Application Of Reinforcement Learning For Automated Contents Validation Towards Self-Improving Online Courseware

Noboru Matsuda and Machi Shimmei

Center for Educational Informatics Department of Computer Science

North Carolina State University

INTRODUCTION

Online education has been growing rapidly for the last decade with exponential growth of diverse students population (Shapiro et al., 2017) and adaptive technology enhancements (Lerís, Sein-Echaluce, Hernández, & Bueno, 2017). However, building practical online courseware is extremely costly—it requires extensive knowledge and expertise in theories of learning and teaching (Clark & Mayer, 2003; Slavich & Zimbardo, 2012). Most of the time, instructional designers and instructors design an initial courseware from their honest intuition, and then the courseware will be iteratively modified to meet better learning outcome. Though, iterative software engineering is a norm for almost any sort of practical software applications (Fishman, Marx, Blumenfeld, Krajcik, & Soloway, 2004), it requires significant knowledge to identify issues to be fixed for improvement.

It is therefore critical to develop a transformative theory of practical learning-engineering methods for iterative online courseware creation. Without such methods, it is not likely to have sustainable system of online education. What if the courseware improves itself over time?

The larger goal of our current project is to develop a self-improving online courseware that automatically detects and fixes ineffective parts of the existing courseware relative to students’ learning achievement. As a step towards achieving this pivotal goal, we propose to develop an integrated development environ- ment (IDE) where human and AI collaboratively build online courseware through iterative design engineering—a machine detects issues and a human fixes them. As a step towards the proposed human- AI collaboration, this paper describes an innovative application of a reinforcement learning technique called RAFINE (**R**einforcement learning **A**pplication **F**or **IN**cremental courseware **E**ngineering). The RAFINE aims to identify ineffective instructional elements on existing online courseware given a record of individual students’ learning activity logs.

In the rest of the paper, we first discuss related works followed by a detailed description of RAFINE. We then describe details about a simulation study and results as a proof of concept.

RELATED WORKS

Reinforcement learning (RL) has been used for educational applications in particular to compute effective pedagogical strategies for adaptive tutoring. Previous works applied RL to find optimal pedagogical decisions such as teaching actions (Rafferty, Brunskill, Griffiths, & Shafto, 2015), hint messages (Martin & Arroyo, 2004), dialogue moves (Min Chi, VanLehn, Litman, & Jordan, 2011; Tetreault, Bohus, & Litman, 2007), learning activities (Shen & Chi, 2016), and navigation (Iglesias, Martinez, Aler, & Fernandez, 2009). Other studies have applied RL to compute effective domain models such as model solutions (Barnes & Stamper, 2008). The effects of educational RL policy have been tested both with real and simulated data where some studies showed a positive effect of the policy (Beck, Woolf, & Beal, 2000; M. Chi, Koedinger, Gordon, Jordan, & VanLehn, 2011) while others did not (Iglesias et al., 2009).

It is fairly common that the computed policies in the previous educational applications were optimized for learning outcome and learning time (Beck et al., 2000). To the best of our knowledge, the previous works are all mostly about computing the optimal pedagogical decisions. No research has been conducted to apply RL to identify ineffective instructional elements. Furthermore, under the framework of the ordinal RL, the rejected instructional contents do not necessarily have a flaw—the second best might be as effective as the best. The current paper demonstrates *how RL can be applied to identify ineffective instructional contents on existing online courseware*.

SOLUTION: RAFINE

Overview of the RAFINE method

We consider the RAFINE method as Human-AI collaboration to improve the quality of existing online courseware. An initial version of the online courseware will be used by students and their activities will be logged. These activity data consist of standard clickstream data and students’ responses (and their correctness) for formative assessments. Since students’ activity data show a chronological record of their behavior on the online courseware, we call them the *learning trajectory* data hereafter.

The RAFINE method first consolidates learning trajectories collected from *all* students into a single state transition graph, called a *learning trajectory graph* (LTG), and annotats the states with predefined rewards. A value iteration technique is then applied to compute a *converse policy* that shows the worst activities to be taken to achieve the expected learning outcomes. As a consequence, the converse policy corresponds to a set of instructional elements that have the least likelihood to contribute to students’ learning.

Our central hypothesis is that those instructional elements that *frequently* appear as a converse policy across different states in a given LTG are likely to be ineffective and hence the subject for refinement. Those instructional elements identified as ineffective will then be presented to courseware developers as a recommendation for a courseware modification. The RAFINE method will be iteratively applied to the revised courseware by collecting a new batch of learning trajectory data to further improve the courseware.

