Option Tracing: Beyond Correctness Analysis in Knowledge Tracing

Aritra Ghosh[1], Jay Raspat[2], Andrew Lan[1]

[1]University of Massachusetts Amherst[†], [2] Independent Consultant

Abstract. Knowledge tracing refers to a family of methods that estimate each student's knowledge component/skill mastery level from their past responses to questions. One key limitation of most existing knowledge tracing methods is that they can only estimate an *overall* knowledge level of a student per knowledge component/skill since they analyze only the (usually binary-valued) correctness of student responses. Therefore, it is hard to use them to diagnose specific student errors. In this paper, we extend existing knowledge tracing methods beyond correctness prediction to the task of predicting the exact option students select in multiple choice questions. We quantitatively evaluate the performance of our option tracing methods on two large-scale student response datasets. We also qualitatively evaluate their ability in identifying common student errors in the form of clusters of incorrect options across different questions that correspond to the same error.

1 Introduction

Knowledge tracing (KT) [9] refers to a family of student modeling methods that estimate student mastery levels on knowledge components/skills/concepts from their past responses to questions/items and predict their future performance. These estimates and predictions can be used to i) provide feedback to students on their progress, especially in intelligent tutoring systems [44] and ii) drive personalization, i.e., selecting the action that each learner should take next to maximize their learning outcomes [10,28,36]. Many different KT methods have been developed, ranging from hidden Markov model-based Bayesian knowledge tracing methods [21,33,46], factor analysis-based methods such as learning factor analysis [5], performance factor analysis [34], and the item Difficulty, student ability, skill, and student skill practice history (DAS3H) method [7], to deep learning-based methods [15,31,32,35,45,47]. These methods have enjoyed various degrees of success; some of these methods, including most Bayesian knowledge tracing and factor analysis-based methods, exhibit excellent interpretability while other, deep learning-based methods trade off interpretability for excellent predictive accuracy on students' future performance.

However, one key limitation of these KT methods is that they operate exclusively on (usually binary-valued) response data that indicates whether a student

[†]This work is supported by the National Science Foundation under grant IIS-1917713. We also thank the reviewers for their constructive feedback.

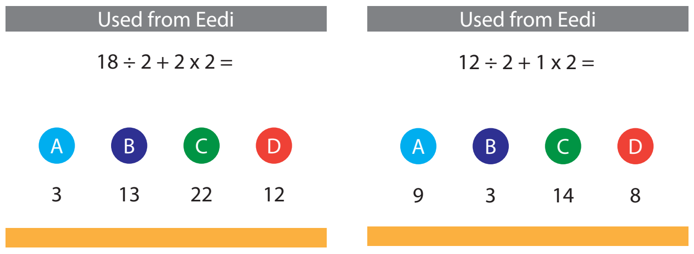

Fig. 1. Some distractor options in well-designed MCQs are potentially capable of capturing typical student errors. Option C in both questions here correspond to the error of not mastering the order of operations and always working left to right.

responds to a question correctly or not. Therefore, they can only estimate students' *overall* mastery level on each knowledge component. However, not all incorrect responses are equal: there can be numerous incorrect ways to answer a math question [27], caused by different underlying errors. Studies have shown that only a fraction of incorrect answers generated by students can be anticipated and explained by cognitive models integrated into intelligent tutoring systems [24,36,41], teachers [11], and numerical simulations [11,37]. Typical underlying errors include having a "buggy rule" [4], exhibiting a certain misconception [12,13,38], or a general lack of knowledge on certain knowledge components [2]. Since it is hard to diagnose such student errors from correctness data alone, we need to develop KT methods that analyze full student responses.

Some datasets, including the large-scale Eedi[1] [43] and EdNet[2] [8] datasets, contain the exact options students select on multiple choice questions (MCQs); this option data provides us with an opportunity to extend existing KT methods to analyze specific student option selections rather than their answer correctness. In an ideal situation, well-designed MCQs should have well-crafted incorrect distractor options that each corresponds to one or more typical student errors; Figure 1 shows an example from the Eedi dataset for two questions on the subject brackets, indices, division, multiplication, addition, subtraction (BIDMAS). Option C in both questions correspond to the same error of not fully mastering "order of operations" and always working left to right. However, manually identifying these errors is an unscalable and labor-intensive process since most existing MCQs do not come with consistent labels on the error(s) underlying each incorrect option. Therefore, it is important to explore whether we can develop KT methods to identify errors each incorrect option corresponds to and potentially diagnose student errors automatically. These methods would then be useful through i) informing teachers to communicate with students to understand the source of their errors, ii) enabling the development of automated feedback [19], and iii) enabling the design of alternative instructional approaches such as asking students to criticize erroneous examples [1].

