Optimizing Representations for Question Sequencing using Reinforcement Learning

PAPER ID #34

This paper studies the use of Reinforcement Learning (RL) policies for optimizing the sequencing of learning materials to maximize learning as measured by expected future student performance. We conduct the training completely offline based on the publicly available dataset EdNet, a large-scale hierarchical dataset of diverse student activities collected by an existing online learning platform used by tens of thousands of students. We confront the challenges of offline RL through the construction of a student model in the form of a Markov Decision Process derived from observations in the dataset. A feature pool is created from the raw student logs, from which, different state representations are formed by sampling greedily using an iterative augmentation procedure to optimize the state-space. The paper explores the influence of the state representations on the performance of the RL agent and its robustness towards perturbations on the environment dynamics. We show that a larger, more complex representation constituting of more features, yields better policy performance. We also show that the policies derived from the larger representations are more robust towards perturbations in the expected environment induced by stronger and weaker learners. This work is a first step towards optimizing representations when designing policies for sequencing educational content that can be used in the real world.

ACM Reference Format:
Paper ID #34. 2018. Optimizing Representations for Question Sequencing using Reinforcement Learning. In . ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

E-learning platforms have seen a surge in popularity over the last decade [14], spurred on by the increased internet penetration into developing communities [4]. The target demographic has expanded beyond casual users/students as more organizations adopt e-learning to train their workforce and actively engage them in life-long learning [44]. E-learning platforms come with intrinsic advantages over traditional learning [29] and the concept of 'intelligent' learning platforms has recently emerged as a new trend within this domain [19].

These platforms are driven by expansion of research into applied artificial intelligence (AI) and have been addressed in literature under various names such as 'Intelligent Tutoring Systems' (ITS) [8] or 'Pedagogical Agents' [42]. The common objective is to create an adaptive learning environment, with an emphasis on 'generativity' or the ability to generate customized problems, hints, or support for each individual student [44]. Traditional self-paced online courses, such as those used in MOOCs (Massive Open Online Courses), create a static learning path often without contextualized feedback or personal adaptive guidance. In such scenarios, a user might feel disinclined to progress through the course material if they encounter difficulties in following the generic course structure. This will often lead to them losing interest and disengage from the course by dropping out, as is so often observed in distance education [44]. Therefore, there is a growing need for a personalized sequencing of content/support that can adapt to the individual differences of the student as well as their evolving pedagogical requirement throughout the course progression [36]. Intelligent tutoring systems, pedagogical agents and the like, are the proposed solution within the applied AI in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
Manuscript submitted to ACM

education community. With large educational datasets becoming more prevalent[7][31], these AI-driven solutions become more feasible.

Research in cognitive science has long demonstrated the strong correlation between topic sequencing and learning [34]. Static e-learning platforms lack the capacity to respond to a student's 'cognitive state' and therefore perform poorly relative to a human tutor [44]. A students' cognitive state is a term used to describe an explicit representation of their learning characteristics [20]. It can be composed of multiple features such as a student's general competency or aptitude, learning style and confidence level [44]. One of the main difficulties in constructing an intelligent learning environment, is approximating this cognitive state since it is not directly observable [12]. However, a more achievable goal is to model this state to a degree where an agent can leverage the information to provide effective sequencing [44]. This model can be constructed in a variety of ways and is commonly referred to as the 'student model' in literature [20][44]. The fidelity of the model could significantly impact the agent's performance since a more complex representation of the cognitive state will provide more information to the pedagogical agent [44].

While constructing a model is usually the first challenge, another that soon follows is the need for an optimization algorithm to utilise the model in enhancing learning. Reinforcement learning (RL) provides the perfect mathematical machinery to optimize a learning sequence towards this predefined objective [12]. RL is a class of machine learning algorithms, that optimize a control problem where an 'agent', learns an optimal 'policy' that maps 'states' to a given 'action' in such a way as to maximized a 'reward' function [40]. One can quickly observe the relevance of such algorithms in the context of ITS. Here, a learning sequence is optimized based on a numerical reward (for example test marks) by a pedagogical agent that prescribes actions (adaptive feedback or sequencing of content) based on different states (approximating users' cognitive states).

Our objective is to develop an adaptive RL based pedagogical agent that is able to optimize the sequence of learning materials (questions and lectures) to maximize student performance as measured by the expected ability to answer questions correctly at varying levels of difficulties. This agent will have the capacity to respond to a student's current state as they progress through the learning material. The agent will be trained completely offline using the EdNet dataset, an extensive collection of student logs from an online learning platform, *Santa* [7]. The paper addresses the following challenges. First, how to create a student model in the form of a data-driven Markov Decision Process (MDP) and utilise a Dynamic Programming based algorithm, Policy Iteration, to derive an optimal policy. Second, how to optimize the representation of the model, using features derived from the students' logs, to be able to feasibly compute a policy that can be evaluated against students with varying learning characteristics.

Our methodology uses a greedy iterative augmentation procedure to augment the representation space, by incrementally adding new features and choosing the best performing representations greedily. We show that a larger, more complex representation constituting of more features, yields better policy performance and is more robust towards perturbations in the model induced by students with different learning characteristics. Our inferred policy obtained high expected cumulative reward against perturbations corresponding to different student types in the model-based RL analysis. These results demonstrates the potential of using large scale educational data sets to inform the design of sequencing algorithms, as well as the need to handle the uncertainty induced by unseen or out of distribution actions and robustness analysis against different types of students.

2 RELATED WORK

This paper relates to prior work in student modeling as well as automatically sequencing educational content to students [12]. A common approach in prior work is to integrate learning/cognitive theory into the construction of the student model. This imparts domain knowledge into the workflow and is shown to yield positive results [12]. An example of such approach is the work by Bassen et al. [2]. Their objective was to optimize the sequencing of learning material from different knowledge components (KCs), to 'maximize learning'. Their reward function is based on the difference between a post-test score (taken by users after completing the course) and a pre-test score (taken before the course). This metric is denoted as the Normalized Learning Gain (NLG). Training the agent with human participants is far too resource intensive. Instead their training is performed on a 'simulated learner' based on Bayesian Knowledge Tracing (BKT), a cognitive model that aims to estimate a learner's mastery of different skills. The learner's response to a particular question can be simulated based on the mastery of the related skill. The parameters of the BKT were set based on domain knowledge. Segal et al. [36] also utilised a similar cognitive based model, Item Response Theory (IRT) [18] to simulate student responses to questions of different level of difficulty. This is especially relevant since their objective was to sequentially deliver questions of differing levels of difficulty (rather than KCs) to maximize learning gains.

