interspeech 2008 incorporating SST 08

The best of both worlds: Unifying conventional dialog systems and POMDPs

Jason D. Williams

AT&T Labs – Research, Shannon Laboratory, 180 Park Ave., Florham Park, NJ 07932, USA

jdw@research.att.com

Abstract

Partially observable Markov decision processes (POMDPs) and conventional design practices offer two very different but complementary approaches to building spoken dialog systems. Whereas conventional manual design readily incorporates business rules, domain knowledge, and contextually appropriate system language, POMDPs employ optimization to produce more detailed dialog plans and better robustness to speech recognition errors. In this paper we propose a novel method for integrating these two approaches, capturing both of their strengths. The POMDP and conventional dialog manager run in parallel; the conventional dialog manager nominates a *set* of one *or more* actions, and the POMDP chooses the optimal action. Experiments using a real dialog system confirm that this unified architecture yields better performance than using a conventional dialog manager alone, and also demonstrate an improvement in optimization speed and reliability vs. a pure POMDP.

Index Terms: dialogue modelling, dialogue management, spoken dialogue systems, partially observable Markov decision process, planning under uncertainty

10.21437/Interspeech.2008-355

1. Introduction

Conventional dialog design practices are well-established in industry. A developer creates a detailed dialog plan by hand, encoding their knowledge of the task and business rules. For example, a system can be designed so that passwords are verified before account access is granted. In addition, the dialog designer can craft each prompt to the current dialog context, and this is important because prompt wording has a strong effect on the user's speech patterns and satisfaction. This conventional approach is well-documented [1] and has been used to develop hundreds of successful commercial dialog systems.

Even so, speech recognition technology remains imperfect: speech recognition errors are common and undermine dialog systems. To tackle this, the research community has begun applying partially observable Markov decision processes (POMDPs) to dialog control. POMDPs depart from conventional practices in two ways: first, rather than maintaining a single hypotheses for the dialog state, they maintain a distribution over *many* hypotheses for the correct dialog state. Second, POMDPs choose actions using an optimization process, in which a developer specifies high-level goals and the optimization works out the detailed dialog plan. Because of these innovations, POMDP-based dialog systems have, in research settings, shown more robustness to speech recognition errors, yielding shorter dialogs with higher task completion rates [2, 3].

However, in the classical POMDP formulation, the optimization process is free to choose any action at any time. As a result, there is no obvious way to incorporate domain knowledge or constraints such as business rules. For example, it is obvious that the system should never print a ticket before it has asked for the origin city, but there is no direct way to communicate this to the optimization process. At best, obvious domain properties will be needlessly re-discovered. At worst, spurious actions will be taken with real users, an especially serious concern if POMDP-based systems are going to handle financial or medical transactions. Moreover, actions are drawn from a prespecified set, and so it is unclear how prompt wordings can be tailored to dialog context. Despite the POMDP's increased robustness to errors, these issues form important barriers to adoption in industry.

In this paper, we seek "the best of both worlds": to combine the robustness of the POMDP with the developer control afforded in conventional approaches. In our method, the (conventional) dialog manager and POMDP run in parallel, but the dialog manager is augmented so that it outputs one *or more* allowed actions at each time-step. The POMDP then chooses the best action *from this limited set*. Results from a real voice dialer application show that adding the POMDP machinery to a standard dialog system yields a significant improvement. Moreover, because the set of policies considered by the optimization is informed by the developer, spurious action choices are pruned, and optimization runs faster and more reliably than in a classical POMDP.

Although past work has combined conventional dialog managers with a fully-observable Markov decision process (MDP) [4, 5], the extension from an MDP to a POMDP is non-trivial: whereas the MDP state is a simple function of the dialog state, the POMDP belief state is decoupled from the conventional dialog state. Other past work has proposed using multiple POMDPs and selecting actions among these using hand-crafted rules [3]. By contrast, the method here allows an arbitrary conventional dialog manager to be used.

In the remainder of this paper, Section 2 reviews the conventional and POMDP approaches, Section 3 explains the method, Section 4 presents the voice dialer application and illustrates application of the method, Section 5 provides results of a comparison with conventional techniques and a classical POMDP, and Section 6 concludes.

2. Background

We begin by formalizing the problem. At each turn in a dialog, the dialog system takes a speech action a, such as "Where are you leaving from?". A user then responds with action u, such as "Boston". This u is processed by the speech recognition engine to produce an observation o, such as "AUSTIN". The dialog system examines o, updates its internal state, and outputs another a. The conventional and POMDP approaches differ in how they maintain this internal state, and how they choose actions given the state.

