Going Deeper with Deep Knowledge Tracing

Xiaolu Xiong, Siyuan Zhao, Eric G.
Van Inwegen, Joseph E. Beck
Worcester Polytechnic Institute
100 Institute Rd
Worcester, MA 01609
508-831-5000
{xxiong, szhao, egvaninwegen,
josephbeck}@wpi.edu

ABSTRACT

Over the last couple of decades, there have been a large variety of approaches towards modeling student knowledge within intelligent tutoring systems. With the booming development of deep learning and large-scale artificial neural networks, there have been empirical successes in a number of machine learning and data mining applications, including student knowledge modeling. Deep Knowledge Tracing (DKT), a pioneer algorithm that utilizes recurrent neural networks to model student learning, reports substantial improvements in prediction performance. To help the EDM community better understand the promising techniques of deep learning, we examine DKT alongside two well-studied models for knowledge modeling, PFA and BKT. In addition to sharing a primer on the internal computational structures of DKT, we also report on potential issues that arise from data formatting. We take steps to reproduce the experiments of Deep Knowledge Tracing by implementing a DKT algorithm using Google's TensorFlow framework; we also reproduce similar results on new datasets. We determine that the DKT findings don't hold an overall edge when compared to the PFA model, when applied to properly prepared datasets that are limited to main (i.e. non-scaffolding) questions. More importantly, during the investigation of DKT, we not only discovered a data quality issue in a public available data set, but we also detected a vulnerability of DKT at how it handles multiple skill sequences.

Keywords

Knowledge tracing, deep learning, recurrent neural networks, student modeling, performance factors analysis, data quality

1. INTRODUCTION

Deep Learning (DL) is an emerging approach within the machine learning research community. A series of deep learning algorithms have been proposed in recent years to move machine learning systems toward the discovery of multiple levels of representation and they already had important empirical successes in a number of traditional AI applications such as computer vision and natural language processing [8]. Much more recently, Google's deep learning networks [7] beat a top human player at the game of Go. Most research in deep learning (e.g. Google's deep learning algorithm) has been focused on the studies of artificial neural networks.

Deep knowledge tracing (DKT), the recent adoption of recurrent neural nets (RNNs) in the area of educational data mining, achieved dramatic improvement over well-known Bayesian Knowledge Tracing models (BKT) and the results of it have been demonstrated to be able to discover the latent structure in skill concepts and can be used for curriculum optimization [1].

Driven by both noble goals (testing the reproducibility of scientific findings) and some selfish ones (how did they do so much better at predicting student performance?!), we set out to take the theories, algorithms, and code from the DKT paper and apply them ourselves to the same data and more data sets. As to the goal of reproducing the findings, we were motivated by studies discussing the importance of reproducibility [5]. In addition to applying DKT to the same data, we also tested the algorithm on a different ASSISTments dataset (which covers data in 2014-2015 school year), as well as the one of data sets from KDD Cup 2010. In our experiments with the original DKT algorithm, we uncovered three aspects of the ASSISTments 2009-2010 data set that, when accounted for, drastically reduce the effectiveness of the DKT algorithm. These can broadly be summarized as 1). an error in reporting the data (wherein rows of data were randomly duplicated). 2). an inconsistency of skill tagging, and 3). the use of information ignored by PFA and BKT. We will discuss these three inconsistencies and their impacts on the prediction accuracies in section 3.

2. DEEP KNOWLEDGE TRACING AND OTHER STUDENT MODELING TECHNIQUES

When describing neural networks, the use of 'deep' conventionally refers to the use of multiple processing layers; the 'Deep' in DKT refers to the recurrent structure of the network and the 'depth' of information over time. This family of neural nets represents latent knowledge state, along with its temporal dynamics, using large vectors of artificial neurons, and allows the latent variable representation of student knowledge to be learned from data rather than hard-coded.

