Self-attention in Knowledge Tracing: Why It Works

Shi Pu(✉) and Lee Becker

Educational Testing Service, 660 Rosedale Road, Princeton, NJ 08540, USA
{spu,lbecker001}@ets.org

Abstract. Knowledge tracing refers to the dynamic assessment of a learner's mastery of skills. There has been widespread adoption of the *self-attention* mechanism in knowledge-tracing models in recent years. These models consistently report performance gains over baseline knowledge tracing models in public datasets. However, why the *self-attention* mechanism works in knowledge tracing is unknown.

This study argues that the ability to encode when a learner attempts to answer the same item multiple times in a row (henceforth referred to as *repeated attempts*) is a significant reason why *self-attention* models perform better than other deep knowledge tracing models. We present two experiments to support our argument. We use context-aware knowledge tracing (AKT) as our example *self-attention* model and dynamic key-value memory networks (DKVMN) and deep performance factors analysis (DPFA) as our baseline models. Firstly, we show that removing repeated attempts from datasets closes the performance gap between the AKT and the baseline models. Secondly, we present DPFA+, an extension of DPFA that is able to consume manually crafted repeated attempts features. We demonstrate that DPFA+ performs better than AKT across all datasets with manually crafted repeated attempts features.

Keywords: Deep knowledge tracing · Self-attention · Knowledge tracing

1 Introduction

Knowledge tracing refers to the dynamic assessment of a learner's mastery of skills. It is usually used with a recommendation policy to achieve personalized learning. For example, in mastery learning, a learner is only permitted to move on to the next skill when a knowledge-tracing model estimates the learner has achieved mastery of the prerequisite skills [10].

A knowledge-tracing task is structured as a sequential prediction problem in which a knowledge-tracing model tries to predict whether a learner will be able to correctly answer the next item given their past item responses. Throughout this paper, we use "item" to refer to a question or exercise that a learner completes and "item response" to refer to the correctness of the learner's answer to the item.

M. M. Rodrigo et al. (Eds.): AIED 2022, LNCS 13355, pp. 731–736, 2022.
https://doi.org/10.1007/978-3-031-11644-5_75

While early knowledge-tracing models are relatively straightforward [2,5,6], modern models are structured as complicated deep neural networks [1,4,9,11,12]. A family of these deep neural networks has adopted the *self-attention* mechanism from the natural language processing (NLP) community and shown remarkable success in specific datasets.

What is *self-attention*? Deep neural networks represent a learner's item response (or an item) as a multidimensional vector. This vector is referred to as the embedding of an item response. This embedding is a blend of model-readable (not human-readable) features. Hypothetically, these features include the skill associated with the item, the difficulty of the item, and the mastery of skills from the item response, among other aspects. The *self-attention* mechanism models a learner item response embedding as a weighted sum of previous learner item response embeddings, followed by a non-linear transformation so that the item response embedding is a non-linear function of the previous item response embeddings.

While *self-attention* offers a remarkable performance boost to knowledge-tracing models, why the mechanism works is unknown. The *self-attention* mechanism originated from the NLP community, whose scientists seek to dynamically model a word's (or part of a word's) embedding (i.e., a multidimensional vector representing a word's semantic and syntactical features) based on its context. For example, "bank" may refer to a financial institution or a river bank depending on the context. Self-attention allows a word's embedding to change based on the other words in the sentence or paragraph.

Likewise, *self-attention* in deep knowledge tracing allows the embedding of a learner's response to change based on the learner's previous responses to items (because it is a non-linear function of previous item responses). However, this flexibility appears unnecessary for knowledge tracing, given that the skill associated with an item, the difficulty of an item, and the mastery of skills from a correct/incorrect item response all appear to be context-independent.

Therefore, we are interested in uncovering what context-dependent features *self-attention* extract from the data. One of the most promising candidates is the number of repeated attempts made on an item. For example, Ghosh et al. [4] reported AKT significantly outperformed deep knowledge tracing (DKT) [7], a recurrent neural network knowledge-tracing model, on ASSISTments 2017 where repeated attempts are widespread. Using the same dataset, Gervet et al. [3] documented the self-attentive knowledge-tracing (SAKT) model to perform similarly to DKT after repeated attempts were removed.

There is theoretical grounding to believe the number of repeated attempts is a valuable feature for knowledge tracing. First, how well a learner masters a skill is affected by how soon the learner answers an item correctly. In addition, items in public datasets are often multiple-choice questions, making the second or third attempt easier than the first attempt. Finally, learning systems often provide feedback after an unsuccessful trial (e.g., coding practices). Accordingly, later attempts at an item are significantly more likely to succeed than first attempts.

