Deep Knowledge Tracing with Transformers

Chapter · June 2020

DOI: 10.1007/978-3-030-52240-7_46

4 authors, including:

Shi Pu
Educational Testing Service
14 PUBLICATIONS **99** CITATIONS
SEE PROFILE

Michael Yudelson
ACT, Inc.
78 PUBLICATIONS **1,615** CITATIONS
SEE PROFILE

Lu Ou
ACT, Inc.
17 PUBLICATIONS **184** CITATIONS
SEE PROFILE

Deep Knowledge Tracing with Transformers

Shi Pu, Michael Yudelson, Lu Ou, and Yuchi Huang

ACT,Inc.
500 ACT Drive., Iowa City, IA 52245, USA
{scott.pu, michael.yudelson, lu.ou, yuchi.huang}@act.org

Abstract. Bayesian Knowledge Tracing (BKT) is a traditional approach to tracking student skill acquisition in intelligent tutoring systems. The fast development of deep learning methods in recent years has prompted researchers to apply recurrent neural networks (RNN) to intelligent tutoring and computer-supported learning domains. As a result, deep learning models have shown performance superior to traditional models like BKT. RNNs, however, are less efficient when applied to long sequences of time-series data with extended patterns that are typical for learning systems. In this work, we propose a Transformer-based model to trace students' knowledge. We modified the Transformer structure to account for 1) the structure of the problem and 2) the elapsed time between problem steps. The use of problem structures allows the model to leverage between-problem relations. The inclusion of elapsed time opens the opportunity to address forgetting. Our approach outperforms the state-of-the-art methods in the literature by roughly 10% in AUC with frequently used public datasets.

Keywords: Bayesian Knowledge Tracing · Deep Knowledge Tracing · Transformer.

1 Introduction

Bayesian Knowledge Tracing (BKT) is an approach to modeling skill acquisition of students working with intelligent tutoring systems. It is an extensively studied and widely discussed method. When one talks about problem-solving support, BKT is frequently one of the methods to be considered. However, BKT is far from an ideal solution, and multiple improvements and extensions were suggested to it over the years. One of such extensions is Deep Knowledge Tracing (DKT). At the cost of interpretability, DKT and its variants continue to generate state-of-the-art performance on a wide range of benchmark datasets.

The first DKT [8] adopted the Recurrent Neural Network (RNN) architecture from the deep learning community. Recent publications on DKT discuss various RNN architectures to adapt to various student learning theories as well as explore new deep learning models. Our work is inspired by both and proposes to use the Transformer architecture to model students' knowledge state.

The Transformer model was first proposed by the Google Brain team [13] to generate better neural translations. It soon became the dominant model

The code and data is available at https://github.com/scott-pu-pennstate/dktt_light

in many of the Natural Language Processing (NLP) problems. Its most well-known variations include Bidirectional Encoder Representations from Transformers (BERT) [2] that learned contextualized word representations and Generative Pre-Training (GPT) [9] that learned the language model. The main advantage of the Transformer over RNN is its ability to learn long-range dependencies [2].

When predicting student performance on a future problem step or question item, the Transformer model pays *attention* to the entire prior history of performance. The weight of attention depends on the similarity between the future and past question items. This architecture will not model a student's skill mastery as a vector of hidden variables as RNN does. Instead, a student's knowledge state is distributed over the entire interaction history.

We modified the Transformer architecture so that it does not directly learn the representation of each question. Instead, it learns the representation of the underlying Q-matrix that relates knowledge components to question items, including the cases when multiple knowledge components are associated with an item. This modification allows us to leverage the expert-labeled question-skill association. Further, we allow the attention weight between question items to decay as students work on questions or problems, which effectively represents forgetting.

We tested our model on several datasets that have been mentioned in the literature before and were available to us. The results showed that the modified Transformer outperformed state-of-the-art results known to date in all of the cases we considered.