Other approaches forgo the framework of established learning models and instead manually design their simulators further integrating domain expertise. Dorça et al. [11] employed a probabilistic model to simulate the learning process. Instead of sequencing activities by KCs or difficulties, they sequenced activities based on its associated learning style i.e. visual, verbal etc. Therefore, their simulator was manually designed based on research surrounding these principles. Iglesias et al. [20] utilised an expert derived artificial Markov Decision Process to act as the student model. This entails manually describing the state space, transition probabilities and rewards. Similar to previous student models, this MDP can be used to simulate student responses to train the RL agent.

The works described so far do not utilise historical data to derive their student model. While integrating expert knowledge can be beneficial, a completely data-free proposition could impart strong biases. In our implementation, we take an alternative approach in using a purely data-driven model. There are existing literature which also do the same. For example, the authors of [38][41][5][35] employed data-driven MDPs as their student model. Different than the handcrafted MDP in [20], the transition probabilities and reward functions in these MDPs were obtained from the aggregated statistics observed in the dataset. Data-driven student models in literature were not only limited to data-driven MDPs. For instance, Beck et al. [3] utilised a linear regression model denoted as Population Student Model (PSM). PSM was trained on student trace data from a learning software and could simulate time taken and probability of a correct response.

Data-driven simulators require a quality training corpus that is sufficiently large and varied [38][21]. In contrast to EdNet (a massive dataset collected over several years), the authors of previous papers were limited to much smaller scale datasets that were collected from a single cohort. Our implementation is the first in RL based pedagogical agents to utilise a large scale educational dataset.

3 BACKGROUND: REINFORCEMENT LEARNING & MARKOV DECISION PROCESSES

The RL framework is governed by a Markov Decision Process [40][24][6] that is defined by the tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S} = s_1, .., s_n$ defines the complete space of every possible state, $\mathcal{A} = a_1, .., a_n$ represents all available actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ designating the transition probabilities between states conditioned on an action and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ denotes the reward function that is conditioned on the state, action and observed next state.

The goal of the agent is to maximize the **cumulative** reward it accumulates from each state. This cumulative reward is usually discounted by a factor γ raised to the power of t, to represent a lower perceived value for rewards received further in the future. The policy is a mapping of optimal actions to each state in $\mathcal{S}$. The discounted cumulative reward is denoted as the 'return', and the return from a particular state is associated with a policy π and the transition

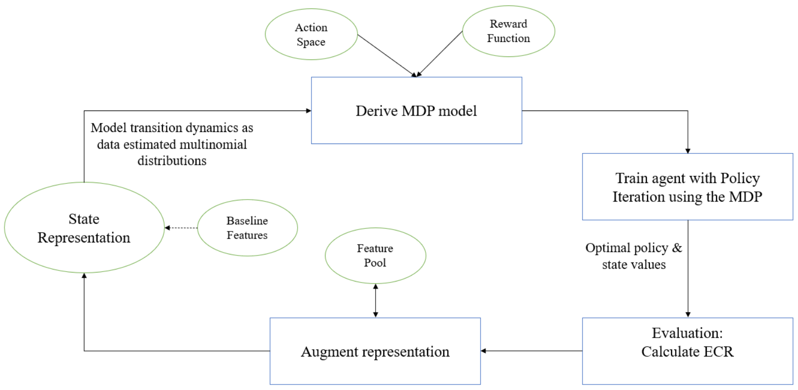
Action Space
Reward Function
Derive MDP model
Model transition dynamics as data estimated multinomial distributions
Train agent with Policy Iteration using the MDP
State Representation
Baseline Features
Feature Pool
Optimal policy & state values
Augment representation
Evaluation: Calculate ECR

Fig. 1. Greedy iterative feature augmentation pipeline

dynamics $\mathcal{P}$. The reward function ties the agents optimization goal with the modeller's actual objective. Therefore its design must ensure those two criteria are aligned. Note that the policy can be stochastic where it prescribes a distribution of actions at each state. The return of a state (or the state value) under a policy π can be defined as $V(s) = \mathbb{E}_{a\sim\pi,r,s\sim\mathcal{P}}[\sum_{t=0}^{\inf}\gamma^t r(s_t, a_t) \mid s_0 = s]$.

In 'tabular' RL, a state value is updated only from the experiences that involve the state [40]. If a state is not encountered in the agent's past experiences, then nothing can be said about its value. 'Function Approximation' can help alleviate this problem, by generalizing a state's value based on features i.e. states with similar features will have similar values. As in supervised learning, parametrized function approximators learn weight vectors that approximate a state's value from its features. Even with function approximation, online model-free RL algorithms are generally appropriate for domains where environment interaction is computationally cheap and feasible [21].

The alternative to model-free RL is model-based RL. These approaches are more prevalent in domains where environment interactions are cost prohibitive [21][25]. Model-based RL requires a model that holds information regarding the environment dynamics i.e. the transition probabilities $\mathcal{P}$ and reward function $\mathcal{R}$. The model mimics the behaviour of an environment and allows inferences about how the environment will respond to actions [40]. In approaches where the model is driven by sample data from the environment, model-based RL tends to be more sample efficient than model-free RL [22][32]. However, an important caveat to model-based RL is that the real-world performance will be heavily influenced by the quality of the model [21].

4 METHODOLOGY

Our methodology is outlined in Figure 1. To find an optimal representation for the state-space of the agent we begin with a base representation consisting of simple features to form a base MDP model. Policy Iterations uses the model to obtain an optimal policy, which can subsequently be evaluated. A greedy iterative augmentation procedure (visualized in figure 1) is then employed to augment the representation space, by incrementally adding new features and choosing the best performing representations greedily. The process is also handling the uncertainty induced by unseen or out of distribution (OOD) actions in the model-based RL analysis. As a final step, we analyze the robustness of the policies using Expected Cumulative Reward against perturbations corresponding to students with varying learning characteristics.

