ABSTRACT

Knowledge tracing (KT) models, e.g., the deep knowledge tracing (DKT) model, track an individual learner’s acquisition of skills over time by examining the learner’s performance on questions related to those skills. A practical limitation in most existing KT models is that all questions nested under a particular skill are treated as equivalent observations of a learner’s ability, which is an inaccurate assumption in real-world educational scenarios. To overcome this limitation we introduce qDKT, a variant of DKT that models every learner’s success probability on individual questions over time. First, qDKT incorporates graph Laplacian regularization to smooth predictions under each skill, which is particularly useful when the number of questions in the dataset is big. Second, qDKT uses an initialization scheme inspired by the fastText algorithm, which has found success in a variety of language modeling tasks. Our experiments on several real-world datasets show that qDKT achieves state-of-art performance on predicting learner outcomes. Because of this, qDKT can serve as a simple, yet tough-to-beat, baseline for new question-centric KT models.

1. INTRODUCTION

Knowledge tracing (KT) models are useful tools which provide educators with actionable insights into learners’ progress [24, 17]. Given a learner’s performance history, these methods predict their proficiency across a predetermined set of skills (i.e., knowledge components or concepts). One of the most popular methods for tracking this cognitive development is the Bayesian Knowledge Tracing (BKT) framework [3, 16, 24] which applies hidden Markov models [1] to learn each learner’s guess, slip, and learn probabilities for each skill. Another approach to modeling the dynamics of skill acquisition is SPARFA-Trace [11] which uses Kalman filtering [9] to model learner skill acquisition. An advantage of SPARFA-Trace is that it can, unlike BKT models, relate individual questions to multiple skills. Recently, deep learning techniques have been applied to the KT problem to create Deep Knowledge Tracking (DKT) [19] which models the sequence prediction task using a Long Short-Term Memory (LSTM) network [8].

All of the aforementioned KT models track an individual learner’s knowledge at the skill level. Under the KT framework, the time series data modeled consists of learner skill interaction sequences, given by $X_t = \{(s_i, a_i)\}_{t=1}^T$ where $s_i$ is the skill index attempted by the $i^{th}$ learner at discrete time step $t$, while $a_i \in \{0, 1\}$ is the assessment of the learner’s response, with 0 indicating an incorrect response and 1 indicating a correct response.

The key assumption underpinning these models is that all questions nested under a particular skill are equivalent. This assumption, however, is generally unrealistic in real-world educational datasets. First, a mapping of questions to skills is not always available and obtaining such a mapping requires the intervention of subject matter experts, which is both costly and time-consuming. Second, questions in real-world educational datasets are never homogeneous, but rather exhibit significant variations in difficulty and discrimination [5]. In other words, different questions convey differing levels of information about a particular learner’s mastery of the underlying skill, and methods for modeling learner’s acquisition of skills over time should take such information into account.

However, simply substituting questions for skills in a traditional KT model is insufficient to accomplish this goal. To illustrate this, we selected two commonly used educational datasets, ASSISTments2009 and ASSISTments2017 [7]. We first ran the standard DKT model using the skill-level information provided with each dataset. We then re-ran the DKT model but used the question identifiers themselves, rather than the skills, for modeling performance. Concretely, the time series data modeled consisted of learners’ question interaction sequences, given by $X_t = \{(q_i, a_i)\}_{t=1}^T$, where $q_i$ denotes the question answered by learner $i$ at time $t$. The AUC for both of these model variants are shown in Table 1.

We note that for the ASSISTments 2017 dataset that this question-centric approach provides a moderate improvement in AUC but for the ASSISTments 2009 dataset the question-
Our main contributions are summarized as follows: it is apparent that question-level modeling has the potential to overfitting when the number of observations per question is small.

Table 1: AUC scores for DKT vs its variant with qDKT performs at par with the state-of-the-art skill-level DKT model on ASSISTments 2009 dataset, and improves it by 5% and 6% on ASSISTments 2017 dataset and Statics 2011 dataset respectively.

Coupling the fastText-inspired scheme with the Laplacian regularizer, qDKT gives gains of 2% in AUC score as compared to the skill-centric DKT model for ASSISTments 2009, while also capturing question-specific characteristics.

2. PROBLEM STATEMENT AND DKT OVERVIEW

Each learner’s performance record contains the questions attempted, time at which each question was attempted, and the assessment of each response (either correct or incorrect). Also, assume that the skill associated with every question is known.

