Inducing Effective Pedagogical Strategies Using Learning Context Features

Min Chi[1], Kurt VanLehn[2], Diane Litman[3] and Pamela Jordan[4]

[1] Machine Learning Department, Carnegie Mellon University, PA, 15213 USA
minchi@cs.cmu.edu
[2] School of Computing and Informatics, Arizona State University, AZ, 85287 USA
Kurt.Vanlehn@asu.edu
[3] Department of Computer Science, University of Pittsburgh, PA, 15260 USA
litman@cs.pitt.edu
[4] Department of Biomedical Informatics, University of Pittsburgh, Pittsburgh, PA
15260 pjordan@pitt.edu

Abstract. Effective pedagogical strategies are important for e-learning environments. While it is assumed that an effective learning environment should craft and adapt its actions to the user's needs, it is often not clear how to do so. In this paper, we used a Natural Language Tutoring System named Cordillera and applied Reinforcement Learning (RL) to induce pedagogical strategies directly from pre-existing human user interaction corpora. 50 features were explored to model the learning context. Of these features, domain-oriented and system performance features were the most influential while user performance and background features were rarely selected. The induced pedagogical strategies were then evaluated on real users and results were compared with pre-existing human user interaction corpora. Overall, our results show that RL is a feasible approach to induce effective, adaptive pedagogical strategies by using a relatively small training corpus. Moreover, we believe that our approach can be used to develop other adaptive and personalized learning environments.

1 Introduction

Natural Language (NL) Tutoring Systems are a form of Intelligent Tutoring Systems (ITSs) that use natural dialogue for instructional purposes such as helping students to learn a subject by engaging in a natural language conversation. Why2-Atlas and Why2-AutoTutor [1], for example, are NL tutoring systems that teach students conceptual physics. One central component of NL Tutoring Systems is the dialogue manager, which uses *dialogue strategies* to decide what action to take at each point during the tutorial dialogue. For tutoring systems, dialogue strategies are also referred to as pedagogical strategies.

It is commonly believed that an effective tutoring system would craft and adapt its actions to the students' needs based upon their current knowledge level, general aptitude, and other salient features [2]. However, most pedagogical strategies for ITSs are encoded as hand-coded rules that seek to implement cognitive and/or pedagogical theories. Typically, the theories are considerably more general than the specific decisions that designers must make, which makes it difficult to tell if a specific pedagogical strategy is consistent with the theory. Moreover, it is often not easy to empirically evaluate these decisions because the overall effectiveness of the system depends on many factors, such as the usability of the system, how easily the dialogues are understood, and so on. Ideally, several versions of a system are created, each employing a different pedagogical strategy. Data is then collected with human subjects interacting with these different versions of the system and the results are compared. Due to the high cost of experiments, only a handful of strategies are typically explored. Yet, many such other reasonable ones are still possible.

In recent years, work on the design of NL non-tutoring Dialogue Systems has involved an increasing number of data-driven methodologies. Among these, Reinforcement Learning (RL) has been widely applied [3]. RL is a machine learning method that centers on the maximization of expected rewards. It has many features well-suited to the problem of designing the dialogue manager such as unobservable states, delayed rewards, and so on. Its primary advantage is its ability to compute an optimal policy within a much larger search space, using a relatively small training corpus. In this work, rather than implementing pedagogical strategies drawn from human experts or theories, we applied RL to derive pedagogical strategies using pre-existing interactivity data.

