Learning User Models with Limited Reinforcement: An Adaptive Human-Robot Interaction System

Finale Doshi and Nicholas Roy
Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology,
32 Vassar St., Cambridge, MA 02139.
{finale|nickroy}@mit.edu

Abstract

Spoken dialog allows for natural human-robot interaction, but ambiguous phrases or noisy speech recognition can lead to considerable uncertainty during the conversation. Planning algorithms such as the Partially Observable Markov Decision Process (POMDP) have successfully overcome this uncertainty and generated reasonable interactions during natural dialogs between people and mobile robots. However, like all dialog systems, a POMDP is defined by a large number of parameters that may be difficult to specify a priori from domain knowledge. Even with an online adaptive system, learning these parameters may require a tedious training period from the user.

In this paper, we present an approach which lets the agent decide when it needs more information to be an effective dialog manager. If the agent feels that it is familiar with a situation, it acts based on its current understanding of the dialog. When faced with an unfamiliar scenario, the agent asks its human user what he or she would do in the agent's situation—advice that we believe is relatively easy for humans to give. Our approach both avoids a training period of constant questioning and allows the agent to discover the consequences of a poor decision without actually making mistakes. We demonstrate our approach both in simulation and on a dialog manager for a robotic wheelchair application.

Introduction

Spoken language allows for natural human-robot interaction, and the ability for a robot to take verbal commands can be especially useful when interacting with those who have limited mobility. The role of a dialog management system is to take dialog from a user—in our case, output from a voice recognition system—and interpret it to determine what action (if any) to take in response. In our work, we focus on a dialog manager for a robotic wheelchair (see Figure 1). The dialog manager's goal is to discover where the user wishes to go and command the wheelchair's navigation software to take the wheelchair to the desired location.

While navigating to a given location may seem to be a well-defined task, several factors make the dialog management challenging. First, the voice recognition system is often noisy—for example, the system may hear the words "coffee machine" when the user asks to go to "copy machine". Even with perfect voice recognition, ambiguities may occur when people use different names for the same location (such as "my desk" and "my office"). Users may also use the same word to refer to multiple locations (such as "elevator" when there are multiple elevators). Finally, to make decisions under uncertainty, the dialog manager must understand the user's preferences: How tolerant is the user of mistakes? How likely is the user to be frustrated by additional questions?

Figure 1: Our dialog manager allows for more natural human communication with a robotic wheelchair.

A good dialog manager must trade between asking questions to reduce its uncertainty (thus avoiding errors), and fulfilling the user's request within a reasonable amount of time. Partially Observable Markov Decision Processes (POMDPs) provide a theoretical framework for making decisions under uncertainty and have been successfully applied to dialog management situations. The ability to manage dialog uncertainty has made POMDPs attractive in assistive health-care (Roy et al., 2000; Hoey et al., 2005) and dialog management domains (Williams and Young, 2005; Litman et al., 2000), where the agent must reason about how to respond to user requests. Unfortunately, such real-world problems typically require a large number of parameters that are difficult to specify *a priori*.

One way to handle the problem of specifying the parameters corresponding to vocabulary, word error rate, user preference, etc. is to learn the model parameters online. In particular, we have shown previously that reinforcement

learning can be an effective way to learn dialog models online while interacting with users (Doshi and Roy, 2007a). Reinforcement learning is a form of learning in which the agent receives numeric feedback (or "reward") after every action. The agent adjusts its actions based on the feedback, and, over time, it learns how to maximize the reward it expects to receive.

While the reinforcement learning approach has been demonstrated in a wide variety of problems, including human-robot interaction (Litman et al., 2000), it has not met with widespread adoption in dialog management systems for several reasons. First, requiring the user to supply reward feedback after each action may be tedious, leading to frustration and inaccurate results. Second, in the reinforcement learning framework, the dialog manager will only learn about the consequences of a poor decision after making a mistake and experiencing a large negative reward. Experiencing a large penalty allows for rapid learning but can quickly lead to user dissatisfaction with the overall system. Finally, humans are notoriously bad at giving accurate numerical feedback which can cause the system to learn to do the wrong thing.

In this work, we present an alternative approach to online learning in human-robot interaction in which we learn a POMDP model online from data and use that model to derive a correct interaction strategy. By building an explicit model, the interaction agent can both assess its confidence in its own decision making and decide when additional training is needed. Instead of a reward signal after each interaction, we propose the concept of a meta-query, that is, a question about an action that the agent should take. These meta-queries take an intuitive form:

> "I think you definitely want me to go to the printer. Should I go to the printer?"

The agent uses these queries to learn about the user's preferences (for example, risk aversion) as well as discover information about their word choice and voice recognition noise. The agent asks a meta-query only if it is sufficiently confused about what action to take next. This active learning scheme limits the amount of feedback that is required, easing the training burden on the user. We show that such a system can adapt to users in a real robotic wheelchair application.

The remaining sections are organized as follows: Section I describes the basic POMDP dialog model and Section II describes how we incorporate the unknown model parameters into a larger POMDP. We present our algorithm in Section III and results in Section IV. Sections V and VI summarize our results and relate them to other work in POMDP model learning.

