Approximately Optimal Teaching of Approximately Optimal Learners

Jacob Whitehill and Javier Movellan

Abstract—We propose a method of generating teaching policies for use in intelligent tutoring systems (ITS) for concept learning tasks [1], e.g., teaching students the meanings of words by showing images that exemplify their meanings à la Rosetta Stone [2] and Duo Lingo [3]. The approach is grounded in control theory and capitalizes on recent work by [4], [5] that frames the "teaching" problem as that of finding approximately optimal teaching policies for approximately optimal learners (AOTAOL). Our work expands on [4], [5] in several ways: (1) We develop a novel student model in which the teacher's actions can *partially* eliminate hypotheses about the curriculum. (2) With our student model, inference can be conducted *analytically* rather than numerically, thus allowing computationally efficient planning to optimize learning. (3) We develop a reinforcement learning-based hierarchical control technique that allows the teaching policy to search through *deeper* learning trajectories. We demonstrate our approach in a novel ITS for foreign language learning similar to Rosetta Stone and show that the automatically generated AOTAOL teaching policy performs favorably compared to two hand-crafted teaching policies.

Index Terms—Intelligent tutoring systems, stochastic optimal control, partially observable Markov decision processes

1 Introduction

Intelligent tutoring systems (ITS)—i.e., computers that can teach humans automatically—have been pursued by psychologists and computer scientists for over 50 years. The earliest ITS focused on simple "flashcard"-style instruction of foreign language vocabulary words [6], [7]. Nowadays, ITS can teach complex cognitive skills such as computer programming, geometry, and high school physics [8], [9], [10], [11]. Modern ITS employ graphics, videos, and animations [8] to engage their students, and a few systems even use web cameras and physiological sensors to perceive their students' affective states [8], [12]. To date, ITS in a variety of subjects have reached thousands of schools and benefited hundreds of thousands of students around the world [13].

Despite this success, ITS have not yet fulfilled their potential to impact modern education in a major way, and most students finish school with little to no contact with automated teaching systems. Before ITS can proliferate at a massive scale, two inter-related challenges must be overcome by the ITS research community: (1) how to reduce the labor of scaling up ITS to a wider range of learning domains, and (2) how to develop more powerful and more accurate models of how students learn, and integrate them into ITS to achieve higher learning gains.

- *J. Whitehill is with the Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA 01609. E-mail: jrwhitehill@wpi.edu.*
- *J. Movellan is with the Institute for Neural Computation, University of California, San Diego, La Jolla, CA 92093. E-mail: movellan@mplab.ucsd.edu.*

Manuscript received 21 Apr. 2016; revised 25 Jan. 2017; accepted 2 Apr. 2017. Date of publication 12 Apr. 2017; date of current version 18 June 2018.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TLT.2017.2692761

Scaling Up the Development of ITS. The effort involved in building an ITS—not just collecting the educational content, but in deciding how the automated tutor should serve that content to the student—can be formidable. When developing an ITS, the programmer must solve a fundamental *control problem*: what action should the ITS take, based on the history of interaction with the student, at each moment in time? For example, if the student makes a mistake while executing a problem step of a physics exercise, should the ITS give a hint after 5 seconds? After 10 seconds? Should it switch to an easier problem? Or should it simply give the student another chance to find the correct answer? While manually constructing rules for how to teach may work for very small teaching problems, that approach quickly becomes intractable as the complexity of the curriculum and the number of the available sensors (e.g., web camera, heart rate monitor, electrodermal sensors) increase.

Using More Accurate Learning Models. To date, most prominent ITS are based on Bower's binary all-or-none learning model [6], [7], [10], [14], [15]. Under the model, each skill is assumed to be either "learned" or "not learned". Although the teacher does not observe the student's state directly, he/she can infer the probability that the state is "learned" based on the history of the student's responses and the teacher's actions; this is known as Bayesian Knowledge Tracing [16]. Bower's model has two important limitations: (1) it does not allow for fine gradations of knowledge; and (2) it does not consider dependencies between the different concepts being learned. It is likely that ITS could be improved if more accurate student models were used. However, more complex student models increase the difficulty of finding good control policies that use those models. Moreover, the resulting teaching policies may be very complex and not amenable to real time implementations.

Control-Theoretic Approaches to Teaching. One approach to developing control policies for ITS that use more powerful learning models is to employ a computational framework such as stochastic optimal control theory. Under this framework, teaching can be seen as a Partially Observable Markov Decision Process (POMDP), in which an agent has to make decisions based on partial, uncertain information about the environment in order to achieve some goals. In the case of ITS, the agent is the intelligent tutor; the decisions are about how to teach (e.g., when to introduce a new concept to the student, when to give a hint), and the "environment" is the cognitive and emotional state of the student (e.g., how proficient he/she is in various skills, how he/she is feeling). The cost function might be the amount of time it takes the student to learn a set of skills to a performance criterion, or the fraction of quiz questions that a student answered incorrectly after 1 hour of training. The environment is considered "uncertain" because the agent cannot know with complete certainty what the student knows or how he/she feels. Instead, the tutor must infer the student's state based on the student's responses to test questions and practice problems, or perhaps an image of the student's face obtained through a web-camera or measurements from a physiological sensor. The potential benefit of POMDPs to ITS is that they can significantly reduce the burden of constructing teaching policies by hand: once the key POMDP variables that describe the student and tutor have been defined (see Section 4), a variety of algorithms exist for computing teaching strategies automatically.

The idea of modeling automated teaching as a stochastic optimal control problem has existed almost as long as the field of ITS itself—POMDPs were the computational backbone of the early ITS from the 1960s-70s (see Related Work section) [6], [7]. However, this early work focused mostly on *analytical* and *exact* solutions to small, highly constrained teaching problems using Bower's all-or-none student model. This greatly limited the utility of stochastic optimal control for teaching, and during the 1980s-1990s, the optimal control approach to building ITS languished. Since the 1990s, a variety of new algorithms have been developed by the machine learning and reinforcement learning communities [17], [18], [19], [20], which greatly expand the breadth of problems to which POMDPs can be usefully applied. Given these new tools, more research is needed on *how* exactly to use them effectively to find good control policies for ITS.

AOTAOL. One control-theoretic approach to creating automated teaching systems, which was recently proposed by [4], [5], models the student as an *approximately Bayesian learner* who uses information he/she receives from the teacher to make rational inferences about the concepts he/she is learning. This approach yields a learner model that is powerful (e.g., it can predict not just whether the student will make a mistake, but what *particular* mistakes the student will make), yet simple (e.g., it has relatively few parameters). Given this learner model, reinforcement learning algorithms are then used to find *approximately* optimal solutions to the teaching problem. We call this approach Approximately Optimal Teaching of Approximately Optimal Learners (AOTAOL).

In our paper we expand the previous work on AOTAOL [4], [5] to handle a more powerful student model, and develop a method of optimizing the teaching strategy more efficiently. We focus on the "Rosetta Stone" language learning problem [2]: the ITS teaches students the meanings of words in a foreign vocabulary by showing images that represent the words' meanings. For example, to teach the meaning of the Spanish word "roja", the tutor might show the student pictures of a red apple, a red chair, and a red car. The student's task is to infer that "roja" means "red". Besides being a popular teaching method for many adult learners, the Rosetta Stone approach may be useful for young children, who do not yet have a native language into which foreign words could be translated. Foreign language vocabulary learning is an important and difficult task for many people; it has been investigated by researchers since the early days of ITS [6], [7] and more recently has inspired competitions in industry such as Memrise [21]. Compared to conventional approaches in which the principal computational task is deciding which word to present to the student at each timestep, in the Rosetta Stone task we face the additional difficulty of deciding *how* to teach that word, i.e., which images to display to represent that word's meaning.

Our AOTAOL approach to building an ITS consists of three steps. (1) First, we develop a model of how students learn by assuming they update their knowledge state in an approximately optimal manner, based on the evidence they receive from the teacher. Under this model the student beliefs are represented as a continuous probability distribution which is updated using standard probabilistic inference rules. This approach has proven very successful at modeling perceptual and concept-learning tasks [1], [22]. (2) Next, we fit the parameters of the student model using data obtained from real students on the teaching task. (3) Finally, we develop a novel hierarchical control architecture based on policy gradient [20] for computing approximately optimal teaching policies that can be implemented in real time.

