Quizzing Policy Using Reinforcement Learning for Inferring the Student Knowledge State

Joy He-Yueya
University of Washington
joyhe@cs.washington.edu

Adish Singla
MPI-SWS
adishs@mpi-sws.org

ABSTRACT

The prevalence of online education systems provides opportunities to deliver personalized learning at scale. Educational systems need to assess students so that they can provide better curricula tailored to each student's unique needs. Since there is a limited amount of time for quizzing a student, we need to test each student using those questions that capture the most information about their level of understanding of various concepts. In this paper, we formally pose the problem and present multiple approaches for learning a quizzing policy to determine a personalized sequence of questions for each student that best predicts their knowledge state. We first introduce simple heuristics including random selection and an uncertainty sampling approach inspired by an active learning framework. We then develop a reinforcement learning (RL) approach for designing a quizzing policy. Using simulations of students' knowledge states, we provide initial evidence that an RL-based approach can improve over simple heuristics. We further demonstrate the effectiveness of our approaches using a real-world dataset consisting of over 1.5 million examples of students' answers to mathematics questions from Eedi, an online educational platform.

Keywords

reinforcement learning, knowledge state, quizzing policy

1. INTRODUCTION

Online education systems are making high-quality education more accessible for students across the globe. These systems provide various educational resources such as instructional videos and exercises. To provide personalized curricula for improving the learning outcomes of students, an online education system needs to accurately infer each student's knowledge state (i.e., their level of understanding of various concepts) by quizzing them. This is a challenging task because the quizzing time is limited. To make the most efficient use of each student's time, it is important to prioritize those questions that reveal the most information about the student's knowledge.

We focus on a specific goal for student assessment: *given a limit to the number questions we are allowed to ask each student, how can we determine a sequence of questions for each student that best predicts their knowledge state?* Specifically, when an education system needs to assess a student for inferring their knowledge, the system suggests a personalized question to query for the student and gets their response to the question. Based on the student's response history (i.e., a sequence of question-response pairs), the system selects another question to query for the student until it has exhausted its query budget (i.e., the maximum number of queries allowed). We refer to the function that provides the next question to query based on students' response histories as quizzing policy (QP).

We define the task of learning a QP in the context of the NeurIPS 2020 Education Challenge [27] launched by Eedi [6], an online educational platform with thousands of active users daily around the globe. We consider a set of 948 multiple-choice mathematics questions that correspond to 57 different concepts. Specifically, the task is to obtain a limited set of answers from each student for inferring the student's knowledge on the 57 concepts and then predict the student's performance on unseen questions based on the inferred knowledge state.

The key challenge in designing a QP is related to a crucial task in machine learning: active learning (AL). For many learning tasks (e.g., image classification, text classification), obtaining sufficient labeled data for training high-performance models is costly [16, 18, 32]. AL aims to reduce the amount of annotated data needed by having the model carefully select which data points should be labeled.

Existing methods for AL include heuristics such as selecting the data points about which the model is most uncertain (i.e., uncertainty sampling) [15, 26, 31, 24], picking the instances about which a set of possible different models disagree the most (i.e., query by committee) [23, 10], or choosing the example that can lead to the most immediate improvement in model performance (i.e., estimated error reduction) [22, 12].

In addition to these heuristics for AL, recent studies [30, 19, 9] have explored how to use reinforcement learning (RL) to learn the AL strategy itself. RL [20, 25] is a powerful

framework where an agent learns how to make good decisions (actions) in different situations (states) through trial and error. In the RL terminology, the action space provides the set of actions that can be taken by the RL agent at a given point in time; the state space defines the "state of the world" that is visible to the RL agent; and the reward function assigns a value to the outcome of each action taken by the RL agent. In this case, the set of possible instances to be labeled defines the action space; the state space is a representation of the sequence of instances that have already been annotated; and the gain in prediction accuracy as a result of an action defines the reward. The RL agent learns to improve its decision-making over time based on the reward signals it receives. Inspired by these studies, we investigate using RL to learn a QP for personalized student assessment.

1.1 Our approach and contributions

In this paper, we formalize the problem of learning a QP for inferring the student knowledge state and present several different approaches including simple heuristics and an RL-based approach. Our contributions are:

- We formulate the problem of learning a QP to infer student knowledge.
- We propose simple heuristics (i.e., random selection, uncertainty sampling) and an RL-based approach for learning a QP.
- We evaluate the performance of different QPs on a synthetic dataset and a publicly available dataset consisting of over 1.5 million examples of students' answers to mathematics questions from Eedi.

