EasyChair Preprint

№ 345

Modeling Hint-Taking Behavior and Knowledge State of Students with Multi-Task Learning

Ritwick Chaudhry, Harvineet Singh, Pradeep Dogga and Shiv Kumar Saini

July 14, 2018

Modeling Hint-Taking Behavior and Knowledge State of Students with Multi-Task Learning

Ritwick Chaudhry*†
Indian Institute of Technology
Bombay
Mumbai, India
ritwickchaudhry@gmail.com

Harvineet Singh*
Adobe Research
Bengaluru, India
harvines@adobe.com

Pradeep Dogga†
Indian Institute of Technology
Kharagpur
Kharagpur, India
pradeepdogga@gmail.com

Shiv Kumar Saini
Adobe Research
Bengaluru, India
shsaini@adobe.com

ABSTRACT

Interactive learning environments facilitate learning by providing hints to fill the gaps in the understanding of a concept. Studies suggest that hints are not used optimally by learners. Either they are used unnecessarily or not used at all. It has been shown that learning outcomes can be improved by providing hints when needed. An effective hint-taking prediction model can be used by a learning environment to make adaptive decisions on whether to withhold or provide hints. Past work on student behavior modeling has focused extensively on the task of modeling a learner's state of knowledge over time, referred to as knowledge tracing. The other aspects of a learner's behavior such as tendency to use hints has garnered limited attention. Past knowledge tracing models either ignore the questions where a hint was taken or label hints taken as an incorrect response. We propose a multi-task memory-augmented deep learning model to jointly predict the hint-taking and the knowledge tracing task. The model incorporates the effect of past responses as well as hints taken on both the tasks. We apply the model on two datasets – ASSISTments 2009-10 skill builder dataset and Junyi Academy Math Practicing Log. The results show that deep learning models efficiently leverage the sequential information present in a learner's responses. The proposed model significantly out-performs the past work on hint prediction by at least 12% points. Moreover, we demonstrate that jointly modeling the two tasks improves performance consistently across the tasks and the datasets, albeit by a small amount.

1. INTRODUCTION

E-learning is changing knowledge creation and sharing in a profound way by bringing personalized learning experiences to a learner's device. Assessments in the form of quizzes or assignments form an important component of an e-learning software. A personalized e-learning environment identifies the gaps in understanding of a concept and effectively uses learning aids such as hints to fill these gaps. *Knowledge tracing* is the task of estimating a learner's state of knowledge over time with the goal of predicting the performance of the learner in future assessments. Knowledge tracing is used for deciding which question to ask in an adaptive learning environment. Current set of knowledge tracing models neither incorporate the effect of a learning aid on the level of understanding of a concept nor predict whether a learner is likely to use a learning aid.

A learning aid, common to many interactive learning environments, is the option to take a hint during an assessment [3]. However, the data shows that learners tend to use hints inappropriately. One problem is that of abusing hints [2]. They tend to spend less time on solving the assessment and opt for hint without attempting to solve the problem. Figure 1 shows the percentage of responses with correct answers, incorrect answers, and percent directly opted for hint by each question. The x-axis is sorted by the percent of correct responses for a question in increasing order. The data for this chart is from ASSISTments dataset [14] for 2009-2010.[1] As expected, % hint taken is negatively correlated with % correct. In other words, more learners tend to take hints on difficult questions. However, as Figure 2 shows, the hint takers tend to spend less time on a question than the learners who attempt the question, irrespective of whether the question is correctly or incorrectly answered. The research on this subject shows that the learners who attempt a question tend to have a higher probability of achieving proficiency in the subject [19]. Also, the learners who use hints very frequently tend to have the lowest learning rate [13]. Section 3 presents a review of the literature on hints as a learning aid. The literature shows that hints are an important learning aid but offering hints indiscriminately can lead to poor learning outcomes. A personalized e-learning

*These authors contributed equally
†Work done during an internship at Adobe Research

[1]The dataset is available at https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010.

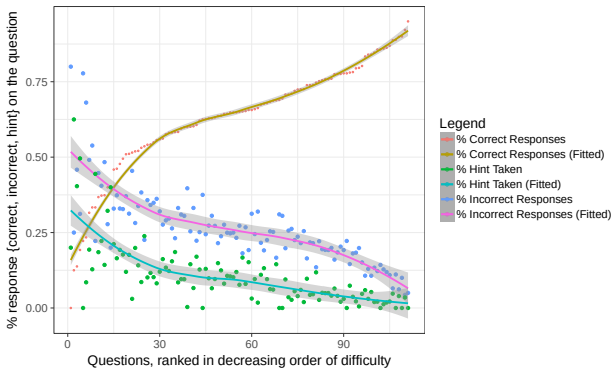


Figure 1: Percent of correct attempts, incorrect attempts, hints opted for each question in ASSISTments data. The questions are sorted by % correct responses.

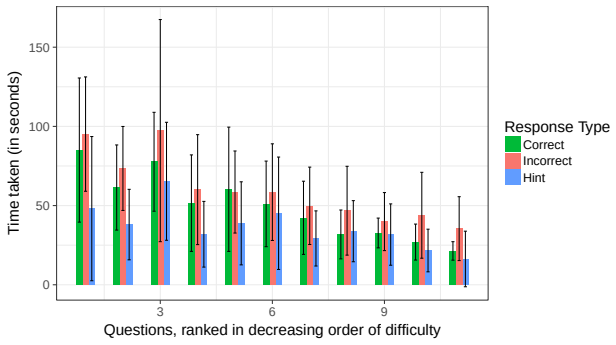


Figure 2: The box plot shows the distribution of time taken to attempt a question when response was correct (in green color), incorrect (in red), and when hint was taken (in blue). x-axis is sorted from lowest % correct on left to highest in right.

environment can use likelihood of taking a hint and the effect of taking a hint on learning to decide whether to show a hint. For example, the environment can proactively suggest hints to students who are stuck with a concept and have a low likelihood of taking a hint themselves.

