Knowledge Tracing with Sequential Key-Value Memory Networks

Ghodai Abdelrahman and Qing Wang
Research School of Computer Science, Australian National University
Canberra, ACT 0200
{ghodai.abdelrahman,qing.wang}@anu.edu.au

ABSTRACT

Can machines trace human knowledge like humans? Knowledge tracing (KT) is a fundamental task in a wide range of applications in education, such as massive open online courses (MOOCs), intelligent tutoring systems, educational games, and learning management systems. It models dynamics in a student's knowledge states in relation to different learning concepts through their interactions with learning activities. Recently, several attempts have been made to use deep learning models for tackling the KT problem. Although these deep learning models have shown promising results, they have limitations: either lack the ability to go deeper to trace how specific concepts in a knowledge state are mastered by a student, or fail to capture long-term dependencies in an exercise sequence. In this paper, we address these limitations by proposing a novel deep learning model for knowledge tracing, namely Sequential Key-Value Memory Networks (SKVMN). This model unifies the strengths of recurrent modelling capacity and memory capacity of the existing deep learning KT models for modelling student learning. We have extensively evaluated our proposed model on five benchmark datasets. The experimental results show that (1) SKVMN outperforms the state-of-the-art KT models on all datasets, (2) SKVMN can better discover the correlation between latent concepts and questions, and (3) SKVMN can trace the knowledge state of students dynamics, and a leverage sequential dependencies in an exercise sequence for improved predication accuracy.

KEYWORDS

Knowledge Tracing; Memory Networks; Deep Learning; Sequence Modelling; Key-Value Memory

1 INTRODUCTION

One of the prominent features of human intelligence is the ability to track their current knowledge states in mastering specific skills or concepts. This enables humans to identify gaps in their knowledge states to personalise their learning experience. With the success of artificial intelligence (AI) in modeling various areas of human cognition [7, 10, 19, 21, 34], the question has arisen: can machines trace human knowledge like humans? This motivated the study of *knowledge tracing* (KT), which aims to model the knowledge states of students in mastering skills and concepts, through a sequence of learning activities they participate in [6, 26, 36].

Knowledge tracing is of fundamental importance to a wide range of applications in education, such as massive open online courses (MOOCs), intelligent tutoring systems, educational games, and learning management systems. Improvements in knowledge tracing can drive novel techniques to advance human learning. For example, knowledge tracing can be used to discover students' individual learning needs so that personalised learning and support can be provided to fulfill diverse capabilities of each student [15]. It can also be used by human experts to design new measures of student learning and new teaching materials based on learning strengths and weaknesses of students [27]. Nonetheless, tracing human knowledge using machines is a rather challenging task. This is due to the complexity of human learning behaviors (e.g., memorising, guessing, forgetting, etc.) and the inherent difficulties of modeling human knowledge (i.e. skills and prior background) [26].

Existing knowledge tracing models can be generally classified into two categories: traditional machine learning KT models and deep learning KT models. Among traditional machine learning KT models, *Bayesian Knowledge Tracing* (BKT) is the most popular [6], which models the knowledge tracing problem as predicting the state of a dynamical system that has hidden latent variables (i.e. learning concepts). In addition to BKT, probabilistic graphical models such as Hidden Markov Models (HMMs) [1, 6] or Bayesian belief networks [32] have also been used to model knowledge tracing. To keep the inference computation tractable, traditional machine learning KT models use discrete random state variables with simple transition regimes, which limits their ability to represent complex dynamics between learning concepts. Moreover, these models often assume a first-order Markov chain for an exercise sequence (i.e. considering the most recent observation to be representing the whole history) which also limits their ability to model long-term dependencies in an exercise sequence.

Inspired by recent advances in deep learning [19], several deep learning KT models have been proposed. A pioneer work by Piech et al. [26] reported *Deep Knowledge Tracing* (DKT), which uses a Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) units [11, 31] to predict student performance on new exercises given their past learning history. In DKT, a student's knowledge states are represented by a sequence of hidden states that successively encode relevant information from past observations over time. Although DKT has achieved substantial improvements in prediction performance over BKT, due to the limitation of representing a knowledge state by one hidden state, it lacks the ability to go deeper to trace how specific concepts are mastered by a student (i.e., concept states) in a knowledge state. To deal with this limitation, *Dynamic Key-Value Memory Networks* (DKVMN) [36] was proposed to model a student's knowledge state as a complex function over
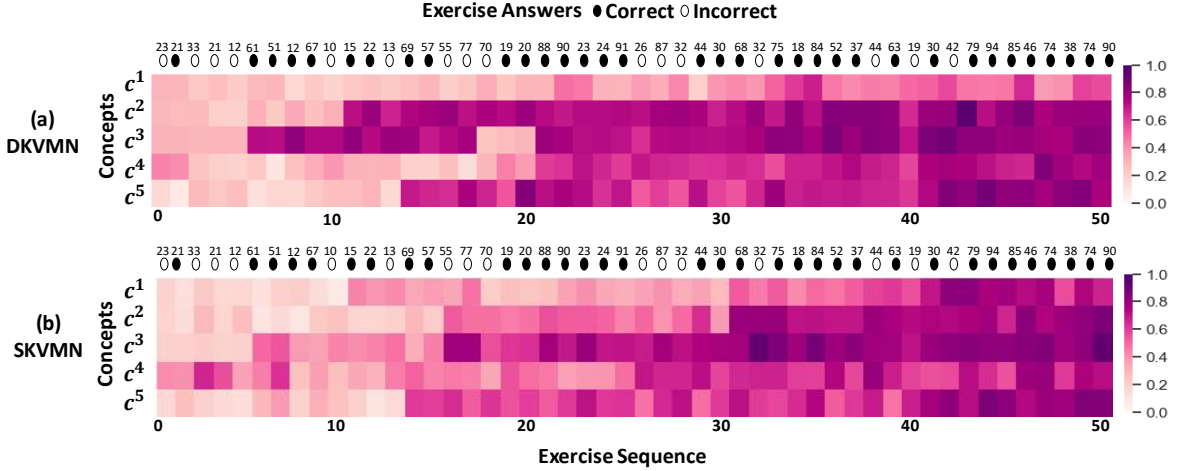
Figure 1: An illustration of how a student's knowledge states are evolving for a sequence of 50 exercises in the dataset ASSIST-ments2009 using: (a) DKVMN and (b) SKVMN. There are 5 concepts $\{c^1, \ldots, c^5\}$ underlying these 50 exercises. Each row depicts the evolution of the concept states for a specific concept, and each column depicts the knowledge state at a certain time step.

all underlying concept states using a key-value memory. <mark>Their idea of augmenting DKVMN with an auxiliary memory follows the concepts of Memory-Augmented Neural Networks (MANN)</mark> [12, 28]. However, DKVMN acquires the knowledge growth through the most recent exercise and thus fails to capture long-term dependencies in an exercise sequence (i.e., relevant past experience to a new observation).