2 Related Work

2.1 Deep Knowledge Tracing

Deep Knowledge Tracing term was first introduced by Piech and colleagues in 2015 [8]. The authors applied an RNN based architecture and the result outperformed the traditional BKT model by approximately 35% as per the Area Under ROC curve (AUC). Khajah, Lindsey, and Mozer [3] questioned how much of the reported improvement comes from learning the representation of student-acquired knowledge. They enhanced the BKT model by allowing it to model students' non-knowledge-related regularities in the data, e.g., students' abilities. The enhanced BKT showed performance indistinguishable from the RNN-based DKT. However, as enhanced BKT requires intense feature engineering, DKT remains an elegant alternative. Soon after, Zhang et al[16] demonstrated a simple approach to further improve RNN-based DKT: incorporating more features. Their best model scored an AUC of 0.867 on the ASSISTment 09-10 dataset.

Recent work on DKT follows two general patterns. First, studies like [17] and [4] tried to adjust the RNN structure so that it is consistent with the students' learning process. For example, the Dynamic Key-Value Memory Networks by Zhang and colleagues [17] explicitly maintained knowledge components and

knowledge states as matrices in their RNN-based model. Further, Nagatan and colleagues [4] intentionally modeled students' forgetting behavior in RNN but achieved only limited success.

Another group of papers seeks to leverage recently developed deep learning models in the NLP domain (cf. [6] and [10]). Pandey and Karypis [6] used a self-attention model, where student's performance on a question item is directly compared with their past performance, without the need of maintaining a *knowledge state*. Ralla and colleagues [10] borrowed a widely used pre-training architecture in the NLP to pre-train students' interactions for downstream tasks. However, their pre-trained interactions are difficult to apply across platforms as the definition of interactions is platform-dependent.

3 Deep Knowledge Tracing with Transformer

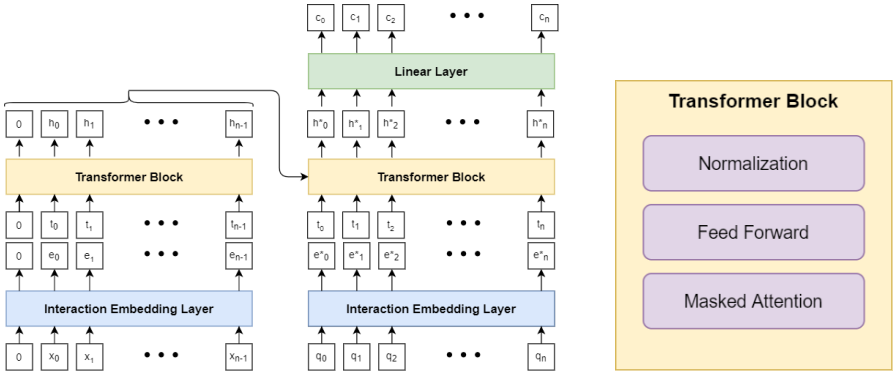
DKT with Transformer is a sequence-to-sequence problem. The main inputs to the Transformer model is a sequence tuples $x_i = (q_i, c_i)$. Here, q_i represents the question item a student is trying to answer (or a problem step they are trying to solve), and $c_i \in 0, 1$ represents whether the response is correct. The goal of the model is a sequence of correctness estimates, c_{i+1}, representing whether a student correctly solved the next question q_{i+1}.

In addition to the main input and target, we leverage two auxiliary inputs in the model. First, we feed q_{i+1} to the Transformer model. Thus, it does not have to guess what question a student is going to answer next. Second, we feed the timestamp t_i (measured as the number of hours since x_0) to the model. This input is the key information to model forgetting behavior. Thus, formally, the Transformer model is trying to predict $P(c_{i+1} = 1|x_0, .., x_i, t_0, ..., t_i, q_{i+1})$.