Model representation

The unit of analysis of the RAFINE method is an *instructional element* that constitutes online courseware. In the current study, we deal with three types of instructional elements: (1) videos, (2) formative assess- ments (aka quizzes), and (3) hint messages associated with formative assessments. We assume that all assessment quizzes are equipped with hint messages.

Let  be a set of instructional elements appearing in the given learning trajectories. We assume that the target courseware was used by a large number of students hence  contains all instructional elements on the target online courseware. Let *aiT*, a *learning activity*, be an instructional element taken (e.g., watching a video or answering a quiz) by student *i* at time *T*. Let *LTi* be a *learning trajectory* for student *i* who has *ni* learning activities. *LTi* is a chronological record of learning activities:

𝐿𝑇# = {𝑎#(, … , 𝑎#+, | 𝑎#. ∈ , 𝑘 = 1, … , 𝑛#}.

We assume a presence of a *skill model* that contains a set of skills each representing a unit of knowledge that students have to learn, aka knowledge components (Koedinger, Corbett, & Perfetti, 2012). This assumption implies that each instructional element is tagged with a single skill in the given skill model. The RAFINE method is applied to each individual skill separately. Let 𝜇 be a set of instructional elements for skill 𝜇. Learning trajectories are also broken down into individual skills. Let *LTi* 𝜇 be the learning trajectory that contains only learning activities about skill 𝜇. The RAFINE method must be applied to each bundle of 𝜇 and *LTi* 𝜇 for all 𝜇 separately. A single application of the RAFINE method identifies ineffec- tive instructional elements relative to a particular skill.

For a sake of simplicity without a loss of generality, let’s assume that there is only one skill in our target online courseware. We therefore eliminate the skill index from  and *LT* in the following descriptions unless otherwise desired for a clarification.

In the learning trajectory graph, states represent *learning status* and edges represent learning activities taken that caused a change in status. We define a *learning status* for student *i* at time *T* for a particular skill 𝜇 as an intermediate state of learning represented as a pair of Action History and Mastery Level;

<***ahi,T***, p*i,T*(𝜇)>. Action History ***ahi,T*** is a binary vector <*ahi*1, …, *ahiK*> where *ahim* shows whether student *i* has taken the *m*-th instructional element in 𝜇 by time *T* (assuming the instructional elements are ordered and |𝜇 | = K).

Mastery Level p*i,T*(𝜇) is a scalar value showing a predicted probability of student *i* applying skill 𝜇 correctly, should he/she answer an assessment quiz for the skill 𝜇 at time *T*. The value of Mastery Level is rounded down to the nearest multiple of 0.05 (e.g., 0.18 becomes 0.15). Mastery Level, p*i,T*(𝜇), will be computed based on the history of learning activities with an underlying assumption that commitment to a learning activity for a particular skill would increase Mastery Level by a specific amount. There are several known techniques available to achieve this goal including Bayesian models (e.g., Corbett & Anderson, 1995) and regression models (e.g., M. Chi et al., 2011). As long as Mastery Level is monoton- ically updated, any student-modeling technique would work for the RAFINE method.

To consolidate individual students’ learning trajectories into a single learning trajectory graph (LTG), each individual student’s learning trajectories are first converted into a *learning trajectory path*. This is done by chronologically traversing a learning trajectory while creating states each representing an intermediate learning status <***ahi,T***, p*i,T*(𝜇)>. While traversing the learning trajectory, ***ahi,T*** and p*i,T*(𝜇) are updated accordingly. For example, assume there are six instructional elements: Video1, Video2, Quiz1, Quiz2, Hint1, and Hint2. A state *s* <101000, 0.40> indicates that a student had watched Video1 and took Quiz1 before reaching the state *s*. It also indicates that a predicted Mastery Level 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.

All individual students’ learning transition paths are then aggregated into an LTG by merging the same states. As a consequence, the states in the LTG generally have multiple incoming and outgoing edges. Note that in the 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 <***ah***, p(𝜇)>. In an LTG, the states where the value of the Mastery Level, p(𝜇), is greater than a pre-defined threshold (which is usually 0.85) are called *terminal states*—meaning that students became proficient in applying skill 𝜇. All outgoing edges at terminal states are discarded.

