Modeling learning behavior of Learnnavi users

Marie Biolková and Ghali Chraibi
Department of Computer Science, EPFL, Switzerland

***Abstract*—Knowledge tracing techniques are widely used to study learning behavior and are important for improving educational technologies (EdTechs), which are becoming increasingly popular. We conduct an extensive study and compare various knowledge tracing models using data collected from an EdTech start-up, Lernnavi. Focusing on their Mathematics track only, we found that GRU with 16 recurrent units can best model users' mastery. It achieves an AUC of 0.677 and RMSE of 0.476 on the task of predicting whether the student will answer correctly or not. We also analyzed the learning curves and discussed possible improvements.**

I. Introduction

Students are increasingly turning to online learning as it is a convenient and less expensive alternative to traditional learning, which allow the students to progress at their own pace. However, when the COVID-19 pandemic emerged, online education became mandatory worldwide and many academic institutions found themselves unprepared. In order to provide good quality personalized online education in an automated fashion, we need to design algorithms that can mimic the learning process of students to provide them with a suitable learning path.

An effort to achieve this has been developed under the name of *knowledge tracing*. It aims at modelling the student's knowledge state and predict their future success in a particular skill. Although there has been tremendous progress in this area, modeling students' mastery remains a challenge.

In this report, we conduct a survey of knowledge tracing techniques. We apply them to real-world data collected by an EdTech, Lernnavi [1], specializing in online tutoring, and discuss the implications of our findings.

A. Research Questions

We try to address the question of how much are Learnnavi's users learning. In particular, we are interested in finding out which topics the users find challenging in order to provide them with relevant practice examples in the future, as well as evaluating to what extent we can model their learning progress. We focus on the study of Mathematics, but the approach can be extended to other domains.

II. Data Description

The data used in this project was provided by Learnnavi [1], an online platform where users can both study and assess their level in different topics of Mathematics and German. It allows users to follow their progress and obtain feedback on their submissions.

The data consists of several tables representing the different entities of the platform. Each question on the platform is classified by topics which have a hierarchical tree structure. In our work, we analyse all data that are sub-topics of 5 main topics appearing on Lerrnavi. These include Functions, Equations, Numbers and number quantities, Elementary geometry and Terms.

In total, the data we received consists of 819'862 trials from 13'790 users on questions divided into 557 distinct topics. Focusing on the Mathematics-related questions and after cleaning the data, our dataset finally consists of **121'281 trials from 5'192 users** on questions divided into **272 topics**.

III. Exploratory Analysis

After applying all filters, our data consists of 24.4% wrong, 21.4% partially correct and 54.2% correct answers. The distribution of the number of questions answered is heavily right-skewed, with most students answering only a few questions. On average, the students engaged in 23 tasks, but the standard deviation is very high, around 33. This is because there are some motivated students that remain on the platform for a long time, completing hundreds of tasks (the maximum was 941). The skew of the distribution implies that the median is much lower than the mean, around 14 tasks.

As mentioned, the Mathematics questions are divided in five parent topics. We first observe in Figure 1 that there is a large imbalance in the distribution of data per parent topic. Also, most of the trials are correct, but partially correct answers are also well represented in nearly every parent topic. This implies that these trials may have a considerable impact depending on how we treat them. Figure 2 further indicates us that the trials are similarly distributed in terms of difficulty across the topics, but that most of the questions have a low difficulty (1 or 2). The latter can be problematic as users may have a limited progression by not answering challenging questions. Similarly, our models may poorly predict users' mastery when confronted with a difficult question, assuming that the estimated difficulty of questions are realistic.

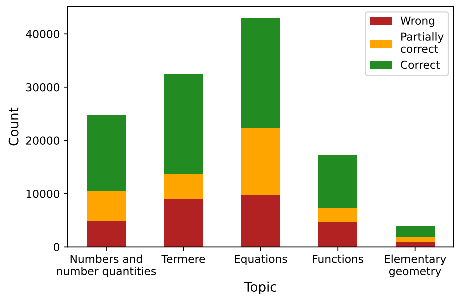


Figure 1. Distribution of answers across topics and evaluations.