Key Contributions. The chief goal of our work is to explore whether the teaching policies automatically developed by our AOTAOL approach are competitive with respect to carefully hand-crafted teaching policies. Compared to prior work, our paper makes the following research contributions: (1) *Model fine gradations in student's knowledge*: In [4], [5], the student's belief updates are constrained so that, after every "teach" action executed by the teacher, the probability assigned to each concept is either set to 0 or left untouched. This simplifies the control problem but does not support subtle teaching actions whereby an example X "probably" represents concept A but "might possibly" represent concept B. Our student model is more flexible and allows the "teach" actions to *partially* eliminate hypotheses about a word's meaning (see Fig. 2). This flexibility is crucial to modeling the Rosetta Stone learning task (see Section 9). Moreover, in our model the learner's inference process can be conducted *analytically* (not just numerically), which accelerates the teacher's planning process. (2) *Deep policy search*: In [4], [5], the teaching action is selected at runtime by performing a shallow recursive forward look-ahead search of depth 2. However, shallow exploration discounts the value of "information-gathering" actions such as "ask" and "test" actions because their value may take several timesteps to be realized. It is thus useful to devise new approximate methods for searching deeper into the time horizon in real time. The policy gradient technique we proposed is one such method. (3) We illustrate our

approach by *developing an ITS for the "Rosetta Stone" language learning task*. We show empirically that this system—whose controller is computed automatically within the AOTAOL framework—is competitive when compared to two baseline controllers that were created by hand. Moreover, we compare the AOTAOL model to Bayesian Knowledge Tracing for the task of predicting students' test scores and explore the conditions under which AOTAOL does better.

2 Related Work

Some of the earliest intelligent tutoring systems were developed at Stanford University and focused on "flashcard"-style learning of foreign vocabulary words [6], [7] where the student was shown a foreign word in association with its meaning in her native language. The student in such systems was typically modeled as a two-state "all-or-none" learner formulated by Bower [14], and the role of the teacher was to execute actions so that the student exited the tutoring session in the "learned" state for each word with high probability. Interestingly, automated teaching in this form was one of the first control problems to be formalized as a Partially Observable Markov Decision Process [23]. However, the early research on optimal teaching focused on exact solutions, which are possible only in very small teaching problems using simple models of the student. By the mid 1970s, the optimal control approach to teaching mostly died off.

Around 1980, John Anderson and colleagues at Carnegie-Mellon University pioneered the "cognitive tutor" movement, based loosely on the ACT-R theory of cognition [24], [25]. The design of cognitive tutors focuses on how to deconstruct complex skills such as algebra into small "chunks" that can be learned independently or in simple sequences. The student's knowledge of the different chunks is effectively formulated as a stochastic filtering problem on top of Bower's all-or-none model. This process is called "knowledge tracing" in the ITS literature. The cognitive tutor movement gave rise to some of the most successful automated teaching systems to date, including the Algebra Tutor [13], Geometry Tutor, and LISP Tutor [11].

Since the early 2000s, the ITS community has experienced a "renaissance" of interest in applying principled probabilistic approaches to optimize the teaching process [4], [5], [26], [27], [28], [29], [30], [31], [32]. These developments are likely due to the significant progress made contemporaneously in the fields of machine learning and reinforcement learning. New reinforcement learning algorithms, including approximate methods for solving POMDPs such as belief compression [17], point-based value iteration [18], [19], and policy gradient approaches such as [20], may hold promise for developing automated teaching controllers. Some of these approaches are *model-free* and attempt to learn and improve the goodness of the teaching policy by sampling outcomes from real students in an online fashion—n-armed bandits are one example of this approach [33]. Other work, such as that of Rafferty et al. [4], [5] as well as our own paper, uses a *model-based* approach whereby a model of how the student learns is constructed explicitly, and the teaching policy can be optimized using simulation.

Finally, our work is also related to Bayesian models of human *concept learning* and *concept teaching* [1], [22], [34], [35], [36]. In particular, the Bayesian student model of our paper was inspired by the works of [1], [22], [35], and [36], which are concerned with modeling how a student infers concepts from examples.

3 Language Learning Task

Notation. We represent random variables with capital letters (e.g., Q_t, Y_t) and instantiated values of random variables with lower-case letters (e.g., q_t, y_t). We sometimes write $Y_1, \ldots, Y_t$ as $Y_{1:t}$.

In this paper we formalize the teaching problem as follows: a student is trying to learn a vocabulary of n words (e.g., *roja*), each of which can mean any one of m possible concepts (e.g., "red"); we refer to the words by their indices $\{1, \ldots, n\}$ and the concepts by their indices $\{1, \ldots, m\}$. The ground truth definition of each word j is known by the teacher and denoted by variable $W_j \in \{1, \ldots, m\}$; there is no restriction against synonyms (i.e., $W_j = W_{j'}, j \neq j'$ is permissible). There are different images (in set $\mathcal{Y}$) that the teacher might show the student. Each image can represent one of several possible concepts. For example, the left image in Fig. 2 could represent "man", "salad", "fruit", "eat", and possibly even "pink shirt", but it probably does not represent "office" or "wombat". In addition, an image can represent concepts with different probabilities—"eat" might be represented with probability 0.5, whereas "pink shirt" might be represented with probability 0.05. The student associates word j with concept i according to the "strength", as expressed by this probability, with which the image represents each concept i.

The teacher's task in our setting is to help the student learn the mapping from words to concepts as quickly as possible. At each timestep, the teacher can execute one of three different kinds of actions:

- *Teach word j*: the teacher shows the student an image y to teach the meaning of word j. We allow for both *positive* and *negative* examples—in a positive example, image y represents word j's meaning (i.e., the image represents the concept to which word j corresponds); in a negative example, image y does *not* represent the word's meaning. Teach actions cause the student to revise her belief about the meaning of word j.
- *Ask word j*: the teacher presents the student with two different images y_1 and y_2 and asks the student, "Which of the two images is more likely to represent word j?" The student then responds with either a 1 or 2 corresponding to her answer. Asking a question reduces the teacher's uncertainty about what the student knows.
- *Test*: the teacher gives the student a set of questions, each of which asks the student to select (from a list) the true meaning for a particular word j. If the student passes the test, then the learning session is over; otherwise, the learning session proceeds with the teacher's new knowledge of the student's beliefs. The "test" action both reduces the teacher's uncertainty, and also enables the student to graduate from the session; however, it is typically relatively expensive (in terms of time) compared to the "teach" and "ask" actions.

The teacher decides which action to execute at each timestep t based on a *policy*, which is a mapping from the

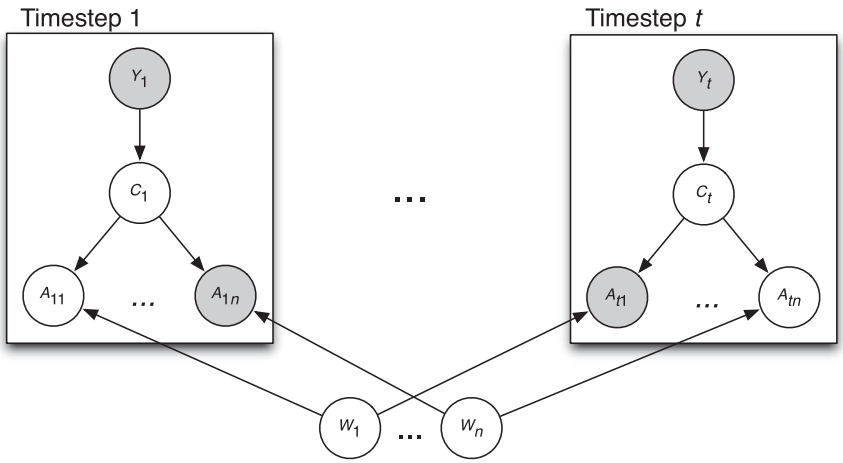


Fig. 1. Student model. Only shaded variables are observed by the student. $W_j \in \{1, \ldots, m\}$ is the definition of word j. Y_t is the image shown to the student at timestep t. $C_t \in \{1, \ldots, m\}$ is the concept represented by image Y_t. $A_{tj} \in \{0, 1\}$ is the "answer" of whether or not image Y_t represents concept W_j. The student's task is to infer the $W_1, \ldots, W_n$ based on the evidence (images and answers) provided by the teacher up through time t.

teacher's current belief of the student's state, to a probability distribution over the teacher's next action.

4 Teaching as a POMDP

We pose the "Rosetta Stone" learning task as a discrete-time, continuous-state, discrete-action POMDP. Formally, a POMDP consists of a state space $\mathcal{S}$, an action space $\mathcal{U}$, an observation space $\mathcal{O}$, a state transition dynamics model $P(s_{t+1} \mid s_t, u_t)$, an observation likelihood model $P(o_t \mid s_t, u_t)$, a time horizon T, and a real-valued cost function $c : \mathcal{U} \to \mathcal{R}$. The premise is that, at time t, the teacher chooses an action $U_t = u$ based on its control policy π. Executing this action then causes student to transition from state $S_t = s$ to $S_{t+1} = s'$, and also to output a response ("observation", from the teacher's perspective) $O_t = o$, e.g., the answer to a question posed by the teacher. The teacher uses both u and o to update its belief about the student's state, and it then chooses the next action U_{t+1}, etc. This process repeats until $t = T$. Associated with each action u is a real-valued cost, defined by cost function $c(u)$. An *optimal* policy π^* is one that minimizes the expected sum of costs from $t = 1$ to $t = T$

$$\pi^* \doteq \arg\min_{\pi} \mathbb{E}\left[\sum_{t=1}^{T} c(U_t) \mid \pi\right]. \tag{1}$$

In our setting, the state space consists of all possible beliefs that the student might have about the words' meanings, along with values for certain learning parameters of the student (described later); hence, the state space is uncountably infinite (compared with only 2 states in [14]). The observation space consists of all possible answers ($\{1, 2\}$ for "ask" actions, or $\{1, \ldots, m\}^n$ for "test" actions) that the student might give in response to a question or a test given by the teacher. The action space consists of all possible word + image combinations for "teach" actions, all possible word + image pair combinations for "ask" images, as well as a single "test" action. In the experiment we conduct in Section 9, this amounts to over 30,000 possible actions at every timestep. For our application, we define the cost function as the expected length of time (in seconds) needed to execute each action; these costs will be estimated empirically from data collected of human subjects.