While most previous work on using RL to train non-tutoring dialogue systems has been successful [3], whether it can be used to improve the effectiveness of NL tutoring systems is still an open question. One major source of uncertainty comes from the fact that the rewards used in RL are much more delayed in NL tutoring systems than those in non-tutoring dialogue systems. Much of this work in NL non-tutoring Dialogue Systems is focused on systems that obtain information or search databases such as querying bus schedules [4]. For example, in non-tutoring Systems like the train scheduler, the interaction time is often less than 20 minutes, and the number of interactions within user-dialogue systems is generally less than 20 turns [3]. In the training corpora reported here, the time is roughly 4-9 hours and the number of interactions is about 280 turns. More immediate rewards are more effective than more delayed rewards for RL induction. This is because the issue of assigning credit for a decision, attributing responsibility to the relevant decision is substantially easier in the former case. The more we delay rewards, the more difficult it becomes to identify the decision(s) responsible for our success or failure. Additionally, to train an RL model, a large amount of data is generally needed. In this work, we use human data only instead of data from simulators as in applying RL in non-tutoring dialogue systems. This is because the cause of human learning is still an open question and thus it would be difficult to accurately simulate students' responses to the tutor and simulate how students would learn. Given the high cost of collecting human data, we were more likely to encounter the issue of data sparsity.

For RL, as with all machine learning tasks, success is dependent upon an effective state representation or state model. An effective state representation should be an accurate and compact model of the learning context. Compared with non-tutoring Dialogue Systems, where success is primarily a function of communication efficiency, communication efficiency is only one of the factors determining whether a student learns well from an NL tutoring system. Moreover, the other factors are not well understood, so to be conservative, states need to contain features for anything that is likely to affect learning. Hence, state models for RL applications to tutoring systems tend to be much larger than state models for non-tutoring applications. Unfortunately, as states increase in size and complexity, we risk making the learning problem intractable or the decision space too large to sample effectively. In order to obtain an effective state model that both minimizes state size while retaining sufficient relevant information about the learning context, we began with a large set of features to which we applied a series of feature-selection methods in order to reduce them to a tractable subset. Before describing our approach in detail, we will briefly describe the two types of tutorial decisions covered by the induced pedagogical policies.

2 Two Types of Tutorial Decisions

Among the many tutorial decisions that must be made, we focus on two types of decisions, Elicit/Tell (ET) and Justify/Skip-Justify (JS). The ET decision asks "should the tutor *elicit* the next problem-solving step from the student, or should he or she *tell* the student the next step directly?". For example, when the next step is to select a principle to apply and the target principle is the "definition of Kinetic Energy', the tutor can choose to elicit this from the student by asking the question, "Which principle will help you calculate the rock's kinetic energy at T0?" By contrast, the tutor can elect to tell the student the step by stating, "To calculate the rock's kinetic energy at T0, let's apply the definition of Kinetic Energy." The JS decision asks "should the tutor include a *justify* for a step just taken or or not". For example, after deciding to use the "definition of Kinetic Energy", the tutor can choose to ask the student why the principle is applicable or to skip to ask. There is no widespread consensus on how or when any of these actions should be taken [5–8]. This is why our research objective is to derive policies for them from empirical data.

3 Applying RL to Induce Pedagogical Strategies

Previous research on using RL to improve non-tutoring dialogue systems (e.g. [9]) has typically used Markov Decision Processes (MDPs) [10] to model dialogue data. The central idea behind this approach is to transform the problem of inducing effective pedagogical strategies into computing an optimal policy for an agent that is choosing actions in an MDP. An MDP formally corresponds to a 4-tuple (S, A, T, R), in which: $S = \{S_1, \cdots, S_n\}$ is a state space; $A = \{A_1, \cdots, A_m\}$ is an action space represented by a set of action variables; $T : S \times A \times S \rightarrow$ *[0, 1]* is a set of transition probabilities $P(S_j|S_i, A_k)$, which is the probability that the model would transition from state S_i to state S_j after the agent takes action A_k; $R : S \times A \times S \rightarrow R$ assigns rewards to state transitions. Finally, $\pi : S \rightarrow A$ is defined as a policy, which determines which action the agent should take in each state in order to maximize the expected reward.

