# Few hundred parameters outperform few hundred thousand?

Amar Lalwani
funtoot
2nd floor, Sancia House,
14th Cross,1st Stage Domlur,
Bengaluru 560071, India
amar.lalwani@funtoot.com

Sweety Agrawal
funtoot
2nd floor, Sancia House,
14th Cross, 1st Stage Domlur,
Bengaluru 560071, India
sweety.agrawal@funtoot.com

## ABSTRACT
Knowledge Tracing plays a key role to personalize learning in an Intelligent Tutoring System including funtoot. Bayesian Knowledge Tracing, apart from other models, is the simplest well-studied model which is known to work well. Recently, Deep Knowledge Tracing based on Deep Neural Networks, was proposed with huge promises. But, soon after, it was discovered that the gains achieved by DKT were not of significant magnitude as compared to Performance Factor Analysis [13] and BKT and its variants proposed in [6]. In the quest of examining and studying these models, we experiment with them on our dataset. We also introduce a logical extension of DKT, Multi-Skill DKT, to incorporate items requiring knowledge of multiple skills. We show that PFA clearly outperforms all the above mentioned models when the AUC results were averaged on skills while PFA and DKT, both were equally good, when they were averaged on all data points.

## Keywords
Deep Knowledge Tracing, Adaptive Learning, funtoot, Bayesian Knowledge Tracing, Intelligent Tutoring System, Performance Factor Analysis

## 1. INTRODUCTION
An Intelligent Tutoring System's main aspect is to deliver the instruction and provide feedback as and when required. To do that, the system requires to measure the knowledge state of a student with respect to the content available. The system continuously monitors the student's performance, updates the knowledge state and based on that takes further decisions. The techniques capable of performing these functions are called Knowledge Tracing models.

Bayesian Knowledge Tracing [2] has been one of the most predominantly researched models in the educational data mining domain. BKT is a 2-state skill specific model, where the student's knowledge state can take either of the two values: learned or unlearned. Moreover, a skill once learned cannot be unlearned. These assumptions make it a very simple and constrained model and has led lots of researchers to extend the model by enhancing it with new features to improve its performance; making it less constrained so to say. For instance [10] extend BKT in the scenario where the students do not necessarily use the system in the same day.

Authors of [14] proposed an individualized BKT model which fits not only the skill specific parameters, but also student specific parameters and have reported significant gains over standard BKT.

Educational data mining techniques can now very accurately predict how much a student has learned a Knowledge Component (KC). But it doesn't give information about the exact moment when the KC was learnt. [3] discusses a technique about finding a moment of learning.

Another model Performance Factor Analysis (PFA) is a logistic regression model proposed in [7] which showed better performance than standard BKT. Unlike BKT, PFA can incorporate items with multiple skills. PFA makes predictions based on the item difficulty and historical performances of a student. [4] has compared BKT and PFA by using various model fitting parameter models like Expectation Maximization (EM) and Brute Force (BF). Knowledge tracing models with EM have shown performance comparable to PFA[4].

The most recently published model - DKT [9] is the newest technique in this area of research. DKT is an LSTM [5] network, a variant of recurrent neural network [11] which takes as input a series of exercises attempted by the student and correspondingly a binary digit suggesting if the exercise was answered correctly or not. DKT has shown significant gains over BKT which is a very tempting gain for any researcher in this community to look into and study further. Papers like [6], [13] and [12] did just that.

Authors in [13] have pointed out few irregularities in the dataset used by authors in [9] which, when accounted for, reduce the gain reported by using DKT. They also reported that DKT doesn't quite hold an edge when the results are compared with PFA.

Another standard framework for modelling student responses, Temporal extension of Item Response Theory (IRT) is compared with DKT in [12]. Authors have reported that the variants of IRT consistently matched or outperformed DKT.