2.1 DKT Model

DKT uses an LSTM to predict a learner’s future performance using their previous assessment history. As discussed earlier, the input to the model is a time series which consists of learners’ skill interaction sequences, given by $X_t = \{(s_i, a_i)\}_{i=1}^T$. Here we restrict our discussion to a single learner and will omit the superscript $i$ throughout.

The forward equations of the DKT model are given in [1]–[7]:

$$x_t = W_v \cdot v_t,$$ (1)

$$i_t = \sigma(W_x x_t + U_i h_{t-1} + b_i),$$ (2)

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f),$$ (3)

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o),$$ (4)

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma'(W_x x_t + U_i h_{t-1} + b_i),$$ (5)

$$h_t = o_t \odot c_t,$$ (6)

$$y_t = \sigma(W_y h_t + b_y),$$ (7)

where $\sigma$ is the sigmoid function, $\sigma'$ is the hyperbolic tangent function, and the operator $\odot$ denotes the element-wise multiplication. In words, the input at time step $t$ is the skill interaction tuple $(s_i, a_i)$ which is encoded by an arbitrary high-dimensional one-hot vector, $v_t \in \{0, 1\}^{2M}$, where $M$ is the number of skills. Using an embedding matrix, $W_{x,v} \in \mathbb{R}^{K \times 2M}$, $v_t$ is mapped to a low-dimensional vector, $x_t \in \mathbb{R}^K$, $K \ll M$ (2), which serves as the input to the LSTM cell. $x_t$ is passed through each of the input, forget, and output gates and, in the end, the LSTM returns $h_t$ – the estimate of the learner’s current knowledge state.

The final output of the model is $y_t \in \mathbb{R}^M$ which predicts the learner’s success probabilities for all the $M$ skills for the next time step $t + 1$.

2.1.1 Loss in the DKT Model.

The output of the DKT model, $y_t$, predicts the learner’s proficiency over the skills for the next time step $t + 1$. During training, the assessment $(a_{t+1})$ of the learner’s response to
the question indexed by \( q_{t+1} \) is known beforehand. The success probability for the skill associated with \( q_{t+1} \) is given by \( y_i[s_{t+1}] \). Since DKT assumes that mastery in the skill is equivalent to mastery in any of the questions under it (i.e., all questions under a skill are equivalent), a trained DKT model should predict the success probability at the skill to be the same as the assessment. This rationale motivates the model should predict the success probability at the skill to be outputted by \( y \), where \( y \) is the assessment of the response to question \( q \). For each skill, Let a learner’s question interaction sequence \( X \) denotes the question answered at time \( t \) and \( a_t \in \{0, 1\} \) is the assessment of the response to question \( q_t \). Our goal is to output \( y_t \in R^N \) which predicts the learner’s success probabilities for all the \( N \) questions at the next time step \( t + 1 \). qDKT utilizes the same architecture as DKT as specified in \( 4 \) - \( 7 \), but with \( v_i \in \{0, 1\}^{2N} \), \( W_{xv} \in R^{K \times 2N} \), and \( y \in R^N \). The updated loss \( \ell_t \) from \( 8 \) at time \( t \) is then given by:

\[
\ell_t = l(y_i[q_{t+1}], a_{t+1}) \tag{8}
\]

where \( l \) is binary cross-entropy loss.

### 2.2 Proposed Model - qDKT

We now introduce our proposed method for KT modeling at the question-level, which we dub qDKT. Our method considers a modified problem statement where we estimate a learner’s success probability for each question rather than for each skill. Let a learner’s question interaction sequence \( X = \{(q_t, a_t)\}_{t=1}^{T-1} \) until time step \( T - 1 \) be given, where \( q_t \) denotes the question answered at time \( t \) and \( a_t \in \{0, 1\} \) is the assessment of the response to question \( q_t \). Our goal is to output \( y_t \in R^N \) which predicts the learner’s success probabilities for all the \( N \) questions at the next time step \( t + 1 \). qDKT utilizes the same architecture as DKT as specified in \( 4 \) - \( 7 \), but with \( v_i \in \{0, 1\}^{2N} \), \( W_{xv} \in R^{K \times 2N} \), and \( y \in R^N \). The updated loss \( \ell_t \) from \( 8 \) at time \( t \) is then given by:

\[
\ell_t = l(y_i[q_{t+1}], a_{t+1}) \tag{9}
\]

We shall refer to this model as the base qDKT model where the prefix \( q \) represents question-level modeling.

