Knowledge Tracing Using Transformer

Ruiwei Xiao, Yuxuan Tao, Huiyi Tang, Shuman Wang

Abstract

In recent decades, intelligent tutoring systems (ITS) have benefited an increasing group of students and educational practitioners for their adaptive tutoring advice and question recommendation. Knowledge tracing, as one of the most significant processes in ITS that determines if the system is able to recommend appropriate questions to students, draws a great number of researchers' attention. In this research, we focused on how to model the KT problem using transformers, how to optimize the accuracy and interpretation of the prediction result, and whether the transformer outperforms other machine learning models. The preliminary results are 1) our transformer achieved 71.00% AUC within five to ten minutes, and 2) we proposed two approaches to make the transformer model more informative and self-explanatory.

1. Introduction

With an increasing amount of online teaching and learning activities due to the pandemic, the intelligent tutoring system (ITS) has received more attention, as well as higher requirements. ITS provides real-time, personalized question-recommendation and tutoring for each student based on the student's mastery and learning ability (Figure 1). This process is called Knowledge Tracing (KT), and the specific technology that gears KT also evolves following the development of machine learning technology. Originally, reinforcement learning was used to predict student's performance, and the corresponding KT model is called Bayesian Knowledge Tracing (BKT). When deep learning models outperformed the Bayesian models in accuracy, Deep Knowledge Tracing (DKT) model was then implemented. Currently, transformers show high performance in predicting sequential data in both accuracy and efficiency, therefore, we became interested in implementing the Transformer Knowledge Tracing (TKT) model and observing its performance.

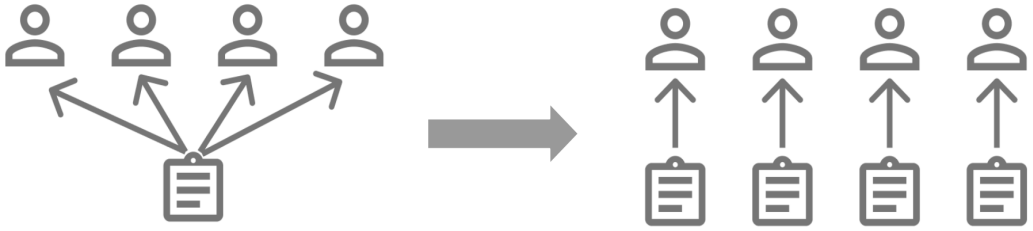

Figure 1. Personalizing Assignment for Each Student

In our work, we would like to narrow down the KT problem to a specific question set. In other words, we would like to predict how the student will perform on the next question on the basis of the student's previous performance. More specifically, this project primarily focuses on answering the following three questions: 1) Does TKT outperform other KT models in accuracy? 2) How to improve TKT models' performance? and, 3) How to make TKT more informative? This project is adapted from the SAINT model (Choi, Y, et. al, 2020), and our adjustments improved the test result and broadened the model's usability.

2. Background

Knowledge Tracing

Knowledge Tracing (KT)—where a machine models the knowledge of a student as they interact with coursework—is a well-established problem in computer-supported education (Piech et al., 2015). The implementation of the Knowledge Tracing model has evolved through the progress of artificial intelligence technology (e.g. decision tree, Hidden Markov Chain, deep neural network, etc.). Figure 2 shows the process of our specified KT process.

