CS230

Deep Knowledge Tracing for Foreign Language Acquisition

Menglin Liu Brown(mnbrown), Issac Valenzuela(issac23), Blain W Engeda(blain)

1 Introduction

Knowledge Tracing(KT) is a task to predict each student's mastery of learned knowledge on future interactions. It can also suggest new sequences of learning items based on an individual's learning state. Optimizing this task helps students reach their full learning potentials.

We plan to build a knowledge tracing model to recommend personalized word learning sequence for foreign language acquisition purposes.

Using student info and a word, our algorithm will output predictions on whether the student can recall the given word correctly.

2 Related Work

Several models have been used in the prediction of student's future performance based on their past activity, most notably Deep Knowledge Tracing (DKT) and Bayesian Knowledge Tracing (BKT). Deep Knowledge Tracing applies Long Short Term Memory (LSTM) and recurrent neural networks (RNN), and Chris Piech, et al found that this method outperformed other models for all three data sets that they tested on[1]. Several researchers have drawn from these existing models, proposing Dynamic Key-Value Memory Network (DKVMN)[2], Transformer-based models[3], and other knowledge tracing techniques[4]. DKVMN adds to DKT by involving two memory matrices, key and value[2]. This model traces how a student's knowledge changes over time, by allowing the values matrix to be dynamic while holding the keys matrix static[2]. Our project was built off of these existing methods mentioned to specifically build a model which assists in a student's learning of a new language.

3 Dataset

3.1 Initial Analysis

We obtained real student learning data on Duolingo collected from Harvard Dataverse[5]. The raw data contains over 12 million records where each row represents a word that a user was tested on in a particular session. Notable columns include the word being learned, userID, timestamp, counts of the word seen as well as correctness in the past, language being learned, percentage of correct recall on a given word in this session(named p_recall), and etc.

One of the key observations from this dataset is that the data is imbalanced, with 80% labeled as 1(recall correctly) and 20% labeled as 0(recall incorrectly).

The other observation is that each row in the dataset represents a unique word in a given session and accounts for the total times that word was seen in the session. This meant that our labels were based on session performance of the total times users saw that word in that session instead of their score (0/1) after seeing that word just one more time.

CS230: Deep Learning, Spring 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3.2 Data Preparation

Following the initial analysis, we found that we'll need to pre-process the data before training the model. Specifically, we took the following steps to pre-process the data:

1. Label: Derive binary labels from p_recall column where label is 1 when p_recall is 100% and 0 otherwise. We set a high threshold for label 1 based on the data imbalance issue described above.
2. Filtering: 1) Filter by language and use english learning records only. 2) Filter out and discard records of users with only one answer.
3. Factorize the words to encode string values into continuous numerical values.
4. Cross word with answer to form a synthetic cross feature, following the recommendation from literature review Deep Knowledge Tracing (DKT) [1].
5. Generate sequence data grouped by user id to get the learning trace per user.
6. Split the data into three datasets (training, validation and testing).
7. Apply one-hot encoding.
8. Pad the sequences to the same size.
9. Order within each sequence according to the timestamp of response in ascending order

Table 1 shows a summary of the dataset after the above pre-processing steps.

Table 1: Post Pre-Processing Data Summary

# of total user learning sequences	41792
Training set size	26746
Validation set size	6686
Testing set size	8358
Number of words	1739
Cross feature dimension	3478

4 Methods

The problem is formulated as a binary classification problem with the label is defined as $y_i \in \{0, 1\}$ where 0 indicates the student recalls the word incorrectly and 1 indicates the student recalls the word correctly.

We will minimize the cost function of binary cross-entropy loss:

$$\frac{1}{m}\sum_{i=1}^{m} -\left(y_i * \log\left(p_i\right) + \left(1 - y_i\right) * \log\left(1 - p_i\right)\right)$$

where m is the number of training examples.

4.1 Classical Methods

We started with non-RNN models to get a baseline. We first applied logistic regression using features including timestamp, delta since last session, count of history seen, count of history correct, one-hot encoded word, and etc. Since our initial results were poor with logistic regression, besides improving our existing model, we used another classical method, the Random Forest Classifier, which took in the same inputs as the logistic regression model.