In this paper, we present a new KT model, called *Sequential Key-Value Memory Networks* (SKVMN). This model provides three advantages over the existing deep learning KT models:

- First, <mark>SKVMN unifies the strengths of both recurrent modelling capacity of DKT and memory capacity of DKVMN for modelling student learning.</mark> We observe that, although a key-value memory can help trace concept states of a student, it is not effective in modeling long-term dependencies on sequential data. We remedy this issue by incorporating LSTMs into the sequence modelling for a student's knowledge states over time. Thus, <mark>SKVMN is not only augmented with a key-value memory to enhance representation capability of knowledge states at each time step, but also can provide recurrent modelling capability for capturing dependencies among knowledge states at different time steps in a sequence.</mark>
- Second, <mark>SKVMN uses a modified LSTM with hops, called *Hop-LSTM*, in its sequence modelling.</mark> Hop-LSTM deviates from the standard LSTM architecture by using a triangular layer for discovering sequential dependencies between exercises in a sequence. Then, the model may hop across LSTM cells according to the relevancy of the latent learning concepts. This enables relevant exercises that correlate to similar concepts to be processed together. In doing so, the inference becomes faster and the capacity of capturing long-term dependencies in an exercise sequence is enhanced.
- Third, SKVMN improves the write process of DKVMN in order to better represent knowledge states stored in a

key-value memory. In DKVMN, the current knowledge state is not considered when calculating the knowledge growth of a new exercise. This means that the previous learning experience is ignored. For example, when a student attempts the same question multiple times, the same knowledge growth would be added to the knowledge state, regardless of whether the student has previously answered this question or how many times the answers were correct. SKVMN solves this issue by using a summary vector as input for the write process, which reflects both the current knowledge state of a student and the prior difficulty of a new question.

We have extensively evaluated our proposed model SKVMN on five well-established KT benchmark datasets, and compared it with the state-of-the-art KT models. The experimental results show that (1) SKVMN outperforms the existing KT models, including DKT and DKVMN, on all five datasets, (2) SKVMN can better discover the correlation between latent concepts and questions, and (3) SKVMN can the knowledge state of students dynamics, and leverage sequential dependencies between exercises in an exercise sequence for improved predication accuracy.

Figure 1 illustrates how a student's knowledge states are evolving as the student attempts a sequence of 50 exercises in DKVMN and SKVMN. We can see that, compared with the knowledge states of DKVMN depicted in Figure 1.(a), our model SKVMN provides a smoother transition between two successive concept states as depicted in Figure 1.(b). Moreover, SKVMN captures a smooth, progressive evolution of knowledge states over time (i.e., through this exercise sequence), which more accurately reflects the way a student learns (will be discussed in detail in Section 5.4).

The reminder of this paper is organized as follows. Section 2 defines the knowledge tracing problem. Section 3 presents our proposed KT model SKVMN. Sections 4 and 5 discuss the experimental design and results. The related work is presented in Section 6. We conclude the paper in Section 7.
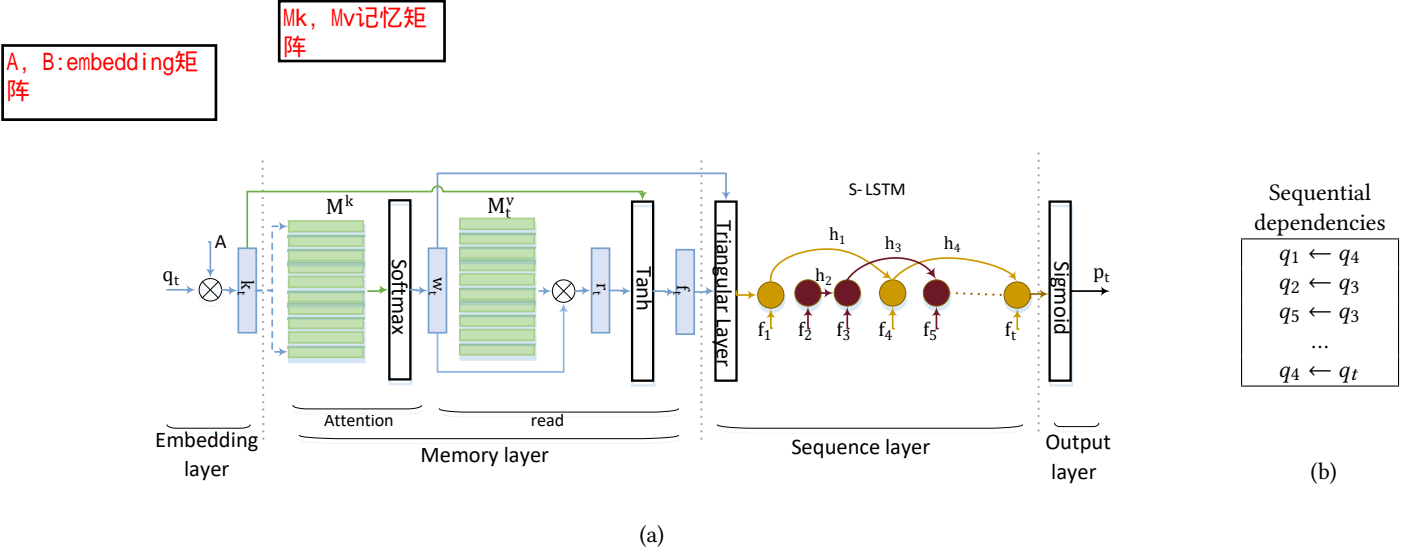
Figure 2: An illustration of the SKVMN model at time step t, where q_t is the input question, the exercise history $X = \langle (q_1, y_1), (q_2, y_2), \ldots, (q_{t-1}, y_{t-1}) \rangle$, and each f_i is the summary vector of the question q_i for $i \in [1, t]$: (a) the model has four layers, namely the embedding, memory, sequence and output layers; and (b) sequential dependencies among questions in X.

2 PROBLEM FORMULATION

Broadly speaking, knowledge tracing is to track down students' knowledge states over time through a sequence of observations on how the students interact with given learning activities. In this paper, we formulate knowledge tracing as a sequence prediction problem in machine learning, which learns the knowledge state of a student based on an exercise answering history.

Let $Q = \{q_1, \ldots, q_{|Q|}\}$ be the set of all distinct question tags in a dataset. Each $q_i \in Q$ may have a different level of difficulty, which is not explicitly provided. An *exercise* x_i is a pair (q_i, y_i) consisting of a question tag q_i and a binary variable $y_i \in \{0, 1\}$ representing the answer, where 0 means that q_i is incorrectly answered and 1 means that q_i is correctly answered. When a student interacts with the questions in Q, a history of exercises $X = \langle x_1, x_2, \ldots, x_{t-1} \rangle$ undertaken by the student can be observed. Based on a history of exercises $X = \langle x_1, x_2, \ldots, x_{t-1} \rangle$, we want to predict the probability of correctly answering a new question at time step t by the student, i.e., $p_t = (y_t = 1|q_t, X)$.

We assume that the questions in Q are associated with N latent concepts C. The *concept state* of each latent concept $c^i \in C$ is a random variable describing the mastery level of the student on this latent concept. At each time step t, the *knowledge state* of a student is modelled as a set of all concept states of the student, each corresponding to a latent concept in C at time step t.

3 SEQUENTIAL KEY-VALUE MEMORY NETWORKS

In this section, we introduce our model *Sequential Key-Value Memory Networks* (SKVMN). We first present an overview for SKVMN. Then, we show how a key-value memory can be attended, read and written in our model. To leverage sequential dependencies among latent concepts for predication, we then present a modified LSTMs, called *Hop-LSTMs*. Lastly, we discuss the optimisation techniques used in the model.

3.1 Model Overview

The SKVMN model is augmented with a key-value memory $\langle \mathbf{M^k}, \mathbf{M^v} \rangle$ following the work in [36], i.e., a pair of one static matrix $\mathbf{M^k}$ of size $N \times d_k$, called the *key matrix*, and one dynamic matrix $\mathbf{M^v}$ of size $N \times d_v$, called the *value matrix*. Both the key matrix and the value matrix have the same N memory slots, but they may differ in their state dimensions d_k and d_v. The key matrix stores the latent concepts underlying questions, and the value matrix stores the concept states of a student (i.e., the knowledge state) which can be changed dynamically based on student learning.