[1] https://eedi.com/projects/neurips-education-challenge
[2] https://github.com/riiid/ednet

1.1 Contributions

In this paper, we develop option tracing (OT), a KT framework that uses the exact option each student selects on each question as both input and predicted output. We extend several existing KT methods to the OT setting, including a long short-term memory (LSTM) network-based method, deep knowledge tracing (DKT) [35], a graph convolutional network-based method, graph-based interaction model for knowledge tracing (GIKT) [45], and an attention network-based method, attentive knowledge tracing (AKT) [15]. We emphasize that the goal of this paper is **NOT** to compare all KT methods; instead, our goal is to study how can we generalize them to analyze student option selections in MCQs. Therefore, we only study some representative methods. We conduct the following experiments on the Eedi and EdNet datasets: First, we quantitatively evaluate our OT methods under both the collaborative filtering (CF) setup (introduced by the NeurIPS 2020 Education Challenge [43]) and the typical KT setup on the task of option prediction. Second, we qualitatively demonstrate the interpretability exhibited by our OT framework using clustering algorithms to group incorrect options across multiple questions into clusters of shared underlying errors. Results show that the learned clusters match up with those manually identified by a domain expert to some degree. Therefore, OT can potentially offer a *bottom-up* approach for error identification by extracting student errors from actual data instead of the typical *top-down* approach of anticipating errors before seeing data. Our implementation will be publicly available at `https://github.com/arghosh/OptionTracing`.

2 Related Work

The options students select in MCQs can be regarded as a type of *categorical data*, which has previously been studied in both the item response theory (IRT) and recommender systems research communities. However, in both cases, most prior works focus on the case where the categories are *ordered*. In IRT research, polytomous IRT-based models [25,26,30] are used to model students' responses with multiple ordered categories, such as letter grades and partial credits. In recommender systems research, neural collaborative filtering (NCF)-based methods are used to model star ratings provided by users on items [20]. There are relatively few models for *unordered* categorical data such as the nominal response model (NRM) from the IRT research community, which has been applied to the analysis of MCQs [40,42].

3 Data and Problem Setup

The Eedi dataset contains the responses of more than 100,000 students to 27,613 MCQs across 389 labeled subjects, totaling over 15 million responses over the course of more than a year. Each response corresponds to the exact option a student selected on each question (among four options, $\{A,B,C,D\}$). We will also use a small subset of the Eedi data where we have access to the exact question (in the form of images) for quantitative analysis; this dataset contains the responses of more than $4,900$ students to 948 questions, totaling in over 1.3 million responses. The EdNet dataset contains the responses of more than 700,000 students to 13,169 MCQs across 189 labeled subjects, totaling over 95 million responses over the course of more than two years.

We use two experimental setups for evaluation purposes. First, in the CF setup, the task is to predict each student's responses to a subset of questions that they responded to, given their responses to other questions (possibly in the future). Popular methods for this setup are neural collaborative filtering (NCF) [20] and graph convolutional networks (GCN) [3,23]. Second, in the KT setup for evaluating KT methods, the task is to predict each student's responses to future questions based on their entire past response history.

3.1 Problem Setup

Each student's performance record consists of a sequence of responses to questions assigned at a series of discrete time steps. For student i at time step t, we denote the combination of the question that they answered, the set of subjects this question covers, their binary-valued response correctness, the option they chose, and the correct option to this question as a tuple, $(q_t^i, \{s_{t,j}^i\}_{j=1}^{n_t^i}, r_t^i, y_t^i, c_t^i)$, where $q_t^i \in \mathbb{N}^+$ is the question index, $s_{t,j}^i \in \mathbb{N}^+$ denotes the index of the j^{th} subject, $j \in 1, \ldots, n_t^i$ since each question can be tagged with multiple subjects, $r_t^i \in \{0,1\}$ is the response correctness (1 corresponds to a correct response), $y_t^i \in \{A, B, C, D\}$ is the option the student selected, and $c_t^i \in \{A, B, C, D\}$ is the correct option for this question. In the CF setup, we associate a mask variable $m_t^i \in \{0,1\}$ with each time step, where 1 represents that the timestep is part of the training set. This variable helps us to mask out responses we need to predict when we compute the training loss. Given observed responses $\{(q_t^i, \{s_{t,j}^i\}_{j=1}^{n_t^i}, r_t^i, y_t^i, c_t^i)\}_{t:m_t^i=1}$, the task is to predict the exact options students select on questions in the test set, i.e., $y_{t'}^{i'}$ for $(t', i') : m_t^i = 0$. In the KT setup, we observe each student's entire history of responses to questions; thus, given their past history up to time $t-1$ as $\{(q_\tau^i, \{s_{\tau,j}^i\}_{j=1}^{n_\tau^i}, r_\tau^i, y_\tau^i, c_\tau^i)\}_{\tau=1}^{t-1}$, our goal is to predict y_t^i at the current time step, t. Under these notations, existing KT methods focus on predicting response correctness, r_t^i.