A conventional dialog manager maintains a state n such as a form or frame and relies on two functions for control, G and

Accepted after peer review of full paper

F. For a given dialog state n, $G(n) = a$ decides which system action to output, and then after observation o has been received, $F(n, o) = n'$ decides how to update the dialog state n to yield n'. This process repeats until the dialog is over. The important point is that G and F are written by hand, for example in a language such as VoiceXML.

By contrast, the POMDP tracks a probability distribution over many dialog states. In the POMDP, there are a set of *hidden* states, where each hidden state s represents a possible state of the conversation, including quantities such as the user's action u, the user's underlying goals, and the dialog history [2]. Because the true state of the conversation isn't known, the POMDP maintains a *belief state* (probability distribution) over these hidden states, b, where $b(s)$ is the *belief* (probability) that s is the true state. By adding models of how the hidden state changes and how the observation is corrupted, it is straightforward to update this distribution – i.e., $b'(s') = p(s'|a, o, b)$ – and there are methods for doing this efficiently and/or approximately [6, 7]. The belief state has the desirable property of accumulating information across all of the actions and observations over the course of the entire dialog history, and provides robustness to speech recognition errors.

In principle a developer could write a function to choose actions $G(b) = a$, but in practice it is not easy for a person to see how to make use of all of the information in the belief state. Instead, reinforcement learning is applied, in which a developer specifies high-level goals in the form of a reward function, $R(s, a)$. R assigns a measure of goodness to each state/action pair and communicates, for example, the relative values of short dialogs and successful task completion. An optimization procedure then searches for the best action to take in each belief state in order to maximize the sum of rewards over the whole dialog. The result is a value function $Q(b, a)$, which estimates the long-term reward of taking action a at belief state b. The optimal action in belief state b is then $a^* = \arg\max_a Q(b, a)$.

In practice, the domain of $Q(b, a)$ is too large and compression is applied. One method is the so-called "summary" method [3]: the intuition is to map b and a into lower-dimensional feature vectors $\hat{b}$ and $\hat{a}$, and to estimate a value function $\hat{Q}(\hat{b}, \hat{a})$ in this compressed space. For example, $\hat{b}$ might reduce a distribution over all cities into the probability of only the most likely city, and $\hat{a}$ might compress the class of *confirm* actions (*confirm(london)*,*confirm(boston)*) into a single *confirm(most-likely-city)*.

3. Method

To unify these two approaches, several changes are made. The conventional dialog manager is extended in three respects: first, its action selection function $G(n) = a$ is changed to output a *set* of one *or more* (M) allowable actions given a dialog state n, each with a corresponding summary action, $G(n) = \{(a_{(1)}, \hat{a}_{(1)}), \ldots, (a_{(M)}, \hat{a}_{(M)})\}$. Next, its transition function $F(n, o) = n'$ is extended to allow for different transitions depending on which of these action was taken, and it is also given access to the resulting POMDP belief state, $F(n, a, o, b') = n'$. A (human) dialog designer still designs the contents of the state n and writes the functions G and F.

For action selection, compression will be applied but the state features used for action selection will be a function of both the belief state b *and* the dialog state n. This state feature vector is written $\hat{x}$ and is computed by a feature-function $H(b, n) = \hat{x}$. The POMDP value function is correspondingly re-cast to assign values to these feature vectors, $\hat{Q}(\hat{x}, \hat{a})$.

The unified dialog manager operates as follows. At each time-step, the dialog manager is in state n and the POMDP is in belief state b. The dialog manager *nominates* a set of m allowable actions, where each action $a_{(m)}$ includes its summarized counterpart $\hat{a}_{(m)}$. The state features are computed as $\hat{x} = H(b, n)$. Then, the POMDP value function $\hat{Q}(\hat{x}, \hat{a})$ is evaluated for *only* those actions nominated by the dialog manager (not all actions), and the index m^* of the action that maximizes the POMDP value function is returned:

$$m^* = \arg \max_{m \in [1,M]} \hat{Q}(\hat{x}, \hat{a}_m). \quad (1)$$

Action a_{m^*} is then output and reward r and observation o are received. The POMDP updates its belief state $b'(s') = p(s'|a_{m^*}, o, b)$ and the dialog manager transitions to dialog state $n' = F(n, a_{m^*}, o, b')$. An example of this process taken from the real system described below is shown in Figure 3.

