A case study of applying decision theory in the real world: POMDPs and spoken dialog systems

Jason D. Williams
AT&T Labs – Research
jdw@research.att.com

October 26, 2010

Abstract

Spoken dialog systems present a classic example of planning under uncertainty. Speech recognition errors are ubiquitous and impossible to detect reliably, so the state of the conversation can never be known with certainty. Despite this, the system must choose actions to make progress to a long term goal. As such, decision theory, and in particular partially-observable Markov decision processes (POMDPs), present an attractive approach to building spoken dialog systems. Initial work on "toy" dialog systems validated the benefits of the POMDP approach; however, it also found that straightforward application of POMDPs could not scale to real-world problems. Subsequent work by a number of research teams has scaled up planning and belief monitoring, incorporated high-fidelity user simulations, and married commercial development practices with automatic optimization. Today, statistical dialog systems are being fielded by research labs for public use. This chapter traces the history of POMDP-based spoken dialog systems, and sketches avenues for future work.

1 Introduction

Spoken dialog systems (SDSs) are a widespread commercial technology with a broad range of applications. For example, currently deployed telephone-based SDSs enable callers to check their bank balance, get airline gate information, or find the status of a train. In a car, an SDS enables drivers to change the music, check traffic conditions or get driving directions. SDSs on mobile devices enable people to find a business, send a message, dial a contact, or set a social-networking status. Analysts estimate the total market for SDSs is in the billions of US dollars per year [33].

Although widespread, spoken dialog systems remain challenging to build. First, to hear and understand users, spoken dialog systems use Automatic Speech Recognition (ASR) which is prone to errors. Despite years of research, ASR is still imperfect, and yields the wrong answer 20–30% of the time for

non-trivial tasks. As a result, a dialog system can never know the user's true intentions – i.e., to the dialog system, the state of the world is *partially observable*. Moreover, dialog is a temporal process that requires careful *planning*: early decisions affect the long-term outcome, and there are important trade-offs between confirming current hypotheses ("Flying to Boston, is that right?"), gathering more information ("When would you like to travel?"), and committing to the current hypothesis ("Ok, issuing a ticket from New York to Boston for flight 103 on March 15.").

In industry, these two issues are addressed through hand-crafted heuristics. Directed questions ("Please say the time you would like to depart."), confirmations ("Eleven thirty, is that right?"), and local accept/reject decisions ("Sorry, I didn't understand. What time was that?") help reduce uncertainty; and dialog plans – carefully designed by experts – are highly constrained. Although these techniques are sufficient for certain commercial applications, their scope and robustness are inherently limited. Increasing automation by only a few percent would have real commercial impact. Moreover, increasing robustness is an important step toward moving new applications of spoken dialog systems out of the research lab into widespread use, in domains such as robotics [50, 20], eldercare [41], handheld device interaction [29], situated interaction [9], and others.

With this in mind, researchers at several laboratories have turned to decision theory as a framework for building spoken dialog systems. With sequential decisions and a partially observable state, dialog systems present a classic example of decision-making under uncertainty, for which *partially-observable Markov decision processes* (POMDPs) are an attractive method. Initial work at several research laboratories applying POMDPs to toy spoken dialog systems in 2000–2005 suggested that POMDPs were indeed capable of achieving significantly better performance than the traditional approach of hand-crafting dialog control. However, this early work also identified numerous barriers to commercial use.

Since that early pioneering work, the research community has made substantial progress. Current approaches are now capable of handling a virtually unbounded number of possible dialog states, system actions, and observations, yet perform on-line inference in real-time, and perform off-line planning quickly. Methods have been developed for incorporating business rules into the policy, encoding structured domain knowledge into the state, and automatically learning transition dynamics from unlabelled data. Together these techniques have enabled POMDPs to scale to real-world dialog systems, producing better robustness to speech recognition errors, better task completion rates, and shorter dialogs.

This chapter has three broad goals. First, this chapter aims to present the spoken dialog task and explain why POMDPs are an attractive solution compared to current practice in industry, and related approaches in research (Sections 2 and 3). Second, this chapter details how POMDPs have been adapted to the requirements of this real-world task (Section 4). Third, this chapter identifies open problems for POMDP-based dialog systems, and suggests avenues for

