Dynamic Key-Value Memory Networks for Knowledge Tracing

Jiani Zhang[1,2], Xingjian Shi[3], Irwin King[1,2], Dit-Yan Yeung[3]

[1]Shenzhen Key Laboratory of Rich Media Big Data Analytics and Application,
Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
[2]Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{jnzhang, king}@cse.cuhk.edu.hk

[3]Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Kowloon, Hong Kong
{xshiab, dyyeung}@cse.ust.hk

ABSTRACT

Knowledge Tracing (KT) is a task of tracing evolving knowledge state of students with respect to one or more concepts as they engage in a sequence of learning activities. One important purpose of KT is to personalize the practice sequence to help students learn knowledge concepts efficiently. However, existing methods such as Bayesian Knowledge Tracing and Deep Knowledge Tracing either model knowledge state for each predefined concept separately or fail to pinpoint exactly which concepts a student is good at or unfamiliar with. To solve these problems, this work introduces a new model called Dynamic Key-Value Memory Networks (DKVMN) that can exploit the relationships between underlying concepts and directly output a student's mastery level of each concept. Unlike standard memory-augmented neural networks that facilitate a single memory matrix or two static memory matrices, our model has one static matrix called *key*, which stores the knowledge concepts and the other dynamic matrix called *value*, which stores and updates the mastery levels of corresponding concepts. Experiments show that our model consistently outperforms the state-of-the-art model in a range of KT datasets. Moreover, the DKVMN model can automatically discover underlying concepts of exercises typically performed by human annotations and depict the changing knowledge state of a student.

Keywords

Massive Open Online Courses; Knowledge Tracing; Concept Discovery; Deep Learning; Dynamic Key-Value Memory Networks

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052580.

1. INTRODUCTION

With the advent of massive open online courses and intelligent tutoring systems in the web, students can get appropriate guidance and acquire relevant knowledge in the process of solving exercises. When an exercise is posted, a student must apply one or more concepts to solve the exercise. For example, when a student attempts to solve the exercise "1+2", then he or she should apply the concept of "integer addition"; when a student attempts to solve "1+2+3.4", then he or she should apply the concepts of "integer addition" and "decimal addition". The probability that a student can answer the exercise correctly is based on the student's *knowledge state*, which stands for the depth and robustness of the underlying concepts the student has mastered.

The goal of *knowledge tracing* (KT) is to trace the knowledge state of students based on their past exercise performance. KT is an essential task in online learning platforms. Tutors can give proper hints and tailor the sequence of practice exercises based on the personal strengths and weaknesses of students. Students can be made aware of their learning progress and may devote more energy to less-familiar concepts to learn more efficiently.

Although effectively modeling the knowledge of students has high educational impact, using numerical simulations to represent the human learning process is inherently difficult [22]. Usually, KT is formulated as a supervised sequence learning problem: given a student's past exercise interactions $\mathcal{X} = \{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_{t-1}\}$, predict the probability that the student will answer a new exercise correctly, i.e., $p(r_t = 1|q_t, \mathcal{X})$. Input $\mathbf{x}_t = (q_t, r_t)$ is a tuple containing the exercise q_t, which student attempts at the timestamp t, and the correctness of the student's answer r_t. We model $\mathcal{X}$ as observed variables and a student's knowledge state $\mathcal{S} = \{\mathbf{s}_1,\mathbf{s}_2,...,\mathbf{s}_{t-1}\}$ of N underlying concepts $\mathcal{C} = \{c^1, c^2, ..., c^N\}$ as a hidden process.

Existing methods such as Bayesian Knowledge Tracing (BKT) [3] and Deep Knowledge Tracing (DKT) [22] model the knowledge state of students either in a concept specific manner or in one summarized hidden vector, as shown in Figure 1. In BKT, a student's *knowledge state* $\mathbf{s}_t$ is analyzed into different *concept state*s $\{\mathbf{s}_t^i\}$ and BKT models each con-

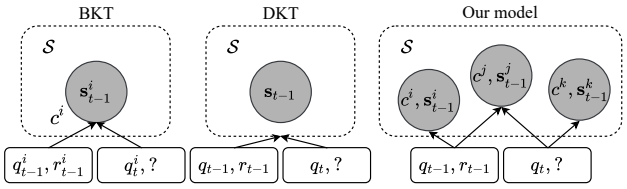


Figure 1: Model differences among BKT, DKT, and our model. BKT is concept specific. DKT uses a summarized hidden vector to model the knowledge state. Our model maintains the *concept state* for each concept simultaneously and all concept states constitute the *knowledge state* of a student.

cept state separately. BKT assumes the concept state as a binary latent variable, *known* and *unknown*, and uses a Hidden Markov model to update the posterior distribution of the binary concept state. Therefore, BKT cannot capture the relationship between different concepts. Moreover, to keep the Bayesian inference tractable, BKT uses discrete random variables and simple transition models to describe the evolvement of each concept state. As a result, although BKT can output the student's mastery level of some predefined concepts, it lacks the ability to extract undefined concepts and model complex concept state transitions.

Besides solving the problem from the Bayesian perspective, a deep learning method named DKT [22] exploits a variant of recurrent neural networks (RNNs) called long short-term memory (LSTM) [9]. LSTM assumes a high-dimensional and continuous representation of the underlying knowledge state $\mathcal{S}$. The nonlinear input-to-state and state-to-state transitions of DKT have stronger representational power than those of BKT. No human-labeled annotation is required. However, DKT summarizes a student's knowledge state of all concepts in one hidden state, which makes it difficult to trace how much a student has mastered a certain concept and pinpoint which concepts a student is good at or unfamiliar with [11, 31].

The present work introduces a new model called Dynamic Key-Value Memory Networks (DKVMN) that combines the best of two worlds: the ability to exploit the relationship between concepts and the ability to trace each concept state. Our DKVMN model can automatically learn the correlation between input exercises and underlying concepts and maintain a concept state for each concept. At each timestamp, only related concept states will be updated. For instance, in Figure 1, when a new exercise q_t comes, the model finds that q_t requires the application of concept c^j and c^k. Then we read the corresponding concept states $\mathbf{s}^j_{t-1}$ and $\mathbf{s}^k_{t-1}$ to predict whether the student will answer the exercise correctly. After the student completes the exercise, our model will update these two concept states. All concept states constitute the *knowledge state* $\mathcal{S}$ of a student.

