Tracing Knowledge of Student based on Academic Knowledge with Machine Learning and Deep Learning

Jiayu Hu¹, Mingyi Li¹, Huimin Mao¹, Cheng Liu*¹
¹Northwest University, School of information science and technology, Xi’an, China
*Corresponding Author email: lc@nwu.edu.cn

Abstract. Because the number of students not attending school is expanding at an alarming pace, and because of the COVID-19 epidemic, 102 countries have implemented nationwide closures to conduct local shut-downs and temporarily close schools. This slowed down learning possibilities and intellectual growth even more. Every country’s equity disparities may widen. As a result, we must restructure our educational system so that students can gain correct knowledge and teachers can track how much each student has learned. As a result, Machine Learning and Deep Learning are the most effective solutions for this type of problem. As a result, we’re releasing a method for tackling this challenge using Tabnet, Transformers, LGBM, and a variety of other machine learning approaches for student knowledge tracing. Keywords: Deep Learning, PCA, Variance, Self-Attentive Transformers, XGBoost, Catboost, LGBM, Ensembling, Bagging, Boosting and distributions.

1. Introduction
The COVID-19 pandemic from 2020 changed the world overwhelmingly. Specifically, in many countries, it is prohibited or not recommended to communicate face-to-face frequently, including in environments for education. The shutdown of schools and higher education institutions in reaction to the COVID-19 epidemic affected a total of 1.725 billion students worldwide. 192 countries had adopted state-wide closures, according to the UNESCO Monitoring Report see Figure 1. To tackle such situations, some companies have attempted and are attempting to make an AI-tutor, that helps students learn online, by providing questions and lectures that are personalized for each student. But most of them are not able to achieve that goal. So, we are sharing a model for Knowledge Tracing [5] the probability that a certain student can answer a certain question, by using the learning history of the student. Accuracy of the model has been calculated by ROCAUC score [6].

Figure 1 Duration of School Closures in weeks until 25 Jan 2021
Our complete ensemble Model insights can be summarized in 4 points written below:

(i) **LSTM for obtaining query/key/value**, [12] Transformer has had a lot of success with knowledge tracing (e.g. SAKT). However, it has a difficulty with the amount of information that a query/key/value can hold. Because the information from earlier samples cannot be stored in the query/key/value of SAKT, it is difficult to consider patterns such as “If a student answers a vocabulary problem first and then solves a grammar problem, this student is more likely to make a mistake.” To overcome this problem, we used a DKT LSTM model to obtain query/key/value of SAKT-like Transformer.

(ii) **To solve the problem of ‘intercontainer’ leaking, special indexing/masking techniques are used**, a column called ‘container id’ appears in the datasets used for this problem, indicating that some questions are served together. For example, if a student’s container id for four questions is the same, the student must first answer or explain each of the four questions before viewing the answer or explanation. As a result, when training Transformer or RNN variants, some information (e.g., whether the student answered correctly or incorrectly) from samples with the same container id cannot be used. This is referred to as a "intercontainer" leakage problem. We employed clever indexing (Harris et al. 2020) and a modified upper triangular mask similar to the one used in SAINT to overcome this problem [1].

(iii) **Additional hand-crafted features that provide information that Transformer is unable to consider**, when training Transformer, most people set the sequence length and don’t take into account data older than the sequence. With a few outliers. To get over Transformer’s limitation, we devised a straightforward approach that generates additional handcrafted features based on a student’s history. For example, we created a feature that displays “the percentage of times the student’s answer to the question was correct in all of this student’s history”. Because such features contain information that Transformer cannot capture, such features improved performance as expected.

(iv) **LGBM with loop Feature Engineering** To train the model, we utilised LGBM with 14 features and 15 million data points. The dataset is too large to pre-process with a for loop in Python; there are alternative tools and frameworks (SQL, Spark, Apache Beam, Desk) that could make feature engineering much faster, but if we’re smart and develop predictive features, we can get away with only using for loops. Forward feature engineering appears to be a solid strategy to try with this case (build 1 new feature that we think would be predictive based on the situation, run the pipeline, and see if the val score increases; if it does, the feature is predictive, and we should include it). We are concerned when we receive only a tiny improvement; nonetheless, it is sometimes preferable to forego that function because our experimentation process is going to get slower).

