"Can A Language Model Represent Math Strategies?": Learning Math Strategies from Big Data using BERT

Abisha Thapa Magar
University of Memphis
Memphis, TN, USA
thpmagar@memphis.edu

Anup Shakya
University of Memphis
Memphis, TN, USA
ashakya@memphis.edu

Stephen E. Fancsali
Carnegie Learning
Pittsburgh, PA, USA
sfancsali@carnegielearning.com

Vasile Rus
University of Memphis
Memphis, TN, USA
vrus@memphis.edu

April Murphy
Carnegie Learning
Pittsburgh, PA, USA
amurphy@carnegielearning.com

Steve Ritter
Carnegie Learning
Pittsburgh, PA, USA
sritter@carnegielearning.com

Deepak Venugopal
University of Memphis
Memphis, TN, USA
dvngopal@memphis.edu

Abstract

AI models have shown a remarkable ability to perform representation learning using large-scale data. In particular, the emergence of Large Language Models (LLMs) attests to the capability of AI models to learn complex hidden structures in a bottom-up manner without requiring a lot of human expertise. In this paper, we leverage these models to learn Math learning strategies at scale. Specifically, we use student interaction data from the MATHia Intelligent Tutoring System to learn strategies based on sequences of actions performed by students. To do this, we develop an AI model based on BERT (Bidirectional Encoder Representations From Transformers) that has two main components. First, we pre-train BERT using an approach known as Masked Language Modeling to learn embeddings for strategies. The embeddings represent strategies in a vector form while preserving their semantics. Next, we fine-tune the model to predict if students are likely to apply a correct strategy to solve a novel problem. We demonstrate using a large dataset collected from 655 schools that our approach where we pre-train to learn strategies from a sample of schools can be fine-tuned with a small number of examples to make accurate predictions over student data collected from other schools.

CCS Concepts

• **Computing methodologies** → **Neural networks**; **Artificial intelligence**; • **Applied computing** → **Interactive learning environments**.

Keywords

Math strategies, Representation learning, BERT, Big Data, Transformers

ACM Reference Format:
Abisha Thapa Magar, Anup Shakya, Stephen E. Fancsali, Vasile Rus, April Murphy, Steve Ritter, and Deepak Venugopal. 2025. "Can A Language Model Represent Math Strategies?": Learning Math Strategies from Big Data using BERT. In *LAK25: The 15th International Learning Analytics and Knowledge Conference (LAK 2025), March 03–07, 2025, Dublin, Ireland.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3706468.3706558

1 Introduction

Recent advances in Artificial Intelligence (AI) have revolutionized domains such as Natural Language Processing and Computer Vision. In particular, combining the power of big data with vast computational resources, Large Language Models (LLMs) have demonstrated an extraordinary ability to understand and learn representations for complex language structures. In this paper, we leverage these advances in a novel domain. That is, we build AI models to represent the structure of *math strategies* in problem solving. In general, when students learn to solve math problems, they typically execute strategies that could be a function of various factors such as experience with similar problems, general expertise in the topic, cognitive abilities, prior learning, etc. Through AI models, we represent such math strategies in a *bottom-up* manner, where we analyze data to extract patterns of strategic behavior.

In the past, human experts have explored the factors that influence math strategies used by students [18]. However, human experts are expensive and limited in their ability to analyze large data from thousands, tens of thousands, or even millions of students. On the other hand, AI models can utilize the diversity in large-scale data to represent strategies effectively. In this paper, we develop a model to represent strategies from big data using Bi-Directional Encoding Representations From Transformers (BERT) [11], a state-of-the-art approach that has been widely used for language understanding.

In particular, we learn our models on data collected from MATHia, a well-known Intelligent Tutoring System (ITS) [24]. In MATHia, a learner interacts with the ITS performing actions to complete steps to solve a problem. We define the notion of a strategy in this space as a sequence of steps completed by the learner. However, the difficulty is that by observing the completion of steps alone, it is hard to establish that the students exhibited strategic behavior. Therefore, we define the notion of *intentional strategies* based on an instructional design available in some MATHia content known as *optional tasks*. Here, learners can work on optional tasks that provide instructional scaffolding and support to help them complete steps in the problem. Further, the design is flexible to allow learners to jump between steps in the optional task and the non-optional steps.

We use BERT *pre-training* to learn representations for strategies based on sequences of actions performed by a learner. We pre-train the model in an unsupervised manner by masking steps in the sequence and inferring the masked steps from a bidirectional context, i.e., from both left-to-right and right-to-left directions in the sequence. The pre-training learns *embeddings* for strategies such that it preserves the semantics of a strategy. We then *fine-tune* the pre-trained model to predict if a learner's strategy can correctly solve novel problems. Importantly, fine-tuning only requires a small number of labeled examples since the pre-trained model has already learned to encode strategies.

We collect 414K instances (student, problem pairs) from 655 schools where students used MATHia in a 7^{th} grade workspace (used to teach a specific domain). We pre-train the model to learn strategies in an unsupervised manner from data collected from a sample of schools and fine-tune the model with a small number of labeled examples from other schools. We show that our model achieves over 80% accuracy in our prediction task, thus demonstrating that our pre-trained model learns to represent general strategies underlying the data in a bottom-up manner without requiring explicit guidance from human experts.

2 Related Work

Ritter et al. [22] provides a comprehensive survey on different approaches used to identify student strategies. Classical model tracing [5] based methods can be used to identify if certain strategies can improve learner mastery. In [21], based on model tracing, an approach was proposed that can evaluate the effectiveness of an unknown strategy. Based on its effectiveness, the strategy can be used to augment the set of known correct/incorrect strategies and thus incrementally build the set of labeled strategies. Several discriminative learning methods have been explored to identify strategies. For instance, labeled datasets were created based on students interacting with the ITS [7], and in each of these cases, the interactions are labeled manually based on one/more experts. Such labeled datasets can be used for training Machine Learning models. However, the quality of strategy identification is heavily dependent upon the feature specification [20]. In [8], a Machine learning model was learned from data labeled from Cognitive Tutor, and similar approaches have been used in several other contexts. For instance, in [9], a Machine learning model was used for inferring strategies from SQL-tutor, in [12] for strategies in role-playing games, in [23] for strategies in understanding conceptual physics, etc. Sequence modeling has often been used to detect strategies from sequential datasets. Specifically, Betty's brain [1] is a virtual environment where students learn about scientific processes by teaching a virtual agent called Betty. This is an open-ended learning environment where students have the flexibility to perform their tasks in a variety of different ways which yields different strategies. Other studies have used this environment to identify strategies through sequence mining [14, 15]. In [32], sequence pattern mining was applied to a MOOCs platform to analyze activity sequences of learners [32]. Shakya et al. [26, 27] developed a scalable approach to train LSTMs using interaction data from Mathia. In conversational agents, we can view strategies in the space of dialogue acts [4] and/or actions [2]. Therefore, one approach that has been explored is to map language into sequences of dialogue acts and identify higher-level pedagogical *modes* from these sequences [17, 25, 31].