5 Student Model

When devising the student model, our goals were to find a model in which (1) the student's learning process can be modeled as Bayesian inference; (2) fine-grained beliefs about the meaning of each word can be represented naturally; and (3) the belief updates can be computed efficiently. The model described below satisfies these criteria.

The student's task is to infer the values of variables $W_1, \ldots, W_n$ (the words' meanings) given the information she receives from the teacher—see the probabilistic graphical model shown in Fig. 1. Only shaded variables are observed by the student, and the other variables must be inferred, using Bayesian inference, based on the values of the observed variables. At each timestep t when the teacher executes a "teach" action, the teacher shows the student an image $Y_t \in \mathcal{Y}$. Variable $C_t \in \{1, \ldots, m\}$, which is not observed by the learner, represents *which single concept the student believes the image Y_t to represent*.[1] We denote the student's belief that an image $y \in \mathcal{Y}$ evokes concept i with the function $h : \{1, \ldots, m\} \times \mathcal{Y} \to [0, 1]$, where $\sum_i h(i \mid y) = 1$ for every y. We do not model the image pixels themselves; instead, h can be estimated by querying human subjects, e.g., on the Mechanical Turk (see Section 9).

In addition to showing an image Y_t, the teacher also reveals exactly *one* of the n variables $A_{t1}, \ldots, A_{tn}$. Each $A_{tj} \in \{0, 1\}$ represents the "answer" to whether or not image Y_t represents the meaning of word j (i.e., W_j). Specifically, we define $P(A_{tj} = 1 \mid C_t = i, W_j = i) = 1$ and $P(A_{tj} = 1 \mid C_t = i, W_j \neq i) = 0$. The word j for which the "answer" A_{tj} is revealed at timestep t is specified by variable $Q_t \in \{1, \ldots, n\}$ (which is represented in Fig. 1 *implicitly* by which particular answer variable is shaded at each timestep). Hence, at each timestep t, the learner observes two variables: Y_t and A_{tq_t}. For instance, if the teacher shows word 3 to the student at time $t = 6$ and says that the image does in fact represent word 3, then $Q_6 = 3$, and the student observes that variable $A_{6,3} = 1$. The other $n - 1$ "answer" variables $A_{t,j \neq q_t}$,

1. In our model, the student's belief about what each image represents does not necessarily have to correspond to "ground-truth". The teacher will simply use the student's belief in order to help the student to associate words with concepts.

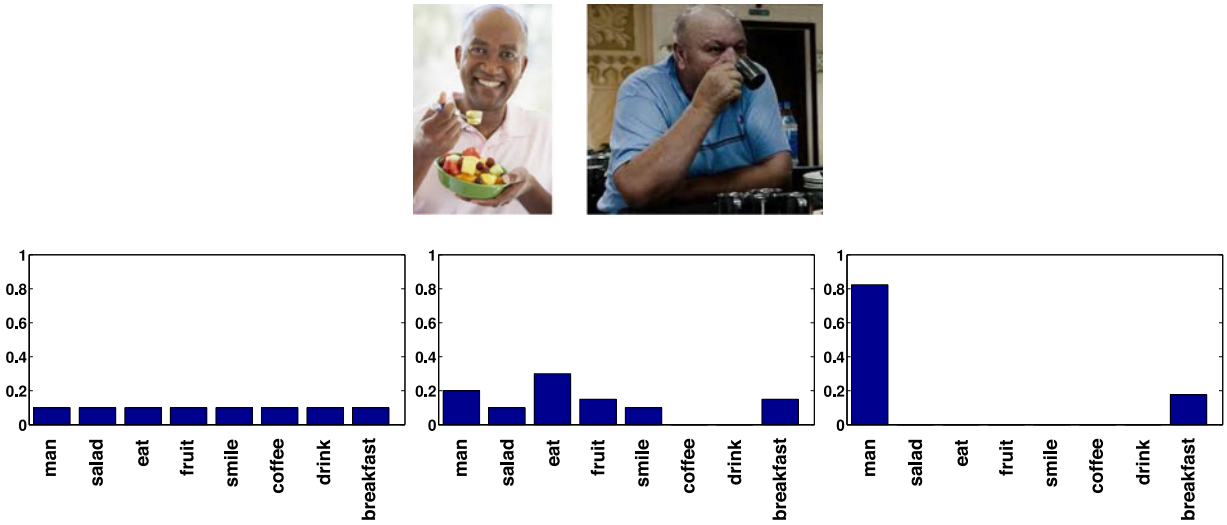


Fig. 2. *Top*: Two example images that could be used to teach the Hungarian word *férfi*. (*Bottom* from left to right): Prior belief $M_{1,1,\cdot}$ about the word *férfi*, posterior belief $M_{2,1,\cdot}$ about the word *férfi* after seeing the left image, and posterior belief $M_{3,1,\cdot}$ after seeing both images.

which represent whether the other words are represented by image Y_t, are not observed by the learner because the teacher reveals only one "answer" variable per timestep.

Based on all the images $Y_{1:t}$ and all the answers $A_{1q_1}, \ldots, A_{tq_t}$ that the student observes, she can infer the meanings of the words by updating her prior belief using Bayesian inference. We define M_t to be an $n \times m$ matrix specifying the student's belief at time t about the words' meanings, where entry M_{tji} specifies the student's posterior belief $P(W_j = i \mid y_{1:t}, a_{1q_1}, \ldots, a_{tq_t})$ that word j means concept i given the images and answers she has observed. Using probabilistic inference and exploiting the conditional-independence structure encoded in the graphical model (see Appendix for a derivation, which can be found on the IEEE Xplore Digital Library at https://ieeexplore.ieee.org/document/7895197/), the student updates her belief as

$$M_{t+1,ji} \propto M_{tji} h(i \mid y_t)^{a_{tj}} (1 - h(i \mid y_t))^{(1-a_{tj})}. \tag{2}$$

We note that the inference above can be performed *analytically*, which helps to accelerate policy optimization in the teacher (Section 6).

Example. Suppose a student is learning the meaning of (Hungarian) word *férfi* and that the set of possible concepts is { man, salad, eat, fruit, smile, coffee, drink, breakfast }. For simplicity we assume the student's prior belief over the word's meaning is uniform (see the left-most bar graph in Fig. 2). Now, suppose that at timestep $t = 1$ the teacher shows the student the image portrayed in the left image in Fig. 2 and says that *férfi* is represented by that image, i.e., $A_1 = 1$. The most prominent concepts represented by the left image (in our example) are "eat", followed by "man", and so on. After analyzing the image for its concepts, the student will then conduct inference based on Equation (2) and obtains the posterior belief about the meaning of *férfi* (middle graph of Fig. 2). At this point, the student is certain that *férfi* does not mean "coffee" or "drink"; however, there are still several candidate meanings, including "man", "salad", etc. If the teacher subsequently ($t = 2$) shows the right image in Fig. 2 and says that this image also represents *férfi*, then the student will update her belief again to obtain the distribution (right graph of Fig. 2). The student now believes that *férfi* probably means "man" but might possibly mean "breakfast".

5.1 Approximately Optimal Learning

The student model described above assumes that humans are "perfectly Bayesian" which is unrealistic [36]. In order to model students more accurately, we add two kinds of noise: *absorption* and *belief update strength*.

Absorption. Similar to [4], we allow for the possibility that sometimes the student does not "absorb" an example word + image pair. This could be, for example, because the student was too tired to process the example. If a student does not "absorb" the example at time t, then the student's belief does not change at that timestep, i.e., $M_{t+1,ji} = M_{tji}$ for all j. The student is modeled to absorb a "teach" action at time t with probability $\alpha_t \in (0, 1]$. Once the student has absorbed word-image pair (j, y), teaching that same word j with the same image y has no effect for the remainder of the teaching session. In our model, once a word-image pair is absorbed, it is never "forgotten". Retention and memory decay could, however, be addressed in more sophisticated models (see Section 11).