For the reproducibility of experimental results and facilitating research in this area, the code and dataset are publicly available.[1]

1.2 Related work

AL is a popular methodology in machine learning that aims to reduce the amount of annotated data needed by having the model carefully select which data points should be labeled. The task of designing a QP is closely related to AL because the goal is to optimally select a set of questions to ask students to gain the most information about their knowledge states. Uncertainty sampling [15, 26, 31, 24] is one of the most popular heuristics for AL because it is straightforward and computationally efficient. Specifically, it suggests labeling instances that are closest to the model's decision boundary (i.e., the most uncertain). Woodward and Finn [30] propose the first application of RL to the task of AL for image classification. Other studies [19, 9] explore how to train an AL policy that can generalize across diverse datasets.

RL has also been applied to various tasks in education such as learning an instructional policy [2, 3, 5, 13, 17, 21, 28], learning a hint policy for helping students solve multi-step problems [7], and generating new educational tasks [1]. We introduce a different policy, a quizzing policy for inferring the student knowledge state, which has not been designed using RL in previous literature.

[1]https://github.com/joyheyueya/quizzing-policy

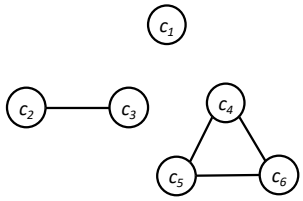

Figure 1: Graphical representation of knowledge. This is an example of an undirected graph where each node (circle) represents a concept, and each edge connects a pair of similar concepts: c_1 is an independent concept, c_2 and c_3 are similar, and c_4, c_5, and c_6 are similar.

There is prior work on the efficient assessment of knowledge [8]. Our student knowledge model is inspired by the knowledge components (i.e., concepts / skills) used in Bayesian Knowledge Tracing (BKT) [4], which represents the state for each knowledge component as a binary variable: 1 if the knowledge component is known, 0 otherwise.

2. PROBLEM FORMULATION

In this section, we formalize the problem of learning a quizzing policy (QP) for inferring the student knowledge state.

2.1 Student knowledge state

Our goal is to infer student knowledge on a set of n concepts $C = \{c_1, ..., c_n\}$ associated with a set of m questions $X = \{x_1, ..., x_m\}$. For simplicity, each question corresponds to a single concept, but each concept might be associated with more than one question ($m >> n$). A student's knowledge state h is defined as $h = [v_1, ..., v_n]$ where $v_1, ..., v_n$ are binary variables that indicate whether or not the student knows each concept in C: $v_i = 1$ if c_i is known, and $v_i = 0$ otherwise. Formally, we define a hypothesis space $\mathcal{H}$ for all possible knowledge states: $\mathcal{H} = \{0, 1\}^n$. We assume h is fixed during the assessment.

2.2 Graphical representation of knowledge

We consider two assumptions that are useful for inferring the student knowledge state: 1) difficult concepts are more likely to be unknown, and easy concepts are more likely to be known; 2) similar concepts are more likely to have the same value (i.e., a student who knows one concept is also likely to know the other concepts that are similar to the one that is already known). These influences can be represented by an undirected graph where each node corresponds to a concept, and each edge connects a pair of concepts that are similar (see Figure 1). In the Eedi dataset (described in Section 4.2.1), we consider every pair of concepts that share the same super-concept to be similar (e.g., there is an edge between "Rearranging Formula and Equations" and "Substitution into Formula" because they are both under the same super-concept "Formula"). Based on this graphical structure, we model a student's knowledge state using a Markov Random Field (MRF).

An MRF is a probability distribution over a set of variables that satisfy certain properties defined by an undirected graph. In our case, we define a probability distribution p over binary variables $v_1, ..., v_n$ defined by an undirected graph $G = (V \cup F, E)$ where V is the set of nodes (con-

cepts), F is the set of factors that define a set of functions over the variables that they are connected with, and E is the set of edges (see Figure 2).