Rewards

A reward value of a particular state depends on the Mastery Level, p(𝜇), 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(𝜇) was not increased) whereas the other student watched a video (i.e., p(𝜇) 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:

−0.14 (𝑚𝑙(𝑠) = 𝑚𝑙(𝑠A) < 0.85)

𝑅(𝑠, 𝑎, 𝑠′) = : −0.05 (𝑚𝑙(𝑠) < 𝑚𝑙(𝑠′) < 0.85)

0.95 F0.85 ≤ 𝑚𝑙(𝑠A)H

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 so that the RL would find the shortest path to a terminal state 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.

Converse Policy

Given the reward function *R* as mentioned above, a value function for state *s* under a policy 𝜋 is defined as follows, where S is a set of all states in a given LTG:

𝑉K(𝑠) = L 𝑇(𝑠, 𝜋(𝑠), 𝑠′)(R(s, 𝜋(𝑠), sA) + γ𝑉K(𝑠′))

PA∈Q

In the current implementation, the discount factor  is arbitrarily set to be 0.9. A transition model *T*(*s*, *a*, *s*’) is derived from the learning trajectory data collected from 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 (Wiering & van Otterlo, 2012). However, for the purpose of Rafine, we need to know which instructional elements should not be taken—i.e., we need to know which action has the least expected reward. There- fore, through the value iteration, the value function is updated as follows where *A*(*s*) shows a set of actions appearing in outgoing edges at state *s*:

𝑉(𝑠) ← min L 𝑇(𝑠, 𝑎, 𝑠′)(𝑅(𝑠, 𝑎, 𝑠A) + 𝛾𝑉(𝑠′))

V∈W(P)

PA∈Q

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*:

𝜋(𝑠) = argmin L 𝑇(𝑠, 𝑎, 𝑠′)(𝑅(𝑠, 𝑎, 𝑠A) + 𝛾𝑉K(𝑠A))

V∈W(P)

P\∈Q

EVALUATION STUDY

Our central hypothesis is that those instructional elements that *frequently* appear as a converse policy across different states in a given LTG are likely to be ineffective and hence should be revised. To test this hypothesis, we conducted an evaluation study with hypothetical learning trajectories generated by simulated students.

Although any instructional element can be selected as a converse policy, the current version of RAFINE only includes videos and hints in its recommendation. This is because there are known quantitative methods, e.g. item response theory (Baker, 2001), that can be used to evaluate the quality of assessment items.

Three instances of online courseware were created to control the quality of courseware with varying ratios of a number of effective instructional elements to all instructional elements on the courseware. We assumed that there was only one skill involved in the mock online courseware. All three instances of courseware had the same structure: they consisted of three pages (Page0, 1, 2), and each page included three lecture videos and three formative assessments (i.e., quizzes). All quizzes had hints associated. All instructional elements on the mock courseware (9 videos and 9 hints total) except assessment quizzes (for the reason mentioned above) were coded as either effective or ineffective. The high-quality courseware had a 8:1 split (8 effective video / hint and 1 ineffective video / hint); the moderate-quality courseware had a 4:5 split; and the low-quality courseware had a 1:8 split. Let’s call them H (High), M (Moderate), and L (Low) courseware hereafter.

Simulated students started from Page0 and randomly took a total of 10 to 14 instructional elements. At least two instructional elements must be taken to proceed a page. When simulated students answered a quiz incorrectly, they were forced to review the associated hint and take the same quiz again. The simu- lated student’s performance on the assessment quizzes was determined by their latent proficiency that indicates a probability of answering a quiz correctly. In the real world, the latent proficiency increases according to the actual learning activities taken and student’s latent trait of learning that determines the learning rate. To simulate the growth in the latent proficiency *pi,T*(𝜇), we used a logistic regression model representing a probability of student *i* answering a quiz about the skill 𝜇 correctly at time *T* as shown below:

𝑝#,^(𝜇) = \_ 1 d

1 + 𝑒ab,,c

𝑍#,^ = 𝑍#,^a( + 𝛿F𝑎#,^a(H

The ⌈𝑥⌉ operator is to round down the value *x* to the nearest multiple of 0.05. Logit (*Zi,T*) was directly increased with an ad-hoc function (*ai,T-1*) that models the growth of the latent proficiency when the learning activity *ai,T-1* was taken by simulated student *i* at time T-1. The function  was defined by the learning rate, the effectiveness of the instructional element taken, and (when the learning activity was an assessment quiz) the correctness of a quiz answer.