In the rest of the paper, we provide evidence that the ability to encode repeated attempts is a significant reason why *self-attention* models perform better than other deep knowledge tracing models. We demonstrate that AKT outperforms strong baseline models when there are considerable repeated attempts in datasets but that AKT performs on par with those same baseline models when the repetitions are removed. We then manually encode the number of repeated attempts as a feature into a strong baseline, showing that the baseline outperforms AKT in all datasets where repeated attempts are present.

Table 1. Data statistics

Datasets	Learners	Items	Responses	% of multiple attempts	% of repeated attempts
ASISSTments 2017	1709	4117	942K	58.36	52.12
CSEDM 2019	87	35	2.77K	68.20	66.37
Statics 2011	316	987	205K	34.23	31.62

2 Experiment Setup

We choose three public datasets for our experiments. These datasets provide item identifiers and timestamps for all item responses. More importantly, all the datasets have multiple attempts[1] and repeated attempts. Table 1 reports the descriptive statistics for each dataset. The details of the datasets are as follows:

ASSISTments 2017.[2] This dataset records secondary school students' answers to math problems using the ASSISTments tutoring system.

STATICS 2011.[3] This dataset is from an engineering course on statics in fall 2011. We follow [8] for data preprocessing. Unlike previous studies that keep only students' first responses to items, this study keeps all responses.

CSEDM 2019.[4] This dataset is from a study of novice programmers working with the ITAP intelligent tutoring system. We remove all hints in the data.

For each dataset, we evaluate a model with student-stratified five-fold cross-validation. For each fold, 60% of students are used as training data, 20% students are used as validation data, and 20% of students are used as test data. We use the average test area under the curve (AUC) as the evaluation metric.

We set the maximum length of a response sequence to be 200. Learner response sequences shorter than 200 are padded with 0 to the left, and sequences longer than 200 are folded. We used the hyperparameter reported by the original authors when possible. If the model has not been experimented with a dataset before, we use the validation data for hyperparameter tuning.

[1] Multiple attempts is different from repeated attempts that they may or may not happen consecutively.

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

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

[4] https://pslcdatashop.web.cmu.edu/Files?datasetId=2865.

Table 2. Test AUC w/o repeated attempts

Datasets	ASSISTments 2017		CSEDM 2019		Statics 2011	
	With repeat	No repeat	With repeat	No repeat	With repeat	No repeat
AKT	**0.7620**	0.7862	**0.7602**	0.7720	**0.7958**	**0.8059**
DKVMN	0.7217	**0.7917**	0.7342	0.7843	0.7867	0.7951
DPFA	0.7167	0.7885	0.7412	**0.8031**	0.7795	0.7930

3 Experiment 1: Repeated Attempts and Self-attention

If having a representation of "repeated attempts" explains why self-attention models work better than other KT models, then we should observe *self-attention* models outperforming baseline models in a dataset with repeated attempts. More importantly, we should observe the closure of the performance gap if we remove repeated attempts from a dataset by keeping only learners' first item responses.

We choose AKT as our example self-attention model. We choose DKVMN and DPFA [8] (a modern version of performance factors analysis [5]) as our baselines. Neither baseline uses *self-attention*, and both baselines have outstanding performances on multiple public datasets.

We experiment with the models on two versions of the dataset, one that includes repetition and the other that excludes repetition by keeping only a learner's first response to an item. Table 2 shows the performance of the models on both versions of the dataset. Across all datasets, AKT outperforms the baseline models when we keep the repeated attempts in the data. The performance gap is most evident in ASSISTments 2017. However, if we remove the repeated attempts from the data by keeping only learners' first attempts, AKT's performances is very close to the baselines' performances. AKT even underperforms in a small dataset like CSEMD 2019. The result suggests that the presence of repeated attempts is crucial for AKT to achieve its performance potential.

4 Experiment 2: Repeated Attempts as Features

We present DPFA+, which is an extension of DPFA [8]. When predicting whether a learner can correctly answer the next question, DPFA considers the difficulty of the next item and the number of similar items the learner has answered correctly or incorrectly in the past. DPFA+ extends the model by incorporating model features for number of attempts made on previous items.

