Investigating the impact of additional variables on Deep Knowledge Tracing

Othman Bensouda Koraichi
Graduate School of Education
Stanford University
othmanb@stanford.edu

Elena Pittarokoili
Graduate School of Education
Stanford University
elenapit@stanford.edu

Abstract

Knowledge Tracing is the task of modelling student knowledge over time so that we can accurately predict how students will perform on future interactions. Improvement on this task means that questions can be tailored to students based on their individual needs, and content which is predicted to be too easy or too hard can be skipped or delayed. Bayesian Knowledge Tracing, an AI algorithm originally stated by Corbett and Anderson in 1995 [4] , has been outperformed by Deep Knowledge Tracing in 2015 on a dataset containing young learners, in a paper written by Piech et al. [3]. DKT uses RNNs that take sequences of answers to capture more complex representations of student knowledge. In this paper, we first verify whether DKT performs better than BKT on a dataset containing adult learners, and we test different models to assess the improvement provided by additional variables. Our results suggest that a simple DKT model performs better than a hypertuned BKT model, and that some additional variables drastically improve the performance of the DKT algorithm.

1 Introduction

Student knowledge measurement is central in learning environments to provide feedback, evaluate educational programs, set goals and make learning-oriented decisions. This task is inherently difficult as human learning is grounded in the complexity of both the human brain and human knowledge. Thus, the use of complex models seems appropriate. Nevertheless, previous work mainly relies on Hidden Markov models containing several flaws.

Knowledge tracing is the task of modeling student knowledge over time so that it can be predicted how students will perform on future interactions. Each time the student answers an exercise, a prediction is made as to whether or not they would answer an exercise of each type correctly on their next interaction. Deep Knowledge Tracing (DKT) is a subset of Knowledge Tracing in which flexible recurrent neural networks (RNN) are applied for the task of modeling student knowledge. Current applications of DKT models have proved to outperform traditional knowledge tracing methods, such as Bayesian Knowledge Tracing (BKT). This method, first published by Chris Piech et al. in 2015 [3], has proved to outperform traditional knowledge tracing methods. However, RNNs were trained on datasets which only contained binary data about the correctness of answers (e.g whether a user correctly answered (1) or incorrectly answered (0) a question). The latest available literature suggested incorporating several new features to further ameliorate the accuracy of the DKT algorithm. Hence, the core of the project is to write a Deep Learning algorithm that predicts whether a student is going to answer a question correctly within a quiz. This task is important because it allows intelligent

EDUC 259, 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

tutoring systems to calibrate the difficulty of the questions that they will show to students. If a student is shown too many questions that they can't answer, the student will become frustrated, and will stop learning. If a student is shown too many questions that they can answer easily, the student will become uninterested, and will stop learning. While I find replicating Chris Piech's work with our own data interesting, I also want to investigate whether adding other inputs, such as time to answer, skill, sub-skill etc. would further improve the accuracy of the DKT algorithm.

The main contributions of this paper are the following :

- We test the performance of the simplest form of Deep Knowledge Tracing algorithm on new data obtained from adult learners
- We test the importance of multiple variables that correspond to the learner's interaction with the question, e.g. time to answer or to the characteristics of the question e.g. skill, sub-skill, question id etc.
- We compare our proposed models with the baseline i.e. Bayesian Knowledge Tracing model.

  These answers are important because they can inform the companies that develop intelligent tutoring systems, about which algorithm they should use to better predict performance (BKT or DKT), and which data they need to collect in order to feed these algorithms.

These answers are important because they can inform the companies that develop intelligent tutoring systems, about which algorithm they should use to better predict performance (BKT or DKT), and which data they need to collect in order to feed these algorithms.

2 Related work

Modeling students' knowledge is a fundamental part of intelligent tutoring systems. Key aspects of modern intelligent tutoring systems, such as deciding which problems to give students, are reliant upon accurately estimating each student's knowledge state at any given time. Giving students appropriate amounts of practice on each skill promotes complete and efficient learning; both over-practice and under practice can be avoided through having student knowledge models that are as accurate and dependable as possible [5].