Recent paper [6] studies DKT even further and explains why DKT might be better. It has been pointed out that DKT inherently exploits the characteristics of the data which standard models like BKT cannot. So, in order to make a fair comparison between the two, authors have presented three different variants of BKT with forgetting, skill discovery and latent abilities which might help BKT make use of information from the data the way DKT does.

Having introduced these variants, the authors also make a point that Knowledge Tracing might not require the "depth" that deep learning models offer.

Being an Intelligent Tutoring System, funtoot's tutor module requires sophisticated knowledge tracing technique which models the process of knowledge acquisition and helps students achieve mastery. One such model operates at the level of LGs (discussed in section 2) which models the committance and avoidance of them with time and practice. In the context of this paper, these LG models are of prime importance to us and henceforth we will refer LGs as skills. Also, considering user experience, we need a model which can be used for predictions in real time without compromising on user latency.

In this paper, we test standard BKT, the variants of BKT, DKT and PFA on the funtoot dataset and examine the results. We also introduce a logical and trivial extension of DKT to accommodate the items which involve multiple skills. Out of all the models considered in this article, PFA is one such model which allows items with multiple skills. But in our dataset, each of the skills in the item has its own response and hence it is modelled separately in PFA.

The rest of the paper is organized as follows: section 2 gives a brief introduction to our product funtoot and its knowledge graph. Section 3 discusses the experiments on funtoot dataset and results. Section 4 discusses the future work and conclusion.

## 2. FUNTOOT

Funtoot[1] is a personalized digital tutor which is currently being used actively in around 125 schools all over India with the total of 99,842 students registered. The curriculum of math and science for grades 2 to 9 is covered by funtoot.

Schools in India are typically affiliated with one of the boards of education[2]. Curriculum for math and science from the following boards of education are included in funtoot:

- CBSE[3] board for grades 2 to 9,
- Karnataka State Board[4] for grades 2 to 8,
- ICSE[5] board for grades 2 to 8 and
- IGCSE[6] board for grades 2 to 3.

[1]`http://www.funtoot.com/`
[2]`https://en.wikipedia.org/wiki/Boards_of_Education_in_India`
[3]`https://en.wikipedia.org/wiki/Central_Board_of_Secondary_Education`
[4]`https://en.wikipedia.org/wiki/Karnataka_Secondary_Education_Examination_Board`
[5]`https://en.wikipedia.org/wiki/Indian_Certificate_of_Secondary_Education`
[6]`https://en.wikipedia.org/wiki/International_General_Certificate_of_Secondary_Education`

### 2.1 Funtoot Knowledge Graph

Pedagogy team at funtoot has created a funtoot ontology around the subjects Math and Science. This ontology represents the various learning units of any subject and their relationships, which is created based on human expertise in the subject matter. All the above mentioned curricula are later derived from this funtoot ontology based on the age group and grade.

An ontology for a subject is created as follows:

1. a subject is broken down into the smallest teachable sub-sub-concepts

2. it is then mapped to determine inter-dependencies/connections between concepts, sub-concepts (*sc*) and sub-sub-concepts (*ssc*) as shown in the figure 1,
   Consider the example shown in figure 1. Subject *Math* contains a concept *Triangle*, and *Triangle* contains a sub-concept *Congruency.* Sub-concept contains two sub-sub-concepts: *Rules of Congruency* and *Applications of Congruency.* Sub-sub-concepts are connected by "depends-on" relationship. Here, *Applications of Congruency* is dependent on *Rules of Congruency*, which suggests that the latter is a prerequisite for the former.

3. learning gaps (definition 1) are determined in the sub-sub-concepts

   Definition 1. *Learning Gap (LG): "A learning gap is a relative performance of a student in a specific skill, i.e. difference of what a student was supposed to learn, and what he actually learned in a skill. [7]"*

   *"A misunderstanding of a concept or a lack of knowledge about a concept that is required for a student to solve or answer a particular question is also a learning gap"*

   For instance, a question "Solve 12 + 18" is given to student *Alice*. If *Alice* makes a mistake while adding carry and answers 20, we say that a LG (*carry-over error*) has been *committed.* Had she answered 30, this LG would have been said to be *avoided.* This question might also have other LGs which could have been committed simultaneously with the LG mentioned above. If the response is correct, all the LGs of a question are said to have been avoided.