4 Methodology

In this section, we detail our OT methods for both the CF and KT setups. Before delving into the individual methods, we start with a set of unified modules that apply to all methods in this paper. The question embedding module $\mathrm{E}_q : q \to \mathbb{R}^d$ transforms the question index q_t^i to a d-dimensional, learnable real-valued vector in $\mathbb{R}^d$. Similarly, the response embedding module $\mathrm{E}_r : r \to \mathbb{R}^d$ transforms the response correctness r_t^i to $\mathbb{R}^d$ and the option embedding module $\mathrm{E}_o : \{A, B, C, D\} \to \mathbb{R}^d$ transforms the correct option c_t^i and the chosen option y_t^i to vectors in $\mathbb{R}^d$. We do not use separate embeddings for every question-option

(q, o) pair since that leads to overfitting in our experiments; instead, the $2d$-dimensional embedding for (q, o) is obtained using $[\mathrm{E}_q(q) \oplus \mathrm{E}_o(o)]$ where $\oplus$ is the concatenation operator. The subject embedding module $\mathrm{E}_s : s \rightarrow \mathbb{R}^d$ transforms the subject index to $\mathbb{R}^d$. Since each question may be tagged with several subjects, we define the final subject embedding as $\mathrm{E}_s(\{s^i_{t,j}\}_{j=1}^{n^i_t}) = \sum_{j=1}^{n^i_t} \mathrm{E}_s(s^i_{t,j})$. Some of the methods (such as NCF) use a user embedding module $\mathrm{E}_u : i \rightarrow \mathbb{R}^d$ that transforms the student index to $\mathbb{R}^d$. For simplicity, we use the same d-dimensional vector for all embedding modules; however, the dimensions of each module can be different. We train all model parameters, denoted as Θ, which contains the embeddings listed here and other model parameters specific to each individual method, by minimizing the negative log-likelihood of the selected options as

$$\underset{\Theta}{\text{minimize}} \quad -\sum_{i=1}^{|\text{Students}|} \sum_{t=1}^{|\text{Sequence}_i|} \sum_{o \in \{A,B,C,D\}} \mathbb{1}[y^i_t = o] \log p(o|q^i_t; \Theta),$$

where $\mathbb{1}$ is the indicator function. Since the options are unordered categories, the resulting loss function corresponds to the common cross-entropy loss [16].

4.1 Option Prediction under the CF Setup

NCF. NCF is one of the most popular CF methods for user-item interaction data. In the option prediction task, students correspond to users and questions corresponds to items. The input for NCF at time step t for student i, $\mathbf{x}^i_t$, is

$$\mathbf{x}^i_t = [\mathrm{E}_q(q^i_t) \oplus \mathrm{E}_u(i) \oplus \mathrm{E}_s(\{s^i_t\})].$$

Predictive probabilities $p(y^i_t = o)$ over four options $o \in \{A, B, C, D\}$ are calculated using the softmax function [16],

$$\mathbf{z}^i_t = f(\mathbf{x}^i_t) \in \mathbb{R}^4, \quad p(y^i_t = o|\mathbf{x}^i_t) = [\text{softmax}(\mathbf{z}^i_t)]_o, \quad \hat{y}^i_t = \underset{o \in \{A,B,C,D\}}{\text{argmax}} [\mathbf{z}^i_t]_o,$$

where $f(\cdot)$ denotes a feed-forward, fully-connected neural network and $[]_o$ refers to the o^{th} entry of a vector. In NCF, the model parameters are the weights and biases in the feed-forward neural network $f(\cdot)$; this prediction module is shared by the subsequent methods.

PO-BiDKT. The main drawback of NCF is that the student embedding is static and not updated as students answer more questions and their knowledge states evolve. Recurrent neural networks, and in particular LSTM-type models are capable of modeling evolving knowledge as hidden states [35]. However, we cannot directly use methods such as DKT in the CF setup since the student's responses at some time steps in their response sequence are not observed. Therefore, we use the following method to handle evolving knowledge states using recurrent networks with missing observations. The input at each time step is given by

$$\mathbf{x}_t = [\mathrm{E}_q(q^i_t) \oplus \mathrm{E}_o(c^i_t) \oplus \mathrm{E}_s(\{s^i_t\}) \oplus \big(\mathrm{E}_o(y^i_t) \odot m^i_t\big) \oplus \big(\mathrm{E}_l(r^i_t) \odot m^i_t\big)], \quad (1)$$

where $\odot$ denotes the element-wise multiplication between two vectors. We mask the option embeddings and response correctness embeddings using m_t^i for time steps where we do not observe them but still use the question embedding as input. We also extend the base LSTM module in DKT to a bi-directional LSTM (Bi-LSTM) [17]. Here, we compute two latent knowledge states using two separate LSTM modules, the forward state $\overrightarrow{\mathbf{h}}_t$ that summarizes the student's past response history and the backward state $\overleftarrow{\mathbf{h}}_t$ that summarizes the student's future response history at time step t as

$$\overrightarrow{\mathbf{h}}_{t+1} = \text{Forward LSTM}(\overrightarrow{\mathbf{h}}_t, \mathbf{x}_t), \;\; \overleftarrow{\mathbf{h}}_{t-1} = \text{Backward LSTM}(\overleftarrow{\mathbf{h}}_t, \mathbf{x}_t).$$