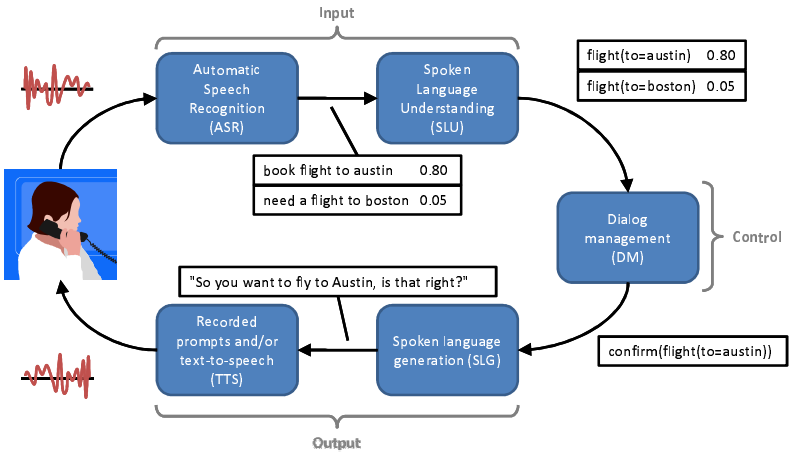

Figure 1: Components of a spoken dialog system.

future research (Section 5).

2 Background: spoken dialog systems

In general spoken dialog systems have a common logical architecture, with three modules: input, output, and control (Figure 1). The input module operates in two stages: automatic speech recognition (ASR) and spoken language understanding (SLU). First, ASR converts audio of the user's speech into an ordered *N-Best* list of the N best hypotheses for the words spoken, where the top ($n = 1$) hypothesis is the recognizer's best guess. For example, hypothesis $n = 1$ might be "Book a flight to Austin" and hypothesis $n = 2$ might be "Need a flight to Boston". In the SLU step, each N-Best entry is converted to a dialog act, which expresses the user's intention. For example, hypothesis $n = 1$ might be `flight(to=austin)` and hypothesis $n = 2$ might be `flight(to=boston)`.

The input module also produces a confidence score for each hypothesis, which is effectively a context-independent probability of correctness for each hypothesis on the N-Best list. For example, the confidence for $n = 1$ might be 0.80 and the confidence for $n = 2$ might be 0.05.

The control module receives the N-Best list of user dialog acts and performs two tasks. First, the control module tracks the current state of the dialog, accumulating history as required. For example, the control module might track how many times each piece of information has been requested, and what has been recognized so far. Second, based on the current state, the control module chooses which dialog act to reply to the user, such as `confirm(flight(to=austin))`.

Finally the output module receives the system's dialog act and presents it to the user. The dialog act is converted to a string of words using natural language generation (NLG). For example, `confirm(flight(to=austin))` might

	Yes/no	City/state	How may I help you?
Example phrase(s)	yes, nope	Tucson, Arizona	Help with um a bill
Distinct user intents	2	$\sim 30,000$	~ 250
In-grammar accuracy	99.8%	85.1%	89.5%
In-grammar rate	92.3%	91.0%	86.8%
Overall accuracy	92.1%	77.6%	77.7%
Correct accept rate	89.6%	60.3%	73.3%
False accept rate	1.8%	4.9%	8.3%

Table 1: Example accuracy of three grammars from commercially deployed spoken dialog systems running at AT&T. Even for simple tasks, ASR errors are unavoidable.

be converted to "So you want to fly to Austin, is that right?" Then the words are rendered as audio for the user, using pre-recorded prompts, text-to-speech (TTS), or a mixture of the two.

This architecture may be extended with additional modalities. For example, additional inputs might include the user's gesture, eye gaze, or keyboard entry; and additional outputs might include a graphical user interface [29], an animated head [9], or robotic articulators [59]. The focus of this chapter is the speech modality, as this underpins a wide range of applications and presents two important broad problems: *recognition errors* and *plan complexity*.

First, speech recognition errors are ubiquitous, and impossible to detect reliably. Table 1 shows accuracy for the top ($n = 1$) recognition result in several commercially deployed dialog systems. Simple recognition tasks yield near-perfect accuracy when the user speaks "in-grammar" – within the catalog of words and phrases the recognizer is capable of recognizing. For example, a yes/no grammar might recognize "yes", "no", and synonyms like "nope", "yup", "yeah", etc. This grammar might be used following a question like "So you want to fly to Austin, is that right?". In-grammar accuracy for more complex recognition tasks is less than perfect but still rather high.

Unfortunately input to the ASR is often "out-of-grammar": words or other sounds which are outside the bounds of what the recognizer is capable of understanding. Table 1 shows only 86.8%-92.3% of utterances are in-grammar. Users sometimes provide additional information ("Yes and I'm leaving tomorrow"), change the topic ("Wait which airport do you have me leaving from?"), engage in dialog repair ("No I said tomorrow"), get distracted ([to a co-worker] "Just a minute, I'm on the phone."), hesitate ("Umm... well..."), or remain silent. In addition, background noises (cars, televisions, other people) and communication channel problems (mobile phone network problems, Voice Over IP problems) can intrude. All of these can yield spurious recognition results.