4.1 Data Description

EdNet is massive dataset of student logs from a MOOC learning platform in South Korea, *Santa*, collected by *Riiid! AI Research*[1]. Santa covers a preparation course for the TOEIC (Test of English for International Communication) English proficiency exam. There are a total of 131,441,538 interactions collected from 784,309 students on the e-learning platform. Each student's interaction log is stored on an individual csv (Comma-Separated Values) file. These consist of user records of questions attempted, lectures watched and explanations reviewed, along with other meta information. EdNet logs are presented at 4 levels of hierarchy with higher levels providing higher fidelity logs, such as logs of lectures watched and the explanations reviewed. These are recorded in real time to provide an accurate chronological record of the students' interaction with the platform. At the highest hierarchy, EdNet records detailed actions such as playing/pausing lectures and payment related information.

EdNet does not provide any post-test/pre-test record for their users. Therefore, the reward in the student model can only be a function of what is present in the logs. With this limitation in mind, we chose our reward function to be a sort of proxy to student performance and base it on the correctness of a student's response towards an activity. We also integrate the activity's difficulty into the reward design such that correct responses on harder levels indicate stronger performance and attain higher rewards. Inversely, incorrect responses on easier levels attain a larger punishment (negative reward). This is to avoid the agent abusing the reward function by always assigning easy questions. Lectures do not have a correct/incorrect responses and so a default reward of '0' is assigned.

4.2 State Space Representation

The main incentive in exploring the data is to observe potential candidates for the action space, state features and reward function. EdNet provides a total 13,169 questions and 1,021 lectures tagged with 293 types of skills [7]. Each question and lecture is segregated into one unique 'part', with 7 parts in total. Each part was grouped based on some meaningful domain criteria like the KCs in Bassen et al. [2]. Note that EdNet offers a finer grouping of question/lectures according to the 'skills' (293 in total) they entail. However the question bank is more unevenly distributed with respect to the skills grouping, which can lead to difficulties in getting equal support (or supporting observations) for each class from the dataset. Therefore, 'part' was chosen on the basis that its 7 unique classes is computationally feasible.

A balance must be made between a manageable action space size and a useful pedagogical agent. Considering the large support available in EdNet, we decided to extend the action space, discretizing the questions in terms of difficulty, similar to that in Segal et al. [36]. Unfortunately EdNet does not classify questions/lectures into difficulty levels, meaning that they would need to be inferred from the student logs. One way of achieving this is to measure the percentage of correct answers submitted for a question and compare it with other questions in the question bank. A distribution of question difficulties can be created with each item being a unique question in the question bank. Quantiles can then be derived from this distribution to evenly split the questions into discrete levels. The difficulty levels were quantized into 4 levels ranging from 1-4.

Initially, a symmetrical reward function was designed with rewards for questions ranging from 1 to 4 if answered correctly or -1 to -4 if answered incorrectly, depending on the level of difficulty. However, this led to a right-skewed cumulative reward distribution across the population of users, suggesting that the reward design was somewhat too generous. As we wanted this distribution to mimic a normal distribution usually exhibited in student grades [17], we doubled the penalty of incorrect answers transforming the range towards -2 to -8.

[1]https://www.riiid.co/

Feature Pool	Bins	Description
"av time"	4	The cumulative average of the elapsed time measured at each activity.
"correct so far"	4	The ratio of correct responses to the number of activities attempted.
"prev correct"	3	A flag to indicate whether the user answered correctly in the previous question. Fixed value for lectures.
"expl received"	4	Cumulative count of explanations reviewed by the user.
"steps-since-last"	8	A count of the number of steps since the current part was last encountered.
"lects consumed"	4	A cumulative count of the lectures consumed by a user.
"slow answer"	2	A flag to indicate whether the user's elapsed time for the preceding question was above the average elapsed time for that question.
"steps in part"	4	The cumulative count of how many steps a user has spent in the current part.
"avg fam"	4	The average part familiarity across all the 7 parts.
"topic fam"	4	captures part familiarity of the previously chosen action and disregards the familiarity of other parts.

Table 1. Initial Feature pool for state space

The dataset in EdNet is longitudinal, where observations are made on the same subjects (users) across a period of time [10]. Most of the state features derived in this section will be longitudinal/temporal in nature too since not much can be derived from a single isolated observation.

Several studies have shown the significant impact of the representation choice on the final agents performance, with some arguing it is just as influential as optimization algorithm itself [41][37]. This aligns with the theoretical understanding of why human tutors usually outperform their computer counterparts, in that they are able to adapt to certain cues exhibited by the student during learning [44]. The representations of the model consist of features that infer such 'cues' from the dataset. To investigate the impact of the representation on performance, we first create a feature pool, from which different representations are formed using the greedy iterative augmentation algorithm.

We begin with base features that are widely seen in similar RL driven Intelligent Tutoring implementations [2][20][41][5]. We also chose these features as they were readily derivable from the logs. This sets a reasonable minimum requirement for the data gathering process, should this implementation be repeated with other datasets. The features, their descrpitions, and the associated granularity of their representations, are shown in Table 1.

Relative to prior work that limit the feature size to be binary [2][28][41] our feature space is considerably larger. With each user covering on average 440 activities in their learning period [7], a binary split would lose a lot of information on the evolution of a feature value throughout the course. Our features in contrast, have up to 8 bins. This ultimately imposes a necessity for a quality training corpus that can provide sufficient support for each of the many unique combinations within the feature space. This is where the scale of EdNet provides a distinct advantage relative to previous implementations.

5 STUDENT MODEL CONSTRUCTION

The subsequent step in the pipeline is to derive an MDP-based student model. The methodology chosen is based on the works of [41][38] where the transition probabilities are modelled as multinomial distributions derived from state transition counts observed in the dataset as shown in equation 1. Intuitively, this means that a particular outcome s_i, of enacting action a_k in state s_j has a probability given by to the number of times that outcome was observed in the dataset, normalized by the sum of all possible outcomes observed under the same conditions.

$$\hat{p}(s_i \mid s_j, a_k) = \frac{c(s_i, s_j, a_k)}{\sum_{i=1}^{n} c(s_i, s_j, a_k)} \tag{1}$$