The final latent knowledge state is the concatenation of the two states as $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t \oplus \overleftarrow{\mathbf{h}}_t]$. The parameters include two sets of parameters for the forward and backward LSTMs in addition to the parameters for the fully connected network $f(\cdot)$. We call this method partially observed bi-directional DKT, or PO-BiDKT. The output to the prediction module is computed using

$$\mathbf{z}_t^i = f([\mathbf{h}_t^i \oplus \mathrm{E}_q(q_t^i) \oplus \mathrm{E}_o(c_t^i) \oplus \mathrm{E}_s(\{s_t^i\})]) \in \mathbb{R}^4. \tag{2}$$

GCN-augmented PO-BiDKT (BiGIKT). In our datasets, each question is tagged with a few subjects by question designers or domain experts. These subject tags provide important information on how these questions are related since we expect questions from the same subject to have some shared features. GCNs excel at formulating these relations and learning from graph-structured data. Since we can represent the question-subject association matrix using a bipartite graph, (loosely) following GIKT [45], we connect GCNs with PO-BiDKT to jointly learn question and subject embeddings using the structure imposed by the subject tags. In this method, we use hierarchical representations of subjects and questions: starting with initial subject and question embeddings $\mathrm{E}_s(s_t^i)$ and $\mathrm{E}_q(q_t^i)$, the first layer GCN embedding for the j^{th} subject and the second layer GCN embedding for the i^{th} question are computed as

$$\mathbf{s}_j^1 = \tanh\Big(\mathbf{W}_s^s \mathrm{E}_s(s_j) + \frac{\sum_{i \in N_j^s} \mathbf{W}_s^q \mathrm{E}_q(q_i)}{|N_j^s|}\Big), \; \mathbf{q}_i^2 = \tanh\Big(\mathbf{W}_q^q \mathrm{E}_q(q_i) + \frac{\sum_{j \in N_i^q} \mathbf{W}_q^s \mathbf{s}_i^1}{|N_i^q|}\Big),$$

where N_j^s (N_i^q) denotes the set of questions (subjects) associated with subject (question) s_j (q_i) and $\mathbf{W}_s^s$, $\mathbf{W}_s^q$, $\mathbf{W}_q^s$ and $\mathbf{W}_q^q$ are learnable parameter matrices. The hyperbolic tangent (tanh) non-linearity operate entry-wise on vectors. We replace the subject embeddings $\mathrm{E}_s(s_t^i)$ and the question embedding $\mathrm{E}_q(q_t^i)$ in the base Bi-LSTM (Eq. 1 and Eq. 2) with these GCN-based embeddings. The model parameters of this method include the GCN weight parameter matrices in addition to the Bi-LSTM parameters.

4.2 Option Prediction under the KT Setup

In the KT setup, we predict future responses using only past responses and assume that every past student response is observed. We extend several existing neural network-based KT methods for the option prediction task.

DKT. We apply a simple modification to the DKT method [35] to extend it to i) predict options instead of response correctness and ii) handle questions that are tagged with multiple subjects (the original DKT method assumes that each question is tagged with a single subject). We use

$$\mathbf{x}_t = [\mathrm{E}_q(q_t^i) \oplus \mathrm{E}_o(c_t^i) \oplus \mathrm{E}_s(\{s_t^i\}) \oplus \mathrm{E}_o(a_t^i) \oplus \mathrm{E}_l(r_t^i)]$$

as the input to the DKT LSTM input module. The student's hidden knowledge states are computed using the LSTM model as $\mathbf{h}_{t+1} = \mathrm{LSTM}(\mathbf{h}_t, \mathbf{x}_t)$. The predictive probabilities of selecting each option are computed using

$$\mathbf{z}_t^i = f([\mathbf{h}_t^i \oplus \mathrm{E}_q(q_t^i) \oplus \mathrm{E}_o(c_t^i) \oplus \mathrm{E}_s(\{s_t^i\})]) \in \mathbb{R}^4,\ \hat{y}_t^i = \underset{o \in \{A,B,C,D\}}{\mathrm{argmax}} [\mathbf{z}_t^i]_o.$$

DKVMN. Instead of using LSTMs to model latent knowledge state transitions, the dynamic key-value memory network (DKVMN) method uses a key-value memory network to retrieve and update knowledge at every time step using an external memory module as $\mathbf{h}_{t+1} = \mathrm{MemoryModule}(\mathbf{h}_t, \mathbf{x}_t)$; refer to [47] for details. We use the same input and output structure for the DKVMN memory module as that for DKT.