   In figure 1, *Applications of Congruency* is an ssc containing $LG_1$, $LG_2$ and $LG_3$. Learning gaps can have "induce" relationships. In our example, $LG_1$ induces $LG_2$.

4. inter-dependencies get refined based on the data-points received by funtoot through the user's interaction

5. an SSC is further divided into six Bloom's Taxonomy Learning Objectives (*btlos*) using Bloom's Taxonomy [1]. Each learning objective has five difficulty

[7]`http://edglossary.org/learning-gap/`

| BTLO / Difficulty Level | Remember | Understand | Apply | Analyze | Evaluate | Create |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |

Table 1: Btlos, Difficulty levels ⇒ Complexities

levels as shown in table 1. Each cell (for instance, Remember1, Apply2 and so on) in table 1 is called a *complexity* in funtoot.

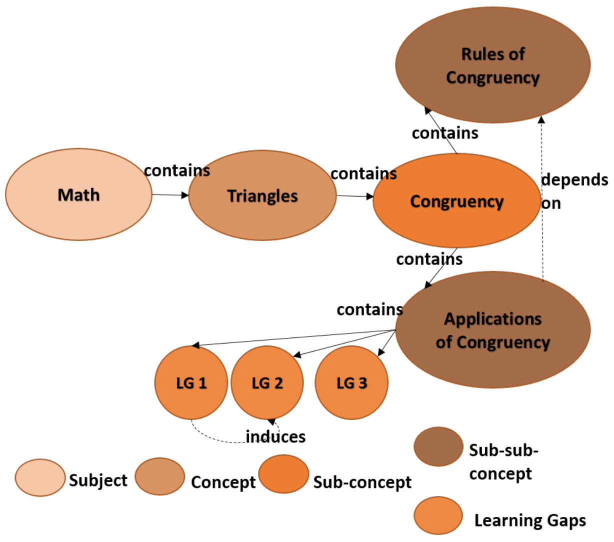

Figure 1: Funtoot Knowledge Graph

## 2.2 Dataset

During a student's interaction with funtoot, information like: session, the scope of the question (which includes $grade - subject - topic - subtopic - subsubtopic - complexity - question$), question identifier, start time, total attempts allowed based on the student's performance, time taken, attempts taken, information about hints, LGs committed in each attempt, assistance provided and so on is logged.

In the study presented in this paper, we model LG as a skill. We aim to predict a student's proficiency in a particular LG. When a student is presented with an item, several attempts are provided to solve it. In an unsuccessful attempt a student might commit more than one LG as explained in section 2 and the same LG can also be committed in several attempts. We know apriori the set of LGs that are exposed by a question. With this information at hand, we need an impression of each of these LGs for the student in the context of this item.

Consider a hypothetical example. *Alice* attempts an item $q$ from a subtopic *Rules of Congruency* having skills $s_1, s_2, s_3$. The series of attempts is shown in table 2.

| Attempt no. | $s_1$ | $s_2$ | $s_3$ |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 |
| 4 | 1 | 1 | 1 |
| Overall Outcome | 0 | 0 | 1 |

Table 2: Attempts made by *Alice* while solving $q$

In the above table, 1 represents avoidance and 0 represents committance. As shown in the table, *Alice* committed $s_1$ in attempts 1, 2 and 3. *Alice* committed $s_2$ in attempt 3. *Alice* avoided $s_3$ in all attempts. The overall outcome of *Alice* in LGs $s_1$, $s_2$ and $s_3$ is $(0, 0, 1)$ which is a logical AND over all attempts. This means that $s_1$ and $s_2$ are committed and $s_3$ is avoided. From now on, we will refer these outcomes as committances and avoidances and they will be used for modelling. So this problem attempt of *Alice* gives rise to three data points.