3 Approach

In general, the type of strategies that can be inferred depends upon the environment in which we interact with the learner. For instance, in a classroom environment, a teacher who can directly observe student activities in different contexts can have a much deeper insight into the strategies a student is learning and applying. However, in this case, it is hard to scale up due to the human effort and expertise involved. On the other hand, an ITS environment typically would have a more limited bandwidth in terms of what it can communicate about student strategies. For instance, suppose the design choices made in the ITS force a student down predictable/specific paths, then clearly the student cannot explore alternate strategies. On the other hand, a more flexible design of the ITS interface could let the student explore and learn different strategies. The advantage though is the massive scale at which data can be collected through an ITS. Combined with advances in AI, this gives us a unique opportunity to understand strategies in a *bottom-up* manner. Specifically, we start with the data and use AI models to represent strategies embedded within the data without explicitly telling the model the type of strategies it needs to understand. This is analogous to Large Language Models (LLMs) learning language automatically without being explicitly taught the semantics, grammar, and linguistic rules that are part of natural language.

A top-down approach on the other hand defines strategies at a conceptual level (e.g. guess and check, pattern finding, tabular methods, etc.) and then grounds these strategies in the data to gain insights (e.g. how do learners learn strategies? Which strategies are more beneficial?). The advantage of a bottom-up approach is that it can help us learn strategies from patterns that may not necessarily conform to prior assumptions and therefore, could lead to novel design improvements in the ITS to support unique learning styles.

3.1 Strategies in MATHia

In MATHia, to solve a problem, learners perform actions, where each action is intended to complete a *goal node* which indicates a step towards solving the problem. We use the terms step and goal-node interchangeably. However, there are several *paths* to complete each goal node. Therefore, similar to Newell and Simon [19], here, we define a strategy as *sequences of steps completed by performing actions within the action-space.* An example is illustrated in Fig. 1.

The example in Fig. 1 shows a problem from the workspace in 7^{th}-grade math for calculating *percent increase and percent decrease.* There are 5 steps that need to be completed to solve the given problem. Each step is associated with one or more *skills* (knowledge components [16]). Thus, each step provides an opportunity for the learner to improve the skills associated with that step. There is no fixed ordering of steps that need to be completed, i.e., the learner is free to choose steps based on different orders. These orders could indicate the thought process of the learner. Further, at each step, if a learner makes a mistake, they are given an opportunity to repeat the step till it is completed without error. However, the learner does not need to continually work on the same step till completion. Between repetitions, they are free to choose other actions, i.e., they can work on other steps before returning to the error step. Further, they can also ask for hints to help them complete a step. The top-level hint provides scaffolding while the more detailed hint leads them more directly towards the action needed to complete the step.

Thus, we can see that there are several *paths* that the learner can take to navigate through the action space. A visual representation of actions followed by students can be represented as a directed graph. An example of this is illustrated in Fig. 1 (b). Here, the directed arrows show an action that the student performed to go from one step to another and the thickness shows how often that action was repeated. In Fig. 1 (c), we show the completed steps, and the order in which they were completed is shown by the directed arrows. This path through the action space is a strategy of the learner.

3.2 Intentional Strategies

It is important to distinguish between *intentional strategies* and sequences of actions that are not intentional. For example, by permuting the steps, we can obtain several variants to the path shown in Fig. 1 (c). However, the fact that a learner simply changes the ordering over steps in a randomized manner may not always suggest the intention of executing strategies that are fundamentally different from one another. Therefore, we instead need to identify more intentional actions that are suggestive of *strategic behavior*, as compared to non-strategic behavior. To do this, we use a specific instructional design in Mathia called *optional tasks* that allows us to define such intentional strategic behavior.

The idea of optional tasks is to allow a learner to work on one or more tasks within a problem context that can help the student complete steps in the original problem. Thus, a student who works out a specific optional task intentionally makes choices in the action space which could constitute a valid strategy. We illustrate an example of the optional task design for the problem shown in Fig. 2 (a). As seen in the example, the student has an opportunity to choose a specific optional task based on the problem, solve steps within the optional task, and then proceed to solve the problem. Note that the learner has to make several decisions that are indicative of strategic behavior. Specifically i) *if* they need to work on an optional task, ii) *which* optional task to select based on the problem context, iii) *when* in the problem-solving process to select the optional task, and iv) *type* of optional task-steps completed by the learner. Thus, making these decisions will be informative of the intentional strategy followed by the learner. Fig. 2 (b) shows the actions of a learner who attempts two optional tasks. As seen here, the action space is considerably more complex. Note that not all steps in the optional tasks need to be completed and the learner is free to jump between optional and non-optional steps. One possible path where a learner has completed a partial number of optional task steps is illustrated in Fig. 2 (c).

3.3 BERT Model

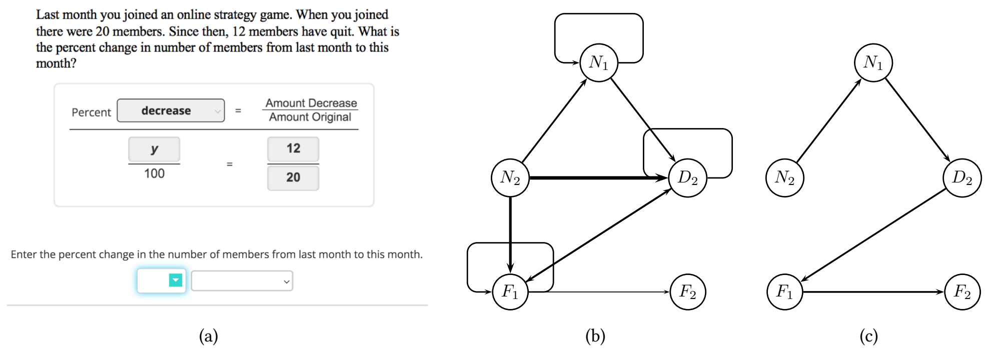
BERT [10] has been widely used for learning a representation for language understanding. Specifically, we use layers of *transformers* [30] to progressively transform an input sequence (in case of language, a sentence) and learn a vector representation or *embedding* for that sequence. In contrast to generative models such as GPT [3], the main advantage of BERT is that it uses *bi-directional* context to understand the input sequence and learn its representation. Specifically, when we consider generative models, they need to generate the next token from previously generated tokens. Therefore, they typically use a conditional model where a token is sampled conditioned on previous tokens. This means that they can only use uni-directional context, i.e., the model can only learn from *left-to-right.* However, it turns out that for understanding a sequence, bi-directional models are strictly more powerful than uni-directional models [10].

In our case, using BERT we learn a representation for the strategy from actions along both left-to-right and right-to-left directions. That is, we can look at actions performed by learners during the later stages in their problem-solving strategy to understand their thought processes during the earlier stages. Similarly, we can also use the model to understand how earlier decisions in their strategy can influence their subsequent actions. Combining the two perspectives yields a much more effective representation learning for strategies. Next, we describe in detail the following components of our approach, i) *pre-training* the model in an unsupervised manner to learn representations and ii) *fine-tuning* the model in a supervised manner after initializing it with the pre-trained model.

3.3.1 Pre-training.

We pre-train our model in an unsupervised manner using the Masked Language Model (MLM) approach. Specifically, we mask steps in the sequence of actions and train using a learning objective that minimizes the loss over predicting the masked steps. To do this, we first encode our sequence as follows. Assume that a student s has worked on problem p performing the sequence of actions $a_1(s,p)\ a_2(s,p)\ \dots\ a_n(s,p)$. If $a_i(s,p)$ results in the completion of a step in p, then, we consider $a_i(s,p)$ as part of the sequence of steps that represents the strategy of s for solving p. We then tag $a_i(s,p)$ with meta-data about how the step was completed. Specifically, we add a tag corresponding to the *hint*, if the step was completed with the help of a hint from the ITS. Similarly, if the step had an error, we indicate that in a tag.