Where $c(s_i, s_j, a_k)$ is the count of observed transitions where enacting action a_k in state s_j and leads to the next state s_i. This provides the transition probabilities component of the MDP student model for each (s, a, s') or the three argument dynamic [40]. In many standard MDP definitions [41][38][5], the reward function is also defined in terms of the three argument dynamic i.e. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. This assumes a deterministic environment reward with respect to a given (s, a, s'). Recall that our rewards are a function of the **student's correctness** and the **level of difficulty** and can take values $r \in \{-8, -6, -4, -2, 0, 1, 2, 3, 4\}$. While the level of difficulty is captured by the action in the (s, a, s') tuple, the correctness is only captured in the states when 'prev_correct' (refer table 1) is included in the representation. Without it, the reward function is stochastic with respect to a given (s, a, s') . A solution for this is to consider the 'four argument' (s, a, r, s') environment dynamics as described by [40] in Equation 2:

$$p(s', r \mid s, a) = Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}, \tag{2}$$

The estimated dynamics, $\hat{p}(s_i, r \mid s_j, a_k)$, can be derived from the data, in a similar fashion with the three argument dynamics i.e. equation 1, with small changes to the count arguments. Using the four argument dynamics, an expected reward can then be calculated for the three argument reward function, $r(s, a, s') = \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s]$ as shown in equation 3.

$$r(s_i \mid s_j, a_k) = \sum_{r \in \mathcal{R}} r \frac{\hat{p}(s_i, r \mid s_j, a_k)}{\hat{p}(s_i \mid s_j, a_k)} \tag{3}$$

Where $\mathcal{R}$ is the discrete set of all the possible rewards (9 in total). With this, our MDP model consisting of the three argument transition probabilities and reward function is defined. Note that we prioritise a three argument dynamic instead of four since it is more standard in relevant literature and is the required format for the Policy Iteration library [9] used in the training the agent.

The dynamics are of course dependent on the state representations. As we continuously augment the representations, the support for each unique state will inevitably fall due to further division of the observations. Another factor in providing a balanced distribution of support within the transitions, is the variety of actions chosen in each state. This ultimately depends on the action space (described in section 4.2) and the behaviour policy used to obtain the dataset. A higher fidelity action space will lead to a blowup in the size transition space i.e. the unique combinations of (s, a, s'). The behaviour policy is the strategy used in taking the actions observed in the dataset. Although it is unknown in most cases, it is important that the behaviour policy is sufficiently varied in terms of its action choices to ensure a balanced distribution of support. Because of this, a random behaviour policy fits the objective well [38]. Since users in EdNet are allowed to select the 'part' and the type of activity they work on [7], we can make an assumption that this random criteria is **partially** fulfilled. The caveat here is that not all users have access to all parts i.e. free users are limited to parts 2 and 5 only.

Algorithm 1 Greedy iterative feature augmentations

```
Input: Feature pool $\Omega$, Dataset $\mathcal{D}$, Max. number of features $\mathcal{N}$ (optional)
Set: Optimal feature representation $\mathcal{S}^*$.
while $size(\mathcal{S}^*) \leq \mathcal{N}$ do
  for $\omega_i \in \Omega$ do
    Set: $\mathcal{S}_i = \mathcal{S}^* + \omega_i$
    $MDP = Construct\_MDP(\mathcal{S}_i, \mathcal{D})$
    $\pi^* = Policy\_Iteration(MDP)$
    $ECR_i = Calculate\_ECR(\pi^*)$
  end for
  Set $\mathcal{S}^* = \mathcal{S}_i$ with highest $ECR_i$.
  Remove feature from pool $\Omega = \Omega - \omega_i$
end while
```

6 SELECTING AN OPTIMUM REPRESENTATION

We now explore a methodology for selecting an 'optimal' representation. An agent's capacity to provide an adaptive learning environment, depends on its ability to leverage information on the student's current cognitive state. This information is inferred from features extracted from the logs. With more features in the representation, one should expect a better approximation of the students cognitive state and consequently a better equipped pedagogical agent to provide effective sequencing. We utilise a greedy 'wrapper' approach in obtaining our optimal representation [37]. This involves a search of the feature space and generation of several candidate feature subsets. Each of these subsets will be evaluated based on its corresponding policy derived from a standard RL solution algorithm such as value- or policy-iteration. The complete procedure is outlined in algorithm 1. Note the limit on the number of features $\mathcal{N}$ can be based off a computational limit or a threshold of minimum support for every unique combination in the feature space.

While our search procedure involves exhaustively looping through every remaining feature in the feature pool Ω to form the subsets, one can alternatively employ a different search algorithm such Monte Carlo tree search [16] or correlation based feature selection [37] to create more informed subsets that are likely to be better candidates. These techniques would be useful in limiting the number of iterations needed in larger feature pools.

7 POLICY EVALUATION

Previous RL applications have been successful in domains such as board games [39], video games [30] and robotics [23]. In these fields, the optimal policy (or set of policies) can be evaluated accurately and inexpensively through simulators or by direct interaction with the environment. However, when the domain involves human interaction, the evaluation becomes more challenging, with live interaction being expensive [27]. Developing simulators is also not straightforward as it could potentially induce some form of bias and have questionable real world accuracy, especially when it comes to complex subjects such as the human mind. The problem becomes amplified when there are multiple policies to evaluate (as there usually are), further increasing the cost of empirical evaluations [21]. Therefore, the offline policy evaluation (OPE) field has been developed specifically to address this issue, where an evaluator needs to provide reliable estimates of policy performance using only past collected data [27].

Round	Representation	ECR	ECR Diff. (%)	Representation Size
Base	MDP_B	238.44	-	64
1	MDP_1	283.63	18.95	256
2	MDP_2	387.42	36.6	2048
3	MDP_3	392.05	1.19	4096
4	MDP_4	396.00	1.01	16384
5	MDP_5	396.00	0	65536

Table 2. ECR results showing best performing representation at each iteration

Representation	Features
MDP_B	topic_fam, correct_so_far,av_time
MDP_1	topic_fam, correct_so_far,av_time,expl_received
MDP_2	topic_fam, correct_so_far,av_time,expl_received, ssl
MDP_3	topic_fam, correct_so_far,av_time,expl_received, ssl, prev_correct
MDP_4	topic_fam, correct_so_far,av_time,expl_received, ssl, prev_correct,av_fam
MDP_5	topic_fam, correct_so_far,av_time,expl_received, ssl, prev_correct,av_fam, time_in_part

Table 3. Representation outputted by Algorithm for each round

8 EVALUATING POLICIES: ECR

As part of the greedy feature augmentation algorithm pipeline, an evaluation metric for π^* is required for each representation at every iteration. Offline policy evaluation (OPE), i.e. the performance estimation of an RL policy without empirical testing, is a field of research in itself, with many proposed techniques and each with their unique advantages/disadvantages [13, 33, 43]. To evaluate the outputted policies we use the Expected Cumulative Reward (ECR) metric which computes the average of the expected return (or cumulative reward) under the policy, across all initial states in the dataset [27].

$$ECR = \mathbb{E}_{s_0 \sim \mathcal{D}, \pi^*} Q(s_0, \pi^*(s_0)) \tag{4}$$

The $Q(s, a)$ function is simply the 'action value' of taking action a in state s and thereafter following π, and s_0 is the initial state. In each state representation, the initial state of every user in the EdNet is the same. This is because we lack any prior information of the user before they begin the course on *Santa*. Traditionally in ITS implementations, the initial state would capture information from the students pre-test scores and so would vary across the students in the dataset. For EdNet, the ECR is simply the state value of the unique initial state of each representation $ECR = V_{\pi^*}(s_0)$ given the policy π^*.