4.2 Sequence Models

We now switched to apply deep learning methods, specifically sequence models, because given the nature of the problem where user practice is time series data and there are dependencies and latent relationships among the time-series elements. Our experiments carried out in three phases as discussed in the following:

4.2.1 Phase I: RNN Models and Variants

Inspired by Deep Knowleage Tracing(DKT)[1], which is a pioneering paper to use deep learning for knowledge tracing, we first employed a RNN model using LSTM network. The specific model architecture can be found below:

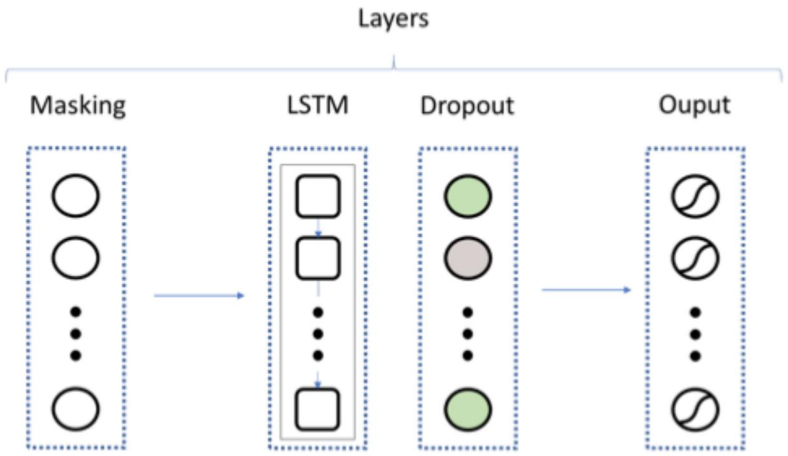


Figure 1: Deep Knowleage Tracing(DKT) Model Architecture [1]

The DKT model consists of four layers, specifically 1) input layer with dimension ($batch\_size$, $sequence\_length$, $cross\_feature\_dimension$) in one-hot encoding of input feature and masked with padding to conform all sequences to the same length ; 2) LSTM layer whose dimension is a hyperparameter; 3) dropout layer to prevent overfitting; 4) output layer whose dimension equals to the number of words using sigmoid as activation function.

We then tried variants of DKT model architecture which replaced LSTM layer with GRU layer.

Although this is a promising method, there are two caveats in the DKT model, which we will address respectively in phase II and phase III.

4.2.2 Phase II: Transfer Learning

One caveat of the DKT method is that the one-hot encoding of input sequences cannot capture the relationships across words in the students' learning history and treats all questions as equally likely with regard to their relationship with each other. Given the context of our problem space of language acquisition performance prediction, we hypothesize the relevance among the words learned by a student should affect how well the student performs.

In effort to address this issue, We used transfer learning to leverage the GloVe pretrained word embedding [1] and modified the DKT model architecture by adding a new embedding layer between the masked input layer and LSTM layer.

4.2.3 Phase III: Attention Mechanism and Transformers

Another caveat of DKT is that it neglects how a student's knowledge state evolves as they practice on different learned words overtime. Such evolving knowledge state can result in the student's recall ability stronger in some words and weaker in some other words. This observation led us to explore transformer architecture to augment DKT model with attention mechanism to learn attention weights of importance of the questions in sequences. Specifically, we tried the following two model architectures:

• Dynamic Key-Value Memory Network (DKVMN)[2]: This model utilizes Long Short-Term Memory Networks (Bi-LSTM) and an additive attention mechanism. It uses the memory matrices: Key(K),

[1]GloVe: Global Vectors for Word Representation by Jeffrey Pennington, Richard Socher, Christopher D. Manning

Value(V) to capture the knowledge state evolvement. It also encodes input as a question and response pair, different from phase II encoding. Details seen in Figure 2.