For this experiment we have used data of $6^{th}$ grade CBSE math from date $2015-07-25$ to $2017-01-30$. Syllabus descendant hierarchy for this dataset is as follows: 22 topics, 69 subtopics, 119 sub-sub-topics, 541 complexities and $1,524$ problems. This dataset has $26,06,022$ entries of problem attempts involving 442 skills. This data is about 176 schools with $11,820$ students and $1,524$ problems. From this dataset, the data of students having less than 100 problem attempts were excluded. This gives us $24,47,027$ problem attempts involving 442 skills with 7780 students and $1,523$ problems. Finally, we have $56,04,227$ data points where $42,68,503$ are avoidances (class 1) and $13,35,724$ are committance (class 0).

In the context of the example shown in table 2, the length of $Alice's$ attempt to solve a question $q$ can be said as three, as there are three skills involved. Given this definition, of length of the problem attempt, figure 2 shows the distribution of the length of the problem attempts in the dataset. 38.18% of the total problem attempts have 1 skill, i.e., length is 1 and 29.47% of the problem attempts have length 2.

## 3. EXPERIMENTS

In this section, we discuss the experiments done on our dataset and report the results. Consider a hypothetical dataset of student *Alice* attempting questions $q_1$ and $q_2$ in the same order. Question $q_1$ has three skills $A, B$ and $C$, question $q_2$ has two skills $B$ and $C$. *Alice* gets only one attempt for both the questions wherein she commits skill $B$ and $C$ and skill $B$ in questions $q_1$ and $q_2$ respectively. This example is used in this section to explain the training datasets for each of the techniques.

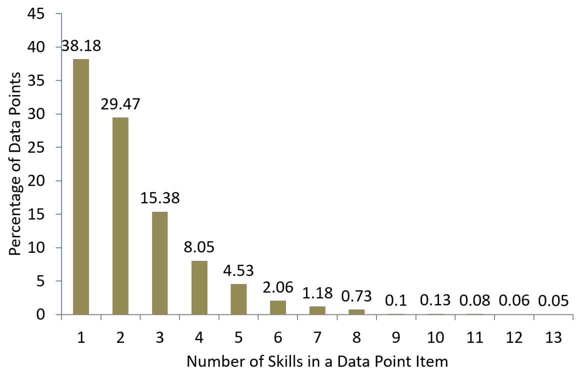

Figure 2: Data Distribution

## 3.1 Bayesian Knowledge Tracing

After DKT [9], authors in [6] have explored and hypothesized the properties of the data which DKT exploits while the standard BKT cannot. To equip BKT with those capabilities, the authors have proposed three variants of BKT: BKT with forgetting (BKT+F), BKT with skill discovery (BKT+S) and BKT with latent-abilities (BKT+A).

We have used the author's implementation of BKT and its three variants published on `https://github.com/robert-lindsey/WCRP/tree/forgetting` to train on our dataset. The data format required by these BKT variants is as shown in table 3. As discussed in the earlier section 1,

| skill ID | response series |
|---|---|
| A | 1 |
| B | 0, 0 |
| C | 0, 1 |

Table 3: BKT data format

BKT is a skill specific model and thus, three models need to be built one each for skills $A$, $B$ and $C$. Each model needs the time series of responses as shown in the table 3.

All variants of BKT except the ones where skill discovery is involved, namely BKT, BKT+F, BKT+A and BKT+FA operate on the skills provided by the data. The remaining variants: BKT+S and BKT+FSA completely ignore the expert tagged skills available in the data. This is achieved by setting the non-parametric prior, $\beta$ on the expert tagged skills as 0.

