Incorporating Wide Context Information for Deep Knowledge Tracing using Attentional Bi-interaction

Raghava Krishnan, Janmajay Singh, Masahiro Sato, Qian Zhang and Tomoko Ohkuma

Fuji Xerox Co Ltd., Yokohama, Japan

Abstract

Online learning platforms also known as Computer Aided Education Systems have recently grown in importance owing to their ability to personalize study plans in accordance with individual student requirements. Learning platforms have modeled student knowledge state using student responses with the recently popular Deep Knowledge Tracing (DKT) technique. Using context information has also proven effective in various predictive problems prompting learning platforms to store a variety of context features about a student's performance history. An example context may be response time, where shorter times to answer questions may indicate higher mastery of a skill. Therefore, it is crucial to incorporate context features in the most effective way possible. Most of the research in DKT either use no context features, or use a set of context features that span only a narrow set of student characteristics. To overcome this, we identify a wide set of context features and incorporate their interactions into the DKT model. We then observe the effects of incorporating these additional context feature interactions and also propose an adaptive weighting technique that learns the appropriate context feature interactions were evaluated using AUC scores.

Keywords

Computer Aided Education, Adaptive learning, personalization, sequential modeling

1. Introduction

Computer Aided Education (CAE) systems aim to personalize the study plan of a user to best suit his/her needs. This is achieved through the process of Knowledge Tracing where the current knowledge state of the user is estimated using the history of their interactions with the system, and this estimated knowledge state is used to predict the future performance of the user. Accurately predicted future student performances are then used as cues to better personalize the study plan of each user. In addition to a history of user responses, CAE systems usually also store additional metadata related to user performance history, like response time, type of question, number of attempts, etc. Adomavicius et al. [1] refer to this additional information as contexts or context features, and give an interactional view of context as having a cyclical

L2D'21: First International Workshop on Enabling Data-Driven Decisions from Learning on the Web, March 12, 2021, Jerusalem, IL

raghava.krishnan@fujixerox.co.jp (R. Krishnan); janmajay.singh@fujixerox.co.jp (J. Singh);

sato.masahiro@fujixerox.co.jp (M. Sato); qian.zhang@fujixerox.co.jp (Q. Zhang); ohkuma.tomoko@fujixerox.co.jp (T. Ohkuma)

^{© 2021} Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). CEUR Workshop Proceedings (CEUR-WS.org)

relationship with an underlying activity. In our case, the activity is a student's response and context is the additional information.

A popular approach to knowledge tracing over the past few years had been Deep Knowledge Tracing [2] (DKT) which learns a continuous representation of the knowledge state as compared to the discrete variable representation used in Bayesian Knowledge Tracing [3, 4, 5] (BKT). The drawback with DKT is that it used only the history of student responses, while there are other factors that affect a student's performance while using an online learning platform, such as forgetting, learning ability, etc. This is partially overcome in [6] which uses a technique called bi-interaction in a framework called Bi-interaction Deep Knowledge Tracing (BIDKT). This technique incorporates context in the form of second degree interactions between input question response and context features relating to forgetting behavior, where interactions are the inner product or Hadamard product of the embedding vectors of the context features. This technique showed us that using context feature interactions is an effective way of incorporating the above mentioned factors in a knowledge tracing model.

A drawback of BIDKT was using only a small set of additional context features (narrow), in this case, student forgetting behavior. While including only a few features led to a reasonable improvement in performance, it would be of interest to know if the trend would have continued to rise as more related contexts were identified and included in the model. Additionally, the bi-interaction technique used, assigned the same weight to all feature interactions. This might be a problem if a larger set of context features are used, as important interactions might get diluted along with unimportant ones. This may result in either saturation or even a drop in model performance.