In this method, action selection can be viewed as a general reinforcement learning problem, where states are feature vectors $\hat{x}$. This enables any general-purpose reinforcement learning technique to be applied, such as value function approximation, in either an off-line or on-line setting. The only requirement is that the learning technique produce an estimate of $\hat{Q}(\hat{x}, \hat{a})$. Moreover, intuitively, the effect of constraining which actions are available prunes the space of policies, so if the constraints are well-informed, then optimization ought to converge to the optimal policy faster.

4. Example dialog system

To test the method we applied it to an existing voice dialer application, which has been accessible within the AT&T research lab for several years and which receives daily calls. The dialer's vocabulary consists of 50,000 AT&T employees. Since many employees have the same name, the dialer can disambiguate by asking for the callee's location. The dialer can also disambiguate between multiple phone listings for the same person (office and mobile) and can indicate when a callee has no number listed. This dialog manager tracks a variety of elements in its state n, including the most recently recognized callee, how many callees share that callee's name, whether the callee has been confirmed, and many others. This existing dialer was used as our baseline, labelled as "HC" in the results in Section 5.

The POMDP was then created. The belief state followed the SDS-POMDP model [2] and maintained a belief state over all callees. The user model and speech recognition models used to update the belief state were based on based on transcribed logs from 320 calls.

The existing dialer was then extended in two respects. First, rather than tracking the most recently recognized callees, it instead obtained the most likely callee from the POMDP belief state. Second, it was altered to nominate a set of one or more allowable actions using knowledge about this domain. For example, on the first turn of the dialog, the only allowed action was to ask for the callee's name. Once a callee has been recognized, the callee can be queried again or confirmed. Additional actions are allowed depending on the properties of the most likely callee – for example, if the top callee is ambiguous, then asking for the callee's city and state is allowed; and if the top callee has both a cellphone and office phone listed, then asking for the type of phone is allowed. The transfer action is permitted only after the system has attempted confirmation. This unified controller was called "HC+POMDP".

Because the actions were nominated by the hand-crafted dialog manager, it was straightforward to tailor the prompt wordings to the dialog context. For example, the first request for the callee's name was "First and last name", whereas the second was "Sorry, name please?". These both appeared to the planning algorithm as the summary action $\hat{a} = \mathit{AskName}$. Also, when a callee's name is ambiguous, it ought to be confirmed with the callee's location, and this was easy to implement.

For comparison, another controller was created which nominated every action at every time-step. Its actions also acted on the most likely callee in the belief state but no other restrictions were imposed. It could, for example, transfer a call to a callee who has not been confirmed, or ask for the city and state even if the top callee was not ambiguous. This controller was called "POMDP".

For optimization, the state features include 2 continuous features and several discrete features. The continuous features are taken from the belief state and are the probability that the top callee is correct, and the probability that the top callee's type of phone (office or cell) is correct. The discrete features are how many phone types the top callee has (none, one, two), whether the top callee is ambiguous (yes, no), and whether confirmation has yet been requested for the top callee (yes, no).

Finally, a simple reward function was created which assigns -1 per system action plus +/-20 for correctly/incorrectly transferring the caller at the end of the call.

Optimization was performed on "POMDP" and "HC+POMDP" using dialog simulation with the user and ASR models estimated from the 320 transcribed calls. The optimization method roughly followed summary point-based value iteration [3]. Space limitations preclude a complete description; in sketch, K synthetic dialogs were generated by randomly choosing allowed actions. The space of state features was quantized into small regions, and a transition and reward function over these regions were estimated by frequency counting, applying some smoothing to mitigate data sparsity. Straightforward value iteration was then applied to the estimated transition and reward functions to produce a value function $\hat{Q}(\hat{x}, \hat{a})$. The optimization procedure, simulation environment, state features, and action set were identical for "POMDP" and "HC+POMDP": the only difference was whether the set of allowed actions was constrained or not.

5. Results

Using the system described above, optimization was conducted for various numbers of K dialogs for "POMDP" and "HC+POMDP", ranging from $K = 10$ to $K = 10,000$. After optimization, each policy was evaluated in simulation for 1000 dialogs to find the average return, average task completion rate, and average dialog length. The simulation environment for optimization and evaluation were identical. For each value of K, this whole process (optimization and evaluation) was run 10 times, and the results of the 10 runs were averaged. 1000 simulated dialogs were also run with the baseline "HC", using the same simulation environment.

