Machine Learning in Education

Payam Goodarzi[1]

Abstract— **It is indeed beyond question, that Machine Learning (ML) has been prevailing over the last decade. ML has already proved its tremendous power and suitability for solving difficult problems in the realm of computer science. This rapid-growing branch of artificial intelligence (AI) is arresting a great deal of attention on itself due to its promising results, and it keeps getting harder to find an area, where no trace of ML could be found. One of its interesting applications is in education. As the number of E-Learning systems continuously rises, ML contributes more and more to improve the performance of these systems to accomplish the ultimate objective, which is simplicity and effectiveness of the learning process. ML can help E-Learning systems in many different ways to become more intelligence than before, for instance by analyzing the performance and learning behaviour of students individually, which can provide useful information for adjusting the tutoring system appropriately. By taking this into consideration, that AI has a revolutionary influence on education, one question arises, will be able to replace the teachers entirely in the nearest future with ML? Well, It is certainly a fishy question. In order to answer this question, i believe we should have a deeper insight into the technical parts of Intelligence Tutoring Systems (ITSs). In this paper we bring one highly intelligence ITS [1] into focus, which employs Reinforcement Learning (RL) to induction of effective and adaptive pedagogical strategies.**

I. INTRODUCTION

The capability of solving a wide variety of challenging problems is the most sensational property of human being. The main goal of Reinforcement Learning is to artificially produce it by creating an agent, who like a human can learn for themselves to achieve successful strategies that lead to the greatest rewards by interacting with an environment. This paradigm of learning by trial-and-error, solely from rewards or punishments, molds somehow our real life. As it's demonstrated in figure 1 generally all RL Systems consist of an Agent and an Environment. The agent acts, thereby goes from one state to another and instead receives feed-backs from environments. Based on the feed-backs the agent try to improve its actions, in a manner that leads him to the goal with the highest possible reward. To instantiate, in one ITS teacher plays the role of the agent and students are the environment. During the tutoring process the teacher makes decision at each state and proceed to next state. States in ITS could be a set of all possible steps in a tutoring workflow from the beginning to the end, where the teaching process is over. One piece of this puzzle is missing! right, the Reward. When the teacher gets feed-backs? in ITSs, where the RL algorithms are applied like *Policy Iteration* in [1], defining a proper reward function is problematic. Providing immediate feed-backs is more effective than receiving delayed feed-backs. Short-term returns allow the agent to react quicker and rectify its actions appropriately, eventually this leads to an more adaptive behaviour. But in ITSs like [1] the feed-backs are not available until the whole tutoring is over. In this case we deal with long-term feed-backs, that complicate it to track those specific actions, which yield to the highest reward value. One way to overcome this issue is to propagate back the reward value by putting a so called discount factor into the calculation, we investigate it more in section III.

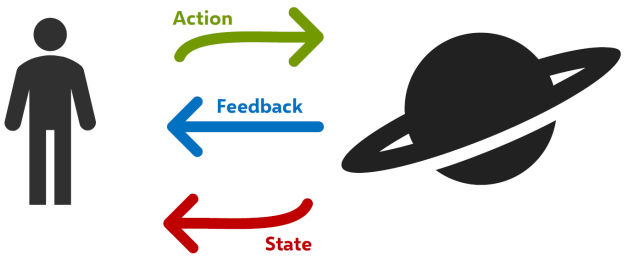

Fig. 1. Reinforcement Learning.

In general RL algorithms are classified into two types depending on the agent understanding and perception of the environment, *model-based* and *model-free*. By model we mean anything that an agent can use to predict how the environment will respond to its actions. As mentioned before in [1] policy iteration was used, which is one model-based algorithm, since the agent relies on exploratory gathered data and models the environment by means of that. In section IV we will see more of the exploratory corpus in [1]. The foundation of almost every RL system is *Markov Decision Process* (MDP). Section II explains briefly about MDP and why a markovian process can be extended to a RL system.