In addition, unlike standard memory-augmented neural networks (MANNs) that facilitate a single memory matrix [6, 24, 30] or a variation with two static memory matrices [17, 27], our model has one static matrix called *key*, which stores the concept representations and the other dynamic matrix called *value*, which stores and updates the student's understanding (concept state) of each concept. The terms *static* and *dynamic* matrices are respectively analogous to *immutable* and *mutable* objects as keys and values in the dictionary data structure (e.g., Python's dictionary). Meanwhile, our training process is analogous to object creation. After the keys are created, they will be fixed (i.e., immutable) during testing.

The network with two static memory matrices is not suitable for solving the KT task because learning is not a static process. Learning builds upon and is shaped by previous knowledge in human memory [8]. The model with a single dynamic matrix maps the exercise with the correct answer and the exercise with the incorrect answer to different concept states, which does not match our cognition. Experiments show that our DKVMN model outperforms the MANN model with a single memory matrix and the state-of-the-art model.

Our main contributions are summarized as follows:

1. The utility of MANNs is exploited to better simulate the learning process of students.
2. A novel DKVMN model with one static *key* matrix and one dynamic *value* matrix is proposed.
3. Our model can automatically discover concepts, a task that is typically performed by human experts, and depict the evolving knowledge state of students.
4. Our end-to-end trainable model consistently outperforms BKT and DKT on one synthetic and three real-world datasets respectively.

2. RELATED WORKS

2.1 Knowledge Tracing

The KT task evaluates the knowledge state of a student based simply on the correctness or incorrectness r_t of a student's answers in the process of solving exercises q_t. In this study q_t is an exercise tag and $r_t \in \{0, 1\}$ is a binary response (1 is correct and 0 is incorrect). No secondary data are incorporated [11].

BKT [3] is a highly constrained and structured model [11] because it models concept-specific performance, i.e., an individual instantiation of BKT is made for each concept, and BKT assumes knowledge state as a binary variable. Many following variations were raised by integrating personalization study [19, 33], exercise diversity [20], and other information [4, 23] into the Bayesian framework.

DKT [22] exploits the utility of LSTM [9] to break the restriction of skill separation and binary state assumption. LSTM uses hidden states as a kind of summary of the past sequence of inputs, and the same parameters are shared over different time steps. Experiments in [22] showed that DKT outperforms previous Bayesian models by a large margin in terms of prediction accuracy. This study was the first attempt to integrate deep learning models [14, 25], which have achieved significant success in other areas, including computer vision [13] and natural language processing [16] into KT.

2.2 Memory-Augmented Neural Networks

Inspired by computer architecture, a particular neural network module called external memory was proposed to enhance the ability of a network to capture long-term dependencies and solve algorithmic problems [7]. MANN have led the progress in various areas, such as question answering [30,

27, 1, 17], natural language transduction [8], algorithm inference [6, 10], and one-shot learning [24, 28].

The typical external memory module contains two parts, a memory matrix that stores the information and a controller that communicates with the environment and reads or writes to the memory. The reading and writing operations are achieved through additional attention mechanisms. Most previous works [6, 27, 24] use a similar way to compute the read weight. For an input $\mathbf{k}_t$, a cosine similarity or an inner product $K[\mathbf{k}_t, \mathbf{M}_t(i)]$ of the input and each memory slot $\mathbf{M}_t(i)$ is computed, which then goes through a softmax with a positive key strength β_t to obtain a read weight $\mathbf{w}_t^r$: $w_t^r(i) = \text{Softmax}(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(i)])$, where $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$. For the write process, an attention mechanism of focusing both by content and by location is proposed in [6] to facilitate all the locations of the memory. In addition, a pure content-based memory writer named least recently used access (LRUA) module is raised in [24] to write the key either to the least recently used memory location or to the most recently used memory location.

Owing to the recurrence introduced in the read and write operations, MANN is a special kind of RNN as well. However, MANN is different from conventional RNNs like the LSTM used in DKT in three aspects. First, traditional RNN models use a single hidden state vector to encode the temporal information, whereas, MANN uses an external memory matrix that can increase storage capacity [30]. Second, the state-to-state transition of traditional RNNs is unstructured and global, whereas MANN uses read and write operations to encourage local state transitions [6]. Third, the number of parameters in traditional RNNs is tied to the size of hidden states [24]. For MANN, increasing the number of memory slots will not increase the number of parameters, an outcome that is more computationally efficient.

3. MODEL

In this section, we first introduce the way to exploit the existing MANN model to solve the KT problem. We then show the deficiencies of MANN and describe our DKVMN model. In our description below, we denote vectors with bold small letters and matrices with bold capital letters.

3.1 Memory-Augmented Neural Network for Knowledge Tracing

To solve the KT problem, the external memory matrix of MANN is treated as the knowledge state of a student. The overall structure of the model is shown in Figure 2a. The memory, denoted as $\mathbf{M}_t$, is an $N \times M$ matrix, where N is the number of memory locations, and M is the vector size at each location. At each timestamp t, the input for MANN is a joint embedding $\mathbf{v}_t$ of (q_t, r_t), where each q_t comes from a set of Q distinct exercise tags and r_t is a binary value indicating whether the student answered the exercise correctly. The embedding vector $\mathbf{v}_t$ is used to compute the read weight $\mathbf{w}_t^r$ and the write weight $\mathbf{w}_t^w$.

In our implementation, we choose the cosine similarity attention mechanism to compute $\mathbf{w}_t^r$ and the LRUA mechanism [24] to compute $\mathbf{w}_t^w$. Details of these two attention mechanisms are shown in the appendix. The intuition of the MANN is that when a student answers the exercise that has been stored in the memory with the same response, $\mathbf{v}_t$ will be written to the previously used memory locations and when a new exercise arrives or the student gets a different response, $\mathbf{v}_t$ will be written to the least recently used memory locations.

In the read process, the read content $\mathbf{r}_t$ is obtained by the weighted sum of all memory slots with the read weight $\mathbf{w}_t^r$:

$$\mathbf{r}_t = \sum_{i=1}^{N} w_t^r(i)\mathbf{M}_t(i). \tag{1}$$

The output $\mathbf{p}_t \in \mathbb{R}^Q$, which is computed from $\mathbf{r}_t$, indicates the probability that the student can answer each exercise correctly in the next timestamp.

In the write process, we first erase unnecessary contents in the memory using the erase signal $\mathbf{e}_t$ and the write weight $\mathbf{w}_t^r$, and then add $\mathbf{v}_t$ into the memory using the add signal $\mathbf{a}_t$ [6]. For further details, see Section 3.2.3.