AKT. We also adapt AKT, an attention network-based, state-of-the-art KT method for the option prediction task. AKT computes a query, a key, and a value vector for each time step, and then uses the similarity between the query and key vectors at different time steps to attend to questions in the past and use their corresponding value vectors to retrieve acquired knowledge in the past. We compute the query, key, and value vectors as $\mathbf{q}_t = \mathbf{W}^Q\mathbf{n}$, $\mathbf{k}_t = \mathbf{W}^K\mathbf{n}$, and $\mathbf{v}_t = \mathbf{W}^V[\mathrm{E}_l(r_t^i) \oplus \mathrm{E}_q(q_t^i) \oplus \mathrm{E}_o(y_t^i) \oplus \mathrm{E}_o(c_t^i)]$ respectively, where $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ are the query, key, and value projection matrices and $\mathbf{n} = [\mathrm{E}_q(q_t^i) \oplus \mathrm{E}_s(\{s_t^i\}) \oplus \mathrm{E}_o(c_t^i)]$. The retrieved latent knowledge state is then computed as $\mathbf{h}_t = g\Big(\sum_{\tau<t} \alpha_{t,\tau}\mathbf{v}_\tau\Big)$, where g is another feedforward network and $\alpha_{t,\tau}$ is the normalized attention score between the query at the current time step t and the key at a past time step τ. For AKT, we employ the exponential decay module to compute the attention scores [15] and then compute the output using the attention-weighted value $\mathbf{h}_t^i$ and a fully connected network $f(\cdot)$.

5 Experiments

Experimental Setup. In addition to the option prediction task, we also evaluate all methods under the standard, binary-valued response correctness prediction task. We do not need to use a separate set of methods; instead, we can simply replace the final output layer of the option predictor module ($f : \cdot \to \mathbb{R}^4$) with an output layer that consists of a single node ($f : \cdot \to \mathbb{R}^1$) for all OT methods; the resulting loss function corresponds to standard binary cross entropy loss. For option prediction, we use both accuracy and macro F_1 score as evaluation metrics. For correctness prediction, we use accuracy as the only evaluation metric which aligns with the option prediction task. We compute the F_1

	Option Prediction				Correctness Prediction	
	Accuracy		Average Macro F_1 Score		Accuracy	
Model	Eedi	EdNet	Eedi	EdNet	Eedi	EdNet
NCF	64.75±0.02	67.24±0.01	0.2824 ± 0.002	0.2552 ± 0.001	72.6±0.03	71.49±0.01
PO-BiDKT	65.87±0.01	**69.42±0.01**	**0.3283 ± 0.001**	**0.3260 ± 0.001**	75.18±0.01	**75.21±0.02**
BiGIKT	**66.16±0.02**	69.29±0.02	0.3261 ± 0.001	0.3168 ± 0.001	**75.62±0.02**	75.07±0.01
DKT	65.95±0.44	68.03±0.09	0.313± 0.008	0.2887 ± 0.005	74.7±0.34	73.19±0.06
DKVMN	**66.03±0.49**	68.01±0.1	**0.3152 ± 0.007**	0.2842 ± 0.005	**74.75±0.3**	73.02±0.06
AKT	65.91±0.47	**68.44±0.09**	0.3139 ± 0.007	**0.3062 ± 0.004**	74.65±0.31	**73.6±0.06**

Table 1. Performance of all methods under the CF (top half) and KT (bottom half) setups on both datasets. Best results are in **bold.**

score for each question-option pair individually and average across all such pairs. This metric treats every option in every question equally, thus magnifying the impact of options that are rarely selected. For reference, on the Eedi and EdNet datasets, the selection probabilities across options for an average question (from most frequent to least frequent) are 57%, 25%, 11%, 7% and 66%, 20%, 10%, 4%, respectively. For option prediction, a random classifier has an average macro F_1 score and an accuracy score of 0.25 on both of these datasets, while a majority class classifier has an average macro F_1 score (accuracy) of 0.184 (57%) and 0.205 (66%) on the Eedi and EdNet datasets, respectively.

Training and Testing. We perform standard k-fold cross-validation (with $k = 5$) for all methods on both datasets. Under the CF setup, on average 20% of the time steps (for each student) are randomly chosen as the held out test set, 20% of time steps are randomly chosen as the validation set, and the other 60% are chosen as the training set to train all methods. Under the KT setup, all time steps for a randomly chosen 20% of students are used as the test set, and the validation and training sets are constructed similarly.