After the introductory sections we delve into the work done by [1] and see closely how policy iteration improves the learning gain of students by allowing the Cordillera [2] to tailor its behaviour with respect to needs of students. Cordillera [2] is the ITS involved in this project, a Natural Language (NL) tutoring system, which teaches students introductory physics.

II. MARKOV DECISION PROCESS

MDP provides the framework to model a system. Every system, that can be considered as a sequential decision process wherein, at each discrete step the system is responsible for selecting the next action to take, is a MDP. With other words each decision making process, that satisfy the Markov

[1]Payam Goodarzi is computer science master student at the free university of berlin payam.goodarzi@fu-berlin.de

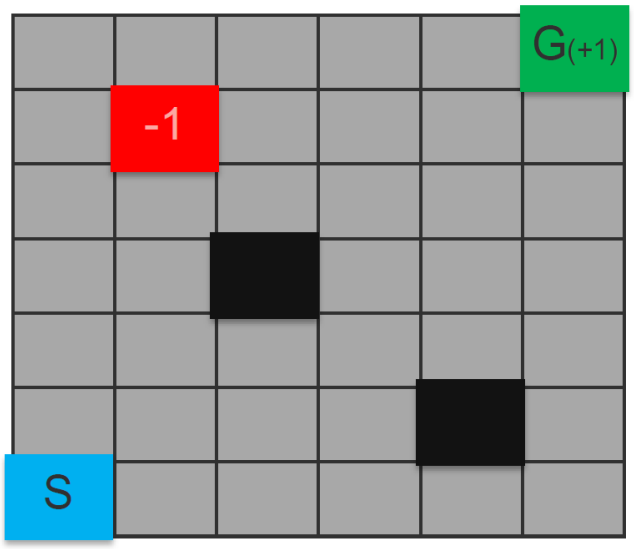

Fig. 2. Example for MDP. A grid, where the agent starts from *S* and the goal is to reach *G* with highest reward value.

Property, is one MDP and can be handled mathematically. Markov property refers to *memoryless* property of a stochastic process, it means, that conditional transition probability to future state depends only upon the present state, not on the sequence of events that preceded it. The following are the main components of a MDP:

- **State set:** S It is a set of all states in the environment.
- **Action set:** $A(s)$ Set of all possible actions, that the agent can take.
- **Model or Transition probability:** $T(s, a, s') \sim Pr(s'|s, a)$ Probability of going from state s to s' taking action a.
- **Reward function:** $R(s), R(s, a, s')$ The feed-back given, if the agent goes from state s to s' by taking action a.
- **Policy:** $\pi\,(s) \rightarrow a$ Given state s, it outputs the action to take. We actually look for the optimal policy π^*.

The grid in figure 2 is a widely used example in RL books or tutorial, we use it also to get a better sense of MDP. In this example states are the cells of the grid, the agent has only these possible actions $A = move - up, move - down, move - right, move - left$, the rewards of grey cells is 0.5 and the red one -1, so the agent should avoid the red cell. Starting from S the goal is to reach the cell G by moving through the cells with the highest final reward values. The optimal policy π^* would be the best sequence of actions, that maximizes the sum of all rewards given on the way from S to G.

As exemplified here, ITSs are also in some way a real-life example of MDP. For many types of E-Learning environments, the system's behaviour can be also viewed as a sequential decision process. The teacher begins the tutoring from a start point S and at each state chooses one action from the set $A(s)$, such as asking a question, telling the student to proceed, suggesting extra learning material, giving some hints for solving a problem, showing a video, explaining a topic, etc. After the teacher reached state G (End of the tutoring), function $R(s)$ provides a feed-back to the teacher. The teacher should find the best pedagogical strategy, the optimal policy π^*, that yields the highest reward, which for instance in [1] is the learning gain of the student. But how we can mathematically calculate this optimal policy? Next section answers this question by putting light on the *policy iteration* algorithm, since it's the main tool used in [1] to induce the adaptive pedagogical policies.