Typical RNNs suffer from the now famous problems of vanishing and exploding gradients, which are inherent to deep networks. Figure 1 shows an unrolled RNN; there are loops at hidden layers, allowing information to be retained; this is the 'depth' of an RNN. When building a deep neural net, the standard activation functions , and cumulative backpropagation error signals either shrink rapidly or grow out of bounds. i.e., they either decay or grow exponentially ('vanish' or 'explode'). Long short-term memory (LSTM) model [14] is introduced to deal with the vanishing gradient problem; it also achieves remarkable results on many previously un-learnable tasks. LSTM, a variation of recurrent neural networks, contains LSTM units in addition to regular RNN units. LSTM units have two unique gates: forget and input gates

to determine when to forget previous information, and which current information is important to remember.

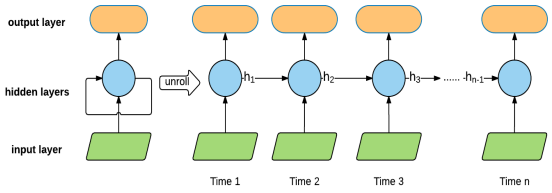

Figure 1. An unrolled Recurrent Neural Network (RNN)

The idea behind LSTM is simple. Some of the units are called constant error carousels (CEC). Each CEC uses an activation function *f*, the identity function, and has a connection to itself with fixed weight of 1.0. Due to *f*'s constant derivative of 1.0, errors backpropagated through a CEC cannot vanish or explode but stay the same magnitude. CECs are connected to several nonlinear adaptive units needed for learning nonlinear behavior. Weight changes of these units often profit from error signals, which propagate far back in time through CECs. CECs are the main reason why LSTM nets can learn to discover the importance of (memorize) events that happened thousands of discrete time steps ago while previous RNNs routinely fail in cases of minimal time lags of 10 steps. LSTM learns to solve many previously un-learnable DL tasks and clearly outperformed previous RNNs on tasks both in terms of reliability and speed [1].

In the DKT algorithm, at any time step, the input to RNNs is the student performance on a single problem of the skill that the student is currently working on. Since RNNs only accept fixed length of vectors as the input, we used one-hot encoding to convert student performance into fixed length of vectors whose all elements are 0s except for a single 1. The single 1 in the vector indicates two things: which skill was answered and if the skill was answered correctly. This data presentation draws a clear distinction between DKT and other student modeling methods, such as Bayesian Knowledge Tracing and Performance Factor Analysis.

The Bayesian Knowledge Tracing (BKT) model [10] is a 2-state dynamic Bayesian network where student performance is the observed variable and student knowledge is the latent data. The model takes student performances and uses them to estimate the student level of knowledge on a given skill. The standard BKT model is defined by four parameters: initial knowledge and learning rate (learning parameters) and slip and guess (mediating parameters). The two learning parameters can be considered as the likelihood the student knows the skill before he even starts on an assignment (initial knowledge, K_0) and the probability a student will acquire a skill as a result of an opportunity to practice it (learning rate). The guess parameter represents the fact that a student may sometimes generate a correct response in spite of not knowing the correct skill. The slip parameter acknowledges that even students who understand a skill can make an occasional mistake. Guess and slip can be considered analogous to false positive and false negative. BKT typically uses the Expectation Maximization algorithm to estimate these four parameters from training data. Based on the estimated knowledge, student performance at a particular practice opportunity can be calculated except the very first one, which only apples the value of K_0.

Skills vary in difficulties and amount of practices needed to master, so values for four BKT parameters are skill dependent. This lead to one major weakness of BKT [11]: it lacks the ability to handle multi-skill questions since it works by looking at the historical observation of a skill and cannot accommodate all skills simultaneously. One simple workaround is treating the multiple skill combination as a new joint skill and estimate a set of parameters for this new skill. Another common solution to this issue is to associate the performance on multiple skill questions with all required skills, by listing the performance sequence repeatedly [12]. This makes the model see this piece of evidence multiple times for each one of required skills. As a result, a multiple skill question is multiple single skill questions.