Another reason to model the hint-taking behavior is to improve the performance of a knowledge tracing model. The existing knowledge tracing models do not model the hint-taking behavior. Section 2 presents the past work on knowledge tracing and hint-taking prediction. Traditional knowledge tracing models either tag a hint taken as an incorrect response or remove the data point where hints were taken. The two responses, i.e. attempting to solve a question and taking a hint directly, tend to result in different learning outcomes. Hence, conflating an incorrect response with a hint taken can deteriorate model performance. We show that explicitly modeling the hint-taking behavior improves performance of the model. Additionally, a higher propensity to take hints might be informative about the likelihood of answering questions correctly [19, 13]. Hence, throwing away the data points where hints were taken is akin to throwing away useful information. Conversely, knowledge tracing tasks contain information about whether a student is likely to take a hint. The synergies between the knowledge tracing and the hint-taking task motivates the application of a multi-task learning model [8]. Another important modeling consideration is the parameterization of the skill level. A knowledge tracing model is parameterized by deciding the level of heterogeneity in a learner's skill level and the question difficulty parameters. In the traditional knowledge tracing models, one might represent the skill level using one common parameter for all concepts or use a different parameter for each concept or a group of concepts clustered based on domain knowledge. Recently, deep learning based models have been used for knowledge tracing [23, 16, 34] which automatically capture the dependencies between different concepts based on the student response sequences. We extend the memory-augmented deep learning model proposed by Zhang *et al.* [34] to include hints taken in the past as an input and the prediction of hint-taking as an auxiliary task. We call this model `Colearn`. Section 4 describes the proposed model. Section 6 describes the evaluation methodology and estimation approach, including how the model hyperparameters are set.

The proposed model is compared with the baseline models from traditional approaches as well as deep learning based approaches. Section 7 describes the baseline models. We perform experiments on two popular datasets – ASSISTments 2009-2010 skill builder dataset and Junyi Academy Math Practicing Log. Section 5 describes the two datasets. Both the datasets contain information on whether a hint was taken. ASSISTments dataset contains the information whether a learner first attempted a question or directly took a hint. However, Junyi dataset contains noisy information on hints taken as it contains information on whether a hint was taken regardless of whether a hint was taken first or the question was attempted prior to it. The importance of this distinction is supported by past studies.

Results show that a memory-augmented deep learning model improves hint prediction performance from 79.10% to 91.12% on ASSISTments dataset and from 77.62% to 92.31%. `Colearn`, which is a multi-task memory-augmented deep learning model, further improves, by a small margin, the performance of the hint-taking prediction task by 0.63% and 0.03% point, respectively for the two datasets. Additionally, `Colearn` improves the performance on the knowledge tracing task for ASSISTment dataset by 0.25% point and for Junyi dataset by 0.18% points. Note that the baseline model for knowledge tracing is another memory-augmented deep learning model. Although the effect on performance is small, a benefit of the joint modeling of the two tasks is that we can work with only one model instead of two while training and scoring.

One of the criticisms of the deep learning based approaches is that the estimated parameters do not enhance our understanding of how the world works. We try to understand the meaning of the estimated parameters, especially the question embedding vectors, in Section 7.3. The analysis shows that a question embedding tends to capture question's difficulty.

In summary, the main contributions of this work are fourfold. First, we show a large improvement in the perfor-

mance of the hint-taking prediction task by using a memory-augmented deep learning model. Second, we motivate joint modeling of knowledge tracing and hint-taking prediction tasks which have been modeled separately in the prior work. Third, we extend a recent memory-augmented deep learning model for knowledge tracing to the task of hint-taking prediction. The proposed model, `Colearn`, incorporates the sequence of correct, incorrect response as well as hint-taking behavior on past questions as inputs. The model adds the hint-taking prediction as an auxiliary task. Fourth, we extensively evaluate the proposed model on two real-world datasets and show that our approach outperforms the competitive baselines on both the tasks.

2. RELATED WORK

This paper builds on the literature on knowledge tracing and on learning aids such as hints. Knowledge tracing in an interactive learning environment is an extensively studied area. Different approaches have been proposed in past.

Item Response Theory or IRT models the probability that a student answers a question correctly as a function of the following two parameters: one representing the student's skill level and the second representing the question difficulty [12]. The probability that a student answers a question correctly decreases with the question difficulty and increases with the student skill level, all else being equal. The student skill level and question difficulty are scalars which are estimated from data. Recent extensions to IRT, such as Hierarchical IRT, partition questions into groups, e.g. based on concepts covered, and model student skill level and item difficulty for each group separately [30]. However, these models do not use the information present in the sequence of responses. This results in incorrect responses followed by correct responses to be treated the same as the reverse sequence. Intuitively, a knowledge tracing model should put more weight on the performance on recent responses.

Bayesian Knowledge Tracing or BKT is another widely-used model. It uses information in the sequence of responses. BKT uses a Hidden Markov Model with the student skill as the latent variable and the responses as the observed variables [11]. One reason for the popularity of BKT is that, unlike IRT, it models student's skill in each concept separately. This information can be used by a learning system to personalize a learning activity. For example, a learning system can repeat a concept, switch to a new concept or skip a concept altogether based on the estimates of the skill level attained in the concepts.

Deep Learning based approaches have been employed due to the flexibility these approaches provide in modeling the skill of a student and the difficulty level of a question. Piech *et al.* [23] use Long Short-Term Memory (LSTM) cells to model sequence of student responses. They show significant improvement over BKT in predicting the student responses on many datasets. There has been concern voiced due to the lack of interpretability of the Deep Learning based approaches. Khajah *et al.* [16] show that DKT's performance can be matched by modifying BKT model. However, matching DKT's performance required significant domain knowledge on the processes involved in the learning process and insights from DKT model [16]. On the other hand, a Deep Learning based model performs well even without explicitly building a domain specific knowledge into the model. Memory-augmented neural networks, proposed for this task by Zhang *et al.* [34], provide even more flexibility to model student skill and question difficulty. A similar network architecture has been used for question-answering on free-form text documents [20].

Hints as a study help strategy has been extensively studied. The literature on how to provide hints has focused on whether to provide hints on-demand or proactively. Duong *et al.* [13] propose a model incorporating hint usage information in knowledge tracing. However, they do not use this information to predict the probability that a user will take a hint or not. Castro *et al.* [9] use a technique called *tabling method* to predict whether a student will attempt or take a hint in the next question. The model does not consider the complete sequence of student responses in the past and it is difficult to train for the longer sequences. This results in poor performance of the model.