I. The POMDP Model

Formally, a POMDP consists of the n-tuple $\{S, A, O, T, \Omega, R, \gamma\}$. S, A, and O are sets of states, actions, and observations. In our wheelchair command-and-control scenario, the states represent locations to which the user may wish to go. The user's desired location cannot be directly observed and must be inferred from a set of noisy observations—in our case, keywords from a voice recognition system. The actions represent physical locations to which the wheelchair may drive, as well as questions that the wheelchair may ask the user. Figure 2 shows a cartoon of a simple dialog model.

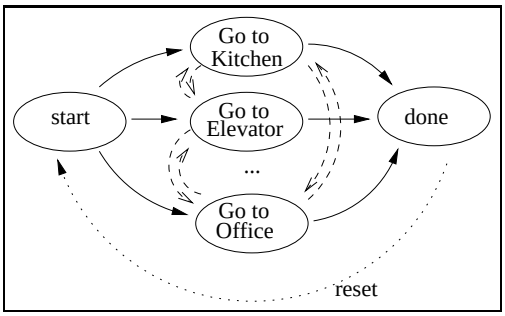


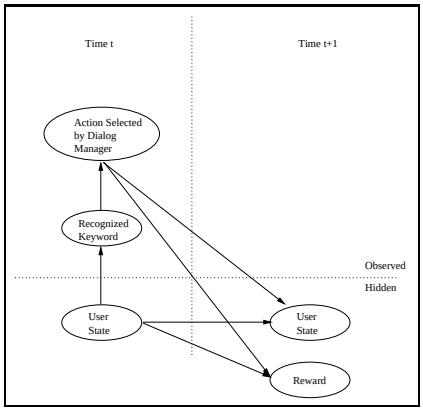
Figure 2: A toy example of a dialog POMDP. The nodes in the graph are different states of the dialog (i.e., user intents). Solid lines indicate likely transitions; we assume that the user is unlikely to change their intent before their original request is fulfilled. The system automatically resets once we reach the end state.

The transition function $T(s'|s,a)$ is a distribution over the states to which the agent may transition after taking action a from state s. Similarly, the observation function $\Omega(o|s,a)$ is a distribution over observations o that may be seen in state s after taking action a. If the observations are keywords, for example, the observation model might encode that the keyword "coffee" is commonly heard when the user wishes to go to the coffee machine. The reward function $R(s,a)$ specifies the agent's immediate reward for each state-action pair. In the wheelchair scenario, the agent may incur a small negative reward for asking a clarification question about where the user wishes to go. Similarly, it may incur a large penalty for taking the user to an incorrect location. Finally, the discount factor $\gamma \in [0,1)$ measures the relative importance of current and future rewards.

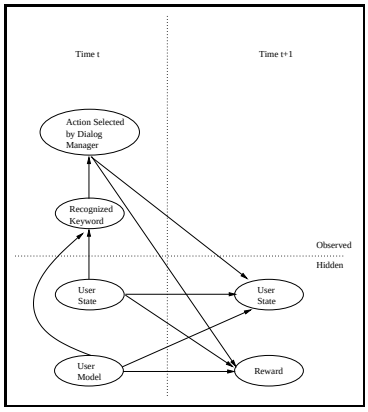
Since the true state—the user's intent—is hidden from the agent, it must choose actions based only on past actions and observations. In general, the optimal action to take now will depend on *all* prior actions and observations; however, keeping a history of the entire dialog to date can become quite cumbersome. Fortunately, it is sufficient to store only a distribution over possible user intents—known as a belief—as a sufficient statistic for the past history of actions and observations. If the agent takes action a and hears observation o from an initial belief b, we can easily update the belief using Bayes rule:

$$b^{a,o}(s) = \frac{\Omega(o|s',a)\sum_{s\in S} T(s'|s,a)b(s)}{\sum_{\sigma\in S}\Omega(o|\sigma,a)\sum_{s\in S} T(\sigma|s,a)b(s)} \quad (1)$$

If the agent has a set of POMDP model parameters that accurately describe the user, then it can simply solve the POMDP for the dialog management policy. The solution



(a) Standard POMDP model



(b) Model-Uncertainty POMDP

Figure 3: (a) The standard POMDP model. (b) The extended POMDP model. In both cases, the arrows show which parts of the model are affected by each other from time t to $t+1$. Not drawn are the dependencies from time $t+1$ onwards, such as the user state and user model's effect on the recognized keyword at time $t+1$.

to a dialog POMDP model is a policy that maps beliefs to actions. If the goal is to maximize the expected discounted reward, then the optimal policy can be found by solving the Bellman equations:

$$V^*(b) = \max_{a \in A} Q^*(b,a), \quad (2)$$

$$Q^*(b,a) = R(b,a) + \gamma \sum_{o \in O} \Omega(o|b,a) V^*(b^{a,o}), \quad (3)$$

where the optimal value function $V^*(b)$ is the expected discounted reward that an agent will receive if its current belief is b and $Q^*(b,a)$ is the value of taking action a in belief b. The optimal policy $\pi^* : P(S) \rightarrow A$ can be extracted from the value function using

$$\pi^* = \max_{a \in A} Q^*(b,a). \quad (4)$$

The exact solution to equation 3 is PSPACE-hard but point-based approximations (Pineau et al., 2003) can be used to find high quality solutions efficiently.