Given an input question q_t at time step t, the SKVMN model retrieves the knowledge state of a student from the key-value memory $\langle \mathbf{M^k}, \mathbf{M^v} \rangle$, and predicts the probability of correctly answering the question q_t by the student. Figure 2.(a) illustrates the SKVMN model at time step t, which consists of four layers: the embedding, memory, sequence and output layers.

- The embedding layer is responsible for mapping an input question at time step t into a high-dimensional vector space.
- The memory layer involves two processes: attention and read, where the attention process provides an addressing mechanism for the input question q_t to allocate the relevant information from the key-value memory, and the read process uses the attention vector to retrieve the current knowledge state of the student from the value matrix $\mathbf{M^v_t}$. The details of the attention and read processes will be discussed in Sections 3.2.1 and 3.2.2.
- The sequence layer consists of a set of recurrently connected LSTM cells, where LSTM cells are connected based on their sequential dependencies determined by a Triangular layer as depicted in Figure 2.(b). The details of the sequence layer will be discussed in Section 3.3.
- The output layer generates the probability of correctly answering the input question q_t.

After a student has attempted the input question q_t with the answer y_t, the value matrix in the key-value memory of the SKVMN model needs to be updated in order to reflect the latest knowledge state of the student. Figure 3 depicts how the value matrix is transited from $\mathbf{M}_t^v$ at time step t to $\mathbf{M}_{t+1}^v$ at time step t + 1 using the write process. The details of the write process will be discussed in Section 3.2.3.

3.2 Attention, Read and Write

There are three processes relating to access to a key-value memory in our model: attention, read and write. In the following, we elaborate these processes.

3.2.1 Attention. Given a question $q_t \in Q$ as input, the model represents q_t as a "one-hot" vector of length $|Q|$ in which all entries are zero, except for the entry that corresponds to q_t.

In order to map q_t into a continuous vector space, q_t is multiplied by an embedding matrix $\mathbf{A} \in \mathbb{R}^{|Q| \times d_k}$, which generates a continuous embedding vector $\mathbf{k}_t \in \mathbb{R}^{d_k}$, where d_k is the embedding dimension. Then, an attention vector $\mathbf{w}_t$ is obtained by applying the Softmax function to the inner product between the embedding vector $\mathbf{k}_t$ and each key slot $\mathbf{M}^k(i)$ in the key matrix $\mathbf{M}^k$ as follows:

$$\mathbf{w}_t(i) = \text{Softmax}(\mathbf{k}_t^T \mathbf{M}^k(i)) \quad (1)$$

where $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$. Conceptually, $\mathbf{w}_t$ represents the correlation between the question q_t and the underlying latent concepts stored in the key matrix $\mathbf{M}^k$.

3.2.2 Read. For each exercise $x_t = (q_t, y_t)$, the model uses its corresponding attention vector $\mathbf{w}_t$ to retrieve the concept states of the student with regard to the question q_t from the value matrix $\mathbf{M}_t^v$. Specifically, the read process takes the attention vector $\mathbf{w}_t$ as input and yields a read vector $\mathbf{r}_t$ which is the weighted sum of all values being attended by $\mathbf{w}_t$ in the memory slots of the value matrix $\mathbf{M}_t^v$, i.e.,

$$\mathbf{r}_t = \sum_{i=1}^{N} \mathbf{w}_t(i) \mathbf{M}_t^v(i) \quad (2)$$

The read vector $\mathbf{r}_t$ is concatenated with the embedding vector $\mathbf{k}_t$. Then, the combined vector is fed to a Tanh layer to calculate the summary vector $\mathbf{f}_t$:

$$\mathbf{f}_t = \text{Tanh}(\mathbf{W}_1^T[\mathbf{r}_t, \mathbf{k}_t] + \mathbf{b}_1) \quad (3)$$

where $\text{Tanh}(z_i) = (e^{z_i} - e^{-z_i})/(e^{z_i} + e^{-z_i})$, $\mathbf{W}_1$ is the weight matrix of the Tanh layer, and $\mathbf{b}_1$ is the bias vector. While the read vector $\mathbf{r}_t$ represents the student's knowledge state with respect to the relevant concepts of the current question q_t, the summary vector $\mathbf{f}_t$ adds prior information of the question (e.g., the level of difficulty) to this knowledge state. Intuitively, $\mathbf{f}_t$ represents how well the student has mastered the latent concepts relevant to the question q_t before attempting this question.

3.2.3 Write. The write process occurs each time after the student has attempted a question. The purpose of this write process is to update the concept states of the student in the value matrix $\mathbf{M}_t^v$ using the knowledge growth gained through attempting the question q_t. This update leads to the transition of the value matrix from from $\mathbf{M}_t^v$ to $\mathbf{M}_{t+1}^v$ as depicted in Figure 3.
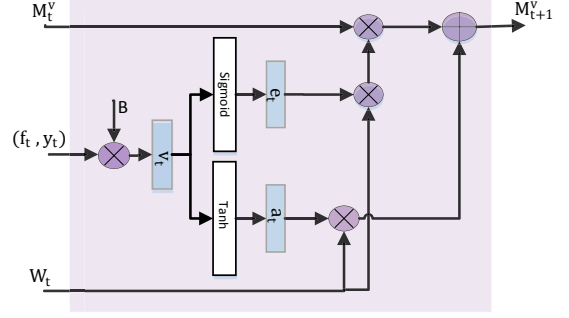

Figure 3: An illustration of the write process in the SKVMN model, which transits the value matrix from $\mathbf{M}_t^v$ at time step t to $\mathbf{M}_{t+1}^v$ at time step t + 1, where $(\mathbf{f}_t, y_t)$ and $\mathbf{w}_t$ are the input to the write process at time step t.

To calculate the knowledge growth, our model considers not only the correctness of the answer y_t for q_t, but also the student's mastery level of the concept states $\mathbf{f}_t$ before attempting q_t. Each $(\mathbf{f}_t, y_t)$ is represented as a vector of length $2|Q|$ and multiplied by an embedding matrix $\mathbf{B} \in \mathbb{R}^{2|Q| \times d_v}$ to get a write vector $\mathbf{v}_t$, which represents the knowledge growth of the student obtained by attempting the question.

Similar to other memory augmented networks [12, 36], the write process proceeds with two gates: *erase gate* and *add gate*. The former controls what information to erase from the knowledge state after attempting the latest exercise x_t, while the latter controls what information to add into the knowledge state. From the knowledge tracing perspective, these two gates capture the forgetting and enhancing aspects of learning knowledge, respectively.

With the write vector $\mathbf{v}_t$ for the knowledge growth, an erase vector $\mathbf{e}_t$ is calculated as:

$$\mathbf{e}_t = \text{sigmoid}(\mathbf{W}_e^T \cdot \mathbf{v}_t + b_e) \quad (4)$$

where $\text{Sigmoid}(z_i) = 1/(1 + e^{-z_i})$ and $\mathbf{W}_e$ is the weight matrix. Using the attention vector $\mathbf{w}_t$, the value matrix $\tilde{\mathbf{M}}_{t+1}^v$ after applying the erase vector $\mathbf{e}_t$ is

$$\tilde{\mathbf{M}}_{t+1}^v(i) = \mathbf{M}_t^v(i)[1 - \mathbf{w}_t(i)\mathbf{e}_t]. \quad (5)$$

Then, an add vector $\mathbf{a}_t$ is calculated by

$$\mathbf{a}_t = \text{Tanh}(\mathbf{W}_a^T \cdot \mathbf{v}_t + b_a) \quad (6)$$

where $\mathbf{W}_a$ is the weight matrix. Finally, the value matrix $\mathbf{M}_{t+1}^v$ for the next knowledge state is updated as:

$$\mathbf{M}_{t+1}^v(i) = \tilde{\mathbf{M}}_{t+1}^v(i) + \mathbf{w}_t(i)\mathbf{a}_t \quad (7)$$

3.3 Sequence Modelling

Now we discuss the sequence modelling approach used at the sequence layer for predicting the probability of correctly answering the question q_t based on the exercise history.