MANN uses N memory slots to encode the knowledge state of a student and has a larger capacity than LSTM, which only encodes knowledge state in a single hidden vector.

3.2 Dynamic Key-Value Memory Networks

Despite being more powerful than LSTM in storing the past performance of students, MANN still has deficiencies when applied to the KT task. In MANN, the content we read lies in the same space as the content we write. However, for tasks like KT, the input and the prediction, which are the exercises the student receives and the correctness of the student's answer have different types. Therefore, the way to embed the exercise and the response jointly as the attention key does not make sense. Furthermore, MANN cannot explicitly model the underlying concepts for input exercises. Knowledge state of a particular concept is dispersed and cannot be traced.

To solve these problems, our DKVMN model uses key-value pairs rather than a single matrix for the memory structure. Instead of attending, reading, and writing to the same memory matrix in MANN, our DKVMN model attends input to the *key* component, which is immutable, and reads and writes to the corresponding *value* component.

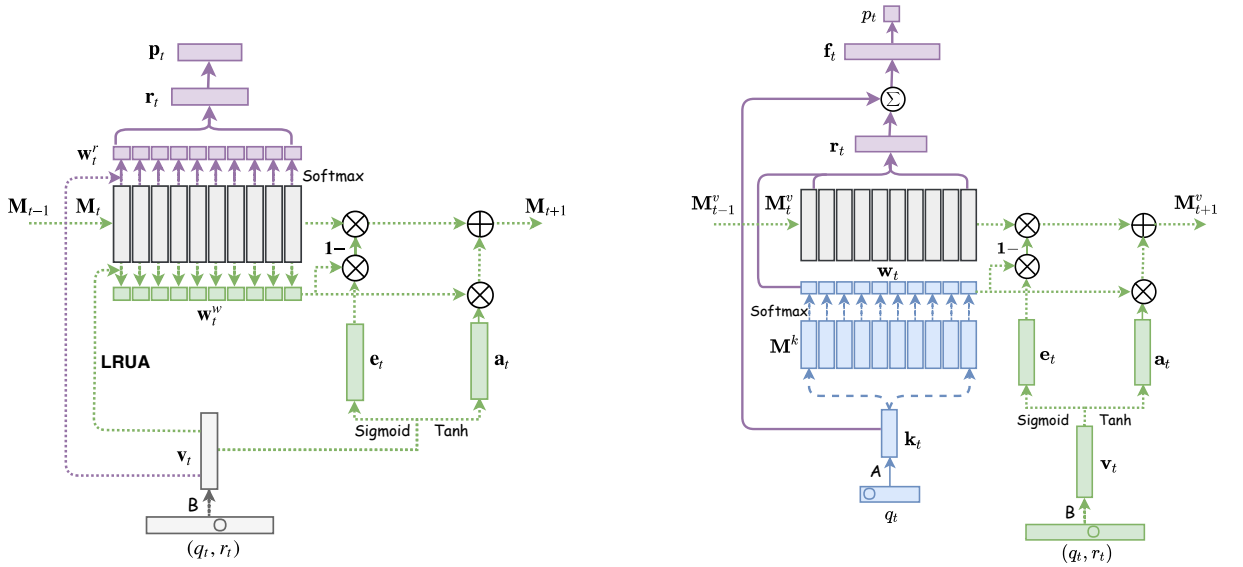
Unlike MANN, at each timestamp, DKVMN takes a discrete exercise tag q_t, outputs the probability of response $p(r_t|q_t)$, and then updates the memory with exercise and response tuple (q_t, r_t). Here, q_t also comes from a set with Q distinct exercise tags and r_t is a binary value. We further assume there are N latent concepts $\{c^1, c^2, ..., c^N\}$ underlying the exercises. These concepts are stored in the *key* matrix $\mathbf{M}^k$ (of size $N \times d_k$) and the student's mastery levels of each concept, i.e., concept states $\{\mathbf{s}_t^1, \mathbf{s}_t^2, ..., \mathbf{s}_t^N\}$ are stored in the *value* matrix $\mathbf{M}_t^v$ (of size $N \times d_v$), which changes over time.

DKVMN traces the knowledge of a student by reading and writing to the *value* matrix using the correlation weight computed from the input exercise and the *key* matrix. The model details are elaborated in the following sections.

3.2.1 Correlation Weight

The input exercise q_t is first multiplied by an embedding matrix $\mathbf{A}$ (of size $Q \times d_k$) to get a continuous embedding vector $\mathbf{k}_t$ of dimension d_k. The correlation weight is further computed by taking the softmax activation of the inner product between $\mathbf{k}_t$ and each *key* slot $\mathbf{M}^k(i)$:

$$w_t(i) = \text{Softmax}(\mathbf{k}_t^T \mathbf{M}^k(i)), \tag{2}$$



(a) Architecture for Memory-Augmented Neural Networks.

(b) Architecture for Dynamic Key-Value Memory Networks.

Figure 2: In both architecture, the model is only drawn at the timestamp t, where the purple components describe the read process and the green components describe the write process. The blue components in the DKVMN model denote the attention process to compute the corresponding weight. (Best viewed in color.)

where $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$ and is differentiable. Both the read and write processes will use this weight vector $\mathbf{w}_t$, which represents the correlation between exercise and each latent concept.

3.2.2 Read process

When an exercise q_t comes, the read content $\mathbf{r}_t$ is retrieved by the weighted sum of all memory slots in the *value* matrix using $\mathbf{w}_t$:

$$\mathbf{r}_t = \sum_{i=1}^{N} w_t(i)\mathbf{M}_t^v(i). \tag{3}$$

The calculated read content $\mathbf{r}_t$ is treated as a summary of the student's mastery level of this exercise. Given that each exercise has its own difficulty, we concatenate the read content $\mathbf{r}_t$ and the input exercise embedding $\mathbf{k}_t$ and then pass it through a fully connected layer with a Tanh activation to get a summary vector $\mathbf{f}_t$, which contains both the student's mastery level and the prior difficulty of the exercise:

$$\mathbf{f}_t = \text{Tanh}(\mathbf{W}_1^T[\mathbf{r}_t, \mathbf{k}_t] + \mathbf{b}_1), \tag{4}$$

where $\text{Tanh}(z_i) = (e^{z_i} - e^{-z_i})/(e^{z_i} + e^{-z_i})$.