The typical MLM approach is to mask a small percentage of tokens uniformly at random and predict the masked tokens. To do this, the model encodes the tokens into embeddings through layers of transformers. The embeddings are then decoded back to generate a token sequence and the loss is computed using the difference between the original sequence and the decoded sequence. Note that the model is learning using an approach similar to the



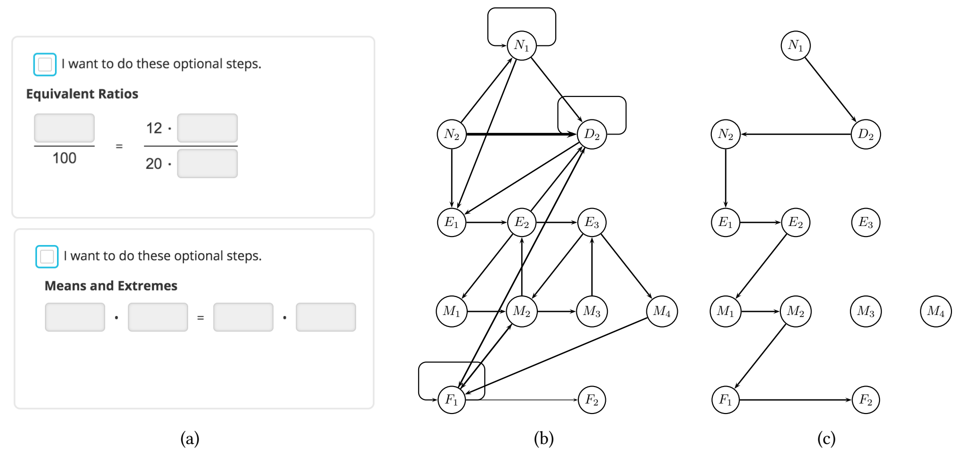
Last month you joined an online strategy game. When you joined there were 20 members. Since then, 12 members have quit. What is the percent change in number of members from last month to this month?

Percent decrease = Amount Decrease / Amount Original

y / 100 = 12 / 20

Enter the percent change in the number of members from last month to this month.

Figure 1: (a) Ratio problem in MATHia, the boxes are fillable slots that learners can complete in any order. (b) An illustration of actions performed by the learner in the form of a directed graph. N_1 indicates the numerator for the LHS in the first equation and N_2, D_2 indicates the numerator and denominator of the RHS. F_1, F_2 indicate the final answer steps. Loops indicate that students repeatedly tried the same step. The thickness of the arrow is an indicator of the number of times the student repeated the action of going from one step to the other. (c) A possible path in the action space. The arrows indicate an ordering of steps completed by the learner.



I want to do these optional steps.

Equivalent Ratios

/ 100 = 12 · / 20 ·

I want to do these optional steps.

Means and Extremes

· = ·

Figure 2: (a) Two optional tasks for the ratio problem in Fig. 1 (a). (b) An illustration of actions performed by the learner. $E_1 \ldots E_3$ indicates the optional task steps corresponding to equivalent ratios and $M_1 \ldots M_4$ indicates the optional tasks corresponding to the Means and Extremes. Note that learners can jump between steps of optional tasks and non-optional tasks. (3) An illustration of a possible path taken by a learner. Note that not all optional tasks need to be completed.

cloze task [29]. An illustration of MLM pre-training is shown in Fig. 3.

In our case, we want to regularize the pre-training to learn strategies that are more intentional. To do this, we use two approaches. First, we use prior knowledge about the student to sample from the data. Specifically, a student graduates from a MATHia workspace if they have shown mastery over skills within that workspace. On the other hand, students may be promoted from a workspace without achieving mastery. We assume that intentional strategies are more likely to be observed from graduated students as opposed to promoted students. Therefore, we bias the data distribution during pre-training where the majority of instances correspond to token sequences obtained from graduated students. We still want the model to learn the difference between intentional and unintentional strategies and therefore, similar to negative sampling in language modeling, we use a small number of instances from non-graduated/promoted students. Next, in typical pre-training, tokens are masked randomly. In our case, we mask optional steps jointly with non-optional steps. This has the effect that the pre-trained model is forced to learn the dependency between optional tasks and non-optional steps that signifies intentional strategies.

3.3.2 Model Architecture for Pre-training. The architecture that we use for pre-training is a multi-layer bidirectional Transformer encoder [30] which uses context in both directions. The main hyperparameters include the number of Transformer blocks (L), the

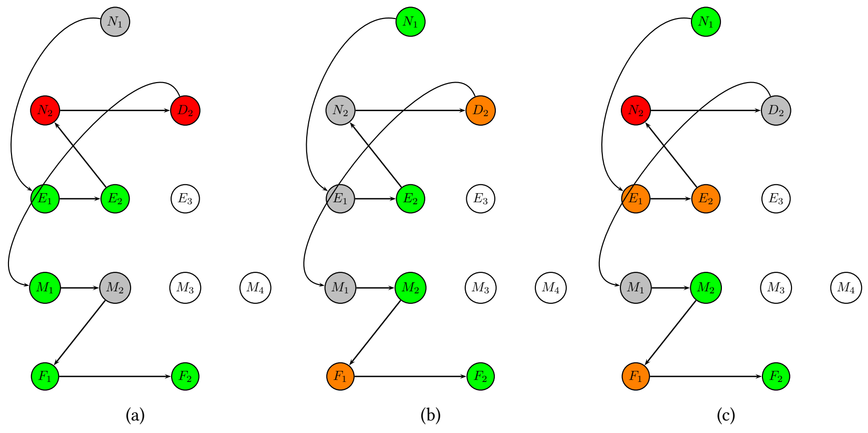


Figure 3: Illustrating MLM training for strategies in BERT. The red nodes indicate steps that are tagged with an error prior to completion, the orange nodes indicate steps for which a hint was used to complete the step and the green nodes indicate steps completed without errors or hints. As shown in (a), (b), (c), for MLM, we mask out different steps shown by gray-color and predict these steps based on the other non-masked steps.

size of the hidden representation (H), and the number of attention heads (A) within each block to implement the attention mechanism. For our BERT models, we use $L = 4$, $H = 64$, and $A = 8$. The input to the model always has a special character represented as $[CLS]$ followed by the sequence of action tokens and ends with a $[EOS]$ token, representing the end of the input sequence. Each token is then encoded by the Transformer blocks into embeddings. Let $T_i \in \mathbb{R}^H$ be the embedding for the i-th token. The embedding of the first token $[CLS]$ is a representation of the full sequence. Further, the model uses a special type of encoding called *positional encodings* [30] to encode the positions of tokens within the sequence. Positional encodings encode positions using a fixed-dimension vector. Thus, the i-th token in the sequence is encoded with positional encoding $P_i \in \mathbb{R}^H$. The overall representation for the i-th token in the sequence is the sum of the positional encodings and the token embeddings, $E_i = T_i + P_i$. This process is repeated by each transformer block in sequence until the final embeddings are obtained after the last block. One of the key advantages of representations learned by the pre-trained model is that they try to preserve *semantic equivalence* between sequences. That is, for sequences that are learned to be approximately similar (semantically), the embeddings learned for the $[CLS]$ tokens in both sequences will be close to each other. In our case, the pre-training separates the embeddings such that groups consist of strategies that are learned to be semantically similar to each other, and similar strategies are represented by the model using similar embeddings.