III. Policy Iteration

Policy iteration is a model-based algorithm built upon a equation called, *Bellman Equation*, named after Richard Bellman who introduced dynamic programming in 1953. Equation 1 is the very celebrated bellman equation and is simply the average over all possible actions at states s, each weighted by its probability of occurring, given policy π at state s. The Value function $v^\pi(s)$ states that the value of the start state s must equal the discounted value of expected next state plus the reward expected along the way till the terminal state. With other words it shows us, *how good* is to be at state s. γ in the equation 1 is the discount factor, that plays a vital role, it discretizes the value function and ensures, that the value decreases continuously on the way to the termination point. Thus the rewards of the beginning states are more weighted than the last ones, which is great, since the value of early states (short-term reward) matters more than the last states (long-term reward) as we discussed already.

$$v^\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s, a)\,[r(s, a, s') + \gamma v^\pi(s')] \tag{1}$$

Policy iteration finds the optimal policy π^* by iterating over all policies. It consists of two parts, at each iteration it evaluates a policy by calculating the value function $v^\pi(s)$ and then improves the policy by solving the equation 2.

$$\pi(s) \leftarrow argmax_a \sum_{s'} p(s'|s, a)\,[r(s, a, s') + \gamma v(s')] \tag{2}$$

In a finite MDP such as NL tutoring systems in different domains like math, physic and so forth, policy iteration can be applied to find the optimal strategy to deliver some knowledge components to students. In next sections we evaluate the application of policy iteration on Cordillera [2].

IV. Application of RL in ITS

A team led by Min Chi from Carnegie Mellon University in Pittsburgh attempted to empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. They addressed technical challenges in applying RL to Cordillera [2]. The project is divided into two phases as follow:

- **Training phase**
- **Test phase**

In the test phase some sort of evaluations were performed to find out whether the induced policies by RL fulfill our expectations. The training phase basically involves defining an appropriate state representation S, a reasonable action space $A(s)$, and an appropriate reward function $R(s)$.

In the training phase a training corpus Γ was also collected by letting 64 students interact with the NL tutoring system Cordillera, while it took actions randomly. One key different of this project in comparison to other related works in this area is that they gathered this training corpus by training *human student* not simulated student. Because creating accurate simulated students is not easy, since studying and analyzing human learning behaviour is especially challenging and there is many hidden factors involved in human learning, that are poorly understood. Collecting data on NL tutoring systems is very expensive, therefor many previous research used pre-existing data extracted from logs of other ITSs. The problem of pre-existing data is, that they are usually not completely suitable and accurate for a specific tutoring system, at the other hand pre-processing the data is always a troublesome task. All these arguments discourage the team of this project from using the pre-existing data.

For RL, as with all machine learning tasks, success depends upon an effective state representation S. Ideally S should include all of the relevant dialogue history necessary to determine which action should be taken next. One obvious but impractical choice is to use a complete record of the dialogue to the present point; however, in practice we need to compress the dialogue history to make the space tractable. In other words, an effective representation S should be an accurate and compact model of the learning context. The challenge thus lies in identifying useful state features. In more complex domains, state representation S gets bigger and more complicated, because there are many factors which might determine whether a student learns well from the ITS. Hence the states need to include all features for anything that is likely to influence the learning. In next section we investigate feature selection procedure in this project.

V. State feature selection for inducing pedagogical policies

A group of experts defined a large set of potential state features denoted by Ω, then $\hat{m}$(maximum number of features) features were selected by performing some feature selection techniques. $\hat{m}$ should be large enough to represent states included essential features that help the system to take good decisions. They capped number of features in each policy at six ($\hat{m} = 6$), which means that there can be as many as $2^6 = 64$ rules in the learned policy. There were six different categories of features involved in this project considered by previous research [3].