3.3.1 Sequential dependencies. An exercise history X of a student may contain a long sequence of exercises, for example, the average sequence length in the ASSISTments2009 dataset is 233 ± 100 questions per sequence. However, as different exercises

may correlate to different latent concepts, not all exercises in X can equally contribute to the prediction of answering a given question q_t. Thus, we observe that, by hopping across irrelevant exercises in X with regard to q_t, recurrent models can be applied on a shorter and more relevant sequence, leading to more efficient and accurate prediction performance.

For each question in Q, since its attention vector reflects the correlation between this question and the latent concepts in C, we consider that the similarity between two attention vectors can provide a good indication of how their corresponding questions are relevant in terms of their correlations with latent concepts. For example, suppose that we have three latent concepts $C = \{c^1, c^2, c^3\}$, and two attention vectors $\mathbf{w_1} = [0.15, 0.25, 0.6]^T$ and $\mathbf{w_2} = [0.2, 0.3, 0.5]^T$ that correspond to the questions q_1 and q_2, respectively. Then q_1 and q_2 are considered as being relevant if both $\mathbf{w_1}$ and $\mathbf{w_2}$ are mapped to a vector $[0, 0, 1]^T$ where 0, 1 and 2 refer to the value ranges "low", "middle" and "high", respectively.

Now, the question is: how to identify similar attention vectors? For this, we use the triangular membership function [17]:

$$\mu(x) = max(min(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0), \tag{8}$$

where the parameters a and c determine the feet and the parameter b determines the peak of a triangle. We use three triangular membership functions for three value ranges: low (0), medium (1), and high (2). Each real-valued component in an attention vector is mapped to one of the three value ranges. Each attention vector $\mathbf{w_t}$ is associated with an identity vector $\mathbf{d_t}$. Similar attention vectors have the same identity vector, while dissimilar attention vectors have different identity vectors.

Then, at each time step t, for the current question q_t and an exercise history X, we say q_t is *sequentially dependent* on $q_{t-\lambda}$ in X with $\lambda \in (0, t)$, denoted as $q_{t-\lambda} \leftarrow q_t$, if the following two conditions are satisfied:

- The attention vectors of $q_{t-\lambda}$ and q_t have the same identity vector, i.e., $\mathbf{d}_{t-\lambda} = \mathbf{d_t}$, and
- There is no other q_j in X such that $\mathbf{d_j} = \mathbf{d_t}$ and $j > t - \lambda$, i.e., $q_{t-\lambda}$ is the most recent exercise in X that is relevant to q_t.

Over time, given a sequence of questions $\langle q_1, q_2, \ldots, q_{|Q|} \rangle$, we thus have an exercise history X' in which exercises are partitioned into a set of subsequences $\{X'_1, \ldots, X'_n\}$ with $\Sigma_{k=1}^{n} |X'_k| = |X'|$ and, for any two consecutive exercises (q_i, y_i) and (q_j, y_j) in the subsequence X'_k, q_j is sequentially dependent on q_i, i.e. $q_i \leftarrow q_j$. In Figure 2, based on the sequential dependencies presented in Figure 2.(b), X is partitioned into $\langle q_1, q_4, \ldots, q_t \rangle$ and $\langle q_2, q_3, q_5, \ldots \rangle$, which correspond to the recurrently connected LSTM cells depicted in Figure 2.(a). We will discuss further details in the following.

3.3.2 Hop-LSTM. Based on sequence dependencies between exercises in a sequence, a modified LSTM with hops, called Hop-LSTM, is used to predict the probability of correctly answering a new question by a student. Different from the standard LSTM architecture, Hop-LSTM allows us to recurrently connect the LSTM cells for questions based on the relevance of their latent concepts. More precisely, two LSTM cells in Hop-LSTM are connected only if the input question of one LSTM cell is sequentially dependent

on the input question of the other LSTM cell. This means that Hop-LSTM has the capability of hopping across the LSTM cells when their input questions are irrelevant to the current question q_t.

Formally, at each time step t, for the question q_t, if there is an exercise $(q_{t-\lambda}, y_{t-\lambda}) \in X$ with $\lambda \in (0, t)$ and $q_{t-\lambda} \leftarrow q_t$, then the current LSTM cell takes the summary vector $\mathbf{f}_t$ and the hidden state $\mathbf{h}_{t-\lambda}$ as input. Moreover, this LSTM cell updates the cell state $\mathbf{c}_{t-\lambda}$ into the cell state $\mathbf{c}_t$, and generates the new hidden state $\mathbf{h}_t$ as output. As in [13], the LSTM cell used in our work has three gates: *forget gate* $\mathbf{g}_t$, *input gate* $\mathbf{i}_t$, and *output gate* $\mathbf{o}_t$ in addition to the hidden state $\mathbf{h}_t$ and the cell state $\mathbf{c}_t$:

$$\mathbf{g}_t = Sigmoid(\mathbf{W}_g[\mathbf{h}_{t-\lambda}, \mathbf{f}_t] + \mathbf{b}_g) \tag{9}$$

$$\mathbf{i}_t = Sigmoid(\mathbf{W}_i[\mathbf{h}_{t-\lambda}, \mathbf{f}_t] + \mathbf{b}_i) \tag{10}$$

$$\mathbf{o}_t = Sigmoid(\mathbf{W}_o[\mathbf{h}_{t-\lambda}, \mathbf{f}_t] + \mathbf{b}_o) \tag{11}$$

$$\tilde{\mathbf{c}}_t = Tanh(\mathbf{W}_c[\mathbf{h}_{t-\lambda}, \mathbf{f}_t] + \mathbf{b}_c) \tag{12}$$

$$\mathbf{c}_t = \mathbf{g}_t \odot \mathbf{c}_{t-\lambda} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{13}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot Tanh(\mathbf{c}_t) \tag{14}$$

Then, the output vector $\mathbf{h}_t$ of the curent LSTM cell is sent to a Sigmoid layer, which calculates the probability p_t of correctly answering the current question q_t by

$$p_t = Sigmoid(\mathbf{W}_2^T \cdot \mathbf{h}_t + \mathbf{b}_2). \tag{15}$$

3.4 Model Optimisation

To optimise the model, we use the cross-entropy loss function between the predicted probability of being correctly answered p_t and the true answer y_t. The following objective function is defined over training data:

$$\mathcal{L} = -\sum_t (y_t \log p_t + (1 - y_t) \log(1 - p_t)) \tag{16}$$

We initialise of the memory matrices ($\mathbf{M_t^k}$ and $\mathbf{M_t^v}$) and embedding matrices ($\mathbf{A}$ and $\mathbf{B}$) using a random Gaussian distribution $N(0, \sigma)$. While for weights and biases of the neural layers, we use Glorot uniform random initialization [9] for a faster convergence. These randomly initialized parameters are optimized using the stochastic gradient decent (SGD) mechanism [3]. As we use Hop-LSTM at the sequence layer, during the backpropagation of the gradients, only the parameters of the connected LSTM cells (i.e. the ones responsible for the current prediction error) are updated. Other parameters, such as the embedding matrices, weight matrices, and bias vectors, are updated in each backpropagation iteration based on loss function values.

Note that, through training, the model can discover relevant latent concepts for each question and store their state values in the value matrix $\mathbf{M^v}$.