3.1 Transformer

DKT focuses on learning the representations of student interactions. In the Transformer model, the *Interaction Embedding Layer* is responsible for learning the static representation of each interaction x_i. The *Transformer Blocks* are responsible for learning the context-dependent representation of each interaction. The context of interaction, x_i, has two parts: 1) all previous interactions $(x_0, x_1, ..., x_{i-1})$, and 2) their respective timestamps $(t_0, t_1, ..., t_{i-1})$.

Figure 1(a) represents a simplified version of the Transformer model with 1 layer of Transformer blocks. The Transformer follows an encoder-decoder structure (the left part is the encoder, and the right part is the decoder). Students' shifted interactions sequence $(0, x_0, x_1, ..., x_{n-1})$ is first embedded as $(0, e_0, e_1, ..., e_{n-1})$ by the *Interaction Embedding Layer*. Afterward the e_i sequence and interaction time sequence, t_i, are jointly processed by a stack of Transformers to learn a hidden presentation sequence, h_i. Then, on the decoder side, the h_i sequence from the encoder, the embedded questions sequence, e_{i+1}, and question time sequence, t_{i+1}, are jointly processed by another stack of



(a) Transformer-overview (b) Transformer-block

Fig. 1: Transformer Architecture

Transformer blocks to generate the decoder hidden representations: h^*_{i+1}. The final decoder outputs pass through a linear layer to generate the predicted correct possibility sequence and are compared with observed target correctness values c_{i+1} to calculate the loss value.

Figure 1(b) takes a more detailed look at the composition of a Transformer block that has three layers: the *Masked Attention Layer*, the *Feedforward Layer*, and the *Normalization Layer*. The *Masked Attention Layer* transforms the static interpretation, e_i, into the contextualized representation, h_i and h^*_{i+1}. We spend more time on explaining the *Interaction Embedding Layer* and *Masked Attention Layer* in the following paragraphs and leave other model details for the Appendix.

3.2 *Interaction Embedding*: Map Interaction and Questions to Vectors

The interaction mapping layer maps student interaction, (q_i, c_i) or a question, q_i[1], to its static representation: a high dimensional vector e_i. The layer first creates an interaction-skill mapping matrix W and skill embedding matrix S, then it computes the embedding for an interaction x_i as :

$$e_i = softmax(W_{i.})S \tag{1}$$

$W_{i.}$ is the ith row of W, representing the weight associated with all latent skills for the interaction x_i. $softmax$ function normalizes these weights. Each column of S is a vector representation of a latent skill. So, the static representation of interaction x_i is a weighted sum of all underlying latent skills.

[1] q_t will be coded as $(q_i, 1)$, making sure that the question and its correct answer/interaction share the same embedding. It also prevents leaking information to the decoder

The architecture of a Transformer model assigns the responsibility to learn the weight to the W matrix, and the responsibility to learn the representation of skills to the S matrix. The design has two benefits. First, it allows us to leverage the expert-labeled interaction skill structure that is common in intelligent tutoring systems. To leverage the expert labeled question item-skill mapping, we initialize the mapping matrix W using the expert labels. For example, if q_i is tagged with s_j in the data, we will initialize the mapping value between $(q_i, 0)$ and $(s_j, 0)$ to be $\frac{(1-smoothing)}{temperature}$, and $\frac{smoothing}{(n_{skills}-1)*temperature}$ for every other skill. *smoothing* and *temperature* are two hyperparameters that control the smoothness of logits passed to the $softmax$ layer. Second, it is flexible enough that if there is more than one potential skill associated with an interaction, the layer can learn that association from the data.

3.3 *Masked Attention*: learn contextualized interaction embedding

The following equations illustrate how a contextualized interaction representation, h_j, is calculated from the static interaction representation e_j.

$$q_j = Qe_j, k_j = Ke_j, v_j = Ve_j \tag{2}$$

$$A_{ij} = \frac{q_j k_i + b(\Delta t_{j-i})}{\sqrt{d_k}} \tag{3}$$

$$h_j = \sum_{i \leq j} softmax(A_{ij}) v_i \tag{4}$$

In Equation(2), the masked attention layer extracts query q_j, key k_j, and value v_j through multiplying the static representation, e_j, with three trainable matrices: Q, K, and V. Key and query could be interpreted as the latent skills associated with interaction e_j, and value is the state of the latent skill (or knowledge state) associated with e_j.