An MRF allows us to calculate the probability of each way of assigning values to binary variables $v_1, ...v_n$, which represent the knowledge state of the corresponding concepts $c_1, ..., c_n$. The probability p has the form:

$$p(v_1, ..., v_n) = \frac{1}{Z} \prod_{\psi_\alpha \in F} \psi_\alpha(v_\alpha) \quad (1)$$

where α represents a subgraph of G, and ψ_α denotes a factor that defines a non-negative function over the set of variables v_α in α. Z is a normalizing constant that ensures the distribution sums to one:

$$Z = \sum_{v_1, ...v_n} \prod_{\psi_\alpha \in F} \psi_\alpha(v_\alpha) \quad (2)$$

We specify factors for an MRF based on two assumptions about variables $v_1, ..., v_n$. For our first assumption that difficult concepts have a higher probability of being unknown, we define unary factors:

$$\psi_i(v_i) = \begin{cases} 1 - \textit{difficulty}_{c_i} & \text{if } v_i = 1 \\ \textit{difficulty}_{c_i} & \text{otherwise} \end{cases}$$

where $\textit{difficulty}_{c_i}$ is a real number that represents the difficulty of the concept c_i that v_i corresponds to, and $0 \leq \textit{difficulty}_{c_i} \leq 1$. A higher $\textit{difficulty}_{c_i}$ value means c_i is more difficult.

For our second assumption that variables corresponding to similar concepts are more likely to have the same values, we define binary factors between every pair of nodes (v_i, v_j) that are connected by an edge in graph G:

$$\psi_{\{i,j\}}(v_i, v_j) = \begin{cases} \textit{influence} & \text{if } v_i = v_j \\ 1 - \textit{influence} & \text{otherwise} \end{cases}$$

where *influence* represents a constant that satisfies $0.5 \leq \textit{influence} \leq 1$. A greater *influence* value means we want to assign a higher probability to an assignment that gives the same values to variables corresponding to similar concepts. In our work, we fix *influence* to be 0.7. We also tried similar values, and they lead to similar results.

2.3 Quizzing policy for knowledge inference

Since there is a cost associated with each question we query students (e.g., time, student's energy), we need to select a limited number of questions that reveal the most about their knowledge state. Thus, student knowledge prediction can be framed as a pool-based active learning (AL) task with a given query budget T. For simplicity, we assume querying each exercise leads to the same cost and define T to be the total number of queries we are allowed.

We describe the AL framework in detail, see Algorithm 1. At a given time step t, we have a labelled set L that consists of all the questions we have asked the student and their responses. Formally, $L = \{(x^i, y^i)\}_{i=1}^t$ where $x^i \in X$, and $y^i \in \{0, 1\}$ is the student's response to x^i ($y^i = 0$ if the response is incorrect, $y^i = 1$ if the response is correct). We also have an unlabelled set U consisting of all the questions that we have not asked). Based on L, we have a belief B_h about the student's knowledge h. Formally, $B_h = [b_1, ..., b_n]$ where b_i is the probability of knowing the concept c_i (i.e., $v_i = 1$ with a probability of b_i). We define $Binary(B_h)$ as a function that converts probabilities into binary values using a threshold of 0.5 (1 if $b_i \geq 0.5$ and 0 otherwise). $Binary(B_h)$ gives the inferred binary knowledge state. We update B_h based on L by running the Loopy Belief Propagation algorithm (LBP) [11] on our graph defined in Section 2.2. LBP takes L as input and outputs the probabilities $b_1, ..., b_n$ ($0 \leq b_i \leq 1$). Additionally, we have a QP that takes B_h as input and outputs the next question to ask the student. Specifically, a policy $\pi(\cdot|B_h)$ provides a probability distribution with support over all questions in U given B_h. We can then sample a question from $\pi(\cdot|B_h)$.

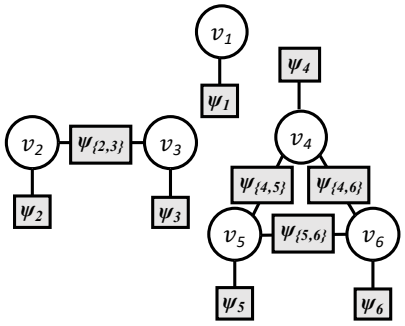

Figure 2: Modeling graphical student knowledge using MRF. This models the knowledge representation in Figure 1 as a factor graph. Each node v_i is a binary variable that represents the knowledge state of the corresponding concept c_i. Factors are represented by rectangles. There is a unary factor for every node and a binary factor between every pair of nodes connected by an edge to model the dependency between variables.