4 EXPERIMENTS

In this section, we present the experiments of evaluating our proposed model SKVMN against the state-of-the-art KT models. These experiments aim to answer the following research questions:

RQ1: What is the optimal size for a key-value memory (i.e., the key and value matrices) of SKVMN?

RQ2: How does SKVMN perform on predicting a student's answers of new questions, given an exercise history?

RQ3: How does SKVMN perform on discovering the correlation between latent concepts and questions?

RQ4: How does SKVMN perform on capturing the evolution of a student's knowledge states?

4.1 Datasets

We use five well-established datasets in the KT literature [15, 26, 36]. Table 1 summarizes the statistics of the data sets.

- **Synthetic-5**[1]: This dataset consists of two subsets: one for training and one for testing. Each subset contains 50 distinct questions which were answered by 4, 000 virtual students. A total number of 200, 000 exercises (i.e. (q_t, y_t)) are contained in the dataset.
- **ASSISTments2009**[2]: This dataset was collected during the school year $2009 - 2010$ using the ASSISTments online education website [3]. The dataset consists of 110 distinct questions answered by 4, 151 students which gives a total number of 325, 637 exercises.
- **ASSISTments2015**[4]: As an update to the ASSISTments2009 dataset, this dataset was released in 2015. It includes 100 distinct questions answered by 19, 840 students with a total number of 683, 801 exercises. This dataset has the largest number of students among the other datasets. Albeit, the average number of exercises per student is low. The original dataset also has some incorrect answer values (i.e. $y_i \notin \{0, 1\}$), which are removed during preprocessing.
- **Statics2011**[5]: This datasets was collected from a statistics course at Carnegie Mellon University during Fall 2011. It contains 1, 223 distinct questions answered by 333 undergraduate students with a total number of 189, 297 exercises. This dataset has the highest exercise per student ratio among all datasets.
- **JunyiAcademy**[6]: This dataset was collected from Junyi Academy [7], which is an education website of providing learning materials and exercises on various scientific courses, on 2015 [5]. It contains 722 distinct questions answered by 199, 549 students with a total number of 25, 628, 935 exercises. It is the largest dataset in terms of the number of exercises.

4.2 Baselines

In order to evaluate the performance of our proposed model, we select the following three KT models as the baselines:

- Bayesian knowledge tracing (BKT) [6], which is based on Bayesian inference in which a knowledge state is modelled as a set of binary variables, each representing the understanding of a single 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]https://www.assistments.org/
[4]ASSISTments2015:https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data
[5]Statics2011:https://pslcdatashop.web.cmu.edu/ DatasetInfo?datasetId=507
[6]Junyi2015: https://datashop.web.cmu.edu/DatasetInfo?datasetId=1198
[7]https://www.junyiacademy.org/

Table 1: Dataset statistics

Dataset	#Questions	#Students	#Exercises	#Exercises per student
Synthetic-5	50	4, 000	200, 000	50
ASSISTments2009	110	4, 151	325, 637	78
ASSISTments2015	100	19, 840	683, 801	34
Statics2011	1, 223	333	189, 297	568
JunyiAcademy	722	199, 549	25, 628, 935	128

Table 2: Comparison of SKVMN with DKVMN under different numbers of memory slots N and state dimensions d, where m refers to the number of parameters in each setting.

Dataset	d	N	SKVMN		DKVMN	
			AUC (%)	m	AUC (%)	m
Synthetic-5	10	50	83.11	15K	82.00	12k
	50	50	83.67	30k	82.66	25k
	100	50	**84.00**	57k	**82.73**	50k
	200	50	83.73	140k	82.71	130k
ASSISTments2009	10	10	**83.63**	7.8k	81.47	7k
	50	20	82.87	35k	**81.57**	31k
	100	10	82.72	71k	81.42	68k
	200	20	82.63	181k	81.37	177k
ASSISTments2015	10	20	**74.84**	16k	**72.68**	14k
	50	10	74.50	31k	72.66	29k
	100	50	74.24	66k	72.64	63k
	200	50	74.20	163k	72.53	153k
Statics2011	10	10	84.50	92.8k	82.72	92k
	50	10	**84.85**	199k	**82.84**	197k
	100	10	84.70	342k	82.71	338k
	200	10	84.76	653k	82.70	649k
JunyiAcademy	10	20	82.50	16k	79.63	14k
	50	10	82.41	31k	79.48	29k
	100	50	**82.67**	66k	79.54	63k
	200	50	82.32	163k	**80.27**	153k

- Deep knowledge tracing (DKT) [26] which uses recurrent neural networks to model student learning.
- Dynamic key-value memory networks (DKVMN) [36] which extends the memory-augmented neural networks (MANN) by a key-value memory and is considered as the state-of-the-art model for knowledge tracing.

Our proposed model is referred to as SKVMN in the experiments.

4.3 Measures

We use the area under the Receiver Operating Characteristic (ROC) curve, referred to as AUC [20], to measure the prediction performance of the KT models. The AUC ranges in value from 0 to 1. An AUC score of 0.5 means random prediction (i.e. coin flipping). The higher an AUC score goes above 0.5, the more accurately a predictive model can perform.
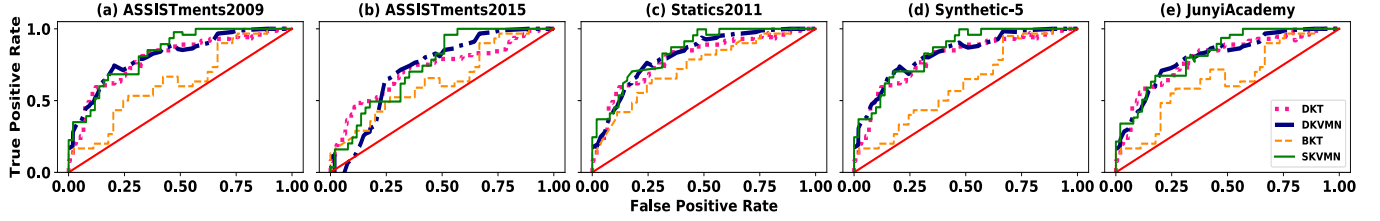
Figure 4: The ROC curve results of the four models BKT, DKT, DKVMN, and SKVMN over five datasets: (a) ASSISTments2009, (b) ASSISTments2015, (c) Statics2011, (d) Synthetic-5, and (e) JunyiAcademy.

4.4 Evaluation Settings

We divided each dataset into 70% for training and validation and 30% for testing, except for Synthetic-5. This is because, as mentioned before, Synthetic-5 itself contains the training and test subsets of the same size. For each training and validation subset, we further divided it using the 5-fold cross validation (e.g. 80% for training and 20% for validation). The validation subset was used to determine the optimal values for the hyperparameters, including the memory slot dimensions d_k for the key matrix and d_v for the value matrix.

Table 3: The AUC results of the four models BKT, DKT, DKVMN, and SKVMN over all datasets.

Dataset	BKT	DKT	DKVMN	SKVMN
Synthetic-5	62.0 ± 0.02	80.3 ± 0.1	82.7 ± 0.1	$\mathbf{84.0 \pm 0.04}$
ASSISTments2009	63.1 ± 0.01	80.5 ± 0.2	81.6 ± 0.1	$\mathbf{83.6 \pm 0.06}$
ASSISTments2015	64.2 ± 0.03	72.5 ± 0.1	72.7 ± 0.1	$\mathbf{74.8 \pm 0.07}$
Statics2011	73.0 ± 0.01	80.2 ± 0.2	82.8 ± 0.1	$\mathbf{84.9 \pm 0.06}$
JunyiAcademy	65.0 ± 0.02	79.2 ± 0.1	80.3 ± 0.4	$\mathbf{82.7 \pm 0.01}$

We utilised the Adam optimizer [16] for SGD implementation with momentum of 0.9 and learning rate γ of 0.01 annealed using a cosine function every 15 epochs for 120 epochs, then it remains fixed at 0.001. The LSTM gradients were clipped to improve the training [25]. For the other baselines, we follow the optimisation procedures indicated in the original work for each of them [6, 26, 36].