Network Architectures and Hyper-parameters. Since the datasets are large, we do minimal hyper-parameter tuning and set most of the values to their default values for all the methods; exploratory experiments found that evaluation results are robust across most parameter values. We set the question, subject, option, response embedding dimension for all methods to $d \in \{32, 64\}$ for the CF setup and $d \in \{64, 128\}$ for the KT setup. We use the Adam optimizer [22] to train all models with a batch size of 64 students to ensure that an entire batch can fit into the memory of our machine (equipped with one NVIDIA Titan X GPU). For all methods, we set the learning rate to $10^{-4}/10^{-3}$ for the Eedi/EdNet dataset and run all the methods for 200 epochs and perform early stopping based on the loss on the validation set. We set the latent knowledge state ($\mathbf{h}_t$) dimension to 256/512 for all methods under CF/KT setup. For NCF, we select the user embedding dimension as $d = 256$.

Results and Discussion. Table 1 lists the performance of all OT methods for both the CF and KT setups for both the option prediction and correctness prediction tasks, on both datasets; we report the averages as well as the standard deviations across the five folds. We observe a significant dropoff ($\sim$ 10%) in the accuracy metric on the option prediction task compared to the correctness

Metric	Adjusted Rand Index	Fowlkes-Mallows index
Score	0.372	0.455

Table 2. Incorrect option clustering quality for a subset of questions in the Eedi dataset using errors labeled by a domain expert. 1 in both metrics indicates perfect clustering.

prediction task, which is as expected since there are four categories to predict (A, B, C, D) instead of two categories (correct/incorrect). As a result of this difference, the correctness prediction task can be seen as a sub-task in the option prediction task by computing the probability a student selects the correct option. The performance of different methods are also quite consistent across all cases.

We observe that recurrent neural network-based methods such as PO-BiDKT perform significantly better than NCF in all cases. This observation suggests that even in a CF setup for model evaluation, methods that take the evolving nature of student knowledge into account are still more effective than popular CF methods that do not account for these temporal dynamics. Overall, we observe that the performance gains on the option prediction task provided by complex model architectures are marginal. This observation suggests that more work needs to be done on the option prediction task to understand the dynamics behind students' decisions to select a specific incorrect option, which motivates our exploration in Section 5.1. In the KT setup, we observe that DKVMN performs best on the Eedi dataset while AKT performs best on the EdNet dataset. This observation suggests that complex neural network architectures such as attention modules are more beneficial when a large amount of training data is available.

In both setups, we observe that the F_1 scores are low for all methods; despite clearly not simply predicting the most frequent option, the performance of these methods leaves significant room for improvement due to class imbalance. Possible approaches to improve prediction accuracy for options that are rarely selected include oversampling them [6]; however, since a student's responses to different questions are not independent data points, how these methods can be applied to the option tracing task is not immediately clear.

5.1 Clustering Incorrect Options

To qualitatively evaluate our option tracing methods, we attempt to group incorrect options across multiple questions into clusters and examine whether question-option pairs in the same cluster correspond to the same underlying error. To this end, we train a modified version of PO-BiDKT on the Eedi dataset [43]; we learn an embedding module $\mathrm{E}_{q,o}(q, o) : q \times o \to \mathbb{R}^d$ for each question-option pair. Then, we compute the option selection probabilities using the latent knowledge state h_t^i and the question-option pair embeddings as $p(o|q_t^i) = \frac{f(h_t^i)^T \mathrm{E}_{q,o}(q_y^i, o)}{\sum_{o'} f(h_t^i)^T \mathrm{E}_{q,o}(q_y^i, o')}$. This modification suffers a small drop in predictive performance but encodes information in the question-option pair embeddings for us to cluster them and search for common student errors.

We selected all incorrect options $(31 \times 3 = 63)$ in questions on subject 33 (BIDMAS) where question images are released on the smaller subset of the Eedi dataset; see [43] for details. A domain expert manually labeled each option based on which error likely resulted in the student selecting it, resulting in a total of 14 high-level errors (errors that cannot be named are excluded), each corresponding to multiple options across different questions; further splitting them into finer-grained errors results in clusters that are not meaningful. We perform k-means clustering [29] on the learned question-option pair embeddings and compare them to the "ground truth" option clusters provided by the expert.

Due to spatial constraints, we only report quantitative results on clustering quality using two commonly used metrics: The adjusted Rand index [39] and the Fowlkes-Mallows index [14]; the former has a range of $[-1, 1]$ while the latter has a range of $[0, 1]$, with 1 corresponding to perfect clustering.