9 RESULTS & DISCUSSION

A summary of the results from the greedy iterative augmentations is provided in table 2. The feature description for the corresponding MDP representations are given in table 3. The base MDP is denoted as 'MDP_B'. The 'ECR Diff.' column shows the percent improvement in ECR relative to the smaller representation preceding it. The 'Representation Size' column illustrates the size of the state feature space. This is dependent on each constituent feature's bin size, i.e. the number of discrete bins allocated. Note that results presented here are only showing the best performing representation at each round of the feature augmentation.

The largest spike in ECR followed at the second round of augmentations with the addition of the ’steps-since-last’ (ssl) feature with an increase of 36.6% over the preceding representation. This features measures the number of steps or activities (questions/lectures) consumed since the current part was last encountered. This feature is inferring the ’forgetting’ element during the learning process and was inspired by the ’spacing effect’ described in Ebbinghaus [15]. Early research in instructional sequencing in language learning used models of forgetting to great success [1]. Our findings concur with this, in that by including ’ssl’ into the feature space, we dramatically increased the agent’s performance. One could argue that this ECR increase was more influenced by the larger bin allocation to ’ssl’ (8 relative to 4 for most other features) rather than the actual utility of the domain information it is measuring. However, if that were the case, then we would expect ’ssl’ to be the first feature added to the base representation. This was not the case since the best performing feature in the first round of augmentations was ’expl_received’, a 4-size bin feature. Nonetheless, further exploration is needed to concretely rule out the factor of the larger bin size.

At the final round of iteration, the best performing representation ’MDP_5’ only equals the performance of preceding representation ’MDP_4’. Though we did not have a specified limit imposed on the number of features, $\mathcal{N}$, the performance plateau exhibited at this final round indicated a suitable termination point for the augmentation algorithm. And since ’MDP_4’ produced equal performance to ’MDP_5’ with a smaller representation size, it was chosen to be the optimum representation within this feature pool. Notice that ’avg_fam’ (a rejected candidate for topic familiarity in the baseline representation) was reintroduced into the feature pool before the greedy iterations algorithm commenced and was automatically selected at the fourth round of augmentations. This was done to leverage its interpretable properties for further domain related analysis such as in section 9.3 (unlike ’part_fam’ which is less interpretable due to the autoencoded feature space).

9.1 Policy Support Analysis

The optimal policy derived from the Policy Iteration algorithm is deterministic, where a single action is prescribed at each state. Meanwhile, the estimated behaviour policy is stochastic, where a distribution across actions is provided at each state. A comparison of the two policies, one from ’MDP_B’ (denoted as PI) and the behaviour policy (denoted as BP) is shown in table 4. Note for comparison purposes, the most common action (with greatest probability mass) in each state in displayed from the stochastic BP policy. Ten random states were sampled for this comparison. The most common actions under the BP policy are mostly within part 5. This is because a large portion of the user population (free users) are limited to certain parts. ’PI Prob’ illustrates the conditional probability of observing the prescribed action under ’PI Policy’ given the state in the dataset. ’PI Support’ provides the actual number of observations in the dataset supporting the $(s, \pi_{PI}(s))$ tuple in the dataset. ’State Support’ is the number of times the state was observed in the dataset. Note that $PI\ Prob = \frac{PI\ Support}{State\ Support}$. ’BP Prob’ is the probability mass of the most common action under the stochastic behaviour policy.

For some states, we can observe very low supporting observations for actions prescribed by the PI Policy. This finding demonstrate that model-based methods can be very statistically biased. In a noisy environment like human learning, the policy’s performance might be vulnerable to the uncertainties of the model. An investigation into the sensitivity of the different representations to noise is performed in section 9.3.

State	PI Policy	PI Prob	State Support	PI Support	BP Policy	BP Prob
212	35	0.000079	304915	24	22	0.172337
344	4	0.028779	384241	11058	24	0.11656
141	32	0.000486	372520	181	24	0.378449
211	35	0.000086	266083	23	22	0.159525
223	35	0.000082	291366	24	24	0.146513
123	9	0.045657	133190	6081	24	0.22811
132	1	0.008887	143803	1278	24	0.262401
444	4	0.025812	675238	17429	29	0.0617
312	35	0.000076	144888	11	22	0.160441
234	3	0.011005	211817	2331	24	0.130688

Table 4. Comparison of policy support for 'MDP_B' (PI) and behaviour policy (BP) from 10 randomly sampled states.

9.2 Penalising Uncertainties: Avoiding Unseen Actions in the Policies

Continuing this analysis, we proceed to check if any policies were enforcing unseen state-actions pairs, with 0 support from dataset. This should not occur under policies derived from tabular methods as explained in section 3. By default, our PI algorithm prescribes state-action pairs a value of 0 if it was never observed. Upon inspection, we discovered that some of the larger representations yielded policies with unseen actions. The policy derived from 'MDP_4' prescribed unseen actions in 10 states. While this was a small fraction of the total state space (around 65,000), unseen actions are an important issue to address because, in the tabular case, any state-action value estimates must be derived only from related experiences [40].

We discovered that the problematic states had very little support in the dataset and were only observed transitioning to themselves, before the episode ends. In the few times the state was visited, a negative or 0 reward was produced. Since these states would only transition to themselves, the values of these valid actions were either negative (or 0). Hence, from the algorithms perspective, an invalid (unseen) action with a default value of 0, was preferable (or equal) to the observed actions.