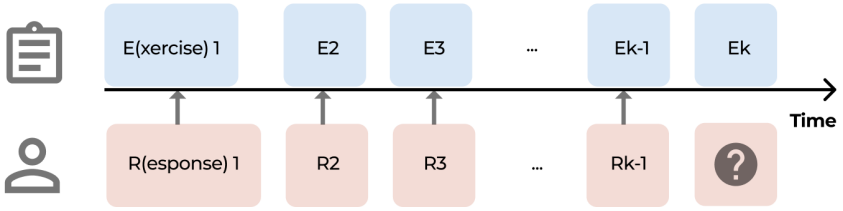

Figure 2. Knowledge Tracing Process

Transformer

In 2017, the Google Brain team proposed an edge-cutting architecture for sequential data modeling called Transformer. Like deep learning models such as recurrent neural networks, transformers also have encoder and decoder; however, in a transformer, they are connected through an attention mechanism. Moreover, compared to Bayesian models and other deep learning models, transformers are able to achieve better performance in both accuracy and time efficiency. According to existing transformers' application on KT problems, it is able to outperform other models by over 10% of accuracy on KT problems (Pu et al., 2020). With solid theoretical and empirical support, we believe that the transformer is a suitable technology to solve KT problems.

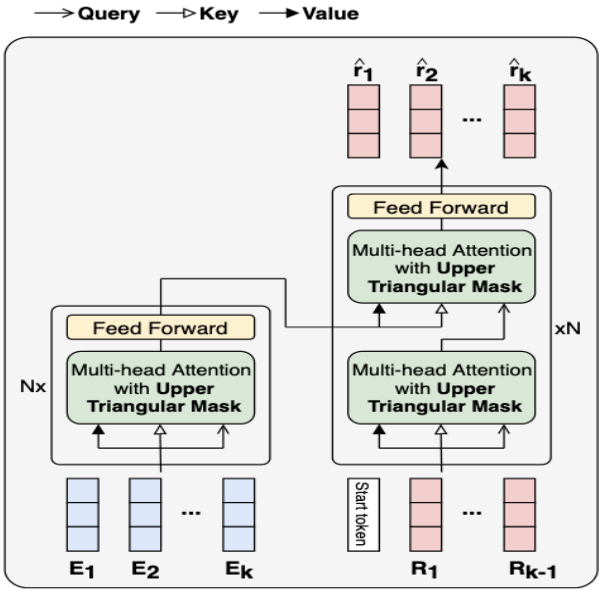

Figure 3. SAINT Model

SAINT (Choi, Y., et. al, 2020)

SAINT (Separated Self-AttentIve Neural Knowledge Tracing) is a transformer-based model for knowledge tracing that has an encoder-decoder structure where the exercise and response embedding sequences separately enter, respectively, the encoder and the decoder (Figure 3). The encoder applies self-attention layers to the sequence of exercise embeddings, and the decoder alternately applies self-attention layers and encoder-decoder attention layers to the

sequence of response embeddings. Separating "interaction" and "exercise" information as two kinds of input allows repeated stacking of attention layers, resulting in an improvement in the area under the receiver operating characteristic curve (AUC). As the authors denoted and our knowledge, SAINT is the first model to suggest an encoder-decoder model for knowledge tracing that applies deep self-attentive layers to exercises and responses separately.

3. Technical Details

Dataset

This research is tested on two datasets: ASSISTments2017 dataset and Ednet dataset. Both datasets capture student's question answering features from online learning platforms. Each row of the dataset is a question-answering record. The ASSISTments2017 dataset has 942, 816 records from 1709 students, most of its features are student's performance related (hint used, time spent, etc.); while the Ednet dataset has 131,441,538 records from 784,309 students, most of its features are question-related (difficulty of the question, skill labels, etc.). Table 1 compares attributes from ASSISTments and EdNet dataset, which may provide insights for data exploration.