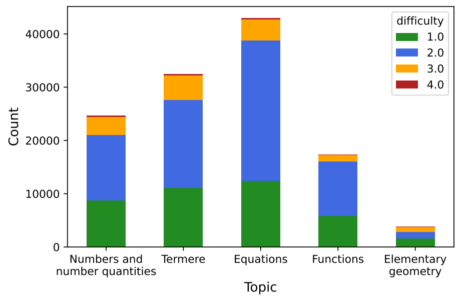


Figure 2. Distribution of answers across topics and difficulties.

Finally, we note from Figure 3 that the distribution of the evaluation per difficulty level does not seem natural. Indeed, one would expect the easier questions to have a high proportion of correct answers and the more difficult questions to have a low proportion of correct answers. Our hypothesis to explain this is that Learnnavi adapts the difficulty level of the questions according to the users' history.

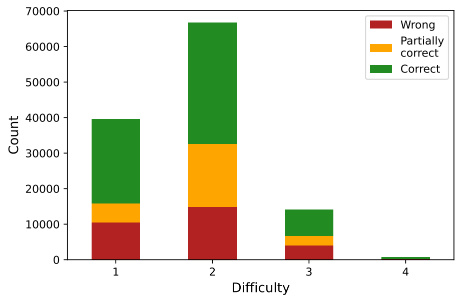


Figure 3. Distribution of answers across difficulties and evaluations.

IV. Proposed Method

A. Data pre-processing

We first decided to consider only the Mathematics part of Learnnavi data. We assume that Mathematics data and German data may behave very differently.

Then, we built a graph of mathematical topics. We used it to associate each question its parent topic in addition to its (leaf) topic. This allows us to be able to consider either the leaf nodes as the skills or the parent nodes as the skills where we assume that the subtopics from a same parent node are training the same skills. The first option provides more fine-grained skills (272), but then less data are available for each skill. On the other hand, the second option considers only a small amount of skills (5) which may be too simplistic, but then each skill has enough representative per skill. Hence for BKT, we considered the second option as in this framework each skill require to train a specific model. However, for the logistic models (AFM and PFA) and DKT, we considered the first option as in these frameworks only one model is built for all the skills and having more skills allow to have more parameters and hence more flexibility for the models to learn.

The evaluation provided by Learnnavi are categorical: an answer can be either wrong, partially correct or correct. However, to fit knowledge tracing model we need mainly require either logical or numerical values. We therefore considered two settings:

- the **binary setting** with a binary evaluation $\{0, 1\}$, considering partially correct answers as wrong. This setting is the norm for many knowledge tracing models such as BKT, AFM and PFA, but does not use the full information.
- the **ternary setting** with a ternary evaluation $\{0, 0.5, 1\}$, taking into account partially correct answers. This setting makes full use of the data available, but requires to adapt the models.

B. Model architectures

1) Bayesian Knowledge Tracing: Bayesian Knowledge Tracing (BKT) [2] is a popular approach for modeling learning behaviour. In its most used form, it employs a Hidden Markov Model (HMM) where the unobserved (latent) states are the student's mastery of a topic [3]. These states are by assumption related to the observed answers to test questions. BKT assumes a division of skills into distinct groups which are unrelated, and fits one model per skill. If the data was collected from multiple students, BKT does not distinguish between them. Furthermore, student cannot forget a skill once it is learned, and the model only supports binary outcomes – either the student has mastered the skill or he/she has not. The mastery can be achieved through practice.

The model is parametrized with $\theta = \{p_0, p_L, p_F, p_S, p_G\}$, where:

- p_0 is the initial probability of mastering the skill
- p_L and p_F are the the (*transition*) probabilities for learning and forgetting, respectively
- p_S and p_G are the (*emission*) probabilities for slipping (answering wrong after mastering a skill) and guessing

(answering correctly without mastering a skill).

The BKT training algorithm solves two tasks at once – parameter learning and inference. To find the parameter estimates, it maximizes the likelihood of the observed data $\mathbf{o} = [o_0, \ldots, o_T]$ of N students up until a time T,

$$\max_{\theta} \prod_{i=1}^{N} \sum_{\boldsymbol{s}_i} p\left(\mathbf{o}_i, \boldsymbol{s}_i\right),$$

where s_i is the latent mastery of a student at time step i. This is done by iteratively optimizing the negative log-likelihood with the Expectation-Maximization (EM) algorithm [4]. Inference then consists of predicting the next observation o_{T+1} given the learned parameters and the past observations.

To satisfy the condition of skills independence, we fit one model per *parent* topic, which ensures that the overlap between skills is minimized. We consider BKT as our baseline.