Moreover, the confidence score cannot reliably identify errors. An ideal confidence score would assign 1.0 to accurate recognitions, and 0.0 to errors. In practice the probabilities are more evenly distributed. Figure 2 shows a receiver operating characteristic (ROC) curve for one question in a commercially spoken

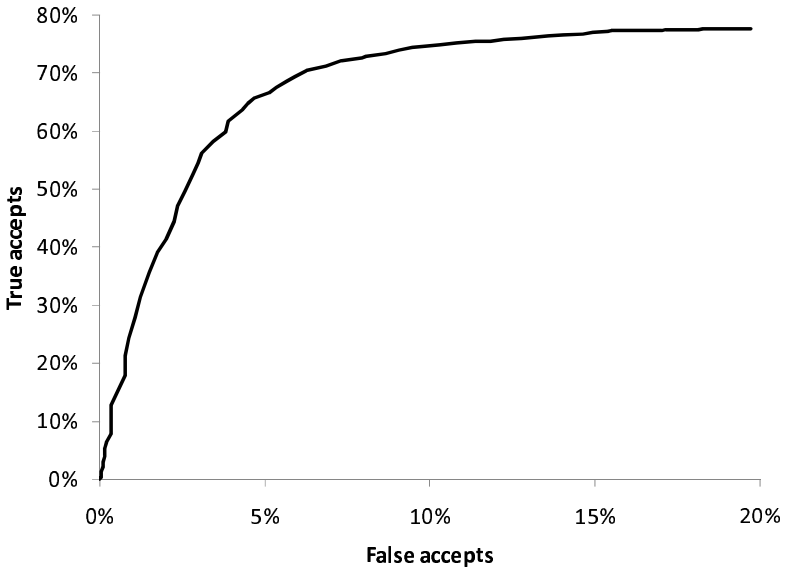

Figure 2: Receiver operating characteristic (ROC) curve for ASR confidence score. A "False accept" is a recognition error which is admitted, and a "True accept" is a correct recognition which is admitted. Taken from a grammar running in a commercially deployed spoken dialog system at AT&T.

dialog system which illustrates that any choice of accept/reject threshold will result in admitting some errors and failing to admit some accurate recognitions.

The second challenge for spoken dialog systems is plan complexity. Dialog is a sequential process in which short-term actions affect the long-term outcome of the dialog, and it can be very difficult to anticipate what the consequences of different actions will be. For example:

- When is it better to ask an open question, such as "How may I help you?", and when is it better to ask more directed questions, such as "Which city are you leaving from?" Open questions can lead to faster dialogs when recognition is accurate, but engender more recognition errors since users' responses are more complex. Directed questions lead to more reliable recognitions but prolong the dialog.

- When is it better to confirm ("Was that Boston?") or commit ("Alright, I'm issuing a ticket from Boston to London.")? Confirming helps ensure information collected is reliable, but prolongs dialogs. In order to complete a task, the system must commit at some point.

- What is the best form for confirmations – explicit ("Are you leaving from Boston?") or implicit ("Ok, Boston. And where to?"). Explicit confirmations produce more regular speech and are thus more reliable, but they prolong the conversation and can be tedious.

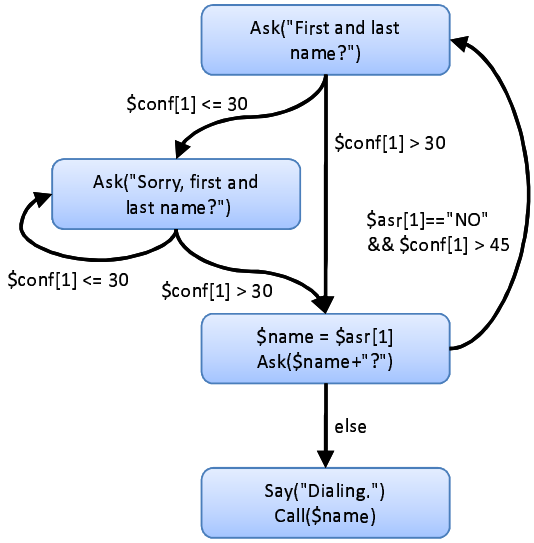

Figure 3: Example of a dialog system design akin to those commonly used in industry. `$asr[1]` is the top recognition hypothesis and `$conf[1]` is its confidence score.

- When is it better to provide more information ("I found five flights, here they are...") or less ("Here is a flight that matches your request...")? Listening to verbose system output is tedious, but users often want to know all of their options.