A mini-batch of 32 is selected during the training for all datasets, except Synthetic-5, for which we use a mini-batch of 8 due to the relatively small number of training samples (i.e. exercises) in the dataset [2]. For each dataset, the training process is repeated five times, each time using a different initialization. We report the average test AUC and the standard deviation over these five runs.

For the Triangular layer, the hyper-parameter values (a, b, c) of each triangular membership function are set based on the empirical analysis of each dataset.

5 RESULTS AND DISCUSSION

In this section, we present the experimental results and discuss our observations from the obtained results.

5.1 Hyperparameters N and d

To explore how the sizes of the key and value matrices can affect the model performance, we have conducted experiments to compare SKVMN with DKVMN under different numbers of memory slots N and state dimensions d, where $d = d_k = d_v$ so as to be consistent

with the previous work [36]. In order to allow a fair comparison between the models SKVMN and DKVMN, we select the same set of state dimensions, (i.e. d = 10, 50, 100, 200) and the same corresponding numbers of memory slots N on the datasets Synthetic-5, ASSISTments2009, ASSISTments2015 and Statics2011 to report the AUC results, following the settings originally reported in [36]. For the dataset JunyiAcademy, it was not considered in the previous work [36]. Considering that JunyiAcademy has the largest numbers of students and exercises among all datasets, we use the same settings for the numbers of memory slots and state dimensions as the ones for the second largest dataset ASSISTments2015. Table 2 presents the AUC results for all five datasets.

As shown in Table 2, compared with DKVMN, our model SKVMN can produce better AUC results with comparable parameters on the datasets Synthetic-5, ASSISTments2015 and Statics2011, and with fewer parameters on the datasets ASSISTments2009 and JunyiAcademy. Particularly, for the dataset ASSISTments2009, SKVMN yields an AUC at 83.63% with N=10, d=10 and m=7.8k, whereas DKVMN yields an AUC at 81.57% with N=20, d=50 and m=31k (nearly 4 times of 7.8k). Similarly, for the dataset JunyiAcademy, SKVMN yields an AUC at 82.67% with N=50, d=100 and m=66k, whereas DKVMN yields an AUC at 80.27% with N=50, d=200 and m=153k (more than twice of 66k).

Note that, the optimal value of N for ASSISTments2015 is higher than the one for its previous version (i.e. ASSISTments2009). This implies that the number of latent concepts N increases in ASSISTments2015 in comparison to ASSISTments2009. Moreover, the optimal value of d generally reflects the complexity of the exercises in a dataset, and the dataset JunyiAcademy has exercises of higher complexity than other real-world datasets.

5.2 Prediction Accuracy

We have conducted experiments on comparing the AUC results of our model SKVMN with the other three KT models: BKT, DKT, and DKVMN. Table 3 presents the AUC results of all the models. It can be seen that our model SKVMN outperformed the other models over all the five datasets. Particularly, the SKVMN model achieved an average AUC value that is at least 2% higher than the state-of-art model DKVMN on all real-world datasets AS-SISTments2009, ASSISTments2015, Statics2011, and JunyiAcademy. Even for the only synthetic dataset (i.e. Synthetic-5), the SKVMN model achieved an average AUC value of 84.0 ± 0.04, in comparison with 82.7 ± 0.1 achieved by DKVMN. Note that the AUC values on ASSISTments2015 are the lowest among all datasets, regardless
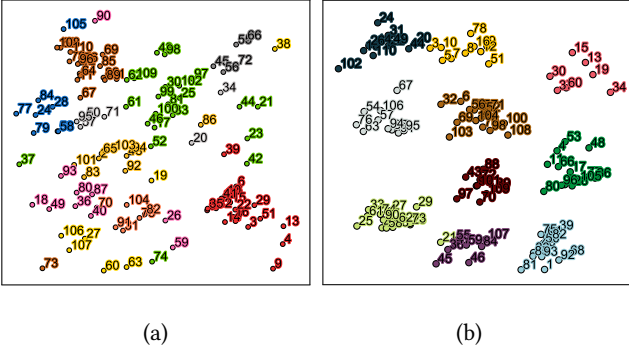
(a)   (b)

Figure 5: Clustering results of questions in the dataset AS-SISTments2009 using: (a) DKVMN, and (b) SKVMN, where questions in the same color are correlating to the same latent concept.

Table 4: Question descriptions in the dataset ASSIST-ments2009, where the questions are clustered according to latent concepts being discovered by SKVMN.

21 Multiplication and Division Integers 23 Absolute Value 25 Subtraction Whole Numbers 27 Order of Operations +,-,/,* () positive reals 29 Counting Methods 33 Ordering Integers 38 Rounding 41 Finding Percents 58 Solving for a variable 61 Estimation 62 Ordering Real Numbers 73 Prime Number 74 Multiplication and Division Positive Decimals 90 Multiplication Whole Numbers	1 Area Trapezoid 39 Volume Rectangular Prism 40 Order of Operations All 50 Pythagorean Theorem 64 Surface Area Rectangular Prism 68 Area Circle 75 Volume Sphere 81 Area Rectangle 82 Area Triangle 83 Area Parallelogram 85 Surface Area Cylinder 92 Rotations 93 Reflection	6 Stem and Leaf Plot 32 Box and Whisker 35 Percent Of 42 Pattern Finding 52 Congruence 56 Reading a Ruler or Scale 69 Least Common Multiple 71 Angles on Parallel Lines Cut by a Transversal 79 Solving Inequalities 98 Intercept 100 Slope 103 Recognize Linear Pattern 104 Simplifying Expressions positive exponents 108 Recognize Quadratic Pattern
43 Write Linear Equation from Situation 70 Equation Solving More Than Two Steps 72 Write Linear Equation from Ordered Pairs 88 Solving Systems of Linear Equations 89 Solving Systems of Linear Equations by Graphing 97 Choose an Equation from Given Information 99 Linear Equations 109 Finding Slope From Equation 101 Angles - Obtuse, Acute, and Right	54 Interior Angles Triangle 57 Perimeter of a Polygon 63 Scale Factor 65 Scientific Notation 67 Percents 76 Computation with Real Numbers 87 Greatest Common Factor 91 Polynomial Factors 94 Translations 95 Midpoint 106 Finding Slope From Situation	2 Area Irregular Figure 4 Table 11 Histogram as Table or Graph 17 Scatter Plot 28 Calculations with Similar Figures 48 Nets of 3D Figures 53 Interior Angles Figures with More than 3 Sides 66 Write Linear Equation from Graph 77 Number Line 80 Unit Conversion Within a System 86 Volume Cylinder 96 Interpreting Coordinate Graphs 105 Finding Slope from Ordered Pairs
14 Proportion 18 Addition and Subtraction Positive Decimals 22 Addition Whole Numbers 20 Addition and Subtraction Integers 24 Addition and Subtraction Fractions 26 Equation Solving Two or Fewer Steps 31 Circumference 44 Square Root 49 Complementary and Supplementary Angles 102 Distributive Property 110 Quadratic Formula to Solve Quadratic Equation		3 Probability of Two Distinct Events 5 Median 7 Mode 8 Mean 9 Range 10 Venn Diagram 12 Circle Graph 16 Probability of a Single Event 51 D.4.8-understanding-concept-of-probabilities 78 Rate
36 Unit Rate 45 Algebraic Simplification 46 Algebraic Solving 47 Percent Discount 55 Divisibility Rules 59 Exponents 84 Effect of Changing Dimensions of a Shape Prportionally 107 Parts of a Polyomial, Terms, Coefficient, Monomial, Exponent, Variable +		13 Equivalent Fractions 15 Fraction Of 19 Multiplication Fractions 30 Ordering Fractions 34 Conversion of Fraction Decimals Percents 37 Ordering Positive Decimals 60 Division Fractions

of the KT models. This reflects the difficulty of the KT task in this dataset due to its lowest exercise per student ratio, which not only makes the training process more difficult but also limits the effective use of sequence information to enhance the prediction performance. Figure 4 illustrates the ROC curves of these four models for each dataset.