Another popular student modeling approach is the Performance Factors Analysis Model (PFA) [9]. PFA is a variant of learning decomposition, based on a reconfiguration of Learning Factor Analysis. Unlike, BKT, it has the ability to handle multiple skill questions. Briefly speaking, it uses the form of the standard logistic regression model with the student performance as the dependent variable. It reconfigures LFA (Learning factors analysis) [13] on its independent variables, by dropping the student variable and replaces the skill variable with question identity. This model estimates parameters for each item's difficulty and also two parameters for each skill reflecting the effects of the prior correct and incorrect responses achieved for that skill. Previous work that compares KT and PFA have shown that PFA to be the superior one [11]. One reason is the richer feature set that PFA can utilize and the fact that learning decomposition models are ensured to reach global maxima while the typical fitting approach of BKT is no guarantee of finding a global, rather than a local maximum.

Beside the theoretical comparison of DKT, BKT, and PFA, we can also compare them visually by looking at the differences between them in terms of inputs data. Consider a simple scenario that a student answers two questions from two skills each, Tables 1-3 compare different training data formats for these three modeling methods under that same scenario of student responses.

Table 1. An example of BKT's training data

Model ID	Skill ID	Response Sequence
1	A	1,0
2	B	0,1

Table 2. An example of PFA's training data

Index ID	Skill ID	Prior Correct	Prior Incorrect	Difficulty	Correct
1	A	0	0	0.7	1
2	A	1	0	0.75	0
3	B	0	0	0.6	0
4	B	0	1	0.65	1

Table 3. An example of DKT's training data

Index ID	One-hot encoding
1	1,0,0,0
2	0,0,1,0
3	0,0,0,1
4	0,1,0,0

3. METHODOLOGY AND DATA SETS

3.1 Implementation of Deep Knowledge Tracing in Tensorflow

The original version of DKT (Lua DKT[1]) was implemented in Lua scripting language using Torch framework and its source code has been released to the public. In order to have a comprehensive understanding of the DKT model, we decided to replicate and implement DKT model in Python and utilize Google's TensorFlow API [3] to help us with building neural networks. TensorFlow is Google Brain's second generation machine learning interface; it is flexible and can be used to express a wide variety of algorithms.

Our implementation of DKT in TensorFlow (TensorFlow DKT[2]) can be described as a directed graph, which is composed of a set of nodes. The graph represents a data flow computation, with extensions for allowing certain nodes to maintain and update persistent state and for branching and looking control, this is crucial for allowing RNN nodes to work on sequential data. In the directed graph, each node has zero or more inputs and zero or more outputs and represents the instantiation of an operation. An operation represents an abstract computation. In our implementation of DKT model, we adapted the loss function of the original DKT algorithm. It has 200 fully-connected hidden nodes in the hidden layer. To speed up the training process, we used mini-batch stochastic gradient descent to minimize the loss function. The batch size for our implementation is 100. For one batch, we randomly select data from 100 students in our training data. After the batch finishes training, 100 students in the batch are removed from the training data. We continue to train the model on next batch until all batches are done. Just as in the original Lua implementation, Dropout [4] was also applied to the hidden layer to avoid over-fitting.

4. DATA SETS

4.1 ASSISTments 2009-2010 Data Set

The original DKT paper conducted one of three of experiments using the ASSISTments 2009-2010 skill builder data set [16]. This data set was gathered from ASSISTments' skill builder problem sets, in which a student achieves mastery by working on similar (often isomorphic) questions until they can correctly answer n right in a row (where n is usually 3). After mastery, students do not commonly rework the same skill. This dataset contains 525,535 rows of student responses; there are 4,217 student ID's and 124 skills. Lua DKT achieved an AUC of 0.86 and noticeably outperformed BKT (AUC = 0.67) on this data set. However, during our investigation on the DKT source code and application, we believe we discovered three issues that have unintentionally inflated the performance of Lua DKT. These issues are:

4.1.1 Duplicated records

To our surprise and dismay, we found that the ASSISTments 2009-2010 data set has a serious issue of quality: large chunks of records are duplications that should not be there for any reason (e.g. see records of order id 36369610). These duplicated rows have the same information but only differ on the "opportunity" and "opportunity_original"; these two features record the number of opportunities a student has practiced on a skill and the number of practices on main problems of a skill respectively. It is impossible to have more than one 'opportunity' count for a single order id. This is definitely an error in the data set and these duplicated records should not be used in any analysis or modeling studies. We counted there are 123,778 rows of duplications out of 525,535 in the data set (23.6%). The existence of duplicated data is an avoidable oversight and ASSISTments team has acknowledged this error on their website. All new experiments in this work and following discussions exclude data of these duplications.