In summary, there is rich literature on predicting the likelihood of a correct response and some recent work in predicting hint usage. However, the literature, to the best of our knowledge, has not modeled these two related problems jointly. Past work on *multi-task learning* (MTL) [8] suggests that adding an auxiliary task can help in improving the performance on both the tasks. MTL has shown considerable benefits in many domains including computer vision [21], natural language processing [17], health diagnostics [35], among others. Our proposed model includes effect of hints on future probability of answering a question correctly. This information can be used to decide when to provide a hint on a particular question.

Our Contribution: We extend the model proposed by Zhang *et al.* [34]. We include the hint usage information by changing the encoding of the inputs to the network. In addition, we add the components which share the network weights for the auxiliary task of predicting the probability of taking a hint. This results in increased prediction accuracy for the tasks of whether the learner will take a hint as well as whether a learner will answer a question accurately.

3. BACKGROUND

There is a large literature on hints as a learning aid that provides motivation for the joint modeling of item response and hint usage. The literature shows that hints are important but prone to misuse if provided indiscriminately. The research also shows that attempting a question and taking a hint directly have different implications for learning a concept.

Mathews *et al.* [19] shows that learners who first attempt to solve a question tend to learn by themselves and have higher probability to master the knowledge. This result has a basis in the theory that the process of attempting a question activates self-explanation, which is an important meta-cognitive skill [4, 10, 7, 22, 25, 29]. While hints are useful learning aid, the research on how hints are used show that easy access to hints may lead to sub-optimal outcomes. In studies

of help-seeking from human tutors, it has been found that those who need help the most are the least likely to ask for it [15, 24, 26]. Computer-based help systems can potentially improve the use of help [32]. Given that many learning environments provide some form of on-demand help, it might seem that effective use of help would be an important factor influencing the learning results obtained with these systems. However, there is evidence that learners are not using the help facilities offered by learning environments effectively [3]. They often ignore the help facilities or use them in ways that are not likely to help learning. They frequently use the system's on-demand hints to get answers, without trying to understand how the answers are derived or the reasons behind the answers [1]. It is shown that the learners who opt for hints very frequently tend to have the lowest learning rate [13]. On the other hand, there is also evidence that, when used appropriately, on-demand help in an interactive learning environment can have a positive impact on performance [1, 5] and learning [27, 31, 32]. Also, providing tutoring with respect to student's help-seeking behavior helps them to become better help seekers and thus better future learners [6]. A request for help is appropriate when a student is stuck while solving a tutor problem but not when she has not yet thought about the problem. Further, students should carefully read and interpret the help given by the system. Aleven *et al.* [2] described a model of help-seeking behavior within a cognitive tutor. The authors have created a taxonomy of errors in student's help-seeking behavior. Based on the frequency of the meta-cognitive bugs defined by their model, it was observed that 36% of the actions taken by students were classified as help abuse bugs and 19% of the actions as help avoidance. To make a better tutoring system which can guide the students in regulating their help-seeking behavior, it is essential to incorporate the effect of hints in knowledge tracing. Traditional knowledge tracing models do not take the hint usage into account.

3.1 Notations

Next, we introduce notations for the joint model. Let the interactions of a learner till time T are denoted by $X = (x_1, x_2, x_3, \ldots, x_T)$. Here, each interaction x_t is an encoding representing the tuple (q_t, r_t, h_t) containing an identifier for the question attempted q_t, a binary indicator r_t, encoding the response, and another binary indicator h_t, encoding hint usage. The hint usage variable is positive only if the hint was taken directly instead of attempting the question first. Let $Q = \{q_t\}_t$ be the set of distinct questions. The interaction tuple can contain additional information collected such as time taken to attempt, type of question, concepts involved in the question and so on. The task of a knowledge tracing model is to predict the probability of correctly answering a question $q_{t'} \in Q, t' > T$, i.e. $\text{Prob}(r_{t'} = 1 | q_{t'}, X)$. And, the task of predicting a hint usage model is to estimate $\text{Prob}(h_{t'} = 1 | q_{t'}, X)$. Both of these tasks are supervised learning problems and can be modeled using a binary classifier. Instead of building two separate models for these tasks, we model them jointly within a deep learning based classification framework.

4. MODEL

Zhang *et al.* [34] proposed a memory-augmented neural network model, called Dynamic Key-Value Memory Networks or `DKVMN`, for knowledge tracing. This model performed better than the baseline models on three real-world datasets. This model is used as a baseline for the proposed multi-task model due its many favorable properties. It does not require extensive feature engineering or metadata information such as mapping of items to skills and the model offers flexibility in adding more tasks as well as inputs. We first give a brief description of their model, followed by our modifications. Reader is referred to Zhang *et al.* [34] for further implementation details regarding the original model.

4.1 Dynamic Key-Value Memory Networks, `DKVMN`

The neural network is designed to store the knowledge state of a learner based on past interactions. This is done using a memory component which works like a key-value store. Each attempted question is mapped to a set of concepts which are the keys in the memory component. The corresponding values are a learner's knowledge state in each of these concepts. The network has a mechanism to update the states because of learner's response to the question. The key-value pairs are modeled using vectors instead of scalars for more representational flexibility. So, for each question the output from the memory component gives a learner's knowledge state. This is compared with the difficulty level of the question, which is the output of another component, to arrive at probability of correctly answering the question. All operations are implemented using differentiable operators like multiplication, addition, sigmoid function on matrices so that the network can be trained end-to-end using gradient descent optimization techniques.

4.2 Proposed Model, `Colearn`

The `DKVMN` model does not consider the effect of taking hints during assessment. It considers hint usage as an incorrect attempt by the learner, as is the standard approach in existing models. However, the update in knowledge state of a learner is different when a question is attempted as opposed to when a hint is taken without any attempt. We modify `DKVMN` to incorporate hint information by changing the input and output layers of the model. Figure 3 shows the modified network. Next, we describe the components of `DKVMN` and our modifications to it.