Finally, $\mathbf{f}_t$ is passed through another fully connected layer with a Sigmoid activation to predict the performance of the student:

$$p_t = \text{Sigmoid}(\mathbf{W}_2^T\mathbf{f}_t + \mathbf{b}_2), \tag{5}$$

where $\text{Sigmoid}(z_i) = 1/(1 + e^{-z_i})$, and p_t is a scalar that represents the probability of answering q_t correctly.

3.2.3 Write process

After the student answers the question q_t, the model will update the *value* matrix according to the correctness of the student's answer. A joint embedding of (q_t, r_t) will be written to the *value* part of the memory with the same correlation weight $\mathbf{w}_t$ used in the read process.

The tuple (q_t, r_t) is embedded with an embedding matrix $\mathbf{B}$ of size $2Q \times d_v$ to obtain the *knowledge growth* $\mathbf{v}_t$ of the student after working on this exercise. When writing the student's *knowledge growth* into the *value* component, the memory is erased first before new information is added [6], a step inspired by the input and forget gates in LSTMs.

Given a write weight (which is the correlation weight $\mathbf{w}_t$ in our model), an *erase vector* $\mathbf{e}_t$ is computed from $\mathbf{v}_t$:

$$\mathbf{e}_t = \text{Sigmoid}(\mathbf{E}^T\mathbf{v}_t + \mathbf{b}_e), \tag{6}$$

where the transformation matrix $\mathbf{E}$ is of shape $d_v \times d_v$, $\mathbf{e}_t$ is a column vector with d_v elements that all lie in the range $(0, 1)$. The memory vectors of *value* component $\mathbf{M}_{t-1}^v(i)$ from the previous timestamp are modified as follows:

$$\tilde{\mathbf{M}}_t^v(i) = \mathbf{M}_{t-1}^v(i)[\mathbf{1} - w_t(i)\mathbf{e}_t], \tag{7}$$

where $\mathbf{1}$ is a row-vector of all 1-s. Therefore, the elements of a memory location are reset to zero only if both the weight at the location and the erase element are one. The memory vector is left unchanged if either the weight or the erase signal is zero.

After erasing, a length d_v *add vector* $\mathbf{a}_t$ is used to update each memory slot:

$$\mathbf{a}_t = \text{Tanh}(\mathbf{D}^T\mathbf{v}_t + \mathbf{b}_\mathbf{a})^T, \tag{8}$$

where the transformation matrix $\mathbf{D}$ is of shape $d_v \times d_v$ and $\mathbf{a}_t$ is a row vector. The *value* memory is updated at each time t by

$$\mathbf{M}_t^v(i) = \tilde{\mathbf{M}}_{t-1}^v(i) + w_t(i)\mathbf{a}_t. \tag{9}$$

This *erase*-followed-by-*add* mechanism allows forgetting and strengthening concept states in the learning process of a student.

Table 1: Test AUC results for all datasets. BKT is the standard BKT. BKT+ is the best-reported result with BKT variations. DKT is the result using LSTM. MANN is the baseline using a single memory matrix. DKVMN is our model.

Datasets	Overview			Test AUC (%)				
	Students	Exercise Tags	Records	BKT	BKT+	DKT	MANN	DKVMN
Synthetic-5	4,000	50	200,000	62	80	80.3±0.1	81.0±0.1	**82.7±0.1**
ASSISTments2009	4,151	110	325,637	63	-	80.5±0.2	79.7±0.1	**81.6±0.1**
ASSISTments2015	19,840	100	683,801	64	-	72.5±0.1	72.3±0.2	**72.7±0.1**
Statics2011	333	1,223	189,297	73	75	80.2±0.2	77.6±0.1	**82.8±0.1**

3.2.4 Training

The overall model architecture is shown in Figure 2b. During training, both embedding matrices $\mathbf{A}$ and $\mathbf{B}$, as well as other parameters and the initial value of $\mathbf{M}^k$ and $\mathbf{M}^v$ are jointly learned by minimizing a standard cross entropy loss between p_t and the true label r_t.

$$\mathcal{L} = -\sum_t (r_t \log p_t + (1 - r_t)\log(1 - p_t)). \quad (10)$$

Our DKVMN model is fully differentiable and can be trained efficiently with stochastic gradient descent (see Section 4.2 for more details).

4. EXPERIMENTS

The prediction accuracy is first evaluated by comparing our DKVMN model with other methods on four datasets, namely one synthetic dataset and three real-world datasets, collected from online learning platforms. Then, comparative experiments of different dimensions of states are performed on DKVMN and DKT for further model exploration. Finally, the ability of our model is verified to discover concepts automatically and depict the knowledge state of students.

The experiment results lead to the following findings:

- DKVMN outperforms the standard MANN and the state-of-the-art method on four datasets.
- DKVMN can produce better results with fewer parameters than DKT.
- DKVMN does not suffer from overfitting, which is a big issue for DKT.
- DKVMN can discover underlying concepts for input exercises precisely.
- DKVMN can depict students' concept states of distinct concepts over time.

We implement the models using MXNet [2] on a computer with a single NVIDIA K40 GPU.

4.1 Datasets

To evaluate performance, we test KT models on four datasets: Synthetic-5, ASSISTments2009, ASSISTments2015, and Statics2011.

Synthetic-5: This dataset[1] simulates 2000 virtual students answering 50 exercises in both the training and testing dataset. Each exercise is drawn from one of five hidden concepts and has different levels of difficulty. We have no access to the underlying concept labels in the training process and simply use them as the ground truth to evaluate the discovered concept results using our DKVMN model.

ASSISTments2009: This dataset [5] is gathered from the ASSISTments online tutoring platform. Owing to duplicated record issues [32], an updated version is released and all previous results on the old dataset are no longer reliable. The experiments in our paper are conducted using the updated "skill-builder" dataset[2]. Records without skill names are discarded in the preprocessing. Thus, the number of records in our experiments is smaller than that in [32]. A total of 4,151 students answer 325,637 exercises along with 110 distinct exercise tags.

ASSISTments2015: ASSISTments2015[3] only contains student responses on 100 skills. After preprocessing (removing the value of *correct* $\notin \{0, 1\}$), 683,801 effective records from 19,840 students are remained in this dataset. Each problem set in this dataset has one associated skill. Although this dataset has the largest number of records, the average records for each student are the lowest.

Statics2011: Statics[4] is from a college-level engineering statics course with 189,297 trials, 333 students and 1,223 exercises tags [26, 12]. In our experiments, a concatenation of problem name and step name is used as an exercise tag; thus it has the maximum number of exercise tags and the maximum number of average records per student.

The complete statistical information for all datasets can be found in Table 1.

