved state of the art performance on major NLP tasks. These models, compared to LSTM predecessors, have the added advantage of enabling parallel training for computational efficiency [18] [2] and being better able to capture long term dependencies. Adapting these strengths to the knowledge tracing task, Self Attentive Knowledge Tracing (SAKT) [12] introduces the earlier attention-based models to solve student performance prediction. SAINT+ [14], which leverages various input features embeddings to assist tutoring on massive online learning platforms, and AKT [7] further improves

Y. Wang and Z. Pardos. Does chronology matter? Sequential vs contextual approaches to knowledge tracing. In A. Mitrovic and N. Bosch, editors, *Proceedings of the 15th International Conference on Educational Data Mining*, pages 601–605, Durham, United Kingdom, July 2022. International Educational Data Mining Society.

https://doi.org/10.5281/zenodo.6852966

the attention-based architecture by introducing monotonic attention and context-aware distance measure.

Hybrid sequential-contextual models have shown success. *Last Query transformer RNN model* [9] LSTM-SAKT [11] and MUSE [19] utilize the combination of transformer encoder and LSTM/GRU and achieved top-5 performance in a Kaggle AIEd Prediction Challenge 2020[1].

3. METHODOLOGY

3.1 Problem Restatement

Our study on sequential effects is based on the task of student performance prediction. We define this task as follows.

Definition 1. Given a student s's past interactive sequence from time step 1 to $t-1$ that contains problem marked by required skill ids $X_{t-1}^{(s)}$ and their correctness of responses $Y_{t-1}^{(s)}$, performance prediction of problem x of student s at time step t is defined as:

$$\hat{y}_t^{(s)} = f(X_{t-1}^{(s)}, Y_{t-1}^{(s)}, x_t^{(s)})$$

where $X_{t-1}^{(s)} = \{x_i^{(s)} \in \mathbb{Z}^+ | 1 \leqslant i \leqslant t-1, i \in \mathbb{Z}^+\}$ and $Y_{t-1}^{(s)} = \{y_i^{(s)} \in \mathbb{Z}^+ | 1 \leqslant i \leqslant t-1, i \in \mathbb{Z}^+\}$

3.2 The BertKT Architecture

To study the positional effect of transformer family of models, we design a simple, general architecture named **B**idirectional **e**ncoder **r**epresentations from **t**ransformers for **K**nowledge **T**racing, or **BertKT**. We design this model instead of utilizing previously proposed models for several reasons:

1. It does not contain redundant structures other than transformer encoder layers and an output layer for our downstream task. This helps us isolate the impact of sequential effects for attention-based models.
2. It adopts the most widely used sine-cosine positional encoding with easy enabling/disabling, which facilitates generalization of our results.
3. This architecture is flexible due to its simplicity to accommodate additional settings for future research.

In this work, we conduct student performance prediction in an auto-regressive fashion supported by an upper-triangular attention masks. We do not utilize the bi-directionality here, but such functionality could be enabled for future use such as for bi-directional pre-training tasks. Thus, we keep here the naming convention "bi-directional" as is.

3.2.1 Knowledge and Positional Encoding

In transformer models, scaled-dot-product self-attention which runs in $O(n^2)$ [18]. Thus we adopt word embedding [1] instead of one-hot encoding in the field of NLP to obtain a learnable knowledge encoding with higher computational efficiency. In this work, we only use skill and responses as input features. Embeddings of such features are element-wise-summed together to form a unified representation named combined embedding.

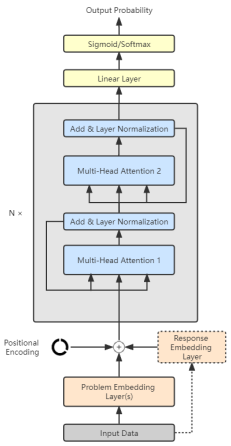

Figure 1: BertKT Architecture Visualization. The shaded part is the self-attention based encoding method.

The transformer model is permutation-invariant [18, 5] without positional encoding, making it suffer from not being able to capture original sequence order. A wide varieties of positional encodings are introduced for better performance or interpretability[8, 7]. To study the effect of positionality in this type of model, we completely remove the sine-cosine positional encoding of our model introduced in section 3.2 and compare the model's performance with the baseline model (positional encoding on).

3.2.2 BertKT encoder

In section 3.2.1, we have introduced our element-wise-summed embedding. On top of that, we add a canonical transformer encoder to form the integrated BertKT Architecture. For computation flow, the skill embedding layer and the response embedding layer first take in skill ids and responses to compute their raw embeddings. These two embeddings together with positional encoding are summed element-wise to form a combined embedding. Secondly, the combined embedding goes through BertKT encoder to learn rich contextualized representations. Such contextualized embedding is further processed by linear output layers followed by sigmoid/softmax activation to output response predictions. The entire BertKT architecture can be viewed in Figure 1.