- **Autonomy (A):** amount of work performed by student. Exp:[tellsSinceElicitA]
- **Background (BG):** general background information about student.
- **Problem Solving Contextual (PS):** encode information about the current problem-solving context. Exp:[StepSimplicityPS]
- **Performance (PM):** Exp:[pctCorrectKCPM]
- **Student dialogue (SD):** characterizes student language. Exp:[stuAverageWordSD]
- **Temporal Situation (T):** time related information about problem solving. Exp:[durationKCBetweenDecisionT]

All of them can be calculated automatically and are discretized using two clustering procedures: the TwoStep procedure bounded the number of clusters in SPSS and the K-means procedure used K-means clustering to locate the optimal cluster centers. After the discretization the values had been binarized. For instance *StepSimplicityPS* that shows the difficulty level of the current context. we have $StepSimplicityPS : [0, 0.38) \rightarrow 0; [0.38, 1] \rightarrow 1$, which means if *StepSimplicityPS* value is below 0.38, it is 0 (hard) otherwise, it is 1 (easy). So far we have defined the state set S, next section describes action set $A(s)$.

VI. Action space

At each discrete step the tutor can choose between only two type of decisions in this project.

- **Elicit/Tell (ET)**
- **Justify/Skip-justify (JS)**

Human one-on-one tutoring is characterized by a mixture of elicits and tells. During the tutoring tutor confronts often with a simple decision, *tell* a students the next step, or *elicit* it from student. Since the tutor should adapt its actions to the student's needs based upon their current knowledge and status ([4]), in such situation most existing ITS decides to elicit ([5], [6], [7]). Figure 3 compares Elicit and Tell. As stated before defining an effective reward function is perhaps the trickiest part of a MDP and specially in ITSs. In next section we take close look at the approaches made in this project for evaluation.

(a) Elicit Version

1. **T:** So let's start with determining the value of v1.
2. **T:** Which principle will help you calculate the rock's instantaneous magnitude of velocity at T1? **{ELICIT}**
3. **S:** definition of kinetic energy
4. **T:** Please write the equation for how the definition of kinetic energy applies to this problem at T1 **{ELICIT}**
5. **S:** ke1 =0.5*m*v1^2
6. **T:** From KE1 = 0.5*m*v1^2, $\cdots$

(b) Tell Version

1. **T:** So let's start with determining the value of v1.
2. **T:** To calculate the rocks instantaneous magnitude of velocity at T1, we will apply the definition of kinetic energy again. **{TELL}**
3. **T:** Let me just write the equation for you: KE1 = 0.5*m*v1^2. **{TELL}**
4. **T:** From KE1 = 0.5*m*v1^2, $\cdots$

Fig. 3. Elicit vs Tell.

VII. Evaluation metrics and Reward function $R(s)$

Contrary to other works [8], they used only one evaluation metric for rating the policies, the *Expected Cumulative Reward* (ECR). ECR can be calculated as follow:

$$ECR_\pi = \sum_{i=1}^{n} \frac{N_i}{N_1 + \cdots + N_n} \times V(s_i) \tag{3}$$

where $s_1, \cdots, s_n$ is the set of all starting states and $v(s_i)$ is the V-values for state s_i ; N_i is the number of times that s_i appears as a start state in the model and it is normalized by dividing $\frac{N_i}{N_1+\cdots+N_n}$. To put in another way, it calculates the sum over all the initial start states and weights them by the frequency with which each state appears as a start state. The higher the ECR value of a policy, the better the policy is supposed to perform.

Let assume that we have two states s_1 and s_2, we calculate the transition probability T of these states from the training corpus Γ. $T_1(s_1|s_1, a_1) = 3/10 = 0.3$ is the probability that the agent at state s_1 comes back to state s_1 taking action a_1 given that, there were 10 times overall that action a_1 was taken. three times out of these the agent transitioned back to state s_1. We have the same for s_2 as follow: $T_1(s_2|s_2, a_2) = 300/1000 = 0.3$. The probability that the agent comes back to state s_2 taking action a_2 is equal to the case, if the agent at state s_1 comes back to s_1 taking a_1. While both set of transitions have the same parameter ($T_1 = T_2$), the second set is more reliable. Clearly ECR lacks reliability, to tackle this issue Tetreault and Litman [8] proposed a Confidential Interval (CI) estimate based upon the available data in the exploratory corpus Γ.