The conventional approach is for a dialog designer to create a flow chart that describes the dialog plan (example, Figure 3). Each node is labeled with a system action, and transitions are labeled with observations or other conditions. Unfortunately the size of the flowchart is bounded by the number of dialog situations that a designer can contemplate – in practice, on the order of 100 or 1000. Yet the space of possible dialog situations is astronomically larger: in practical systems, the space of possible system actions is often on the order of 10^{10} or more, and the space of possible observations – N-Best lists of dialog acts with their associated real-valued confidence scores – is unbounded. Thus the current practice of designing the dialog flow by hand ignores many possible dialog situations, resulting in sub-optimal plans.

These two problems – ASR errors and plan complexity – conspire to produce spurious dialogs. Consider the dialog in Figure 4. In turn 4, the user says their name, but this is mis-recognized as "Athens in Akron", which leads to a cascade of further errors. There is clear room for improvement in the dialog controller: for example, would it be better to have explicitly confirmed "Athens" in turn 5? Perhaps more confidence in "Athens" should have been required? Should off-topic recognitions for this question be disregarded altogether? These are all difficult design questions with consequences to the remainder of the dialog.

T	ID	System / user action
1	M	Are you a registered user ?
2	U	No I'm not. No [NO I'M NOT NO]
3	M	What is your full name ?
4	U	User Name [ATHENS IN AKRON]
5	M	There is an Athens in Georgia and in Greece. Which destination did you want ?
6	U	Neither [NEITHER]
7	M	I'm sorry, I didn't catch that. There is an Athens in Georgia and in Greece. Which destination did you want ?
8	U	Georgia [GEORGIA]
9	M	A flight from Athens... Where do you want to go ?
10	U	Start over [START OVER]
11	M	Do you really want to start over ?
12	U	Yes [YES]
13	M	What city are you leaving from ?
14	U	Hamilton, Ontario [HILTON ONTARIO]
15	M	.....

Figure 4: Sample conversation with a spoken dialog system affected by misunderstandings in the CMU Communicator spoken dialog system, taken from [10]. U indicates a user turn and M indicates a machine turn. In turn 4 the user says their name (omitted for privacy reasons) but they are misrecognized as saying "Athens in Akron". This single error then leads to a complete misalignment between the user and system.

As this example illustrates, ASR errors render the true state of the dialog partially observable, and actions have long-term effects. In sum, dialog is an instance of planning under uncertainty, for which POMDPs provide a principled approach. The next section explains how dialog can be modeled as a POMDP and gives concrete examples of its benefits compared to current practices in industry.

3 Casting a spoken dialog system as a POMDP

In this section, we first develop a model of spoken dialog as a POMDP, then illustrate how a POMDP-based dialog system can outperform traditional approaches. Finally, POMDP-based dialog systems are compared to related methods for learning dialog strategies.

3.1 Spoken dialog as a POMDP

Recall from Chapter 3, Section 3 that a POMDP consists of a tuple $\{\mathcal{A}, \mathcal{O}, \mathcal{S}, \mathcal{T}, \mathcal{Z}, R, \gamma\}$, where $\mathcal{A}$ is a set of actions $a \in \mathcal{A}$, $\mathcal{O}$ is a set of observations $o \in \mathcal{O}$, $\mathcal{S}$ is a set of states $s \in \mathcal{S}$, $\mathcal{T}$ is a transition function $P(s'|s, a)$, $\mathcal{Z}$ is an observation function $P(o'|s', a)$, R is a reward function $R(s, a) \in \mathbb{R}$, and γ is a discount factor $0 \leq \gamma \leq 1$.

To frame spoken dialog as a POMDP, the system's dialog act is cast as the POMDP action a and the ASR/SLU output is cast as the POMDP observation o. For example, POMDP action a might correspond to the system action `ask(city)` ("What city are you leaving from?"), `confirm(city=boston)` ("Was that Boston?"), or `print-ticket(to=boston)` ("Ok, issuing a ticket to Boston.") POMDP observation o might correspond to an N-Best list with 2 items (`city=boston` and `city=austin`), and their corresponding confidence scores.

The hidden state of the dialog – including all quantities which the system cannot directly observe – is cast as the hidden POMDP state s. The exact elements of the hidden state can vary with the domain; here we describe one common factorization known as the SDS-POMDP model, which serves as a base framework for many information-seeking dialogs [87].

In the SDS-POMDP model, the hidden state contains three quantities, $s = (g, h, u)$. g is the *user's goal* – this is the user's long-term aim in the conversation, such as booking a flight from London to Boston on November 23 in economy class. u is the true, unobserved *user action*, such as `yes,city=boston` ("Yes, Boston"), remaining silent, or saying something out-of-grammar which the ASR/SLU cannot recognize. Finally, h is the *dialog history* – an accumulator variable which records aspects of hidden state which the dialog designer feels are important, such as whether the departure city has been asked, confirmed, or not yet discussed. Including this accumulator variable enables system and user behavior to be conditioned on distant history.