### 3. REGULARIZATION FOR qDKT

As seen in Table \( 1 \) the base qDKT model performs poorly for datasets with both a large number of questions and a small number of observations per question. To overcome this, we propose a regularization method for qDKT to combat overfitting. It is reasonable to assume that success probabilities of multiple questions associated with the same skill should not be significantly different for a given learner. Based on this premise, we regularize the variance in success probabilities for questions that fall under the same skill.

\[
R(y) = \sum_{i \in Q} \sum_{j \in Q} 1(i, j) \cdot (y_i - y_j)^2 \tag{10}
\]

where vector \( y \in R^N \) contains success probabilities of all questions \( Q \) in the dataset, \( i, j \in Q \) and \( 1(i, j) \) is 1 if \( i, j \) fall under the same skill, otherwise it is 0.

We add this penalty to the loss and use \( \lambda \) to control the weight of the penalty. Thus, the updated loss function from \( 8 \) with the regularization penalty is:

\[
\ell = l + \lambda \cdot R(y) \tag{11}
\]

#### 3.0.1 Interpretation of the regularizer

Graph theory provides a clean interpretation for the regularization penalty which is also helpful for speeding up its computation. We construct a graph \( G \) with number of nodes equal to the number of questions in the dataset. Two nodes are connected with an edge of weight 1 if the questions are associated with the same skill and with an edge weight of 0 otherwise.

The degree matrix \( D \) of a graph \( G \) is a diagonal matrix with

\[
d_{ii} = \sum_{j \in C_i} w_{ij},
\]

where \( w_{ij} \) is the similarity between node \( i \) and node \( j \) (edge weight), \( C \) is the set containing all the indices directly connected with \( i \) (immediate siblings). The adjacency matrix \( A \) of a graph \( G \) stores the edge weights \( w_{ij} \). Given the degree matrix \( D \) and the adjacency matrix \( A \) of a graph \( G \), the Laplacian matrix \( L \) is defined as:

\[
L = D - A.
\]

Then for any vector \( v \),

\[
v^T L v = \sum_{i,j} w_{ij} \cdot (v_i - v_j)^2. \tag{12}
\]

We can then use \( 12 \) to simplify the regularization penalty of \( 10 \).

\[
R(y) = \sum_{i \in Q} \sum_{j \in Q} 1(i, j) \cdot (y_i - y_j)^2 = y^T L y. \tag{13}
\]

The simplification of the double summation term to a condensed vector-matrix multiplication term is useful to speed up its calculation, especially while training the qDKT model on GPUs.

Further, our approach to model similarity works even when questions are associated with multiple skills. This provides additional flexibility over previous KT models that restrict each question to be associated to exactly one skill. Such flexibility is important for real-world applications where questions commonly evaluate learners on multiple skills simultaneously. Moreover, this formulation can be helpful to incorporate even other measures of similarity like tf-idf similarity \( 14 \) using question text.

### 4. INITIALIZATION FOR qDKT

DKT maps each skill interaction tuple to \( x \in R^d \) via the matrix \( W_{xv} \) (see \( 4 \)). In DKT, the entries of \( W_{xv} \) are initialized with draws from a standard normal distribution. While this approach is straightforward, random embeddings tend to perform extremely poorly in high dimensions where the optimization problem will have an extremely large number of saddle points \( 4 \). To overcome this limitation, we propose a more effective method for initializing \( W_{xv} \) inspired by the fastText architecture.

#### 4.1 Overview of Language Modeling and fastText

In NLP, language models are used to predict the most likely words that can follow a given sequence of words. Such models are often initialized with word embeddings from algorithms like word2vec \( 15 \), fastText and GloVe \( 18 \). At a high level, these algorithms embed words into a high dimensional space such that words that have close semantic relationships will be embedded near one another, while words with low semantic similarity will be embedded further apart \( 6 \).
Let set \( Q \) contain all the question ids and set \( U \) contain all characters. We define a one-to-one mapping \( f: Q \rightarrow U \) which maps a question id to a unique character. To convert learners’ question interaction sequences, \( X = \{(q_t, a_t)\}_{t=1}^T \) into a text corpus, we apply a signal transformation \( Y \) on \( X \) such that \( y_t = f(q_t) + a_t \) where ‘+’ denotes the string concatenation operator. Thus, each question interaction is encoded as a two character string consisting of the question id and the graded response. This interaction encoding constitutes the “words” of our corpus. The “sentences” of our corpus constitute of the string of such encoded interactions by an individual learner. We finally apply fastText to this newly generated “corpus”. For a given question interaction \((q_t, a_t)\), fastText will train the embeddings of the following n-grams \( \{f(q), f(q)+0\} \). Thus, we link the embeddings of \((q, 0)\) and \((q, 1)\) through the embedding of \(f(q)\). The resulting output embedding of fastText is used as our initialization of \( W_{xv} \).