Algorithm 1: Active learning for inferring knowledge

Input: budget T, quizzing policy π
Output: $\hat{h}$
Initialize $L_0 \leftarrow \emptyset$, $U_0 \leftarrow \{x_i\}_{i=1}^m$
for $t = 1, 2, 3, \ldots, T$ **do**
  $B_{h_t} = \text{LBP}(L_{t-1})$
  $x^t \sim \pi(\cdot|B_{h_t})$
  $L_t \leftarrow L_{t-1} \cup (x^t, y^t)$
  $U_t \leftarrow U_{t-1} \backslash x^t$
end
$\hat{h} \leftarrow Binary(B_{h_T})$

Algorithm 1 runs as follows: at each time step t, we first get our current belief B_{h_t} based on the previously labelled set L_{t-1} (i.e., the set of all the questions we have asked the student before time step t and their responses). We then select a question x^t from the previously unlabelled set U_{t-1} to ask the student by sampling from $\pi(\cdot|B_{h_t})$, which defines a probability distribution with support over all questions in U_{t-1} given B_{h_t}. Then, we update U_{t-1} to U_t by removing x^t from U_{t-1} and update L_{t-1} to L_t by adding x^t and its label y^t to L_{t-1}. The quizzing process terminates when the query budget is exhausted. In this work, we fix $T = 10$ as required

by the NeurIPS 2020 Education Challenge [27]. The final output of the algorithm $\hat{h}$ is the student's knowledge state at time step T, which is inferred based on B_{h_T}.

2.4 Evaluation

We evaluate our QPs using two methods. First, we create a synthetic dataset consisting of simulated students (see Section 4.1). We predict each student's knowledge state using Algorithm 1. Given a prediction result $\hat{h}$, we calculate the prediction accuracy using the following equation:

$$Acc(\hat{h}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\hat{h}[i] = h^{\star}[i]) \tag{3}$$

where $h^{\star}$ is the actual knowledge state.

Second, we apply our QPs to the NeurIPS 2020 Education Challenge (see Section 4.2). The challenge is to obtain a limited set of answers from each student for predicting the correctness of their answers to the remaining questions. Our approach to this challenge is to first infer a student's knowledge state using Algorithm 1 and then predict the student's responses to the remaining questions based on the inferred knowledge state. Specifically, we design an additional model (see Section 4.2.2) that takes in our belief about the student's knowledge state at the final time step B_{h_T} and outputs the student's response to each of the m questions. Formally, the vector $\hat{\mathcal{Y}} \in \mathbb{R}^m$ denotes the output of the model. We calculate the prediction accuracy as:

$$Acc(\hat{\mathcal{Y}}) = \frac{1}{|U_T|}\sum_{x_i \in U_T} \mathbb{1}(\hat{\mathcal{Y}}[i] = \mathcal{Y}^{\star}[i]) \tag{4}$$

where U_T is the set of unlabelled questions at the final time step (unseen by the model), $\mathcal{Y}^{\star}[i]$ is the student's actual response to x_i, and $\hat{\mathcal{Y}}[i]$ is the predicted response.

3. DESIGNING QUIZZING POLICIES

In this section, we present heuristics-based approaches and a reinforcement learning (RL)-based approach to designing a quizzing policy (QP) that takes in a belief B_h about a student's knowledge state and outputs the next question to ask the student.

3.1 Heuristic approaches

We present two simple heuristics for designing a QP: random selection (QP-RANDOM) and uncertainty sampling (QP-UNCERTAIN). QP-RANDOM is straightforward: we always randomly select a question from the unlabelled set U (i.e., $\pi(a|B_h) = \frac{1}{|U|}$ for each $a \in U$). QP-UNCERTAIN suggests picking a question corresponding to a concept that our current model is most uncertain about (i.e., the concept with a probability of being known that is closest to 0.5). Formally, we define:

$$b^{\star} = \arg\min_{b_i \in B_h} |b_i - 0.5|$$

We first pick a concept $c^{\star}$ with a probability of being known that is equal to $b^{\star}$. We break ties randomly. We define $U_{c^{\star}}$ as the set of questions that have not been asked and are associated with $c^{\star}$. We then define the policy:

$$\pi(a|B_h) = \begin{cases} \frac{1}{|U_{c^{\star}}|} & \text{if } a \in U_{c^{\star}} \\ 0 & \text{otherwise} \end{cases}$$

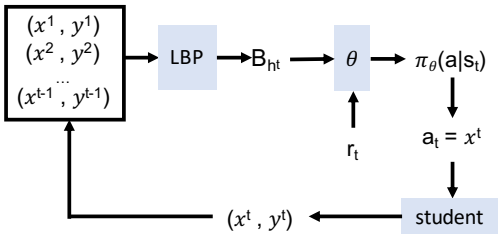

Figure 3: QP-RL approach.