4.2 Implementation Details

The input exercise data are presented to neural networks using "one-hot" input vectors. Specifically, if Q different exercises exist in total, then the exercise tag q_t for the *key* memory part is a length Q vector whose entries are all zero except for the q_t^{th} entry, which is one. Similarly, the combined input $x_t = (q_t, r_t)$ for the *value* matrix component is a length $2Q$ vector, where entry $x_t = q_t + r_t * Q$ is one.

We learn the initial value of both the *key* and the *value* matrix in the training process. Each slot of the *key* memory is the concept embedding and is fixed in the testing process. Meanwhile, the initial value of the *value* memory is the initial state of each concept, which represents the initial difficulty of each concept.

[1]Synthetic-5:https://github.com/chrispiech/DeepKnowledgeTracing/tree/master/data/synthetic

[2]ASSISTments2009:https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010

[3]ASSISTments2015:https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data

[4]Statics2011:https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507

Table 2: Comparison of DKVMN with DKT on four datasets with different numbers of state dimensions and memory size N. "s. dim", "m. size" and "p. num" represent the state dimension, memory size (i.e., the number of concepts N), and the number of parameters, respectively. We choose the state dimensions of 10, 50, 100, and 200 for both DKT and DKVMN. Then for DKVMN, we change memory size for 1, 2, 5, 10, 20, 50, and 100 for each state dimension and report the best test AUC with the corresponding memory size. We likewise compare the number of parameters for both models.

Model	Synthetic-5				ASSISTments2009				ASSISTments2015				Statics2011			
	s. dim	m. size	test auc	p. num	s. dim	m. size	test auc	p. num	s. dim	m. size	test auc	p. num	s. dim	m. size	test auc	p. num
DKT	10	-	80.06	2.4K	10	-	80.38	4.3K	10	-	72.40	4.0K	10	-	78.12	39K
	50	-	80.22	28K	50	-	**80.53**	37K	50	-	**72.52**	36K	50	-	79.86	205K
	100	-	**80.34**	96K	100	-	80.51	114K	100	-	72.49	111K	100	-	80.16	449K
	200	-	80.32	352K	200	-	80.43	388K	200	-	72.45	382K	200	-	**80.20**	1.0M
DKVMN	10	50	82.00	12K	10	10	81.47	7k	10	20	**72.68**	14K	10	10	82.72	92K
	50	50	82.66	25K	50	20	**81.57**	31k	50	10	72.66	29K	50	10	**82.84**	197K
	100	50	**82.73**	50K	100	10	81.42	68k	100	50	72.64	63K	100	10	82.71	338K
	200	50	82.71	130K	200	20	81.37	177k	200	50	72.53	153K	200	10	82.70	649K

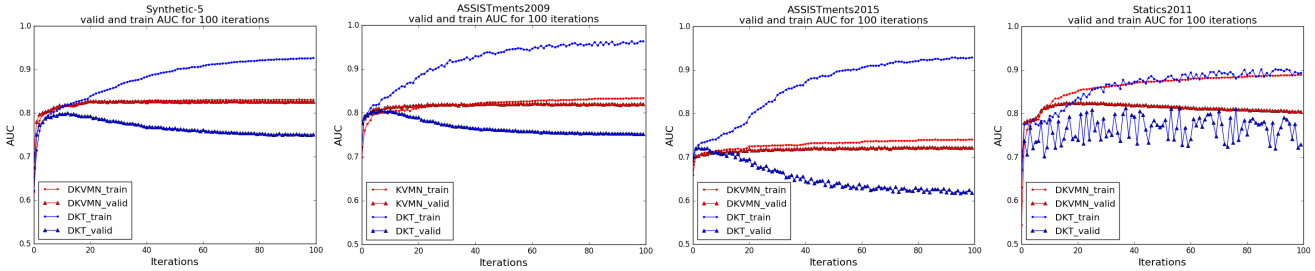


Figure 3: Validation AUC and training AUC of DKVMN and DKT on all datasets. The blue line represents the DKT model, and the red line represents our DKVMN model. The dotted line represents the training AUC and the line with upper triangles represents the validation AUC. (Best viewed in color.)

Of all the datasets, 30% of the sequences were held out as a testing set, except for the synthetic dataset where training and testing datasets had the same size. A total 20% of the training set was split to form a validation set, which was used to select the optimal model architecture and hyperparameters and perform early stopping [18].

The parameters were initialized randomly from a Gaussian distribution with zero mean and standard deviation σ. The initial learning rate was case by case because the number of students, exercise tags, and total answers per dataset varied, but the learning rate γ annealed every 20 epochs by $\gamma/1.5$ until the 100th epoch was reached.

We used LSTM for DKT in our implementation. The standard MANN was implemented using the cosine similarity reading attention mechanism and the LRUA writing attention mechanism. Stochastic gradient descent with momentum and norm clipping [21] were used to train DKT, MANN, and our DKVMN in all the experiments. We consistently set the momentum to be 0.9 and the norm clipping threshold to be 50.0. Given that input sequences are of different lengths, all sequences were set to be a length of 200 (for *synthetic* with a length of 50) and a null symbol was used to pad short sequence to a fixed size of 200.

In all cases, hyperparameters were tuned using the five-fold cross validation. The test area under the curve (AUC) was computed using the model with the highest validation AUC among the 100 epochs. We repeated each training five times with different initializations σ and reported the average test AUC along with the standard deviation.

4.3 Student Performance Prediction

The AUC is measured to evaluate the prediction accuracy on each dataset. An AUC of 50% represents the score achievable by random guessing. A high AUC score accounts for a high prediction performance. Results of the test AUC on all datasets are shown in Table 1.

We compare the DKVMN model with the MANN baseline, the state-of-the-art DKT, the standard BKT model, and, when possible, optimal variations of BKT (BKT+). An interesting observation is that our implemented LSTM achieves better AUC than those in the original papers [22, 11, 32]. The reason may be that our implementations use norm clipping and early stopping, both of which improve the overfitting problem of LSTM. The results of BKT are directly obtained from recent works [11, 32].