Our rationale for this embedding scheme comes from the nature of educational data. Learner question interaction data tends to cluster into components of learners correctly and incorrectly answering certain subsets of questions based on their mastery of underlying subject material [12]. By employing fastText, we create an embedding that is consistent with this feature of educational data to create a robust initialization that is more internally consistent with our data than a purely random initialization.

## 5. Empirical Results

### 5.1 Datasets Description

We consider four datasets for our experiments: ASSISTments 2009, ASSISTments 2017, Statics 2011, and a dataset from Tutor – an online learning platform. The Statics 2011 dataset is from an engineering statics course. Standard pre-processing steps common in the literature are used to clean the data. For ASSISTments 2009 dataset, we follow the pre-processing steps recommended by [23]. Duplicated records and scaffolding problems are removed. Also, since the dataset contains a few questions that are associated with multiple skills, those multiple skills were combined into a new joint skill for skill-level DKT models, along the lines of [23]. However, for qDKT, our Laplacian regularization approach provides needed flexibility when questions fall under multiple skills, doing away with the need of combining multiple skill into one joint skill. For the ASSISTments 2017 dataset, all scaffolding problems are filtered out. Relevant statistics for each dataset are given in Table 2.

### 5.2 Experimental Setup and Evaluation Metrics

Each experiment consists of comparing our proposed qDKT algorithm against the original DKT algorithm for a given dataset. To further quantify the impact of each proposed improvement to the qDKT model we will measure qDKT performance over four different variants: 1) The base qDKT without any regularization and with randomized initialization, 2) qDKT with regularization and randomized initialization, 3) qDKT without regularization but with our proposed initialization scheme and 4) qDKT with both regularization and with our proposed initialization scheme. For all the experiments and datasets, we perform 5-fold cross validation; 70% data is used for training and the rest for testing. We report the average receiver operating characteristics curve (AUC) score to compare each method. All the models are trained using the Adam optimizer [10] with dropout [20] to reduce overfitting.

### 5.3 Results and Discussion

Our results are displayed in Table 3. We see that the base qDKT model without regularization and with randomized initialization outperforms the original DKT model on three of the four datasets used. For the ASSISTments 2009 dataset, base qDKT loses by a large margin. This is due to ASSISTments 2009 dataset having a large number of questions coupled with a low number of observations per question (see Table 1).

We note that the individual addition of either the regularizer or the fastText initialization scheme greatly improves the performance of qDKT for each dataset. We finally note that the combination of both the regularizer and fastText initialization scheme enables qDKT to achieve better performance than DKT for all datasets considered.

### 5.4 Visualizations

For insights into why fastText-inspired question interaction tuple embeddings perform better as compared to those drawn from the standard normal distribution, we performed PCA

| Dataset            | #Learners | #Questions | #Skills | Total learner interaction tuples | Unique skill interaction tuples | Unique question interaction tuples |
|--------------------|-----------|------------|---------|----------------------------------|--------------------------------|----------------------------------|
| ASSISTments 2009   | 4,151     | 16,891     | 111     | 325,637                          | 29,287                         | 221                              |
| ASSISTments 2017   | 1,709     | 1,183      | 86      | 249,105                          | 2,201                          | 171                              |
| Statics 2011       | 333       | 1,223      | 85      | 189,297                          | 190                            | 2,446                            |
| Tutor              | 895       | 5,981      | 1,392   | 437,524                          | 11,622                         | 3,184                            |