In this project reward function is based on a *Normalized Learning Gain* (NLG). NLG compares the knowledge level of student before and after the training irrespective of his/her incoming competence. To evaluate the knowledge of the students two tests were taken, *pretest* and *posttest*. Thereafter NLG was measured as follow:

$$NLG = \frac{posttest - pretest}{1 - pretest} \tag{4}$$

Now that the dialogue system is modeled and the 4-tuple (S, A, T, R) is available we can see one simple example. The policy shown in figure 4 involves only a single feature policy, where the tutor can either elicit or tell, with respect to the value of *StepSimplicityPS*. *StepSimplicityPS* is estimated from the training corpus based on the percentage of correct answers when the tutor has done an Elicit (i.e., asked a question). The higher the value of StepSimplicityPS, the easier is the current context and this particular policy with ECR value of 8.09 tells the agent to either Elicit or Tell, while if the StepSimplicityPS is 0 (Rule 1) the tutor should ask a question (Elicit). CI estimations states, that there is 95% chance that the ECR of the learned policy is between a lower-bound of 4.37 and an upper-bound of 12.07.

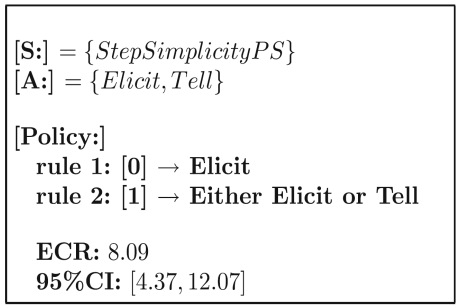

Fig. 4. Using StepSimplicityPS to induce a single feature policy on ET decision.

To demonstrate, that the policies with more than only one feature could score higher ECR, we put another example policy with three features into comparison with the single feature policy. The example policy in figure 5 beside StepSimplicityPS takes two other feature into account. $TuConceptsToWordsPS : [0, 0.074) \rightarrow 0; [0.074, 1] \rightarrow 1$ represent the ratio of the physic concepts to words in the tutor's expressions so far. $TuAvgWordsSesPS : [0, 22.58) \rightarrow 0; [22.58, \infty] \rightarrow 1$ encodes the average number of words in tutor turns in a session. This feature reflects how verbose the tutor is in the current session. Since each of features was discretized and has two outcome, this three-feature state representation result in a state space of $2^3 = 8$. As you can see the three-feature policy has higher ECR (14,25 vs 8,09) and also the lower-bound value is increased (18,12 vs 4,37). Thus the three-feature is more effective and robust than the single-feature policy. This reemphasizes the importance of the state features and certify the fact, that the state representation has a strong impact on the outcome of the whole system. Not necessarily the higher number of features involved can yield to a better result, but actually the features, which are highly correlated with the learning gain of students are more of the interest.

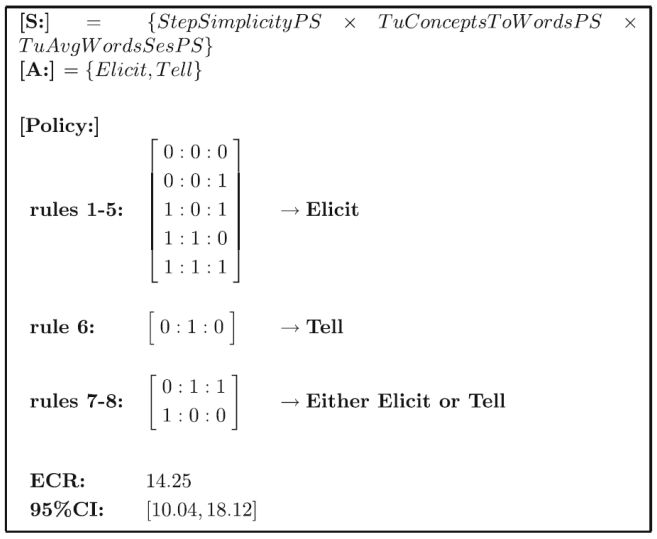

Fig. 5. An example of selected best policy on ET decisions.