The set of possible actions, A, is small and well-defined. In our application, we have $A = \{Elicit, Tell\}$ for inducing pedagogical strategies on ET decisions and $A = \{Justify, Skip - Justify\}$ for inducing those on JS decisions. The set of possible states, S, however is not well-defined in advance and can potentially be astronomically large if we include everything that could possibly influence the effectiveness of a tutorial action. In this study, we assumed that S is the Cartesian product of a set of state features $F = \{F_1, \cdots, F_p\}$ and our challenge now becomes finding a set of features F to model the state or learning context compactly and yet effectively. Features must be operational, in that there is some way to determine their value prior to just before each tutor action in the dialogue. For instance, one operational feature would be a count of the number of words uttered by the student since the last tutor turn.

Each student-system interaction dialogue d can be viewed as a trajectory in the chosen state space determined by the system actions and student responses:

$$S_1 \xrightarrow{A_1, R_1} S_2 \xrightarrow{A_2, R_2} \cdots S_n \xrightarrow{A_n, R_n}$$

Here $S_i \xrightarrow{A_i, R_i} S_{i+1}$ indicated that at the i_{th} turn in the tutorial dialogue d, the system was in state S_i, executed action A_i, received reward R_i, and then transferred into state S_{i+1}. Because our primary interest is to improve students' learning, we used Normalized Learning Gain (NLG) as the reward because it measures students' gain *irrespective of their incoming competence*. The NLG is defined as: $NLG = \frac{posttest - pretest}{1 - pretest}$. Here *posttest* and *pretest* refer to the students' test scores before and after the training respectively; and 1 is the maximum score. Given that a student's NLG will not be available until the entire tutorial dialogue is completed, only terminal dialogue states have non-zero rewards. Thus for a tutorial dialogue d, $R_1 \cdots$, R_{n-1} are all equal to 0 and only the final reward equal to the student's $NLG \times 100$, which is in the range of (-∞, 100].

Once the MDP structure $\{S, A, R\}$ has been defined, the transition probabilities T are estimated from the training corpus, which is the collection of dialogues, as: $T = \{p(S_j|S_i, A_k)\}_{i,j=1,\cdots,n}^{k=1,\cdots,m}$. More specifically, $p(S_j|S_i, A_k)$ is calculated by taking the number of times that the dialogue is in state S_i, the tutor took action A_k, and the dialogue was next in state S_j divided by the number of times the dialogue was in S_i and the tutor took A_k. The reliability of these estimates clearly depends upon the size and structure of the training data. Once a complete MDP is constructed, a dynamic programming approach can be used to learn the optimal control policy π^* and here we used the toolkit developed by Tetreault and Litman [11]. The rest of this section presents a few critical details of the process, but many others must be omitted to save space.

3.1 Knowledge Component (KC) Based Pedagogical Strategies

In the learning literature, it is commonly assumed that relevant knowledge in domains such as math and science is structured as a set of independent but co-occurring Knowledge Components (KCs) and that KC's are learned independently. A KC is "a generalization of everyday terms like concept, principle, fact, or skill, and cognitive science terms like schema, production rule, misconception, or facet" [12]. For the purposes of ITSs, these are the atomic units of knowledge.

The domain selected for this project is a subset of the physics work-energy domain, which is characterized by eight primary KCs. For instance, one KC is the definition of kinetic energy ($KE = \frac{1}{2} * m * v^2$) and another is the definition of gravitational potential energy ($GPE = m*g*h$). It is assumed that a tutorial dialogue about one KC (e.g., kinetic energy) will have no impact on the student's understanding of any other KC (e.g, of potential energy). This is an idealization, but it has served ITS developers well for many decades, and is a fundamental assumption of many cognitive models [13, 14].