In this paper we posit that using additional context features (wide) should lead to improved performance of knowledge tracing models. We further hypothesize that existing models may not be well suited to effectively use additional contexts since they do not weigh contexts by their importances. To verify these ideas, we first identify additional contexts which may provide important cues for future student performance prediction. We then analyze how current best model performance changes with wider contexts. Finally, we propose a new technique by modifying BIDKT which adaptively learns weights for contexts via an Attention network similar to [7] and see how it compares to identified baselines.

2. Related Work

Knowledge Tracing: Since the emergence of Long Short-term Memory(LSTM) Networks, the Deep Knowledge Tracing model has been the most popular knowledge tracing technique [2, 6, 8]. There have been variations and extensions of DKT such as [9] that use Memory Networks to model individual skill levels more effectively, while [10] use hop LSTMs to use only relevant past exercises to estimate the current skill level. There have also been efforts to separately model the student's ability in the Dynamic Key-Value Memory Networks for Knowledge Tracing framework [11]. Although most efforts at knowledge tracing only use sequential models [12], [13] use Convolutional Neural Networks for knowledge tracing, while [14] uses sequential models such as LSTM to estimate parameters of IRT. There have also been a few attempts at using Attention Networks in knowledge tracing. Pandey et al. [15] use

a self-attention mechanism to identify relevant Knowledge Components from past learning interactions of the student.

Using Context Features in Knowledge Tracing: Given the success of using context feature and their interactions in other domains [16, 17, 18, 19, 20, 7, 21], recently there have been efforts in knowledge tracing to incorporate context features in predictive models as well. Sun et al. [22] try to use a wide variety of context features for the task of knowledge tracing, they achieve this by ensembling one of various algorithms such as Decision Tree or Support Vector Machines or Linear Regressor with the Dynamic Key-Value Memory Network architecture of [9]. Zhang et al. [8] propose an Autoencoder architecture to reduce the dimensionality of a large number of features being input to DKT. Attention Networks have been used to incorporate context features as well. Pandey et al. [23] use a self-attention mechanism to incorporate contextual information relating to exercise relations and forgetting.

There have also been efforts to incorporate context features in the form of interactions for the task of knowledge tracing. Vie et al. [24] uses Factorization Machines to model the interaction between a wide variety of features. Nagatani et al. [6] on the other hand models feature interactions using Bi-interaction, a variant of Factorization Machines and additionally inputs these interactions to an LSTM and achieves reasonable results. The model in [6] also forms the basis of our work. Our proposed model aims to improve the model proposed in [6] by increasing the variety of context features and also by proposing a technique to utilize the additional context information in an effective way using the attention mechanism from [7]. While there have been other efforts at using contextual features in the DKT framework [8] and Factorization Machines framework [24], this is the first attempt at using Attentional bi-interation to incorporate context feature interactions in the DKT model.

3. Background

In this section, we will provide some background to the domain of knowledge tracing and also describe the architectures of the DKT and BIDKT models. These models are the basis for our proposed architecture.

3.1. Knowledge Tracing

Knowledge tracing is the process of estimating a student's current knowledge state and using it to predict future performance. Given a sequence of past learning attempts $\mathbf{x}_0 \cdots \mathbf{x}_t$, we need to predict the student's performance for attempt \mathbf{x}_{t+1} . In general, an attempt $\mathbf{x}_t = (q_t, a_t)$ is defined as a tuple that contains the skill set id (q_t) of a question at time step *t* and whether the student response (a_t) to the question is correct or not. In this case, q_t is identified as a skill set id from a set of skills *Q* and a_t is a binary variable. We need to predict a_{t+1} for q_{t+1} .

3.2. Deep Knowledge Tracing

Deep Knowledge Tracing (DKT) shown in Figure 1(a), models students' knowledge state transition using an LSTM, which is a modified version of the RNN. The architecture of the DKT model is from [2] where, at time step *t*, the knowledge state is represented as $\mathbf{h}_t \in \mathbb{R}^k$ where *k* is the hidden state dimension. The DKT model in Figure 1(a) shows the 2 processes the model is supposed to perform i.e. estimating the current knowledge state and predicting future performance.