Belief Update Strength. Assuming the student absorbs a particular image at a particular timestep, the student will update her belief according to Equation (2). However, we assume that the student may only "partially" update her belief according to belief update strength parameter $\beta_t \in (0, 1]$

$$M_{t+1,ji} \propto M_{tji} \left[h(i \mid y)^{a_t} (1 - h(i \mid y)^{1-a_t}) \right]^{\beta_t}. \tag{3}$$

As $\beta_t \to 0$, the student updates her belief less and less.

Given these sources of noise, we define the student's "state" S_t at time t to consist of: the student's belief matrix M_t, the set of word-image pairs (j, y) that have been absorbed, the absorption probability α_t, and the belief update strength β_t.

5.2 Observation Model

The previous section described how the student updates her belief when the teacher executes "teach" actions. We must

also define how the student responds to "ask" and "test" actions.

"Ask". When the teacher executes an "ask" action at timestep t, it shows the student *two* images $Y_t^1, Y_t^2 \in \mathcal{Y}$, along with a word $Q_t = j$, and asks the student which of the two images *more probably* represents the meaning of word j. The student then gives a binary response $O_t \in \{1, 2\}$ to indicate which image (1 or 2) is the correct answer. Under the proposed student model, the probability that the first image represents concept W_j, given the student's belief M_t, is equal to $p^1 \doteq \sum_i M_{tji} h(i \mid y_t^1)$ Similarly, the probability that the second image represents concept W_j, given M_t, is $p^2 \doteq \sum_i M_{tji} h(i \mid y_t^2)$. We define the probability that the student answers with $O_t = 1$ as $\frac{1}{1+p^2/p^1}$.

"Test". "Test" actions consist of questions of the form, "Which of the concepts $\{1, \ldots, m\}$ is the true meaning of word j?" When answering each such question, we assume that the student selects concept i for word j with probability M_{tji}, where M_t is the student's belief matrix at the time when the test was given. Notice that, in contrast to [14], our model not only predicts whether or not the student will answer a test question correctly, but also predicts the specific word the student chooses if she is incorrect.

6 Teacher Model

We now consider the teaching problem from the perspective of the *teacher*. The teacher's goal is to execute "teach", "ask", and "test" actions in some sequence so as to help the student pass the test as quickly as possible. This implicitly requires that the student learn the words, i.e., that the student update her belief M_t to match the words' true definitions. The teacher knows the true meanings $W_1, \ldots, W_n$ of all the words. Although the teacher is assumed to know the *model* of the student, i.e., the process by which the student updates her beliefs when shown a sequence of word+image pairs, the teacher does not know the exact *state* of the student S_t. Instead, the teacher must make teaching decisions based on a control policy π and its *belief* B_t, which is the teacher's belief (a probability distribution) at time t over the student's state S_t, given the sequence of actions and observations through time $t-1$. Since the student's state S_t (see Section 5) is a continuous-valued "belief state", the teacher's state in our setting is a "belief about beliefs". Note that this teacher model is more powerful than the model in [14], in which the student's knowledge is modeled using a single real number for each item representing the probability that the student has learnt the item (all-or-none learning).

The teacher computes B_{t+1} (see section below) based on its prior belief B_t, the teacher's action U_t, and the student's response O_t. The teacher also knows how the student perceives concepts in images, i.e., the function h (see Section 5). The teacher's action U_t at time t contains multiple components depending on U_t's associated type. For "teach" actions, the teacher shows the student a particular "query" word $Q_t \in \{1, \ldots, n\}$ along with a particular image Y_t and an "answer" A_t. For "teach" actions, $U_t = [\text{teach}, Y_t, Q_t, A_t]$. For "ask" actions, the teacher presents the student with two images Y_t^1 and Y_t^2 along with a query word Q_t and asks the student which image more probably represents word Q_t. Hence, for "ask" actions, $U_t = [\text{ask}, Y_t^1, Y_t^2, Q_t]$. Finally, for "test" actions, the teacher asks the student about all n words' meanings. For "test" actions, $U_t = [\text{test}]$.

6.1 Representing and Updating B_t

After executing action U_t and receiving observation O_t from the student, the teacher computes its new belief B_{t+1} about the student's new state S_{t+1} according to

$$B_{t+1} \doteq \int P(s_{t+1} \mid s_t, u_t) P(o_t \mid s_t, u_t) B_t \, dS_t. \tag{4}$$

The first term in the integral represents the transition dynamics of the student's state, which is given by Equation (3). The second term is the observation likelihood model of the student (see Section 5.2). The third term is the teacher's prior belief. Computing the integral exactly is infeasible since there is no analytical solution. Instead, we use a particle filter with importance resampling [37] to represent this distribution approximately. Each particle stores a possible value for the student state S_t, along with a weight ω_p, such that the sum of the weights over all particles is 1. At each timestep, when the teacher executes action U_t and then receives an observation O_t from the student, each particle is updated according to Section 5 and reweighted according to how well it explains O_t according to the observation model in Section 5.2. The particle weights are then re-normalized.

6.2 The Teacher's Control Policy

At each timestep, the teacher acts according to a control policy π, which together with the teacher's current belief B_t dictates what the teacher's next action will be. In our implementation, π is a stochastic logistic policy parameterized by a weight vector for each possible action. The policy is selected (in Section 7) so as to (locally) minimize the expected cost of teaching, which we define as $\sum_{t=1}^{T} c(u_t)$ where $c(u_t)$ is the expected amount of time, in seconds, necessary to execute action u_t. We estimate c based on data from real students. We set $T = 500$; if the student passes the test before time T, then she enters a terminal state in which actions incur 0 cost.

7 Computing a Policy

Given the POMDP definition described above, the task is to compute a teaching policy. We note that standard POMDP solvers (e.g., zmdp [38]) are not applicable to our problem because the state space itself, not just the belief state, is continuous.

For our automated teacher, we employ a novel, two-tiered hierarchical control architecture. The macro-controller uses policy gradient [20] descent to optimize the policy parameters offline when computational constraints are less pressing. Instead of exhaustively searching through a tree exponentially large in the the planning horizon, policy gradient approaches sample many *linear* trajectories of length T (T is the POMDP time horizon). Hence, a policy optimized with policy gradient can choose actions whose benefit might not be realized until much further in the future.

7.1 Macro-Controller

The macro-controller decides which action type (teach, ask, test) to execute, as well as which particular word to teach/

ask, based on the teacher's current belief B_t. More precisely, the macro-controller uses features x_t computed from the particle distribution to compute a probability distribution π over actions. We define

$$\pi(x_t) \doteq P(U_t = u \mid x_t) \propto \exp(x_t^\top \theta_u),$$

where U_t is the action executed at time t, x_t is a vector encoding certain features of the teacher's particles at time t, (see below) and θ_u is a weight vector associated with action $u \in \mathcal{U}$, where $\mathcal{U}$ is the action space of the macro-controller. We define

$$\mathcal{U} = \{\text{test}, \text{teach}_1, \ldots, \text{teach}_n, \text{ask}_1, \ldots, \text{ask}_n\},$$

where n is the number of words being taught. For simplicity, we restrict the "teach" actions to only show positive examples for each word, i.e., A_t is always 1.

Feature vector x_t encodes how "good" (close to ground-truth) the teacher believes the student's belief to be, as well as how uncertain the teacher is about the student's belief. See Appendix for more details, available in the online supplemental material.

To compute the policy weight vectors, we use the REINFORCE [20] policy gradient technique. Since we have a model of the student, we can run simulations to estimate the value and gradient of a particular policy as parameterized by $\{\theta_u\}_{u \in \mathcal{U}}$. Whenever the macro-controller decides either to teach or ask about word j, it queries the *micro*-controller (see below) to determine the particular image(s) it should present for word j. For each gradient estimate, we ran 500 simulations of at most 500 timesteps (a simulation can end early if the simulated student passes the test) using 100 particles. The learning rate was set to 0.005, and the policy was optimized (with a starting point of $\theta_u = \mathbf{0}\ \forall u$) over 400 gradient descent steps.

At runtime, we compute x_t based on the particles expressing the teacher's belief B_t, and then we use policy π to compute the probability of each action. The particular action chosen by the teacher at that timestep is then sampled according to these probabilities.

7.2 Micro-Controller

Given the decision at time t to teach (or ask about) word j, the teacher needs a mechanism to select an image for j. For the micro-controller we use a one-step lookahead search based on a information-maximization heuristic. In particular, for the "teach" actions, the teacher selects image y so as to maximize the expected increase, from time t to $t+1$, in the "goodness" of the student's belief about word j. For "ask" actions, the teacher chooses y_1, y_2 so as to maximize the expected reduction in the teacher's "total uncertainty" from time t to $t+1$. See Appendix for details, available in the online supplemental material.