3.2 RL-based approach

We now propose an RL-based approach (QP-RL) for learning a QP. An RL agent learns how to make good decisions over time by interacting with an environment that is typically modeled as a Markov Decision Process (MDP). In our problem setting, we define the MDP $M = (S, A, P, R, s_0)$ as follows:

- The state space S is the set of beliefs B_h about student knowledge (i.e., $S = \{[b_1, ..., b_n] | 0 \leq b_i \leq 1\}$);
- The action space A is the set of questions that have not been asked;
- The transition dynamics $P : S \times A \times S \to \mathbb{R}$ define the probability of transitioning from one state to another by taking a particular action. In our case, we transition to state s_{t+1} from s_t based on the student's response y^t.
- The reward function $R : S \times A \times S \to \mathbb{R}$ is defined as the difference in prediction accuracy between the current time step and previous step: for predicting student knowledge, given the inferred knowledge state h_{t+1} after taking action a_t, we calculate the reward for time step t as $Acc(h_{t+1}) - Acc(h_t)$;
- The initial state s_0 corresponds to the initial belief about student knowledge: each concept has a 0.5 probability of being known.

Figure 3 shows an overview of the QP-RL approach. For training the RL agent, we consider an episodic, finite-horizon setting. During each episode, we train on one student's data, and the length of the episode is the query budget T. At each time step t, we run the LBP algorithm that takes in the student's response history $L_{t-1} = \{(x^i, y^i)\}_{i=1}^{t-1}$ to update our belief about the student's knowledge state B_{h_t}. Then, the RL model, which is a neural network with parameters θ, takes B_{h_t} as input (i.e., $s_t = B_{h_t}$) and outputs a vector $\mathbf{p_c} \in \mathbb{R}^n$ which represents the probability of selecting a question corresponding to each of the n concepts. We first select a concept c_i by sampling based on $\mathbf{p_c}$ and then randomly select one question from U_{c_i} (a set of questions that have not been asked and are associated with c_i). We then define the final policy parametrized by θ: $\pi_\theta(a|B_{h_t}) = \frac{\mathbf{p_c}[i]}{|U_{c_i}|}$ for $c_i \in C$ and $a \in A$. Our policy $\pi_\theta(a|B_{h_t})$ allows us to select the next question to query and add the next question-response pair (x^t, y^t) to the response history. We then update B_{h_t} based on the updated response history using the LBP algorithm. We calculate the reward for the current time step $r_t = Acc(Binary(B_{h_{t+1}})) - Acc(Binary(B_{h_t}))$.

We use REINFORCE policy gradient method [25, 29] to learn our policy π_θ parametrized by θ. In each episode corresponding to a single student, the RL agent performs an update as follows. First, an initial state s_0 (the initial belief that each concept has a 0.5 probability of being known by the student) is generated. Then, the policy π_θ is executed until the episode ends, generating a sequence of experience given by $(s_t, a_t, r_t)_{t=1,2,...,T}$. Then, in this episode, for each $t \in \{1, 2, ..., T\}$, we use the following gradient update with η as learning rate:

$$\theta \leftarrow \theta + \eta \cdot \underbrace{\left(\sum_{\tau=t}^{T} r_\tau\right) \cdot \left(\nabla_\theta \log\left(\pi_\theta(a_t \mid s_t)\right)\right)}_{\text{gradient at time step } t \text{ in an episode}} \quad (5)$$

In experiments, we use the architecture used in [7]. Specifically, the policy network is a 3-layer fully connected neural network with the following architecture: the input layer has $n = 57$ units for B_h; the first and second hidden layers have 128 hidden units; and the output is a vector $\mathbf{p_c} \in \mathbb{R}^n$ where $n = 57$ to produce a probability of selecting each of the 57 concepts. The first two hidden layers use ReLU activations, and the final layer uses the softmax function to ensure probabilities sum to 1. We use ADAM [14] optimizer for training.