We assumed that simulated students’ learning was facilitated more (i.e., a greater increase in logit) when they took effective instructional elements than ineffective elements. We also assumed that students learned more by answering a quiz correctly than incorrectly. For example, when a simulated student with a high learning rate watched an effective video, the logit was increased by 0.35, but only by 0.15 when an ineffective video was watched. For a simulated student with a low learning rate, the logit was increased by 0.31 and 0.11 respectively for effective and ineffective videos.

To control learning rate, five types of simulated students were created with different learning rates. They were labeled from R1 (the highest learning rate) to R5 (the lowest). In the simulation, 20% of simulated students were R1, 30% R2, 20% R3, 20% R4, and 10% R5—roughly reflecting a slightly skewed student population.

Under these assumptions, simulated students’ learning trajectories were randomly generated. For each quality of courseware (H, M, and L), 100 instances of mock courseware were created with 1,000 simulat- ed students. Each of the learning trajectory datasets was then converted into a learning trajectory graph (LTG). As a result, 300 LTG’s were created, 100 each for H, M, and L courseware. In an LTG, Action History was encoded as a 27-bit binary vector (3 types of instructional elements, 9 each); and the Mastery Level is a decimal number (a multiple of 0.05). The latent proficiency described above was used as an estimate for Mastery Level (instead of actually applying a student model technique).

For each of the 300 LTG’s, the value iteration technique was applied to compute a converse policy. As a result, 300 sets of converse policy were created, each suggesting which instructional elements were ineffective on the corresponding online courseware. Note that this simulation study models a large scale field trial with real students as if 300 instances of online courseware were tested each with 1,000 students participating. After these trials, the Rafine makes a recommendation for refinement for each instance of the courseware..

RESULTS

For the following analysis, we first evaluate the accuracy of a converse policy. We then discuss the accuracy of recommendation, which by definition is a subset of all instructional elements on the given online courseware that Rafine identifies as ineffective.

Overall Accuracy of the Converse Policy

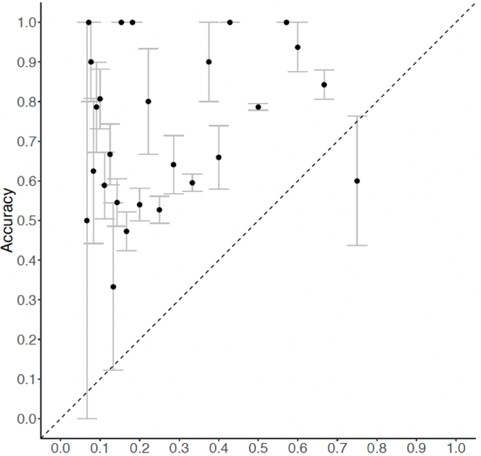
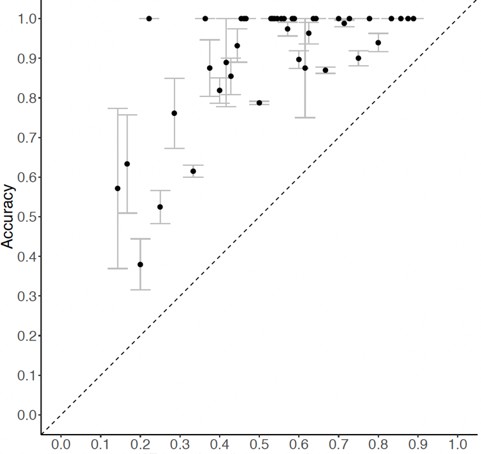
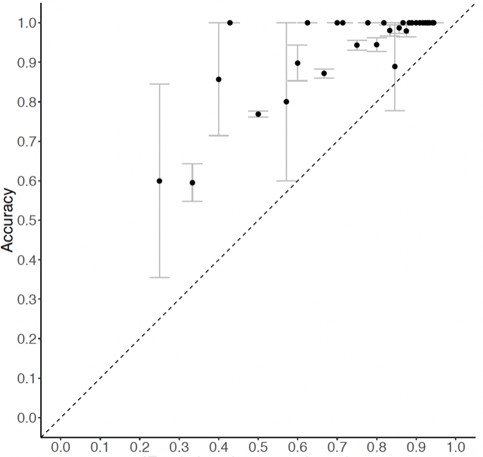
We first evaluate the overall accuracy of a converse policy as a predictor of ineffective instructional elements. Notice that if there are *N* states in a learning trajectory graph (LTG), there are *N* converse policies generated. The accuracy of the converse policy is a ratio *n*/*N* where *n* is the number of states on which an ineffective instructional element is suggested as a converse policy.