2) Additive factors model: The Additive factors model (AFM) is a generalized linear mixed effects model, proposed in an effort to capture the transfer of learning across skills [5]. AFM assumes a random effect on the student's prior knowledge (θ_n) and the learning rate γ, allowing each skill k to be learned at a different pace (though this rate is the same for all students). Moreover, it incorporates skill difficulty β_k. Note that each task can use multiple skills. By assumption, each new opportunity improves the learning. Formally, it can be written in the logistic form, for student n and task i:

$$p_{n,i} = \frac{1}{1+e^{-\pi_{n,i}}} \text{ with } \pi_{n,i} = \theta_n + \sum_k q_{i,k} \cdot (\beta_k + \gamma_k \cdot T_{n,k}).$$

Above, $q_{i,k}$ is an indicator of skill k being used in item i and $T_{n,k}$ represents the number of opportunities the student had at skill k.

In summary, the main features of this model are the fact that students may have different prior knowledge of a skill, and that unlike BKT it takes into account multiple skills at once, learning how students apply acquired knowledge in different contexts.

3) Performance Factors Analysis: Performance Factors Analysis (PFA) [6] is a variant of AFM, only it assumes that the learning rate is different for correct and wrong answers. In mathematical terms, we would alter AFM as

$$\pi_{n,i} = \theta_n + \sum_k q_{i,k} \cdot (\beta_k + \gamma_k \cdot s_{n,k} + \rho_k \cdot f_{n,k}),$$

with s_{n_k} being the number of prior successes of student n at skill k and $f_{n,k}$ the failures. Notice that these quantities are now assigned distinct learning rates γ_k and ρ_k.

Our AFM and PFA models assume that a question only belongs to the topic it is directly assigned. One could try to consider all parent nodes as well.

4) Deep Knowledge Tracing (DKT): To improve on the limitations of the aforementioned models, Piech et al. (2015) proposed to use a recurrent neural network (RNN). This approach was adopted as Deep Knowledge Tracing, where the depth is referring to having many time steps to unfold [7]. The idea behind using a flexible neural architecture is that learning depends on many factors, including the student, the material, the time spent learning. These are difficult to capture in terms of well-defined features: instead, using a lot of data, one could let a model learn these quantities implicitly. To that end, RNNs are a suitable choice as they were designed to deal with sequential data, where the input may differ in length. They perform a mapping from the input $\mathbf{x}$ to the output $\mathbf{y}$ using hidden states $\mathbf{h}$, which encode important information from to past to use when making predictions. At each timestep t, we then have

$$\boldsymbol{y_t} = \sigma\left(W_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y\right) \tag{1}$$

$$\boldsymbol{h_t} = \tanh\left(W_{hx}\boldsymbol{x}_t + W_{hh}\boldsymbol{h_{t-1}} + \boldsymbol{b_h}\right). \tag{2}$$

The matrices W_{hx}, W_{hh} W_{yh}, the initial state $\mathbf{h}_0$ and the bias vectors $\mathbf{b}_h$, $\mathbf{b}_y$ are learnable parameters.

More complex variants of RNN is a so-called Long-Short Term Memory network (LSTM), and Gated Recurrent Units (GRU). These incorporate the concept of *gating*, which helps tackle one of the problems encountered by a vanilla RNNs, that typically struggle with long sequences. While GRU consists of 2 gates and maintains only the hidden state $\mathbf{h}$, LSTM has 3 gates and in addition keeps track of a *cell* state, making it more complex both in terms of number of parameters and memory footprint. Although LSTM tends to outperform GRU and vanilla RNN, this increased complexity can be a burden on some tasks. In our experiments, we compare the performance of the 3 architectures – RNN, GRU and LSTM – to see what is the most suitable choice for the Lernnavi knowledge tracing task.

We implement DKT in Keras [8]. The input sequence is encoded with a one-hot vector, so that the input data are a tensor of the shape $N \times 2 *$ #skills $\times T$, where T is the maximum sequence length. The input from students with fewer responses is padded accordingly, and the padded values are masked in the model. The model is optimized using the binary cross entropy for 20 epochs using Adam [9] and a batch size of 32 samples.