3.3 Manipulations of Sequential Factors

We manipulate the model structures to control the intensity of sequential signals learned in the hidden space. In terms of naming, we add the prefix *static-* to the resulting models with sequential signals modeling disabled.

3.3.1 Static BertKT

In this section, we describe a modification to BertKT's positional encoding to explore sequential influences of the pro-

[1] `https://www.kaggle.com/c/riiid-test-answer-prediction`

duced hidden space on the models' performance. Specifically, we remove completely the sine-cosine positional encoding of BertKT and name it static-BertKT. For any position i in a sequence ranging from $1, ..., n$ attending to other items, the model can not distinguish the order of those items even if they are manually arranged in the correct order. This is because all other items are treated the same when computing attention regardless of their position. In such a sense, disabling the positional encoding is equivalent to disabling the exclusive indication for the model to capture explicit positionality. We compare the performance of BertKT and static BertKT in section 6.1.

3.3.2 Static DKT

In this section, we describe our modification to the sequential LSTM structures in DKT to observe performance changes due to effectively disabling weights associated with sequentiality modeling.

Static DKT Experiment

For DKT and all its variants we utilize in this work we employ an LSTM as the core computational structure. The formula for a canonical LSTM structure is as follows.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$
$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad (5)$$
$$h_t = o_t \circ tanh(c_t) \quad (6)$$

where
$x_t \in \mathbb{R}^d$: input vector of time step t
$f_t \in (0,1)^h$: forget gate
$i_t \in (0,1)^h$: input gate
$o_t \in (0,1)^h$: output gate
$h_t \in (-1,1)^h$: hidden state vector of time step t
$\tilde{c}_t \in (-1,1)^h$: cell input
$c_t \in \mathbb{R}^h$: cell state
$W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: learnable weight and bias; d and h are the dimension of input features and that of hidden states.

Concretely, the hidden state of time step $t+1$ is dependent on that of t, the current input item at $t+1$, but not on any future time step ranging from $[t+2, ..., T]$, where T is the ending time step. Here we would like to disable the sequential, hidden-state transitioning dynamics to compare results with the original DKT model. Concretely, we set each of $U_f, U_i, U_o, U_c \in \mathbb{R}^{h \times h}$ to an identity matrix $I \in \mathbb{R}^{h \times h}$. Obviously for any hidden state h, an identity matrix multiplication equal to no transformation exerted at all. Also the h could be combined with the bias term b. However, we still keep matrix U here for 2 reasons:

1. For the completeness of math formulation as this is a special case of the dynamic LSTM with learnable U.
2. In terms of implementation, U could be changed easily from a placeholder to other values for future research experiments.

Table 1: Model names and the corresponding explanations

Model/Setting	-
BertKT	baseline BertKT
static-BertKT	BertKT with positional encoding removed
DKT	baseline DKT
static-DKT	DKT with static hidden state update matrix

In the end, we have the following *static-LSTM* as the new component of the corresponding DKT model we refer to as static-DKT.

$$f_t = \sigma(W_f x_t + I h_{t-1} + b_f) \quad (7)$$
$$i_t = \sigma(W_i x_t + I h_{t-1} + b_i) \quad (8)$$
$$o_t = \sigma(W_o x_t + I h_{t-1} + b_o) \quad (9)$$
$$\tilde{c}_t = tanh(W_c x_t + I h_{t-1} + b_c) \quad (10)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad (11)$$
$$h_t = o_t \circ tanh(c_t) \quad (12)$$

where each gate update function remains the same but the explicit hidden state update is disabled.

Note that we do not hamper the sequential update format of cell state c and hidden state h, as they are part of the nature of a classic LSTM model. The hidden state is still sequentially updated but only implicitly through W_f, W_t, W_c and W_o. In summary, this experiment is to explore to what extent that sequential modeling factor of DKT contributes to its total predictive power in terms of evaluations metrics we introduce in section 5.2.

4. COMPARISON MODELS

In this section, we provide a reference table 1 to summarize the name of each model and their corresponding settings.

5. EXPERIMENTAL SETUP

5.1 Datasets

We evaluate the performance of BertKT and other comparison models based on four benchmark datasets: ASSISTments 2009-2010[2] [6], ASSISTments 2012-2013[3], Cognitive Tutor Bridge to Algebra 2006-2007[4] and Cognitive Tutor Bridge to Algebra 2008-2009[5] [15]. Among these benchmarks, ASSISTments datasets are collected from ASSISTments online tutoring system primarily for secondary school mathematics. Cognitive Tutor (now MATHia) is an intelligent tutoring system and the datasets we used are from its Algebra curricula. The two Cognitive Tutor datasets we use were used as the official development/competition dataset for the KDD Cup 2010 Challenge [16].