As an additional heuristic to decrease run-time cost, we "pruned down" the number of actions through which the micro-controller must search in the following manner: We used a simple "baseline" teaching policy (see Section 9) to conduct teaching simulations to explore many different belief states encountered by the teacher, and we collected the set of all actions chosen by the micro-controller at such encountered belief states. Once collected, the micro-controller only searches through only this pruned set. In practice, we found this strategy to be a useful trade-off between choosing good teaching actions and taking a small amount of time to do so.

7.3 Scalability

The run-time cost of optimizing the macro-controller is proportional to mnp, where n is the number of words, m is the number of concepts, and p is the number of particles in the particle filter. Training tends to accelerate (i.e., the run-time cost of each gradient descent step decreases) during optimization since the optimization objective is to minimize the amount of time until the student passes the test. For the small experiment ($n = 10, m = 11, p = 100$) described in this paper (see sections below), optimization on a single machine containing an 8-core CPU took roughly 8 hours. We can thus estimate that, to optimize a teacher for a vocabulary of $n = 100$ words and $m = 101$ corresponding concepts, roughly 800 machine-hours would be needed (but see note on parallelization below). To mitigate the computational cost for a large vocabulary consisting of thousands of words, a hierarchical approach to teaching might be employed whereby the vocabulary is partitioned into smaller subsets (on the order of 50 to 100 words per subset), and a separate teacher could be trained for each subset.

Parallelization. Many simulations runs are needed to estimate the value and gradient of each policy vector at each round of optimization. However, these can be parallelized with very little data communication between threads (only the policy vectors need to be sent to each processor, and only the scalar value or the gradient vector needs to be returned). Hence, they can be parallelized easily across a large-scale computing cluster, not just on a shared-memory machine.

In the micro-controller, the time complexity scales with the number of images (linearly for "teach" actions; quadratically for "ask" actions, which present pairs of images). The time cost can be reduced using the "pruning" mechanism described in Section 7.2.

8 Procedure to Train Automatic Teacher

Given the models of the student and teacher, the particle-based belief update mechanism, and the method of computing a teaching policy, we are now ready to put all the parts together and create an automated language teacher. The procedure comprises the following steps:

(1) Collect database of images $\mathcal{Y}$ with which to teach the words.
(2) Query human subjects on which concepts they believe are represented by the images, i.e., estimate $h(\cdot \mid y)$ for each image $y \in \mathcal{Y}$. Note that this does not require any teaching expertise and can be done using ordinary people crowdsourced from the web.
(3) Create a simple hand-crafted teaching policy. This teaching policy does not need to be extremely effective, as its sole purpose is for parameter estimation in the next step.
(4) Collect data from human subjects to estimate the time costs for "teach", "ask", and "test" actions, as well as prior distributions $P(\alpha), P(\beta)$ over student model parameters α and β.

Fig. 3. Some of the images used in our experiment to teach foreign words that mean "woman", "boy", "dog", "eat", etc.

(5) Train the AOTAOL teacher by executing gradient descent on policy parameters $\{\theta_u\}_{u \in \mathcal{U}}$ to locally minimize the expected teaching cost.

In the next section we present an experiment to validate the AOTAOL procedure.

9 Experiment

9.1 Methods

We evaluated the proposed AOTAOL approach in a Rosetta Stone-style concept learning task. Participants had to discover the meaning of 10 words of an artificial foreign language via images that illustrated the concepts they represented in that language: *duzetuzi* = man, *fota* = woman, *nokidono* = boy, *minainami* = girl, *pipesu* = dog, *mekizo* = cat, *xisaxepe* = bird, *botazi* = rabbit, *koto* = eat, and *notesabi* = drink. The concept set consisted of all of the 10 meanings listed above plus an additional "other" concept whose purpose was to account for everything else that participants might perceive in the images. Hence, there were $n = 10$ foreign words and $m = 11$ concepts in total. In our study, we decided to use "teach" actions that present either a single image or a pair of two images. We restricted the teaching actions to show only positive examples, i.e., A_t is always 1. In total, there were $(56 + 56 * 55/2) * 10 = 15{,}960$ possible "teach" actions at each round. There are a similar number of "ask" actions ($56 * 55/2 * 10 = 15{,}400$), as well as 1 test action. This results in an action space of well over 30,000 different possible actions at each timestep.

9.1.1 Apparatus

The experiment required first training the AOTAOL teacher as well as two other teaching strategies for comparison. Training of the AOTAOL teacher followed the procedure described in Section 8:

Step 1 - Collect Images

We used Google Image Search to collect a set $\mathcal{Y}$ of 56 images (see Fig. 3), each of which showed one of the humans/animals in the concept set that was either eating or drinking.

Step 2 - Query Human Subjects to Estimate h

We built a Javascript-based web client to ask human subjects "how strongly or weakly the image represents each of the listed concepts" on a continuous scale of 0-1. We then recruited four members of our laboratory to answer this question for all 56 images. Each distribution $h(\cdot \mid y)$ was estimated by averaging the subjects' responses and normalizing.

Step 3 - Create Baseline Teacher for Parameter Estimation

Given $\mathcal{Y}$ and h, we created a baseline teaching strategy by hand, which we call Hand Crafted 1 (HC1). Although the procedure in Section 8 does not require HC1 to be very effective, we took care to make HC1 a reasonable control condition for our experiment by using the h learnt above and optimizing over policy parameters.

HC1. At each timestep, word j is selected uniformly at random from the vocabulary. Then, to teach word j, the teacher randomly selects an image y with probability proportional to $h(W_j \mid y)$. Because h was estimated by querying human subjects (see above), the images chosen to represent words' meanings are very sensible. Every δ rounds, the HC1 teacher gives a test to the student; if the student passes (accuracy ≥ 90 percent), then the teaching session is over. Otherwise, teaching continues for another δ rounds, then the teacher gives another test, etc. We optimized δ in simulation and used $\delta = 35$.

Step 4 - Estimate Teaching Parameters

We conducted a "parameter estimation" experiment on 42 subjects from the Amazon Mechanical Turk in order to estimate the time costs of each action type as well as the distributions $P(\alpha), P(\beta)$. During this phase, subjects were taught using either HC1, or with an AOTAOL policy with manually selected (non-optimized) parameters. The estimated time costs were $c(\text{teach}) = 10.74$, $c(\text{ask}) = 7.53$, and $c(\text{test}) = 106.46$ seconds.

Step 5 - Train the AOTAOL Teacher

We used REINFORCE (see Section 7) to conduct policy gradient descent (400 iterations, $T = 500$, learning rate of

0.005) on the stochastic logistic policy's weight vectors to minimize the expected time to complete the task. We call the resulting policy the *AOTAOL* Policy.

For comparison, we created one additional hand-crafted teaching policy, which we call Hand Crafted 2 (*HC2*): This teacher utilizes the student's responses to the test questions and keeps track of how many times it has taught each word. At each round, it computes the set of words that have been taught fewer than ϵ times and chooses one of those words uniformly at random. For the chosen word, the HC2 then chooses an image with probability proportional to $h(W_j \mid y)$. As soon as it has determined that each of the n words has been taught ϵ times, it gives the student a test. If the student passes, then the session is over; otherwise, HC2 sets the counter of every word that the student answered incorrectly on the test to 0. It then resumes selecting a word that has been taught fewer than ϵ times. We optimized ϵ in simulation and used $\epsilon = 3$.

Test at Start of Teaching Session. In all three conditions (AOTAOL, HC1, and HC2), we programmed the teacher to give every student a test during her/his very first round. This enabled us to verify that each student came into the session with no prior knowledge of the vocabulary (see Section 9.4 below).

9.1.2 Procedure

We compared the three conditions AOTAOL, HC1, and HC2 in terms of teaching effectiveness. The experiment was conducted using a custom Javascript-based web client that executes actions according to the teaching policy of each subject's experimental condition. Subjects were assigned uniformly at random to the three conditions at the start of each learning session. At each timestep of the session, the program either *teaches* the student the meaning of a word by showing an image or image pair that represent its meaning; *asks* the student to indicate which of two images more probably represents a particular word's meaning; or *tests* the student on all the words by asking him/her the meaning of every word in the foreign vocabulary. Each learning session proceeded until the student passed the test (accuracy ≥ 90 percent). The *dependent variable* was the elapsed time between starting the learning session and passing the test. Because we were interested more in how students discover the words' meanings rather than how well students can remember them, we allowed students to use a "notepad" (a textbox in our software) during the learning session.

Note that an empirical comparison with Rafferty, et al.'s work [4], [5] is not straightforward because, in their teaching model, every "teach" action *completely* eliminates one or more hypotheses about a concept. However, in the Rosetta Stone problem, the "teach" actions often only *partially* eliminate hypotheses about a word's meaning. While the belief update equation in Rafferty, et al.'s model (Equation (3) in [5]) could conceivably be modified so that the posterior probabilities of individual hypotheses are attenuated but not set to 0, it is unclear how exactly each posterior probability would be computed.