Table 2 lists these metrics on the learned question-option pair embeddings based on the ground truth expert labeling. Overall, the clustering performance is acceptable but not excellent. We observe that some errors such as "sign error in calculation involving negative numbers" have relatively easy-to-identify corresponding option clusters (5 out of 8 options labeled by the expert as corresponding to that error are put into the same cluster). On the other hand, some options such as $69D$ and $293C$ (the left half of Figure 1) correspond to the same error but are not grouped into the same cluster. One possible explanation is that students may not consistently demonstrate an error, as found in prior research [41]; among students who selected $69D$, only 51% selected $293C$ while 34% of them selected the correct option, $293B$. Therefore, further work is required to study whether more robust KT methods and clustering algorithms can identify error clusters more effectively. Nevertheless, our approach produces a starting point to reduce the effort for domain experts to manually label errors and provides them a way to do it under data-driven support.

6 Conclusions and Future Work

Analyzing the exact options students select across multiple choice questions has the potential to uncover their error modes and help teachers to provide targeted feedback to improve learning outcomes. In this paper, we proposed a set of methods to extend common knowledge tracing methods that analyze only the correctness of students' responses to questions to analyze the exact options they select on multiple choice questions. We validated these methods with quantitative experiments on two large-scale datasets in terms of their ability to predict the options students select on each question and qualitative experiments in terms of clustering incorrect options according to underlying errors. There are many avenues for future work. First, we need to develop methods that are aware of the evolving nature of student errors. One possible approach is to develop methods that can explicitly account for the recurrence of past errors, such as using a neural copy mechanism [18]; these methods may help us track students' progress in correcting their errors. Second, low F_1 scores for the option prediction task suggest that it is much more challenging than the typical correctness prediction task in knowledge tracing literature and thus deserves more attention.

References

1. Adams, D.M., McLaren, B.M., Durkin, K., Mayer, R.E., Rittle-Johnson, B., Isotani, S., Van Velsen, M.: Using erroneous examples to improve mathematics learning with a web-based tutoring system. Computers in Human Behavior **36**, 401–411 (2014)
2. Anderson, J.R., Jeffries, R.: Novice lisp errors: Undetected losses of information from working memory. Human–Computer Interaction **1**(2), 107–131 (1985)
3. Berg, R.v.d., Kipf, T.N., Welling, M.: Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263 (2017)
4. Brown, J.S., Burton, R.R.: Diagnostic models for procedural bugs in basic mathematical skills. Cognitive science **2**(2), 155–192 (1978)
5. Cen, H., Koedinger, K., Junker, B.: Learning factors analysis–A general method for cognitive model evaluation and improvement. In: Proc. International Conference on Intelligent Tutoring Systems. pp. 164–175 (2006)
6. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)
7. Choffin, B., Popineau, F., Bourda, Y., Vie, J.J.: DAS3H: Modeling student learning and forgetting for pptimally scheduling distributed practice of skills. In: Proc. International Conference on Educational Data Mining. pp. 29–38 (2019)
8. Choi, Y., Lee, Y., Shin, D., Cho, J., Park, S., Lee, S., Baek, J., Bae, C., Kim, B., Heo, J.: Ednet: A large-scale hierarchical dataset in education. In: Proc. International Conference on Artificial Intelligence in Education. pp. 69–73. Springer (2020)
9. Corbett, A., Anderson, J.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-adapted Interaction **4**(4), 253–278 (Dec 1994)
10. Doroudi, S., Aleven, V., Brunskill, E.: Where's the reward? International Journal of Artificial Intelligence in Education **29**(4), 568–620 (2019)
11. Erickson, J.A., Botelho, A.F., McAteer, S., Varatharaj, A., Heffernan, N.T.: The automated grading of student open responses in mathematics. In: Proc. International Conference on Learning Analytics & Knowledge. pp. 615–624 (2020)
12. Feldman, M.Q., Cho, J.Y., Ong, M., Gulwani, S., Popović, Z., Andersen, E.: Automatic diagnosis of students' misconceptions in K-8 mathematics. In: Proc. CHI Conference on Human Factors in Computing Systems. pp. 1–12 (2018)
13. Feng, J., Zhang, B., Li, Y., Xu, Q.: Bayesian diagnosis tracing: Application of procedural misconceptions in knowledge tracing. In: Proc. International Conference on Artificial Intelligence in Education. pp. 84–88. Springer (2019)
14. Fowlkes, E.B., Mallows, C.L.: A method for comparing two hierarchical clusterings. Journal of the American statistical association **78**(383), 553–569 (1983)
15. Ghosh, A., Heffernan, N., Lan, A.S.: Context-aware attentive knowledge tracing. In: Proc. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2330–2339 (2020)
16. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
17. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 6645–6649 (2013)
18. Gu, J., Lu, Z., Li, H., Li, V.O.: Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393 (2016)