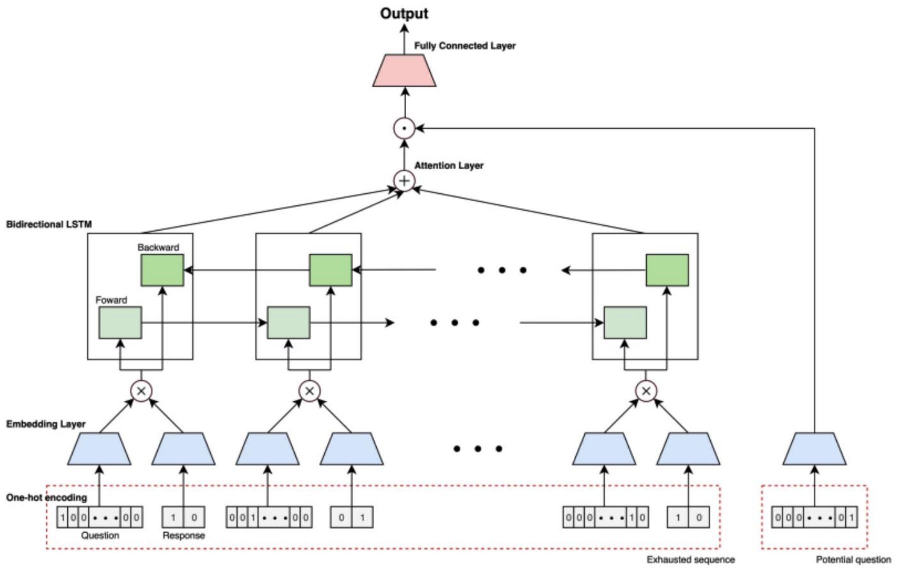


Figure 2: Dynamic Key-Value Memory Network(DKVMN) Model Architecture [2]

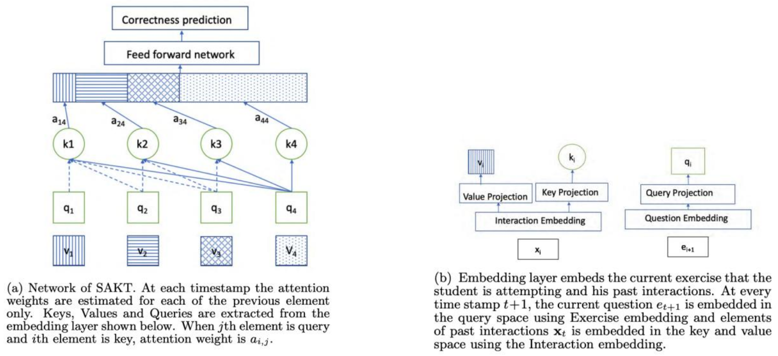
• Self-Attentive Knowledge Tracing (SAKT)[3]: This model was the first to introduce pure attention-based transformer method into KT task. It showed promising results on sparse data. Its main component, the self-attention layer uses the scaled dot-product attention mechanism to learn attention matrices using multi-attention heads, where attention score is defined as:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

5 Experiments/Results/Discussion

5.1 Classical Methods Results

We used Random Forest Classifier as the baseline as it helps address the unbalanced data issue. Our results were initially falsely high, until after conducting feature importance on the model. Here, we



(a) Network of SAKT. At each timestamp the attention weights are estimated for each of the previous element only. Keys, Values and Queries are extracted from the embedding layer shown below. When jth element is query and ith element is key, attention weight is $a_{i,j}$.

(b) Embedding layer embeds the current exercise that the student is attempting and his past interactions. At every time stamp $t+1$, the current question e_{t+1} is embedded in the query space using Exercise embedding and elements of past interactions $\mathbf{x}_t$ is embedded in the key and value space using the Interaction embedding.

Figure 3: Self-Attentive Knowledge Tracing (SAKT) Model Architecture [3]

realized that we were including a feature $session\_correct$ in our input data that indicated whether or not the user actually got the word correct, which is not supposed to be known when the word is asked. Removing this feature allowed us to get reasonable results that showed better performance than the Logistic Regression model. This model suffered an issue of low performance on negative labels with F1 score for negative label of 0.11 and positive label of 0.89.