4.2.1 Input Layer

In the update phase of the model, instead of using one-hot encoding of (q_t, r_t), we encode (q_t, r_t, h_t) into a vector of length $2|Q| + 1$, where Q is the set of distinct questions. The first $|Q|$ dimensions are a one-hot vector representing the correct attempt on the question, i.e. in case of a correct attempt, the vector has 1 at the index of the question and has 0 everywhere else. Similarly, the next $|Q|$ dimensions encode incorrect attempt. The last dimension of the vector is a binary value indicating whether a hint is taken. This input encoding changes the way the value vectors in the memory component are changed due to the information whether a hint is used or not is also present. An example of the input encoding is illustrated in Table 1 where there is a total of two exercises.

We tried different ways of representing the three outcomes, *viz.* correct response, incorrect response, and hint taken. These included one-hot encoding with all three outcomes

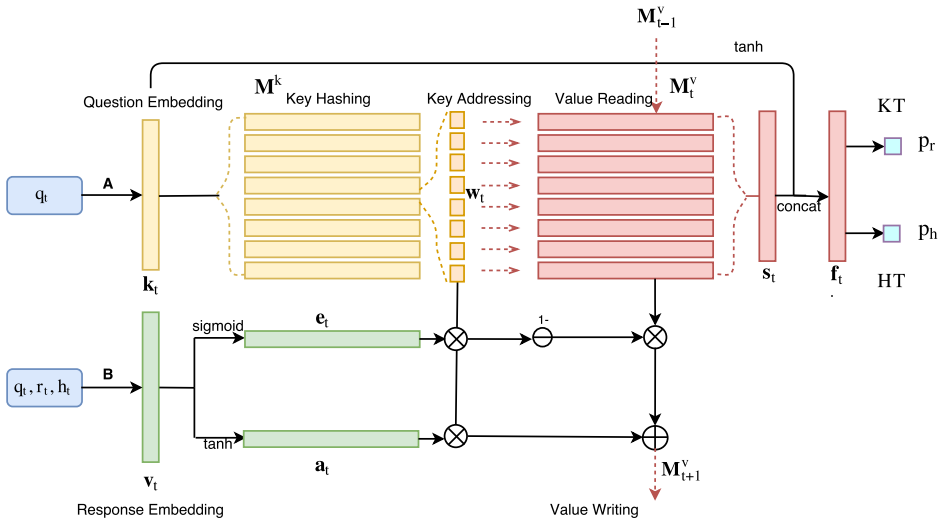


Figure 3: Architecture of the neural network for joint modelling of knowledge state and hint use. KT and HT refer to Knowledge tracing and Hint-Taking tasks.

Response	Encoding	
	DKVMN	Colearn
Q2-Correct	$(0, 1, 0, 0)$	$(0, 1, 0, 0, 0)$
Q2-Incorrect	$(0, 0, 0, 1)$	$(0, 0, 0, 1, 0)$
Q2-Hint	–	$(0, 0, 0, 0, 1)$
Q1-Hint	–	$(0, 0, 0, 0, 1)$

Table 1: Response encoding in case of two exercise tags

with a length of $3|Q|$. The chosen encoding gave the best results in the experiments. This encoding represents response on two different questions where hints are taken with the same vector (see example in Table 1). Since the network already incorporates index of the current question as a separate input, using $|Q|$ extra dimensions for hint encoding in update phase adds more parameters which are not required.

4.3 Key-Value Store

Key-value memory networks, introduced in [20], have an explicit memory component which is an array of pairs of memory slots where each slot is a real-valued vector. Given a query, the relevant information is fetched from the slots using an attention-based mechanism depending on which slots are relevant for that query. The mechanism has three major components which are described next.

- **Key Hashing:** The *key* part of the pairs holds the static information representing the various hidden concepts using vectors. Each of the key vectors $(\mathbf{M}^k(1), \ldots, \mathbf{M}^k(n))$ represents a concept.

- **Key Addressing:** Given the t^{th} question answered by a student, the relevance of each concept in that question is found out using an attention mechanism. Each question is first converted into an embedding

$$\mathbf{k}_t = \mathbf{A}\mathbf{q}_t \tag{1}$$

and the weight of each concept c_i in q_t is given by

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

where $\mathbf{A}$ is the question embedding matrix, $\mathbf{q}_t$ denotes the one-hot encoded question, $\mathbf{M}^k(i)$ denotes the key vector of the i^{th} concept and $\text{Softmax}(x_i) = e^{x_i} / \sum_j e^{x_j}$. The question embedding vector $\mathbf{k}_t$ obtained from matrix $\mathbf{A}$, the key matrix $\mathbf{M}^k$ are shown in yellow color and attention weight vector $\mathbf{w}_t = (w_t(1), \ldots, w_t(n))$ is shown in orange in Figure 3.

- **Value Reading:** Given the weight $w_t(i)$ of each concept c_i in question q_t given by Equation 2, the student's skill in that question is calculated as the weighted sum of the knowledge in each of the concepts, as taken from value matrix $\mathbf{M}^v_t$. The value matrix is shown in pink color in Figure 3. The student's skill in the question q_t is returned as

$$\mathbf{s}_t = \sum_{i=1}^{n} \mathbf{M}^v_t(i) * w_t(i) \tag{3}$$

This skill is then used to make predictions about the student's response correctness and hint usage.

- **Value Writing:** Once we get student's actual response to the question, knowledge state is updated. This part is shown in green color in Figure 3. The update in each of the concept c_i's value vectors are also weighted according to the calculated weight $w_t(i)$ of the concept (2). The student's response is encoded in a vector, $\mathbf{x}_t$ of size $2|Q|+1$ to represent a correct attempt or an incorrect attempt or a hint taken.

$$\mathbf{x}_t = \text{encoded tuple}(q_t, r_t, h_t)$$

This encoding, described in 4.2, is then converted into an embedding $\mathbf{v}_t$, given by

$$\mathbf{v}_t = \mathbf{B}\mathbf{x}_t$$

where $\mathbf{B}$ is the response embedding matrix. When updating the student's knowledge state, the memory is erased

first before new information is added.
The erase vector $\mathbf{e}_t$ is calculated as

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

where $\mathbf{E}$ is a linear transformation matrix, $\mathbf{b}_e$ is the bias and $\text{Sigmoid}(x_i) = 1/(1+e^{x_i})$.
The addition vector $\mathbf{a}_t$ is calculated as

$$\mathbf{a}_t = \text{Tanh}(\mathbf{D}^T \mathbf{v}_t + \mathbf{b}_a)$$

where $\mathbf{D}$ is a linear transformation matrix, $\mathbf{b}_a$ is the bias and $\text{Tanh}(x_i) = (e^{x_i} - e^{-x_i})/(e^{x_i} + e^{-x_i})$.
After the t^{th} response, the value matrix is updated as

$$\mathbf{M}_t^v(i) = \mathbf{M}_{t-1}^v(i) \odot [1 - w_t(i)\mathbf{e}_t] + w_t(i)\mathbf{a}_t$$

Thus, the model adds and forgets student knowledge in concepts as more and more assessments are attempted.