When dealing with a specific KC, the expectation is that the tutor's best policy for teaching that KC (e.g., when to Elicit vs, when to Tell) would be based upon the student's mastery of the KC in question, its intrinsic difficulty, and other relevant, but not necessarily known, factors specific to that KC. In other words, an optimal policy for one KC might not be optimal for another. Therefore, one assumption made in this paper is that inducing pedagogical policies specific to each KC would be more effective than inducing an overall KC-general policy. In order to learn a policy for each KC, we annotated our tutoring dialogues and action decisions with the KCs covered by each action. For each KC, the final kappa was ≥ 0.77, which is fairly high given the complexity of the task. Additionally, a domain expert also mapped the pre-/post test problems to the sets of relevant KCs. This resulted in a KC-specific NLG score for each student. Thus, for the decision of when to Elicit vs. Tell about the definition of kinetic energy KC_{20}, we consider all and only the dialogue about that KC and consider only the learning gains on that KC.

Given these independence assumptions, the overall problem of inducing a policy for ET decisions and a policy for JS decisions is decomposed into 8 sub-problems of each kind, one per KC. Among the eight KCs, KC_1 does not arise in any JS decisions and thus only an ET policy was induced for it. For each of the remaining seven KCs, a pairs of policies, one ET policy and one JS policy, were induced. So we induced 15 KC-based NormGain policies. During the tutoring process, there were some decision steps that did not involve any of the eight primary KCs. For them, two KC-general policies, an ET policy and a JS policy, were induced. To sum, a total of 17 NormGain policies were induced in this study.

3.2 Training Corpora

In order to apply RL to induce pedagogical strategies and evaluate the induced strategies, we used Cordillera. Cordillera is a NL tutoring system that teaches introductory physics[12]. To reduce confounds due to imperfect NL understanding, the NL understanding module was replaced with human wizards whose only task is to match students' answers to the closest response from a list of potential responses and they *cannot* make the tutorial decisions. As the first step, we developed an initial version of Cordillera, called random-Cordillera on which both ET and JS decisions on it were made randomly. 64 college students were then trained on random-Cordillera in 2007 and the collected training data is called the Exploratory corpus.

From the Exploratory corpus, we tried our first round of policy induction. It is done by first defining 17 state features and then used some sort of greedy-like procedure to search for a small subset of it as the state representation. For the reward functions, we had dichotomized the NLGs scores so that there were only two levels of reward and thus the derived policies were named DichGain policies. We next tested our hypothesis that these RL-induced policies would improve the effectiveness of a tutoring system. The version of Cordillera that implemented the DichGain policies was named DichGain-Cordillera. Except following the policies (random vs. DichGain), the remaining components of Cordillera, including the GUI interface, the same training problems, and the tutorial scripts, were left untouched. DichGain-Cordillera's effectiveness was tested by training a new group of 37 college students in 2008. Results showed that although the DichGain policies generated significantly different patterns of tutorial decisions than the random policy, no significant difference was found between the two groups on the pretest, posttest, or the NLGs.

3.3 Inducing NormGain Strategies

Although the previous experiment seemingly failed to confirm our hypothesis, it did generate more training data. We now have three training corpora: the Exploratory corpus in 2007, the DichGain corpus in 2008, and a combined training corpus dataset consisting of the 101 dialogues from both the Exploratory and the DichGain corpora. This time we started with a larger set of possible state features. We included 50 features based upon six categories of features considered by previous research [15–17] to be relevant. They include not only student's performance and background related features such as student's overall performance but also domain-oriented and system behavior related features. Moreover, we explored more domain -general methods of searching the power set of the 50 features and instead of dichotomizing learning gains as rewards, we used the $NLG \times 100$ directly. Based on the reward function, the induced policies are named normalized Gain (NormGain) policies in the following.

Figure 1 shows an example of a learned NormGain policy on KC_{20}, "Definition of Kinetic Enegy", for JS decisions. The policy involves five features. They are:

TimeInSession: The total time spent in the current session. This feature reflects a student's fatigue level.

nKCs: The number of times the present KC has occurred in the current dialogue. This feature reflects the students' familiarity with the current KC.

pctElicit: The percentage of ET decisions turned out to be elicit during the dialogue. This feature reflects how active a student is overall.

stuAverageWords: The average number of words per student turn. This reflects the student's level of activity and verbosity.

stuAverageConceptSession: The ratio of the number of the student's turns which involves at least one physics concept to all the student turns in this session. This feature reflects how often the student's answers involved at least one physics concepts since the start of the training.