Results for task completion rate are shown in Figure 1. As the number of training dialogs increases, performance of both POMDP and HC+POMDP increase to roughly the same asymptote.[1] With sufficient training dialogs, both POMDP and HC+POMDP are able to out-perform the baseline. However, HC+POMDP reaches this asymptote with many fewer dialogs. Moreover, inspection of the 10 runs at each value of K showed that the HC+POMDP policies were significantly more consistent: Figure 2 shows that the standard deviation of the average total reward per dialog over the 10 runs is lower for HC+POMDP than for POMDP.

[1] Dialog length for all systems was also measured: it hovered very close to 4 system turns and did not show any clear trend.

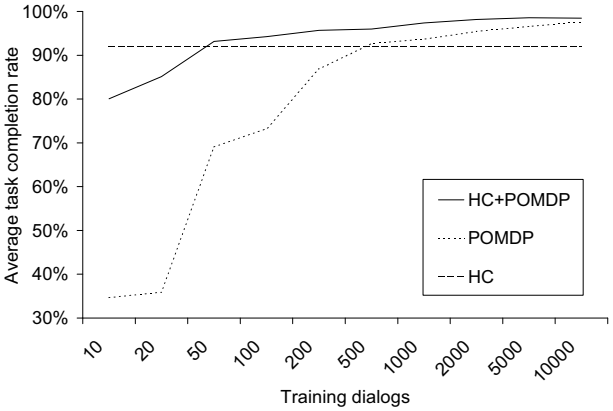

Figure 1: *Number of training dialogs K vs. task completion rate for the HC+POMDP, POMDP, and HC dialog managers.*

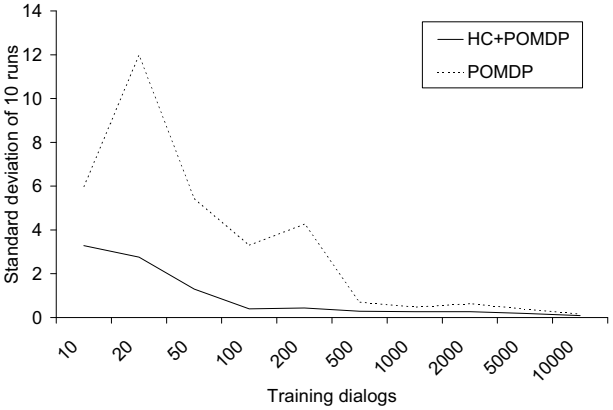

Figure 2: *Number of training dialogs vs. standard deviation in reward measured over 10 independent runs for the HC+POMDP and POMDP dialog managers.*

These results verify that, in dialog simulation at least, incorporating a POMDP into a conventional dialog system increases performance. Moreover, when compared to a pure POMDP, this method reduces training time and yields more consistent results vs. a pure POMDP. In other words, not only does this method combine the strengths of the two methods, it also reduces optimization time and less often produces spurious policies.

One of the policies created using with this method trained on 10,000 simulated dialogs was installed in our internal phone system, and it is now available alongside the baseline system. It uses up to 100 ASR N-Best entries and maintains a dialog beam of up to 100 callers. Its response time is essentially identical to the baseline system (2-3s). A webpage is available which shows the belief state and action selection running in real-time, and we have started to collect usage data. Figure 3 shows an example conversation illustrating operation of the method in detail.