4.4 Final Predictions

The final predictions for both, correct attempt and hint-taking, probabilities are calculated by applying two separate linear transformations followed by a sigmoid activation on $\mathbf{f}_t$ which is given by

$$\mathbf{f}_t = \text{Tanh}(\mathbf{W}_f^T * (\mathbf{s}_t || \mathbf{k}_t) + \mathbf{b}_f) \quad (4)$$

Here, $\mathbf{W}_f$ is a linear transformation, $\mathbf{s}_t$ is the final read knowledge state of the student in question $\mathbf{q}_t$ illustrated earlier in Equation 3, $\mathbf{k}_t$ is the question embedding in Equation 1, $\mathbf{b}_f$ is the bias and $||$ is the concatenation operator. The final probabilities for a correct-attempt and hint-taking are

$$p_r^{\text{pred}} = \text{Sigmoid}(\mathbf{W}_r^T * \mathbf{f}_t + \mathbf{b}_p^r) \quad (5)$$

$$p_h^{\text{pred}} = \text{Sigmoid}(\mathbf{W}_h^T * \mathbf{f}_t + \mathbf{b}_p^h) \quad (6)$$

where both $\mathbf{W}_r^T$, $\mathbf{W}_h^T$ are linear transformations, and $\mathbf{b}_p^r$, $\mathbf{b}_p^h$ are bias vectors.

4.4.1 Prediction Loss at Output Layer:

The output layer of `DKVMN` predicts the probability whether a question will be answered correctly. For the task of predicting whether a hint will be taken in the question, the factors like the knowledge state of the learner, the difficulty level of the question and past hint-taking behavior are important. Since the first two are already being modeled by `DKVMN`, we learn both the tasks simultaneously by using a multi-task learning approach. As shown in Equation 6, the final output layer of `Colearn` adds a linear transformation of $\mathbf{f}_t$ followed by a sigmoid activation to predict the hint-taking task. The loss is given by taking a weighted sum of losses from knowledge tracing and hint-taking prediction and is evaluated as

$$\mathcal{L} = \alpha_1 \text{cross\_entropy}(p_r^{\text{act}}, p_r^{\text{pred}}) + \alpha_2 \text{cross\_entropy}(p_h^{\text{act}}, p_h^{\text{pred}})$$

where p_r^{pred} is given in Equation 5 and p_h^{pred} in Equation 6 are the probabilities predicted at the output layer. The actual values p_r^{act} and p_h^{act} are 0 or 1 depending on the observed response. The cross entropy function

$$\text{cross\_entropy}(p^{\text{act}}, p^{\text{pred}}) = p^{\text{act}} \log(p^{\text{pred}}) + (1 - p^{\text{act}}) \log(1 - p^{\text{pred}})$$

We set both $\alpha_1 = \alpha_2 = 1$ to give equal weight to the knowledge tracing and hint-taking prediction tasks. This loss is backpropagated to update the network weights. When a learner takes a hint, only the loss of the hint-taking prediction is propagated. In other words, the loss for the knowledge tracing task is 0 in this case. The network weights, except the final output layer, are shared between the two tasks (See Figure 3). Multi-task learning acts as a regularizer for learning network weights as with the same set of weights the network should maximize two objectives. It also encourages sharing of knowledge across tasks through sharing of network weights. Experimental results demonstrate that the network trained using multi-task learning marginally outperforms current state-of-the-art models on both the tasks.

5. DATASETS

To evaluate the performance of the model we used the following two datasets:

- **ASSISTments 2009-2010 skill builder dataset**[2]: ASSISTments [14] is an online tutoring system which can be used by teachers for grade school-level Mathematics instruction and evaluation. The system can be used to identify common wrong answers and see student-reports for assignments in a class. The dataset contains activity logs of students solving exercises on the system and it is widely-used as a benchmark dataset for knowledge tracing [23, 34]. Log data includes information such as student responses, time spent on exercise, chronological order of attempts, if a hint is taken, tagged skill for an exercise. We use the updated version of this dataset. It corrects an issue, identified by Xiong *et al.* [33], with duplicated rows in the original version. We use the skill tag corresponding to an exercise as its identifier in the input to the models. Thus, the set of distinct questions, Q, is same as the set of distinct skill tags in the dataset. All rows with an empty skill tag are removed. Some rows contain invalid values in the column specifying student's first action i.e. values other than the permissible ones – {attempt, hint}. These transactions are removed. In case a student has multiple actions on the same exercise, we know whether the first action was a correct attempt, an incorrect attempt or a hint request. For the hint-taking prediction task, only the rows with the first action as a hint request are taken as a positive label.

- **Junyi Academy Math Practicing Log**[3]: Junyi Academy[4] is an e-learning platform, like Khan Academy, where students can practice exercises on various subjects including Mathematics, Biology, Computer Science. Like ASSISTments, the dataset contains attempt, hint taken, time spent, and skill tag information for an exercise. It has transactions for around 200,000 students. To the best of our knowledge, it is one of the largest student interaction datasets. As part of the data cleaning process, rows which contained non-binary values in the columns specifying whether hint was used or not and whether question

[2]ASSISTments 2009-2010 skill builder dataset is available at `https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010`
[3]Junyi Academy Math Practicing Log is available at `datashop.web.cmu.edu/DatasetInfo?datasetId=1198`
[4]`https://www.junyiacademy.org/`

was answered correctly or not were removed. Students with only one transaction in the dataset are removed. If a student requests a hint as one of the actions on a particular exercise, we do not know whether the hint was requested as the first action or it was requested after one or more incorrect attempts. In other words, we only know whether a hint request was one of the actions performed by the student. Therefore, for the hint-taking prediction task, all transactions which contain a hint request, irrespective of being the first action or not, are assigned the positive label. Note that this adds noise to the hint-taking label for this dataset.