4.1.2 Mixing main problems with scaffolding problems

A mastery learning problem set normally contains over a hundred of main problems, and each main problem may have multiple associated scaffolding problems. Scaffolding problems were designed to help students acquire an integrated set of skills through processes of observations and guided practice; they are usually tagged with different skills and have different designs from the main problems. Because of the difference in usage, scaffolding questions should not be treated as the same as main problems. Student modeling methods such as BKT and PFA exclude scaffolding features. The experiment conducted by Lua DKT did not filter out scaffolding problems. This means that Lua DKT had the advantage of additional information; thus, the prediction results cannot be compared fairly with BKT. There are 73,466 rows of records of scaffolding problems.

4.1.3 Repeated response sequences with different skill tagging (Duplication by skill tag)

The 2009-2010 skill builder dataset was created as a subset of the 2009-2010 full dataset. The full dataset from 2009-2010 includes student work from both skill builder assignments (where a student works until a mastery threshold is reached) and more traditional assignments (where a student has a fixed number of problems). Any problem (or assignment) can be tagged with any number of skill tags. Typically, problems have just one skill tag; they seldom are tagged with two skills; they are very rarely tagged with three or more. Depending on the design of the content creator, a problem set may have multiple skill tags; many assignments - especially skill builders - will have the same skill tag for all problems. When the full dataset was decomposed into only mastery style assignments, the problems, and assignments that were tagged with multiple skills were included with a single tag, but repeated for each skill. This means that the sequence of action logs from one student working on one assignment was now repeated once per skill. For models such as RNNs that operate over sequences of vectors and memory on the entire history of

[1] https://github.com/chrispiech/DeepKnowledgeTracing

[2] https://github.com/siyuanzhao/2016-EDM

previous inputs, the issue of duplicated sequences is going to add additional weight onto the duplicated information; this will have undesired effects on RNN models.

For an example, suppose we have a hypothetical scenario that a student answers two problems which have been tagged with skill "A" and "B"; he answers first one correctly and the next one incorrectly. Table 4 shows the data set where responses have been repeated on skill "A" and "B". This format of data can be used in BKT models since BKT can build two models for skill "A" and "B" separately. When applying this sequential data set to DKT, we believe DKT can recognize the pattern when a problem tagged with skill "B" follows a problem tagged with "A"; the skill "B" problem has an extremely high chance to repeat skill "A" problem's response correctness. Note that skill ID can be mapped to skill names, but the order of skill ID is completely arbitrary.

Table 4. An example of repeated multiple-skill sequence

Index ID	Skill ID	Problem ID	Correctness
1	A	3	1
1	B	3	1
2	A	4	0
2	B	4	0

One approach to change the way of how multiple-skill problems are handled is to simply use the combination of skills as a new joint skill. Table 5 shows the data set which uses a joint skill of A and B. In this case, DKT no longer has access to repeated information. PFA and BKT can also adapt this format of data too.

Table 5. An example of joint skills on multiple-skill problems

Index ID	Skill ID	Problem ID	Correctness
1	A, B	3	1
2	A, B	4	0

Table 6. Three variants of ASSISTments 2009-2010 Data set

	09-10 (a)	09-10 (b)	09-10 (c)
Has duplicated records	No	No	No
Has scaffolding problems	Yes	No	No
Repeated multiple-skill sequences	Yes	Yes	No
Joint skills from multiple-skill	No	No	Yes

In order to understand the impact of having scaffolding problems and two approaches to dealing with multiple-skill problems, we generate three different data sets (namely 09-10 (a), 09-10 (b), 09-10 (c)) derivate from the ASSISTments 2009-1010 data set, as summarized in Table 6.