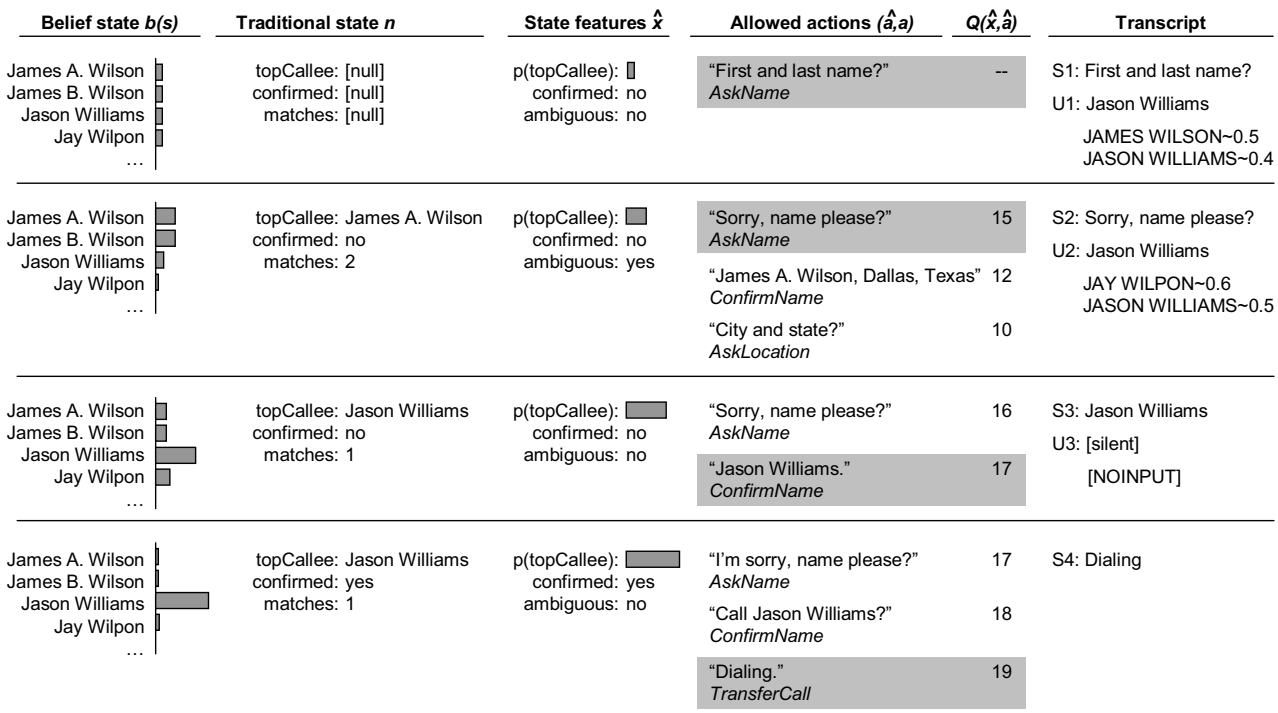

Figure 3: *Illustration of operation of the HC+POMDP dialog manager. The first column shows the POMDP belief state $b(s)$, which is a distribution over all possible callees. The second column shows the conventional dialog state n, which includes the name of the most likely callee in the belief state, and whether the top caller has been confirmed, and how many other callees share the top callee's name. The third column shows some of the the state features in $\hat{x}$, including the belief in the top callee (extracted from b) and whether the callee's name is ambiguous or confirmed (extracted from n). The fourth column shows the actions a (and summary actions $\hat{a}$ in italics) nominated by the conventional dialog manager. Because the conventional dialog manager has been designed by hand, it is straightforward to tailor the prompts to the dialog context – for example, confirming an ambiguous callee includes their location. The fifth column shows the value estimated by the optimization for each summary action given the current state features $\hat{Q}(\hat{x}, \hat{a})$, and the shading indicates the maximum $\hat{Q}$ value, which is output in the sixth column, showing the dialog transcript. The entries in upper-case letters show the results from the recognition process with confidence scores. Note that after the second system turn, "Jason Williams" has appeared on the N-Best list twice, and as a result in the beginning of the third system turn it has acquired the most mass in the belief state.*

6. Conclusions

This paper has presented a novel method to unify conventional dialog design practices in industry with the emerging approach in research based on partially observable Markov decision processes (POMDPs). The POMDP belief state and the conventional dialog state run in parallel, and the conventional dialog manager is augmented so that it nominates a set of one *or more* acceptable actions. The POMDP then chooses an action from this limited set. The method naturally accommodates compression akin to the "summary" method, and this enables the method to scale to non-trivial domains – here a voice dialer application covering 50,000 listings. Simulation experiments drawing on usage data from a real dialog system demonstrated that the method outperformed our existing baseline dialer, while simultaneously requiring less training data than a classical POMDP. We hope this method moves POMDPs an important step closer to commercial readiness.

7. References

[1] M. Cohen, J. Giangola, and J. Balough, *Voice User Interface Design*. Addison Wesley, 2004.

[2] J. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[3] ——, "Scaling POMDPs for spoken dialog management," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2116–2129, 2007.

[4] S. Singh, D. Litman, M. Kearns, and M. Walker, "Optimizing dialogue management with reinforcement leaning: experiments with the NJFun system," *Journal of Artificial Intelligence*, vol. 16, pp. 105–133, 2002.

[5] P. Heeman, "Combining reinforcement learning with information-state update rules," in *Proc HLT-NAACL, Rochester, New York, USA*, 2007, pp. 268–275.

[6] S. Young, J. Williams, J. Schatzmann, M. Stuttle, and K. Weilhammer, "The hidden information state approach to dialogue management," Cambridge University Engineering Department, Technical Report CUED/F-INFENG/TR.544, 2006.

[7] J. Williams, "Using particle filters to track dialogue state," in *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Kyoto, Japan*, 2007.