Although our focus will be heavily skewed towards DKT, we compare the proposed models with a basic Bayesian Knowledge Tracing (BKT) model. We did not tune the parameters of BKT model, which means that we used a fixed slip and guess rates. Existing methods for determining these parameters are prone to the Identifiability Problem: the same performance data can be fit equally well by different parameters, with different implications on system behavior [2].

Researchers have built new methods for instantiating Bayesian Knowledge Tracing, using machine learning to make contextual estimations of the probability that a student has guessed or slipped [6]. We plan to compare this advanced tuned version of Bayesian Knowledge Tracing models with our proposed DKT models.

Some other authors have advanced the existing Bayesian networks framework by adding individualized as well as skill specific parameters in a single step. The novelty of these models is the ability to learn model parameters with the assumption that different students have different initial background knowledge probabilities [7]. This advanced approach to the traditional Bayesian Knowledge Tracing model is food for thought for us that will guide our next steps to advance our Deep Knowledge Tracing approach.

3 Dataset and Features

3.1 Data Description

We test our ability to predict student performance on a dataset that we specifically got for this research. In particular, we are using data generated from Workera, a platform that offers adaptive assessments for adult learners on specific areas of knowledge (e.g. Mathematics, Data Science, Algorithmic Coding) useful for Data Science related positions. The uniqueness of this dataset not only allows us to reproduce and test previous work, but also to incorporate new inputs which have not been analyzed by other researchers to the best of our knowledge.

Our dataset corresponds to a specific time period from August 1st 2021 to February 2nd 2022 (6 months) and contains 9,767 users, 1,258 questions, 45 different domains (assessments), 137 skills and 1,157 sub-skills. The Data Dictionary (Appendix section) summarizes the content of the dataset used in this paper.

3.2 Binarization of the outcome

As described in the Data Dictionary, the response variable, i.e. the variable defining correctness of answers, has 3 outcomes : 1 for a correct answer, 0 for an incorrect answer, and a null value if the user did not answer. For the purpose of this paper, we binarized this variable and considered that no answer is equivalent to having a wrong answer because the student does not know the correct answer. However, in future work we might consider adding this information as an additional variable that discloses whether the wrong answer was actually a "do not know" answer.

3.3 Standardization

In order to make sure that the weights in the Neural network are not biased, we standardize numeric variables with a MinMax scaler :

$$X_{sc} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

Additionally, we had to use one hot encoding to convert categorical variables into binary variables.

3.4 Vectorization and Padding

In order to train a Deep Learning model on learners' interactions, it is necessary to convert the sequences of questions for each user to fixed length vectors x_n. Padding comes from the need to encode sequence data into contiguous batches: in order to make all sequences in a batch fit a given standard length, it is necessary to pad or truncate some sequences. Each learner has their own vector. This vector has to be of the same length, although the number of questions that a learner has answered might vary. In order to solve this issue, we use padding. Vectors that are shorter than the longest vector are padded with some placeholder value. In our dataset, the most active user answered 480 questions. Thus, we created a vector of responses for each learner and used padding so that each sequence reaches 480 units.

3.5 Train, Validation and Test

To be able to evaluate our proposed models, we split our dataset into training, validation and test. In order to do that, we randomly choose the learners that will be in each set. As described above, each learner has a sequence of questions, so we should not split the dataset by rows - that correspond to learner-question interaction, but rather by learners. Simply put, we should not choose 80% of rows as the training set, but 80% of users. Therefore, we use 80% of the users for the training set, 10% of the users for the validation set and 10% of the users for the test set.

4 Methods

4.1 Bayesian Knowledge Tracing

In recent years, Corbett and Anderson's Bayesian Knowledge Tracing model (BKT) [4] has been used to model student knowledge in a variety of systems. BKT models a learner's latent knowledge state as a set of binary variables, each of which represents understanding or non-understanding of a certain domain. A Hidden Markov Model (HMM) is used to update the probabilities across each of these binary variables, as a learner answers exercises of a given domain correctly or incorrectly. We will briefly explain the BKT algorithm for readers that have never heard of this method.