The statistics comparing the two datasets are provided in Table 2.

Statistic	Datasets	
	ASSISTments	Junyi
# of Students	4,151	199,549
# of Exercise/Skill Tags	111	722
# of Concept Tags	–	40
# of Records	325,637	25,628,935
% of Attempts (Both Correct and Incorrect)	92.78%	93.56%
% of Hints	7.22%	6.44%

Table 2: Aggregate statistics from the two datasets

For extracting labels for the prediction tasks, it is assumed that a question is attempted only once. If a hint is taken first then the response is labeled as hint-taken. Else, the response is marked as correct or incorrect based on the outcome. So, if there are instances where multiple responses for a question are observed, we keep the first response on each question and remove subsequent responses. This is done to conform with the standard practice followed while evaluating knowledge tracing models. However, responses to subsequent attempts can also be incorporated in our setup.

6. EVALUATION METHODOLOGY

In each dataset, students and the corresponding transactions are randomly split into two parts – 80% for training and 20% for testing. Training set is further split, out of which 80% (i.e. 64% of total) is used for training the models. The rest 20% (i.e. 16% of total), called validation set, is used to tune hyperparameters of the models. Trained models with different values of hyperparameters are evaluated on the validation set in order to select the best hyperparameters.

6.1 Accuracy Metric

Both the prediction tasks are considered in a classification setting — answering a question correctly or not and taking a hint on a question or not. Hence, we compare the model performance based on Area under ROC curve (AUC) which is a standard classification metric. For knowledge tracing task, we follow the same evaluation procedure as followed by [23, 30, 34]. The model is trained using transactions from the training set. During the testing phase, the model is updated after each question response from the testing set. The updated model is used to perform the prediction for the next question.

6.2 Hyperparameter Tuning

Hyperparameters are learned using the validation set. We used Bayesian Optimization [28] to tune the hyperparameters for `Colearn` model. The model required several hyperparameters which cannot be set by hand easily. The method uses Bayesian techniques instead of gradient-based techniques to optimize the unknown function from the hyperparameter space to validation loss. The objective is to find the set of hyperparameter values which minimizes the validation loss while evaluating the model for only a small number of hyperparameter combinations. The tuned hyperparameters are:

Number of value vectors: Since the number of value vectors represent the number of 'hidden' concepts, this cannot be set by hand. The values were varied from 5 to 50 vectors.

Key vector size: The size of each key vector depends on efficient representation of the difficulty of questions and their similarity to the hidden concepts. The size was varied from 10 to 200.

Value vector size: The value vectors are a representation of the different concepts and an efficient representation depends on the size of these vectors. The size was varied from 10 to 200.

Hyper-parameters obtained for `Colearn` model are as follows – number of value vectors are 20 and 5 for ASSISTments and Junyi respectively, key vector size (i.e. question embedding size) is 50 for both, value vector size (i.e. question-attempt embedding size) is 200 and 100 for ASSISTments and Junyi respectively.

6.3 Training details

Stochastic gradient descent with momentum and norm-clipping was employed to train the weights of the network. The momentum was set to be 0.9 throughout the training and the norm was clipped to a threshold of 50.0. The learning rate was initialized as $5*10^{-2}$ and annealed after every 20 epochs till the learning rate reached 10^{-5}. Since the sequences of responses varied in length, the sequence length was fixed to 200 and 500 in ASSISTments and Junyi, respectively, with appropriate truncation or padding. Batch size for stochastic gradient descent is fixed to 32 and number of epochs is set to 100. Network weights corresponding to the epoch with least validation loss are taken for testing.

After training, learned weight values for the key and value matrices are saved and loaded at beginning of testing each student sequence. Key matrix is kept unchanged throughout the sequence, whereas the value matrix is updated independently for each student sequence as more actions are observed.

To check for robustness to initialization of network weights, we perform training 5 times with different random seeds (to get $\{\text{AUC}_i\}_{i=1}^{5}$). We report the average (i.e. $\overline{\text{AUC}} = \frac{1}{5}\sum_{i=1}^{5}\text{AUC}_i$) and standard deviation (i.e. $[\frac{1}{5}\sum_{i=1}^{5}(\text{AUC}_i - \overline{\text{AUC}})^2]^{\frac{1}{2}}$) of test AUC values across the 5 models.

7. RESULTS AND DISCUSSION

Model	Datasets	
	ASSISTments	Junyi
Colearn	**91.75** ± 0.07%	**92.34** ± 0.009%
DKVMN–hints	91.12 ± 0.06%	92.31 ± 0.01%
HH (n=3)	77.69%	76.66%
HH (n=4)	79.10%	77.62%

Table 3: **Hint-taking Prediction task**. Performance (AUC values) of proposed approach (Colearn) compared with the baselines on two datasets.

Model	Datasets	
	ASSISTments	Junyi
Colearn	**81.48** ± 0.04%	**80.56** ± 0.009%
DKVMN	81.23 ± 0.02%	80.38 ± 0.007%
HIRT	77.40%	79.45%
IRT	76.51%	77.46%

Table 4: **Knowledge Tracing task**. Performance (AUC values) of proposed approach (Colearn) compared with the baselines on two datasets.

To the best of our knowledge, no prior work models both of the prediction tasks jointly. Therefore, we report comparisons with prior work for each task separately. The Colearn results reported are for the model jointly trained on both the tasks.

7.1 Hint-taking Prediction

7.1.1 Baselines

Castro *et al.* [9] proposed a method called Hint-History model (HH) for predicting student actions on next question i.e. whether student will take a hint or attempt the next question. The method considers the sequence of n most recent student actions for predicting action on the next question. They use a technique called *tabling method* which counts the number of times a sequence resulted in a particular action in the training set. For instance, while making a prediction for a student who has taken two hints in a row followed by an attempt, the method finds students with same action sequence in the training set and uses the next-action probability for them as the predicted value in current case i.e. calculate number of times students with this action sequence took hint on the next question divided by total number of such students in the training dataset. These simple approaches have been used for knowledge tracing tasks [13] as well.

The tabling method is compared with two approaches that are proposed in this paper. The first one is using DKVMN [34] model with class labels being hint-taking indicators instead of question correctness (referred to as DKVMN–hints). The second one is Colearn.