4. EXPERIMENTAL EVALUATION

We first evaluate and compare our quizzing policies (QPs) using a synthetic dataset. We then apply our QPs to the Eedi dataset from the NeurIPS 2020 Education Challenge.

4.1 Simulations

We simulate virtual students taking the assessment quiz and test how well we can predict students' knowledge states in a controlled setting using different QPs.

4.1.1 The synthetic dataset

We generate a dataset consisting of 24,000 simulated student knowledge states. To do so, we first construct a graph for representing the student knowledge state that we aims to infer (see Section 2.2) and then get a probability distribution over the binary variables in the knowledge state that satisfies a set of assumptions about the student's knowledge. We then sample ground-truth student knowledge state values from the probability distribution. In this simulation, we use the same 57 concepts in the Eedi dataset (described in Section 4.2.1) for constructing the graph. We assume some of these concepts have different levels of difficulty, and similar concepts are more likely to be assigned the same knowledge state values.[2] Based on these assumptions, we assign a value of difficulty to each of the 57 concepts. We define $\textit{difficulty}_{c_i}$ = 1− the average correctness of the concept c_i across all students' answers in the Eedi dataset.[3] We run the LBP algorithm on the constructed graph to get a probability distribution from which we sample student knowledge states. Specifically, the output of the LBP algorithm gives the probability of knowing each concept, and we sample values of 0 or 1 for each concept to generate the ground-truth student knowledge states in our synthetic dataset.

[2]Although our assumptions might not hold in a real-world setting, the goal of this experiment is to compare different QPs and investigate the potential of QP-RL for learning a strategy tailered to a pre-defined knowledge structure. For instance, compared to the heuristic approach QP-Uncertain, QP-RL should learn to select the questions that are not only uncertain but can also give more information about other questions that are not selected (e.g., selecting questions corresponding to concepts that are connected with a lot of the other concepts).

Table 1: Test performance of different QPs on the synthetic dataset. QP-Uncertain achieves a better performance than QP-Random, and QP-RL improves over QP-Uncertain significantly.

QP	Accuracy
QP-RL	0.721 ± 0.004
QP-Uncertain	0.700 ± 0.002
QP-Random	0.675 ± 0.003

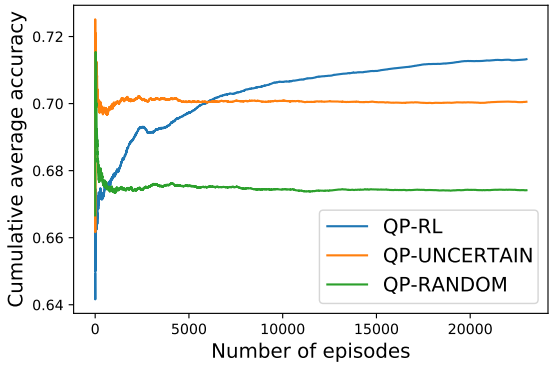

Figure 4: Training performance of QP-RL on the synthetic dataset compared to heuristics. QP-RL improves over QP-Random and QP-Uncertain after about 6,000 episodes of training. The cumulative average accuracy at each episode is calculated as the average accuracy across all previous episodes. It is important to note that QP-Random and QP-Uncertain are fixed policies that are not being trained. The cumulative average accuracy for the first few episodes might seem noisy due to small sample size.

4.1.2 Results

We split the dataset into 23,000 students as the training set and 1,000 students as the test set. We train QP-RL until the cumulative average accuracy converges. Figure 4 shows the training performance of QP-RL compared to fixed heuristics. After training, we run each QP 10 times on the test set to calculate the average accuracy and standard deviation across these 10 trials, see Table 1. Although QP-RL leads to a 2% gain in accuracy compared to QP-Uncertain, it requires a moderate amount of training data ($> 6,000$ students in this case). QP-Uncertain is a less optimal strategy but can achieve a reasonably good performance without any training data. These results provide initial evidence that QP-RL can learn an effective QP, and the performance can be improved further with more data.

[3]For simulations, one could also try other difficulty values, but it does not matter which specific difficulty value we assign to each concept because the goal is to model a setting where we have concepts of varying levels of difficulty.