These are 4 main parameters involved in the Bayesian Knowledge Tracing model :

- P(known): the probability that the student already knew a skill.

- P(will learn): the probability that the student will learn a skill on the next practice opportunity.
- P(slip): the probability that the student will answer incorrectly despite knowing a skill.
- P(guess): the probability that the student will answer correctly despite not knowing a skill.

Every time a student answers a question, the BKT algorithm calculates P(L), the probability that the student has learned the skill they are working on, using the values of these parameters. The formula for P(L) depends on whether their response was correct.

First, we compute the conditional probability that the student learned the skill previously (at time n-1), based on whether they answered the current question (at time n) correctly or incorrectly.

$$P\left(L_{n-1} \mid \text{ Incorrect }_n\right) = \frac{P\left(L_{n-1}\right) * P(S)}{P\left(L_{n-1}\right) * P(S) + \left(1 - P\left(L_{n-1}\right)\right) * \left(1 - P(G)\right)} \tag{2}$$

$$P\left(L_{n-1} \mid \text{ Correct }_n\right) = \frac{P\left(L_{n-1}\right) * \left(1 - P(S)\right)}{P\left(L_{n-1}\right) * \left(1 - P(S)\right) + \left(1 - P\left(L_{n-1}\right)\right) * P(G)} \tag{3}$$

Then, we use the result of the first calculation to compute the conditional probability that the student has learned the skill now (at time n).

$$P\left(L_n \mid \text{ Answer }_n\right) = P\left(L_{n-1} \mid \text{ Answer }_n\right) + \left(\left(1 - P\left(L_{n-1} \mid \text{ Answer }_n\right)\right) * P(T)\right) \tag{4}$$

For the subsequent question, we use P(L) as the new value of P(known). Once P(known) chosen threshold, we say that the student has achieved mastery.

4.2 Deep Knowledge Tracing

Deep Knowledge Tracing consists in applying Recurrent Neural Networks to the problem of predicting student responses to exercises based upon their past activity.

Recurrent Neural Networks (RNN) are feed-forward neural networks that focus on modeling sequence data. The distinctive feature of RNNs is their ability to send information over time steps.

RNNs map an input sequence of vectors x_1 , . . . , x_t , to an output sequence of vectors y_1 , . . . , y_t . This is achieved by computing a sequence of 'hidden' states h_1,...,h_t which can be viewed as successive encodings of relevant information from past observations that will be useful for future predictions. [3]

$$\mathbf{h}_t = \tanh\left(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h\right) \tag{5}$$

$$\mathbf{y}_t = \sigma\left(\mathbf{W}_{yh}\mathbf{h}_t + \mathbf{b}_y\right) \tag{6}$$

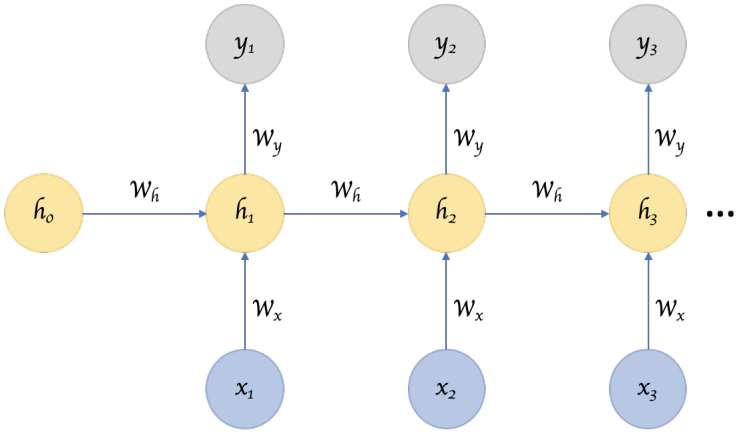

Figure 1: Unrolled Recurrent Neural Network

LSTMs are a more complex variant of RNNs. We choose to apply an LSTM model rather than a vanilla RNN because LSTMs often prove more powerful and avoid the vanishing gradient problem that RNNs suffer from. In LSTMs latent units retain their values until explicitly cleared by the action of a 'forget gate'. They thus more naturally retain information for many time steps, which is believed to make them easier to train. Additionally, hidden units are updated using multiplicative interactions, and they can thus perform more complicated transformations for the same number of latent units. [3]

In our case, since the length of questions answered greatly varies per user, going up to 480, it is important that we use an LSTM rather than an RNN to avoid the vanishing gradient issue.