		Defined features	Feature occurrences
1	Autonomy (A)	5	8
2	Background (BG)	5	1
3	Performance (PM)	12	5
4	Problem Solving Contextual (PS)	15	30
5	Student Dialogue (SD)	10	8
6	Temporal Situation (T)	3	7
7	Total	50	59

Fig. 6. Occurrence of six category features in the final tutorial policies.

The table illustrated in figure 6 lists the number of features defined in each six categories and the feature occurrences in the final policies. For example we can infer the third and fourth columns in row 4 in this table, that there are fifteen PS features defined and they account for thirty out of 59 feature occurrences in the final tutorial policies. In other words, more than half of all feature occurrences in the policies were from PS. Across the 50 features, the most frequent feature *Step-SimplicityPS* appears seven times. Interestingly the domain-oriented features are more involved and affect strongly the behaviour of the tutor during the tutoring process, which is somehow an understandable fact. Depending on the domain, in which the tutoring takes place, the performance of the ITSs varies. In more complex domains is indeed unlikely to see the same performance as in the basic domains.

VIII. General approach and the Procedure

All the participants experienced the same procedure as follow: 1) Background survey (gathering background information), 2) Pre-training (to familiarize the students with the system), 3) Pre-test, 4) Training (interacting with the system and going through the tutoring process), 5) Post-test. Pre-test and the Post-test were identical, but the participants were not aware of that. This project has three stages. Each stage differs in term of the pedagogical policies employed for interactive tutorial decisions.

- **Stage 1, Exploratory-Cordillera:** Cordillera made interactive decisions randomly.
- **Stage 2, DichGain-Cordillera:** The student's NLGs were dichotomized into +100 and -100 as reward functions ([9]).
- **Stage 3, NormGain-Cordillera:** Used normalized $NLG \times 100$ as reward function.

Figure 7 shows, that NormGain group outperformed both the Exploratory and DichGain groups with a significant margin. The induced pedagogical policies by RL improved undoubtedly the performance of the Cordillera. The main difference in the three stages introduced above is the reward function. Without a reasonable, accurate and effective reward function the system can not extract the optimal policy between a large amount of possible policies. Having a smartly defined reward function helps the agent to make smarter decisions and take the efficient path, ultimately it enhances the adaptivity of the agent.

To summarize, we described a practical methodology for using RL to improve effectiveness of an ITS. In a nutshell the methodology is to define and tune accurately the 4 components of MDP (S, A, T, R), feeding the system with an exploratory dataset, applying some feature selection methods to find the most effective state features, compute the optimal dialogue policy according to the learned MDP and finally deploy the induced policies and evaluate the policy on a new group of users.

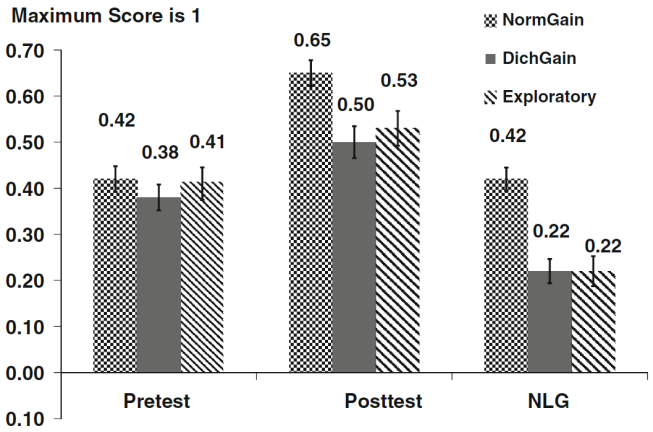

Fig. 7. Compare three groups learning performance under overall grading.