To pre-train our model, we mask tokens in a sequence by replacing the true token with a $[MASK]$ symbol. To ensure that the pre-training learns from the right strategies, we consider the majority of the training data from learners who have graduated from a workspace and to contrast the learning, we also provide the model with a small number of negative examples, i.e., strategies from promoted students. With a probability $1 - p$, we mask random tokens in the sequence and with probability p, we mask tokens based on the following template to force the model to jointly infer optional and non-optional steps. Specifically, we mask a pattern of steps $(S_1 \dots S_k)$ $(O_1 \dots O_k)$ $(S_1^* \dots S_k^*)$, where $(S_1 \dots S_k)$ are steps prior to the first optional task that the learner has chosen to work on, $(O_1 \dots O_k)$ are steps in the optional task and $(S_1^* \dots S_k^*)$ are the steps that occur after at least one or more optional task steps. As in the original BERT model, we mask no more than 15% of steps in the sequence. One technical issue is that we are introducing a new token $[MASK]$ which is not really a true token. This can cause a mismatch since we don't expect to see this type of token in sequences when we later fine-tune the model for specific predictive tasks. Therefore, we instead use an approach like BERT, where instead of always using the special symbol, we replace the token positions selected for masking with a randomly chosen token 10% of the time. We keep the original masked token 10% of the time and 80% of the time, we replace the original token with the $[MASK]$ token. The training is performed using the cross-entropy loss, where we predict the tokens at the masked positions from their embeddings using a *softmax* function. The loss, that is minimized, is specified in the equation below.

$$\ell(\Theta) = \sum_{i=1}^{N} \sum_{j=1}^{K} - \sum_{v} y_{ij}^{v} \log P_{\theta}(y_{ij}^{v} | \mathbf{X}_i) \tag{1}$$

where $\ell(\Theta)$ represents the loss of the model whose parameters are denoted by Θ, $\mathbf{X}_i$ is the i-th input sequence, y_{ij}^{v} is a binary indicator that is 1 if the token at position j for input sequence i is equal to v and 0 otherwise. $P_{\theta}(y_{ij}^{v} | \mathbf{X}_i)$ is the model's output, i.e., the probability that v is the j-th masked token for $\mathbf{X}_i$.

3.3.3 Fine-Tuning. We fine-tune the pre-trained model to understand if strategies can predict student learning outcomes. Specifically, in fine-tuning, we initialize the model with the pre-trained model parameters and then supervise it with a small number of labeled examples to train the model for a specific prediction task. To do this, we plug-in a classification layer on top of the original pre-trained model and train the new model end-to-end using a small number of labeled examples. Thus, the pre-trained model now adapts itself to minimize the predictive loss over the labeled examples. In our case, for an input $(\mathbf{X}_i, y_i)$, where $\mathbf{X}_i$ corresponds to the actions performed by student (s) on a specific problem (p), we assign the label $y_i = 1$ if s learned to solve p successfully and $y_i = 0$ otherwise. However, the notion of what counts as success for a learner may vary. For instance, obtaining the final answer could indicate learning. However, in an ITS like MATHia, learners repeatedly solve a step until they complete it which means all learners eventually obtain the correct answer. Therefore, we instead measure success based on whether a learner understood the concepts that they worked on during the optional task. Specifically, we check if their first attempt to provide the final answer to a problem *after* working on an optional step yields the correct answer. In these instances, we label $y_i = 1$ for fine-tuning the pre-trained model.

The architecture of the fine-tuning model is as follows. We use the pre-trained model to start with and connect the $[CLS]$ token embedding (which encodes the full strategy) to a softmax classifier. Thus, the strategy embedding acts as input for the classifier. Further, we add a hidden layer connected to the softmax classifier to encode a *prior* over skills. Specifically, each problem tests specific pre-defined skills (knowledge components). We estimate a prior over these skills using Bayesian Knowledge Tracing (BKT) [6]. For an instance where we want to predict if a student (s) learns to solve a problem (p) successfully, we use the BKT prior for s corresponding to all the skills tested in p. The illustration of our model is shown in Fig. 4. To train the model, we minimize the cross entropy loss as follows.

$$\ell(\hat{\theta}) = -\sum_{i=1}^{N} y_i \log(\hat{P}_\theta(\mathbf{X}_i) + (1 - y_i)\log(1 - \hat{P}_\theta(\mathbf{X}_i)) \qquad (2)$$

where N is the number of instances used for fine-tuning, y_i is the label for $\mathbf{X}_i$, $\hat{\theta}$ are the parameters of the model and $\hat{P}_\theta(\mathbf{X}_i))$ is the probability output by the model for $\mathbf{X}_i$.

4 Experiments

4.1 Implementation

4.1.1 Dataset. For this study, we collected data from MATHia workspaces related to ratio and proportion problems in 7^{th}-grade math. We collected a total of around 414K instances, where each instance corresponds to the sequence of steps followed by a student when working on a specific problem within the workspace. Each instance is uniquely identified by a student, problem pair. We collected data from 27946 unique students. The total number of unique problems in the workspace was around 280. Data from a total of 655 schools were included in our study. The data format is similar to the tutor interaction format available in the PSLC datashop [28]. Specifically, we record school ID, student ID, and problem name (all of which are anonymized), and a log of all the actions performed by the student. Note that we did not collect personally identifiable information in the data. Thus, we did not have access to demographic information even at the school-level. However, generally speaking, the school districts that use MATHia represent diverse demographic groups. In future, we plan to conduct a more fine-grained study of the effectiveness of our models across these groups. The log data that we collected contains the following information. Timestamp, the step they are working on within the problem, any possible hints that were requested, whether the student completed the step without error, the skills (knowledge components) that are tested within the problem, and if the student graduated or was promoted without graduating.

Each problem included a choice of two optional tasks. Each optional task corresponds to a specific concept in ratio and proportions. Specifically, one of them is related to computing proportions using *Equivalent Ratios* (ER) and the other is related to computing proportions using *Means and Extremes* (ME). We also pre-processed the dataset to remove auto-completed entries since these are not part of a strategy. That is, in several cases, MATHia auto-completes certain steps when the student completes a problem and we do not want these to be included as part of the student's strategy. Further, in some cases, students work out initial examples with step-by-step guidance from MATHia. We removed these since they did not constitute independent work by the learner. Finally, in a small number of cases, the interactions are incomplete possibly due to technical errors. We removed all such instances from our data.