The update equations for LSTMs are significantly more complicated than for RNNs, and can be found in the Appendix.

5 Experiments/Results/Discussion

In this section, we will :

- Assess whether DKT perform better than BKT on our dataset
- Analyze whether new inputs, such as time to answer, skill, sub-skill etc. improve the performance of DKT

In order to answer these questions, we run 9 models.

We first fit a BKT model. We use the pyBKT library [1], which models student mastery of a skills as they progress through series of learning resources and checks for understanding. Mastery is modelled as a latent variable has two states - "knowing" and "not knowing". At each checkpoint, students may be given a learning resource (i.e. question(s) to check for understanding).

Note that manual hypertuning (such as a grid search) is not needed since the model finds the probability of learning, forgetting, slipping and guessing that maximizes the likelihood of observed student responses to questions.

Thereafter, we run 8 different DKT models.

The first DKT model, which is the simplest one, only contains previous answers as inputs. This is the initial idea that has been tested by Chris Piech et al [3] . The results of this model allow us to answer our first research question.

The next 6 models test the importance of each of the 6 additional variables. In order to run this test, each model will have as inputs [previous answer, additional variable]. The goal is to see how each additional variable improves the AUC compared to the simple DKT model.

Ideally, we would also run each combination of variables because some might be correlated. However, this would represent more than 100 models to run, so instead, we run a metamodel (8th model) with all variables to get the best possible AUC and compare it with the other models.

All models are trained on 80% of the data, and parameters are tuned on a 10% validation set. Finally, models are tested on a 10% testing set.

The input of each model is a numpy array of shape Number of users, Max length of questions in the dataset, number of variables.

We first add a masking layer, since we padded the sequences of questions in the pre-processing sections. This layer is able to know when to skip / ignore certain timesteps in sequence inputs. In our case, the model will ignore the value "-50". Note that all our values were between 0 and 1 since we used a MinMax Scaler, so it was necessary to choose -50 (or any other value not in [0,1]) rather than 0.

Then, we add an LSTM layer with 30 neurons. We tested from 10 to 50 neurons, but 30 was the right compromise between computational time and AUC. Also, the objective was to assess whether a DKT model with a very simple architecture would surpass a hypertuned BKT (using pyBKT). The low number of neurons also allows the model to generalize very well. We could hypertune the model more and try to get the best performance, but this was not the objective of this research, and it is something that Workera could do as a next step.

Therafter, we add a dense layer with 1 neuron, with a sigmoid function since we are working with probabilities. Tanh, for example, would not be a good choice in this case since it returns values between -1 and 1.

Since we are classifying a binary output, we use the Binary Cross-Entropy as a loss function.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right) \tag{7}$$

Furthermore, we use the Adam optimizer since its results are generally better than every other optimization algorithms, have faster computation time, and require fewer parameters for tuning. Adam is often recommended as the default optimizer for most of the applications.

We decided to use mini-batches of 128 units, which is an average size that allows to increase the available computational parallelism (typical of big batches), while still providing improved generalization performance (typical of small batches).

The model runs with 20 epochs, which is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset. We tried different numbers of epochs, but models would not improve much after 20 epochs.

Finally, the metric used is the AUC, which measures the total area underneath the ROC curve. The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The following figures summarize the results on the 9 models.