On the Synthetic-5 dataset, the DKVMN model achieves the average test AUC of 82.7%. In our simulation, each exercise is treated as having a distinct skill label. MANN produces an average AUC of 81.0%. DKT produces an AUC value of 80.3%, which is better than the 75% reported in the original paper [22, 11]. BKT and its variant model achieve the AUC of 62% and 80% respectively [11]. The prediction results of DKVMN from the ASSISTments2009 achieve improvement over MANN, DKT, and BKT with 81.6% over 79.7%, 80.5%, and 63% respectively [32]. As this dataset is preprocessed differently from that in [32], their results are not comparable. On the ASSISTments2015 dataset, the test AUC of DKVMN is 72.7%, which is better than 72.3% for MANN, 72.5% for DKT (originally 70% in [32]), and 64% for

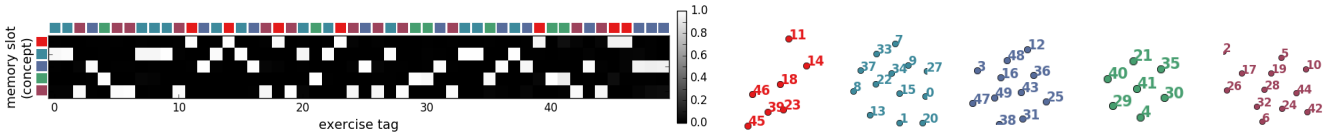


Figure 4: Concept discovery results on the synthetic-5 dataset when the memory size N is set to be 5. In the left heat map, the x-axis represents each exercise and the y-axis represents the correlation weight between the exercise and five latent concepts generated from our DKVMN model. The ground-true concept is labeled on the top of each exercise. In the right exercise clustering graph, each node number represents an exercise. Exercises from the same ground-truth concept are clustered together. (Best viewed in color.)

17 Scatter Plot	1 Area Trapezoid	3 Probability of Two Distinct Events	24 Addition and Subtraction Fractions
21 Multiplication and Division Integers	11 Histogram as Table or Graph	4 Table	28 Calculations with Similar Figures
23 Absolute Value	47 Percent Discount	5 Median	53 Interior Angles Figures with More than 3 Sides
25 Subtraction Whole Numbers	54 Interior Angles Triangle	6 Stem and Leaf Plot	58 Solving for a variable
30 Ordering Fractions	64 Surface Area Rectangular Prism	7 Mode	77 Number Line
33 Ordering Integers	67 Percents	8 Mean	79 Solving Inequalities
37 Ordering Positive Decimals	68 Area Circle	9 Range	84 Effect of Changing Dimensions of a Shape
42 Pattern Finding	69 Least Common Multiple	10 Venn Diagram	105 Finding Slope from Ordered Pairs
43 Write Linear Equation from Situation	73 Prime Number	12 Circle Graph	2 Area Irregular Figure
44 Square Root	76 Computation with Real Numbers	13 Equivalent Fractions	48 Nets of 3D Figures
46 Algebraic Solving	78 Rate	14 Proportion	65 Scientific Notation
52 Congruence	85 Surface Area Cylinder	15 Fraction Of	83 Area Parallelogram
61 Estimation	88 Solving Systems of Linear Equations	16 Probability of a Single Event	86 Volume Cylinder
62 Ordering Real Numbers	89 Solving Systems of Linear Equations by Graphing	22 Addition Whole Numbers	92 Rotations
74 Multiplication and Division Positive Decimals	96 Interpreting Coordinate Graphs	29 Counting Methods	94 Translations
81 Area Rectangle	108 Recognize Quadratic Pattern	32 Box and Whisker	101 Angles - Obtuse, Acute, and Right
97 Choose an Equation from Given Information	110 Quadratic Formula to Solve Quadratic Equation	35 Percent Of	103 Recognize Linear Pattern
98 Intercept	18 Addition and Subtraction Positive Decimals	39 Volume Rectangular Prism	31 Circumference
99 Linear Equations	26 Equation Solving Two or Fewer Steps	41 Finding Percents	70 Equation Solving More Than Two Steps
100 Slope	36 Unit Rate	51 Understanding Concept Of Probabilities	75 Volume Sphere
102 Distributive Property	40 Order of Operations All	19 Multiplication Fractions	82 Area Triangle
109 Finding Slope From Equation	49 Complementary and Supplementary Angles	27 Order of Operations +,-,/,* () positive reals	91 Polynomial Factors
20 Addition and Subtraction Integers	59 Exponents	38 Rounding	104 Simplifying Expressions positive exponents
34 Conversion of Fraction Decimals Percents	80 Unit Conversion Within a System	60 Division Fractions	45 Algebraic Simplification
50 Pythagorean Theorem	87 Greatest Common Factor	63 Scale Factor	55 Divisibility Rules
57 Perimeter of a Polygon	90 Multiplication Whole Numbers	106 Finding Slope From Situation	56 Reading a Ruler or Scale
71 Angles on Parallel Lines Cut by a Transversal	93 Reflection	107 Parts of a Polyomial, Coefficient, Exponent	66 Write Linear Equation from Graph
95 Midpoint			72 Write Linear Equation from Ordered Pairs

Figure 5: Concept discovery results on the ASSSISTments2009 dataset. 110 exercises are clustered into ten concepts. Exercises under the same concept are labeled in the same color in the left picture and also are put in the same block in the right table. (Best viewed in color.)

classic BKT [32]. With regard to Statics2011, which has the maximum number of exercise tags and the minimum number of answers, classical BKT gains the AUC of 73% and BKT cooperating with forgetting, skill discovery, and latent abilities obtains an AUC of 75% [11]. Our implemented DKT leads to an AUC of 80.2%, which is better than the 76% from [11]. MANN only produces the average AUC of 77.6%. However, our DKVMN model achieves an AUC of 82.8%, outperforming all previous models.

In summary, DKVMN performs better than other methods across all the datasets, particularly on the Statics2011 dataset whose number of distinct exercises is large. This result demonstrates that our DKVMN can model student's knowledge well when the number of exercises is very large.

DKVMN can achieve better prediction accuracy over student exercise performance and also requires considerably fewer parameters than the DKT model because of its large external memory capacity. Table 2 compares the DKVMN model with the DKT model using LSTM by traversing different hyperparameters. The table reveals that DKVMN with low state dimensions can achieve better prediction accuracy than DKT with high state dimensions. For instance, on the Statics2011 dataset, DKT reaches the maximum test AUC of 80.20% when the dimension of states equals 200 using 1 million parameters. Meanwhile, DKVMN can achieve the test AUC of 82.84% only with 50 state dimensions using 197 thousand parameters.

Moreover, the DKT model suffers severe overfitting, whereas our DKVMN model does not confront such a problem. As indicated in Figure 3, no huge gap exists between the training AUC and the validation AUC of DKVMN, and the validation AUC of DKVMN increases smoothly. However, as the epoch proceeds, the training AUC of DKT increases continuously, and the validation AUC of DKT only increases in the first several epochs and begins to decrease.