II. Modeling POMDP Uncertainty

The problem with using a POMDP to compute a dialog policy is that some of the individual model parameters $\{S, A, O, T, \Omega, R, \gamma\}$ are difficult to specify. It is reasonable to assume that the parameter sets S, A, and O are fixed and known beforehand. For example, in our dialog management task, S could represent all the places that a user may wish to go based on some map initially provided to the robot. The actions A can be pre-specified clarification questions or movements the wheelchair may take, and the observations O the keywords received from a voice recognition system. However, determining the parameters in T, Ω, and R is more difficult, as these parameters describe the user's preferences and the noise in voice recognition system.

However, just as the user's true intent is hidden from the agent, we can also represent the true parameters of the dialog model as hidden variables. We can therefore extend our basic dialog model by including the model parameters as part of the hidden state. We call this new representation a "model-uncertainty" POMDP in which the state space consists of both the user's intent and the true dialog parameters. In this new POMDP model, the state space becomes the set $\tilde{S} = S \times M$, where S is the user space as before, and M is the space of dialog models as described by all valid values for the model parameters. We note that the new state space $\tilde{S}$ is continuous and high dimensional.

Each state $\tilde{s}$ therefore describes a particular user intent s and a particular user model m. The model component m of the state contains the probability distribution describing how the user state s changes, as in the standard POMDP. The observations and rewards received for taking a particular action for a particular user intent now also depend on the hidden dialog model state. To generate policies tractably, we assume that the model component m itself is fixed, that is, the parameters of the user model do not change over time.

Figure 3(a) shows the standard POMDP process. The arrows in the graph show which parts of the model are affected by each other from time t to $t+1$, for instance, the reward at time t is a function of the state at the previous time and the action chosen by the dialog manager. The parameters defining this function are known *a priori* although every part of the model below the "hidden" line is not directly observed by the dialog manager and must be estimated on-line. In contrast, figure 3(b) shows the extended model. The reward at time t is still a function of the state at the previous time and the action chosen by the dialog manager, but the parameters are not known *a priori* and are therefore hidden model variables that must be estimated along with the user state.

Transition and Observation Uncertainty In the previous section, we introduced the belief as a distribution over possible user states. In the model-uncertainty representa-

tion, our belief is now a joint distribution over both the possible user states and the possible user model parameters. Just as we must specify an initial belief over user intents (for example, in Figure 2 we assume that we begin in a "start" state before the user has any intent), we must now specify an initial distribution over possible dialog models—a Bayesian prior on the models. The Bayesian approach is attractive in the dialog setting because we may have strong notions regarding certain parameters, but the exact values for the full set of parameters is typically difficult to specify. For example, we may not know the exact probability of hearing the word "coffee" if the user wants to go to the coffee machine, but we can guess it is probably high. Similarly, we can guess that there is a significant positive reward for driving to the right location and a significant negative reward for driving to the wrong location. We establish a prior distribution over the model parameters to express our domain knowledge, and improve the prior distribution with experience.

The need to represent the prior belief over models raises the question of how to represent this belief. The user state space is a discrete state space, so a standard histogram or multinomial distribution can be used. However, the model parameters such as the transition functions T are continuous parameters of distributions themselves; a distribution over T is effectively a distribution over distributions.

Since T and Ω are collections of multinomial distributions, the Dirichlet distribution is a natural choice of prior. The Dirichlet distribution places a probability measure over the "simplex" of valid multinomials. Figure 4 shows an example of such a simplex for a discrete random variable X where X can have three different outcomes with different probabilities, e.g., $p(X) = [0.25, 0.25, 0.5]$. Each value of $p(X)$ is a different point on the triangular simplex shown in figure 4 and the Dirichlet gives a measure of the likelihood of each such distribution. If $p(X)$ is in fact a transition probability distribution $p(X) = p(\cdot|s,a)$, then each possible transition probability distribution (i.e., each possible user model) is also some point on this simplex, with probability also described by the Dirichlet. As the agent's confidence in a particular model of user behavior increases, the probability mass of the Dirichlet distribution becomes increasingly concentrated around a single point.

Given a set of parameters $\alpha_1 ... \alpha_m$, the likelihood of the discrete probability distribution $p_1 ... p_m$ is given by

$$P(\underline{p};\underline{\alpha}) = \eta(\underline{\alpha}) \prod_i^m p_i^{\alpha_i - 1} \delta(1 - \sum_i^m p_i),$$

where η is a normalizing constant. The process for updating Dirichlet estimate of the multinomial given additional data is straight-forward. For example, suppose we are given a set of observation parameters $\alpha_1 ... \alpha_{|O|}$ corresponding to a particular *s*,*a*. If we observe observation o_i, then a Bayesian update produces new parameters $(\alpha_1, \ldots, \alpha_{i+1}, \ldots, \alpha_{|O|})$. Thus, we can think of quantity $\alpha_i - 1$ as a count of how many times observation o_i has been seen for the (*s*,*a*) pair. Initially, the expert can specify an educated guess of the distribution—which we take to be the mode of the distribution—and a pre-observation total that represents the expert's confidence in his guess.