AUC per model

Set	**BKT**	**Simple DKT**	**DKT with time to answer**	**DKT with number of attempts**	**DKT with domains**	**DKT with skills**	**DKT with subskills**	**DKT with questions**	**Meta DKT**
Training	0.637	0.639	0.641	0.640	0.661	0.674	0.749	0.758	0.765
Testing	0.630	0.631	0.632	0.632	0.648	0.662	0.742	0.751	0.758

Figure 2: AUC scores for all models tested. BKT is the standard BKT with fixed Slip and Guess rates. DKT models are the result of using LSTM Deep Knowledge Tracing with different variables.

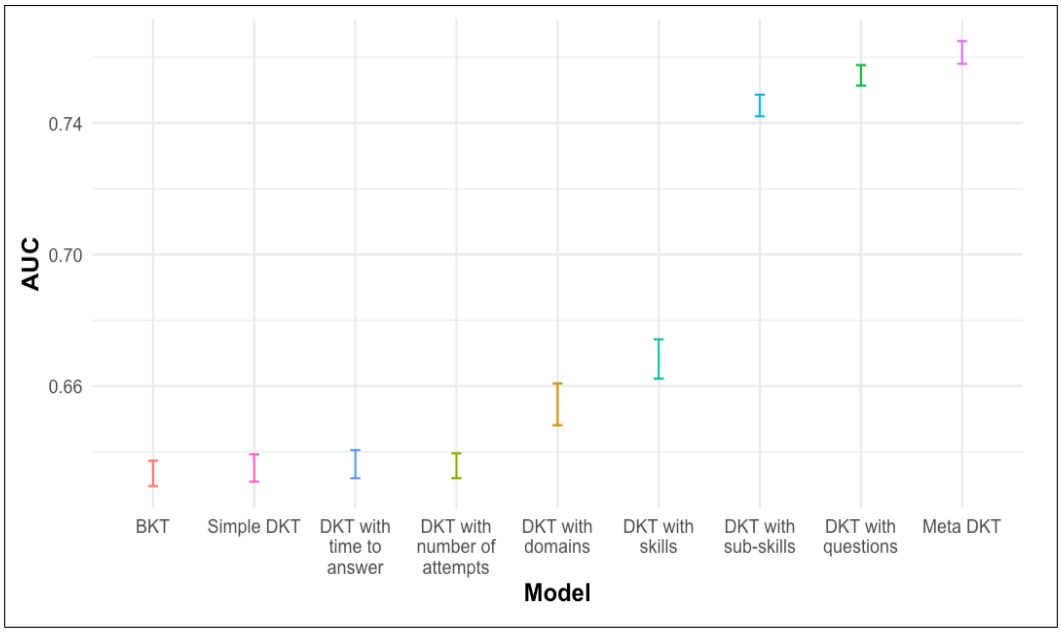

Figure 3: AUC scores for all models tested. Error bars show the testing AUC at the bottom and the training AUC at the top.

All the models that we ran got a similar AUC for the training and testing sets, as outlined by the short length of the error bars. This was expected, as we voluntarily chose a low number of units in the LSTM layer, helping the model to generalize and to run faster.

1) The simple DKT model has a higher AUC than our BKT model, even with a very simple LSTM architecture, which answers our first research question : The DKT model is indeed better, and it generalizes well on a new dataset. Note that we could certainly have improved the performance of the DKT model by adding neurons and layers, but our analysis proves that even a simple architecture leads to better results with DKT.

2) Adding variables improves the model significantly, especially with sub-skills and questions. We can see that the AUC of the Meta DKT is very similar to the AUC of the model with sub-skills and the model with questions. This means that these 2 variables are particularly good predictors of performance. This makes sense, since these are 2 precise variables that allow the model to analyze whether a certain question or sub-skill is particularly hard. In simple terms, the model knows which questions or sub-skills are very hard, so it is capable to predict whether a new user will likely struggle. If 90% of users failed a question, it is very likely that a new user will also fail the question, so this is a very good indicator.