4.4 Concept Discovery

Our DKVMN model has the power to discover underlying patterns or concepts for exercises using the correlation weight $\mathbf{w}$, which is traditionally annotated by experts. The correlation weight between the exercise and the concept implies the strength of their inner relationship. Compared with the conditional influence approach in [22] which computes the dependencies between exercises and then defines a threshold to cluster the exercises, our model directly assigns exercises to concepts. No predefined threshold is required. As a result, our model can discover the concepts of exercises in an end-to-end manner.

Each exercise is usually associated with a single concept. In this case, we assign the exercise to the concept with the largest correlation weight value. From the experiments, we find that our model can intelligently learn sparse weight among concepts, and the discovered concepts reveal a compelling result.

On the Synthetic-5 dataset, each exercise is drawn from a concept c^k, where $k \in 1...5$, such that the ground truth concept can be accessed for all exercises, as shown on the top x-axis of the heat map in Figure 4. Exercises from the same concept are labeled with squares in the same color. The left heat map in Figure 4 shows the correlation weight between 50 distinct exercises and 5 latent concepts (generated from DKVMN when the memory size is five). Each

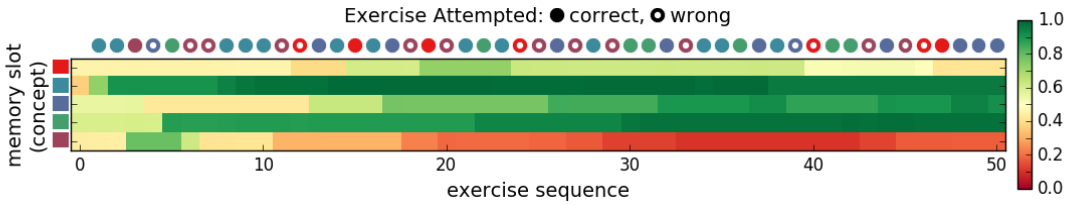


Figure 6: An example of a student's changing knowledge state on 5 concepts. Concepts are marked in different colors on the left side. After answering 50 exercises, the student masters the second, third, and fourth concepts but fails to understand the fifth concept. (Best viewed in color.)

column represents the correlation weight between an exercise and five latent concepts. For each exercise, the weight is sparse where exactly one value approximates 1 and the others approximate 0. After clustering each exercise to the concept with the maximum weight value, we get the graph shown in the right part of Figure 4, which reveals a perfect clustering of five latent concepts. The adjusted mutual information [29] of our clustering result and the ground truth is 1.0.

Moreover, when the memory size N is set to be larger than the ground truth 5, e.g., 50, our model can also end up with 5 exercise clusters and find the appropriate concept for each exercise. Additional results are described in the appendix.

On the ASSISTments2009 dataset, no ground truth concept is used for each exercise. However, the name for each exercise tag can be obtained, as shown in the right part of Figure 5. Each exercise tag is followed by a name. The resulting cluster graph in Figure 5 is drawn using t-SNE [15] by projecting the multi-dimensional correlation weights to the 2-D points. All exercises are grouped into 10 clusters, where the exercises from the same cluster (concept) are labeled in the same color. The clustering graph reveals many reasonable results. Some related exercises are close to one another in the cluster. For example, in the first cluster, 30 *Ordering Fractions*, 33 *Ordering Integers*, 37 *Ordering Positive Decimals*, and 62 *Ordering Real Numbers* are clustered together, which exposes the concept of *elementary arithmetic*.

4.5 Knowledge State Depiction

Our DKVMN can also be used to depict the changing knowledge state of students. Depicting the knowledge state, especially each concept state, is helpful for the users on online learning platforms. If students possess their concept states of all concepts, which pinpoint their strengths and weaknesses, they will be more motivated to fill in the learning gaps independently. A student's changing knowledge state can be obtained in the read process using the following steps.

First, the content in the *value* component is directly used as the read content $\mathbf{r}_t$ in Eq.(3), which can be accessed by setting the correlation weight $\mathbf{w}_t$ to be $[0, .., w_i, ..0]$, where w_i of concept c^i is equal to 1.

Then, we mask the weight of the input content embedding in Eq.(4) to ignore the information of exercises:

$$\mathbf{f}_t = \text{Tanh}([\mathbf{W}_1^r, \mathbf{0}]^T[\mathbf{r}_t, \mathbf{m}_t] + \mathbf{b}_1), \tag{11}$$

where W_1 is split into two parts $\mathbf{W}_1^r$ and $\mathbf{W}_1^m$, and let $\mathbf{W}_1^m = \mathbf{0}$.

Finally, we compute the scalar p as in Eq.(5) to be the predictive mastery level of a concept (concept state).

Figure 6 shows an example of depicting a student's five changing concept states. The first column represents the initial state of each concept before the student answers any exercise, such state differs from concept to concept. Owing to our model's ability to discover concepts for each exercise, each time the student answers an exercise, the concept state of the discovered concept will increase or decrease. For example, when the student answers the first three exercises correctly, concept states of the second and fifth concepts increase; when the student answers the fourth exercise incorrectly, the concept state of the third concept decreases. After answering 50 exercises, the student is shown to have mastered the second, third, and fourth concepts but failed to understand the fifth concept.

5. CONCLUSIONS AND FUTURE WORK

This work proposes a new sequence learning model called DKVMN to tackle the KT problem. The model can be implemented in online learning platforms to improve the study efficiency of students. DKVMN not only outperforms the state-of-the-art DKT but can also trace a student's understanding of each concept over time, which is the main drawback of DKT. Compared with standard MANNs, the key-value pair allows DKVMN to discover underlying concepts for each input exercise and trace a student's knowledge state of all concepts.

For future work, we will incorporate content information into the exercise and concept embeddings to further improve the representations. We will also investigate a hierarchical key-value memory networks structure which can encode the hierarchical relationship between concepts.

6. ACKNOWLEDGEMENT

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 of the General Research Fund) and Ministry of Education of China (D.01.16.00101).

7. APPENDIX

7.1 Concept Discovery

When memory size N is set to 50 for the synthetic-5 dataset, the exercises can still be clustered into five categories. The heat map in Figure 7 describes all correlation weight vectors, which only fall into several concepts. After clustering each exercise to the concept with the maximum weight value, the adjusted mutual information of our clustering result and the ground truth is 0.879. Additionally, if we

cluster using t-SNE (in Figure 7), then the adjusted mutual information will be 1.0, which reveals a perfect result.