9.2 Participants

Ninety (90) human subjects from the Amazon Mechanical Turk started and completed the task during August 2012.

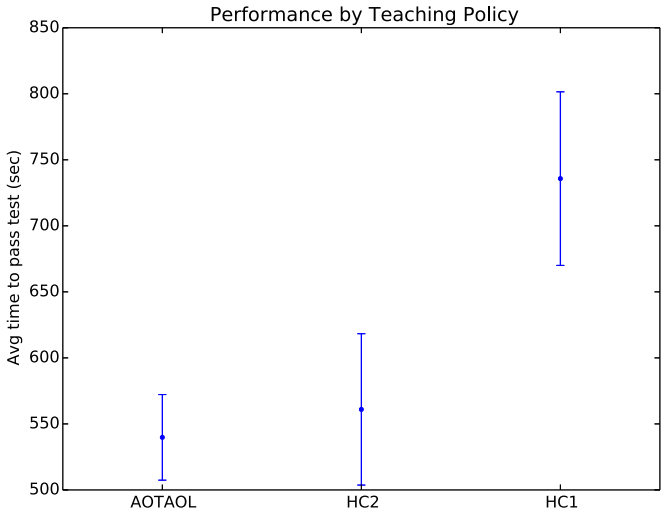


Fig. 4. Students' average time-to-completion (in seconds) for each teaching strategy (AOTAOL ($N = 29$), HC1 ($N = 26$), and HC2 ($N = 45$)) in the word learning experiment. Error bars show standard error of the mean of each group. The AOTAOL teacher delivered the best performance and statistically significantly outperformed HC1.

(An additional 92 subjects began, but did not finish, the task.) At the start of the experiment, each participant was randomly assigned (with equal probability) to one of the three conditions: AOTAOL ($N = 29$), HC1 ($N = 26$), and HC2 ($N = 45$). Subjects were paid \$0.15 for completing the experiment.

9.3 Results: Task Completion

The actual assignment probabilities (over all 182 participants who *started* the task) for the three experimental conditions were 0.357, 0.313, and 0.330 for AOTAOL, HC1, and HC2, respectively. The observed counts, across the three experimental conditions, of participants who *completed* the task (29, 26, and 45 for AOTAOL, HC1, and HC2, respectively), were statistically significantly different from the expected counts according to how conditions were assigned ($\chi^2(2) = 6.5576$, $p = 0.0377$). In particular, the teaching condition with the highest probability of task completion was HC2. One possible explanation is that participants prefer a teacher that teaches only those words that the participant had answered incorrectly on the previous test, rather than sometimes revisiting other words (as is possible with AOTAOL and HC1).

9.4 Results: Teaching Behavior & Effectiveness

Effectiveness. The average time ($\pm$ std.err.) to pass the test (in seconds) was 539.81 ± 32.41 for AOTAOL, 560.99 ± 57.31 for HC2, and 735.77 ± 65.72 for HC1—see Fig. 4. The teaching policy generated by the AOTAOL algorithm (Section 8) delivered the best performance—it was 24 percent faster ($t(53) = 2.7084$, $p < 0.01$, two-tailed) than the HC1 in helping students pass the test. AOTAOL was also 4 percent faster than HC2, though the difference was not statistically significant ($t(62) = -.30, p = 0.77$). Below we analyze the behavior of the different teaching policies and explore two reasons why AOTAOL did not significantly out-perform HC2:

Behavior. Over the 90 participants who completed the task, the average fractions of "teach", "ask", and "test" actions for the three teachers were: $(0.931, 0.005, 0.064)$,

$(0.953, 0, 0.047)$, and $(0.926, 0, 0.074)$ for AOTAOL, HC1, and HC2, respectively. By definition, neither HC1 nor HC2 ever executes an "ask" action (see Section 9.1.1), and even the AOTAOL teacher uses "ask" actions very rarely. This suggests both that the "ask" actions (in the manner they were implemented in this study) are not particularly useful, and that the advantage of the AOTAOL teacher compared to the other teachers must lie elsewhere. For example, even without using the "ask" actions, the AOTAOL teacher is able to select word+image combinations that are especially informative based on the *particular* confusions the student has about a word's meaning (as indicated by her/his responses to the test questions).

Impact of Outliers. Within both the HC1 and HC2 groups, but not in the AOTAOL group, there were several (7) students who passed the test that is given during the very first round of the teaching session—without ever being taught the meaning of a single word. Since the probability of passing the test just by guessing is exceedingly small, we believe that these students likely cheated, e.g., by pairing up with another student who had already deduced the words' meanings. When we exclude these seven participants from the experiment, the difference between the three conditions widens: The average time to pass the test (in seconds) becomes 764.00 ± 67.82 for HC1 and 630.76 ± 56.60 for HC2. Though the difference between HC2 (with outliers removed) and AOTAOL is still not significant, the p value is considerably smaller ($t(57) = -1.36, p = 0.18$).

Students Compensated for Bad Teaching Policies. Since the human subjects in the experiment were economically motivated to complete the task quickly, it is possible that they themselves compensated for poor teaching examples chosen by the teachers. For example, if the student already learned that the word *pipesu* means dog, and yet the automated teacher chose to show him/her even more examples to convey *pipesu*'s meaning, then perhaps the student would simply "click through" those examples quickly to reach more informative word+image combinations. To test this hypothesis, we computed the Pearson correlation for each subject between the time (in seconds) spent on each round in which the teacher executed a "teach" action, and the expected increase in the student's belief "goodness" (see Appendix, available in the online supplemental material) given by the chosen image+word combination during that round. The average correlation, computed over all 90 participants, was $r = 0.31$ and statistically significant ($p < 0.0001$). The average correlations *within* each of the three teaching conditions were also positive. In other words, students tended to spend *less* time on word+image examples that were not very informative. Moreover, the average increase in goodness of the "teach" actions across all students in each condition was highest for AOTAOL (0.20), second-highest for HC2 (0.17), and smallest for HC1 (0.13). Together, these results suggest that, although the AOTAOL teacher provided students with more informative teaching examples than the other teachers, the impact of these better-chosen examples may have been partially attenuated by students' own rational behavior.

9.5 Results: Prediction Accuracy

We also assessed how well the student model from Section 5 predicted students' test performance. In particular, for every student in our experiment, and for every timestep t when he/she took a test, we computed the expected test score (given by the observation model in Section 5.2) under the particle distribution given the history of actions and observations through time $t-1$. (For this analysis, we used 10,000 particles.) We then compared the predictions given by AOTAOL to predictions from a binary all-or-none student model [14] that is used for Bayesian Knowledge Tracing (BKT). Both models were trained on the same pilot data that were used to estimate the time costs of each action (see Section 8); these data spanned the HC1 and AOTAOL (albeit with different parameters) conditions but not HC2.

TABLE 1
Pearson Correlations between Predicted (Either Using AOTAOL or BKT) Test Scores and Actual Test Scores, Either Including All Tests or Excluding the First, for Each of the Three Experimental Conditions (AOTAOL, HC1, and HC2) or All Combined

	All Tests		Exclude 1st Test	
Condition	AOTAOL	BKT	AOTAOL	BKT
AOTAOL	0.96	0.96	0.26	−0.12
HC1	0.71	0.78	−0.15	0.11
HC2	0.70	0.71	0.23	−0.08
All	**0.79**	**0.81**	**0.19**	**−0.07**

Accuracy on All Tests. The model prediction accuracy on all 90 test subjects, as quantified by the Pearson correlation between the estimated and actual test scores for each student, were very similar for the two models: $r = 0.79$ for AOTAOL and 0.81 for BKT. Both correlations were statistically significant ($p < 0.0001$). Correlations between predicted and actual test scores, for both AOTAOL and BKT, were also very similar across the three experimental conditions. See Table 1.

Accuracy After Excluding First Test. As mentioned in Section 9.1.1, all students were given a test *immediately* after starting, and—since they had not yet had any opportunity to learn the words—their scores tended to be very low (around 10 percent). Since both the BKT and AOTAOL models can trivially predict this low score from the fact that no words had yet been taught, the tests given in the first round may inflate estimates of how accurate the models are. Hence, we performed a follow-up analysis in which the very first test for each student is excluded when computing the correlations. This makes the prediction problem much harder because it focuses on how accurately each model can distinguish fine-grained differences in knowledge (rather than between "know nothing" and "know everything"). After removing the first test for each student, the Pearson correlation of AOTAOL's predictions with students' actual test scores was $r = 0.19$ and statistically significant ($p = 0.03$; the corresponding RMSE was 0.275), whereas the correlation of BKT's predictions with actual test scores was negative ($r = -0.07$) and not statistically significant ($p = 0.40$; the corresponding RMSE was 0.197)—see Table 1. Across the three experimental conditions, the AOTAOL model's predictions were most accurate for participants in the AOTAOL teaching condition, though performance on HC2 participants was quite similar.