4.2 ASSISTments 2014-2015 Data Set

Even without the issue of duplicate rows, 2009-2010 skill builder set has lost its timeliness and certainly cannot represent the latest student data in an intelligent tutoring system. So we gathered another data set that covers 2014-2015 school years' student response records [16]. In this experiment, we randomly selected 100 skills from this year's data records. This data set contains 707,944 rows of records; each record represents a response to a main problem in a mastery learning problem set. Each problem set has only one associated skill and we take caution to make sure there is no duplicated row in this data set. We suspect this new data set contains different information that covers student learning patterns, item difficulties and skill dependencies.

4.3 KDD Cup 2010 Data Set

Our last data set comes from the Cognitive Algebra Tutor 2005-2006 Algebra system [6]. This data was provided as a development dataset in the KDD Cup 2010 competition. Although both ASSISTments and Cognitive Algebra Tutor involve using mathematics skills to solve problems, they are actually rather different from each other. ASSISTments serves primarily as computer-assisted practice for students' nightly homework and review lessons while the Cognitive Tutor is part of an integrated curriculum and has more support for learners during the problem-solving process. Another difference in terms of content structure is that the Cognitive Tutor presents a problem to a student that consists of questions (also called steps) of many skills. The Cognitive Tutor uses Knowledge Tracing to determine when a student has mastered a skill. A problem in the tutor can consist of questions of different skills, once a student has mastered a skill, as determined by KT, the student no longer needs to answer questions of that skill within a problem but must answer the other questions which are associated with the un-mastered skills. The number of skills in this dataset is substantially larger than the ASSISTments dataset [15]. One issue of using KDD data on PFA is how to estimate item difficulty feature. In this work, we use a concatenation of problem name and step name. However many such pairs are only attempted by 1 student and the difficulty values of these items are either 1.0 or 0.0, leading to both over-fitting and data leakage. To fix that, we replace difficulty values of these items with skills' difficulty information. Filtering out rows with missing values resulting in 607,026 rows of data with students responded correctly at 75.5% of the time. This KDD data set has 574 students worked on 436 skills in mathematics. The complete statistic information of five data sets can be found in Table 7.

Table 7. Data set statistics

	# records	# Students	# Skills
09-10 (a)	401,757	4,217	124
09-10 (b)	328,292	4,217	124
09-10 (c)	275,459	4,217	146
14-15	707,944	19,457	100
KDD	607,026	574	436

5. RESULTS

Student performance predictions made by each model are tabulated and the accuracy was evaluated in terms of Area Under Curve (AUC) and the square of Pearson correlation (r^2). AUC and r2 provide robust metrics for evaluation predictions where the value being predicted is either a 0 or 1 also represents different information on modeling performance. An AUC of 0.50 always represents the scored achievable by random chance. A higher AUC score represents higher accuracy. r^2 is the square of Pearson correlation coefficient between the observed and predicted values of dependent variable. In the case of r^2, it is normalized relative to the variance in the data set and it is not directly a measure of how good the modeled values are, but rather a way of measuring the proportion of variance we can explain using one or more variables. r^2 is similar to root mean squared error (RMSE) but is more interpretable. For example, it is unclear whether an RMSE of 0.3 is good or bad without knowing more about the data set. However, an r2 of 0.8 indicates the model accounts for most of the variability in the data set. Neither AUC nor r^2 method is a perfect evaluation metric, but, when combined, they account for different aspects of a model and provide us a basis for evaluating our models.

Experiments on every data set have been 5-fold student level cross-validated and all parameters are learned from training data. We used EM to train BKT and the limit of iteration was set to 200. Besides the number of hidden nodes and the size of mini-batch parameters we have discussed, we set the number of epochs of DKT to 100.

The cross-validated model predictions results are shown in Table 8 and Table 9. As can be seen, DKT clearly outperforms BKT on all data sets, but the results are no longer overwhelmingly in favor of DKT (both implementations). Note that Lua DKT implementation which we can access uses regular RNN nodes; TensorFlow DKT uses LSTM nodes.