In the case of DKT, the input \mathbf{x}_t is a one-hot vector, which is the Cartesian product of q_t and a_t . \mathbf{x}_t is then embedded into a dense real-valued vector \mathbf{v}_t . During the knowledge state estimation process, for a given input $\mathbf{x}_t = (q_t, a_t)$ at each time step t, the knowledge state \mathbf{h}_t is updated. The knowledge state \mathbf{h}_t is estimated using the embedded vector \mathbf{v}_t and previous knowledge state \mathbf{h}_{t-1} using the LSTM module. For the prediction process, the output layer is implemented as a linear layer with sigmoid activation. The predicted probability of correct responses to all skill sets $\mathbf{y}_t \in \mathbb{R}^{|Q|}$ formed the model output.

3.3. Bi-Interaction Deep Knowledge Tracing

Bi-Interaction Deep Knowledge Tracing (BIDKT) [6], shown in figure 1(b) is an extension to the DKT model, which integrates interactions between the input question response and context features related to forgetting behavior into the DKT model. The context features used were repeated time gap, sequence time gap and past trial counts which have been described further in Section 5.2.

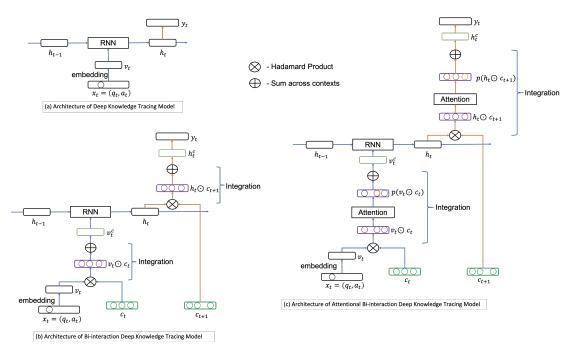


Figure 1: Architectures for (a) deep knowledge tracing (b) bi-interaction deep knowledge tracing and (c) Proposed model: Attentional bi-interaction deep knowledge tracing. In each architecture, the blue arrows describe a process of modeling a student's knowledge while orange arrows describe a process of predicting a student's performance. In our proposed model, the context information (shown in green) is incorporated in the form of context interactions (represented in purple) which are weighted according to importance to obtain a weighted interaction vector (purple with multi-colored components). The incorporation of context information happens at time steps *t* and t + 1 as c_t and c_{t+1}

The input to the RNN module \mathbf{v}_t^c is computed using an integration technique called biinteraction. \mathbf{v}_t^c is the product of the interactions between \mathbf{v}_t the embedded dense real-valued vector of the input \mathbf{x}_t , and \mathbf{c}_t^i , the embedded dense real-valued vector of each context feature relating to forgetting.

$$\mathbf{v}_t^c = \sum_{i=1}^n \mathbf{v}_t \circ \mathbf{c}_t^i \tag{1}$$

Here, *n* is the number of context features. The current knowledge state is computed using the previous knowledge state \mathbf{h}_{t-1} and the product of integration \mathbf{v}_t^c as:

$$\mathbf{h}_t = \phi(\mathbf{v}_t^c, \mathbf{h}_{t-1}) \tag{2}$$

To predict the student's performance at the next attempt, the interaction between the current knowledge state \mathbf{h}_t and context at the next attempt \mathbf{c}_{t+1}^i is computed. The context embedding parameters are shared between the current knowledge state estimation step and the future performance prediction step.

$$\mathbf{h}_{t}^{c} = \sum_{i=1}^{n} \mathbf{h}_{t} \circ \mathbf{c}_{t+1}^{i}$$
(3)

And finally the probability of answering correctly $\mathbf{y}_t \in \mathbb{R}^{|Q|}$ is computed as:

$$\mathbf{y}_t = \sigma(\mathbf{b}^{out} + \mathbf{W}^{out}\mathbf{h}_t^c), \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function, $\mathbf{W}^{out} \in \mathbb{R}^{|Q| \times k}$ is the weight matrix, and $\mathbf{b}^{out} \in \mathbb{R}^{|Q|}$ is the bias vector of the output. The implementation of the output layer is similar to that of DKT.