Interestingly, the correlations between BKT's predicted and the actual test scores *within* each student were quite

high—the average correlation over all students was $r = 0.7027$. This suggests that, although BKT is able to learn that students' scores tend to increase as they progress through the teaching session, the simple binary model has difficulty capturing fine differences between different students' knowledge.

10 Follow-Up Simulations

We conducted several simulation studies to explore issues related to optimization of the AOTAOL model.

10.1 Impact of Local Minima

To explore the severity of the problem of local minima during optimization, we conducted the gradient descent procedure described in Section 7.1, on the same vocabulary and set of concepts presented in Section 9, 24 different times. Even with the same starting point ($\theta_u = \mathbf{0}\ \forall u \in \mathcal{U}$), the cost (see Equation (1)) of the resulting policy is stochastic because each of the gradient and policy evaluations are based on stochastic simulations. The cost of each of the 24 optimized policies, after 400 gradient descent steps (learning rate of 0.001), was estimated as the median of 100 simulation runs. *Results*: A Kruskal-Wallis test confirmed that the costs of the 24 different policies were statistically significantly different ($H = 140.44, p < 0.0001$). However, the median absolute deviation between the individual costs of the 24 different policies and the grand median was relatively small—20.69 sec, or 2.5 percent of the grand median cost of 829.93 sec—suggesting that many local minima are similarly good.

10.2 Relative Importance of Macro- V. Micro-Controller

To explore the relative importance of the macro- versus the micro-controller on students' learning, we compared three policies in simulation: (1) The AOTAOL policy that was optimized and deployed in Section 9. (2) A policy with the same *micro*-controller as AOTAOL, but with each θ_u set to $\mathbf{0}$ so that macro-level actions are chosen uniformly at random. (Note that even this simple controller is not completely useless, as eventually all words will tend to be taught and the student will be able to pass the test.) (3) A policy with the same *macro*-controller as AOTAOL, but with a micro-controller that selects micro-level actions uniformly at randomly. *Results*: In simulation, the median costs of the teachers were: 638.93 sec for teacher (1); 1346.97 sec for teacher (2); and 825.87sec for teacher (3). In other words, there was a greater cost penalty, relative to the AOTAOL teacher (1), for ablating the macro-controller compared to the micro-controller.

10.3 Comparison with Recursive Look-Ahead Search

An alternative control architecture is to search at *runtime* for the best action in a recursive look-ahead search; this is the method used by [4]. In practice, we found that this method was too slow to be practical: for the experiment described in Section 9, the large branching factor (over 30,000 possible actions at each timestep) resulted in large search times even at very shallow depths: we found that, on an Intel i7 2.7 GHz processor, 16 particles in the particle filter (as in [4]), and averaged over 10 trials, a lookahead of one step took 0.014 sec, a lookahead of two steps took 4.511 sec, and a lookahead of hree steps took over 100 seconds (exact time unknown because the experiment did not complete). In contrast, after policy gradient training is complete, the average run-time cost of selecting the next teaching action using the two-staged approach described in Section 7 took only 0.035 seconds (averaged over 10 simulated teaching sessions of 100 timesteps each). Note also that while conducting look-ahead search *offline* can be done for a *fixed* set of states, the uncountably infinite state space of our teaching problem renders this approach infeasible as a general control strategy.

11 Extensions

Thus far we have described how an AOTAOL approach can be used to produce an automated teacher of foreign language in the manner of Rosetta Stone. Here, we describe how the language learning model proposed in Section 5 could be applied to other subject matter, and how it could be extended to model important phenomena such as spacing effects [39], [40] and affective states of the learner.

11.1 Different Subject Matter

The application focus of this paper has been on teaching simple nouns and verbs, but the exact same learner model proposed in Section 5 can also be used to teach more abstract linguistic concepts such as honorifics (e.g., the distinction between the formal pronoun *Sie* versus the informal *Du* in German), as well as literary devices such as irony, sarcasm, or satire. Outside of language, the student model can be used for a variety of concept learning and perceptual expertise tasks, including medical image diagnosis, species identification [41], mathematics, and more.

11.2 Spacing Effects

The learner model in Section 5 could be extended to account for memory decay over time, as well as for "spacing effects" [39], [40], [42], whereby memory retention at some future date can be improved by tuning the amount of time that transpires between subsequent learning events. All that is needed is an inference procedure to calculate the posterior probability of the student's knowledge state, given the learning trajectory so far, at each time t. Due to the modular architecture of AOTAOL-based systems (including our own and that of [4], [5]), this inference procedure can then be "plugged in" (in place of the analytical inference from Equation (2)), and the rest of the automated teaching architecture can remain unchanged.

11.3 Affective States

A key challenge for the ITS community today is to design automated teachers that can harness "affective" observations of the student—e.g., a web camera sensor that measures whether the student is engaged—to teach more effectively. With the AOTAOL approach, such observations can be incorporated seamlessly into the decision-making process. Consider a learning setting in which the student is either "engaged" or "not engaged". When the student is engaged, she processes each image+word combination fully, i.e., her belief update strength β_t is high. When a

student is not engaged, β_t is low. The teacher can estimate β_t through the student's responses to questions and tests. However, these "ask" and "test" actions may be relatively rare, and hence the teacher may be uncertain about β_t. By harnessing additional sensors, the teacher can estimate β_t more precisely and thereby teach more effectively. Specifically, suppose the teacher obtains an "affective observation" Z_t of the student's engagement level [43] via a web camera. Through standard probabilistic inference, the teacher would then compute a belief update using both O_t and Z_t (compare with Equation (4))

$$B_{t+1} = \int P(s_{t+1} \mid s_t, u_t) P(o_t \mid s_t, u_t) P(z_t \mid \beta_t) B_t \, dS_t.$$

The new term $P(z_t \mid \beta_t)$ constrains the possible states s_t that could explain the data observed by the teacher. This term emerges naturally from the model—no heuristic rules to handle Z_t are necessary.

12 Conclusion

We presented a framework—which we call Approximately Optimal Teaching of Approximately Optimal Learners—for automatically developing teaching policies for computer-based tutoring systems. The framework, which is based on modeling students as Bayesian learners [1], [22], [35], was recently proposed by Rafferty, et al. [4], [5]. Our work extends Rafferty, et al.'s approach in several ways: (1) It utilizes a student model that allows for teaching actions that only *partially* eliminate the student's beliefs about the words' meanings; this additional flexibility was necessary to model the Rosetta Stone learning task. (2) It allows analytical updates of core parts of the model, thus saving computation time. (3) It allows for deeper search through students' possible learning trajectories. We demonstrated our framework on the Rosetta Stone [2] problem wherein students learn foreign words via images that illustrate their meaning. From a computational point of view, this problem was very challenging: the automated teacher had to continuously update a probability distribution over the knowledge state of the student, which was itself a probability distribution over word meanings. Moreover, many teaching actions only *partially* eliminate hypotheses about a word's meaning—which is difficult to model using [4]. At every point in time, the teacher had to choose from more than 30,000 possible actions. An experiment on 90 human subjects showed that the teaching policy developed automatically under the AOTAOL framework performed favorably when compared to two carefully hand-crafted teaching policies.

Lessons Learned. Our work combines recent advances in AOTAOL, machine learning, and models of human learning to reduce the effort needed to develop successful teaching policies. It also reinforces the idea that students can be modeled as approximate Bayesian learners [1], [4], [22]. One advantage of these student models is that they are quite sophisticated when compared to previous student models, yet require relatively few parameters. Another advantage is that they enable new teaching policies to be computed automatically when important characteristics of the teaching problem change, e.g., a different vocabulary of words, or a different population of students with different prior knowledge of the words. We showed that AOTAOL models can be used not only to estimate the probability that the student will produce an incorrect response, but can also predict which *particular* mistakes the student will make (Section 5.2). More importantly, these models enabled the development of teaching policies that performed at least as well as two carefully hand-crafted teaching policies.

Scaling to Larger Teaching Problems. This paper considered only a very small-sized teaching problem consisting of 10 words and 11 concepts. Increasing the scale of the problem requires both more human effort—e.g., querying human subjects on which concepts they believe are represented by the images (Section 8)—and also more computational power. While the human effort in our work is modest and can easily be conducted via crowdsourcing on Mechanical Turk, the computational challenge of scaling up is considerable (see Section 7.3 on scalability). One strategy to overcome this challenge is to employ hierarchical control methods; the micro- versus macro-level control strategy that we implemented is one step in this direction.

Future Work. The fields of computer vision and machine perception have made very significant progress over the past 15 years to the point that accurate, automatic real-time perception of students' emotional and cognitive states is becoming possible [8], [12], [43]. One key challenge facing the ITS community is figuring out how to use these new sensors to teach more effectively. Designing teaching policies that take advantage of these sensors is a very complex problem and may not be amenable to traditional approaches. Computational frameworks such as AOTAOL may become increasingly important tools to help develop a new generation of ITS systems that take advantage of modern sensors.