5.2 Sequence Models Results

The model was trained for 10 epochs with a batch size of 32 on a subset of 10k examples. The AUC steadily increases with each epoch with the following results:

5.2.1 Phase I Result Analysis

In phase I, we experimented two variants of DKT model, one with LSTM layer and the other with GRU layer. The model showed promising result given the state-of-art performance of AUC is around 0.6 - 0.7 based on our literature review.[4] The two variants however showed insignificant difference.

5.2.2 Phase II Results Analysis

We then introduced GloVe pretrained embedding layer to capture the relationship between words. Specifically, the Glove version used is glove.6b.300d which contains 6B tokens, 400K vocab and represented in 300d vector. This showed a worse result than DKT models in Phase I. We hypnotize this was due to the embeddings no longer encode the question-answer which based on literature review[1].

5.2.3 Phase III Results Analysis

In phase III, we moved onto attention models to address the caveats in the previous two phases. They both had a boost from Phase II models, although DKVMN still under-performing DKT model. This indicates simply adding an additive attention mechanism doesn't give much lift.

Table 2: Sequence Model Results (AUC)

Phase	Model	Training Set	Val Set	Test Set	Hyper Parameters
I	DKT w/LSTM	0.6547	0.6473	0.6386	hidden_dim=100
I	DKT w/GRU	0.6410	0.6505	0.6402	hidden_dim=100
II	Transfer Learning w/LSTM	0.6224	0.6322	0.5880	hidden_dim=100
II	Transfer Learning w/GRU	0.6210	0.6303	0.5647	hidden_dim=100
III	DKVMN	0.6326	0.6498	0.6260	key_dim = 50, value_dim = 100
III	SAKT	0.6804	0.6786	0.6715	hidden_dim=100, batch_size=512

6 Conclusion/Future Work

The final winning model is the SAKT model, which combines the embedding layer, question-response pair and multi-head self-attention. This model also supposedly works better on sparse data, in this case the vocabularies are relatively more sparse than math skills dataset.

Future work include leveraging the result from this paper of performance prediction to recommend learning sequence, where the performance prediction can act as a simulator for reinforcement learning recommending algorithms.

Contribution

Menglin:

1. Write ups: large majority(>80%) of the writing including introduction, dataset description, methods explanation, experiment result analysis for proposal, milestone, and final report.
2. Data: large majority(>80%) of data analysis, data pre-processing scripts, including both classical and sequence models.
3. Experiments: implement pre-trained embedding layer into DKT model; train, validate and debug sequence models including DKT, transfer learning, transformer models(DKVMN, SAKT).

Issac:

1. Write ups: Section of proposal, data and baseline model descriptions on milestone, Related Work, Classical Methods, Classical Methods results in final report.
2. Data: Data analysis including running feature importance, visualizing feature importance rankings, to extract cause of faulty high score on Random Forest Classifier. Removed features that were negatively affecting our results.
3. Experiments: Implemented Random Forest Classifier and debugged it including fixing false high evaluation scores. Ran tests on multiple models to get results.

Blain:

1. Write ups: Proposal, milestone, all of video + slides
2. Data: worked on Bert Model and data-preprocessing and analysis for BERT and non-RNN, wrote sorting/ordering script.
3. Worked on data visualizations for non-RNN and Sequence models. Set up AWS account and used to test non-RNN and BERT models.

References

[1] Piech, Chris, et al. “Deep Knowledge Tracing.” Advances in Neural Information Processing Systems, 2015, https://proceedings.neurips.cc/paper/2015/file/bac9162b47c56fc8a4d2a519803d51b3-Paper.pdf.

[2] ABDELRAHMAN, WANG, NUNES, et al. "Knowledge Tracing: A Survey" School of Computing, Australian National University

[3] Pu, Shi, et al. “Deep Knowledge Tracing with Transformers.” Lecture Notes in Computer Science, 30 June 2020, pp. 252–256., https://doi.org/10.1007/978-3-030-52240-7_46.

[4] Zhang, Shuai, et al. “Deep Learning Based Recommender System.” ACM Computing Surveys, vol. 52, no. 1, 2020, pp. 1–38., https://doi.org/10.1145/3285029.

[5] Replication Data for: A Trainable Spaced Repetition Model for Language Learning