[Feature:]

TimeInSession:	$[0, 3040.80) \rightarrow 0$;	$[3040.8, \infty] \rightarrow 1$
nKCs:	$[0, 66) \rightarrow 0$;	$[66, \infty] \rightarrow 1$
pctElicit:	$[0, 0.49) \rightarrow 0$;	$[0.49, 1) \rightarrow 1$
stuAverageWords:	$[0, 4.18) \rightarrow 0$;	$[4.18, \infty] \rightarrow 1$
stuAverageConceptSession:	$[0, 0.29) \rightarrow 0$;	$[0.29, 1] \rightarrow 1$

[Policy:]

Justify:

0:0:0:0:0 0:0:1:1:0 0:1:0:0:1 0:0:1:0:0 0:1:0:1:1 0:1:1:0:0 0:1:1:0:1 0:1:1:1:0 0:1:1:1:1 1:0:0:0:0 1:0:0:1:0 1:0:1:0:0 1:0:1:0:1 1:0:1:1:0 1:0:1:1:1 1:1:0:0:1 1:1:1:0:0 1:1:1:0:1 1:1:1:1:0 1:1:1:1:1

Skip-Justify:

0:0:0:0:1 0:0:0:1:0 0:0:0:1:1 0:0:1:0:1 0:0:1:1:1 0:1:0:0:0 0:1:0:1:0 1:0:0:0:1 1:0:0:1:1 1:1:0:0:0 1:1:0:1:0 1:1:0:1:1

Fig. 1. An NormGain Policy on KC_{20} For JS Decisions

MDP generally requires discrete features and thus all the continous features need to be discretized. Figure refFig.ExampleNormGainPolicy describes how each of the five features was discretized. For example, for TimeInSession, if its value is above 3040.80 sec (50.68 min), it is 1 otherwise, it is 0. There were a total of 32 rules learned: in 20 situations the tutor should execute the justification step, in the other 12 situations the tutor should skip. For example, 0:0:0:0:0 is listed as the first situation under the [Justify], it means that when the student has spend less than 50.68 min in this session, the occurrence of KC_{20} in the student's dialogue history is less than 66, the student has got less than 49% of elicit in the past, the average number of words in student's entries is less than 4.18 words, and the percentage of times times that the student mention a physics concept in his/her turn is less than 29%, then the tutor should execute the justification. As you can see, the RL induced policies are very subtle and adaptive to the learning context and they are not like most of the tutorial tactics derived from analyzing human tutorial dialogues.

The resulting 17 NormGain policies were implemented back into Cordillera yielding a new version of the system, named NormGain-Cordillera. In order to test our hypothesis that RL can be used to improve tutoring systems, we tested the effectiveness of NormGain-Cordillera on a new group of students as described in the next section. The section is written as if one large experiment was done with 3 conditions, when in fact the 3 groups of students were run sequentially, as described above.

4 Methods

The purpose of this experiment is to compare the learning gains of students using random-Cordillera, DichGain-Cordillera and NormGain-Cordillera respectively. All participants were required to have basic knowledge of high-school algebra, no experience with college-level physics, and were paid for their time. Each participant took between six and fourteen hours (3-7 sessions) to finish the study in a period of two to three weeks. Each session typically lasted about two hours.

The domain selected here is Physics work-energy domain as covered in a first-year college physics course. The eight primary KCs were: the weight law (KC1), definition of work (KC14), Definition of Kinetic Energy (KC20), Gravitational Potential Energy (KC21), Spring Potential Energy (KC22), Total Mechanical Energy (KC24), Conservation of Total Mechanical Energy (KC27), and Change of Total Mechanical Energy (KC28).