## 3.2 Performance Factor Analysis

Like BKT, PFA being a skill specific model requires a different model to be built for each skill. Logistic Regression model of [8] is used in the implementation of PFA. For each skill, the response is a function of the skill difficulty, number of prior student success (avoidances) responses and number of prior student failure (committances) responses for the skill. From the implementation point of view, the decision function has two variables - the number of prior success instances and the number of prior failure instances for the skill. Also, a bias is added in the decision function (achieved by the intercept) which serves as the skill difficulty. The data format needed by PFA is as shown in figure 4.

| skill ID | no. of failures | no. of successes | response |
|---|---|---|---|
| A | 0 | 0 | 1 |
| B | 0 | 0 | 0 |
| C | 0 | 0 | 0 |
| B | 1 | 0 | 0 |
| C | 1 | 0 | 1 |

Table 4: PFA data format

## 3.3 Deep Knowledge Tracing

The implementation of LSTM based DKT published on `https://github.com/mmkhajah/dkt` is used to train our dataset. The neural network of DKT requires the input as one hot encoding of skills as well as responses for each of them, while output is the probability of correctness of each of the skills. Hence the size of the input is twice the number of skills and that of the output is the number of skills. The serial number in the table 5 shows the order in which the inputs are fed into the network. The input in the table signifies the previous output while the response shows the expected output out of the network. The odd bits in the input represent one hot encoding of the skills while the even bits represent their responses. $X$ in the output shows that the bit can take either 0 or 1.

| serial no. | input | response |
|---|---|---|
| 1 | 0, 0, 0, 0, 0, 0 | 1, X, X |
| 2 | 1, 1, 0, 0, 0, 0 | X, 0, X |
| 3 | 0, 0, 1, 0, 0, 0 | X, X, 0 |
| 4 | 0, 0, 0, 0, 1, 0 | X, 0, X |
| 5 | 0, 0, 1, 0, 0, 0 | X, X, 1 |

Table 5: DKT data format

As discussed in subsection 2.2 that to figure out the final outcomes for the LGs in an item attempt, there is no clear or fixed ordering. But the time series to be fed into the network of DKT requires us to establish the ordering between them. We sample the orderings randomly and average the results on them. The sample dataset in the table 5 is one such ordering. Another random ordering can be seen in the table 6. The skills of the item $q_1$ are in the order $A$, $B$, $C$ in table 5 while their order is $B$, $A$, $C$ in table 6. The other way to get an ordering is to get rid of the ordering itself by merging the data points of the skills in an item which is explained in the following subsection.

| serial no. | input | response |
|---|---|---|
| 1 | 0, 0, 0, 0, 0, 0 | X, 0, X |
| 2 | 0, 0, 1, 0, 0, 0 | 1, X, X |
| 3 | 1, 1, 0, 0, 0, 0 | X, X, 0 |
| 4 | 0, 0, 0, 0, 1, 0 | X, X, 1 |
| 5 | 0, 0, 0, 0, 1, 1 | X, 0, X |

Table 6: Shuffled skills DKT data format

## 3.4 Multi-skill DKT

As explained in the context of DKT, the orderings among the skills in the item are sampled randomly. In order to get rid of such orderings, we introduce an extension of DKT: Multi-skill DKT which can incorporate the items having multiple skills efficiently. It can be seen from the table 7 that the three data points of $q_1$ and two data points of $q_2$ are consolidated and we are left with two data points in total. The size and structure of the inputs and outputs still remain the

| serial no. | input | response |
|---|---|---|
| 1 | 0, 0, 0, 0, 0, 0 | 1, 0, 0 |
| 2 | 1, 1, 1, 0, 1, 0 | X, 0, 1 |

Table 7: Multi-Skill DKT data

same. The only difference is that the input and output can have the information about multiple skills simultaneously.

## 3.5 Results

For all the algorithms, we use three replications of 2-fold cross validation, which gives us 6 folds in total on which the results are averaged. We use Area under the curve of Receiver Operating Characteristics (ROC), which we will refer as the AUC. Paper [6] discusses the inconsistent procedures used to compute and compare performance of BKT and DKT. We therefore compute AUC both by averaging on all data points and by averaging on skills. The results of our experiments on funtoot dataset are shown in figure 3.