4.1.2 Pre-training Model. We used the publicly available BERT implementation in our experiments. We customized the BERT architecture as follows. We used 8 self-attention heads and 4 transformer encoder blocks. We used only 64 dimensions for the embeddings (compared to the 512 used in the original model) since the size of the vocabulary is more limited compared to language models and this significantly reduces the pre-training time. The overall model consists of approximately $200K$ trainable parameters. To pre-train the model, we used training data from 100 schools that had the largest number of instances in the dataset. The total number of instances was equal to 240K. Of these around 160K corresponds to students who graduated from the workspace and 80K corresponds to students who did not graduate (also called promoted students). We sampled the instances such that 90% of the data corresponds to graduated students and 10% corresponds to data from promoted students which serves as negative examples in pre-training. The number of instances we used in pre-training is 130K with around 20K validating the pre-training. That is, we estimate the validation loss based on how well the model can predict masked tokens in the validation data. In each instance, we masked at most 15% of the steps similar to the approach used in the original BERT model. We stopped pre-training when the validation loss was below a threshold. We set the maximum sequence length as 128 and for sequences where the number of steps exceeded this limit, we truncated the sequences to the maximum limit. To perform pre-training, we used two NVIDIA Tesla T4 GPUs in parallel on the Google Vertex AI platform. The pre-training took around 7 hours to complete. The pre-trained models and code is available here [1].

4.1.3 Fine-tuning Model. We initialized the fine-tuning model with the pre-trained model. We then added a hidden linear layer with 50 units on top of the $[CLS]$ token which encodes the strategy embedding followed by a softmax classifier. Further, for a training example corresponding to student s and problem p, we added a

[1]https://github.com/abisha-thapa/education_bert

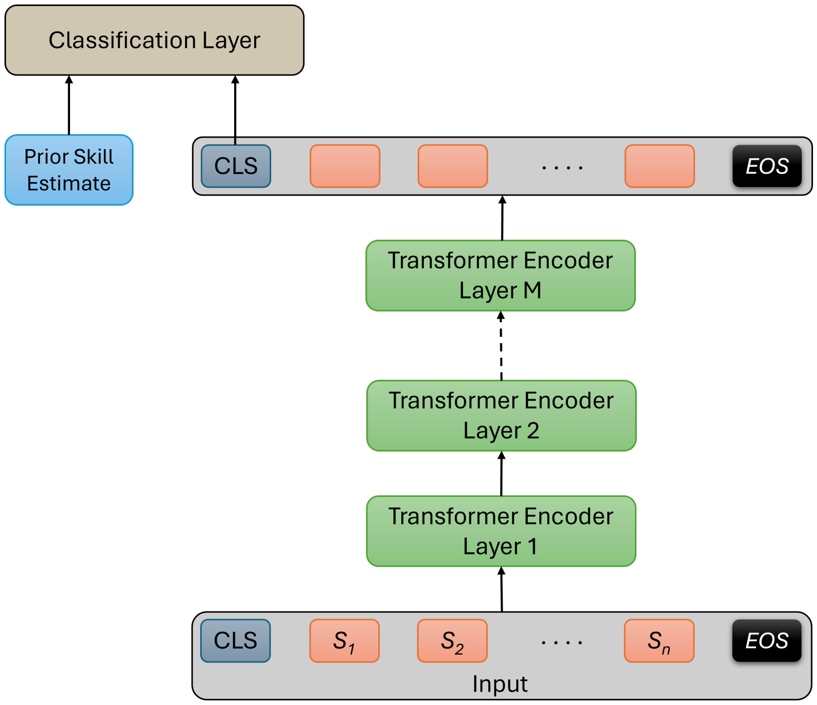
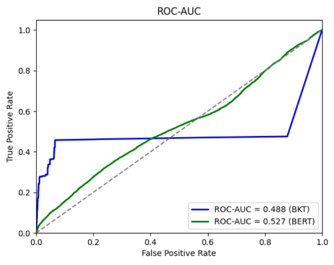

Classification Layer
Prior Skill Estimate
CLS
....
EOS
Transformer Encoder Layer M
Transformer Encoder Layer 2
Transformer Encoder Layer 1
CLS
S_1
S_2
....
S_n
EOS
Input

Figure 4: Illustrating the fine-tuning model architecture. The input sequence of steps completed by a student for a problem is encoded by several transformer layers in sequence to produce embeddings for each of the steps in the input. The start token is *CLS* and the end of the sequence is denoted by *EOS* since we have variable length sequences. The *CLS* token embedding after the last transformer layer encodes the strategy. This is input to a classification layer along with a prior estimate of skills for that student. The full model is trained end-to-end to minimize the cross-entropy loss.

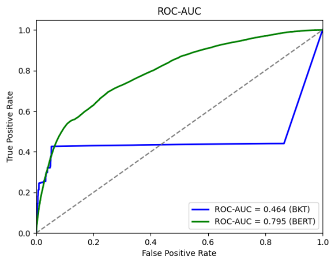
Pre-Training Data			Fine-Tuning Data			
#Schools	#Training Instances	#Validation Instances	#Schools	#High-GR Schools	#Low-GR Schools	#Instances
100	130K	20K	555	264	291	120K

Table 1: Summary of data used in pre-training and fine-tuning. Each instance corresponds to a unique pair s, p, where a student s works on a problem p within the workspace. High-GR refers to schools where at least 80% of students have graduated and Low-GR refers to schools where $< 80\%$ of students graduated from the workspace.

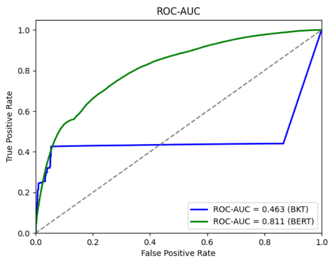
prior for s using a skill vector. Each dimension in the skill vector corresponds to a specific skill in the workspace. We masked all the skills unused in p and for the skills used in p, we set the value as the BKT probability estimate for that skill estimated prior to s attempting p. Further, we also concatenated the skill vector with a vector that encodes changes in the BKT estimates prior to working out the current problem. Specifically, for a skill that is used in p, we compute the prior change in BKT estimate the last time the student attempted to use that skill in a previous problem. For fine-tuned training, we used the following settings. Batch-size = 50, learning rate = $1e{-}05$ and the Adam optimizer. From the 555 schools that were not used for pre-training, we had 120K instances. We considered a school where more than 80% of the students graduate from the workspace as one with a *high graduation rate* and the others as schools with a *low graduation rate*. We had a total of 264 schools with a high graduation rate and the remaining 291 schools had a low graduation rate. The number of instances from high graduation rate schools was around 74K and the number of instances from low graduation rate schools was around 46K. During fine-tuning, we used a small number of labeled examples (we show results varying the number of examples used for fine-tuning) and tested the fine-tuned model on the remaining examples. For fine-tuning, we used data from schools that had a high graduation rate. We do this since data from such schools may show the application of strategies more accurately. This ensures that the fine-tuning does not have added noise due to improper use of strategies. Fine-tuning is much faster compared to pre-training since the pre-trained model has already learned strategy representations. Therefore, for the fine-tuning, we only required a single Tesla T4 GPU. The fine-tuning took less than 25 minutes. A summary of the datasets used in pre-training and fine-tuning is shown in Table 1.



