Learning a Policy Primes Quality Control: Towards Evidence-Based Automation of Learning Engineering

Machi Shimmei
North Carolina State University
Raleigh, NC 27695
mshimme@ncsu.edu

Noboru Matsuda
North Carolina State University
Raleigh, NC 27695
noboru.matsuda@ncsu.edu

ABSTRACT

One of the most challenging issues for online courseware engineering is to maintain the quality of instructional components, such as written text, video, and assessments. Learning engineers would like to know how individual instructional components contributed to students' learning. However, it is a hard task because it requires significant expertise in learning science, learning technology, and subject matter pedagogy. To address this challenge, we propose an innovative application of reinforcement learning (RL) as an assessor of instructional components implemented in given online courseware. After students activities are converted into Markov decision process (MDP), a collection of actions (each corresponds to an instructional component) suggested as a policy is analyzed. As a consequence, the usefulness of individual actions with regards to achieving ideal learning outcomes will be suggested. The proposed RL application is invented for human-in-the-loop learning engineering method called RAFINE. In the RAFINE framework, a machine generates a list of the least contributing instructional components on the given online courseware by interpreting the whole policy. The courseware developers modify those suggested components. As a proof of concept, this paper describes an evaluation study where online learning was simulated on hypothetical online courseware. The results showed that over 90% of ineffective instructional components were correctly identified as ineffective on average.

Keywords

Automated Learning Engineering, Evidence-based learning Engineering, Iterative Courseware development, Reinforcement Learning

1. INTRODUCTION

With the rapidly growing popularity of online courses, there has been a heavy demand for practical learning engineering methods for designing effective online courseware [18]. Even though there are known design principles that provide theoretical insights into designing effective online courses [e.g., 5, 6, 10], such principle-based approaches still require iterative engineering for practical courseware development at scale [8].

One of the challenges in the principle-based iterative learning engineering is to identify issues with the courseware. After an initial version of courseware is used by students, instructors (or learning engineers) analyze the interaction and learning outcome data to improve the quality of the courseware. However, interpreting those data to determine actual refinement plans is extremely challenging and requires significant expertise in learning science, learning technology, and the subject matter pedagogy [5, 20]. The commonly used analytic techniques are the learning curve analysis [14] and the assessment items analysis [16]. However, these techniques only apply to assessment items while other types of instructional components such as video clips and written texts must also be included in the analysis.

There is therefore a gap between an ideal learning-engineering model to *efficiently* build *effective* online courseware and the actual technology infrastructure available for building online courseware. To fill this gap, evidence-based learning engineering method that identifies deficits of the given online courseware is needed.

Our solution is an innovative application of the reinforcement learning (RL) technique that we call RAFINE (Reinforcement learning Application For INcremental courseware Engineering). RAFINE identifies instructional components that have relatively less contribution to students' learning in the given courseware. RAFINE is a building block for the evidence-based, human-in-the-loop, iterative learning engineering method that we call the RAFINE method. Figure 1 shows how a human and a machine collaborate to iteratively improve the quality of courseware in the RAFINE method.

Given a record of individual students' learning trajectory logs on particular online courseware, RAFINE first converts learning trajectories into a state transition graph. Here, states represent students' intermediate learning status and the transition is caused by taking an instructional component (i.e., watching a video). All students' state transition graphs are then consolidated into a single Markov decision process (MDP) by merging the same states. RAFINE then applies a variant version of value iteration technique commonly used for RL to compute a *converse policy* that represents the least optimal instructional components to be taken at any given moment to achieve students' ideal learning goals. The entire policy actions will be then analyzed to identify instructional components that have relatively less contributions to students' learning. The list of detected less-effective components is provided to instructors as a *recommendation* for courseware refinement. Any type of

Machi Shimmei and Noboru Matsuda "Learning a Policy Primes Quality Control: Towards Evidence-Based Automation of Learning Engineering" In: *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, Anna N. Rafferty, Jacob Whitehill, Violetta Cavalli-Sforza, and Cristobal Romero (eds.) 2020, pp. 224 - 232

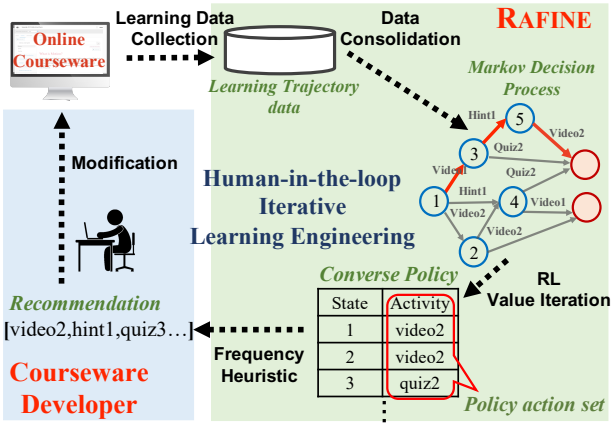

Figure 1 : Overview of the RAFINE method

instructional components such as lecture videos, assessment quizzes and hints can be analyzed by the RAFINE method.

As a proof of concept, the goal of the current paper is to conduct a study to evaluate the validity of implementation of the RAFINE method. We conducted a simulation study where learning log data were generated by simulating learning activities in mock online courseware. The result showed that the RAFINE method correctly identified over 90% of ineffective instructional components in the given mock courseware.

The primary contributions of the current work are essentially theoretical yet have the potential for practical use:
(1) We proposed RAFINE as a building block for evidence-based, human-in-the-loop, iterative online courseware learning engineering method. RAFINE analyzes complicated learning trajectory data and suggests deficits of the courseware by evaluating a broader range of instructional components. The evaluation study showed the theoretical effectiveness of RAFINE.
(2) We innovated a technique to interpret a policy induced by reinforcement learning as a whole to detect a relative weakness among available actions (which correspond to instructional components) with regard to achieving ideal goals.