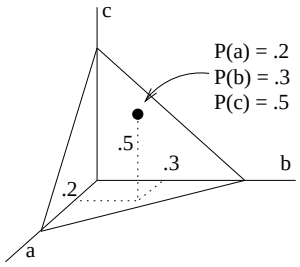


Figure 4: An example simplex for a multinomial that can take three different values (a,b,c). Each point on the simplex corresponds to a valid multinomial distribution; the Dirichlet distribution places a probability measure over this simplex.

Reward Uncertainty Next, we must specify a distribution over rewards. We fix a large positive reward value for driving the user to the correct location, and a small penalty for confirming the correct location with the user (for the minor inconvenience of having to communicate with the robot). These two reward values set a scale for the remaining reward values. We assume that the reward values are uniformly distributed between these ranges. The ranges are expert-specified initially, but the range shrinks as the model of user preferences becomes increasingly certain.

Passive Model Learning The Dirichlet transition, observation and uniform reward priors together specify a distribution over possible POMDP models. The agent can learn some information about the model through user interactions and improve the certainty of the model distribution. For example, suppose that the agent initially hears the word "printer," and user responds to the affirmative when the agent asks if the user wishes to go to printer. Then the agent can increase the probability that word "printer" is associated with the printer location. However, if the user responds to the negative, then the agent can infer that either the word "printer" is not associated with the location printer, or that printer is a commonly the output of a voice recognition error. Likewise, the agent can discover what are the most popular places where the user wishes to go (information about the transition model).

Active Model Learning Other information cannot be learned through user interactions. If the agent is only listening for location keywords, it cannot determine the user's frustration due to a poor action or repeated questions. One option would for the user to input such feedback into the agent; however, even from a small set of user tests in our lab, we found that it was often difficult to explain to users how to input reward values that would lead to the desired behavior from the wheelchair. Such training was also tedious. Thus, we introduced an additional ac-

tion to the dialog manager's options: the meta-query. For example, if the wheelchair is fairly certain that the user wishes to go to the printer, it might ask:

> "I think you definitely want me to go to the printer. Should I go to the printer?"

On the contrary, if the wheelchair thinks that the user may want to go to the printer but is not very certain, it might ask:

> "I think you may want me to go to the printer. Should I go to the printer?"

The choice of adverb gives the user an intuitive sense of the agent's uncertainty. Thus, the user can advise the robotic wheelchair based on their internal preferences. For example, if the user is risk averse, they may respond "yes" to the first question but "no" to the second question. If the user answers a question to the negative, the wheelchair might follow up with further questions such as,

> "In that case, I think I should confirm that you want to go to printer first. Is that correct?"

until it receives an affirmative response (assuming that the observation space has been augmented with yes/no keywords)[1]. These meta-queries are not perfect, since the user cannot know the true source of the wheelchair's confusion, but we believe they can provide a more natural way for the human to instruct the robot. We therefore add a set of meta-queries to the action set of the extended POMDP. Each meta-query has a fixed probability of a "yes" or "no" response for each model, which has the effect of changing the model component of the current belief. For simplicity, we fix the cost of each meta-query across all models.

III. Solving the Model-Uncertainty POMDP

Augmenting the original state space with the model parameters provides a principled way of thinking about the actions that result from uncertain dialog models. In Section IIIA, we validate our approach in simulation by solving this model-uncertainty dialog model directly when only a few discrete parameters are unknown. Unfortunately, the increase in the size of the state space also leads to computational intractability; in Section IIIB, we present an approximation that allows us to scale to real-world problems.

IIIA. Solving the Model-Uncertainty POMDP directly

In general, the parameters transition, observation, and reward functions are continuous-valued, with an infinite number of possible models. As such, the model-uncertainty POMDP is especially difficult to solve using standard methods. In special situations, however, uncertainty in the dialog model may be expressed as a small, discrete set of possible models rather than a continuous distribution, making the model-uncertainty POMDP much easier to solve.

[1] In our tests, we used an abbreviated form of the meta-queries for simulation speed.

Figure 5: Boxplot of dialog manager performance with a discrete set of four possible models. In this case, the user is very intolerant to errors, but the learner does not initially know this. Although the medians of the two policies are not so different, the active learner (right) makes fewer mistakes than the passive learner (left), leading to overall much less user-annoying behavior.