To combat this issue, we modified the MDP representations to strongly penalise the rewards from unseen state-action pairs, in the form of a -9999 reward. This discouraged the agent from choosing such actions even if the only valid actions yielded 0 or negative returns (The worse case is bounded from below at $\sum_{k\rightarrow\infty}\gamma^k \times (-8)$ and is clearly higher than $\sum_{k\rightarrow\infty}\gamma^k \times (-9999)$). With these changes in place, we observe no unseen actions in any of the policies. The performance rank of representations remained constant with the ECR changes almost negligible. This is because the states involved were observed very infrequently and occupied a small probability mass in the transition probabilities. Although this fix was not significant to our results, we wanted to demonstrate a technique in handling the uncertainty induced by unseen or out of distribution (OOD) actions in model-based RL. Interestingly, research by [26] implemented a 'pessimistic policy iteration' that similarly penalises insufficiently supported state-action tuples (filtered by a threshold). Note the analyses that follow will utilise the penalised representations.

In this section we evaluate the policies under the MC Policy Evaluation. A curve is plotted for the returns (cumulative rewards) from the initial state as the rollout progresses until the 1000th step for a total of 100 rollouts. The 95% confidence intervals are plotted around the mean value of the rollouts. This analysis is shown in figure 2. As with the ECR, we can see that the improvements start to diminish significantly after the second round of augmentations. The performance of the estimated stochastic **behaviour policy** under this simulation is illustrated as a baseline. From this comparison, we

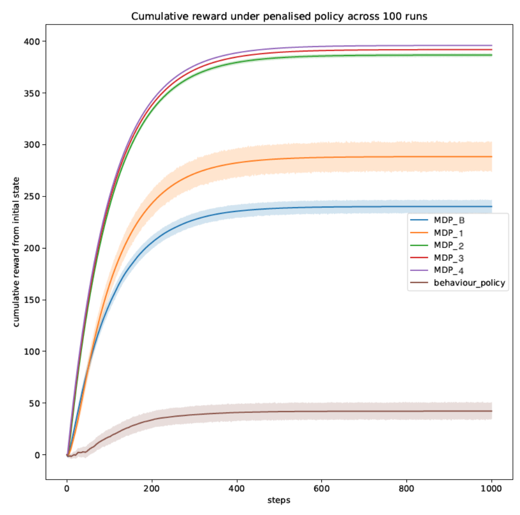

Fig. 2. Returns from the initial state s_0 as the episode progresses under the policies from the associated representations

see much better student performance under the RL policies than the baseline, signalling that the adaptive behaviour of the agent under RL framework is superior than the strategies used in the behaviour policy. We can also conclude that the larger representations exhibit better performance, potentially owing to a better approximation of the cognitive state as hypothesized.

9.3 Evaluating Performance with Different Student Types

A follow up is to test the robustness of the original policies under perturbations of the environment dynamics. These perturbations are domain informed and are designed to correspond to 'stronger' and 'weaker' students types. Algorithm 2 was created to introduce these domain informed perturbations.

Algorithm 2 Domain informed perturbations

```
Input: Set of features to perturb $\bar{\Omega}$, MDP transition probabilities $\mathcal{P}_{MDP}$, set of domain filters for each feature $\psi$, positive perturbation constant $c = 0.05$
for $p_{s,a,s'} \in \mathcal{P}_{MDP}$ do
    $\Delta_{s,a,s'} = p_{s,a,s'} + \sum_{\omega \in \bar{\Omega}} \Delta_\omega$ Where $\Delta_\omega = \begin{cases} c, & \text{if } \omega_s, \omega_{s'} \text{ satisfies } \psi_\omega \\ 0, & \text{else} \end{cases}$
end for
Adjust $\Delta_{s,a,s}$ relative to others within the $s, a$ pair
$\Delta_{s,a,s} = \Delta_{s,a,s} - \frac{1}{|\psi|} \sum_{s'} \Delta_{s,a,s} \; \forall \Delta_{s,a,s'}$
Set perturbed transition probabilities $\bar{\mathcal{P}}_{MDP} = \mathcal{P}_{MDP}$
for $p_{s,a,s'} \in \bar{\mathcal{P}}_{MDP}$ do
    $p_{s,a,s'} = \max(p_{s,a,s'} + \Delta_{s,a,s}, 0)$
end for
$p_{s,a,s'} = \frac{p_{s,a,s'}}{\sum_{s'} p_{s,a,s'}} \; \forall p_{s,a,s'}$
Return: $\bar{\mathcal{P}}_{MDP}$
```

Feature to Perturb	Strong	Weak
Topic_fam	$\omega_{s'} > \omega_s$	$\omega_{s'} = \omega_s$
Correct_so_far	$\omega_{s'} \geq \omega_s$	$\omega_{s'} < \omega_s$
Avg_time	$\omega_{s'} \leq \omega_s$	$\omega_{s'} > \omega_s$

Table 5. Domain perturbation filters,ψ for each feature in $\bar{\Omega}$ for the 'Strong' and 'Weak' perturbed MDPs, $\bar{\mathcal{P}}$ respectively

The filter ψ is a set of domain informed filters for each perturbed feature. In our implementation we perturb three base features that were common in all representations i.e. "topic_fam", "correct_so_far" and "av_time". The domain rules for the two separate perturbations 'Strong' and 'Weak' are defined in table 5. For example, take the feature "correct_so_far" and the 'Strong' user case. Here we set the filter to capture transitions where the next state s' registers a greater or equal value relative to the current state s. When this filter is inputted in algorithm 2, the transitions that satisfy this filter will be boosted by the constant c. This ultimately has the effect of increasing the probability mass of this transition, perturbing the original MDP to make such transitions more likely. The results of performing these two separate perturbations are visualised by performing a policy evaluation algorithm with the original policy but under the perturbed MDPs (Strong & Weak) as the simulators.

Figure 3 shows the results of this analysis for the different representations. Notice that in all the representations, the original MDP always yielded the best performance. This is expected, since the original policy was derived to perform optimally on the original MDP. However, as the representation size increases, the effects of the perturbations becomes less pronounced, almost becoming negligible past 'MDP_1'. To determine if the larger representation would be affected with more features perturbed, we conducted another round of perturbations, this time only on 'MDP_4' and with **all** of its features (barring 'ssl') perturbed. Surprisingly, the performance of the policy was not affected by the perturbations. This could mean that the larger representations are more robust towards deviations from the expected dynamics derived from the data. Hence, we can have more confidence that such policies would be robust in the real-world setting, maintaining its performance for students that exhibit different learning characteristics from our observations in EdNet.