References

[1] J. Tenenbaum, "Bayesian modeling of human concept learning," in *Proc. Advances Neural Inf. Process. Syst.*, 1999, pp. 59–68.

[2] Rosetta Stone, 2012. [Online]. Available: www.rosettastone.com

[3] Duolingo, 2012. [Online]. Available: duolingo.com

[4] A. Rafferty, E. Brunskill, T. Griffiths, and P. Shafto, "Faster teaching by POMDP planning," in *Proc. Int. Conf. Artif. Intell. Educ.*, 2011, pp. 280–287.

[5] A. N. Rafferty, E. Brunskill, T. L. Griffiths, and P. Shafto, "Faster teaching via POMDP planning," *Cognitive Sci.*, vol. 40, pp. 1290–1332, 2015.

[6] R. C. Atkinson, "Adaptive instructional systems: Some attempts to optimize the learning process" in *Human Memory and the Learning Process: Selected Papers of Richard C. Atkinson*. Moscow, Russia: Progress Publishing House, 1974.

[7] J. E. Matheson, *Optimum Teaching Procedures Derived from Mathematical Learning Models*. Stanford, CA, USA: Inst. Eng.-Economic Syst., Stanford Univ., 1964.

[8] B. Woolf, W. Burleson, I. Arroyo, T. Dragon, D. Cooper, and R. Picard, "Affect-aware tutors: Recognising and responding to student affect," *Int. J. Learn. Technol.*, vol. 4, no. 3, pp. 129–164, 2009.

[9] A. Gertner and K. VanLehn, "Andes: A coached problem solving environment for physics," in *Proc. 5th Int. Intell. Tutoring Syst.*, 2000, pp. 133–142.

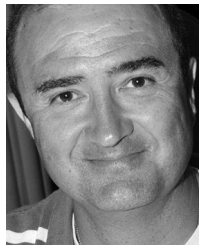
[10] A. T. Corbett and J. R. Anderson, "Student modeling and mastery learning in a computer-based programming tutor," in *Proc. 2nd Int. Conf. Intell. Tutoring Syst.*, 1992, pp. 413–420.

[11] J. R. Anderson, C. F. Boyle, and B. J. Reiser, "Intelligent tutoring systems," *Science*, vol. 228, no. 4698, pp. 456–462, 1985.

[12] S. D'Mello, R. Picard, and A. Graesser, "Towards an affect-sensitive AutoTutor," *IEEE Intell. Syst. Special Issue Intell. Educ. Syst.*, vol. 22, no. 4, pp. 53–61, Jul./Aug. 2007.

[13] K. R. Koedinger and J. R. Anderson, "Intelligent tutoring goes to school in the big city," *Int. J. Artif. Intell. Educ.*, vol. 8, pp. 30–43, 1997.
[14] G. H. Bower, "Application of a model to paired-associate learning," *Psychometrika*, vol. 26, no. 3, pp. 255–280, 1961.
[15] R. D. Smallwood, "The analysis of economic teaching strategies for a simple learning model," *J. Math. Psychology*, vol. 8, no. 2, pp. 285–301, 1971.
[16] A. Corbett and J. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Model. User-Adapted Interaction*, vol. 4, pp. 253–278, 1995.
[17] N. Roy, G. Gordon, and S. Thrun, "Finding approximate POMDP solutions through belief compression," *J. Artif. Intell. Res.*, vol. 23, pp. 1–40, 2005.
[18] M. T. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *J. Artif. Intell. Res.*, vol. 24, pp. 195–220, 2005.
[19] J. M. Porta, N. Vlassis, M. T. Spaan, and P. Poupart, "Point-based value iteration for continuous POMDPs," *J. Mach. Learn. Res.*, vol. 7, pp. 2329–2367, 2006.
[20] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.
[21] Memrise, "Memrise prize," 2015. [Online]. Available: http://www.memrise.com/prize/
[22] F. Xu and J. Tenenbaum, "Word learning as Bayesian inference," *Psychological Rev.*, vol. 114, pp. 245–272, 2007.
[23] R. Smallwood and E. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," *Operations Res.*, vol. 21, no. 5, pp. 1071–1088, 1973.
[24] J. R. Anderson, *The Architecture of Cognition*. Cambridge, MA, USA: Harvard Univ. Press, 1983.
[25] J. R. Anderson, *Rules of the Mind*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, 1993.
[26] Barnes and J. Stamper, "Toward automatic hint generation for logic proof tutoring using historical student data," in *Proc. 9th Int. Conf. Intell. Tutoring Syst.*, 2008, pp. 373–382.
[27] D. Fossati, B. D. Eugenio, S. Ohlsson, C. Brown, L. Chen, and D. Cosejo, "I learn from you, you learn from me: How to make iList learn from students," in *Proc. 14th Int. Conf. Artif. Intell. Educ.*, 2009, pp. 491–498.
[28] R. C. Murray, K. VanLehn, and J. Mostow, "Looking ahead to select tutorial actions: A decision-theoretic approach," *Int. J. Artif. Intell. Educ.*, vol. 14, pp. 235–278, 2004.
[29] M. Chi, K. VanLehn, and D. Litman, "Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics," in *Proc. Int. Conf. Intell. Tutoring Syst.*, 2010, pp. 224–234.
[30] D. T. Green, T. J. Walsh, P. R. Cohen, C. R. Beal, and Y.-H. Chang, "Gender differences and the value of choice in intelligent tutoring systems," in *Proc. Conf. User Model. Adaptation Personalization*, 2011, pp. 341–346.
[31] F. Khan, B. Mutlu, and X. Zhu, "How do humans teach: On curriculum learning and teaching dimension," in *Proc. Advances Neural Inf. Process. Syst.*, 2011, pp. 1449–1457.
[32] X. Zhu, "Machine teaching for Bayesian learners in the exponential family," in *Proc. Advances Neural Inf. Process. Syst.*, 2013, pp. 1905–1913.
[33] A. S. Lan and R. G. Baraniuk, "A contextual bandits framework for personalized learning action selection," in *Proc. 9th Int. Conf. Educ. Data Mining*, 2016, pp. 424–429.
[34] P. Shafto and N. Goodman, "Teaching games: Statistical sampling assumptions for learning in pedagogical situations," in *Proc. 30th Annu. Conf. Cognitive Sci. Soc.*, 2008, pp. 1632–1637.
[35] J. Nelson and J. Movellan, "Active inference in concept learning," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, 2001, pp. 381–384.
[36] J. Nelson and G. Cottrell, "A probabilistic model of eye movements in concept formation," *Neurocomputing*, vol. 70, pp. 2256–2272, 2007.
[37] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian Bayesian state estimation," *IEE Proc. F, Radar Signal Process.*, vol. 140, pp. 107–113, 1993.
[38] T. Smith and R. G. Simmons, "Heuristic search value iteration for POMDPs," in *Proc. Int. Conf. Uncertainty Artif. Intell.*, 2004, pp. 520–527.
[39] N. Cepeda, H. Pashler, E. Vul, J. Wixted, and D. Rohrer, "Distributed practice in verbal recall tasks: A review and quantitative synthesis," *Psychological Bulletin*, vol. 132, pp. 354–380, 2006.
[40] N. Cepeda, N. Coburn, D. Rohrer, J. Wixted, M. Mozer, and H. Pashler, "Optimizing distributed practice: Theoretical analysis and practical implications," *Expermental Psychology*, vol. 56, no. 4, pp. 236–46, 2009.
[41] P. Welinder, S. Branson, S. Belongie, and P. Perona, "The multidimensional wisdom of the crowds," in *Proc. Int. Conf. Advances Neural Inf. Process. Syst.*, 2010, pp. 2424–2432.
[42] M. Mozer, H. Pashler, N. Cepeda, R. Lindsey, and E. Vul, "Predicting the optimal spacing of study: A multiscale context model of memory," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 1321–1329.
[43] J. Whitehill, Z. Serpell, Y.-C. Lin, A. Foster, and J. R. Movellan, "The faces of engagement: Automatic recognition of student engagement from facial expressions," *IEEE Trans. Affect. Comput.*, vol. 5, no. 1, pp. 86–98, Jan.–Mar. 2014.

Jacob Whitehill received the BS degree from Stanford, the MSc degree from the University of the Western Cape, and the PhD from the University of California, San Diego. He is an assistant professor in the Computer Science Department, Worcester Polytechnic Institute (WPI). His research interests include machine learning, computer vision, human behavior recognition, and education. In 2012, he co-founded Emotient, a startup company for automatic emotion recognition.



Javier Movellan received the PhD degree from UC Berkeley. He is a research professor with UCSD, where he founded the Machine Perception Laboratory. His research interests include machine learning, machine perception, automatic analysis of human behavior, and social robots. Prior to his UCSD position, he was a Fulbright scholar with UC Berkeley. In 2012, he founded and became lead researcher at Emotient.