4. Proposed Approach

Our proposed model Attentional Bi-Interaction Deep Knowledge Tracing (ABIDKT), shown in figure 1(c) is an extension to the BIDKT model, which weights interactions between the skill id and context features in the BIDKT model. The original BIDKT model uses a narrow set of context features related only to the long term trait of forgetting. But our goal is to use a wider set of features so as to better estimate the knowledge state and accurately predict future performance. The additional context features are wins, fails, question type, previous attempt response time and difference in previous attempt response time which have been described in detail in Section 5.2.

The wins and fails context features have been picked up from [11], as the paper says that these features can be a good indication of the student's learning ability. The question type feature is important because each question type is associated with a different level of difficulty and therefore this feature serves as a great indicator of correct response probability. The previous attempt response time feature was picked up from [8] and the difference in previous attempt response time feature was used because preliminary analysis showed that it is a good indicator of skill mastery.

However, using this wider set of features could lead to the issue of the important interactions being averaged out. Therefore, the ABIDKT model uses an attention network in a modified

integration technique to weight the important interactions and ensure that they do not get averaged out.

In this case the input to the RNN module \mathbf{v}_t^c is computed using a modified integration technique called attentional bi-interaction. In this integration method, \mathbf{v}_t^c is the product of the interactions between \mathbf{v}_t the embedded dense real-valued vector of the input \mathbf{x}_t , and \mathbf{c}_t^i , the embedded dense real-valued vector of each context feature.

$$\mathbf{v}_t^c = \sum_{i=1}^n p_i(\mathbf{v}_t \odot \mathbf{c}_t^i)$$
(5)

Here *n* is the number of context features and $p_i \in \mathbb{R}$ is the normalized attention weight of the interaction calculated using the attention layer. The attention weight p'_i and p_i the attention weight normalized by the Softmax function are computed as:

$$p'_{i} = \mathbf{h}^{T} tanh(\mathbf{W}^{att}(\mathbf{v}_{t} \odot \mathbf{c}_{t}^{i}) + \mathbf{b}^{att}) \quad and \quad p_{i} = \frac{exp(p'_{i})}{\sum_{i=1}^{n} exp(p'_{i})}$$
(6)

Similar to BIDKT, the current knowledge state is computed using the previous knowledge state \mathbf{h}_{t-1} and the product of integration \mathbf{v}_t^c as:

$$\mathbf{h}_t = \phi(\mathbf{v}_t^c, \mathbf{h}_{t-1}) \tag{7}$$

To predict the student's performance at the next attempt, the weighted interaction between the current knowledge state and context at the next attempt is computed as:

$$\mathbf{h}_{t}^{c} = \sum_{i=1}^{n} p_{i}(\mathbf{h}_{t} \odot \mathbf{c}_{t+1}^{i})$$
(8)

The probability of a correct answer $\mathbf{y}_t \in \mathbb{R}^{|Q|}$ is computed in the same way it is for BIDKT. The implementation of the output layer is also the same as the DKT and BIDKT models. Similar to the architecture of BIDKT, the parameters of context embedding are shared between the current knowledge state estimation step and the future performance prediction step in the ABIDKT model as well. In the case of the attention network parameters, 2 variations were experimented with, one where the attention network parameters are shared and the other where the parameters are not shared.