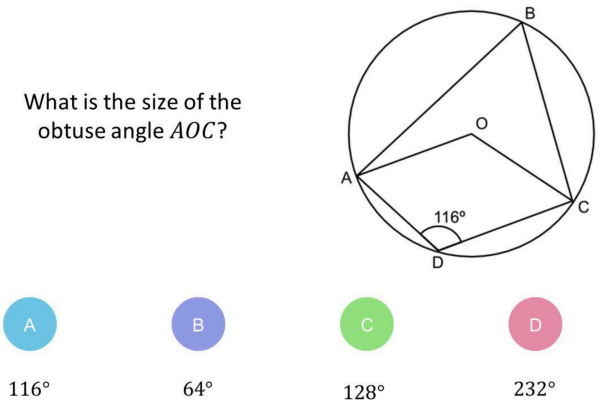

Figure 5: An example of a question in the Eedi dataset [27]. For each multiple-choice question, exactly one choice is correct.

4.2 NeurIPS 2020 Education Challenge

We then apply our QPs to one of the tasks in the NeurIPS 2020 Education Challenge (see Section 4.2), which is to obtain a limited set of answers from each student for predicting the correctness of their answers to the remaining questions. Our approach to this challenge is to first infer a student's knowledge state using Algorithm 1 and then predict the student's responses to the remaining questions based on the inferred knowledge state.

4.2.1 The Eedi dataset

The Eedi dataset contains student responses to multiple-choice questions (see Figure 5) on various math topics, which was collected between September 2018 and May 2020. It contains 948 questions and a total number of $1,508,917$ responses to these questions from $6,148$ students. The dataset is split into the training set (4918 students), the validation set (615 students), and the test set (615 students).

Each question in the dataset is associated with a list of subjects. Each subject covers an area of mathematics. These subjects are arranged in a tree structure by experts based on the generality of the subjects. For instance, "Fractions" is the parent subject of "Multiplying Fractions" and "Simplifying Fractions". For simplicity, we only consider the most granular subject (i.e., the leaves in the tree) as the concept that each question corresponds to. The 948 questions correspond to 57 unique concepts. We consider concepts that share the same super-concept (i.e., parent) to be similar (see Figure 1).

4.2.2 Student performance prediction

To predict a student's responses to unseen questions based on the inferred knowledge state, we propose a neural network-based model that takes in the belief about the student's knowledge B_{h_T} at time $T = 10$ (our belief about their knowledge after we have asked 10 questions) and outputs the probability of answering each of the 948 questions in the dataset correctly. The student performance prediction model is a 3-layer fully connected neural network with the following architecture: the input layer has $n = 57$ units for B_{h_T}; the first hidden layer has 256 hidden units; the second hidden layer has 512 units; and the output is a vector $\hat{\mathcal{Y}} \in \mathbb{R}^m$ where $m = 948$ to represent the probability of correctness for each of the 948 questions. The first two hidden layers use ReLU activations, and the final layer uses the sigmoid function to ensure the output values are between 0 and 1. We use ADAM [14] optimizer for training. We convert the output probabilities into binary values of 0 or 1 (0 if the probability is less than 0.5, 1 otherwise) and calculate the prediction accuracy using Equation 4. We train the model using randomly selected queries until the validation accuracy converges. The model parameters are updated based on binary cross-entropy loss.

Table 2: Test performance different QPs on the Eedi dataset. QP-RL improves slightly over QP-UNCERTAIN.

QP	Accuracy
QP-RL	0.690 ± 0.005
QP-UNCERTAIN	0.680 ± 0.003
QP-RANDOM	0.684 ± 0.003

4.2.3 Results

Given a trained performance prediction model from Section 4.2.2, we then train QP-RL using the difference in final prediction accuracy between time steps as reward signals: $r_t = Acc(Binary(\hat{\mathcal{Y}}_t)) - Acc(Binary(\hat{\mathcal{Y}}_{t-1}))$. After training, we run each QP 10 times on the test set to calculate the average accuracy and standard deviation across these 10 trials. Table 2 shows that QP-RL improves slightly over QP-UNCERTAIN, but the difference between QP-RL and QP-RANDOM is not significant. Results in Section 4.1.2 show that in a more controlled setting, QP-RL already requires a moderate amount of training data ($> 6,000$ students) to improve over heuristics. However, we only have training data from about $5,000$ students in this experiment. Learning a QP from real students' data that are noisy is more challenging, and it may be the case that improving QP-RL further would require a much larger dataset. Even though QP-RL seems to require a substantial amount of training data, this is a one-time training, and the learned policy can be applied to future students.