4.1 Model Specification

The original DPFA is a logistic regression:

$$p_{t+1} = \sigma(\beta_{t+1} + \sum_{i} w_i v_i) \quad \forall i \leq t \tag{1}$$

Table 3. Test AUC for DPFA+

Datasets	ASSISTments 2017	CSEDM 2019	Statics 2011
AKT	0.7620	0.7602	0.7958
DKVMN	0.7217	0.7342	0.7867
DPFA	0.7167	0.7412	0.7795
DPFA+	**0.7794**	**0.7932**	**0.8117**

where β_{t+1} represents the next item difficulty, v_i represents the estimated mastery of skill from past item response i, and w_i represents the relevance of item $t+1$ and the item i. In DPFA, β_{t+1} and v_i are functions of items and correctness of item response. DPFA+ argues that the number of proceeding success and failure on the item should also be part of the function parameters. Specifically, in DPFA+:

$$\beta_{t+1} = W_2(tanh(W_1[d_{t+1} \oplus s_t \oplus f_t] + b_1)) + b_2 \tag{2}$$

$$v_i = W_4(tanh(W_3[d_i \oplus s_i \oplus f_i] + b_3)) + b_4 \tag{3}$$

where d_i is the one hot encoding of item i, s_i is the number of successes in the repeated attempts, f_i is the number of failures in the repeated attempts. For example, if a student attempt an item k three times in a roll with $[success, failure, success]$, the values of s_k are $[1, 1, 2]$ and values of f_k are $[0, 1, 1]$. Note that when modeling the next item difficulty β_{t+1}, only *proceeding* success s_t and *proceeding* failures f_t are visible. $\oplus$ represents the concatenation operation. W_* and b_* are learned parameters. DPFA+ is identical to DPFA in other aspects. We therefore recommend readers to refer to the original paper [8] for model details due to page limit.

4.2 Results

Table 3 presents the average AUC of the DPFA+ model on the test data. DPFA+ performs better than the original DPFA model with the manually encoded repeated attempts features. More importantly, DPFA+ outperforms *self-attention* model AKT. This suggests that while a *self-attention* model like AKT can encode repeated attempts as part of its item embedding, the mechanism is not as effective as manually crafted features.

5 Discussion

In this study, we presented evidence that the ability to encode repeated attempts as features is the reason why *self-attention* models perform better than other deep knowledge models in datasets where such repetition is common.

Repeated attempts are common in systems where a learner can attempt an item multiple times (e.g., coding practice). The ability to model whether a learner can succeed on the nth attempt is useful for personalized learning systems. For example, a system may decide to provide scaffolding only if a learner has already failed a few times and the knowledge-tracing model believes the learner is not likely to succeed on the next attempt. In such situations, DPFA+ is a better choice than *self-attention* models due to its better performance and higher interpretability.

References

1. Choi, Y., et al.: Towards an appropriate query, key, and value computation for knowledge tracing. arXiv preprint arXiv:2002.07033 (2020)
2. Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Model. User-Adap. Inter. **4**(4), 253–278 (1994)
3. Gervet, T., Koedinger, K., Schneider, J., Mitchell, T., et al.: When is deep learning the best approach to knowledge tracing? J. Educ. Data Mining **12**(3), 31–54 (2020)
4. Ghosh, A., Heffernan, N., Lan, A.S.: Context-aware attentive knowledge tracing. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2330–2339 (2020)
5. Pavlik Jr, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis-a new alternative to knowledge tracing. Online Submission (2009)
6. Pelánek, R.: Applications of the ELO rating system in adaptive educational systems. Comput. Educ. **98**, 169–179 (2016)
7. Piech, C., et al.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems, pp. 505–513 (2015)
8. Pu, S., Converse, G., Huang, Y.: Deep performance factors analysis for knowledge tracing. In: Roll, I., McNamara, D., Sosnovsky, S., Luckin, R., Dimitrova, V. (eds.) AIED 2021. LNCS (LNAI), vol. 12748, pp. 331–341. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78292-4_27
9. Pu, S., Yudelson, M., Ou, L., Huang, Y.: Deep knowledge tracing with transformers. In: Bittencourt, I.I., Cukurova, M., Muldner, K., Luckin, R., Millán, E. (eds.) AIED 2020. LNCS (LNAI), vol. 12164, pp. 252–256. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52240-7_46
10. Ritter, S., Yudelson, M., Fancsali, S.E., Berman, S.R.: How mastery learning works at scale. In: Proceedings of the Third (2016) ACM Conference on Learning@ Scale, pp. 71–79 (2016)
11. Shin, D., Shim, Y., Yu, H., Lee, S., Kim, B., Choi, Y.: Saint+: integrating temporal features for ednet correctness prediction. In: LAK21: 11th International Learning Analytics and Knowledge Conference, pp. 490–496 (2021)
12. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: Proceedings of the 26th international conference on World Wide Web, pp. 765–774 (2017)