Table 8. AUC results

	Torch DKT	TensorFlow DKT	PFA	BKT
09-10 (a)	0.79	0.81	0.70	0.60
09-10 (b)	0.79	0.82	0.73	0.63
09-10 (c)	0.73	0.75	0.73	0.63
14-15	0.70	0.70	0.69	0.64
KDD	0.79	0.79	0.71	0.62

Table 9. r^2 results

	Lua DKT	TensorFlow DKT	PFA	BKT
09-10 (a)	0.22	0.29	0.11	0.04
09-10 (b)	0.22	0.31	0.14	0.07
09-10 (c)	0.14	0.18	0.14	0.07
14-15	0.10	0.10	0.09	0.06
KDD	0.21	0.21	0.10	0.05

On the ASSISTments data sets, average DKT prediction performance across two implementations is better than PFA and it is not affected by removing scaffolding, as we change dataset from 09-10 (a) to 09-10 (b). On the other hand, PFA's performance increases from 0.70 to 073 in AUC and 0.11 to 0.14 in r^2 ($p \leq 0.05$), we believe that removing scaffolding helps reducing noise from data and provides PFA with a dataset with lower variance. When we switch to dataset 09-10 (c) where multiple skills were combined into joint skills, the performance of DKT suffers a noticeable hit, average AUC and average r^2 drop from 0.81 to 0.74 and from 0.30 to 0.18 respectively. This observation confirms our suspicion on repeated response sequence inflating the performance of DKT models. On the 09-10 (c) dataset and 14-15 dataset where no repeated response sequences and scaffolding problems, we notice that PFA performs as well as DKT.

A deeper way of looking at the impact of repeated response sequences on data set 09-10 (b) is splitting the prediction results into two, the predictions of leading records and repeated data points. We see that predictions on repeated data points (e.g. skill "B" problems in Table 4) have nearly perfect performance metrics (AUC = 0.97, r^2 = 0.74). On the other hand, the leading records (e.g. skill "A" problems in Table 4) have much lower prediction results (AUC = 0.77, r^2 = 0.23). That said, we also notice these numbers are still higher than 09-10 (c)'s results, which uses joint skill tags to avoid repeated sequences. One can explain this as making DKT to model skills individually can cause data duplications but it also can have benefits on building skill dependencies over time and use such information to make better predictions.

On the KDD dataset, the performance results of two DKT implementations are definitely better than both BKT and PFA ($p \leq 0.05$). There are a few possible reasons for this performance gap between PFA and DKT. First of all, as we have mentioned, we have to adjust item difficulty values for many problems in order to avoid overfitting and data leakage, which leads to the lower predictive power of that feature and lower PFA performance. Another possible explanation of DKT is winning on KDD data set is that DKT can better exploit step responses. The structure of KDD data set made it is difficult to distinguish "main problems" and "scaffolding problems", thus PFA is unable to have a more unified data set for this part of the experiment. That said, the advantage of DKT shows its power on complicated and realistic data sets.

6. DISCUSSION AND CONTRIBUTION

Within this paper, we have compared two well-studied knowledge modeling methods with the emerging Deep Knowledge Tracing algorithm. We have compared these models in terms of their power of predicting student performance in 5 different data sets. Contrary to our expectation, the DKT algorithm did not achieve overwhelmingly better performance when compared to PFA model on ASSISTments data sets when they are properly prepared. DKT appears to perform much better on KDD dataset, but we believe this is due to PFA model undermined by inaccurate item difficulty estimation.

A second interesting finding is that when DKT is fed repeated response sequences derived from the transformation of problems tagged with multiple skills, the overall performance of DKT is certainly better than PFA and BKT. Our explanation is that DKT's implementation backbone, RNNs, has the power of

remembering exact patterns of sequential data and could thus inflate prediction performance on responses tagged with multiple skills and repeated per skill. More discussion and special attention are required when handling multiple skill problems in DKT algorithm.