7.1.2 Results

Table 3 summarizes the results. We compare with HH model for two different values of length of action sequences, $n = 3, 4$. DKVMN–hints shows 12% points improvement in AUC on ASSISTments dataset and 15% on Junyi datset. Colearn further improves the AUC on the two datasets. A memory-augmented deep learning model considers longer term dependencies in student sequences instead of taking a fixed-length history, as is the case with HH. It can also effectively model student-specific variations from individual sequences whereas HH model output is based only on population-level statistics. Lastly, multi-task training, Colearn model, also helps to increase performance on the task by a small margin due to the synergies across the tasks.

7.2 Knowledge Tracing

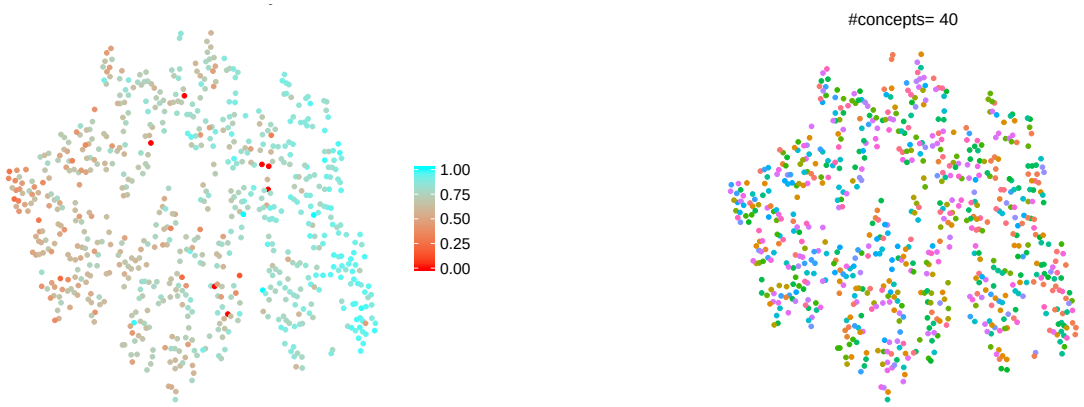
7.2.1 Baselines

We compare our model with three competitive baselines namely DKVMN [34], IRT [30] and Hierarchical IRT (HIRT) [30]. In IRT, student skill level and item difficulty are modeled separately and probability of answering correctly is taken as a pre-determined function of these two quantities such as sigmoid or logistic. In HIRT, related items are grouped together (e.g. those belonging to same concept) and the difficulty of each item is distributed normally around a per-group mean, which is distributed normally around a hyper-prior. DKVMN model was shown to outperform BKT [11] and DKT [16], hence we do not compare with those models. For DKVMN, best performing hyperparameters reported in [34] were taken. Note that the best-reported AUC of DKVMN (81.57%) on ASSISTments dataset differs from what we report for their model (81.23%), for the same hyperparameters. This results from different train-test set proportions, i.e. 20% sequences in test as compared to 30% used by Zhang *et al.* We could replicate DKVMN results using code published by the authors[5] on the dataset split provided by them. For IRT and HIRT models we use the code published by the authors[6]. For the baselines, the transactions where hints are taken are labelled as incorrect responses. This is the same approach followed in the baseline publications.

7.2.2 Results

The AUC values for the different methods on both datasets for knowledge tracing are shown in Table 4. The AUC value for deep learning models is sensitive to the initial values of network weights. Hence, we report average and standard deviation (separated by ±) of the AUC from five, randomly initialized, models. Colearn improves test set AUC on ASSISTments dataset by 4% points and on Junyi by 1% points as compared to HIRT method. The improvement due to multi-task model is consistent across datasets and tasks, albeit small. This means that students' past hint taking behaviour is not predictive of question correctness. Factors such as difficulty of the question and correctness on past attempts mostly can explain their future performance. Interestingly, performance increase is less in case of Junyi dataset than ASSISTments dataset in both the tasks. As discussed earlier, the way hint information is available in Junyi dataset adds some noise to the training signals. In cases where student takes a hint, we do not know whether hint was the first action before any attempt or was taken after making incorrect attempt(s). This might be the reason why we get relatively less advantage from incorporating hint information in Junyi dataset.

[5] https://github.com/jennyzhang0215/DKVMN
[6] https://github.com/Knewton/edm2016



(a) Exercise Difficulty

(b) Exercise Concept Mapping

Figure 4: t-SNE visualizations of question representation for Junyi dataset. Color denotes difficulty (in (a)) and concepts (in (b)) of the questions.

7.3 Discussion on Learned Representations

We have shown that the `Colearn` model performs better than the baseline models. In this section we explore the meaning of the estimated parameters. Specifically, how can we use the estimated parameters to represent a question and what does the representation represent?

To get representation for each question, q_t, we use a question's attention weights over the concepts in the key matrix. Each question is represented by a vector of length equal to the number of latent concepts where the value corresponding to each latent concept in the vector is given by Equation 2. This representation is obtained assuming that a student has not yet started to answer any question. Recall that, before the start of an assessment, the value matrix is set to the initial value matrix, $\mathbf{M}_0^v$. This initial matrix is part of the parameter set and it is estimated. The question representation is a vector that is based on the performance of all students, questions, and responses in the training set but not specific to any one student.

To understand how the question representations are related to each other, we visualize them using t-SNE [18]. Figure 4a and Figure 4b present the t-SNE visualizations of the question representations of the exercise tags in Junyi dataset. ASSISTments dataset is not used for this analysis because it does not contain the concepts for the exercise tag. Each dot in the scatter diagram represents a single exercise tag. The only difference between the two panels is the color used to represent each tag. In Figure 4a each exercise tag is colored according to the difficulty level of the question, with blue color representing the easiest and red color representing the most difficult exercise tags. The difficulty level is estimated using the fraction of correct responses in each question tag. The color of a dot in Figure 4b represents the concept of the exercise tag. There are 40 concepts for 722 exercise tags in Junyi dataset which include concepts like *fractions*, *algebra*, *trigonometry*.