For example, consider a scenario where we already have accurate transition and observation models (say, from some prior work with the voice recognition system), but a new user's preference model is unknown. The user's exact preference model may not matter as long as the dialog manager has roughly the appropriate pattern of behavior. In an extreme case, we may decide to only characterize the user's frustration with an incorrect movement as *low* or *high*, and similarly characterize the user's frustration with an incorrect confirmation as *low* or *high*. The user model can be described by two variables $WrongMovePenalty$ $WrongQuestionPenalty$. The two variables $WrongMovePenalty$ and $WrongQuestionPenalty$ can each take either values of *high* or *low*, so that the model for a particular user might be $m =< WrongMovePenalty = high, WrongQuestionPenalty = low >$. This particular user would be conservative, with a preference to be asked questions repeatedly rather than risk being taken to the wrong location. With only four possible dialog models, the state space is still discrete and small, and we can now solve the model-uncertainty POMDP using a standard algorithm (Pineau et al., 2003).

We show simulated results with this very simple scenario of only four possible preference models in Figure 5[2]. The figure compares the performance of the policy without using meta-queries (left column) to the performance of the policy *with* meta-queries. As expected, the system which has the ability to ask meta-queries can use the questions to gain information about the user's internal preference model. It is able to discern that the user is very sensitive about incorrect movements, and therefore it asks more confirmation questions before taking an action. While the

[2] This work previously appeared in (Doshi and Roy, 2007b).

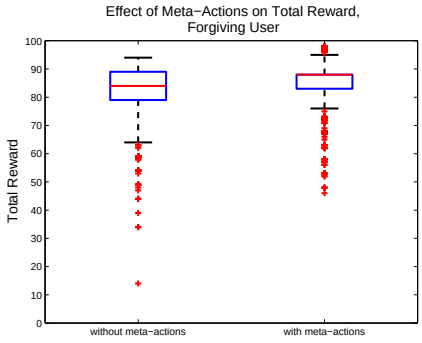


Figure 6: Boxplot of dialog manager performance with a discrete set of four possible models, lenient user. The effect is not as dramatic, but again, the learning dialog manager is able to adapt to the user's preferences and outperform the non-learner, especially in avoiding large mistakes.

difference in medians is not extreme, the reduction in large negative mistakes is substantial—which is particularly important in dialog management, where users will likely find a system that regularly makes mistakes annoying.

In Figure 6, we see similar improvements for the scenario where the user is fairly tolerant to mistakes. Again, the learning dialog manager outperforms the non-learner because it is able to determine the user's internal preference model and therefore ask fewer confirmation questions before acting.

Unfortunately, our approach of representing the user model as discrete values (such as $WrongQuestionPenalty = low$) does not scale well. Experimentally we found that even a modest increase in the number of possible user models from 4 to 48 meant that the model-uncertainty POMDP could no longer be solved using standard solution techniques. While it may be possible to group the possible combinations of user preferences into a few representative models (since the effects of small changes to the preference model may not be apparent to the user), discretizing other parts of the user model such as vocabulary choices quickly produces an exponential number of states. For example, for each keyword the user might utter, we have to consider how likely it is to be heard in each goal location. We therefore turn to approximation techniques which will allow us to represent a larger class of models with continuous parameters.

IIIB. Approximately Solving for a Dialog Policy

Instead of trying to solve for a dialog policy that incorporates both the uncertainty of the user model and the uncertainty of the user state, we separate the problem into two parts: first, we use the current belief over models to establish a representative set of candidate dialog models, and we solve for the optimal policy in each model. We then use these models to choose an action that has minimal risk; if the risk of all other actions is greater than the cost of asking a question for all possible models, we ask a meta-query to improve our estimate of the true user model and reduce the risk of errors. As we interact with the user, we update our collection of possible dialog models to reflect our changing belief about the user model. Table 1 outlines our approach for the continuous dialog parameter case.

Table 1: Dialog model learning approach using Bayes risk and meta-queries.

DIALOG MODEL LEARNING WITH BAYES RISK

- Sample POMDPs from a prior distribution over dialog models.
- Interact with the user:
  - Use the dialog model samples to compute the action with (approximately) minimum Bayes risk.
  - If the risk is larger than a given ε, perform a meta-query.
  - Update each dialog model sample's belief based on the observation received from the user.
- Periodically resample from an updated prior over dialog models.

Minimum Risk Action Selection If we know the correct user model m, then the optimal action to take (either confirming the user intent, or executing an action) is a_{opt}. Let us define a loss function $L(a, a_{opt})$, which describes the cost of taking a different action a. If we know the correct model m, we can solve the model and compute the value of each action a using a standard POMDP solution algorithm to solve equation 2. The loss function of can then be calculated as $Q(b,a) - Q(b,a_{opt})$, where a_{opt} is the optimal action.

We cannot calculate $L(a, a_{opt})$ since we do not know m, but we do have a belief $p_M(m)$ over models that allows us to calculate the expected loss $E_M[L]$. This expected loss is also known as the "Bayes risk":

$$BR(a) = \int_M (Q_m(b_m, a) - Q_m(b_m, a_{opt,m}))p_M(m), \quad (5)$$

where M is the space of dialog models, b_m is the current belief over possible user intents according to dialog model m, and $a_{opt,m}$ is the optimal action for the current belief b_m according to dialog model m. Let $a^* = \arg\max_{a \in A} BR(a)$ be the action with the least risk. If the risk $BR(a^*)$ is less than fixed cost of a meta-query, that is, if the least expected loss is still more than a certain threshold, we perform the meta-query, otherwise we choose the action a^*.