All three groups experienced the identical procedure and materials. More specifically, participants all completed a background survey; read a textbook covering the target domain knowledge; took a pretest; solved the same seven training problems in the same order on Cordillera; and finally took a posttest. The pretest and posttest were identical.

Only three salient differences existed across the three groups:

1. The Exploratory group with a population of 64 was recruited in 2007; the DichGain group with a population of 37 was recruited in 2008; and the NormGain group with a population of 29 was recruited in 2009.
2. Random-Cordillera made random decisions and the DichGain-Cordillera and NormGain-Cordillera followed the induced DichGain and NormGain policies respectively.
3. A group of six human wizards were used by the Exploratory and DichGain groups; but only one of six wizards were involved in the NormGain group.

4.1 Grading

All tests were graded by a single experienced grader who did not know which student belonged to which group. For all identified relevant KCs in a test question, a KC-based score for each KC application was given. We assigned an overall competence to a student by the sum of these KC-based scores and normalizing to a [0,1] interval. We also tried other methods of computing an overall score, and this did not affect the pattern of results discussed below.

5 Results

The primary goal reported below is twofold: first, to test whether our improved RL methodology and software produced more effective pedagogical strategies than either random policies or the policies used by the DichGain group; and second, to determine the features selected in the state models in the NormGain policies.

5.1 Learning Results

A one-way ANOVA showed that there were no significant differences among the three groups on overall training time: $F(2, 122) = 1.831$, $p = .17$. After solving seven training problems on Cordillera, all three groups scored significantly higher in the posttest than pretest: $F(1, 126) = 10.40$, $p = 0.002$ for the Exploratory group, $F(1, 72) = 7.20$, $p = 0.009$ for the DichGain group, and $F(1, 56) = 32.62$, $p = 0.000$ for the NormGain group respectively. The results suggested that the basic practices and problems, domain exposure, and interactivity of Cordillera might cause students to learn even from tutors with non-optimal pedagogical skills.

A one-way ANOVA was used for comparing the learning performance differences among the three groups. While no significant pre-test score differences were found: $F(2, 127) = 0.53, p = 0.59$, there were significant differences among the three groups on both post-test scores and NLG scores: $F(2, 127) = 5.16, p = .007$ and $F(2, 127) = 7.57, p = 0.001$ respectively. Figure 2 compares the three groups on the pre-test, post-test, and NLG scores. Moreover, a t-test comparison showed that the NormGain group out-performed the DichGain on both post-test scores and NLG scores: $t(64) = 3.28, p = .002, d$[5] $= 0.82$ and $t(64) = 3.68, p = 0.000, d = 0.95$ respectively. Similar results were found between the NormGain and Exploratory groups: $t(91) = 2.76, p = .007, d = 0.63$ on post-test, and $t(91) = 3.61, p = 0.000, d = 0.84$ on NLG scores respectively.

To summarize, the comparison among the three groups shows that the NormGain group significantly outperformed both the Exploratory and DichGain groups. These results were consistent both for the post-test scores and the NLGs and the effect sizes were large by Cohen's d criteria.

5.2 Feature Choices in INDUCED POLICIES

Only 30 out of 50 defined features occurred among the 17 NormGain policies. Among them, the most frequent feature appeared seven times. Four features appeared in more than three induced policies and they are:

StepDifficulty (7 Occurrences): which encodes a step's difficulty level and its value is roughly estimated from the Combined Corpus based on the percentage of answers that were correct on the step.

[5] Cohen's d, which is defined as the mean learning gain of the experimental group minus the mean learning gain of the control group, divided by the groups' pooled standard deviation.