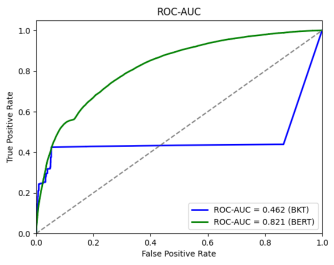
(a) 1% FT



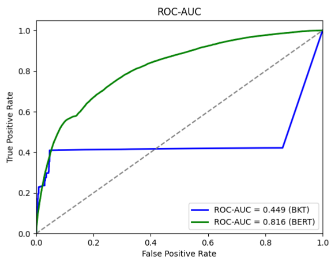
(b) 5% FT



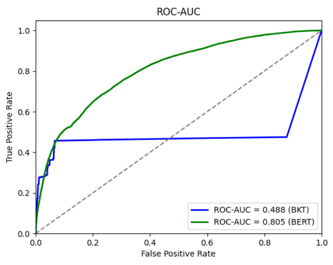
(c) 10% FT



(d) 20% FT



(e) 10% FT, High GR



(f) 10% FT, Low GR

Figure 5: ROC-AUC score comparison. In each case, $X\%$ FT indicates the percentage of data instances used as labeled examples for fine-tuning. The remaining data is used for testing. (a)- (d) shows the ROC-AUC comparing our fine-tuned model predictions (BERT) with the BKT prediction on the test instances. (e), (f) shows the results separately for 10% fine-tuning for high graduation rate (High GR) and low graduation rate (Low GR) schools in the test data.

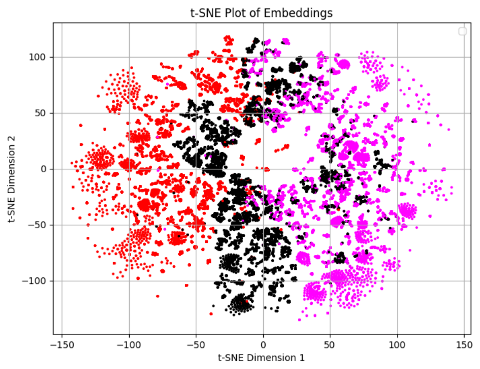
4.2 Fine-tuned Model Performance

We evaluate the performance of our fine-tuned model using ROC-AUC scores since our model outputs probabilities. Specifically, the ROC-AUC score quantifies how well our model can distinguish between strategies that lead to correct answers and those that do not over all possible thresholds. This dichotomy between strategies is useful in understanding strategies in a bottom-up manner (from the data) and can potentially be applicable to classroom practice where students can learn to develop better strategies. In future work, we plan to develop field trials in MATHia to test different strategies.

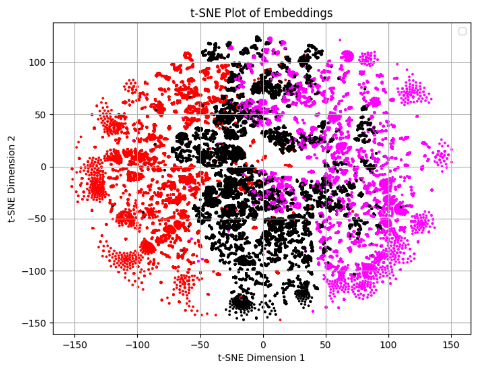
Since we use prior estimates from BKT within our fine-tuning, to evaluate significance of the strategy representations that we learn, we perform an ablation study as follows. We use the correctness probability for the skill corresponding to the final answer in a problem as the BKT prediction for how likely the learner is to correctly solve that problem, and compare this with the predicted probabilities from our fine-tuned model. Fig. 5 (a) - (d) shows our results for varying number of labeled examples. As seen here, the ROC-AUC score increases significantly when we go from 1% of labeled examples (among all instances in the 555 schools not used in pre-training) to just 5%. The ROC-AUC score when we use 10% instances is equal to 0.81. This is significantly higher than using just the BKT probability estimates which clearly demonstrates that the strategy representations learned from the pre-training are highly informative for the fine-tuned model. Further, due to the strategy representation learned during pre-training, we require only a small number of labeled examples for fine-tuning. Fig. 5 (e)-(f) shows the ROC-AUC scores separately for high and low graduation rate schools in the test portion of the data (assuming 10% of instances are used in fine-tuning). In fine-tuning, as mentioned in the previous section, we only use data from high graduation rate schools. We wanted to verify that this does not bias predictions in low graduation rate schools. As seen in Fig. 5 (e)-(f), for both cases our ROC-AUC scores are quite similar with a slight decrease in the low graduation rate schools. Thus, the fine-tuned model is not biased which indicates that the embeddings learned during pre-training represent strategies that generalize across both types of schools.

4.3 Analyzing Embeddings

Fig. 6 shows a visualization of the embeddings learned by the pre-trained model using the t-SNE plot. Specifically, we project the 64-dimension embedding into 2 dimensions. We color each instance based on whether the student has made use of one or more optional tasks to solve the problem. We show separate t-SNE plots for instances corresponding to high graduation rate and those that correspond to low graduation rate schools. In Fig. 6 (a) we observe that the instances that use both tasks have more separation than those that use a single task when compared to their separation in Fig. 6 (b). The separation indicates that the pre-trained model has learned more distinct embeddings for the instances where a single optional task is used compared to the ones where both optional tasks are attempted. In general, to apply a strategy, one may need to choose between following the ER or ME approach. In some cases,



(a)



(b)

Figure 6: Visualization of embedding learned by the pre-trained model using a t-SNE plot. (a) shows the t-SNE plot for instances from schools with high graduation rate and (b) shows the t-SNE plot for instances from schools with low graduation rate. The red and pink colors show that the instance corresponds to the one where exactly one of the optional tasks is used by the student. Specifically, the red instances are sequences where the optional task corresponding to ER has been used and the pink instances are sequences where ME has been used. The black color corresponds to instances where the student uses both optional tasks.

the ER approach may be easier to execute a strategy with compared to ME and vice versa. However, in general, a choice made between the two indicates a more intentional strategic thinking of the student. In Fig. 6 (a) that corresponds to data from schools with high graduation rate, we see that the representation learned for the more optimal strategy (i.e., following a single optional task) is better separated from the one that is sub-optimal. On the other hand, in Fig 6 (b), which corresponds to data from lower graduation rates, this separation is less distinct. This indicates that the pre-trained model is able to uncover this structure within the data in an unsupervised manner.

4.4 Calibrating the Fine-tuned Model

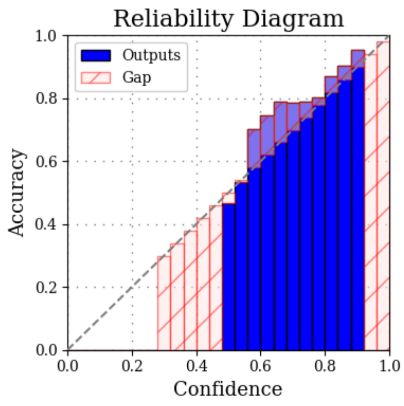
Recall that we train the fine-tuned model using the cross-entropy loss in Eq. (2). Thus, the softmax activation function in our fine-tuned model outputs probabilities. For our model to be well-calibrated, this probability should reflect the likelihood in the ground truth. To improve reliability in probabilities predicted by our fine-tuned model, we calibrate it using the *temperature scaling* approach in [13] by rescaling probabilities using a validation set. In our case, we use 1000 instances from high graduation rate schools to calibrate our model.