7.2 Read Attention Mechanism of MANN

For each input key $\mathbf{k}_t$, we compute the cosine similarity of the key and memory:

$$K[\mathbf{k}_t, \mathbf{M}_t(i)] = \frac{\mathbf{k}_t \cdot \mathbf{M}_t(i)}{\|\mathbf{k}_t\| \cdot \|\mathbf{M}_t(i)\|}, \tag{12}$$

which is then used to compute the read weight $\mathbf{w}^r$ through a softmax with a positive key strength β_t:

$$w_t^r(i) = \frac{\exp(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(i)])}{\sum_j \exp(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(j)])}. \tag{13}$$

7.3 Write Attention Mechanism of MANN

The LRUA model [24] writes the keys either to the least used memory location or the most recently used memory location.

First, a usage weight vector $\mathbf{w}_t^u$ is used to record the usage frequency of all memories: The usage weights are updated at each time-step by decaying the previous usage weights and adding the current reading and writing weights:

$$\mathbf{w}_t^u = \gamma \mathbf{w}_{t-1}^u + \mathbf{w}_t^r + \mathbf{w}_t^w, \tag{14}$$

where γ is a decay parameter. γ is fixed to be 0.9 in our implementation. Then the least-used weight $\mathbf{w}_t^{lu}$ is defined to record the least-used memories using a notation $m(\mathbf{v}, n)$, which denotes n^{th} smallest element of the vector $\mathbf{v}$,

$$w_t^{lu}(i) = \begin{cases} 0 & \text{if } w_t^u(i) > m(\mathbf{w}_t^u, n) \\ 1 & \text{if } w_t^u(i) \leq m(\mathbf{w}_t^u, n), \end{cases} \tag{15}$$

where n is set to equal the number of reads to memory.

Now the write weight $\mathbf{w}_t^w$ is the convex combination of the previous read weights and previous least-used weights:

$$\mathbf{w}_t^w = \sigma(\alpha)\mathbf{w}_{t-1}^r + (1 - \sigma(\alpha))\mathbf{w}_{t-1}^{lu}, \tag{16}$$

where $\sigma(\cdot)$ is a sigmoid function, and α is a scalar gate parameter to interpolate between two weights.

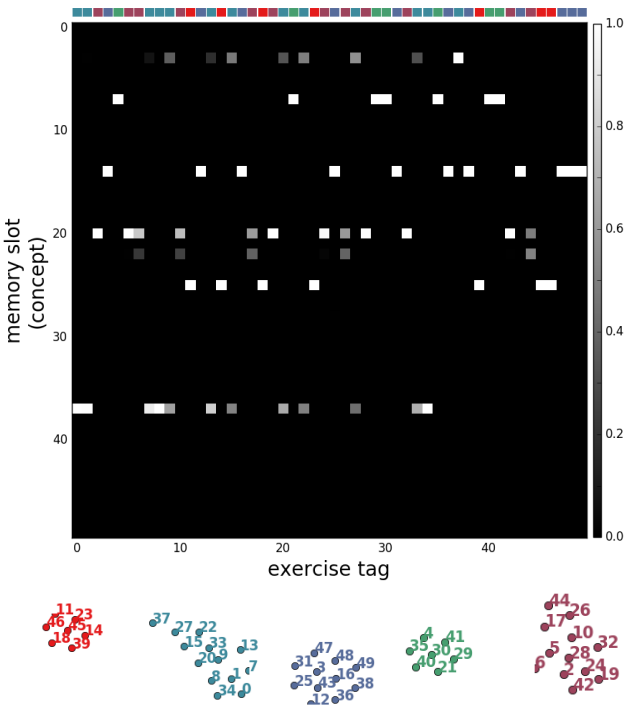


Figure 7: Concept discovery results on the synthetic-5 dataset when the memory size N is set to 50. In the heat map, the x-axis represents each exercise and the y-axis represents the correlation weight between the exercise and five latent concepts generated from our DKVMN model. In the below exercise clustering graph using t-SNE, each node number represents an exercise. Exercises from the same ground-truth concept are clustered together. (Best viewed in color.)

8. REFERENCES

[1] A. Bordes, N. Usunier, S. Chopra, and J. Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[2] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. In *In Neural Information Processing Systems, Workshop on Machine Learning Systems*, 2015.

[3] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. volume 4, pages 253–278. Springer, 1994.

[4] R. S. d Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International Conference on Intelligent Tutoring Systems*, pages 406–415. Springer, 2008.

[5] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*, 19(3):243–266, 2009.

[6] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[7] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[8] E. Grefenstette, K. M. Hermann, M. Suleyman, and P. Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1828–1836, 2015.

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

[10] A. Joulin and T. Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems*, pages 190–198, 2015.

[11] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? In *Educational Data Mining 2016*, 2016.

[12] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the edm community: The pslc datashop. *Handbook of educational data mining*, 43, 2010.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[14] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[15] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[16] 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.

[17] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

[18] G. B. Orr and K.-R. Müller. *Neural networks: tricks of the trade*. Springer, 2003.

[19] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *User Modeling, Adaptation, and Personalization*, pages 255–266. Springer, 2010.

[20] Z. A. Pardos and N. T. Heffernan. Kt-idem: Introducing item difficulty to the knowledge tracing model. In *User Modeling, Adaption and Personalization*, pages 243–254. Springer, 2011.

[21] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318, 2013.

[22] 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.

[23] J. Reye. Student modelling based on belief networks. *International Journal of Artificial Intelligence in Education*, 14(1):63–96, 2004.

[24] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1842–1850, 2016.

[25] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[26] P. Steif and N. Bier. Oli engineering statics - fall 2011. Feb. 2014.

[27] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[28] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[29] P. Viola and W. M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1997.

[30] J. Weston, S. Chopra, and A. Bordes. Memory networks. *International Conference on Learning Representations*, 2015.

[31] K. H. Wilson, X. Xiong, M. Khajah, R. V. Lindsey, S. Zhao, Y. Karklin, E. G. Van Inwegen, B. Han, C. Ekanadham, J. E. Beck, et al. Estimating student proficiency: Deep learning is not the panacea. In *In Neural Information Processing Systems, Workshop on Machine Learning for Education*, 2016.

[32] X. Xiong, S. Zhao, E. G. Van Inwegen, and J. E. Beck. Going deeper with deep knowledge tracing. In *Educational Data Mining 2016*, 2016.

[33] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *Artificial intelligence in education*, pages 171–180. Springer, 2013.