For the case where we consider partially correct answers, we include another input category, so that the input has the shape $N \times 3 *$ #skills $\times T$, and use the Mean Squared Error as a loss function. A major difference is that instead of predicting the probability of the next answer being correct, we predict a *score* which is subsequently thresholded. This is done by replacing the sigmoid by a linear activation in the output layer. We predict a partially correct answer whenever the score is in $\left(\frac{1}{3}, \frac{2}{3}\right)$. This allows us to maintain the ordering of the output classes. An alternative, which we did not consider, is to use 3 output units (wrong, partially correct, correct), use softmax in the last layer and optimize the multiclass cross entropy.

C. Evaluation Protocol

In order to evaluate our models, we construct a 80:20 train/test split, so that no user occurs in both training and test set. Then we fit the model on training data and evaluate on the test data. We report the average root-mean-squared-error (RMSE) and area under the curve (AUC). For BKT, we do this for each topic separately. For PFA and AFM, since they can incorporate data on other topics, we fit them on the combined data, then report the metrics per topic, as well as overall (as a weighted average). This may cause some small variation in the evaluation as it is not straightforward to perform stratified grouped splitting, i.e. such that we ensure users are not in both train and test set and at the same time guarantee a 80:20 split on each topic. As for DKT, since we need to do hyperparameter tuning, we further split the training set to obtain a validation set.

In the ternary setting, which includes the partially correct answers, we report the multi-class AUC, using a method that aggregates the pairwise AUC scores (one-vs-one).

To tune the hyperparameters of DKT, we employ Optuna [10], an automatic optimisation framework. Specifically, we use it to perform Bayesian optimisation which is designed to explore the search space efficiently, by proposing promising configurations to try. It is however expensive to run many trials, so we restrict the search space to recurrent units in $\{4, 8, 16, 32\}$ and dropout between 0.001 and 0.5. Th sampling of dropout is done on a logarithmic scale, as we do not expect strong regularization will be needed with the small amount of data that we have at hand. We run 25 trials with pruning (i.e. an unpromising trial may be stopped early) for each model, with and without the difficulty feature. Moreover, the optimisation stops if no improvement is found in the 5 last trials.

V. Experimental Evaluation And Discussion

In this part, we present and discuss the results of our experiments.

A. Binary setting

First of all, let us discuss the scenario where we assume that all answers are either right or wrong (assuming partially correct is still wrong). The results of are shown in Table I. The the best overall performance was obtained with DKT. The margin by which it outperforms the other methods is however small, especially considering the additional computational overhead incurred. We found that a GRU with 16 recurrent units and dropout of 0.22 worked best, and adding the estimated difficulty as a feature did not improve the result.

Interestingly, BKT is able to achieve the same RMSE as DKT, and is also better than AFM and PFA on most topics (see Table III). The only exception is Elementary geometry, where PFA was the best model, and even AFM did better than BKT on this topic. Perhaps the assumption that each student may have different amount of prior knowledge was particularly important for this topic, which made these models more suitable. On the other hand, the AUC of PFA is particularly disappointing in the case of Number quantities, as it is not too far from simply guessing random outcome (in that case, AUC would be around 0.5).

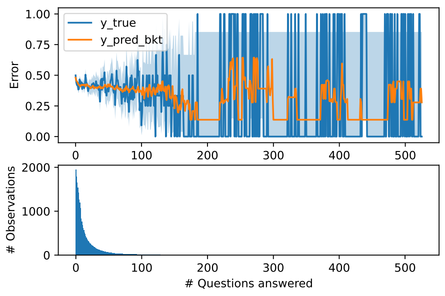
For PFA adding the difficulty helped slightly to reduce the RMSE and increase AUC.

	BKT	AFM	PFA	PFA+	DKT
AUC	**0.476**	0.483	0.480	0.478	**0.476**
RMSE	0.661	0.636	0.651	0.659	**0.677**

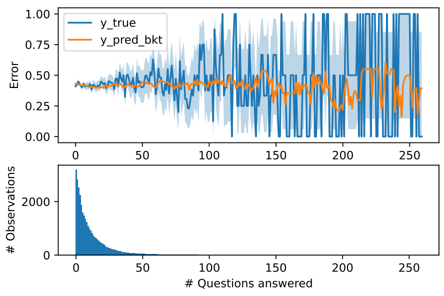
Table I
ROOT-MEAN-SQUARED ERROR (RMSE) AND AREA UNDER THE ROC CURVE (AUC) FOR EACH MODELS. WITH + WE DENOTE MODELS WHICH INCLUDE THE ESTIMATED DIFFICULTY OF THE QUESTION. FOR BKT, THE WEIGHTED AVERAGE IS REPORTED.