Fig. 7 shows the result of the calibration using a reliability diagram and confidence histogram. In the reliability diagram, the x-axis represents the confidence scores (divided into confidence intervals or bins) and the y-axis represents the accuracy of the model (for the different bins). The model is said to be perfectly calibrated if it is equally as confident as it is accurate, in which case the diagram should plot an identity function (or follow the diagonal). For well-calibrated models, accuracy and confidence should be closer to each other in the confidence histogram. To quantify the calibration error as a scalar value, we use Expected Calibration Error (ECE) which approximates the miscalibration between expected confidence and accuracy. We used a 25 bins and computed the prediction confidence for all instances in the test set (except for 10% labeled examples in fine-tuning). Fig 7 (a) and (c) are the plots for the uncalibrated model where we see in the reliability diagram that the plot drifts from the diagonal. Similarly, in the confidence histogram, we see that the average confidence and accuracy are further apart. The ECE for the uncalibrated model is 0.089. Fig 7 (b) and (d) show the post-calibration results. Here, in the reliability diagram, the plot closely follows the diagonal and the confidence histogram shows that the average confidence and accuracy are much closer. The ECE for the calibrated model is 0.078. This shows that the model is now more confident of its predictions.

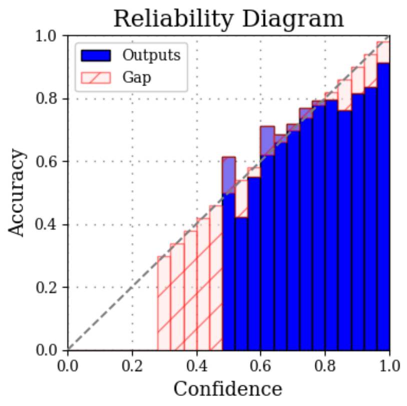
Uncertainty Analysis. We analyzed the probabilities output by our model to understand the types of strategies for which the model has greater uncertainty. Here, we sampled 1K instances each from the highest probabilities, lowest probabilities and mid-point of the distribution. We analyze for each of these, what percentage of instances performed steps from a single optional task. As shown in Fig. 8 (a), it turns out that using a single optional task corresponds to the model having lower uncertainty about the outcome of the strategy. However, if steps from both optional tasks were completed, the model has greater uncertainty since this could be an indicator that the learner is confused. In addition, Fig. 8 (b) shows the average number of actions the learner performed as a function of the probabilities output by the calibrated fine-tuned model over the full test data. As seen here, when the average number of actions increase, the model uncertainty is larger since these instances correspond to less efficient strategies.

5 Conclusion

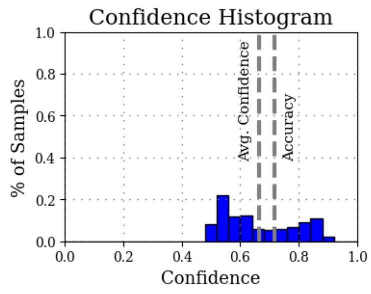
Problem-solving to a large extent is choosing and executing the strategy that is appropriate in the problem context. In this work,



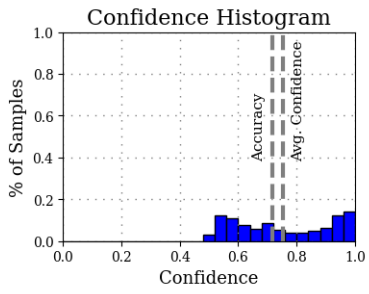
(a)



(b)

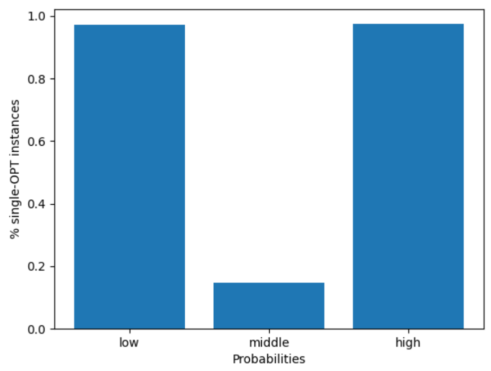


(c)

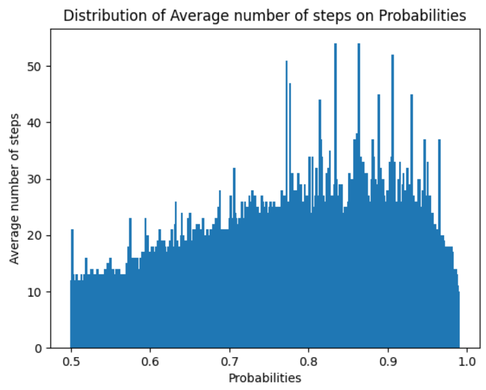


(d)

Figure 7: Illustrating the results of calibrating the fine-tuned model. Fig (a) and (b) show the reliability diagram before and after calibration for the test instances corresponding to the fine-tuning with 10% labeled examples. The closer to the diagonal the accuracy-confidence values are, the better the calibration. Fig (c) and (d) show the confidence histogram before and after calibration. The closer the average confidence and accuracy are, the better the calibration.



(a)



(b)

Figure 8: (a) shows the percentage of instances (among those test instances for which our model outputs the highest/lowest/closest-to-middle probabilities) that use steps from a single optional task. (b) shows the average number of steps in test instances for the probabilities output by our model, shown in the x-axis.

we developed an approach motivated by a key educational goal in math learning, i.e., helping students develop *metacognitive awareness* of when particular problem-solving strategies are efficient. Understanding how these strategies develop and how to help students develop these strategies that can be applied across different contexts is relevant to classroom practice. To this end, the approach that we proposed in this work develops foundational models that use advances in AI to represent strategies by analyzing large-scale data. Specifically, we used BERT, a state-of-the-art approach traditionally used for language modeling, and learned embeddings for strategies based on sequences of actions from student data. Using these embeddings, we developed a fine-tuned model that predicts if learners can apply strategies correctly to solve new problems, thus separating strategies that work well in practice from those that may not. We developed our approach using data from the MATHia ITS from over 600 schools where the ITS was used for a 7^{th} grade math workspace. We showed that our pre-trained model could be fine-tuned with a small number of labeled examples to make calibrated and accurate predictions.

Our future plan is to utilize insights from our model to conduct field trials within MATHia by guiding students towards strategies that are considered to be more effective based on our model and conduct A/B testing to validate if the AI model can help improve learning outcomes such as time to mastery and student engagement. Further, we also plan to generalize our approach that we grounded in the domain of ratio and proportions over other domains. However, note that unlike typical AI applications such as language processing where the structure of language remains similar in books, webpages, etc., in our case, the strategies are highly dependent on the domain. Thus, the AI needs to be aware of the domain that it is learning from as well as the interface used within that domain to learn effective strategy models.

Acknowledgments

This research was supported by an award from the Bill and Melinda Gates Foundation and NSF award #2008812. The opinions, findings, and results are solely the authors' and do not reflect those of the funding agencies.