The rest of the paper is structured as follows. Section 2 discusses related work on evidence-based learning engineering and the applications of reinforcement learning for education. Section 3 technically elaborates how RL works on learning trajectory data and describes how RAFINE interprets the converse policy. Section 4 introduces research questions. Then, section 5 explains how simulation data were created for the evaluation study. Section 6 reports results of the evaluation study along with the research questions, and section 7 discusses the results and limitations. Finally, section 8 concludes that RAFINE serves as a building block for the evidence-based iterative learning engineering method.

2. RELATED WORK

2.1 Evidence-based Learning Engineering

The process of learning engineering includes cognitive task analysis, designing and delivering instructional components, measuring students' understanding, and evaluating the courseware design [6]. Quickly cycling through these tasks is a key for successful iterative improvement of online courseware. Since each of these tasks is considerably labor intensive, the automatization of the engineering process is an important research agenda.

One of the most actively studied areas of learning engineering is student modeling [14]. Both a representation (to understand what must be modeled) and a recognition (to understand how to gauge students' competency with the proposed representation) are important research topics. Cen *et al.* [3] proposed Learning Factors Analysis (LFA) for semi-automated evaluation and improvement of knowledge component (KC) models that represent a set of latent skills and knowledge that students are supposed to learn [12]. LFA performs a combinational search for a KC model that best fits students' learning data across existing KC models. Although LFA requires "seed" KC models, some more recent works reported automatic discovery of KC models from students' learning data [9, 13].

Another actively studied area of learning engineering is an automated question generation. Du and Cardie [7] proposed a method for automatically generating questions. The method identifies the question-worthy sentences from a passage using a hierarchical neural model with a sentence-level sequence tagging. Mazidi and Tarau [15] introduced a method to classify sentences based on what the sentence is communicating as a basis for generating questions. The type of syntactic and semantic constituents of sentences and their arrangement were analyzed in the study.

Automation of assessment grading is also an important part of learning engineering in particular for online courses to be scaled. Zhang *et al.* [25] tackled the task of Automatic Short Answer Grading (ASAG). Short answer questions ask students to answer in natural language with the length of one phrase to one paragraph. The authors compared Deep Belief Networks (DBN) against five machine learning techniques (Naïve Bayes, Logistic Regression, Decision Tree, Artificial Neural Network, and Support Vector Machine) for automatically grading short answer questions.

Yet another essential part of practical learning engineering is to identify deficits of courseware content to be revised. RAFINE focuses on this aspect of the learning engineering. As far as the authors are aware, there have been very few studies in this line of research. Bodily *et al.* [2] mentioned the lack of efforts to use learning analytics as a basis for redesigning an online course. The authors proposed the RISE framework for redesigning instructional components in online courseware. This framework provides a metric that combines the usage of instructional components and students' grades to identify instructional components that are good candidates for improvement efforts. The goal of the RAFINE method is also to detect instructional components that were not very useful for learning. Unlike the RISE framework, however, RAFINE evaluates a potential contribution of implemented instructional components to students' learning and provides a list of instructional components as a recommendation for refinement.

2.2 RL for Decision Making in Education

Reinforcement Learning (RL) has been used in education applications in particular to compute optimal pedagogical strategies for adaptive tutoring.

Shen and Chi [19] applied RL to induce the policy on whether the intelligent tutoring system should propose worked example (WE) or problem solving (PS) to students for the next activity. The authors induced policies using two different rewards: Immediate and Delayed. The policy was computed based on the number of problems solved, the average time taken, the difficulty of the problem, and students' performance on the past PS. The result

showed that immediate policies give more WE while delayed policies give more PS.

Rafferty *et al*. [17] used the partially observable Markov decision process (POMDP) framework to formulate the process of teaching. The authors applied RL to induce optimal teaching actions such as a quiz or an example to minimize the amount of time spent on learning materials. They found that students who learned with RL induced teaching actions spent less time than students who did not.

Iglesias *et al*. [11] applied RL to induce a policy on teaching decisions such as which topic students should do next and which task students should do on the topic. They conducted an evaluation study in a database design course for undergrad students in Computer Science. In the evaluation study, they found that students with the machine generated policy spent less time on the adaptive and intelligent educational system, but they could not find a significant difference in the students' final level of knowledge.

In addition to inducing the sequence of instructional components, RL has been applied to induce dialog moves or narrative events. Chi *et al*. [4] applied RL to induce pedagogical policies that decide whether the tutor should ask students to justify the answer, tell the next step directly, or elicit the next step information from a student in a dialogue-based tutor. Tetreault and Litman [21] estimated the reliability of a policy derived from a spoken dialog tutoring system. Wang *et al*. [22] applied deep RL for interactive narrative generators that tailor each player's story in an educational game. The authors prepared several events of a story and induced policies on how event sequences should unfold based on player interaction logs.

The applications of reinforcement learning to induce pedagogical strategies are widely studied in various subjects from middle school math to college-level database design, and in various kinds of tutoring systems such as task-based, dialog-based, and game-based systems. The effects of educational RL policy have been tested both with real and simulated data. Some showed positive effects of the policy while others did not.

What makes our study different from these studies is the way we use the induced policy. In the previous studies mentioned above, the induced policy is directly used to provide an *optimal action at each learning status*. On the other hand, RAFINE does not use the policy to make a decision on which instructional component students should take next. Instead, RAFINE interprets the induced policy as a whole to identify instructional components that have relatively less contribution to learning. More specifically, RAFINE focuses on how often each instructional component is suggested by a policy (we call this the *frequency heuristic* as described in section 3.5). In RAFINE, the induced policy is not utilized as an educational strategy for students, but an analysis of the policy is used for courseware developers to improve courseware.