19. Gusukuma, L., Bart, A.C., Kafura, D., Ernst, J.: Misconception-driven feedback: Results from an experimental study. In: Proc. ACM Conference on International Computing Education Research. pp. 160–168 (2018)
20. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proc. International Conference on World Wide Web. pp. 173–182 (2017)
21. Khajah, M., Huang, Y., González-Brenes, J., Mozer, M., Brusilovsky, P.: Integrating knowledge tracing and item response theory: A tale of two frameworks. In: Proc. International Workshop on Personalization Approaches in Learning Environments. vol. 1181, pp. 7–15 (2014)
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proc. International Conference on Learning Representations (2015)
23. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
24. Koedinger, K.R., Corbett, A., et al.: Cognitive tutors: Technology bringing learning sciences to the classroom. The Cambridge Handbook of the Learning Sciences pp. 61–77 (2006)
25. Lan, A.S., Studer, C., Baraniuk, R.G.: Matrix recovery from quantized and corrupted measurements. In: IEEE Intl. Conf. on Acoustics, Speech and Signal Processing. pp. 4973–4977 (May 2014)
26. Lan, A.S., Studer, C., Waters, A.E., Baraniuk, R.G.: Tag-aware ordinal sparse factor analysis for learning and content analytics. In: Proc. 6th Intl. Conf. Educ. Data Min. pp. 90–97 (July 2013)
27. Lan, A.S., Vats, D., Waters, A.E., Baraniuk, R.G.: Mathematical language processing: Automatic grading and feedback for open response mathematical questions. In: Proc. ACM Conference on Learning at Scale. pp. 167–176 (2015)
28. Lindsey, R., Shroyer, J., Pashler, H., Mozer, M.: Improving students' long-term knowledge retention through personalized review. Psychological Science **25**(3), 639–647 (Jan 2014)
29. Lloyd, S.: Least squares quantization in pcm. IEEE Transactions on Information Theory **28**(2), 129–137 (1982)
30. Ostini, R., Nering, M.L.: Polytomous item response theory models. No. 144, Sage (2006)
31. Pandey, S., Karypis, G.: A self attentive model for knowledge tracing. In: Proc. International Conference on Educational Data Mining. pp. 384–389 (July 2019)
32. Pandey, S., Srivastava, J.: Rkt: Relation-aware self-attention for knowledge tracing. arXiv preprint arXiv:2008.12736 (2020)
33. Pardos, Z.A., Heffernan, N.T.: Modeling individualization in a Bayesian networks implementation of knowledge tracing. In: Proc. International Conference on User Modeling, Adaptation, and Personalization. pp. 255–266 (2010)
34. Pavlik Jr, P., Cen, H., Koedinger, K.: Performance factors analysis–A new alternative to knowledge tracing. In: Proc. International Conference on Artificial Intelligence in Education (2009)
35. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. In: Proc. Conference on Advances in Neural Information Processing Systems. pp. 505–513 (2015)
36. Ritter, S., Anderson, J.R., Koedinger, K.R., Corbett, A.: Cognitive tutor: Applied research in mathematics education. Psychonomic Bulletin & Review **14**(2), 249–255 (2007)

37. Selent, D.A.: Creating Systems and Applying Large-Scale Methods to Improve Student Remediation in Online Tutoring Systems in Real-time and at Scale. Ph.D. thesis, Worcester Polytechnic Institute (2017)
38. Smith III, J.P., DiSessa, A.A., Roschelle, J.: Misconceptions reconceived: A constructivist analysis of knowledge in transition. The journal of the learning sciences **3**(2), 115–163 (1994)
39. Steinley, D.: Properties of the hubert-arable adjusted rand index. Psychological methods **9**(3), 386 (2004)
40. Thissen, D., Steinberg, L.: A taxonomy of item response models. Psychometrika **51**(4), 567–577 (1986)
41. VanLehn, K.: Bugs are not enough: Empirical studies of bugs, impasses and repairs in procedural skills. The Journal of Mathematical Behavior (1982)
42. Wang, F., Liu, Q., Chen, E., Huang, Z., Chen, Y., Yin, Y., Huang, Z., Wang, S.: Neural cognitive diagnosis for intelligent education systems. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 6153–6161 (2020)
43. Wang, Z., Lamb, A., Saveliev, E., Cameron, P., Zaykov, Y., Hernández-Lobato, J.M., Turner, R.E., Baraniuk, R.G., Barton, C., Jones, S.P., et al.: Diagnostic questions: The neurips 2020 education challenge. arXiv preprint arXiv:2007.12061 (2020)
44. Woolf, B.P.: Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-learning. Morgan Kaufmann (2010)
45. Yang, Y., Shen, J., Qu, Y., Liu, Y., Wang, K., Zhu, Y., Zhang, W., Yu, Y.: Gikt: A graph-based interaction model for knowledge tracing. In: Proc. Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2020)
46. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized bayesian knowledge tracing models. In: International conference on artificial intelligence in education. pp. 171–180. Springer (2013)
47. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: Proc. International Conference on World Wide Web. pp. 765–774 (Apr 2017)