The learning curve for each topic are presented in Figure 4. We used BKT to plot these, as it achieves the lowest RMSE (together with DKT). From these, we make the following observations:

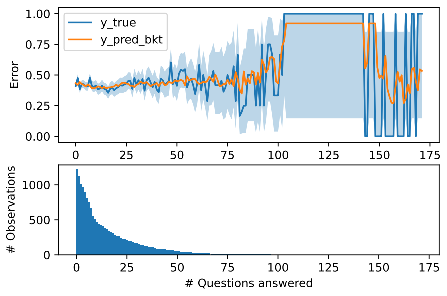
- **Functions.** Looking at the ground truth data (in blue), we see that the initial error rate (for early stages, where the data are most reliable) is just below 0.5. The error rate then decreases slightly as the students practice more questions, but starts increasing again after around 20 questions. This may be because more difficult questions are given to the user from Lernnavi to challenge him/her. The data become too noisy after around 50 questions, as there are not many students left who answered this many questions. The confidence interval becomes too wide after that point and therefore there may be large discrepancies between students, and the BKT model (orange), although offering a good fit in the early stages (especially until question 25), is no longer very accurate. Note that the number of students still involved with the topic goes down after 12 practice questions.
- **Number quantities.** Starting at around 0.5 error rate, we notice a small improvement in the Number quantities knowledge after the first few questions. However the error rate is not very stable already from the start, indicating there are large discrepancies among users, which are further emphasized once the sample size is reduced (after 10-20 questions).
- **Equations.** This topic seems to be on the more challenging side at first, with error rate around 0.6. The users then improve fast, as the ground truth error rate



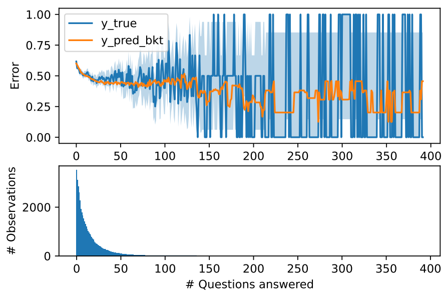
(a) Numbers and number quantities



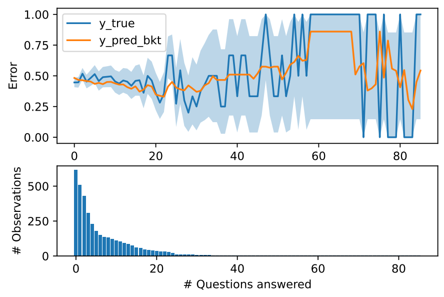
(b) Termere



(c) Functions



(c) Equations



(d) Elementary geometry

Figure 4. Learning curves for different topics as given by the ground truth (blue) and BKT (orange), together with the corresponding histogram of responses. Notice that the curves become very noisy as a large number of students stop practicing.

drops below 0.5 within 20 practice questions. The error rate is stable until about 50 questions, then gets wider and therefore it becomes hard to draw conclusions from the later stages – the BKT fit however suggests that the users do improve their skills over time, despite the uptake being reduced significantly after 10-20 questions.

- **Elementary geometry.** Users studying geometry seem to plateau around a 0.5 error rate in the early stages, then start improving slightly once they have answered at least 10 questions. The confidence interval is wider than for the other topics as fewer students engage in geometry overall. Again, we observe from the histogram that the dropout rate is high after 4-5 questions, so users give up very fast.
- **Termere.** The learning curve for Terms does not exhibit much trends, which means the performance remains unchanged as users are answering more questions. If the difficulty is not progressively increasing, this could indicate that there is little learning going on, otherwise the difficulty is increased in an adequate way. There is no obvious point at which most users leave, we observe an exponential dropout rate.

Overall BKT provides a good fit, however tends to underestimate the error rate. In all of the plots, the fast oscillations in the later stages are due to the reduced sample size as many uses stop practicing. In fact, this was not exclusive to BKT – we noticed that all our models are usually overly optimistic and predict a wrong answer less often than they should.

	PFA+	DKT
AUC	0.531	0.545
RMSE	0.407	0.409

Table II
ROOT-MEAN-SQUARED ERROR (RMSE) AND AREA UNDER THE ROC CURVE (AUC) FOR THE MODELS CONSIDERED IN THE TERNARY SETTING. WITH + WE DENOTE MODELS WHICH INCLUDE THE ESTIMATED DIFFICULTY OF THE QUESTION.