To understand the value added by the converse policy, the chance ratio of *courseware* is defined as a ratio of ineffective to a total number of instructional elements on each courseware—e.g., for L (low) courseware, it is 16/18 = 0.89. The chance ratio of *state* is also defined among instructional elements appearing on outgoing edges of a given state as *a*/*b* where *a* is the unique number of ineffective instruc- tional elements and *b* is the total number of unique instructional elements. In the following analysis, states where the chance ratio is equal to 1.0 or 0.0 were excluded (i.e., instructional elements on the outgoing edges were all ineffective or all effective).

Table 1: Overall accuracy of the converse policy averaged across 100 datasets for each type of courseware.

|  |  |  |  |
| --- | --- | --- | --- |
| Quality | L | M | H |
| Chance Ratio | 0.89 | 0.56 | 0.11 |
| Accuracy | 0.83 | 0.79 | 0.72 |

Table 1 shows the mean accuracy of a converse policy aggregated across 100 datasets for each quality of courseware. The overall accuracy of a converse policy was 0.72 even for the courseware H where only 11% (2 out of 18) of instructional elements were ineffective. *These results imply that the converse policy has a high potential to accurately detect ineffective instructional element.*



Courseware L

Courseware M

Courseware H

Figure 1: The accuracy of converse policy relative to states with the same chance ratio.

We hypothesize that the accuracy of a converse policy is correlated with a chance ratio of state—i.e., if a state has many outgoing edges that correspond to ineffective instructional elements, the value iteration would likely pick one of them as a converse policy. To test this hypothesis, we plotted an accuracy of a converse policy relative to a group of states with the same chance ratio as shown in Figure 1. In the figure, each data point represents a set of states that have the same chance ratio as indicated on the x-axis. The y-axis shows the mean accuracy of a converse policy for a corresponding group of states—i.e., the ratio of states where an ineffective instructional element was selected as the converse policy to the total number of states in the group. The 45-degree line shows the chance rate. In the figure, states where the chance ratio is equal to 1.0 or 0.0 were excluded. Figure 1 indicates that the *converse policy can discrimi- nate ineffective instructional elements from effective instructional elements far better than chance for any state in a given LTG*.

Although the converse policy can detect an ineffective instructional element at each state with a high accuracy, there are normally a notably large number of states in an LTG, so all instructional elements are included in the converse policy. Therefore, filtering the converse policy is essential for Rafine to make an actual recommendation. As our central hypothesis states, we conjecture that the frequency of being selected as a converse policy is a key for the filtering. The next section shows the accuracy of the judge- ment of recommendations for which instructional elements must be replaced based on the frequency heuristic.

Accuracy of Recommendations for Iterative System Improvement

We first tested if the frequency of being selected as a converse policy can be used as a filtering criterion to detect ineffective instructional elements among the converse policy. The average frequency of each instructional element being selected as a converse policy was computed by aggregating frequency values across 100 datasets. On average, each *ineffective* instructional element was selected as a converse policy 28.2 times in L, 30.6 in M, and 33.0 in H per dataset whereas each *effective* instructional element was selected 8.6 times in L, 10.0 in M, and 11.5 in H. The difference between ineffective and effective instructional elements was statistically significant for all three qualities of courseware: for L, *t*(99) = 84.67, *p* < 0.05; for M, *t*(99) = 98.18, *p* < 0.05; for H, *t*(99) = 37.71, *p* < 0.05. *These results suggest that frequency can be used as a filter to indicate ineffective instructional elements among a converse policy*.

The above observation implies that we should be able to find a frequency cut-off to determine which instructional elements must be classified as ineffective. We shall call this heuristic as the *frequency heuristic*. The question is how the cut-off should be determined, but it is rather an empirical call. We therefore compared two different cut-off thresholds—mean ± standard deviation (M±SD). The mean and the standard deviation of the frequency that individual instructional elements were selected as a converse policy were computed. Those instructional elements that appeared as a converse policy more than the cut- off are considered as ineffective. Further analysis revealed that that when the quality of courseware is low (L) to moderate (M), the M–SD cut-off yields better recall and precision than the M+SD cut-off; F1=2\*precision\*recall / (precision + recall) = 0.96 and 0.75 for L and M respectively with M–SD, whereas F1 = 0.38 and 0.58 with M+SD. However, when the quality of courseware is high (H), the M+SD cut-off outperforms M–SD; F1 = 0.65 for M+SD vs. 0.20 for M–SD. This implies that *at the beginning of the iterative courseware engineering, the M–SD cut-off is better, but as the courseware gets improved, the M+SD cut-off should be used*. We would want to detect as many inefficient instructional elements as possible even at a cost of false positives (i.e., the machine suggests refining even effective instructional elements).