Intuitively, equation 5 computes the potential loss due to taking action a instead of the optimal action a_{opt} according to dialog model m and weights that loss by the probability of model m. When we are sufficiently sure about the model, the risk will be low; when we are unsure about the model, the risk may be high but the series of meta-queries will lead us to choose the correct action and avoid the risk. We unfortunately cannot solve equation 5

exactly because the integral is over the model parameters, and the solution would require us to solve for the value functions of an infinite number of POMDPs. However, we can use numerical techniques to find an approximation. Our belief over user states and user models gives us the probability of each model $p(m)$; if we draw sample models from this distribution, we will draw many samples in regions where $p(m)$ is high and few samples from where $p(m)$ is low. The more samples we draw, the better the densities of the samples will represent the distribution from which they were drawn. Thus, we can approximate equation 5 with the sum:

$$BR(a) = \sum_i (Q_i(b_i, a) - Q_i(b_i, a_{opt,i}))w_i, \qquad (6)$$

where Q_i provides the value of taking actions from belief states according to dialog sample i.

By drawing samples from the distribution $p(m)$, we are using the samples to approximate this distribution. However, the distribution over models will change as the wheelchair interacts with the user. The wheelchair must therefore periodically update the sample of dialog models that it is using to approximate its belief over models. If model samples are drawn from the current distribution over models, the weight w_i of each model is simply $\frac{1}{N}$, where N is the number of samples. However, for computational reasons—since we must solve every dialog model that we sample—it may be undesirable to resample models every time some new information changes our belief over possible models. In this case, the original sample set of models can be re-used by changing the weight of each model and representing the distribution $p(m)$ as a set of weighted samples. At each time step, the weight of each model should be adjusted to be proportional to the ratio of the previous likelihood of the sample and its likelihood given new information. While it is possible to provide formal bounds on the number of samples needed to approximate the Bayes risk to a specified degree of accuracy, these bounds are loose and in practice we found that fifteen samples sufficed for our dialog management application.

POMDP Resampling In some cases, the set of weighted samples may no longer accurately represent the true distribution over models, requiring a new set of sample models to be generated. The need for resampling may arise because one of the models becomes far more likely than the other dialog models in our sample set. If one model's weight w_i is close to 1, and the rest are close to 0, then the risk will appear to be quite small. This approximation is reasonable when the risk is truly small, but we do not want the dialog manager to become over-confident due to a poor set of candidate models. Another reason to resample models is that an interaction may have provided information that made all of the models in our current set very unlikely, and we would like our sample set to reflect our current belief over the dialog parameters.

We have two sources of information when it is time to update our sample set of dialog models. One source is the history of the most recent dialog, which consists of action-observation pairs $h = \{a, o\}$. Another source is the set of meta-queries $Q = \{(q, r, h')\}$, where h' is the history of the dialog from the initial belief to the query, q is the query, and r is the user's response to the query. Given h and Q, the posterior probability $p_{M|h,Q}$ over models is:

$$p_{M|h,Q}(m|h, Q) = \eta p(Q|m)p(h|m)p_M(m), \qquad (7)$$

where η is a normalizing constant. Note that if p_M is a Dirichlet distribution, then $\eta' p(h|m)p_M(m)$ is also a Dirichlet distribution since the likelihood $p(h|m)$ is product of multinomials. Recall that updating the Dirichlet distribution corresponded to adding counts—for example, if wheelchair observed the word "printer" after asking the user where he wished to go when the user truly wished to go to the printer, then we would add 1 to the Dirichlet parameter for hearing "printer" given a general query, true user goal is printer. The trouble is that we never know the true user state—we only have actions and observations.

Given a complete dialog, however, and assuming that it is unlikely that the user switched their objective in mid-dialog, it is possible to accurately infer the most likely underlying states from a history of actions and observations using the standard forward-backward algorithm. We can use the output of this algorithm to update the Dirichlet counts. This is a modified form of the standard Expectation-Maximization algorithm, and thus the prior will converge to some local optimal dialog model.

Incorporating meta-query information requires a different approach, since each specific meta-query response provides information about the dialog policy, not the dialog model parameters. We do not have a closed-form expression for $p_{M|h,Q}$, so we must use sampling to draw dialog model samples that are consistent with all of the meta-queries that have been asked so far. Each query in the set Q provides a constraint on the feasible set of dialog models M. Dialog models are feasible if their policy is consistent with the responses in the meta-query.

Computing this feasible set directly is intractable, however, given the set Q, we can check if a sampled dialog POMDP is consistent with the previous meta-query responses stored in Q. Thus, to sample POMDPs, we first sample dialog POMDPs from the updated Dirichlet priors. Next, we solve for the optimal policy of each model (which can be done quickly, since each dialog model sample is discrete and relatively small) and check if each dialog model's policy is consistent with the previous meta-query responses stored in Q.