The training parameters for BIDKT are the skill id (\mathbf{x}_t) embedding matrix **A**, weights of the RNN, weight \mathbf{W}^{out} and bias \mathbf{b}^{out} for prediction and embedding matrix **C** for the context information. In the case of ABIDKT we additionally have to train weight \mathbf{W}^{att} , bias \mathbf{b}^{att} and parameter \mathbf{h}^T of the attention layer. These parameters are jointly learned by minimizing a standard cross entropy loss between the predicted probability of correctly answering the next question for the skill id q_{t+1} and the true label a_{t+1} :

$$\mathcal{L} = -\sum_{t} (a_{t+1} \log(\mathbf{y}_{t}^{T} \delta(q_{t+1})) + (1 - a_{t+1}) \log(1 - \mathbf{y}_{t}^{T} \delta(q_{t+1})))$$
(9)

where $\delta(q_{t+1})$ is the one-hot encoding for which skill id is answered in the next time step t + 1.

The training process for the ABIDKT model is the same as the training process for BIDKT and DKT. The main difference between the models lies in the set of trainable parameters.

5. Experiments

Experiments were conducted to compare the performances of the proposed architecture ABIDKT with BIDKT and DKT with different combinations of context features. The experiments were conducted to verify the following 2 hypotheses:

- 1. The bi-interaction technique used in the BIDKT architecture cannot effectively leverage a wider set of context features than those used in [6]
- 2. Weighting context feature interactions using an attention network ensures that the performance does not saturate even on increasing the number of context features

5-fold cross validation was performed by using a 70% : 10% : 20% ratio for the train:validation:test split as done in the experimental setting of [6]. The details of the databases used, experiments conducted and results obtained are given below.

5.1. Datasets

The datasets chosen for the experiments are the Assistments 2012-2013 [25] dataset which contains information about students studying school level Mathematics with multiple question types, and the Slepemapy.cz [26] dataset which contains data from an online platform that teaches primary school Geography mainly consisting of 2 question types.

Table 1

Statistics of the data.

Dataset	#records	#users	#items
Assistments 2012-2013	5,818,868	45,675	266
slepemapy.cz	10,087,305	87,952	1,458

5.2. Preprocessing

Records where user made only a single attempt of a single skill set item were removed as in [6]. Additionally a few noisy records with negative response times were also removed. Continuous valued context features were preprocessed and discretized to use in BIDKT and ABIDKT models. Further details are as follows:

- 1. repeated time gap: calculated using the difference in time stamp between the current and previous attempt of same skill in minutes.
- 2. sequence time gap: calculated using the difference in time stamp between the current and previous attempt (independent of skill id) in minutes.
- 3. past trial counts: calculated as the count of the number of times the same skill has been attempted in the past.
- 4. wins: calculated as the count of correct responses in the past trials of the same skill.
- 5. fails: calculated as the count of incorrect responses in the past trials of the same skill.
- 6. question type:
 - discrete value in the range 0-5 for the Assistments 2012-2013 dataset.

- binary discrete value 0,1 for the Slepemapy.cz dataset.
- 7. previous attempt response time: time taken for the response of the previous attempt of the same skill, in seconds.
- 8. difference in previous attempt response time: calculated as the difference in response times of the last 2 attempts in seconds, of the same skill.

All features except for question type were discretized using the log_2 scale. The repeated time gap, sequence time gap and past trial counts features are same as the context features used in BIDKT [6]. The additional context features were determined based on common features available across datasets and common features used in KT literature [8, 11].

5.3. Hyper-parameters

The set of hyperparameters which maximized averaged AUC over 5-fold cross validation were chosen for final model implementations. Final results on corresponding test sets were also reported using the AUC metric.