For all the above datasets, we follow a series of conventional pre-processing procedures such as removing all problems not

[2] https://drive.google.com/file/d/1NNXHFRxcArrU0ZJSb9BIL56vmUt5FhlE/view?usp=sharing
[3] https://drive.google.com/file/d/1cU6Ft4R3hLqA7G1rIGArVfelSZvc6RxY/view?usp=sharing
[4] http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp
[5] http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

Table 2: Comparison between default BertKT and static-BertKT. Best metrics are in bold.

Model	BertKT				static-BertKT			
Dataset/Metrics	BCE	AUC	ACC	RMSE	BCE	AUC	ACC	RMSE
ASSISTments 2009	0.4948	0.8149	**0.7562**	0.4055	**0.4932**	**0.8166**	0.7560	**0.4051**
ASSISTments 2012	0.5457	**0.7209**	0.7309	0.4274	**0.5453**	0.7203	**0.7322**	**0.4272**
Cog Tutor 2006	**0.3724**	**0.7872**	**0.8462**	**0.3380**	0.3741	0.7842	0.8456	0.3387
Cog Tutor 2008	**0.2698**	**0.9055**	**0.8881**	**0.2869**	0.2765	0.9010	0.8864	0.2895

Table 3: Comparison between default DKT and static-DKT. Best metrics are in bold.

Model	DKT				static-DKT			
Dataset/Metrics	BCE	AUC	ACC	RMSE	BCE	AUC	ACC	RMSE
ASSISTments 2009	0.4981	**0.8109**	0.7542	0.4070	**0.4975**	0.8108	**0.7548**	**0.4066**
ASSISTments 2012	0.5402	0.7253	0.7350	0.4249	**0.5400**	**0.7256**	**0.7354**	**0.4247**
Cog Tutor 2006	0.3721	0.7870	0.8461	0.3378	**0.3712**	**0.7881**	**0.8471**	**0.3372**
Cog Tutor 2008	0.2679	0.9063	0.8881	0.2864	**0.2675**	**0.9064**	**0.8883**	**0.2862**

associated with a concept/skill and using correct on first attempt responses only.

5.2 Models and Evaluation Metrics

We compare BertKT and DKT with their sequentiality-weakened counterparts. Differences and features of these model have been discussed in section 4; We evaluate each model across four metrics: BCE (Binary cross entropy loss), AUC (Area under the receiver operating characteristic curve), ACC (Accuracy as the proportion of correct classification with threshold 0.5) and RMSE (Root-mean-square error).

5.3 Training and Testing

We take out 20% of the entire data as the development set to conduct hyper-parameter tuning. For training and testing, We perform student-level k-fold cross validation (with $k = 5$) on each model, meaning each student will appear exclusively in training, validation or test set. In each phase of the cross validation, 60% of the dataset is used as the training set, 20% as the validation set to perform early stopping, and the rest 20% as the test set.

We fix a sequence length of 100 for computational efficiency with longer sequences split to subsequences of length 100 and trailing subsequence padded. We use Adam optimizer to update parameters for all models trained on two NVIDIA GeForce RTX 2080 Ti (11 GB) or TITAN Xp GPU (12 GB). All models are implemented using PyTorch with fixed random seeds for reproducibility.

6. EXPERIMENTAL RESULTS

In this section, we compare evaluations results of BertKT and DKT with their variants introduced in section 3. We report evaluation metrics per each model per metrics defined in section 5.2 as well as the relative percentage increase of these metrics between comparison models. Models with the best metrics are in bold. Specifically, we report the percentage increase by the "performance increase/decrease", meaning that increase in {BCE, RMSE} is marked by "-" and that increase in {AUC, ACC} is marked by "+". Decrease means the opposite way.

6.1 BertKT Model Results

Table 2 shows the evaluations metrics for BertKT and static-BertKT. The best performances by metric are in bold. We have explained the meanings of these variant models in table 1. Overall, the baseline BertKT performs better, with 10 bolded cells, than its sequentially disrupted variants. However, the percentage performance differences averaged over the four dataset benchmarks of static-BertKT from BertKT are -0.64%, -0.19%, -0.02%, -0.24% for BCE, AUC, ACC and RMSE, respectively ('+' indicates a performance improvement for the metric, while '-' represents a performance decrease). This suggests that BertKT is insensitive towards sequential permutation even without positional encoding.