Equation(3) calculates the *attention*, A_{ij}, assigned to a past interaction e_i. It has two components: 1) $q_j k_i$, the query-key agreement between e_j and e_i, which could be interpreted as the degree of latent skills overlapping between interaction e_j and e_i; 2) A time gap bias, $b(\Delta t_{j-i})$, which adjusts the attention weight by the time gap between interactions e_j and e_i. The denominator, $\sqrt{d_k}$ is used to normalize the attention magnitude. Refer to the Appendix for a detailed explanation. In theory, if two question items have a large overlap in latent skills and one is immediately followed by the other, attention between these two interactions will be high. At the same time, when two interactions have little overlap in latent skills or are too far apart the attention weight value will be low.

Equation(4) calculates the contextualized representation h_j as a weighted sum of the past value representations. The weight is proportional to the attention weights, normalized by the $softmax$ function. Note, that since the task is to predict the correctness of the next question, c_{j+1}, a mask is forced onto the attention values so that only A_{ij} where $i \leq j$ are used for calculating h_j. In other words, $softmax(A_{kj}) = 0, \forall k > j$.

Table 1: Dataset statistics

Datasets	Interactions	Students	Items	Skills
ASSISTments 2017	943K	1,709	4117	102
STAT F2011	190K	333	1224	81
KDD 2010, Algebra I	4,420K	3,287	1,379	899

4 Experiments

To evaluate the performance of our Transformer approach, we ran 5-fold student-stratified cross-validation on three datasets that are frequently used in the literature. Table 1 lists descriptive statistics for the datasets.

ASSISTments 2017[2]. The data tracks middle and high school students' usage of the ASSISTments online tutoring system. It includes 1,709[3] students , 942,816 interactions, 4,117 questions, and 102 labeled skills.

STAT F2011[4]. This data tracks student scores in a college-level engineering statics course. The data contains 333 students, 189,620 interactions, 1,224 questions, and 81 labeled knowledge components. We followed the pre-processing strategy used by[16] to 1) concatenate problem names and step names into a question, and 2) keep only final attempts for each question.

KDD, A[5]. This data is the challenge set A – Algebra I 2008-2009 data set from the KDD 2010 Educational Data Mining Challenge. It originally contains 3,310 students and 9,426,66 steps. We kept only steps that had knowledge components.

To run student sequences in batches, all sequences must be of the same length *maxlen*. We followed the Pandey and Karypis[6] strategy: 1) for sequences shorter than *maxlen*, we added a special *padding* token to the left, 2) for sequences longer than *maxlen*, we folded the sequence to segments of length *maxlen*, and padded the remainder with the *padding* token to the left. Like other deep learning models, the Transformer's performance depends on the choice of optimal hyperparameters. Table 2 demonstrates all the hyperparameters tuned in our experiment.

5 Results

Table 3 summarizes our findings and compares them to the start-of-the-art Transformer results in the literature, as well as the Bayesian Knowledge Tracing (BKT) model. The main metric that we reported is AUC as it is the standard often used in the literature, but we also reported root mean squared error (RMSE) and accuracy for both the Transformer models and BKT models.

[2] https://sites.google.com/view/assistmentsdatamining

[3] this number is different from the reported statistics in Pandey and Karypis[6] but matches the number in the official data release publication [7])

[4] https://pslcdatashop.web.cmu.edu/ DatasetInfo?datasetId=507

[5] https://pslcdatashop.web.cmu.edu/KDDCup/

Table 2: Hyper-parameters

Aspect	Search Range
Prepossess	max len = 128; smooth = 0.15; temperature = [0.1, 0.6, 0.8]; lambda = 1;
Architecture	hidden size = [32, 64, 128]; hidden layer = [1, 6]; num of heads = [4, 8]
Regularization	dropout = [0.1, 0.15, 0.2]
Train	learning rate = 0.001; batch size = 128

Table 3: Student-stratified 5-fold cross validation results.