The hyper-parameters were set as follows:

- 1. learning rate: varying the learning rate did not have a significant effect on the maximum value of AUC. Various learning rate values between 0.001 and 1 were tried, at values that were approximate multiples of 3 i.e. 0.001, 0.003, 0.01, 0.03, etc. The value was further fine-tuned around the best performing value. Finally, the learning rate was set at 0.7 for the Assistments 2012-2013 dataset, except for the DKT architecture which was fixed at 0.5. For the Slepemapy.cz dataset, the learning rate was fixed at 0.9 for all architectures.
- 2. hidden layer dimensions: Different values of hidden layer dimension between 10 and 100 were tried at differences of 10, and the value was empirically set at 30 for all variations of architectures and datasets.
- 3. dropout: The value of dropout had been set using the best value of dropout from the experiments in [6] at 0.3.
- 4. weight decay: weight decay values were varied between 10^{-6} and 10^{-3} at multiples of 10 i.e. 10^{-5} , 10^{-4} , and the best value varied between different folds in the k-fold cross validation.
- 5. mini batch size: this value was set at 100 for both datasets. For the Slepemapy.cz dataset, although the batch size value in [6] was set at 30, we set it at 100 to speed up processing.
- 6. epochs: the epochs were set as 1.5 times the maximum number of epochs till the point of convergence of AUC across all 5 folds for the BIDKT architecture. The number of epochs was set at 600 and 200 for the Assistments 2012-2013 and Slepemapy.cz datasets respectively and the highest test AUC score among all these epochs was reported.

6. Results and Discussion

The results shown are for the DKT, BIDKT and 2 variations of the ABIDKT architecture respectively. The results for the BIDKT and ABIDKT architectures are shown for different number of features. The feature combinations are:

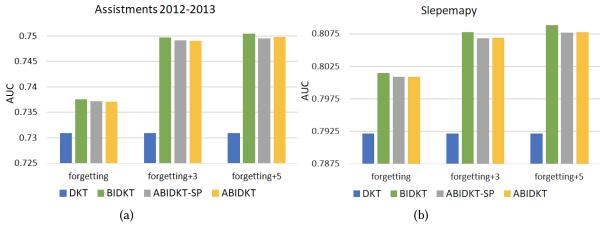


Figure 2: Average test AUC scores of different architectures on (a) the Assistments 2012-2013 dataset and (b) the Slepemapy dataset

- forgetting: repeated time gap + sequence time gap + past trial counts
- **forgetting+3**: forgetting features + wins + fails + question type
- **forgetting+5**: forgetting+3 + previous attempt response time + difference in previous attempt response time

The variations of the ABIDKT architecture are as follows:

- **ABIDKT-SP**: The parameters of the attention network and bi-interaction layer are shared between the knowledge state estimation step and the future performance prediction step similar to the BIDKT architecture
- **ABIDKT**: The parameters of the bi-interaction layer are shared between the knowledge state estimation step and the future performance prediction step, while the parameters of the attention network are trained independently

Figures 2(a) and 2(b) show the average test AUC results across 5 folds, for different combinations of features and being incorporated in different architectures. From the results we can observe that sharing trainable parameters between the knowledge state estimation step and future performance prediction step (ABIDKT-SP) does not have any significant impact on the performance, although not sharing parameters (ABIDKT) does perform marginally better when the number of features are increased for both datasets. The main takeaways from the results are as follows:

Hyperparameter Tuning and Reproducibility. All baselines were reproduced and their hyperparameters were tuned using the same methodology as for the proposed model. We found that our tuning method led to an AUC improvement of 0.7% for both models for the Assistments 2012-2013 dataset compared to values stated in [6]. For Slepemapy.cz, while DKT could be reproduced, we could not match the AUC for BIDKT, primarily because the batch size parameter mentioned in the paper was too small and computation proved very time consuming. On the other hand, while setting a larger batch size led to a more reasonable runtime, the model saw a 1.1% drop in AUC.

Effect of Additional Context Features. Including wide context features led to improvements in AUC for both BIDKT and ABIDKT models and on both datasets, suggesting that identified features encapsulated information indicative of future student performance. Also, in support of hypothesis 1 stated in Section 5, the extent of improvement in BIDKT tapered off and there was negligible change when number of features was increased from 5 to 8.