In a nutshell, based on the AUC results in Table 3, we have the following observations. First, the neural models generally performed better than the Bayesian inference model (i.e. BKT). This is due to the power of these models in learning complex student learning patterns without the need to oversimplify the problem's assumption to keep it within the tractable computation limits as the case in BKT. Second, the memory-augmented models DKVMN and SKVMN performed better than the DKT model that does not utilise an external memory structure. This has empirically verified the

effectiveness of external memory structures in storing past learning experiences of students, as well as facilitating the access of relevant information to enhance the prediction performance. Third, the use of sequential dependencies among exercises in our SKVMN model enhanced the prediction accuracy in comparison to the DKVMN model which primarily considers the latest observed exercise.

5.3 Clustering Questions

To provide insights on how our proposed model SKVMN can correlate questions to their latent concepts, In Figure 5.(a)-5.(b), we present the clustering results of questions based on their correlated concepts in the dataset ASSISTments2009, generated by using DKVMN and SKVMN, respectively. This dataset was selected for two reasons. First, it has a reasonable number of questions (i.e., 110), enabling the visualization of clusters to be readable. Second, each question in this dataset is provided with a description as depicted in Table 4, which is useful for validating how well the model discovers correlations between questions and their latent concepts.

As shown in Figure 5, both DKVMN and SKVMN discover that there are 10 latent concepts relating to the 110 questions in the dataset ASSISTments2009, where all questions in one cluster relate to common latent concepts and are labelled using the same color. It can be noticed that SKVMN performs significantly better than DKVMN since the overlapping between different clusters in SKVMN is smaller than in DKVMN. For example, in Figure 5.(a), question 105 is about curve slop, which is close to the cluster for geometric concepts in brown colour, while it is placed in the cluster for equation system concepts in blue colour. Similarly, other overlaps can be observed such as questions 38, 73, and 26. This indicates that the effectiveness of SKVMN in discovering latent concepts as well as discovering questions that relate to these latent concepts.

We can further verify the effectiveness of SKVMN in discovering latent concepts for questions using the question descriptions in Table 4. For example, in Figure 5.(b), the questions 13, 19, and 30 fall in the same cluster in pink (top right corner). Their provided descriptions are "*Equivalent Fractions*","*Multiplication Fractions*", and "*Ordering Fractions*", respectively, which are all relevant to fractions concepts. Similarly, the questions 1, 81, and 92 have the descriptions "*Area Trapezoid*", "*Area Rectangle*", and "*Rotations*", respectively. These questions fall in the same cluster in light blue (bottom right corner) because they are about geometric concepts, such as area functions and transformations.

Note that, SKVMN depends on identity vectors to aggregate questions with common concepts together. While the DKVMN depends on the attention vectors to perform this aggregation.

5.4 Evolution of Knowledge States

As previously discussed in Section 1, the knowledge states of a student may evolve over time, through learning from a sequence of exercises. In order to illustrate this evolution process, Figure 1 shows a student's knowledge states over a sequence of 50 exercises from the ASSISTments2009 dataset. At each time step, a knowledge state consists of the concept states of five concepts $\{c^1, \ldots, c^5\}$, which are stored in the value matrix of a key-value memory augmented with DKVMN or SKVMN. Figure 1.(a) shows this student's

knowledge states captured by DKVMN, while Figure 1.(b) shows this student's knowledge states captured by SKVMN.

In SKVMN, relevant questions are identified as shown in Table 4. Comparing Figure 1.(a) and Figure 1.(b), it can be visually noticed that SKVMN has smoother updates to the concept states in the value matrix than DKVMN. For example, considering the questions 23, 33 and 61, the student answered the first two questions incorrectly and the last one correctly, which result in a sudden update to the value of c^3 (i.e., the concept state of c^3) in the value matrix of DKVMN but a smoother update to the concept state of c^3 in the value matrix of SKVMN. Another example is the questions 70 and 88 that correlate to same latent concepts, the student answered the first one incorrectly and the second one correctly, which resulted in a significant update to the concept state of c^3 in the DKVMN's memory around indices 18, 19 and 20, while SKVMN's concept state of c^3 decreased in a smoother manner. This means that SKVMN considers the past performance of the student in relevance to this concept. At its core, these differences in capturing concept states are due to the fact that DKVMN's write process only takes the question and the answer to calculate the erase and add vectors, so that the knowledge state of DKVMN is biased to the most recently observed question. SKVNM has resolved this issue by taking into account the summary vector (i.e., current knowledge state and the level of difficulty of the current question) for the write process.

6 RELATED WORK

In this section, we provide a brief review of related research work.

One of the early attempts for developing a KT model was introduced by Corbett and Anderson [6]. Their KT model, called Bayesian Knowledge Tracing (BKT), assumed a knowledge state to be a binary random variable (i.e. know or do not know) and followed a Bayesian inference approach to estimate the values of knowledge states. However, BKT has limitations in modelling dynamics between different concepts due to its oversimplified representation to make the Bayesian inference tractable. Baker et al. [1] extended BKT by introducing an additional layer to the Bayesian inference to represent the contextual information. While their model achieved better results, it was still considered only the latest observation as a first-order Markov chain. Several attempts have been made to extend BKT by individualizing the prior distribution of Bayesian inference parameters [23, 35] so as to customize the model for each individual student. These individualization techniques were proved to reduce prediction errors of the original BKT model. Pardos and Heffernan [24] introduced the use of auxiliary information to the Bayesian inference process, such as item difficulty, and showed that it can further enhance the prediction accuracy.

With the rise of deep learning models [19] and their achieved breakthroughs in sequence modelling [29], such as natural language processing [10, 34], video recognition [7, 21], and signal processing [33], recent studies adopted deep learning models to address the KT problem. Piech et al. [26] proposed the deep knowledge tracing (DKT) model which uses a recurrent neural networks (RNN) [22] to model dynamics in a past exercise sequence and predicts answers for new questions. DKT resolved the limitations of Bayesian inference approaches as RNNs optimization through backpropagation is tractable. Despite this advance, DKT assumed only one hidden

state variable for representing a student's knowledge state, which is an unrealistic assumption for real-world scenarios as a student's knowledge can significantly vary across different learning concepts. To address this limitation, Zhang et al. [36] proposed a model called Dynamic Key-Value Memory Networks (DKVMN), which followed the concepts of Memory-Augmented Neural Networks (MANN) [12, 28]. MANN aim at mimicking the human's brain functionality which combines neural spiking for computation with memory for storing past experiences [8]. Inspired by MANN, DKVMN is augmented with two auxiliary memory structures: the key matrix and the value matrix. The former is used to keep the concepts underlying exercises, while the later one is used to store a knowledge state across these concepts. Results showed that DKVMN outperformed BKT and DKT on standard KT benchmarks, and therefore it is considered the state-of-the-art KT models. However, DKVMN only considers the latest exercise embedding when updating the value matrix, resulting in biased knowledge states that ignore past learning experience. As an example, if we have three related exercises in a sequence, two being answered correctly and the latest being answered incorrectly, DKVMN would be biased to the latest one and update the knowledge state with knowledge loss abruptly. In addition to this, DKVMN has no model capacity to capture long dependencies in an exercise sequence. This assumes a first-order Markov chain to represent a past exercise sequence, which is not satisfactory in many scenarios. Our proposed KT model has addressed the limitations from both DKT and DKVMN.