When AUC is averaged on all the data points, the relative difference in performance between algorithms is very low, 0.83 being the lowest and 0.88 being the highest. PFA and DKT share the highest performance of 0.88 AUC. Multi-skill DKT lags a bit behind DKT by 0.03 AUC units (0.85 AUC). All the variants of BKT also lag behind DKT and PFA by not a very big margin, the highest being 0.05 AUC units. BKT has the lowest AUC of 0.83, BKT+FSA has the highest AUC of 0.85 and the rest of them have an AUC of 0.84, which depicts that they all show equivalent performance.

The relative difference in performance between algorithms is higher when AUC is averaged on skills, the lowest being 0.64 AUC of BKT+F and highest being 0.88 AUC of PFA which is 37.5% gain. PFA with an AUC of 0.88 outperforms all the methods by having a minimum gain of 17% (0.75 AUC of DKT and BKT+FSA) and maximum gain of 37.5% (0.64 AUC of BKT+F). Here also, the magnitude of difference between DKT and Multi-skill DKT is very less, 0.04 AUC units to be precise with Multi-skill DKT lagging behind.

With BKT, BTK+F, BKT+A and BKT+FA having AUCs of 0.65, 0.64, 0.68 and 0.67 respectively, it is clear that *Forgetting* adds no value. The number of skills discovered by both BKT+S and BKT+FSA are in the range of $145 - 175$ compared to 442 original skills. The *Skill Discovery* extension provides reasonable gains which are evident from the AUCs of BKT and BKT+S (9% gain) and BKT+FA and BKT+FSA (12% gain). The magnitude of the gains achieved by *Abilities* extension is very less, 0.003 AUC units in the case of BKT, BKT+A and BKT+F, BKT+FA. Finally, the different variants of BKT achieve a gain of maximum 15% over standard BKT. Notably, the best version of BKT, that is, BKT+FSA and DKT, perform equally.

## 4. DISCUSSION AND FUTURE WORK

Our aim of this study was to explore the performance of standard BKT, all of its variants proposed in [6], PFA and DKT on funtoot dataset. The results we have got are in sync with the results in [6]. When the AUC results were computed by averaging over skills, DKT and BKT+FSA perform equally well while DKT outperforms standard BKT with the gain of 15%. Also, BKT+S gave a performance

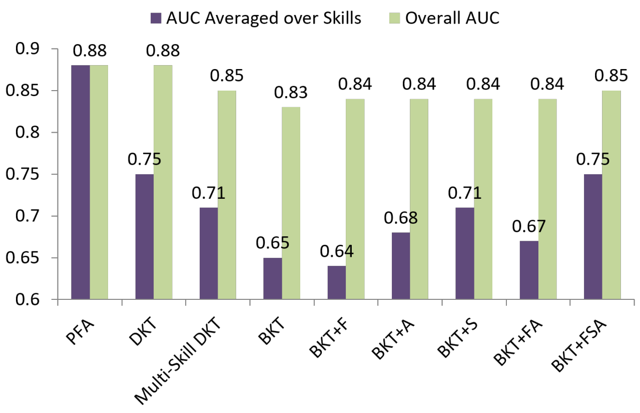

Figure 3: A comparison of PFA, DKT, Multi-skill DKT, BKT and its variants

which was very close to DKT. Though DKT does perform better when the AUC results are averaged over all data points, the magnitude of the gain is significantly low.

Similar kind of results hold true for PFA. PFA achieves a high gain compared to all the models when AUC results are averaged over skills. When AUC results are averaged over all data points, PFA equals DKT's performance and outperforms the rest of the models, though not with a very high margin. This is not consistent with the results in [13] where DKT outperforms PFA though, not overwhelmingly.