Depending on the domain, the SDS-POMDP model may be extended with other quantities, such as the user's level of expertise with the system or the user's emotional state or stress level [14]. Variables representing other world state may also be added – for example, in a dialog system for troubleshooting a network connection, the state of the network router (on/off, working/failed, status lights, etc.) might be included [77].

Returning to the core SDS-POMDP model, we next substitute $s = (g, h, u)$ into the POMDP transition and observation functions. For the transition function we have:

$$\begin{aligned} P(s'|s,a) &= P(g',h',u'|g,h,u,a) \\ &= P(g'|g,h,u,a)P(u'|g',g,h,u,a)P(h'|u',g',g,h,u,a). \end{aligned} \quad (1)$$

Conditional independence is then assumed as follows. The first term in Equation 1, called the *user goal model*, indicates how the user's goal changes (or does not change) at each time-step. It is assumed that the user's goal at each time-step depends only on the previous goal, the dialog history, and the machine's action:

$$P(g'|g,h,u,a) = P(g'|g,h,a). \quad (2)$$

The second term, called the *user action model*, indicates what actions the user is likely to take at each time step. It is assumed the user's action depends on their (current) goal, the dialog history, and the machine action:

$$P(u'|g',g,h,u,a) = P(u'|g',h,a). \quad (3)$$

The third term, called the *dialog history model*, captures relevant historical information about the dialog. This component has access to the most recent value of all variables:

$$P(h'|u',g',g,h,u,a) = P(h'|u',g',h,a). \quad (4)$$

Thus the transition function becomes

$$P(s'|s,a) = P(g'|g,h,a)P(u'|g',h,a)P(h'|u',g',h,a). \quad (5)$$

Similarly, substituting $s = (g, h, u)$ into the POMDP observation function yields $P(o'|g', h', u', a)$. It is assumed that the recognizer output depends only on the user's action:

$$P(o'|g',h',u',a) = P(o'|u'). \quad (6)$$

The POMDP belief state update then becomes

$$\begin{aligned} & b'(g',h',u') \\ &= \eta \cdot p(o'|u') \sum_{g,h,u} P(g'|g,h,a)P(u'|g',h,a)P(h'|u',g',h,a)b(g,h,u) \end{aligned} \quad (7)$$

Figure 5 shows the SDS-POMDP model as an influence diagram.

The models themselves have to be estimated of course – in general this can be done by collecting dialogs, and fitting models to the data. In practice this requires relatively few dialogs – generally a few hundred or less.

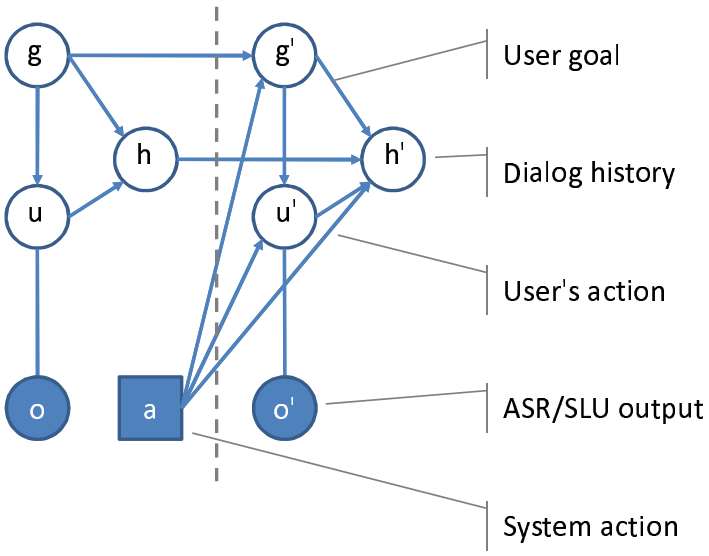

Figure 5: The SDS-POMDP model shown as an influence diagram. The POMDP state s has been factored into 3 components: the user's goal g, the user's true goal u, and the dialog history h. The POMDP observation o is the output from the ASR/SLU.

3.2 Illustration of a POMDP-based dialog system

To illustrate, a simple dialog system is now reviewed. Full details are available in [87] and [76]. In this system, the user is trying to book travel from one city to another in a world with 3 cities, resulting in 6 possible user goals. The system actions include asking for the origin or destination, confirmations, and printing a ticket. The dialog history indicates whether each slot is unasked, filled, or confirmed, yielding a set of 9 dialog histories. The user can say the name of a city, say "from" or "to" a city, say "from a to b", "yes", "no", and remain silent, yielding a set of 18 possible user actions. In addition there is a binary flag indicating whether the dialog is in its first turn or not. Taking the Cartesian product of these components yields 1944 POMDP states. The simulated ASR output includes one hypothesis and no confidence score, yielding 18 possible observations.