3. TECHNICAL DETAILS OF RAFINE

3.1 Overview of the RAFINE Method

In the RAFINE method, an initial version of the online courseware is used by students and their activities are logged. These activity data consist of standard clickstream data including students' responses for formative assessments and their correctness. We call these activity data the *learning trajectory data*.

The right side of Figure 1 shows how learning trajectory data are processed. The learning trajectory data from all students are first consolidated into a single state transition graph called *learning trajectory graph* (LTG). The LTG is a Markov decision process (MDP) where states represent students' intermediate learning status and actions represent instructional components taken. LTG is annotated with predefined *rewards* that represent quantitative benefits of the learning activity that causes transition from one state to another in the LTG. Finally, a value iteration technique is applied to compute a *converse policy* that shows the worst action to be taken at each state to achieve the expected learning outcome (represented as a table in Figure 1). As a consequence, a collection of actions suggested by a converse policy corresponds to a set of instructional components that have the least likelihood at each state to contribute to the ideal learning outcome. We call this collection of actions the *policy action set*.

To create a recommendation for refinement based on the induced policy, RAFINE *interprets the policy as a whole*. That is, all actions in a policy action set is analyzed. Note that in most cases, the number of states in the LTG gets larger than the number of instructional components available on the given online courseware. This implies that all instructional components are likely to be included in a policy action set. The relative effectiveness of individual instructional components is therefore analyzed based on the frequency. We call this heuristic the *frequency heuristic*, which is detailed in section 3.5.

Given the recommendation for refinement, courseware developers revise the courseware. The RAFINE method can be iteratively applied to the revised courseware by collecting a new batch of learning trajectory data to further improve the courseware.

3.2 Model Representation

The unit of analysis of the RAFINE method is an instructional component implemented in the online courseware. Instructional components include video, quiz, hint, written paragraph or any other components used in the courseware. We assume a presence of a skill model that contains a set of skills each representing a piece of knowledge that students must learn (aka, knowledge component), and each instructional component is tagged with a single skill. Under this assumption, RAFINE will be applied for each skill separately. This constraint is rooted in our design decision for a state representation described later that involves a measure of proficiency per skill.

Let LT_i^{φ} be a given learning trajectory for student i regarding skill φ. Let ${a_i}^T$ be an instructional component taken (e.g., watching a video or answering a quiz) by student i at time T. A learning trajectory for student i on skill φ, LT_i^{φ}, is represented with ${a_i}^T$ as follows:

$$LT_i^{\varphi} = \left\{ {a_i}^1, \ldots, {a_i}^{n_i^{\varphi}} \middle|\ {a_i}^T \in \Phi^{\varphi}\ ,\ T = 1, \ldots, n_i^{\varphi} \right\}.$$

LT_i^{φ}: learning trajectory for student i regarding skill φ
${a_i}^T$: an instructional component taken by student i at time T
Φ^{φ}: a set of instructional components regarding skill φ
n_i^{φ}: number of activities taken by student i regarding skill φ

To make the explanations simple, without a loss of generality, let's assume that there is only one skill φ in our target online courseware (recall that RAFINE will be applied for individual skills separately). We therefore eliminate the skill index from Φ^{φ} and LT_i^{φ} in the following descriptions unless otherwise desired.

All learning trajectories LT_i for all students i in the given log data are consolidated into a single learning trajectory graph (LTG), which is an MDP. In the LTG, states represent learning status and

edges represent learning activities taken that caused a change in learning status. To consolidate individual students' learning trajectories into a single LTG, each student's learning trajectory LT_i is first converted into a *learning trajectory path*. This is done by chronologically traversing a learning trajectory LT_i while creating states that represent intermediate learning status.

A learning status consists of a pair of *action history* and *mastery level*; $<\boldsymbol{ah}_{i,T}, p_{i,T}(\varphi)>$. Action history $\boldsymbol{ah}_{i,T}$ is a binary vector $<ah_i^1, \ldots, ah_i^K>$ where ah_i^m shows whether student i has taken the m-th instructional component in Φ^φ by time T (assuming the instructional components are ordered and $|\Phi^\varphi| = K$). Mastery level $p_{i,T}(\varphi)$ is a scalar value showing a predicted probability of student i applying skill φ at time T correctly. The value of mastery level is rounded to the nearest multiple of 0.05 (e.g., 0.12 becomes 0.10) to reduce the number of states in the LTG (which will be otherwise intractable).

Mastery level is computed based on the history of learning activities. An underlying assumption is that commitment to a particular type of learning activity would increase the mastery level by a specific amount. There are several known techniques available to achieve this goal including Bayesian models and regression models. As long as masterly level is monotonically updated, any student-modeling technique would work for the RAFINE method.

While traversing the learning trajectory, $\boldsymbol{ah}_{i,T}$ and $p_{i,T}(\varphi)$ are updated accordingly. For example, assume there are six instructional components; Video1, Video2, Quiz1, Quiz2, Hint1, and Hint2. A state s <101000, 0.4> indicates that a student had watched Video1 and took Quiz1 before reaching the state s. It also indicates that a predicted mastery level for skill φ at the time of arriving at the state s was 0.4. Assume that the student answered Quiz1 incorrectly to reach the state s. Now, the student needed to review Hint1, which caused a transition from s to s' where s' is <101010, 0.45> with an assumption that reviewing a hint increased the master level by 0.05.

A learning trajectory path is a linear graph. It might have a loop back to the same state when a certain instructional component was taken more than once with the increase of mastery level less than 0.05. As a side note, moving between pages in the courseware is not encoded in the LTG, because it is not considered as a learning activity.