	ASSISTment 2017			STAT F2011			KDD 2010, A		
Model	Acc	RMSE	AUC	Acc	RMSE	AUC	Acc	RMSE	AUC
BKT	0.653	0.473	0.628	0.951	0.205	0.821	0.830	0.357	0.744
Literature	.	.	0.734 [6]	.	.	0.853 [6]	.	.	.
Transformer ours	0.743	0.414	**0.806**	0.964	0.170	**0.947**	0.838	0.347	0.784
Gain (%)	.	.	9.81%	.	.	11.02%	.	.	.

The modified Transformer model is superior to BKT on all metrics with a large margin. Most notably, it outperforms the state-of-the-art Transformer models from the literature by 9.81% and 11.02% on ASSISTments 2017 and STAT F2011 datasets. The remarkable gain is not due to the structure of the original transformer model. Pandey and Karypis [6]'s self-attention model is roughly equivalent to a 1-layer Transformer with learned positional encoding. Though their work generates state-of-the-art performance on multiple benchmark datasets, their reported AUC score on ASSISTments 2017 and STAT F2011 is about 10% worse than our modified Transformer. To further illustrate this point, we repeat the experiment on the original Transformer with/without the modified components, as illustrated in Table 4. The original Transformer fails to reach state-or-the-art results on ASSISTments 2017. However, by adding the *interaction skill mapping* and *time-bias* to its structure, the model gains an obvious performance boost.

Figure 2 compares the initial expert-labeled interaction skill mapping and the learned item-skill mapping. There, we show the interaction-skill mapping for the STAT F2011 dataset, where we only include rows tagged with skills. Approximately 85.5 % of interactions kept their initial mapping as the dominant mapping after training, while the remaining 14.5% changed. This suggests that both expert-labeling and flexible mapping are important for the model's performance.

Figure 3 visualizes attention weights in the model trained on STAT F2011 data. We sampled a pseudo sequence of 64 interactions from two expert-labeled skills: a) *replace general loads with force and couple* and b) *find net force position for linear distribution*. The query interaction, x_{63}, belongs to the former latent skill (a).

In the figure, each column represents attention between a pair of interactions. The column on the far left represents attention between interactions x_0 and x_{63}.

Table 4: AUC under Different Architecture

Architecture	ASSISTment 2017	STAT F2011	KDD 2010, A
Transformer: 1-layer original	0.709	0.917	0.772
Transformer: 1-layer + mapping	0.737	0.939	0.772
Transformer: 1-layer + time-bias	0.704	0.931	0.777
Transformer: 1-layer + all	0.773	0.946	**0.784**
Transformer: 6-layer + all	**0.806**	**0.947**	0.775

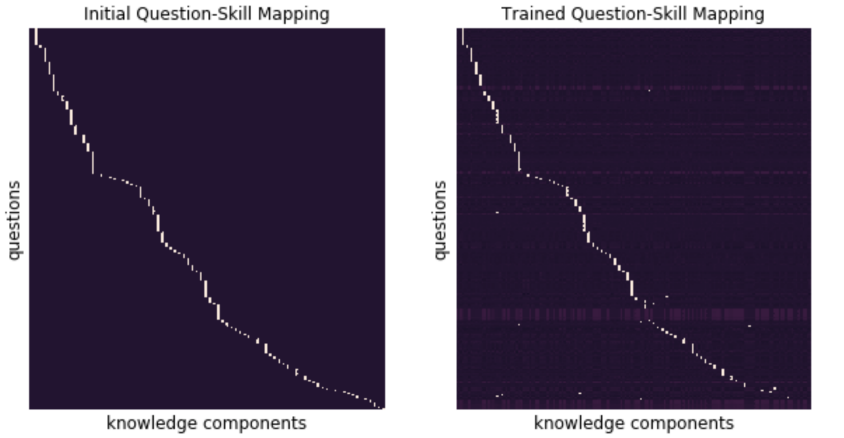


Fig. 2: Comparing Initial Interaction-Skill Mapping with Learnt Interaction-Skill Mapping