6.2 DKT Model Results

In this section, we compare in table 3 DKT with its sequentially weakened counterpart, namely, DKT and static-DKT. Naming conventions have been introduced in table 1 in section 4.

Surprisingly, the static-DKT model was the best performing model on all datasets for all but one metric of one dataset (ASSISTments 2009-AUC). The average percentage performance differences averaged over the four dataset benchmarks of static-DKT from DKT are +0.13%, +0.04%, +0.07%, +0.09% for BCE, AUC, ACC and RMSE, respectively ('+' indicates a performance improvement for the metric, while '-' represents a performance decrease). Even after disabling weights associated with time-slice transitions, the model performs closely to the baseline model and even with a slight increase. This suggests that the embedding weights for input X W_f, W_i, W_o and W_c are still sufficient to capture signals that support the model's original performance.

Across the four metrics and four datasets, DKT improves 0.08% due to a static hidden state transition, while the performance of BertKT decreases by 0.27% due to no positional encoding (static). The average percentage performance loss of BertKT due to being static is -4.3340x larger than that of DKT. This suggests BertKT's positional encoding is more sensitive in modeling the sequence order than DKT in terms of performance.

7. CONCLUSIONS AND DISCUSSION

While our motivating observation was that transformer-based models have weaker explicit modeling of time than LSTM-based models, we find that BertKT is more dependent on its positional encoding than DKT is on its time-slice transition weights. Such sensitivity may be an indication that there is room for improvement for positional encodings to better leverage input sequence order. This improvement is perhaps seen in the performance of AKT [7], which outperforms DKT and employs additional modeling of sequentiality in the form of imposed monotonicity of attention weights with respect to time.

Our results raise the question of why DKT is not negatively affected by disabling its transition weights. Potentially, the temporal signal is being pushed down into the embedding W. Is student growth better captured in the interactions between input embeddings than it is by a generalized recurrent hidden state transition? This may speak to the powerful but simple assumption of the original knowledge tracing model, that learning is a function of *opportunity count*, growth rate, and prior.

8. REFERENCES

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003.

[2] C. Chen and Z. Pardos. Applying recent innovations from nlp to mooc student course trajectory modeling. In *Proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] X. Ding and E. C. Larson. Why deep knowledge tracing has less depth than anticipated. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.

[5] P. Dufter, M. Schmitt, and H. Schütze. Position information in transformers: An overview. *arXiv preprint arXiv:2102.11090*, 2021.

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

[7] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339, 2020.

[8] Z. Huang, D. Liang, P. Xu, and B. Xiang. Improve transformer models with better relative position embeddings. In *Findings of Empirical Methods in Natural Language Processing*, 2020.

[9] S. Jeon. Last query transformer rnn for knowledge tracing. In *Workshop on AI Education@35th AAAI Conference on Artificial Intelligence*, 2021.

[10] M. Kim, Y. Shim, S. Lee, H. Loh, and J. Park. Behavioral testing of deep neural network knowledge tracing models. In *Proceedings of the 14th International Conference on Educational Data Mining*, 2021.

[11] T. Oya and S. Morishima. Lstm-sakt: Lstm-encoded sakt-like transformer for knowledge tracing. In *Workshop on AI Education@35th AAAI Conference on Artificial Intelligence*, 2021.

[12] S. Pandey and G. Karypis. A self-attentive model for knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.

[13] Z. A. Pardos and N. T. Heffernan. Determining the significance of item order in randomized problem sets. In T. Barnes, M. C. Desmarais, C. Romero, and S. Ventura, editors, *Educational Data Mining - EDM 2009, Cordoba, Spain, July 1-3, 2009. Proceedings of the 2nd International Conference on Educational Data Mining*, pages 111–120. www.educationaldatamining.org, 2009.

[14] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi. Saint+: Integrating temporal features for ednet correctness prediction. In *Proceedings of the 11th International Learning Analysis and Knowledge Conference*, 2021.

[15] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Bridge to algebra 2008-2009. challenge data set. In *KDD Cup 2010 Educational Data Mining Challenge*, 2010.

[16] J. Stamper and Z. A. Pardos. The 2010 kdd cup competition dataset: Engaging the machine learning community in predictive learning analytics. *Journal of Learning Analytics*, 3(2):312–316, 2016.

[17] D. K. L. Tran. Riiid! answer correctness prediction kaggle challenge: 4th place solution summary. In *Workshop on AI Education@35th AAAI Conference on Artificial Intelligence*, 2021.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[19] C. Zhang, Y. Jiang, W. Zhang, and C. Gu. Muse: Multi-scale temporal features evolution for knowledge tracing. In *Workshop on AI Education@35th AAAI Conference on Artificial Intelligence*, 2021.