Effect of Attention Layers. Contrary to hypothesis 2, adaptive learned context weights in the form of attention layers did not provide a substantial improvement in model performance, instead consistently achieving an AUC 0.1% less than its BIDKT counterpart. This may be because the added context features are not large in number, and attention layers involve more trainable parameters. Trained on the same amount of data, the benefit from fewer trainable parameters in BIDKT outweighs the adapative weight assignments learned by attention layers.

We conducted further analysis by computing the micro-AUC by binning the predictions based on past trial counts as shown in Figure 3 and computing the percentage improvement of the ABIDKT architecture over the BIDKT architecture for each bin. The bin sizes were chosen so as to balance the number of samples in each bin. This analysis was performed on the Assistments 2012-2013 dataset as this is a Mathematics tutor dataset where each user is bound to have a large number of trials. From these results we can observe that for low trial counts, ABIDKT does not show an improvement over BIDKT, but as the number of trials increase, the percentage improvement of ABIDKT over BIDKT also steadily increases for all sets of features. This shows that ABIDKT may be useful in databases where each student has a large number of trials.

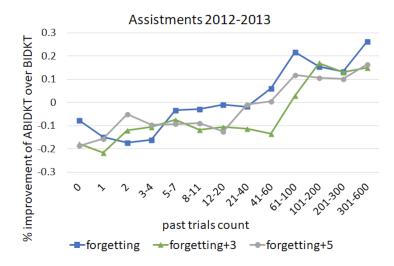


Figure 3: Percentage improvement in micro-AUC of the ABIDKT architecture over the BIDKT architecture on the Assistments 2012-2013 dataset for different feature sets. The AUC scores have been computed by binning the predictions based on past trial counts.

7. Conclusion

The focus of this paper was to observe the effect of using a wider range of context features in the BIDKT model and to propose techniques to effectively incorporate them. We first identified a wider set of context features and incorporated them in the BIDKT model. Experimental results on 2 datasets showed that increasing the number of context features improves the performance of BIDKT significantly, but the performance begins to taper off as the number of features are increased from 5 to 8. We postulated that this could be because of important feature interactions being diluted with other unimportant ones. To overcome this drawback, we proposed a technique that adaptively learns the weights of feature interactions and incorporated this as an attention layer in the BIDKT model. Experimental results of these models on 2 datasets show that this weighting technique was not sufficient to improve the performance of our models. This could be because we were trying to learn additional parameters using the same amount of data. We therefore analyzed the performance of our model across different trial counts and found that our model does outperform BIDKT when the number of past trial counts are high.

In future work, we first plan to implement our models on datasets that have a higher number of trials counts per student. We also plan to modify the architecture of Attention and see if this can perform better than the ABIDKT model. Additionally, we plan to try these approaches in an architecture where skill is modeled separately as in Dynamic key-value Memory Networks for Knowledge Tracing.

References

- G. Adomavicius, B. Mobasher, F. Ricci, A. Tuzhilin, Context-aware recommender systems, AI Magazine 32 (1) 67–80. URL: https://aaai.org/ojs/index.php/aimagazine/article/view/ 2364. doi:10.1609/aimag.v32i3.2364.
- [2] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, J. Sohl-Dickstein, Deep knowledge tracing, in: Advances in neural information processing systems, 2015, pp. 505–513.
- [3] A. T. Corbett, J. R. Anderson, Knowledge tracing: Modeling the acquisition of procedural knowledge, User modeling and user-adapted interaction 4 (1994) 253–278.
- [4] M. Khajah, R. V. Lindsey, M. C. Mozer, How deep is knowledge tracing?, arXiv preprint arXiv:1604.02416 (2016).
- [5] M. V. Yudelson, K. R. Koedinger, G. J. Gordon, Individualized bayesian knowledge tracing models, in: International conference on artificial intelligence in education, Springer, 2013, pp. 171–180.
- [6] K. Nagatani, Q. Zhang, M. Sato, Y.-Y. Chen, F. Chen, T. Ohkuma, Augmenting knowledge tracing by considering forgetting behavior, in: The World Wide Web Conference, 2019, pp. 3101–3107.
- [7] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T.-S. Chua, Attentional factorization machines: Learning the weight of feature interactions via attention networks, arXiv preprint arXiv:1708.04617 (2017).
- [8] L. Zhang, X. Xiong, S. Zhao, A. Botelho, N. T. Heffernan, Incorporating rich features

into deep knowledge tracing, in: Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale, 2017, pp. 169–172.