In our proposed KT model, we developed a modified LSTM, called Hop-LSTM, for sequence modelling. Current recurrent neural network models (RNNs) and their variants, such as LSTMs [13], bi-directional RNNs [30], or other gated RNNs [18], provide the capacity to effectively ingest dependencies in sequential data. However, when sequences are long, it is still difficult to capture long term dependencies. One way to alleviate this issue is to only update a fraction of hidden states based on the current hidden state and input [14]. For example, Yu, Lee and Le [34] proposed a LSTM model that can jump ahead in a sequence to avoid irrelevant words. The jump decision was controlled by a policy gradient reinforcement learning algorithm that works as an active learning technique to sample only important words for the model. Campos et al. [4] proposed a model by augmenting the standard RNN with a binary state update gate function which is responsible for deciding whether to update the hidden state or not based on the number of previous updates performed and a loss term that balances the number of updates (i.e. learning speed) with achieved accuracy. Different from these models, we developed Hop-LSTM in relation to a Triangular layer so that only the hidden states of LSTM cells for relevant exercises are connected. This allows our KT model to identify relevant skills and prior background from the past learning activities (e.g., exercise sequences) for improved prediction accuracy.

7 CONCLUSIONS

In this paper, we introduced a novel model called Sequential Key-Value Memory Networks (SKVMN) for knowledge tracing. SKVMN aimed at overcoming the limitations of the existing KT models. It is augmented with a key-value memory at the memory layer and a modified LSTM, called Hop-LSTM, at the sequence layer. The

experimental results showed that our proposed model outperformed the state-of-the-art models over all datasets. Future work will consider techniques to automatically tune hyper-parameters for Knowledge Tracing models.

ACKNOWLEDGMENTS

REFERENCES

[1] Ryan S. Baker, Albert T. Corbett, and Vincent Aleven. 2008. More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS)*. Berlin, Heidelberg, 406–415.

[2] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade: Second Edition*. Berlin, Heidelberg, 437–478.

[3] Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade: Second Edition*. Berlin, Heidelberg, 421–436.

[4] Víctor Campos, Brendan Jou, Xavier Giró i Nieto, Jordi Torres, and Shih-Fu Chang. 2018. Skip RNN: Learning to Skip State Updates in Recurrent Neural Networks. In *6th International Conference on Learning Representations, (ICLR), Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[5] Haw-Shiuan Chang, Hwai-Jung Hsu, and Kuan-Ta Chen. 2015. Modeling Exercise Relationships in E-Learning: A Unified Approach.. In *Proceedings of the 8th International Conference on Educational Data Mining, (EDM), Madrid, Spain, June 26-29, 2015*. 532–535.

[6] Albert T. Corbett and John R. Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* 4, 4 (01 Dec 1994), 253–278.

[7] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. In *The IEEE Conference on Computer Vision and Pattern Recognition, (CVPR) , Boston, MA, USA, June 7-12, 2015*. 2625–2634.

[8] Charles R Gallistel and Adam Philip King. 2011. *Memory and the computational brain: Why cognitive science will transform neuroscience*. Vol. 6. John Wiley & Sons.

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS). Society for Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy, 249–256.

[10] A. Graves, N. Jaitly, and A. Mohamed. 2013. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*. 273–278.

[11] A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vancouver, BC, Canada, 6645–6649.

[12] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538 (12 Oct 2016), 471.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780.

[14] Yacine Jernite, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Variable computation in recurrent neural networks. In *5th International Conference on Learning Representations, (ICLR), Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[15] Mohammad Khajah, Robert V Lindsey, and Michael C Mozer. 2016. How Deep is Knowledge Tracing?. In *Proceedings of the 9th International Conference on Educational Data Mining, (EDM), Raleigh, North Carolina, USA, June 29 - July 2, 2016*.

[16] Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *international conference on learning representations (ICLR)*.

[17] George J. Klir and Bo Yuan. 1995. Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[18] Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. 2018. FastGRNN: A Fast, Accurate, Stable and Tiny Kilobyte Sized Gated Recurrent Neural Network. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, (NeurIPS), 3-8 December 2018, Montréal, Canada*. 9017–9028.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[20] Charles X. Ling, Jin Huang, and Harry Zhang. 2003. AUC: A Statistically Consistent and More Discriminating Measure Than Accuracy. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*. San Francisco, CA, USA, 519–524.

[21] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. 2016. Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. In *14th European Conference on Computer Vision, (ECCV) , Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*. Cham, 816–833.

[22] Danilo P. Mandic and Jonathon Chambers. 2001. *Recurrent Neural Networks for Prediction: Learning Algorithms,Architectures and Stability*. John Wiley & Sons, Inc., New York, NY, USA.

[23] Zachary A. Pardos and Neil T. Heffernan. 2010. Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. In *Proceedings of the 18th International Conference on User Modeling, Adaptation, and Personalization (UMAP)*. Berlin, Heidelberg, 255–266.

[24] Zachary A. Pardos and Neil T. Heffernan. 2011. KT-IDEM: Introducing Item Difficulty to the Knowledge Tracing Model. In *Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization (UMAP)*. Berlin, Heidelberg, 243–254.

[25] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning (ICML)*. Atlanta, Georgia, USA, III–1310–III–1318.

[26] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. 2015. Deep Knowledge Tracing. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada (NeurIPS)*. Cambridge, MA, USA, 505–513.

[27] Christopher James Piech. 2016. *Uncovering Patterns in Student Work: Machine Learning to Understand Human Learning*. Ph.D. Dissertation. Stanford University.

[28] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with Memory-augmented Neural Networks. In *Proceedings of the 33nd International Conference on Machine Learning, (ICML), New York City, NY, USA, June 19-24, 2016*. 1842–1850.

[29] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Netw.* 61, C (Jan. 2015), 85–117.

[30] M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (Nov 1997), 2673–2681.

[31] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems, December 8-13 2014, Montreal, Quebec, Canada (NeurIPS)*. Cambridge, MA, USA, 3104–3112.

[32] Michael Villano. 1992. Probabilistic Student Models: Bayesian Belief Networks and Knowledge Space Theory. In *Proceedings of the Second International Conference on Intelligent Tutoring Systems (ITS)*. London, UK, UK, 491–498.

[33] P. Wang, A. Jiang, X. Liu, J. Shang, and L. Zhang. 2018. LSTM-Based EEG Classification in Motor Imagery Tasks. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26, 11 (Nov 2018), 2086–2095.

[34] Adams Wei Yu, Hongrae Lee, and Quoc V. Le. 2017. Learning to Skim Text. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, (ACL), Vancouver, Canada, July 30 - August 4, (Volume 1: Long Papers)*. 1880–1890.

[35] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. 2013. Individualized bayesian knowledge tracing models. In *Artificial Intelligence in Education - 16th International Conference, (AIED), Memphis, TN, USA, July 9-13, 2013. Proceedings*. 171–180.

[36] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. 2017. Dynamic Key-Value Memory Networks for Knowledge Tracing. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. Republic and Canton of Geneva, Switzerland, 765–774.