CONCLUSIOnS AND RECOMMENDATIONS FOR FUTURE RESEARCH

We found that when students’ learning trajectories were converted into a learning trajectory graph, computing the worst policy (i.e., the converse policy) using the value iteration, a well-known reinforce- ment learning technique, provides us with a strong clue for the effectiveness of instructional elements used in the online courseware. The converse policy is a collection of a state-action pair showing the worst action (i.e., the least effective instructional element) to be taken at a certain state. Since the number of states in a given learning trajectory is very large, all instructional elements appear in the converse policy. The frequency heuristic then differentiates those that are highly likely ineffective instructional elements from others. The proposed method, RAFINE, provides online courseware developers with an evidence- based recommendation to iteratively improve the courseware content.

The current work is a step toward realizing a fully-autonomous, self-improving online courseware— machine identifies issues and human fixes them. As for the current state of the art, we recommend GIFT to provide us API for RAFINE to give feedback to courseware developers on the quality of the individual instructional element. One idea is to flag instructional elements that are identified to be ineffective on the authoring tool GUI while the developer is editing the content. Another idea is to provide a courseware developer’s dashboard that shows a birdview of courseware elements with an annotation for their predict- ed effective.

To yield a better prediction, RAFINE must be fed a learning trajectory graph that contains diverse learning activities. The GIFT online courseware therefore should provide students with a decent flexibility on selecting learning activities on their own.

REFERENCES

Note: Please use reference style provided… it will automatically place a 6pt space between each reference.

Baker, F. (2001). *The Basics of Item Response Theory*. College Park, MD: ERIC Clearinghouse on Assessment and Evaluation.

Barnes, T., & Stamper, J. C. (2008). Toward automatic hint generation for logic proof tutoring using historical student data. In B. P. Woolf, E. Aïmeur, R. Nkambou & S. Lajoie (Eds.), *Proceedings of International Conference on Intelligent Tutoring Systems* (pp. 373-382): Springer.

Beck, J. E., Woolf, B. P., & Beal, C. R. (2000). ADVISOR: A machine learning architecture for intelligent tutor construction *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (pp. 552-557).

Chi, M., Koedinger, K. R., Gordon, G., Jordan, P., & VanLehn, K. (2011). Instructional factors analysis: A cognitive model for multiple instructional interventions. In J. Stamper, Z. Pardos, M. Mavrikis & B. M. McLaren (Eds.), *Proceedings of the 4th International Conference on Educational Data Mining* (pp. 61-70).

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.

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.

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge.

*User Modeling and User Adapted Interaction, 4*(4), 253-278.

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

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

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

Lerís, D., Sein-Echaluce, M. L., Hernández, M., & Bueno, C. (2017). Validation of indicators for implementing an adaptive platform for MOOCs. *Computers in Human Behavior, 72*, 783-795. doi: 10.1016/j.chb.2016.07.054

Martin, K. N., & Arroyo, I. (2004). Agentx: Using reinforcement learning to improve the effectiveness of intelligent tutoring systems. In J. C. Lester, R. M. Vicari & F. Paraguaçu (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 564-572).

Rafferty, A. N., Brunskill, E., Griffiths, T. L., & Shafto, P. (2015). Faster Teaching via POMDP Planning. *Cognitive Science*, 1-43.

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>

Shen, S., & Chi, M. (2016). Reinforcement Learning: the Sooner the Better, or the Later the Better? *Proceedings of the International Conference on User Modeling and Adaptive Personalization* (pp. 13-17).

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

Tetreault, J. R., Bohus, D., & Litman, D. J. (2007). Estimating the reliability of mdp policies: a confidence interval approach *Proc. of HLT-NAACL* (pp. 276-283).

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

ABOUT THE AUTHORS

**Dr. Noboru Matsuda** is an Associate Professor in the Department of Computer Science at North Carolina State Univeristy, and the director of the Innovative Educational Computing Laboratory. He leads the NSF funded project on the data-driven learning engineering methods to build adaptive online courseware where the research team investigates scalable AI techniques towards evidence-based learning engineering to build adaptive online courseware.

**Machi Shimmei** is a PhD student in the Department of Computer Science at North Carolina State University. Her research forcus is on the innovation and application of cutting-edge technologies to facilitate research on education and the science of learning.