The column on the far right represents attention between interactions x_{62} and x_{63}. The first row represents the combined attention. The second row focuses on time bias. The third row visualizes the query-key agreement. As we expected, time bias (in general) assigns a higher value to closer interactions (columns on the far right) than distant interactions (columns on the far left). Further, the query-key agreement will have a higher value to similar skill interactions (column labeled with A). Also, the time bias adjusts the attention, so that distant same skill events (A in on the left side) are assigned lower attention than closer same-skill events (A on the right side).

6 Conclusions and Future Work

In this paper, we proposed a Transformer-based approach to model student learning. We modified the original Transformer architecture so that it is aware of the structure of interactions and the time between them. We validated the model by comparing it to known alternatives using real-world datasets. The results indicate that the modified Transformer outperforms the state-of-the-art Transformer model's results found in the literature by approximately 10%.

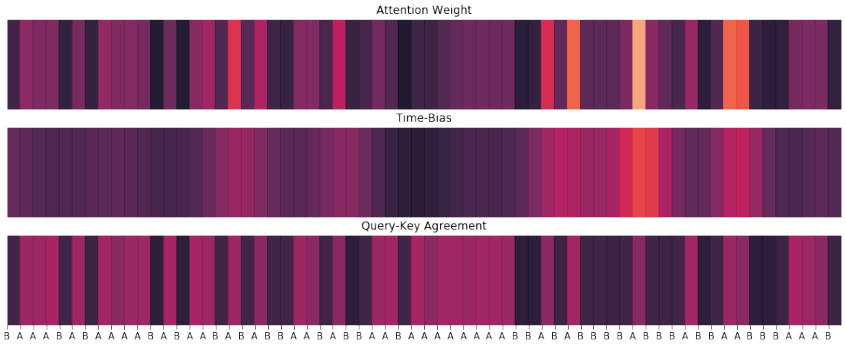


Fig. 3: Visualize Attention

For future work, we intend to explore how to efficiently incorporate more feature information into the Transformer architecture. For example, Zhang and colleagues [16] showed that adding engineered features could boost the performance of RNN-based DKT models. Furthermore, our interaction skill mapping cannot utilize complex structural information about latent skills, for example, when the latent skills come from a hierarchical structure such as skill taxonomy. We intend to continue exploring how to flexibly represent this structure as a part of the Transformer architecture.

7 Appendix

Time Bias

In Equation(3), time bias, $b(\Delta t_{j-i})$ is used to adjust the attention, A_{ij}, between interaction e_j and e_i. Formally:

$$b(\Delta t_{j-i}) = uK^*(f(g(\Delta t_{j-i}))) \tag{5}$$

$\Delta t_{j-i} = t_j - t_i$ represents the time gap between inputs x_i and x_j. g is a negative exponential function, $g(x) = e^{-\lambda x}$, that fits well with human's forgetting behavior [15]. f is the sinusoidal positional embedding function [13]. u and K^* are the learned vector and matrix from the Transformer model.

Time bias replaces the positional encoding [13] to provide time information to the *Masked Attention Layer*. In NLP community, effective alternatives to positional encoding is an important research topic [5, 14, 1, 12]. Our time bias approach is based on the work of Dai and colleagues [1] and Shaw and colleagues [11]. However, our approach is novel in two aspects: 1) we are using time difference Δt instead of position difference ΔP, and 2) time delta is transformed before fitting to the model. The bias term models the forgetting behavior: the attention magnitude between interactions x_i and x_j will be adjusted by their distance in time, with negative exponential decaying. Though forgetting behavior has been modeled in RNN architecture [3, 4], we are the first one to model

forgetting behavior in the Transformer architecture.