All individual students' learning trajectory paths are then aggregated into an LTG by merging the same states. As a consequence, the states in an LTG generally have multiple incoming and outgoing edges. Note that in an LTG, student and time (i.e., the parameters i and T in an individual student's learning trajectory path) are abstracted. Therefore, in the following explanations, a tuple representing a state is denoted as $<\boldsymbol{ah}, p(\varphi)>$.

In an LTG, the states where the value of the mastery level, p(φ), is greater than a pre-defined threshold (which is usually 0.85) are called *terminal states*—meaning that students became proficient in applying skill φ. All outgoing edges at terminal states are discarded.

3.3 Reward

A reward value of a particular state depends on the mastery level, p(φ), both at the current and successor states. As an example, consider two students who landed on the same state s, but then took different learning activities. One student reached a successor state by answering an assessment quiz incorrectly (i.e., p(φ) was not increased) whereas the other student watched a video (i.e., p(φ) was increased). In our model, a reward for state s where the student took a learning activity a to reach a successor state s' is defined as:

$$R(s,a,s') = \begin{cases} -0.14 & (ml(s) = ml(s') < 0.85) \\ -0.05 & (ml(s) < ml(s') < 0.85) \\ 0.95 & (0.85 \leq ml(s')) \end{cases}$$

In the equations above, $ml(s)$ returns the mastery level at the state s. A reward at the state s becomes the greatest when the successor state is a terminal state. Otherwise, the rewards are set to be small negative values to reflect students' time commitment while computing a policy as shown in the next section. We assume that the mastery level grows monotonic, i.e., students never unlearn. Therefore, a reward where $ml(s) > ml(s')$ is undefined.

3.4 Converse Policy

Given the reward function R, a value function for state s under a policy π is defined as follows, where $\boldsymbol{S}$ is a set of all states in a given LTG:

$$V^\pi(s) = \sum_{s' \in \boldsymbol{S}} T(s,\pi(s),s')\{\mathrm{R}(\mathrm{s},\pi(s),\mathrm{s}') + \gamma V^\pi(s')\}$$

In the current implementation, the discount factor γ is arbitrarily set to be 0.9. A transition model $T(s, a, s')$ is derived from the collected learning trajectory data of actual students as the probability of students reaching state s' when they took a learning activity a at state s.

In general, a policy suggests an action to be taken in a certain state to maximize the value function [23]. However, considering the purpose of RAFINE, we need to know which instructional components should not be taken—i.e., we need to know which action has the least expected reward. Therefore, through the value iteration, the value function is updated as follows where $A(s)$ shows a set of actions available at state s (i.e., instructional components taken by students at state s):

$$V(s) \leftarrow \min_{a \in A(s)} \sum_{s' \in \boldsymbol{S}} T(s,a,s')\{R(s,a,s') + \gamma V(s')\}$$

After the value function is converged, the action that minimizes the value function for state s is identified. We shall call this policy the *converse policy*:

$$\pi(s) = \operatorname*{argmin}_{a \in A(s)} \sum_{s' \in \boldsymbol{S}} T(s,a,s')\{R(s,a,s') + \gamma V^\pi(s')\}$$

3.5 Frequency Heuristic

Because of the binary vector and mastery level in the state representation, the number of states in any given learning trajectory graph (LTG) is many times more than the number of available actions (i.e., instructional components). Hence it is often the case that each of the instructional components in the courseware is selected as a policy action many times. Therefore, whether the instructional component has been selected as an action is not a sufficient criterion to decide the component should be included in a recommendation for refinement.

To create a recommendation from the induced converse policy, RAFINE interprets the collection of policy actions over all states in the LTG based on a frequency— *actions that frequently appear in the converse policy action set* will be included in a recommendation as culprit for poor performance. We call this heuristic the *frequency heuristic*. The frequency heuristic is based on the hypothesis that relatively ineffective instructional components tend to appear in a

policy action set of a given converse policy more frequently than effective ones. We will verify this hypothesis by comparing the mean normalized frequency of ineffective and effective components in the policy action set.

The empirical question is then "how frequent is frequent?" We examined two frequency cut-offs through an evaluation study as described in the results section.

4. RESEARCH QUESTIONS

Our central question is whether RAFINE can suggest deficits of the courseware based on the students' learning trajectory data. To address this question, we divide it into the following two research questions:

RQ1: How robust is the converse policy as a detector for relatively ineffective instructional components against different conditions of learning data?

RQ2: How accurately does the frequency heuristic compose a recommendation for courseware refinement?

To answer these questions, and also as a proof of concept for RAFINE, we conducted an evaluation study as described in the next section.

5. EVALUATION STUDY

For a rigorous evaluation of the RAFINE method, it is necessary to conduct a study with learning data collected from students working on actual online courseware. As mentioned earlier, the online courseware must be structured with a skill model tagged to individual instructional components to apply RAFINE. To the authors' knowledge, however, no such online courseware is available at this moment and building RAFINE compatible online courseware requires a considerable amount of time. Therefore, we conducted a simulation study as a proof of concept towards an evaluation with actual students. The current evaluation study uses hypothetical learning trajectories in mock online courseware.The results from the current simulation study justify future efforts of building an RAFINE compatible online courseware or tagging a KC model to individual instructional components in existing courseware.

In this evaluation study, we address the research questions mentioned in section 4. We created mock online courseware where there was only one skill involved. As described above, when there were multiple skills involved, RAFINE had to be applied separately to each skill. Therefore, this assumption does not harm the generality of the study.

All instructional components in mock online courseware were tagged as either *effective* or *ineffective*. In the current simulation study, we included three types of instructional components: (1) videos, (2) formative assessments (aka quizzes), and (3) hint messages associated with formative assessments. Learning trajectories were generated by simulating students' learning activities. For the sake of explanation, we use a phrase 'simulated students' to refer to hypothetical students in the simulation.