B. *Ternary setting*

In this part, we considered an additional category – the partially correct answers (Table II). The best model turned out to be GRU once again, with the same hyperparameters. The RMSE is by design reduced when partial credit is introduced, but the error did not drop drastically as we would hope when the model can make a prediction of 0.5. Furthermore, the multi-class AUC is poor for both models. An inspection of the confusion matrix revealed that the model almost never predicted wrong answers, and tends to concentrate most predictions around 0.5, i.e. partially correct answers. It is therefore clear that the model simply learned to bet safe, as it is punished for large deviations from the

		Functions	Number quantities	Equations	Elementary geometry	Terms
BKT	AUC	**0.704**	**0.645**	**0.683**	0.599	**0.628**
	RMSE	**0.459**	**0.479**	**0.474**	0.493	**0.483**
AFM	AUC	0.636	0.582	0.661	0.638	0.555
	RMSE	0.481	0.490	0.479	0.480	0.501
PFA	AUC	0.658	0.602	0.667	**0.661**	0.564
	RMSE	0.477	0.486	0.477	**0.474**	0.500

Table III
PER-TOPIC PERFORMANCE OF SOME OF THE MODELS ON THE BINARY TASK.

correct label, which are lower when it predicts partially correct.

We also note that AFM and PFA, which are logistic models, are not aimed for ternary inputs and are therefore not suitable for the task, which our experiments confirmed.

Another reason why the models suffered poor performance is that many students drop out after only a few trials. This can be seen from the histograms of engagement in Figure 4. Without enough data on the student, it is difficult to model the student's behavior, and the model has to guess. To improve the quality of our models, we could try to focus on cohorts where the users are engaged for a substantial enough of a period.

The notion of partial credit, while providing some information, is not easy to grasp and therefore leverage. It would perhaps be beneficial to assign a *score* as evaluation, indicating to what degree the answer was correct. This would definitely address the confusion regarding the partially correct answers.

VI. Conclusion and Future Work

In conclusion, we achieved to adapt several knowledge tracing models on Learnnavi data, and obtained an AUC score of 0.677 with a GRU. Our efforts to improve the models by including additional information (such as the estimated difficulty and the partial credit) were not particularly successful, and call for a more careful handling.

In this study, we observed that the notion of partial credit implemented in Learnnavi is difficult to use and interpret. Indeed, in an MCQ the latter considers equally correct the answers of someone who would tick all the choices (answering randomly) and the answers of someone who understands the topic but misses one tick each time. Based on this observation, our model may be trained on a very noisy data in the ternary setting. Hence, the binary setting seems more realistic and provides better results.

Another limitation with the data is that the time series of the users are of unequal size and mostly short which is not ideal to train some of the models. This is due to the fact that the number of answers per user follows a power law. Many users try Learnnavi platform once or twice and only few users work rigorously on the platform. Improving retention may therefore be beneficial.

The next step would be to focus more on the preprocessing of the data. This could allow the models to learn less noisy information and provide more accurate results. A first approach would be to filter out users who have made little use of the application and to consider using the classroom features from Learnnavi which ensures a more controlled environment where the users are engaged for a longer period, even if it concerns only a little amount of data.

Also, having access to more data would surely help to have more accurate models especially for Deep Knowledge Tracing models.

References

[1] "Lernnavi," http://https://www.lernnavi.ch/, accessed: 2022-05-20.

[2] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User modeling and user-adapted interaction*, vol. 4, no. 4, pp. 253–278, 1994.

[3] J. E. Beck and K.-m. Chang, "Identifiability: A fundamental problem of student modeling," in *International Conference on User Modeling*. Springer, 2007, pp. 137–146.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[5] H. Cen, K. Koedinger, and B. Junker, "Comparing two irt models for conjunctive skills," in *International Conference on Intelligent Tutoring Systems*. Springer, 2008, pp. 796–798.

[6] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger, "Performance factors analysis–a new alternative to knowledge tracing." *Online Submission*, 2009.

[7] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," *Advances in neural information processing systems*, vol. 28, 2015.

[8] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[9] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[10] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, ser. KDD ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2623–2631. [Online]. Available: https://doi.org/10.1145/3292500.3330701