Simple transition function models are assumed – for example, it is assumed that the user's goal stays fixed and that the user sometimes provides additional information in their responses. The observation function is parameterized by p_{err}, the probability of making a uniformly distributed ASR confusion error. The reward function assigns $+10$ for printing the correct ticket, -10 for printing the wrong ticket, and -1 for each question asked.

With the transition function, observation function, and reward function in place, POMDP optimization can be applied to produce a dialog plan $\pi(b)$.

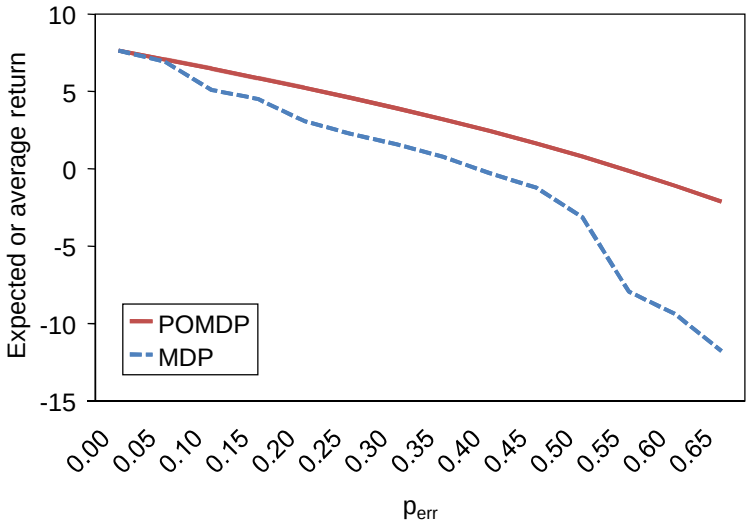

Figure 6: Expected or average return of POMDP policies and an MDP baseline. The horizontal axis p_{err} refers to the probability of making a simulated speech recognition error. As errors increase the POMDP outperforms the MDP by an increasing margin.

Here the "Perseus" implementation [63] of point-based value iteration is used (Chapter 3, Section 3.3.2). Running Perseus with 500 belief points finds good policies.

Two types of baseline policies were also developed on the same system: a (fully-observable) Markov decision process (MDP) akin to others used in the literature [60], and three hand-crafted controllers. The MDP (Chapter 3, Section 2) is trained using the same reward function, but differs from the POMDP in that it does not model the partial observability of the user's goals. Rather, it tracks a single hypothesis for the user's goal, updating it whenever it sees a new recognition result. The three hand-crafted controllers implement three common dialog strategies found in commercial systems.

Average return for the POMDP and MDP are shown in Figure 6 for various ASR error rates on the horizontal axis. As errors increase performance of all systems decreases; however the POMDP outperforms the MDP by an increasing margin. The POMDP shows a similar performance gain over hand-crafted controllers [85].

As this example illustrates, POMDPs are more robust to ASR errors, and in practice this leads to shorter dialogs with higher task completion rates. The POMDP achieves this by performing several types of reasoning which traditional systems cannot. First, POMDPs synthesize information across multiple dialog turns including multiple N-Best lists, whereas traditional systems make local accept/reject decisions considering only the top recognition result, and cannot accumulate noisy evidence over time. For example, consider Figure 7, in

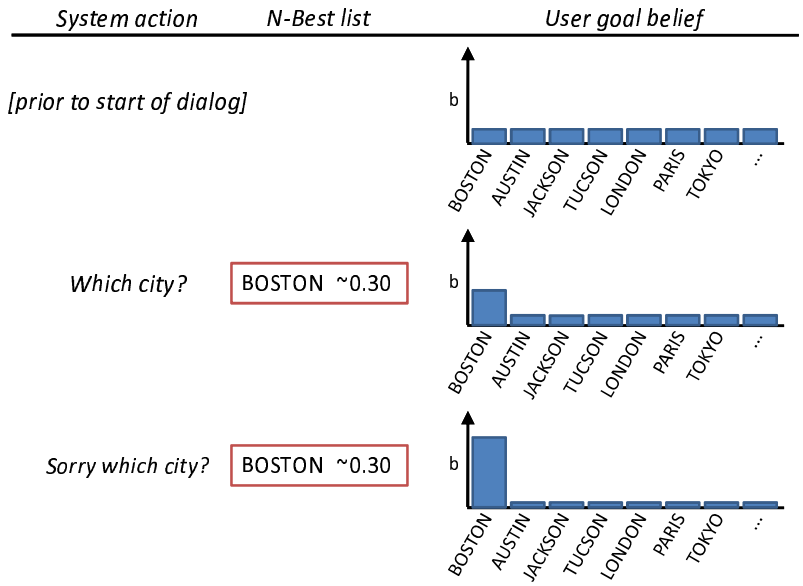

Figure 7: Example conversation with a spoken dialog system illustrating two successive low-confidence recognitions. In this example, both recognitions are correct. The POMDP accumulates weak evidence over time, whereas traditional methods would ignore both recognitions because they are below the rejection threshold (here set to 0.40). In effect, traditional methods are discarding possibly useful information.