2. **Methods**

In this section, we’ll go through the strategies we utilised in our Transformer model as well as the features we employed in our Transformer and LGBM models. We created three sorts of features: query features, memory features, and custom features, the contents of which are listed below.

2.1. **Features**

2.1.1. **Query Feature’s**

These are the features available before [2] students answer the questions. All of the features are listed down below:

- ‘content id’ : the problems have an ID code. We applied 512-dimensional embedding layer to this feature.
- ‘part’ : the TOEIC component of the test For this functionality, we used one-hot encoding.
- ‘tags’ : the question’s tag codes For this functionality, we used one-hot encoding.
- ‘normalized time delta’ : the time interval between the current and previous timestamps. For this feature, we used standardisation.
- ‘normalized log timestamp’ : We applied standardisation to log(1 + timestamp).
• ‘Correct answer’: the answer to the question. For this functionality, we used one-hot encoding.
• ‘normalized container id delta’: the difference between the current and previous container ids, divided by 1000.
• ‘Content type id delta’: the difference between the current content type id and the previous content type id.
• ‘normalized absolute position’: We used normalized position instead of using positional encoding.

2.1.2. Memory Feature’s
The memory features are listed below. Memory features differ from inquiry features in that memory features are available after students have answered the question:
• ‘explanation’: whether or not the Student saw a question explanation after answering the question for this feature, we employed one-hot encoding.
• ‘correctness’: whether the Student’s answer is correct or not. We used one-hot encoding for this feature.
• ‘normalized elapsed time’: the time it took the Student to respond to the inquiry for this feature, we used standardization.
• ‘Student answer’: The Student’s answer to the question. We applied one-hot encoding for this feature.

2.1.3. Additional Feature’s
Additional hand-crafted features are shown below. These features were developed to address the limitation of Transformer-based models, which can never take into account samples older than the sequence length [3]. As an additional input to our model, we created 57 features for input to Transformer model and 14 features for LGBM model.
• ‘answered correctly s count’: The number of correct answers given by the student is counted.
• ‘elapsed_time_s_avg’: average time taken by student to give answer
• ‘explanation_s_avg’: If an explanation for a particular question was available, it would be average.
• ‘elapsed_time_s_sum’: sum of time taken by student to give answer
• ‘timestamp_s_recency_1’ and ‘timestamp_s_recency 2’: the most recent time stamp at which the student gave answer.
• ‘lecture_features’: how many lectures the student saw for each part/tag/type of in the past.
• ‘content_id_features’: how many times the Student’s answer to the content id was accurate and what percentage of the time the Student’s answer was right previously interacted with the content id.

2.2. Techniques on Transformer
2.2.1. Fancy Indexing
We applied MLP to query features, concatenated the output with memory features, and then applied MLP again after getting query and memory features. Then, to avoid the “inter-container” leaking problem, we used LSTM and indexing. For instance, if the container id is 1, the indices are [1, 1, 2, 2, 3] for a certain sample. We use [-2,-2,-1,-1, 0] for indexing. This forbids the use of the same information Id of the job container. As shown in Figure 2. We demonstrated the concept of shifting. After using LSTM, we did indexing.
2.2.2. Transformer and masking
We concatenated the result with query features and applied MLP again after that, then used it as a Transformer query \([4,10]\). Then we combined the Transformer query with memory characteristics, applied MLP again, and used it as key/value for Transformer. We employed a modified upper triangular mask that differed from transformer to avoid ‘inter-container’ leakage. If we have a sample of container id like, \([1,2,3,4,5,6,7,8,9]\) then the mask will be,

\[
\begin{align*}
1,1,1,1,1,1,1,1,1 \\
0,1,1,1,1,1,1,1,1 \\
0,0,1,1,1,1,1,1,1 \\
0,0,0,1,1,1,1,1,1 \\
0,0,0,0,1,1,1,1,1 \\
1,0,0,0,0,1,1,1,1 \\
1,1,0,0,0,0,1,1,1 \\
1,1,1,0,0,0,0,1,1 \\
1,1,1,1,0,0,0,0,1
\end{align*}
\]

3. Experimentation

3.1. LGBM
We used LGBM model with 14 additional Features \([7]\) and 15M data points to train the model. For loop Feature engineering we used other tools and frameworks like (SQL, Spark, Apache Beam, Dask) where you could make feature engineering much faster. Because just using loops can make our training process slow. For considering which Feature is important we ran the pipeline and check if val score increase, if it increase that feature is predictive then we should add it. Because it’s better to discard the features which are less important to us. For the hyper-parameters we use ‘num leaves’ : 200, ‘feature fraction’ : 10, ‘bagging fraction’ : 0.80 and for other parameters we use ‘num boost round’ : 10000, ‘early stopping rounds’ : 10 \([8]\).