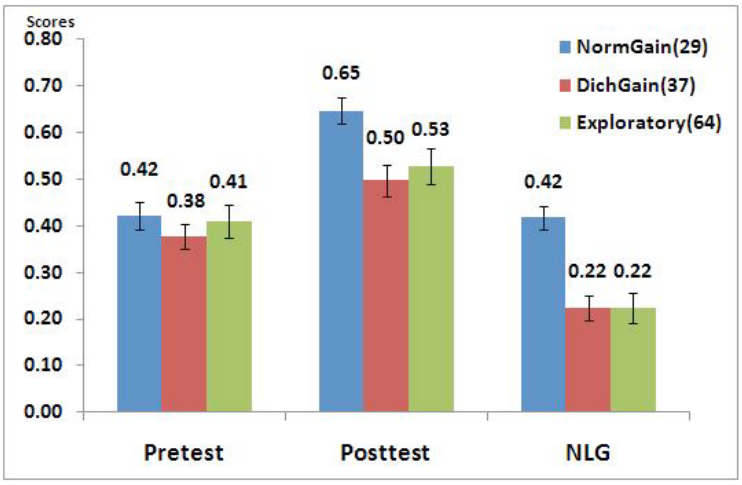

Fig. 2. Compare Three Groups Learning Performance under Overall Grading

ConceptToWordRatio (5 Occurences): which represents the ratio of the physics concepts to words in the tutor's dialogue.

NumberTellsSinceElicit (5 Occurences): which represents the number of tells the student has received since the last elicit.

TimeBetweenDecisions (4 Occurences): which represents the time since the last tutorial decision was made on the current KC.

While StepDifficulty can be seen as domain-oriented feature, the remaining three features are all the system-behavior related features. The high occurrence of StepDifficulty in the NormGain policies is not very surprising because it has been widely believed that difficulty level is an important factor for the system to behave adaptively and effectively. The frequent involvement of System-behavior related features in the induced policy maybe because these features might reflect student's general aptitude, the activeness of their knowledge on a specific KC, and so on. For example, NumberTellsSinceElicit reflects how interactive a student has been recently and TimeBetweenDecisions reflect how active a student's knowledge on the current KC is. When TimeBetweenDecisions is high, it means that the tutor has not mentioned the KC recently so the student's knowledge on the current KC may be still or forgotten.

Much to our surprise, the features related to the students' overall or recent performance (e.g., error rate) and background (e.g., MSAT, VSAT, gender, pretest score) appeared the least or none in the NormGain policies. Although space does not permit a detailed discussion of the prevalence of features, it appears to be a mixture of easily anticipated dependencies (e.g., step difficulty) and a few surprises (why doesn't error rate matter?).

6 Conclusions

We presented a general data-driven method that can be used to improve NL tutoring system over time. We built and improved a large NL tutoring system using our methodology, and showed that RL is able to effectively search a very large continous space of dialogue policies (After discretized, the space is $\geq 2^{50}$ in size) using a relatively small amount of training dialogue data (64 subjects in Exploratory group and 37 in the DichGain group). A post-hoc comparison showed that our learned policy outperformed both sets of training policies in terms of learning performance. This success supports the hypothesis that RL-induced rules are effective and that the approach taken in this project was a feasible one. However, inducing effective tutorial tactics was not trivial. The DichGain tutorial tactics did not seem to be more effective than the random decisions in Random-Cordillera. A number of factors were changed in deriving NormGain policies from the process of inducing DichGain policies. These included the feature choices, the choice of training corpora, feature selection methods, and so on. So it is still not clear which factor or factors caused a change in effectiveness.

Although the discussion of induced features has been cursory, it nonetheless appears that the learning context features that make the most difference for determining when to Tell vs. Elicit and when to Justify vs. Skip-Justify are not always the ones that one would first think of given current theories of learning and tutoring. For instance, it is widely believed that effective tutors adapt their behavior to the individual student knowledge level. However, such feature did not appear in the NormGain policies. Indeed, individualized tutoring is considered a Grand Challenge by the National Academy of Engineering. However, such features appeared to play little role in the effective tutorial policies induced from our data. Overall, our results suggested that when building an accurate learning context model, adding domain-oriented and the system behavior related features would be beneficial.