10 DOMAIN RELATED FINDINGS

By observing the state values and policies derived from the RL algorithms, we can discover interesting insights in how the agent perceives the information in the states and how it behaves accordingly. In figure 4, we plot the derived state values against two features in 'MDP_B'. Based on our reward design, the state values indicates the future user performance. From the agent's perspective, the expected future performance is much higher when the student has a high correct-incorrect answer ratio. However, the relationship between the average time is more complex. At higher values of 'correct_so_far', a higher 'av_time' entails a larger state value, but when 'correct_so_far' is low, the opposite is true. Even if the policies themselves are not used, findings like this can inform us of useful features and their relationships in predicting future user performance.

Looking in the action choices in the policies, we discover that the RL algorithms tend to put preference on question level 4 actions. Indeed these do yield the highest reward and the lowest punishment in our reward design. One possible extension is to investigate how a change in the reward function design would impact the policy preferences. Even if the policies are not deployed, such patterns can be useful as a technique in letting the data guide pedagogical strategies.

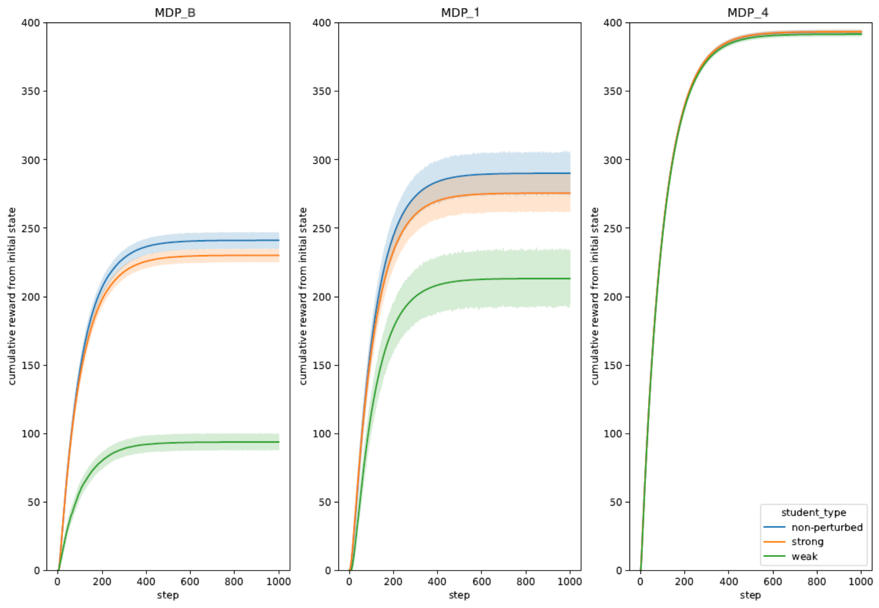

Fig. 3. MC Policy Evaluation of the original policy under the perturbed MDPs

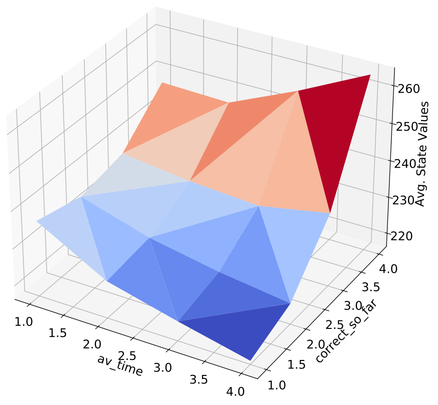

Fig. 4. State values vs features

11 CONCLUSIONS AND FUTURE WORK

In this paper we approached the challenge of designing an adaptive RL based pedagogical agent that can provide an optimized sequencing of learning materials to maximize learning. Training an RL agent with actual users is far too resource intensive. Therefore, we simultaneously tackle the problem of training and evaluating an RL algorithm offline based only on pre-collected data. A purely data-driven student model was created for this purpose. We hypothesized that a complex model is required to capture the intricacies of human learning. To investigate this theory, a large dataset, EdNet, was necessary to provide sufficient support for the models.

Our student model was constructed in the form of a data-derived MDP, with the transition and reward dynamics estimated from the observations in the data. The raw logs were transformed into domain inspired features. An action set was established to delineate our agent's possible controls and a reward function was designed to incentivise the agent in maximizing learning. Without NLG, we base our measurement of learning on the perceived student performance. By using the MDPs we then trained our agents with the model-based Policy Iteration algorithm. To determine whether a more complex model yields better tutoring, we employed a greedy iterative augmentation procedure. The ECR metric guided how we chose our features and demonstrated the positive relationship between representation complexity and policy performance. In our analyses we discovered issues with Out of Distribution actions in the policies and presented a solution in the form of penalising rewards. We further evaluated our policies using a modified Monte Carlo Policy Evaluation algorithm and tested their robustness against domain informed perturbations of the dynamics. We show that the larger representation are less impacted by the perturbations and therefore can provide a more equal learning experience for stronger or weaker students.

Several limitations are acknowledged which consequently open up further investigations. The influence of the bin-size on feature preference in the representations was discussed briefly but lacked conclusive evidence to rule out entirely. This work is necessary to ensure that the features are selected based only on the utility of the domain information it captures. From our model-based policy analyses we also discovered out-of-distribution actions in the policy space. Though we managed to remedy the problems for completely unseen actions through strong penalisation, the next course of action is to also penalise low supported actions/states **variably** according to their uncertainty as was explored by [26][45]. Finally, the inferred policies should be evaluated in the real world in a controlled study.

REFERENCES

[1] Richard C Atkinson. 1972. Optimizing the learning of a second-language vocabulary. *Journal of experimental psychology* 96, 1 (1972), 124.

[2] Jonathan Bassen, Bharathan Balaji, Michael Schaarschmidt, Candace Thille, Jay Painter, Dawn Zimmaro, Alex Games, Ethan Fast, and John C Mitchell. 2020. Reinforcement learning for the adaptive scheduling of educational activities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[3] Joseph Beck, Beverly Park Woolf, and Carole R Beal. 2000. ADVISOR: A machine learning architecture for intelligent tutor construction. *AAAI/IAAI* 2000, 552-557 (2000), 1–2.