5. CONCLUSION

Student assessment is a crucial component of many online education systems for improving student learning outcomes. Inferring student knowledge state by quizzing poses a technical challenge: maximizing accuracy while minimizing the quizzing cost. In this paper, we show initial evidence that reinforcement learning (RL) provides a potential solution, improving over heuristics given sufficient training data.

There are several research directions for future work. Further gains in accuracy could be achieved by exploring more powerful RL techniques and more complex student knowledge modeling techniques. In this work, we model all concepts that share the same super-concept as having the same relationship; however, there could be prerequisites as well as weaker and stronger relationships in reality. It would be important to study whether varying the influence values between concepts would lead to gains in model performance.

6. REFERENCES

[1] U. Z. Ahmed, M. Christakis, A. Efremov, N. Fernandez, A. Ghosh, A. Roychoudhury, and A. Singla. Synthesizing tasks for block-based programming. In *NeurIPS*, 2020.

[2] J. Bassen, B. Balaji, M. Schaarschmidt, C. Thille, J. Painter, D. Zimmaro, A. Games, E. Fast, and J. C. Mitchell. Reinforcement learning for the adaptive scheduling of educational activities. In *CHI*, pages 1–12, 2020.

[3] M. Chi, K. VanLehn, D. Litman, and P. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1):137–180, 2011.

[4] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[5] S. Doroudi, V. Aleven, and E. Brunskill. Where's the reward? *International Journal of Artificial Intelligence in Education*, 29(4):568–620, 2019.

[6] Eedi. Eedi: Online maths lessons with live teacher support. `https://eedi.com`.

[7] A. Efremov, A. Ghosh, and A. Singla. Zero-shot learning of hint policy via reinforcement learning and program synthesis. In *EDM*, 2020.

[8] J.-C. Falmagne, M. Koppen, M. Villano, J.-P. Doignon, and L. Johannesen. Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review*, 97(2):201, 1990.

[9] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. *CoRR*, abs/1708.02383, 2017.

[10] R. Gilad-Bachrach, A. Navot, and N. Tishby. Query by committee made real. In *NIPS*, volume 5, pages 443–450, 2005.

[11] M. R. Gormley and J. Eisner. Structured belief propagation for nlp. In *ACL*, pages 5–6, 2015.

[12] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, pages 417–424, 2006.

[13] A. Iglesias, P. Martínez, R. Aler, and F. Fernández. Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. *Knowledge-Based Systems*, 22(4):266–270, 2009.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[15] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR*, pages 3–12, 1994.

[16] Y. Liu. Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of chemical information and computer sciences*, 44(6):1936–1941, 2004.

[17] T. Mandel, Y.-E. Liu, S. Levine, E. Brunskill, and Z. Popovic. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, pages 1077–1084, 2014.

[18] R. Moskovitch, Y. Elovici, and L. Rokach. Detection of unknown computer worms based on behavioral classification of the host. *Computational Statistics & Data Analysis*, 52(9):4544–4566, 2008.

[19] K. Pang, M. Dong, Y. Wu, and T. Hospedales. Meta-learning transferable active learning policies by deep reinforcement learning. *CoRR*, abs/1806.04798, 2018.

[20] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[21] J. Rollinson and E. Brunskill. From predictive models to instructional policies. In *EDM*, 2015.

[22] N. Roy and A. McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML*, pages 441–448, 2001.

[23] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT*, pages 287–294, 1992.

[24] A. Singla, S. Tschiatschek, and A. Krause. Actively learning hemimetrics with applications to eliciting user preferences. In *ICML*, pages 412–420, 2016.

[25] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[26] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[27] Z. Wang, A. Lamb, E. Saveliev, P. Cameron, Y. Zaykov, J. M. Hernández-Lobato, R. E. Turner, R. G. Baraniuk, C. Barton, S. P. Jones, et al. Diagnostic questions: The neurips 2020 education challenge. *CoRR*, abs/2007.12061, 2020.

[28] J. Whitehill and J. Movellan. Approximately optimal teaching of approximately optimal learners. *IEEE Transactions on Learning Technologies*, 11(2):152–164, 2017.

[29] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[30] M. Woodward and C. Finn. Active one-shot learning. *CoRR*, abs/1702.06559, 2017.

[31] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127, 2015.

[32] J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end autonomous driving. *CoRR*, abs/1605.06450, 2016.