IX. Conclusion

Having evaluated the terrific project [1] as one example of the application of machine learning in education now i arrive at my conclusion and try to answer our first question, whether the teachers will be replaceable with ML in the future?. The project presented on this paper was done in 2010, meanwhile many other valuable research in this area had been developed with the help of new techniques and tools, such as *Recurrent Neural Networks* (RNN). For example In "Deep Knowledge Tracing" [10] they use a machine that models the knowledge of a student as they interact with coursework using RNN. Or the research of the university of California Berkeley [11] which serves as a foundation for applying sequential, generative models towards creating personalized recommenders in Massive Open Online Courses (MOOC) using "Long Short-Term Memory" architecture of the RNN and many other cutting edge projects, that make progresses and push the boundaries of ML towards the ultimate goals of E-Learning system, each of these works are weapons in our arsenal. The progression of artificial intelligence is indeed undeniable and unstoppable. The burst of technology in the field of computer science and its rapid growth during the last century may escalated falsely our expectations of computers. The tempo of the developments is not always steady and there is ups and downs. Take a look at the history of physic, after Isaac Newton it took the humanity more than about a century, till another genius Albert Einstein was raised and changed our whole understanding of the

universe and opened a new chapter in physic. The road to our objective of AI is a long bumpy road. Doubtlessly we will be some day able to replace the teacher with ML, but it is excessively optimistic to believe, that this will be realized within the next ten or even twenty years.

REFERENCES

[1] Chi, Min, et al. "Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies."User Modelingand User-Adapted Interaction21.1-2 (2011): 137-180.

[2] VanLehn, K., Jordan, P., 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, pp. 1720, 2007b

[3] 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.): Artificial Intelligence in Education, Building Technology Rich Learning Contexts that Work, Proceedings of the 13th International Conference on Artificial Intelligence in Education, AIED 2007, vol. 158 of Frontiers in Artificial Intelligence and Applications, pp. 270277, Los Angeles, California, USA, July 913. IOS Press, Amsterdam (2007)

[4] Pain, H., Porayska-Pomsta, K.: Affect in one-to-one tutoring. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.): Intelligent Tutoring Systems, 8th International Conference, ITS 2006, p. 817, Jhongli, Taiwan, 2630 June 2006, Proceedings, vol. 4053 of Lecture Notes in Computer Science. Springer, Berlin (2006)

[5] Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A.: Intelligent tutoring goes to school in the big city. Int. J. Artif. Intell. Educ. 8(1), 3043 (1997)

[6] VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M.: The andes physics tutoring system: lessons learned. Int. J. Artif. Intell. Educ. 15(3), 147204 (2005)

[7] Litman, D.J., Silliman, S.: Itspoke: an intelligent tutoring spoken dialogue system. In: Demonstration Papers at HLT-NAACL 2004, pp. 58. Association for Computational Linguistics, Morristown, NJ, USA (2004)

[8] Tetreault, J.R., Litman, D.J.:Areinforcement learning approach to evaluating state representations in spoken dialogue systems. Speech Commun. 50(89), 683696 (2008)

[9] Chi, M., Jordan, P.W., VanLehn, K., Litman, D.J.: To elicit or to tell: does it matter?. In: Dimitrova, V., Mizoguchi,R., du Boulay,B., Graesser, A.C. (eds.)AIED, pp. 197204. IOS Press,Amsterdam (2009)

[10] Piech, Chris, et al. "Deep knowledge tracing." Advances in Neural Information Processing Systems. 2015.

[11] Tang, Steven, Joshua C. Peterson, and Zachary A. Pardos. "Modelling Student Behavior using Granular Large Scale Action Data from a MOOC." arXiv preprint arXiv:1608.04789 (2016).

[12] Sutton, Richard S., and Andrew G. Barto.Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.