- [9] J. Zhang, X. Shi, I. King, D.-Y. Yeung, Dynamic key-value memory networks for knowledge tracing, in: Proceedings of the 26th international conference on World Wide Web, 2017, pp. 765–774.
- [10] G. Abdelrahman, Q. Wang, Knowledge tracing with sequential key-value memory networks, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 175–184.
- [11] S. Minn, M. C. Desmarais, F. Zhu, J. Xiao, J. Wang, Dynamic student classiffication on memory networks for knowledge tracing, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2019, pp. 163–174.
- [12] S. Shen, Q. Liu, E. Chen, H. Wu, Z. Huang, W. Zhao, Y. Su, H. Ma, S. Wang, Convolutional knowledge tracing: Modeling individualization in student learning process, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1857–1860. URL: https://doi.org/10.1145/3397271.3401288.
- [13] S. Yang, M. Zhu, J. Hou, X. Lu, Deep knowledge tracing with convolutions, 2020. arXiv:2008.01169.
- [14] C.-K. Yeung, Deep-irt: Make deep learning based knowledge tracing explainable using item response theory, arXiv preprint arXiv:1904.11738 (2019).
- S. Pandey, G. Karypis, A self-attentive model for knowledge tracing, CoRR abs/1907.06837 (2019). URL: http://arxiv.org/abs/1907.06837. arXiv:1907.06837.
- [16] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.
- [17] S. Rendle, Factorization machines, in: 2010 IEEE International Conference on Data Mining, IEEE, 2010, pp. 995–1000.
- [18] Z. Pan, E. Chen, Q. Liu, T. Xu, H. Ma, H. Lin, Sparse factorization machines for clickthrough rate prediction, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016, pp. 400–409.
- [19] J. Pan, J. Xu, A. L. Ruiz, W. Zhao, S. Pan, Y. Sun, Q. Lu, Field-weighted factorization machines for click-through rate prediction in display advertising, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1349–1357.
- [20] N. Gui, D. Ge, Z. Hu, Afs: An attention-based mechanism for supervised feature selection, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 3705–3713.
- [21] B. Liu, C. Zhu, G. Li, W. Zhang, J. Lai, R. Tang, X. He, Z. Li, Y. Yu, Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction, arXiv preprint arXiv:2003.11235 (2020).
- [22] X. Sun, X. Zhao, Y. Ma, X. Yuan, F. He, J. Feng, Muti-behavior features based knowledge tracking using decision tree improved dkvmn, in: Proceedings of the ACM Turing Celebration Conference-China, 2019, pp. 1–6.
- [23] S. Pandey, J. Srivastava, Rkt: Relation-aware self-attention for knowledge tracing, arXiv preprint arXiv:2008.12736 (2020).

- [24] J.-J. Vie, H. Kashima, Knowledge tracing machines: Factorization machines for knowledge tracing, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 750–757.
- [25] M. Feng, N. Heffernan, K. Koedinger, Addressing the assessment challenge with an online system that tutors as it assesses, User Modeling and User-Adapted Interaction 19 (2009) 243–266. URL: https://doi.org/10.1007/s11257-009-9063-7. doi:10.1007/ s11257-009-9063-7.
- [26] J. Papousek, R. Pelánek, V. Stanislav, Adaptive geography practice data set, Journal of Learning Analytics 3 (2016) 317–321. URL: https://doi.org/10.18608/jla.2016.32.17. doi:10. 18608/jla.2016.32.17.