which the same hypothesis is recognized twice, with low confidence. Whereas a conventional system would discard each recognition, POMDP-based systems are able to use this information. Similarly, POMDP-based approaches are able to identify commonality across N-Best lists – for example, in Figure 8. Even though the $N = 2$ item was never the single most likely item on a *local* N-Best list, it is the *globally* most likely user goal after the second turn. This behavior is not possible with traditional approaches.

In computing the belief state, POMDPs also incorporate prior expectations about users' behavior and goals. This additional source of information provides another source of robustness not available to traditional systems. For example, in Figure 9, the N-Best list contains two competing alternatives for the user's action. A traditional system would simply take the top hypothesis; a POMDP is able to view each of these in light of how likely they are given the current dialog context, and emerge with the correct answer. The same can be observed with competing user goals, as in Figure 10.

In sum, the belief state provides a *cumulative* confidence score for the user's goal over the entire dialog – including all N-Best lists and the confidence scores they contain – as well as prior expectations about user behavior and goal preferences. A distribution over multiple dialog state hypotheses adds inherent

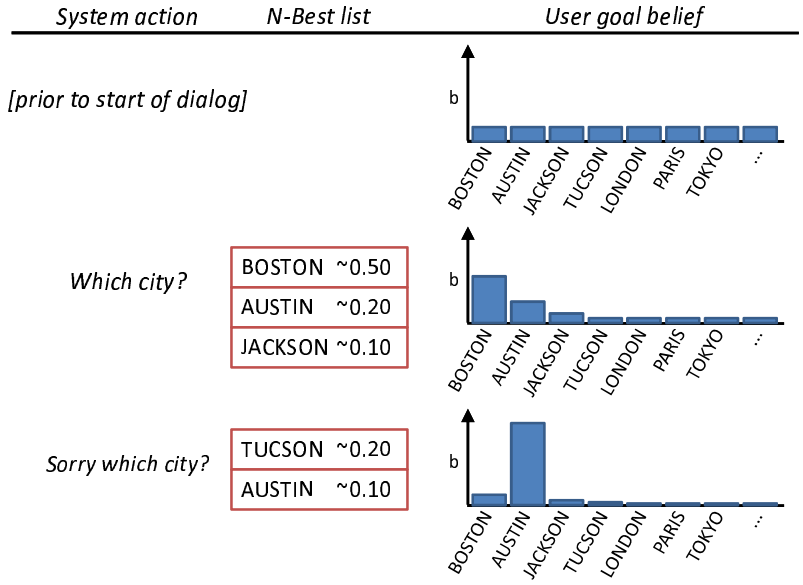

Figure 8: Example conversation with a spoken dialog system illustrating how tracking multiple dialog state hypotheses is able to identify commonality across ASR N-Best lists. In this example, the correct answer appears in the N=2 position on two successive recognitions. The POMDP is able to identify this commonality whereas traditional methods cannot.

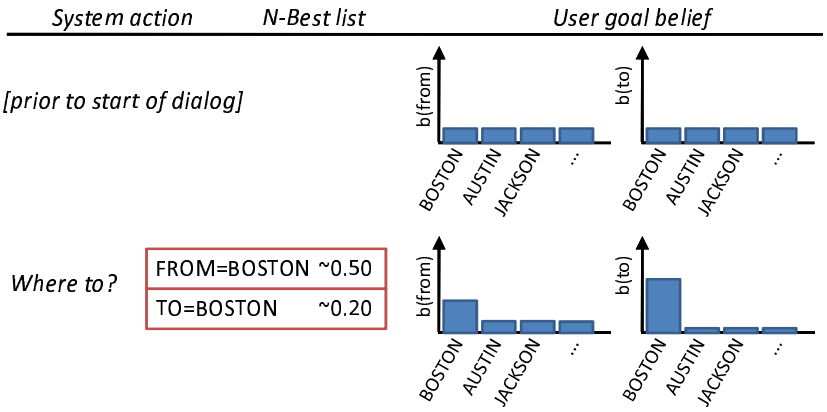

Figure 9: Illustration of how the POMDP is able to incorporate prior expectations about a user's behaviors. Here the N-Best list contains two competing hypotheses with similar ASR probabilities for the user's action; the effect of the user action model $P(u'|g', h, a)$ is to prefer the more likely user action – `TO=BOSTON`, which more directly answers the system's question, even though it is recognized with a lower confidence score.