One of the hypothesis is that the question representation captures the concept map [34]. If this was the case then the exercise tags within a concept should be close in the question representation space. However, Figure 4b shows that the exercise tags within a concept do not cluster together. In fact, the exercise tags seem to be randomly scattered in the question representation space. On the other hand the color of the exercise tags in Figure 4a shows a definite pattern with the easiest question tags towards the left and the most difficult ones towards the right. This shows that the question representation vectors tend to capture the difficulty level of an exercise tag. Note that, the question representation vector might capture other aspects such as prerequisite map. However, a complete in-depth analysis is out of the scope of this paper and left for future explorations.

8. CONCLUSION

Assessments (specifically, formative ones) are an important part of an interactive learning system as they help learners to gauge their progress. If a learner is stuck at a particular question, many learning platforms provide learning aids in the form of hints. Predicting when to provide an option of taking an hint is essential to regulating its excessive use or to avoid underuse. The probability of taking a hint relates to modeling the knowledge state of a learner during an assessment, which has been studied separately as knowledge tracing. Hence, we jointly modeled the hint-taking prediction task along with the knowledge tracing task. Through experiments we showed that our approach outperforms the baseline hint-taking prediction models and marginally improve on baseline knowledge tracing models. The approach proposed in the paper can be easily extended to incorporate other types of learning aids such as interactive tutorials, links to reading material and videos.

Better knowledge tracing and hint-taking models allow an e-learning system to make decisions such as number of questions to ask, the sequence of questions and whether to show a hint based on learner's proficiency. Such decisions affect the long-term learning outcomes. Future work involves integrating the predictions for the two tasks to develop strategies for optimizing long-term learning outcomes. High accuracy on both the tasks, as demonstrated, will allow to build student simulators for evaluating such strategies.

9. REFERENCES

[1] V. Aleven and K. R. Koedinger. Limitations of student control: Do students know when they need help? In *Intelligent tutoring systems*, volume 1839, pages 292–303. Springer, 2000.

[2] V. Aleven, B. Mclaren, I. Roll, and K. Koedinger. Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 16(2):101–128, 2006.

[3] V. Aleven, E. Stahl, S. Schworm, F. Fischer, and R. Wallace. Help seeking and help design in interactive learning environments. *Review of educational research*, 73(3):277–320, 2003.

[4] V. A. Aleven and K. R. Koedinger. An effective metacognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive science*, 26(2):147–179, 2002.

[5] T. Bartholomé, E. Stahl, S. Pieschl, and R. Bromme. What matters in help-seeking? a study of help effectiveness and learner-related factors. *Computers in Human Behavior*, 22(1):113–129, 2006.

[6] J. D. Bransford and D. L. Schwartz. Chapter 3: Rethinking transfer: A simple proposal with multiple implications. *Review of research in education*, 24(1):61–100, 1999.

[7] A. Bunt, C. Conati, and K. Muldner. Scaffolding self-explanation to improve learning in exploratory learning environments. In *Intelligent Tutoring Systems*, pages 109–156. Springer, 2004.

[8] R. Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, July 1997.

[9] F. E. V. Castro, S. Adjei, T. Colombo, and N. Heffernan. Building models to predict hint-or-attempt actions of students. *International Educational Data Mining Society*, 2015.

[10] C. Conati and K. Vanlehn. Toward computer-based support of meta-cognitive skills: A computational framework to coach self-explanation. *International Journal of Artificial Intelligence in Education (IJAIED)*, 11:389–415, 2000.

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

[12] F. Drasgow and C. L. Hulin. Item response theory. *Handbook of industrial and organizational psychology*, 1:577–636, 1990.

[13] H. Duong, L. Zhu, Y. Wang, and N. T. Heffernan. A prediction model that uses the sequence of attempts and hints to better predict knowledge: "better to attempt the problem first, rather than ask for a hint". In *EDM*, 2013.

[14] 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.

[15] S. A. Karabenick and J. R. Knapp. Help seeking and the need for academic assistance. *Journal of educational psychology*, 80(3):406, 1988.

[16] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*, 2016.

[17] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *HLT-NAACL*, pages 912–921, 2015.

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

[19] M. Mathews and T. Mitrović. *How Does Students' Help-Seeking Behaviour Affect Learning?*, pages 363–372. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[20] A. H. Miller, A. Fisch, J. Dodge, A. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1400–1409, 2016.

[21] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[22] A. Mitrovic. Self-explanation in a data normalization tutor. 2003.

[23] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 505–513, Cambridge, MA, USA, 2015. MIT Press.

[24] M. Puustinen. Help-seeking behavior in a problem-solving situation: Development of self-regulation. *European Journal of Psychology of education*, 13(2):271–282, 1998.

[25] A. Renkl. Worked-out examples: Instructional explanations support learning by self-explanations. *Learning and instruction*, 12(5):529–556, 2002.

[26] A. M. Ryan, M. H. Gheen, and C. Midgley. Why do some students avoid asking for help? an examination of the interplay among students' academic efficacy, teachers' social–emotional role, and the classroom goal structure. *Journal of educational psychology*, 90(3):528, 1998.

[27] S. Schworm and A. Renkl. Learning by solved example problems: Instructional explanations reduce self-explanation activity. In *Proceedings of the Cognitive Science Society*, volume 24, 2002.

[28] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[29] J. G. Trafton and S. B. Trickett. Note-taking for self-explanation and problem solving. *Human-computer interaction*, 16(1):1–38, 2001.

[30] K. H. Wilson, Y. Karklin, B. Han, and C. Ekanadham. Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation. *arXiv preprint arXiv:1604.02336*, 2016.

[31] D. Wood. Scaffolding, contingent tutoring, and

computer-supported learning. *International Journal of Artificial Intelligence in Education*, 12(3):280–293, 2001.

[32] H. Wood and D. Wood. Help seeking, learning and contingent tutoring. *Computers & Education*, 33(2):153–169, 1999.

[33] X. Xiong, S. Zhao, E. Van Inwegen, and J. Beck. Going deeper with deep knowledge tracing. In *EDM*, pages 545–550, 2016.

[34] J. Zhang, X. Shi, I. King, and D.-Y. Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web*, pages 765–774. International World Wide Web Conferences Steering Committee, 2017.

[35] J. Zhou, J. Liu, V. A. Narayan, and J. Ye. Modeling disease progression via fused sparse group lasso. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1095–1103. ACM, 2012.