In the real world, the growth of mastery level depends on the learning activities actually taken and students' latent traits of learning. In the current simulation, the masterly level shows a probability of answering a quiz correctly and the simulated students' performance on a quiz was determined by the masterly level. The growth of the mastery level, $p_{i,T}$, was simulated using a logistic regression model as shown below:

Table 1: The means μ and standard deviations σ used for the simulation study to model the growth of mastery level.

(a)

The value of $(\mu_1\ ,\ \sigma_1)$ where $\delta_1(c, e(a_{i,T-1}))\ \sim \max(0, \mathcal{N}(\mu_1\ ,\ \sigma_1^2))$

	Contrast: c		
Effectiveness: $e(a_{i,T-1})$	Large	Moderate	Small
Effective	0.5 0.01	0.4 0.05	0.3 0.10
Ineffective	-0.1 0.01	0.0 0.05	0.1 0.10

(b)

The value of $(\mu_2\ ,\ \sigma_2)$ where $\delta_2(rspns(a_{i,T-1}))\sim \max(0, N(\mu_2, \sigma_2^2))$, or 0 if $a_{i,T-1}$ was not a quiz.

$rspns(a_{i,T-1} \in quiz)$	μ_2 σ_2
Correct	0.05 0.01
Incorrect	0.03 0.01

Table 2: The means and standard deviations used for computing the initial logit $Z_{i,0}$.

The value of (μ_0, σ_0) where $Z_{i,0} \sim \max(0, \mathcal{N}(\mu_0\ ,\ \sigma_0^2))$

	Contrast: c		
	Large	Moderate	Small
μ_0, σ_0	-0.95 0.01	-0.95 0.10	-0.95 0.20

$$p_{i,T} = \left[\frac{1}{1 + e^{-Z_{i,T}}}\right]$$

$$Z_{i,T} = Z_{i,T-1} + \delta_1\big(c, e(a_{i,T-1})\big) + \delta_2(rspns(a_{i,T-1}))$$

The $[X]$ operator is to round the value X to the nearest multiple of 0.05 and $a_{i,T-1}$ is an instructional component that a simulated student *i* took at time *T*-1.

Logit $Z_{i,T}$ was directly increased with $\delta_1(c, e(a_{i,T-1})) + \delta_2(rspns(a_{i,T-1}))$. δ_1 and δ_2 model learning gain obtained by taking an action $a_{i,T-1}$. $\delta_1(c, e(a_{i,T-1}))$ is a rectified random variable that follows a normal distribution with mean μ_1 and standard deviations σ_1, i. e., $\max(0, \mathcal{N}(\mu_1\ ,\ \sigma_1^2))$. μ_1 and σ_1 are given a priori based on c and $e(a_{i,T-1})$.

c and $e(a_{i,T-1})$ that represent *contrast* and *effectiveness* respectively were the parameters controlled to create several online learning scenarios for research question RQ1. We controlled the difference in impact on students' learning (i.e., masterly level) between effective and ineffective instructional components using two parameters: (i) c that represents the *contrast* in the increase of logit between effective and ineffective instructional elements—large vs. moderate vs. small, and (ii) $e(a_{i,T-1})$ that represents the *effectiveness* of the instructional element $a_{i,T-1}$—effective vs. ineffective.

The fundamental assumptions were that (1) the larger the contrast, the larger the differences of μ_1 when effective vs. ineffective instructional components were taken, and (2) the larger the contrast, the smaller the σ_1 was. For example, if c = "large", (μ_1, σ_1) was (0.5, 0.01) vs. (-0.1, 0.01) for $e(a_{i,T-1})$ = "effective" vs. "ineffective." However, they were (0.3, 0.1) vs. (0.1, 0.1) if c = "small." Table 1(a) shows μ_1 and σ_1 for different *contrast* and *effectiveness*.

$\delta_2(rspns(a_{i,T-1}))$ is also a rectified random variable that follows a normal distribution with mean μ_2 and standard deviations σ_2. The variable δ_2 was set to be zero if $a_{i,T-1}$ was not a quiz. Otherwise, μ_2 and σ_2 were determined a priori based on a student's response

rate, *rspns*=correct/incorrect. We assume that when a student was able to answer the quiz correctly, logit $Z_{i,T}$ increases more than when the student was not able to answer it. Table 1(b) shows μ_2 and σ_2 for correct and incorrect responses respectively.

Student's initial logit $Z_{i,0}$ also followed a rectified normal distribution with μ_0 and σ_0^2. These were given a priori based on the contrast parameter, *c*, as shown in Table 2.

In addition to three learning scenarios with different *contrasts*, we also created three versions of mock online courseware with different *qualities*. The quality of courseware was operationalized as the ratio of a number of effective to ineffective instructional components in the courseware. Three types of qualities are implemented in this study: High, Medium and Low. The higher the quality, the larger the proportion of effective instructional components. In the simulation study, each page in the mock online courseware included 3 lecture videos, 3 quizzes, and 3 hint messages each associated with a quiz. The low, medium, and high-quality courseware included 80-90%, 50-60%, and 10-20% ineffective instructional components.

Two instances of mock courseware (with a different number of pages) were created for each level of quality. Those six instances of courseware were crossed with three levels of contrast, resulting in 18 different simulated-learning scenarios. In each scenario, simulated students took a total of 10 to 30 instructional components.