Table 1. Comparing ASSISTments, EdNet, and other educational datasets (Choi, Y., et. al, 2020)

	ASSISTments			Syn-5	Junyi	Stat-2011	EdNet			
	2009	2012	2015				KT1	KT2	KT3	KT4
# of students	4,217	46,674	19,917	4,000	247,606	335	784,309	297,444	297,915	297,915
# of questions	26,688	179,999	100	50	722	1,362	13,169	13,169	13,169	13,169
# of tags	123	265	-	5	41	27	188	188	293	293
# of lectures	0	0	0	0	0	0	0	0	1,021	1,021
# of logs	346,860	6,123,270	708,631	200,000	25,925,922	361,092	95,293,926	56,360,602	89,270,654	131,441,538
# of types of logs	3	3	1	1	1	5	1	3	4	13
Public available	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Contents available	No	No	No	No	Yes	Yes	No	No	No	No
From real-world	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Collecting period	1y	1y	1y	-	2y 2m	4m	2y 7m	1y 3m	1y 3m	1y 3m

Input Embedding

In both datasets, we first ruled out columns that depict irrelevant features, and then grouped rows by student id. The grouped data is further organized into Student_Data objects. Within each Student_Data object, the question-answering data are ordered by question-answering time in ascending order.

Similar to other researchers' design in SAINT model's input, we also decided to embed exercise information and student's response separately. Additionally, in order to make full use of information in the dataset and to improve predicting accuracy, we have tried 4 different combinations of features as input embedding as shown in Table 2.

Table 2. Four different combinations of input embedding

Embedding	question_id	skill_id	hint_use	time_use	correctness
#1	√	√	×	×	√
#2	√	√	√	×	√
#3	√	√	×	√	√
#4	√	√	√	√	√

Model Selection

Based on the literature review under the background section, we believe that the transformer is a promising model to model KT problems. Nevertheless, to further validate the literature review result, we tested the ASSISTments2017 dataset on both LSTM (Figure 4-1, a recurrent neural network model) and the original SAINT (Figure 4-2, a transformer model) with Embedding#1 (the most basic input embedding) to figure out which model has better performance.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 1000)              4008000
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 1001
=================================================================
Total params: 4,009,001
Trainable params: 4,009,001
Non-trainable params: 0
```

Figure 4-1. LSTM Model Summary