Table 2: Table summarizing dataset statistics.
| Dataset         | DKT             | Base qDKT | Base qDKT w/ Laplacian regularizer | Base qDKT w/ fastText | Base qDKT w/ fastText and regularizer |
|-----------------|-----------------|-----------|-----------------------------------|------------------------|----------------------------------------|
| ASSISTments 2009| 0.740 ± 0.002   | 0.678 ± 0.004 | 0.738 ± 0.003                     | 0.740 ± 0.004         | 0.762 ± 0.005                          |
| ASSISTments 2017| 0.721 ± 0.002   | 0.742 ± 0.003 | 0.753 ± 0.005                     | 0.772 ± 0.004         | 0.770 ± 0.005                          |
| Statics 2011    | 0.770 ± 0.003   | 0.822 ± 0.003 | 0.825 ± 0.002                     | 0.832 ± 0.003         | 0.834 ± 0.002                          |
| Tutor           | 0.856 ± 0.003   | 0.875 ± 0.002 | 0.882 ± 0.001                     | 0.890 ± 0.0008        | 0.895 ± 0.001                          |

Table 3: AUC scores for each algorithm and dataset. We see that both the addition of the regularizer and the improved initialization scheme improve performance on all datasets over the original DKT model. Combining both the regularizer and our proposed initialization scheme achieves the best performance over all algorithms.

Figure 1: PCA visualization of the columns of $W_{vx}$ after training qDKT with fastText initialization (left) and DKT with randomized initialization (right). Points are colored red (circle as marker) for correct responses and blue (triangle as marker) for incorrect responses. We observe a clear separation between the correct/incorrect responses for the case of qDKT but see no such separation for DKT. This separation in the embedding space is one factor that leads to the improved predictive performance seen with the fastText initialization scheme.

Figure 2: Some of the question assessment embeddings have been labeled from Figure 1. Question ID are used as labels while color/marker indicates the assessment. In case of fastText initialization (left), one can observe a plane along the third principal component about which the assessment embeddings are approximately reflected. In case of fastText-inspired embeddings (Figure 1 left), one can observe a clear separation between correct and incorrect responses, whereas no such separation occurs for the case of
random initialization.

In Figure 2 we label some of the points in Figure 1 to further illustrate the symmetry we gain through the fastText initialization. The label has the question ID information, while the color/marker indicates the assessment. One can observe that in the case of fastText-inspired initialization (left image), the separating plane (approximately along the third principal component) serves as a reflector for question assessment embeddings.

Symmetry about some plane is expected at the beginning of the training since we initialize the embedding of \((q, 0)\) with the sum of embeddings of \(\{f(q), '0' f(q) + '0'\}\) and likewise for \((q, 1)\) for any question \(q\). However, the preservation of the reflecting plane post-training suggests that such a geometry is useful and maintained by the model throughout training.

6. CONCLUSION
We have proposed qDKT, a novel model for knowledge tracing for educational data. Our method improves on prior art by predicting student performance at the question-level, rather than at the skill level. We have further proposed novel regularization and initialization schemes that greatly improve the performance of our method across several real-world datasets when compared with the traditional knowledge tracing methods. We propose that qDKT can provide a simple, yet tough-to-beat baseline, for new question-centric KT models to come.

7. REFERENCES
[1] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.
[2] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
[3] 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.
[4] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.
[5] S. E. Embretson and S. P. Reise. *Item response theory*. Psychology Press, 2013.
[6] Y. Goldberg and O. Levy. word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.
[7] D. J. Hand. Statistical analysis of network data: Methods and models by eric d. kolaczyk. *International Statistical Review*, 78(1):135–135, 2010.
[8] S. Hochreiter and J. Schmidhuber. Long short-term memory, *Neural computation*, 9(8):1735–1780, 1997.
[9] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.
[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[11] A. S. Lan, C. Studer, and R. G. Baraniuk. Time-varying learning and content analytics via sparse factor analysis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 452–461. ACM, 2014.
[12] A. S. Lan, A. E. Waters, C. Studer, and R. G. Baraniuk. Sparse factor analysis for learning and content analytics. *The Journal of Machine Learning Research*, 15(1):1959–2008, 2014.
[13] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.
[14] J. H. Martin and D. Jurášky. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall Upper Saddle River, 2009.
[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
[16] Z. Pardos and N. Heffernan. Navigating the parameter space of bayesian knowledge tracing models: Visualizations of the convergence of the expectation maximization algorithm. In *Educational Data Mining 2010*, 2010.
[17] R. Pelánek. Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, 27(3-5):313–350, 2017.
[18] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
[19] 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.
[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
[21] K. Vanlehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006.
[22] T. Wang, F. Ma, and J. Gao. Deep hierarchical knowledge tracing. In *EDM*, 2019.
[23] X. Xiong, S. Zhao, E. G. Van Inwegen, and J. E. Beck. Going deeper with deep knowledge tracing. *International Educational Data Mining Society*, 2016.
[24] 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.