Acknowledgments NSF (#0325054) supported this work and NSF (#SBE-0836012) supported its publication. We thank Collin Lynch and the reviewers for helpful comments.

References

1. VanLehn, K., Jordan, P.W., Rosé, C.P., Bhembe, D., et al.: The architecture of why2-atlas: A coach for qualitative physics essay writing. In Cerri, S.A., Gouardères, G., Paraguaçu, F., eds.: Intelligent Tutoring Systems. Volume 2363 of Lecture Notes in Computer Science., Springer (2002) 158–167
2. Chi, M.T.H., Siler, S.A., Jeong, H., Yamauchi, T., Hausmann, R.G.: Learning from human tutoring. Cognitive Science **25** (2001) 471–533
3. Singh, S.P., Litman, D.J., Kearns, M.J., Walker, M.A.: Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. J. Artif. Intell. Res. (JAIR) **16** (2002) 105–133

4. Raux, A., Langner, B., Bohus, D., Black, A.W., Eskenazi, M.: Let's go public! taking a spoken dialog system to the real world. In: Proceedings of Interspeech (Eurospeech). (2005)
5. Collins, A., Brown, J.S., Newman, S.E.: Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In Resnick, L.B., ed.: Knowing, learning and instruction: Essays in honor of Robert Glaser. Lawrence Erlbaum Associates: Hillsdale New Jersey (1989) 453–494
6. Chi, M.T.H., de Leeuw, N., Chiu, M.H., LaVancher, C.: Eliciting self-explanations improves understanding. Cognitive Science **18**(3) (1994) 439–477
7. Conati, C., VanLehn, K.: Toward computer-based support of meta-cognitive skills: a computational framework to coach self-explanation. International Journal of Artificial Intelligence in Education **11** (2000) 398–415
8. Katz, S., ODonnell, G., Kay, H.: An approach to analyzing the role and structure of reflective dialogue. International Journal of Artificial Intelligence and Education **11** (2000) 320–343
9. Singh, S.P., Kearns, M.J., Litman, D.J., Walker, M.A.: Reinforcement learning for spoken dialogue systems. In Solla, S.A., Leen, T.K., Müller, K.R., eds.: NIPS, The MIT Press (1999) 956–962
10. Sutton, R.S., Barto, A.G.: Reinforcement Learning. MIT Press Bradford Books (1998)
11. Tetreault, J.R., Litman, D.J.: A reinforcement learning approach to evaluating state representations in spoken dialogue systems. Speech Communication **50**(8-9) (2008) 683–696
12. VanLehn, K., Jordan, P.W., Litman, D.: Developing pedagogically effective tutorial dialogue tactics: Experiments and a testbed. In: Proceedings of SLaTE Workshop on Speech and Language Technology in Education ISCA Tutorial and Research Workshop. (2007)
13. Anderson, J.R.: The architecture of cognition. Cambridge, Mass. : Harvard University Press (1983)
14. Newell, A., ed.: Unified Theories of Cognition. Harvard University Press; Reprint edition (1994)
15. Moore, J.D., Porayska-Pomsta, K., Varges, S., Zinn, C.: Generating tutorial feedback with affect. In Barr, V., Markov, Z., eds.: FLAIRS Conference, AAAI Press (2004)
16. Beck, J., Woolf, B.P., Beal, C.R.: Advisor: A machine learning architecture for intelligent tutor construction. In: AAAI/IAAI, AAAI Press / The MIT Press (2000) 552–557
17. Forbes-Riley, K., Litman, D.J., Purandare, A., Rotaru, M., Tetreault, J.R.: Comparing linguistic features for modeling learning in computer tutoring. In Luckin, R., Koedinger, K.R., Greer, J.E., eds.: AIED. Volume 158 of Frontiers in Artificial Intelligence and Applications., IOS Press (2007) 270–277