Last, but not least, during the investigation of DKT, we discovered an issue in data quality arising from duplicated information in a publicly available data set. The duplication issues (caused by unclear transformational rules and some other as-of-yet-to-be-ascertained cause) allowed us a natural experiment to examine the impact of duplications on the robustness of these algorithms. These discoveries (the data duplications and their subsequent impact) should serve as a reminder of the importance of data preprocessing and transformation procedures in the work of knowledge discovery and data mining. Or, put another way, while we advance new algorithms and fine tune their parameters, we should also consider (and, if possible, report on) the robustness of the algorithms to common data glitches.

7. FUTURE WORK AND CONCLUSION

There are several directions for further research in the area of DKT modeling. Prior work [2] has shown that the use of context-dependent RNN language model improved the performance in the task of the Wall Street Journal speech recognition task. More features like student features (e.g. prior knowledge, completion rates, time on learning, etc.), and content features (problem difficulty, skill hierarchies, etc.) may be available and could be used. A context-dependent DKT implementation could be created by adding an extra input vector containing these features.

Another open area for future work is that DKT and other deep learning algorithms are not restricted to one kind of output or application. It is also possible that we could apply deep learning algorithms on other modeling challenges such as wheel spinning, mastery speed, and affect detection.

In conclusion, our work here focuses on a primitive investigation of DKT and aims to provide us deeper insight on how DKT works. Overall, this paper suggests that DKT remains a promising approach to modeling student knowledge; however, we see that data which contains problems tagged with multiple skills has to be dealt carefully in DKT modeling. But, considering that this implementation of DKT: a) only relied on the sequences of student responses (just as BKT does) and no other information on skills and problems and b) performs substantially better than BKT and as good as PFA, we believe that DKT has great potential to outperform other methods when it utilizes more features.

8. ACKNOWLEDGEMENTS

We thank multiple current NSF grants (ACI-1440753, DRL-1252297, DRL-1109483, DRL-1316736, DGE-1535428 & DRL-1031398), the US Dept. of Ed (IES R305A120125 & R305C100024 and GAANN), and the ONR.

9. REFERENCES

[1] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep Knowledge Tracing. In Advances in Neural Information Processing Systems (pp. 505-513).

[2] Mikolov, T., & Zweig, G. (2012, July). Context dependent recurrent neural network language model. In SLT (pp. 234-239).

[3] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., & Ghemawat, S. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. White paper, Google Research.

[4] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

[5] Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. Science, 349(6251), aac4716.

[6] Stamper, J., Niculescu-Mizil, A., Ritter, S., Gordon, G.J., & Koedinger, K.R. (2010). Algebra I 2005-2006. Challenge data set from KDD Cup 2010 Educational Data Mining Challenge. Find it at http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

[7] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.

[8] Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009, June). Curriculum learning. In Proceedings of the 26th annual international conference on machine learning (pp. 41-48). ACM.

[9] Pavlik Jr, P. I., Cen, H., & Koedinger, K. R. (2009). Performance Factors Analysis-A New Alternative to Knowledge Tracing. Online Submission.

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

[11] Gong, Y., Beck, J. E., & Heffernan, N. T. (2010, June). Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In Intelligent tutoring systems (pp. 35-44). Springer Berlin Heidelberg.

[12] Heathcote, A., Brown, S., & Mewhort, D. J. K. (2000). The power law repealed: The case for an exponential law of practice. Psychonomic bulletin & review, 7(2), 185-207.

[13] Cen, H., Koedinger, K., & Junker, B. (2006, June). Learning factors analysis–a general method for cognitive model evaluation and improvement. In Intelligent tutoring systems (pp. 164-175). Springer Berlin Heidelberg.

[14] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[15] Pardos, Z. A., & Heffernan, N. T. (2011). KT-IDEM: Introducing item difficulty to the knowledge tracing model. In User Modeling, Adaption and Personalization (pp. 243-254). Springer Berlin Heidelberg.

[16] ASSISTments Data. (2015). Retrieved March 07, 2016, from https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010