```
  | Name  | Type  | Params
-------------------------------
0 | model | SAINT | 45.1 M
-------------------------------
45.1 M    Trainable params
0         Non-trainable params
45.1 M    Total params
180.530   Total estimated model params size (MB)
```

Figure 4-2. SAINT Model Summary

4. Experimental Results and Analysis

To answer the questions in the introduction section, we designed and conducted four sets of experiments as follows. The running environment is Google Colab with GPU accelerator and high-RAM mode.

1) RNN (LSTM) vs Transformer

We tested ASSISTments2017 dataset on an LSTM model and a transformer model with the same input features (embedding#1), and the test result is in table 3. From the table, we can know that 1) the AUC of LSTM is over 2% higher than that of a transformer; 2) the time for training a transformer is only 1/12 to 1/6 of the total time for training an RNN model.

Table 3. RNN vs. Transformer

	AUC	Epochs	Time
LSTM	0.7274	50	About an hour
Transformer	0.7041	1	About 5-10 minutes

Although the AUC values indicate that LSTM is slightly more accurate than the transformer in predicting KT problems, the transformer's significant advantage in time efficiency may make up for the minor gap in accuracy.

2) Different Inputs Combinations

After examining two different models, we decided to optimize the transformer from multiple aspects. Since we believed that providing more information to the model may be helpful for improving accuracy, we tried to feed the same model (with upper mask, MAX_SEQ_LEN=500, n_encoder=n_decoder=6, n_heads=8) four different input embeddings. The results are shown in table 4.

Table 4. Accuracy of Different Input Embeddings

	Input	AUC
Embedding#1	question_id, skill_id, correctness	0.7041
Embedding#2	question_id, skill_id, hint_use, correctness	0.7100
Embedding#3	question_id, skill_id, time_use, correctness	0.6890
Embedding#4	question_id, skill_id, hint_use, time_use, correctness	0.6917

From table 4's result, embedding#2 achieved the best result among 4 types of embeddings, which on the one hand, proves that more information can help on increasing AUC; on the other hand, embedding#4, which carried more information, failed to achieve the highest AUC.

3) Different Masks

The SAINT model's source code provides two types of masks for model building: Upper Triangular Mask and Lower Triangular Mask. By testing both masks on the same model, we obtained the result in table 5.

Table 5. Accuracy of Different Masks

	Description	AUC
Upper	Upper Triangular Mask	0.7100
Lower	Lower Triangular Mask	0.5000

The results in table 5 strongly encouraged us to use the Upper Triangular Mask in the further steps. It is noticeable that the AUC of Lower Triangular Mask equals random guess

result, however, the implementation of Upper Triangular Mask and Lower Triangular Mask is mirrored, and we simply used torch.triu() and torch.tril() that provided by PyTorch library, such result is unlikely resulted from mistakes during implementation.

4) Hyper-Parameter Tuning

Lastly, we tried to optimize the value of MAX_SEQ_LEN and enc_heads / dec_heads. The model in this step takes embedding#2 as input and uses the Upper Triangular Mask as mask. The results of this step are shown in table 6. Accuracy of Different Parameter Values.

Table 6. Accuracy of Different Parameter Values

Parameter	Value	AUC
MAX_SEQ_LEN	100	0.5302
	200	0.5945
	500	0.7100
enc_heads / dec_heads	1	0.6779
	8	0.7100

The result shows that 1) Longer input sequence (MAX_SEQ_LEN=500) can achieve better AUC; 2) multi-heads mechanism does help in improving AUC.

The best-performance model has been saved in "model_saint.pt".

5. Discussion

The results in the last section lead to the answer to the three questions that we proposed in the introduction.

1) Does TKT outperform other KT models in accuracy? According to our literature review, transformers seem to be the best model for predicting KT problems. However, in our experiment settings, the DKT (LSTM) model outperformed transformers in AUC by more than 2%, although transformers are always significantly faster than the DKT model.

2) How to improve TKT models' performance? The AUC would increase when the transformer model is more informative. Nevertheless, some features can be so distractive that lead to decreasing AUC. In our experiment, adding hint_use to the input led to AUC increasing while adding time_use had the opposite result. This may be caused by ignorance in data preprocessing since we should compare the time used to other students who answered the same question instead of using the value directly. As for hyper-parameters, when Input = Embedding#2, mask = Upper Triangular Mask, MAX_SEQ_LEN = 500, enc_heads = dec_heads = 8, the transformer achieved the best AUC.