6 Conclusion/Future Work

Effectively modeling student knowledge would have high educational impact, but the task has many inherent challenges. In this paper, we assessed the improvement of Deep Knowledge Tracing on adult learners, using an LSTM, over the widely used Bayesian Knowledge Tracing. We then studied various configurations of DKT to understand which variables have a significant importance on student knowledge modeling. We concluded that a simple DKT model was indeed better than a hypertuned BKT, even on adult learners, and that adding other variables to the model led to notable enhancement of DKT. We would advise platforms to use the DKT model with the questions variable. The results with this variable are significantly better than the BKT model, and the model still runs really fast, especially with a simple architecture. While our goal was not to find the best architecture for this specific dataset but rather to compare different models, we would advise platforms to test various architectures and carefully hypertune parameters of the DKT model with questions, in order to get the best predictions for their users. As a final note, one disadvantage of DKT over BKT is that it

requires large amounts of training data, and so is well suited to an online educational platform, but not a small classroom environment. Further research could incorporate other educational impacts such as dropout prediction, and validate hypotheses posed in education literature such as modeling how students forget. Next steps could also comprise multiclass classification to encompass the "Do not know" category instead of working with a binary outcome.

References

[1] Wang F. Pardos Z.A Badrinath, A. pybkt: An accessible python library of bayesian knowledge tracing models. *In S. Hsiao, and S. Sahebi (Eds.) Proceedings of the 14th International Conference on Educational Data Mining (EDM)*, pages 468–474, 2021.

[2] Joseph E. Beck. Difficulties in inferring student knowledge from observations (and why you should care). 2007.

[3] Jonathan Huang Surya Ganguli Mehran Sahami Leonidas Guibas Jascha Sohl-Dickstein Chris Piech, Jonathan Bassen. Deep knowledge tracing. 2015.

[4] Anderson J.R. Corbett, A.T. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction)*, page 253–278, 1995.

[5] Brian Junker Hao Cen, Kenneth Koedinger. Is over practice necessary? improving learning efficiency with the cognitive tutor. 2007.

[6] Vincent Aleven S.J.d. Baker, Albert T. Corbett. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. 2008.

[7] Neil T. Heffernan Zachary A. Pardos. Modeling individualization in a bayesian networks implementation of knowledge tracing. 2010.

7 Appendix

7.1 Data Dictionary

Variable Name	Description	Example
USER ID	Numeric variable uniquely identifying a user	136896
DOMAIN	Character variable representing the domain of expertise	Software Engineering
SKILL	Character variable describing the skill corresponding to a question	Coding methods
SUB-SKILL	Character variable describing the sub-skill within a certain skill	Implementing a code solution to solve a problem
QUESTION ID	Numeric variable uniquely identifying a question	908
DATE	Date variable representing the time at which a question has been answered	2021-08-01 19:46:52
TIME TO ANSWER	Numeric variable representing the time that the user took to answer a question (in seconds)	186
ATTEMPT	Numeric variable describing the times that a specific user was shown this particular question	4
CORRECT/INCORRECT/DO NOT KNOW	Categorical variable describing the outcome : 1 if correct, 0 if incorrect, null if the user does not know the answer	0
LEARNING TIME	Numeric variable representing the learning time estimated by Workera, in hours	1
SCORE	Numeric variable representing the score reached by the user for a certain domain at the end of the assessment	188
TARGET SCORE	Numeric variable representing the target score aimed by the user based on a targeted job position	191

7.2 LSTM

The following equations are the update equations for an LSTM model :

$$\mathrm{i}_t = \sigma\left(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{b}_i\right)$$

$$\mathrm{g}_t = \sigma\left(\mathbf{W}_{gx}\mathbf{x}_t + \mathbf{W}_{gh}\mathbf{h}_{t-1} + \mathbf{b}_g\right)$$

$$\mathrm{f}_t = \sigma\left(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{b}_f\right)$$

$$\mathrm{o}_t = \sigma\left(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{b}_o\right)$$

$$\mathrm{h}_t = \mathbf{o}_t \odot \mathbf{m}_t$$

$$\mathrm{m}_t = \mathbf{f}_t \odot \mathbf{m}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

$$\mathrm{z}_t = \mathbf{W}_{zm}\mathbf{m}_t + \mathbf{b}_z$$

$$\mathrm{y}_t = \sigma\left(\mathbf{z}_t\right)$$