Learning trajectories of students were randomly generated as follows. At first, for each simulated student, the number of instructional components to be taken was randomly decided. Either a video or a quiz was then randomly selected as the first learning activity. If it was a quiz, the student might show a hint before trying to answer the quiz at 0.05 probability. When the student answered a quiz, the correctness of the quiz response was determined randomly using the mastery level as the probability distribution. When the response was incorrect, either requesting a hint or retaking the same quiz (as a next instructional component) was randomly determined based on the probability distribution reported in [1]. Let quiz_x be a quiz with an ID x that student answered incorrectly. The probability distribution is as follows: (i) Try quiz_x at 0.78 probability, (ii) show hint_x at 0.20, (iii) give up and move to different quiz or video at 0.02 (these two are randomly selected). The same distribution is applied when the student showed hint_x. This process was repeated for the number of instructional components to be taken. Simulated students were able to retake the same instructional components.

For each of 18 learning scenarios, 100 course offerings were created each with 1,000 simulated students. In other words, this simulation study modeled a large-scale field trial as if 1800 instances of online course offerings were tested each with 1,000 student participants.

For each course-offering simulation, the learning trajectory data were converted into a learning trajectory graph (LTG). As a consequence, 1,800 instances of LTGs were generated. The manipulation of logit described above was used to estimate mastery level in LTG. For each of the 1,800 LTGs, the value iteration technique was applied to compute a converse policy. From each converse policy, the frequency heuristic was applied to generate a recommendation for refinement for a corresponding instance of online courseware.

6. RESULTS

6.1 Overview of the Data

To verify the feasibility of the simulation data, we computed a correlation between the ratio of effective to ineffective instructional components taken by a student and the final mastery level. The data showed a strong positive correlation, $r = 0.70$, $t(1799998) = 1314.56$, $p < 0.001$, suggesting that the final mastery level was significantly higher when simulated students took relatively more effective instructional components than ineffective ones.

6.2 Converse Policy-based Recommendation

6.2.1 Frequency Heuristic

The hypothesis under the frequency heuristic is that relatively ineffective instructional components tend to appear in a policy action set of a given converse policy more frequently than effective ones. To verify this hypothesis, we first compare the normalized frequency of ineffective components in the policy action set with that of effective ones. We also answer RQ1: How robust is the converse policy as a detector for ineffective instructional components against different conditions of learning data?

The frequency of an instructional component ι selected as a converse policy action was normalized as follows. Let π be a converse policy and **S** be the set of states in the LTG. The normalized frequency of an instructional component *ι* is calculated by the following equation.

$$Normalized\ Freqency(\iota) = NF(\iota) = \frac{|\mathbf{S}^{\pi}(\iota)|}{|\mathbf{S}^{\mathcal{A}}(\iota)|}$$

$\mathbf{S}^{\pi}(\iota) = \{s|\pi(s) = \iota\}$: A set of states in the LTG where ι is the converse policy action.
$\mathbf{S}^{\mathcal{A}}(\iota) = \{s|\iota \in \mathcal{A}_s\}$ (where $\mathcal{A}_s$ is a set of actions available from state *s*): A set of states where the instructional component ι was taken.
$|X|$: Number of elements in X
Also notice that $\mathbf{S}^{\pi}(\iota) \subset \mathbf{S}^{\mathcal{A}}(\iota)$.

We then tested if there was a significant difference in the mean normalized frequencies between effective and ineffective instructional components. Table 3 shows the mean normalized frequencies of ineffective and effective instructional components and those standard deviations. The effect size is a ratio of the difference between two means to the standard deviation. Table 3 suggests that regardless of the quality and contrast, ineffective instructional components were selected as a converse policy action notably many times more than effective ones. The differences were all statistically significant using t-test (p<<0.01). The data also suggest that the difference in the frequencies between ineffective and effective components becomes the smallest (as indicated by the smallest effect size) when contrast is small and quality is high, as we expected.

These results support the hypothesis that *relatively ineffective instructional components tend to appear in a converse policy action set more frequently than effective instructional components. It is also shown that the converse policy is robust enough to discriminate the effectiveness of the instructional component regardless of the quality (operationalized as the ratio of effective vs. ineffective components) and the contrasts (operationalized as the difference in the growth of logit between effective and ineffective components).*

Table 3: Comparison of the mean normalized frequency between ineffective (Inef.) and effective (Ef.) instructional components. A number in the parentheses shows an effect size.

	Contrast					
	Large		Moderate		Small	
Quality	Inef.	Ef.	Inef.	Ef.	Inef.	Ef.
High	0.7±0.2	0.2±0.1 (4.0)	0.7+0.1	0.1±0.1 (5.7)	0.5±0.1	0.2±0.1 (3.1)
Med.	0.4±0.1	0.1±0.05 (7.9)	0.4±0.1	0.1±0.04 (8.5)	0.4±0.1	0.2±0.1 (3.6)
Low	0.4±0.1	0.04±0.04 (9.2)	0.4±0.1	0.04±0.03 (10.0)	0.4±0.1	0.1±0.1 (4.5)

6.2.2 Accuracy of recommendation

We next evaluate the precision and recall scores of recommendations created by frequency heuristic to answer RQ2: How accurately does the frequency heuristic compose a recommendation?

To compose a recommendation, we need to define a cut-off value. As a reminder, those instructional components whose normalized frequency is more than a pre-defined cut-off are labeled as "ineffective" and included in the recommendation. What the cut-off value should be is an empirical call.