[4] Joanne Capper. 2001. E-learning growth and promise for the developing world. *TechKnowLogia* 2, 2 (2001), 7–10.

[5] Min Chi, Kurt VanLehn, and Diane Litman. 2010. Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. In *International conference on intelligent tutoring systems*. Springer, 224–234.

[6] Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. 2010. Inducing effective pedagogical strategies using learning context features. In *International conference on user modeling, adaptation, and personalization*. Springer, 147–158.

[7] Youngduck Choi, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park, Seewoo Lee, Jineon Baek, Chan Bae, Byungsoo Kim, and Jaewe Heo. 2020. Ednet: A large-scale hierarchical dataset in education. In *International Conference on Artificial Intelligence in Education*. Springer, 69–73.

[8] Albert T Corbett, Kenneth R Koedinger, and John R Anderson. 1997. Intelligent tutoring systems. In *Handbook of human-computer interaction*. Elsevier, 849–874.

[9] Mariee Josee Cross. 2015. Markov Decision Processes (MDP) Toolbox.

[10] Peter Diggle, Peter J Diggle, Patrick Heagerty, Kung-Yee Liang, Patrick J Heagerty, Scott Zeger, et al. 2002. *Analysis of longitudinal data*. Oxford university press.

[11] Fabiano A Dorça, Luciano V Lima, Márcia A Fernandes, and Carlos R Lopes. 2013. Comparing strategies for modeling students learning styles through reinforcement learning in adaptive and intelligent educational systems: An experimental analysis. *Expert Systems with Applications* 40, 6 (2013), 2092–2101.

[12] Shayan Doroudi, Vincent Aleven, and Emma Brunskill. 2019. Where's the reward? *International Journal of Artificial Intelligence in Education* 29, 4 (2019), 568–620.

[13] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601* (2011).

[14] E Duffin. 2019. E-learning and digital education-Statistics & Facts. *Retrieved December* 22 (2019), 2019.

[15] Hermann Ebbinghaus. 1885. *Über das gedächtnis: untersuchungen zur experimentellen psychologie*. Duncker & Humblot.

[16] Romaric Gaudel and Michele Sebag. 2010. Feature selection as a one-player game. In *International Conference on Machine Learning*. 359–366.

[17] Thomas Grosges and Dominique Barchiesi. 2007. European credit transfer and accumulation system: An alternative way to calculate the ECTS grades. *Higher Education in Europe* 32, 2-3 (2007), 213–227.

[18] Ronald K Hambleton, Richard J Shavelson, Noreen M Webb, Hariharan Swaminathan, and H Jane Rogers. 1991. *Fundamentals of item response theory*. Vol. 2. Sage.

[19] Wayne Holmes, Maya Bialik, and Charles Fadel. 2019. Artificial intelligence in education. *Boston: Center for Curriculum Redesign* (2019).

[20] Ana Iglesias, Paloma Martínez, Ricardo Aler, and Fernando Fernández. 2009. Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. *Knowledge-Based Systems* 22, 4 (2009), 266–270.

[21] Song Ju, Shitian Shen, Hamoon Azizsoltani, Tiffany Barnes, and Min Chi. 2019. Importance Sampling to Identify Empirically Valid Policies and their Critical Decisions.. In *EDM (Workshops)*. 69–78.

[22] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. 2019. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374* (2019).

[23] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (2013), 1238–1274.

[24] Hoang Le, Cameron Voloshin, and Yisong Yue. 2019. Batch policy learning under constraints. In *International Conference on Machine Learning*. PMLR, 3703–3712.

[25] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[26] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. 2020. Provably good batch off-policy reinforcement learning without great exploration. *Advances in Neural Information Processing Systems* 33 (2020), 1264–1274.

[27] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. 2014. Offline policy evaluation across representations with applications to educational games.. In *AAMAS*, Vol. 1077.

[28] Ye Mao. 2019. One minute is enough: Early prediction of student success and event-level difficulty during novice programming tasks. In *In: Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019)*.

[29] Robin Mason and Frank Rennie. 2006. *Elearning: The key concepts*. Routledge.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[31] Chen Pojen, Hsieh Mingen, and Tsai Tzuyang. 2020. Junyi Academy Online Learning Activity Dataset: A large-scale public online learning activity dataset from elementary to senior high school students. *Dataset available from https://www.kaggle.com/junyiacademy/learning-activity-public-dataset-by-junyi-academy* (2020).

[32] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. 2018. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081* (2018).

[33] Doina Precup. 2000. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series* (2000), 80.

[34] Frank E Ritter, Josef Nerb, Erno Lehtinen, and Timothy M O'Shea. 2007. *In order to learn: How the sequence of topics influences learning*. Oxford University Press.

[35] Jonathan Rowe, Bob Pokorny, Benjamin Goldberg, Bradford Mott, and James Lester. 2017. Toward simulated students for reinforcement learning-driven tutorial planning in GIFT. In *Proceedings of R. Sottilare (Ed.) 5th Annual GIFT Users Symposium. Orlando, FL*.

[36] Avi Segal, Yossi Ben David, Joseph Jay Williams, Kobi Gal, and Yaar Shalom. 2018. Combining difficulty ranking with multi-armed bandits to sequence educational content. In *International conference on artificial intelligence in education*. Springer, 317–321.

[37] Shitian Shen and Min Chi. 2016. Aim Low: Correlation-Based Feature Selection for Model-Based Reinforcement Learning. *International Educational Data Mining Society* (2016).

[38] Shitian Shen and Min Chi. 2016. Reinforcement learning: the sooner the better, or the later the better?. In *Proceedings of the 2016 conference on user modeling adaptation and personalization*. 37–44.

[39] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.

[40] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[41] Joel R Tetreault and Diane J Litman. 2008. A reinforcement learning approach to evaluating state representations in spoken dialogue systems. *Speech Communication* 50, 8-9 (2008), 683–696.

[42] George Veletsianos and Gregory S Russell. 2014. Pedagogical agents. In *Handbook of research on educational communications and technology*. Springer, 759–769.

[43] Cameron Voloshin, Hoang M Le, Nan Jiang, and Yisong Yue. 2019. Empirical study of off-policy policy evaluation for reinforcement learning. *arXiv preprint arXiv:1911.06854* (2019).

[44] Beverly Park Woolf. 2010. *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann.

[45] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239* (2020).