Feedforward Layer
The feedforward layer has exactly the same structure as the original Transformer:

$$F(h_i) = max(0, h_i W_1 + b_1)W_2 + b_2$$

Normalization Layer
The normalization layer applies normalization across the feature dimension:

$$N(h_i) = (h_i - \mu(h_i))/\sigma(h_i)$$

Residual Connection
In the side Transformer blocks, there are residual connections wrapping the *masked attention layer* and *feedforward layer*. So, $y = x + layer(\frac{x-\mu(x)}{\sigma(x)})$.

Linear Layer + Loss
When calculating the loss, q_j is encoded as $q_{j0} = (q_j, 0)$ and $q_{j1} = (q_j, 1)$ to utilize the weights inside the *Interaction Embedding Layer*:

$$e_{j0} = softmax(W_{j0.})S, \quad e_{j1} = softmax(W_{j1.})S$$

$$p_j = \frac{exp(h_j e_{j1})}{exp(h_j e_{j0}) + exp(h_j e_{j1})}$$

$$Loss = -\sum_j c_j log(p_j) + (1 - c_j) log(1 - p_j)$$

References

1. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., Salakhutdinov, R.: Transformer-XL: Attentive Language Models beyond a Fixed-Length Context (2019), archive
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
3. Khajah, M., Lindsey, R.V., Mozer, M.C.: How deep is knowledge tracing? Proceedings of the 9th International Conference on Educational Data Mining, EDM 2016 pp. 94–101 (2016)
4. Nagatani, K., Chen, Y.Y., Zhang, Q., Chen, F., Sato, M., Ohkuma, T.: Augmenting knowledge tracing by considering forgetting behavior. The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019 pp. 3101–3107 (2019). https://doi.org/10.1145/3308558.3313565
5. Omote, Y., Tamura, A., Ninomiya, T.: Dependency-based relative positional encoding for transformer nmt. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019). pp. 854–861 (2019)

6. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing. arXiv preprint arXiv:1907.06837 (2019)
7. Patikorn, T., T., H.N., Baker, R.S.: ASSISTments Longitudinal Data Mining Competition. Iningnternational Conference on Educational Data Mining (2018)
8. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. In: Advances in neural information processing systems. pp. 505–513 (2015)
9. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8), 9 (2019)
10. Ralla, A., Siddiqie, S., Krishna Reddy, P., Mondal, A.: Assessment Modeling: Fundamental Pre-training Tasks for Interactive Educational Systems Youngduck. ACM International Conference Proceeding Series pp. 209–213 (2020). https://doi.org/10.1145/1122445.1122456
11. Shaw, P., Uszkoreit, J., Vaswani, A.: Self-attention with relative position representations. arXiv preprint arXiv:1803.02155 (2018)
12. Shiv, V., Quirk, C.: Novel positional encodings to enable tree-based transformers. In: Advances in Neural Information Processing Systems. pp. 12058–12068 (2019)
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
14. Wang, B., Zhao, D., Lioma, C., Li, Q., Zhang, P., Simonsen, J.G.: Encoding word order in complex embeddings. arXiv preprint arXiv:1912.12333 (2019)
15. White, K.G.: Forgetting functions. Animal Learning and Behavior **29**(3), 193–207 (2001). https://doi.org/10.3758/BF03192887
16. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. 26th International World Wide Web Conference, WWW 2017 pp. 765–774 (2017). https://doi.org/10.1145/3038912.3052580
17. Zhang, L., Xiong, X., Zhao, S., Botelho, A., Heffernan, N.T.: Incorporating rich features into deep knowledge tracing. L@S 2017 - Proceedings of the 4th (2017) ACM Conference on Learning at Scale pp. 169–172 (2017). https://doi.org/10.1145/3051457.3053976