In the current study, we compared two cut-off values using mean (M) and standard deviation (SD) of the normalized frequency: M+SD vs. M–SD. To evaluate the accuracy of recommendation, we computed Precision and Recall as follows:

$$Precision = \frac{|\Phi^{R}_{\text{ineff}}|}{|\Phi^{R}|}$$

$$Recall = \frac{|\Phi^{R}_{\text{ineff}}|}{|\Phi_{\text{ineff}}|}$$

$$F1 = 2 * \frac{Precision \cdot Recall}{Precision + Recall}$$

$|\Phi^{R}_{\text{ineff}}|$: Number of *ineffective* instructional components included in a recommendation
$|\Phi^{R}|$: Number of total instructional components included in a recommendation
$|\Phi_{\text{ineff}}|$: Number of *ineffective* instructional components in courseware

We investigated how precision and recall scores vary depending on the cut-off and the condition of the learning data (contrast, quality). Figure 2 shows precision and recall scores comparing M-SD and M+SD cut-offs for each quality of the courseware. For each data point, three levels of contrasts are aggregated, because there was no notable difference among them. The figure show that when the quality of courseware is low to medium, the M-SD cut-off had better recall and precision scores than M+SD. F1 score for M-SD was 0.99 and 0.92 for low and medium qualities respectively. On the other hand, when the quality is high, the M+SD cut-off outperformed M-SD. F1 scores of M+SD for high quality courseware was 0.88.

In sum, the frequency heuristic adequately works to determine which instructional components must be taken into a recommendation for courseware refinement. In the current simulation study, over 90% of ineffective instructional components were correctly taken into a recommendation when an appropriate cut-off was used based on the maturity of the courseware. When the courseware is newly built (which is usually in a low to medium quality), the M–SD cut-off should be used, whereas the M+SD cut-off should be used for matured (high-quality) courseware. In the current study, even with the high-quality courseware where only 10-20% of all instructional components in the courseware are ineffective, RAFINE was able to correctly include ineffective components in the recommendation with the M+SD cut-off.

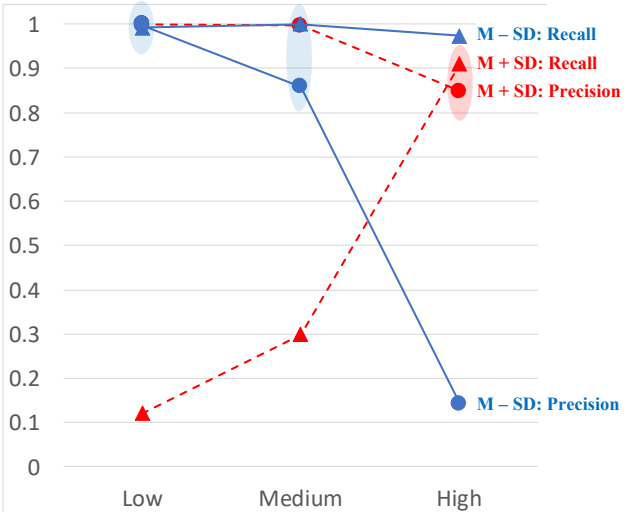

Figure 2 : Precision and recall of a recommendation. The X-axis represents the quality of the courseware. Red dashed lines show results from M+SD and blue solid lines show M–SD.

7. DISCUSSION AND LIMITATIONS

In the evaluation study, we had two research questions. RQ1: How robust is the converse policy as a detector for relatively ineffective instructional components against different conditions of learning data? RQ2: How accurately does the frequency heuristic compose a recommendation?

First, the comparison of the normalized frequency revealed that relatively ineffective instructional components tend to appear in a policy action set significantly more frequently than effective ones regardless of the contrast and the quality of courseware. This suggests that the converse policy as a detector for relatively ineffective instructional components is robust enough against different conditions of learning data (RQ1)

Second, we evaluated the accuracy of the recommendation created by the frequency heuristic to answer RQ2. The results showed that when we use a different cut-off depending on the maturity of courseware, the recommendation created by the frequency heuristic accurately includes ineffective instructional components.

The results from the evaluation study showed that RAFINE can find deficits of the existing courseware by analyzing learning trajectory data on behalf of human experts. Although videos, quizzes and hints are evaluated in the evaluation study, RAFINE could also analyze other types of instructional components such like written paragraphs, tables, figures, etc. However, accurately tracking how students review these instructional components while learning is not straightforward—e.g., the ordinal clickstream data do not convey whether a student was reading a text instruction or not.

One limitation of this study is that there are several assumptions about learning trajectory data. First, we assume the presence of the KC model. Instructional components should be tagged with a KC to apply the RAFINE method. Therefore, the recommendation created by RAFINE changes depending on a KC model. Methods to build a good cognitive model that captures the fine KC model are studied as mentioned in the related works.

Second, we also assume that the students' masterly level is measured correctly. Since the reward function depends on the change of masterly level from the current state to the next state, it is essential that the measured masterly level is not far from the actual level of students' understanding on a skill.

Third, variations in the learning trajectory graph are critical when applying the RAFINE method. To get better performance, RAFINE must be fed a learning trajectory graph that contains diverse learning activities. If there is only one path in a learning trajectory graph, for example, the converse policy has no choice but to select an instructional component that appears in the path as a converse policy action.

One question that is not addressed in the current study is about the students' differences—how much the students' individual differences affect the "effectiveness" of each instructional component. Instructional components that are quite effective for one group of students may not be as effective for another group of students. Although it is out of the scope of the current paper, we have two working hypotheses for future studies. One hypothesis is about the majority rule—the big data overrides the individual human factors and detects the latent trends. Another hypothesis is about the individualized student model—entering individual student factors into the student model used to compute the mastery level, e.g., the individualized additive factor model [24]. Further studies will be necessary to address these issues in detail.

8. CONCLUSION

We found that the RAFINE method could serve as a building block for the evidence-based, human-in-the-loop, iterative online courseware learning engineering method by detecting the deficits of the courseware. RAFINE analyzes learning trajectory data collected from existing online courseware using the reinforcement learning technique and identifies ineffective instructional components. The detected components are provided to courseware developers as a recommendation for refinement. Given the recommendation, courseware developers can efficiently improve the courseware by modifying the listed instructional components.