IV. Results

Simulation Figure 7 shows results from a simulated dialog manager for our wheelchair application. The states consisted of locations where the user wanted to go, and the observations consisted of keywords extracted from utterances. Actions included open-ended questions, confirming a particular state, and driving to a particular location. Above, we see the usefulness of the Bayes-risk approach (compared to stochastic actions selection based on the weights of the sampled models) when the reward

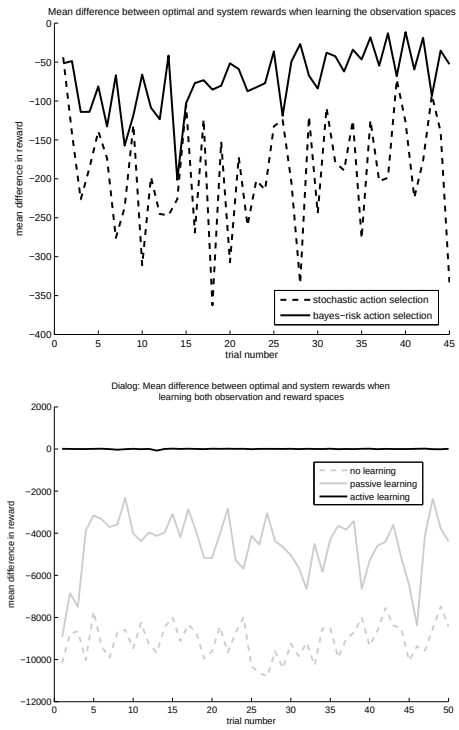


Figure 7: Dialog manager simulation results. Top: results from learning only the observation model. Bottom: benefits of active learning when learning both the observation and reward model.

model is known. In this case, the Bayes risk action selection allows us to choose non-risky actions.

The usefulness of our approach is even more dramatic when the reward prior is uninformative (below). In this case, the dialog manager can improve somewhat by passively updating its priors based on what it has heard (solid gray line). However, simply listening cannot provide the dialog manager information about the user's preferences. Moreover, since the active learning system asks the user for help whenever it is confused, this system does not suffer from an initial dip in performance before the model estimate converges to the true model. The meta-queries allow the active learner to learn while avoiding mistakes, thus maintaining a high level of performance.

Robotic Wheelchair We also validated our approach on a dialog manager for a robotic wheelchair with a simple user study. The underlying POMDP, with 10 states, 38 observations, and 21 actions, used keywords from a voice recognition system output as observations. Initially, each state had one strongly mapped observation (such as 'printer' for the printer location). The remaining observations received uniform initial priors. Four users conducted 12-15 interactions (20-25 minutes) with the system.

By asking meta-queries, the dialog manager was able to successfully complete all 57 interactions without making a serious error, that is, trying to drive the user to an incorrect location. Table 2 shows that the proportion of dialogs with meta-queries decreased significantly from the

Table 2: Proportion of dialogs with meta-queries by location. The decrease in the number of meta-queries from the second to the third time the location was asked for is statistically significant at the $p = 0.05$ level.

First half of interactions	.79
Second half of interactions	.48

Table 3: Proportion of dialogs with meta-queries. The decrease in the proportion of queries is significant at the $p = 0.05$ level.

How often place was requested	First time	Second time	Third Time
Total number of requests	29	15	8
Number of requests with meta-queries	22	11	2
Proportion of requests with meta-queries	.76	.73	.25

user's initial interactions with the system to his or her final interactions with the system. The proportions are relatively high because the users, being new with the system, asked for a variety of locations or experimented with different vocabulary. In Table 3, we see that the system often asked a meta-query the first or second time a user asked to go to a particular location, but there was a significant drop in the number of meta-queries after the third time the user requested to go to the same place. We note that our active-learning approach allowed our dialog manager to automatically determine the length of its training phase. Our trials were too small to draw strong conclusions from these results (especially since we were unable to account for the effect of the novelty factor in the interactions), but our quantitative analysis does show statistically significant decreases in the number of meta-queries asked as the dialog manager adapts to each user.

Next we present some sample dialogs to illustrate our system's learning. Table 4 contains a sample dialog where the system used a meta-query to establish the meaning of a new word. In the initial dialog, the word 'forecast' is new to the system, that is, it has no prior on what that word may mean. The system determines that the Bayes risk is great enough to warrant asking for a policy clarification. As a result, future dialogs are biased toward choosing POMDP samples that satisfy the initial policy query request, which allows later dialogs to avoid the same kind of training.

In Table 5, the system adapts to a situation where two words, 'coffee' and 'copy' are often mistaken for each other in the voice recognition system. We stress that at the beginning of the dialog, the system was initialized to believe that the word 'coffee' was most likely if the user wished to go to the coffee machine, while the word 'copy' was most likely if the user wished to go to the copy machine. The other key words—"tea" and "kitchen"—-in this segment began with uniform priors.

During the first conversation, the agent is confused by the many new words. Even after asking a meta-query,

Early Conversation
ROBOT: How can I help you?
USER: Give me the forecast.
ROBOT: I'm confused. What action should I take now?
The system pops up a meta-query window, and user indicates action "show weather" is correct.
ROBOT: Showers.
Later Conversation (with the same user)
ROBOT: How can I help you?
USER: What's the forecast for today?
ROBOT: Do you want the weather?
The system has partially the word "forecast" and confirms the correct state.
USER: Yup.
ROBOT: Showers.