3) How to make TKT more informative? Comparing the result of input embedding#1 to #4, we would know that one way to make the model more informative is to appropriately embed more features into the input tensor. Another way is to implement an explainer layer for the model. Similar to deep learning models, transformers are black boxes

to users. In 2021, Janizek, J. D., Sturmfels, P., & Lee, S. I., proposed a design that visualizes the internal correlation between input elements by calculating Integrated Gradients (figure 5). This implementation is currently working on BERT, and we believe that we can integrate such ideas into KT models as well.

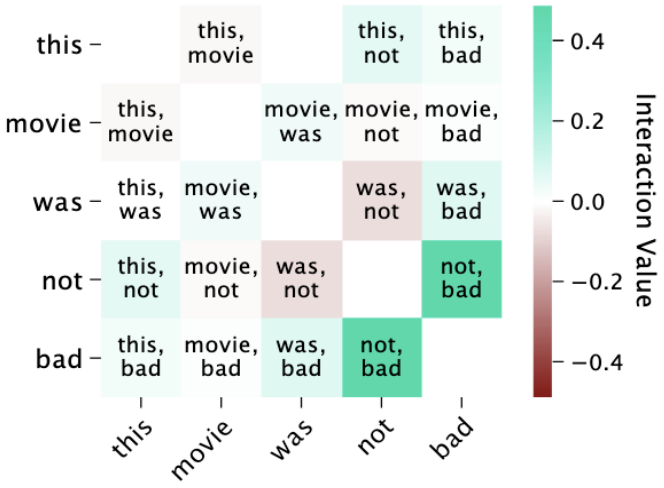

Figure 5. Visualization of Explaining Explanations

6. Future Work

There are several ideas that have been proposed but not finished yet due to the shortage of time or relevant experiences, and we intend to implement them in the future.

We have tested four different combinations of input embedding on two datasets. Each time we switch to a different embedding, we have to edit the internal structures of several classes of the transformer model, which is quite demanding especially when we want to reproduce a result that we generated before. Thus, in the future, we want to provide a uniform interface in that developers can specify what kind of embedding they want to reduce coupling.

Another feature that we are really interested in is the interpreter. This feature can reveal more relationships between the prediction and previous input, therefore providing more insights to both students and educational practitioners.

7. Contributions

In this paper, we have three major contributions. Firstly, by doing the literature review and some experiments, we compared the performance of different knowledge tracing models implemented by different techniques (reinforcement learning, LSTM, Transformer), and concluded that the transformer is a promising model for KT problems. Secondly, we optimized the accuracy of the transformer by modifying the input embedding, using different masks within the model, and tuning parameters. Thirdly, inspired by interpreters for language models, we proposed a design to optimize the transformer for KT problems to a more interpretable form.

All the code and data for this project can be found in Appendix Section A: Source Code; more detailed contributions of each team member are shown in Appendix Section B: Contributions of Group Members.

References

[1] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. Advances in neural information processing systems, 28.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[3] Pu, S., Yudelson, M., Ou, L., & Huang, Y. (2020, July). Deep knowledge tracing with transformers. In International Conference on Artificial Intelligence in Education (pp. 252-256). Springer, Cham.

[4] Choi, Y., Lee, Y., Cho, J., Baek, J., Kim, B., Cha, Y., ... & Heo, J. (2020, August). Towards an appropriate query, key, and value computation for knowledge tracing. In Proceedings of the Seventh ACM Conference on Learning@ Scale (pp. 341-344).

[5] Janizek, J. D., Sturmfels, P., & Lee, S. I. (2021). Explaining explanations: Axiomatic feature interactions for deep networks. Journal of Machine Learning Research, 22(104), 1-54.

[6] Choi, Y., Lee, Y., Shin, D., Cho, J., Park, S., Lee, S., ... & Heo, J. (2020, July). Ednet: A large-scale hierarchical dataset in education. In *International Conference on Artificial Intelligence in Education* (pp. 69-73). Springer, Cham.

[8] Xiao, Z., Li, Z., and Pardos, Z. (2018). AutoQuiz: A Personalized, Adaptive, Test Practice System (Abstract Only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (*SIGCSE '18*). Association for Computing Machinery, New York, NY, USA, 1089. https://doi.org/10.1145/3159450.3162317

[9] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. *Advances in neural information processing systems*, *28*.

Appendix

Section A: Source Code

[1] Github Repo: https://github.com/xiaoruiwei1998/CSE543_Final_Project_TKT

[2] Google Drive (with code and dataset):

https://drive.google.com/drive/folders/1BarPUHLNk7bt4ulbxr42cy19bmbl-rYL?usp=sharing

Section B: Contributions of Group Members

Task	**Details**		**Contributor**
Source Code	**Problem Defining**	NA	Ruiwei Xiao
	Data Preprocessing	Data Cleaning	Yuxuan Tao
		Type Changes	
	Model Building	LSTM (RNN)	Huiyi Tang
		Transformer	Huiyi Tang Ruiwei Xiao
	Optimization Design	Input Embedding	Shuman Wang
		Mask Choosing	Ruiwei Xiao
		Parameter Tuning	Shuman Wang Ruiwei Xiao
Paper	**Section 1, 2, 3, 5, 6, 7**		Ruiwei Xiao
	Section 1, 3, 5, 7		Yuxuan Tao
	Section 1, 4, 5, 7		Huiyi Tang
	Section 1, 4, 5, 7		Shuman Wang