References

[1] Gautam Biswas, James R. Segedy, and Kritya Bunchongchit. 2016. From Design to Implementation to Practice a Learning by Teaching System: Betty's Brain. *International Journal of Artificial Intelligence in Education* 26, 1 (2016), 350–364.
[2] Kristy Elizabeth Boyer, Eun Young Ha, Robert Phillips, Michael D. Wallis, Mladen A. Vouk, and James C. Lester. 2010. Dialogue act modeling in a complex task-oriented domain. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue.* 297–305.
[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *CoRR* abs/2005.14165 (2020). arXiv:2005.14165 https://arxiv.org/abs/2005.14165
[4] Whitney L. Cade, Jessica L. Copeland, Natalie K. Person, and Sidney K. D'Mello. 2008. Dialogue Modes in Expert Tutoring. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS).* 470–479.
[5] Albert T. Corbett. 2001. Cognitive Computer Tutors: Solving the Two-Sigma Problem. In *Proceedings of the 8th International Conference on User Modeling 2001.* 137–147.
[6] Albert T Corbett and John R Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction* 4 (1994), 253–278.
[7] Ryan Shaun Joazeiro de Baker, Albert T. Corbett, and Angela Wagner. 2006. Human Classification of Low-Fidelity Replays of Student Actions. In *Proceedings of the Educational Data Mining Workshop at the 8th International Conference on Intelligent Tutoring Systems.*
[8] Ryan Shaun Joazeiro de Baker and Adriana M. J. B. de Carvalho. 2008. Labeling Student Behavior Faster and More Precisely with Text Replays. In *Educational Data Mining 2008, The 1st International Conference on Educational Data Mining, Montreal, Québec, Canada, June 20-21, 2008. Proceedings*, Ryan Shaun Joazeiro de Baker, Tiffany Barnes, and Joseph E. Beck (Eds.). www.educationaldatamining.org, 38–47.
[9] Ryan Shaun Joazeiro de Baker, Antonija Mitrovic, and Moffat Mathews. 2010. Detecting Gaming the System in Constraint-Based Tutors. In *User Modeling, Adaptation, and Personalization, 18th International Conference, UMAP 2010, Big Island, HI, USA, June 20-24, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6075)*, Paul De Bra, Alfred Kobsa, and David N. Chin (Eds.). Springer, 267–278.
[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805
[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186.
[12] Kristen E. DiCerbo and Khusro Kidwai. 2013. Detecting Player Goals from Game Log Files. In *Proceedings of the 6th International Conference on Educational Data Mining, Memphis, Tennessee, USA, July 6-9, 2013*, Sidney K. D'Mello, Rafael A. Calvo, and Andrew Olney (Eds.). International Educational Data Mining Society, 314–315.
[13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1321–1330. https://proceedings.mlr.press/v70/guo17a.html
[14] John S. Kinnebrew and Gautam Biswas. 2012. Identifying Learning Behaviors by Contextualizing Differential Sequence Mining with Action Features and Performance Evolution. In *Proceedings of the 5th International Conference on Educational Data Mining, Chania, Greece, June 19-21, 2012*, Kalina Yacef, Osmar R. Zaïane, Arnon Hershkovitz, Michael Yudelson, and John C. Stamper (Eds.). www.educationaldatamining.org, 57–64.
[15] John S. Kinnebrew, Kirk M. Loretz, and Gautam Biswas. 2013. A Contextualized, Differential Sequence Mining Method to Derive Students' Learning Behavior Patterns. *Journal of Educational Data Mining* 5, 1 (2013), 190–219.
[16] Kenneth R. Koedinger, Albert T. Corbett, and Charles Perfetti. 2012. The Knowledge-Learning-Instruction Framework: Bridging the Science-Practice Chasm to Enhance Robust Student Learning. *Cogn. Sci.* 36 (2012), 757–798.
[17] Nabin Maharjan, Dipesh Gautam, and Vasile Rus. 2018. Assessing Free Student Answers in Tutorial Dialogues Using LSTM Models. In *Artificial Intelligence in Education - 19th International Conference, AIED.* 193–198.
[18] Gyöngyvér Molnár, Samuel Greiff, and Benő Csapó. 2013. Inductive reasoning, domain specific and complex problem solving: Relations and development. *Thinking Skills and Creativity* 9 (2013), 35–45.
[19] Allen Newell and Herbert Simon. 1972. *Human Problem Solving.* Prentice Hall.
[20] Luc Paquette, Adriana M. J. B. de Carvalho, Ryan S. Baker, and Jaclyn Ocumpaugh. 2014. Reengineering the Feature Distillation Process: A case study in detection of Gaming the System. In *Proceedings of the 7th International Conference on Educational Data Mining, EDM 2014, London, UK, July 4-7, 2014*, John C. Stamper, Zachary A. Pardos, Manolis Mavrikis, and Bruce M. McLaren (Eds.). International Educational Data Mining Society (IEDMS), 284–287.
[21] Steve Ritter. 1997. Communication, cooperation and competition among multiple tutor agents. In *Artificial Intelligence in Education: Knowledge and media in learning systems.* 31–38.
[22] Steve Ritter, Ryan Baker, Vasile Rus, and Gautam Biswas. 2019. Identifying Strategies in Student Problem Solving. *Design Recommendations for Intelligent Tutoring Systems* 7 (2019), 59–70.
[23] Elizabeth Rowe, Ryan S. Baker, Jodi Asbell-Clarke, Emily Kasman, and William J. Hawkins. 2014. Building Automated Detectors of Gameplay Strategies to Measure Implicit Science Learning. In *Proceedings of the 7th International Conference on Educational Data Mining, EDM 2014, London, UK, July 4-7, 2014*, John C. Stamper, Zachary A. Pardos, Manolis Mavrikis, and Bruce M. McLaren (Eds.). International Educational Data Mining Society (IEDMS), 337–338.
[24] Vasile Rus, Sidney K. D'Mello, Xiangen Hu, and Arthur C. Graesser. 2013. Recent Advances in Conversational Intelligent Tutoring Systems. *AI Magazine* 34, 3 (2013), 42–54.
[25] Vasile Rus, Nabin Maharjan, Lasang Jimba Tamang, Michael Yudelson, Susan R. Berman, Stephen E. Fancsali, and Steven Ritter. 2017. An Analysis of Human

Tutors' Actions in Tutorial Dialogues. In *Proceedings of the International Florida Artificial Intelligence Research Society Conference*. 122–127.

[26] Anup Shakya, Vasile Rus, and Deepak Venugopal. 2021. Student Strategy Prediction using a Neuro-Symbolic Approach. In *Proceedings of the 14th International Educational Data Mining Conference (EDM 21)*.

[27] Anup Shakya, Vasile Rus, and Deepak Venugopal. 2023. Scalable and Equitable Math Problem Solving Strategy Prediction in Big Educational Data. In *Proceedings of the 16th International Educational Data Mining Conference (EDM 23)*.

[28] John C. Stamper, Kenneth R. Koedinger, Ryan Shaun Joazeiro de Baker, Alida Skogsholm, Brett Leber, Sandy Demi, Shawnwen Yu, and Duncan Spencer. 2011. DataShop: A Data Repository and Analysis Service for the Learning Science Community. In *AIED*, Vol. 6738. 628.

[29] Wilson L. Taylor. 1953. "Cloze Procedure": A New Tool for Measuring Readability. *Journalism & Mass Communication Quarterly* 30 (1953), 415 – 433. https://api.semanticscholar.org/CorpusID:206666846

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[31] Deepak Venugopal and Vasile Rus. 2016. Joint Inference for Mode Identification in Tutorial Dialogues. In *COLING 2016, 26th International Conference on Computational Linguistics*. ACL, 2000–2011.

[32] J. Wong, Mohammad Khalil, M. Baars, B. D. Koning, and F. Paas. 2019. Exploring sequences of learner activities in relation to self-regulated learning in a massive open online course. *Computers & Education* (2019).