Table 4: Learning a new word: The system learns the meaning of a new word through a meta-query.

it is confused by hearing the word "copy" since the user seems to want to go to the kitchen. After several interactions, the agent goes to the correct location. We note that even in this tricky scenario, the use of the meta-queries helped the agent focus on to the kitchen when the only word it recognized—"copy"—seemed to indicate that the user wished to go to another location. In the later dialog, we see the effect of the learning: upon hearing the words "kitchen" and "copy," the agent correctly infers that the user wishes to drive to the kitchen.

V. Discussion and Related Work

Prior work in POMDP learning has also considered incorporating the parameters of the POMDP as additional hidden states into a larger model-uncertainty POMDP. One approach, the Beetle algorithm (Poupart et al., 2006) solves the problem analytically by casting an MDP with unknown parameters as a POMDP. However, this approach would be difficult to extend to the POMDP case as the state space would now be doubly continuous and very high dimensional. Note that our goal is policy learning, rather than explicitly learning a vocabulary as in (Lopes and Teixeira, 2000; Lopes and Chauhan, 2007). We learn additional word models only as a function of how they affect the policy.

An approach that is closer to ours is the Medusa algorithm (Jaulmes et al., 2005). The Medusa algorithm also samples POMDPs from a prior over model parameters, but the actions are chosen stochastically and queries are made to a state oracle, not for policy information. While the Medusa approach can allow for faster learning, our action selection criterion is more robust (something valuable for dialog management applications with a robotic wheelchair) and our queries provide a more natural way for the user to provide feedback to the robot. To our knowledge, this is the first example of using Bayesian reinforcement learning on a real-world application.

Early Conversation
ROBOT: How can I help you?
USER: I want some tea from the kitchen.
ROBOT: How can I help you?
USER: The tea is in the kitchen near the coffee machine.
The system heard "copy" instead of "coffee" and is confused about the new word "kitchen."
ROBOT: How can I help you?
USER: Go to the kitchen.
ROBOT: I'm confused. What action should I take now?
The user indicates "Confirm the coffee machine area" is the correct action.
ROBOT: Do you want to go to the coffee machine area?
USER: Yup.
… *The system asks a few more (non-meta) questions before re-confirming the coffee machine state.*
ROBOT: Do you want to go to the coffee machine area?
USER: Yup.
ROBOT: Going to the coffee machine area.
Later Conversation (with same user)
ROBOT: How can I help you?
USER: Take me to the kitchen.
The system hears a noisy combination of "nick" and "kitchen" (mostly "kitchen").
ROBOT: How can I help you?
USER: I want some coffee.
The system hears "copy" instead of "coffee," but with its learned observation model, it confirms correctly.
ROBOT: Do you want to go to the coffee machine area?
USER: Yup.
ROBOT: Going to the coffee machine area.

Table 5: Learning about observation noise: the system learns new words for a noisy pair of states.

VI. Conclusion

We developed an approach for active learning for dialog managers that robustly adapts to different users. Meta-queries—questions about actions that the agent is thinking of taking—and a risk-averse action selection criterion allowed our agent to behave robustly even when its knowledge of the POMDP model was uncertain. Moreover, we demonstrated that this model could be successfully transferred to a dialog manager on a robotic wheelchair.

References

Doshi, F. and Roy, N. (2007a). Efficient model learning for dialog management. In *Proceedings of Human-Robot Interaction (HRI 2007)*, Washington, DC.

Doshi, F. and Roy, N. (2007b). Efficient model learning for dialog management. In *Technical Report SS-07-07*, Palo Alto, CA. AAAI Press.

Hoey, J., Poupart, P., Boutilier, C., and Mihailidis, A. (2005). Pomdp models for assistive technology. *IATSL Technical Report*.

Jaulmes, R., Pineau, J., and Precup, D. (2005). Learning in non-stationary partially observable markov decision processes. ECML Workshop.

Litman, D., Singh, S., Kearns, M., and Walker, M. (2000). NJFun: a reinforcement learning spoken dialogue

system. In *Proceedings of the ANLP/NAACL 2000 Workshop on Conversational Systems*, Seattle.

Lopes, L. S. and Chauhan, A. (2007). How many words can my robot learn? an approach and experiments with one-class learning. *Interaction Studies*, 8(1):53–81.

Lopes, L. S. and Teixeira, A. (2000). Human-robot interaction through spoken language dialogue. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 528–534.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for pomdps. *IJCAI*.

Poupart, P., Vlassis, N., Hoey, J., and Regan, K. (2006). An analytic solution to discrete bayesian reinforcement learning. In *ICML*, pages 697–704, New York, NY, USA. ACM Press.

Roy, N., Pineau, J., and Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the ACL*, Hong Kong.

Williams, J. and Young, S. (2005). Scaling up pomdps for dialogue management: The "summary pomdp" method. In *Proceedings of the IEEE ASRU Workshop*.