The above results reinforce the hypothesis proposed in [6] that the domain of knowledge tracing seems to be shallow and may not require the depth that the deep neural networks offer. The predictive or the explanatory power of a model can also be characterized in terms of the number of parameters the model fits. One of the reasons why DKT is expected to be more successful than other models, at the cost of interpretability, is that it has weights in the order of hundreds of thousands. Moreover, being made up of a layer of LSTM cells, DKT has the capability of looking back arbitrary number of timesteps. On the contrary, variants of BKT and PFA are very simple and interpretable models. Their simplicity can easily be attributed to the small number of parameters they fit.

Standard BKT needs four parameters: *pInit* (the probability that the student is in learned state before the first practice), *pLearn* (the probability that the student transitions from not learned state to the learned state at each practice), *pGuess* (the probability that the student guesses the answer being in the unlearned state) and *pSlip* (the probability that the student accidentally makes a mistake being in the learned state). In PFA, it is even better, only three parameters are learned per skill - item difficulty and one coefficient each for prior failures and successes. With this, the total parameters for a few hundred skills (which is true in our case) would be a few hundred parameters: $three \times number\ of\ skills$. Hence, in our context, it seems appropriate to say that few hundred parameters are better than few hundred thousand parameters.

Both BKT and DKT, in an abstract sense, are the models which maintain the knowledge state of the student. With each response of the student, the knowledge states are updated and those states are used to generate future predictions. They both require the time series data of the student's responses. This is significantly different than the type of data required by PFA. PFA operates on abstract features of student's interactions like total number of prior successes and failures. It occurs to us that the abstract features are smoother than the time series data of responses. It seems the domain of knowledge tracing can be deciphered better if the abstract features are used instead of detailed trail of responses which might be noisy. More studies and experiments are required to validate this point.

The skills used in our experiment are the LGs from the funtoot Knowledge Graph which are tagged at the level of subsubtopic which acts as a context of LG. Also, an LG can occur in multiple subsubtopics. The discovered skills in our experiments of BKT+S and BKT+FSA were in the range of $145 - 175$ which is close to the number of subsubtopics (119) in our dataset. We suspect that there is some relation between the subsubtopics in our dataset and the skills discovered. We would like to investigate this further in future. DKT also supports skill discovery as proposed in [9] which we would look into in future to compare the skills discovered by several algorithms.

Funtoot dataset has items with multiple skills which forced us to extend DKT and come up with Multi-skill DKT. This variant of DKT underperformed marginally as compared to DKT. We do not have a clear understanding about why this is so and hence this also requires further study. Since we have used a very crude dataset, that is, does not contain features about attempts, time durations, hints, item context, etc., it would be interesting to use them with DKT and see if the depth of DKT can exploit them.

## 5. REFERENCES

[1] L. W. Anderson, D. R. Krathwohl, and B. S. Bloom. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.* Allyn & Bacon, 2001.

[2] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[3] R. S. d Baker, A. B. Goldstein, and N. T. Heffernan. Detecting the moment of learning. In *International Conference on Intelligent Tutoring Systems*, pages 25–34. Springer, 2010.

[4] Y. Gong, J. E. Beck, and N. T. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *International conference on intelligent tutoring systems*, pages 35–44. Springer, 2010.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? In *Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016)*, pages 94–101, 2016.

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

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, pages 505–513, 2015.

[10] Y. Qiu, Y. Qi, H. Lu, Z. Pardos, and N. Heffernan. Does time matter? modeling the effect of time with bayesian knowledge tracing. In *Educational Data Mining 2011*, 2010.

[11] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[12] K. H. Wilson, Y. Karklin, B. Han, and C. Ekanadham. Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016)*, pages 539–544, 2016.

[13] X. Xiong, S. Zhao, E. G. Van Inwegen, and J. E. Beck. Going deeper with deep knowledge tracing. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016)*, pages 545–550, 2016.

[14] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *International Conference on Artificial Intelligence in Education*, pages 171–180. Springer, 2013.