In addition to providing a new evidence-based learning engineering method, we also proposed a technique called the *frequency heuristic* and contributed to the community of applications of reinforcement learning (RL). The frequency heuristic is a novel way of interpreting the policy for evaluating the actions in MDP. It operates differently from the conventional applications of RL in which the policy is used for optimization. In RAFINE, the frequency heuristic is applied to the converse policy to detect ineffective instructional components (i.e., action) that had relatively less contribution to learning. In the evaluation study, we demonstrated that the frequency heuristic over the converse policy is potentially a powerful analytic tool to detect a relative weakness among available actions.

For future studies, it is crucial to measure the actual effectiveness of the proposed method in authentic learning settings and apply the method to real students' learning data.

9. ACKNOWLEDGEMENTS

This study was partially supported by National Science Foundation grant Award No. 1623702.

10. REFERENCES

[1] Aleven, V., & Koedinger, K. R. (2000). Limitations of student control: Do students know when they need help? In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *Proceedings of 5th International Conference on Intelligent Tutoring Systems* (pp. 292-303): Springer Verlag.

[2] Bodily, R., Nyland, R., & Wiley, D. (2017). The RISE Framework: Using Learning Analytics to Automatically Identify Open Educational Resources for Continuous Improvement. . *The International Review of Research in Open and Distributed Learning, 18*(2).

[3] Cen, H., Koedinger, R. K., & Junker, B. W. (2006). Learning Factors Analysis - A General Method for Cognitive Model Evaluation and Improvement. In *International Conference on Intelligent Tutoring Systems* (pp. 12).

[4] Chi, M., VanLehn, K., Litman, D., & Jordan, P. (2011). An evaluation of pedagogical tutorial tactics for a natural language tutoring system: A reinforcement learning approach. *International Journal of Artificial Intelligence in Education, 21*(1-2), 83-113.

[5] Clark, R., & Mayer, R. E. (2003). *e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning*. San Francisco, CA: John Wiley & Sons.

[6] Dede, C., Richards, J., & Saxberg, B. (2018). *Learning Engineering for Online Education: Theoretical Contexts and Design-based Examples*: Routledge.

[7] Du, X., & Cardie, C. (2017). Identifying where to focus in reading comprehension for neural question generation. *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2067-2073.

[8] Fishman, B., Marx, R. W., Blumenfeld, P., Krajcik, J., & Soloway, E. (2004). Creating a Framework for Research on Systemic Technology Innovations. *The Journal of the Learning Sciences, 13*(1), 43-76. doi:10.2307/1466932

[9] González-Brenes, J. P., & Mostow, J. (2012). Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models. *International Educational Data Mining Society*.

[10] Guàrdia, L., Maina, M., & Sangrà, A. (2013). MOOC design principles: A pedagogical approach from the learner's perspective. *elearning papers*(33).

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

[12] Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The Knowledge-Learning-Instruction Framework: Bridging the Science-Practice Chasm to Enhance Robust Student Learning. *Cognitive Science, 36*, 757-798. doi:10.1111/j.1551-6709.2012.01245.x

[13] Lindsey, R. V., Khajah, M., & Mozer, M. C. (2014). Automatic discovery of cognitive skills to improve the

prediction of student learning. *In Advances in neural information processing systems*, 1386-1394.

[14] Martin, B., Mitrovic, A., Koedinger, K. R., & Mathan, S. (2011). Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Interaction, 21*(3), 249-283. doi:10.1007/s11257-010-9084-2

[15] Mazidi, K., & Tarau, P. (2016). Automatic question generation: from NLU to NLG. *International Conference on Intelligent Tutoring Systems*, pp.23-33.

[16] Milligan, S. K., & Griffin, P. (2016). Understanding learning and learning design in MOOCs: A measurement-based interpretation. *Journal of Learning Analytics, 3*(2), 88-115.

[17] Rafferty, A. N., Brunskill, E., Griffiths, T. L., & Shafto, P. (2015). Faster Teaching via POMDP Planning. *Cogn Sci, 40*(6), 1290-1332. doi:10.1111/cogs.12290

[18] Shapiro, H. B., Lee, C. H., Wyman Roth, N. E., Li, K., Çetinkaya-Rundel, M., & Canelas, D. A. (2017). Understanding the massive open online course (MOOC) student experience: An examination of attitudes, motivations, and barriers. *Computers & Education, 110*, 35-50. doi:http://dx.doi.org/10.1016/j.compedu.2017.03.003

[19] Shen, S., & Chi, M. (2016). Reinforcement Learning: the Sooner the Better, or the Later the Better? *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization. ACM*, pp.37-44.

[20] Slavich, G., & Zimbardo, P. (2012). Transformational Teaching: Theoretical Underpinnings, Basic Principles, and Core Methods. *Educational Psychology Review, 24*(4), 569-608. doi:10.1007/s10648-012-9199-6

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

[22] Wang, P., Rowe, J. P., Min, W., Mott, B. W., & Lester, J. C. (2017). *Interactive Narrative Personalization with Deep Reinforcement Learning.* Paper presented at the IJCAI.

[23] Wiering, M., & van Otterlo, M. (Eds.). (2012). *Reinforcement Learning*. Heidelbereg, Berlin: Springer.

[24] Yudelson, M., Koedinger, K., & Gordon, G. (2013). Individualized Bayesian Knowledge Tracing Models. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Artificial Intelligence in Education* (Vol. 7926, pp. 171-180): Springer Berlin Heidelberg.

[25] Zhang, Y., Shah, R., & Chi, M. (2016). Deep Learning+ Student Modeling+ Clustering: A Recipe for Effective Automatic Short Answer Grading. *International Educational Data Mining Society*.