Following LGBM training, we discovered that ‘prior question had explanation’ is completely useless for the model, and ‘previous question’ does not appear to be relevant in predicting success on the present question in Figure 3. It also appears that the model is unconcerned with ‘answered correctly avg u’ \([13]\). This could be due to the fact that utilising ‘answered correctly avg u’ is sufficient for making a decision. In Figure 3 we can see a high negative correlation between ‘answered correctly avg c’ and ‘answered correctly avg u’ and positive correlation between prediction and ‘answered correctly avg c’ which means when the model sees an easy question (high ‘answered correctly avg c’) it does not need to look at how good the student is. When it needs to look at harder questions, then it matters to see if the student was doing good previously or not.
3.2. Complete Architecture

We first obtain Query Features and Memory Features and then concatenate the output with Memory Features and again applied MLP. Then, we applied LSTM for avoiding ‘inter-container’ leakage problem which prohibits the use of same ‘container id’ after LSTM we concatenate output with Query Features and applied MLP again. Again we concatenated output for transformer with Memory Features and use it as a key/value. To avoid ‘inter container’ leakage problem we used an upper triangular mask. For the hyper parameters we used ‘max seq’: 100, ‘embed dim’ : 128, ‘dropout’: 0.2, ‘forward expansion’: 1, ‘num layers’: 1, ‘heads’: 8 and after the output of Transformer we weighted averaged with the output of LGBM model on the basis of individual model Binary Cross Entropy Loss [11], [9]as you can see in Figure 4.

4. Conclusion

In this paper, we described the model for 'Knowledge Tracing' of the students based on his previous knowledge. Our method achieved a major improvement compared with the current state-of-the-art paper. Additionally, we pointed out problem of the existing methods and the then proposed the solutions. We believe the insights obtained in this paper help the experts and the community of the AI Education area, and hopefully the students who have to learn online due to the COVID-19 pandemic. With the help of our model and extensive research, every student with an internet connection can receive highquality
education. The most significant advantage of an online courses model is that our classroom and instructor are (theoretically) available 24 hours a day, seven days a week.

References
[1] Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungssoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In Proceedings of the Seventh ACM Conference on Learning@Scale, pages 341–344, 2020a.
[2] Youngduck Choi, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park, Seewoo Lee, Jineon Baek, Chan Bae, Byungssoo Kim, and Jaewe Heo. Ednet: A large-scale hierarchical dataset in education. In International Conference on Artificial Intelligence in Education, pages 69–73. Springer, 2020b.
[3] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860, 2019.
[4] TM Dame. Optimization of moment masking for co spectral line surveys. arXiv preprint arXiv:1101.1499, 2011.
[5] Xinyi Ding and Eric C Larson. On the interpretability of deep learning based models for knowledge tracing. arXiv preprint arXiv:2101.11335, 2021.
[6] Kai Feng, Han Hong, Ke Tang, and Jingyuan Wang. Decision making with machine learning and roc curves. Available at SSRN 3382962, 2019.
[7] Jeff Heaton. An empirical analysis of feature engineering for predictive modeling. In SoutheastCon 2016, pages 1–6. IEEE, 2016.
[8] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In International Conference on Artificial Intelligence and Statistics, pages 1540–1552. PMLR, 2020.
[9] Richard Maclin and David Opitz. Popular ensemble methods: An empirical study. ArXiv e-prints, pages arXiv–1106, 2011.
[10] Nikola Milosevic, Gangamma Kalappa, Hesam Dadafarin, Mahmoud Azimae, and Goran Nenadic. Mask: A flexible framework to facilitate deidentification of clinical texts. ArXiv preprint arXiv:2005.11687, 2020.
[11] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. Journal of artificial intelligence research, 11:169–198, 1999.
[12] Ralf C Staudemeyer and Eric Rothstein Morris. Understanding lstm–a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586, 2019.
[13] Yuxin Sun, Benny Chain, Samuel Kaski, and John Shawe-Taylor. Correlated feature selection with extended exclusive group lasso. arXiv preprint arXiv:2002.12460, 2020.