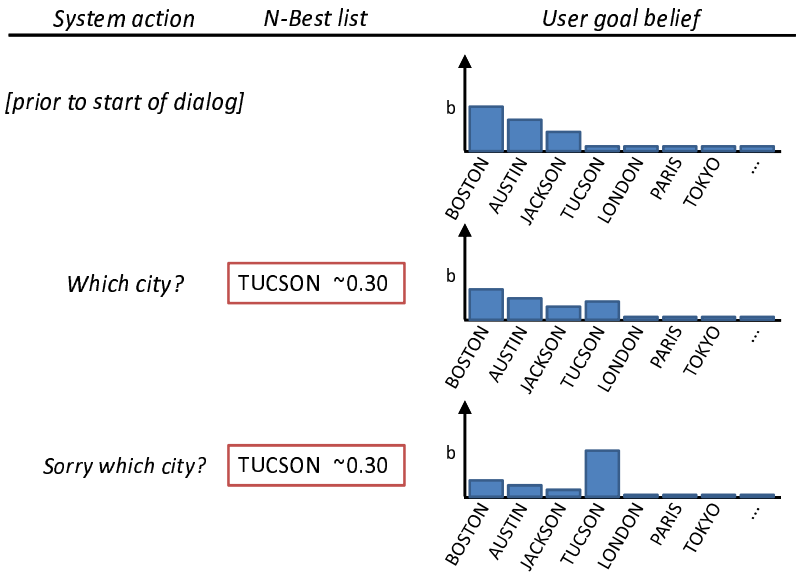

Figure 10: Illustration of how the POMDP is able to integrate a prior over user goals. Here the same recognition result is received twice: `TUCSON` with low confidence. After the first update, the belief in `TUCSON` has increased, but the goal with the highest prior still has the most belief. After the second update, `TUCSON` now has the most belief. The POMDP is able to trade-off between prior beliefs and evidence – ultimately, sufficient evidence can overcome any non-zero prior.

robustness, because even if an error is introduced into one dialog hypothesis, it can later be discarded in favor of other, uncontaminated dialog hypotheses.

POMDPs also choose actions differently than traditional approaches, and concretely the benefit is that POMDPs construct much more detailed dialog plans. A large-scale commercial dialog system design might contain 100 pages of flowcharts (akin to Figure 3), comprising perhaps 1000 dialog situations. Increasing this by an order of magnitude is unimaginable, since no single human designer would be able to conceptualize the entire design in their mind. By contrast, POMDPs can consider many more dialog situations. Moreover, POMDP policies are performing explicit planning with respect to a global optimization criterion (the reward function), but with a hand-crafted dialog plan it is not known what the criteria are – indeed, dialog designers are often making educated guesses at the optimal choice of action [82].

3.3 Related work

Many other approaches have been suggested for automatically choosing actions in a spoken dialog systems. One approach is to choose actions to maximize an immediate utility, rather than a long term reward [28, 1, 43, 44, 9]. This can be thought of as setting the POMDP discount factor to 0. This approach avoids the complexities of performing long-term planning, but requires that the immediate utilities ensure the agent makes progress toward a long-term goal. However, in this approach, designing a utility function can require more trial and error.

In a POMDP, the belief state is a strict probabilistic interpretations of a distribution over hidden states. Other techniques for tracking multiple dialog instead track *scores* which are monotonic but not necessarily probabilities [26, 25]. These scores do not correspond to probabilities – in other words, they are an approximation to Eq 7. Scores can be easier to compute than probabilities; however, the effects of the approximations made have not been studied. Also, scores can be more difficult to interpret: for example, a score of 0.5 may correspond to a probability of 0.9 or 0.1. At present, it is not known whether using scores degrades overall dialog performance compared to maintaining proper probabilities.

It is also possible to track multiple dialog states, but choose actions according to hand-crafted rules rather than by optimization [26]. In this approach, a person decides how to make use of the belief state. This is a difficult task to which dialog designers are generally not accustomed. Moreover, a dialog designer can only consider a small number of dialog situations, whereas an optimization process can create much more detailed dialog plans. Experiments with usability subjects have shown that optimization, done properly, yields more successful dialogs than hand-crafted design by better exploiting the information contained in the distribution over multiple dialog states [90, 71].

Another method for dialog control is to mimic actions observed in a dialog corpus using supervised learning [22, 35, 34, 27]. This approach may be attractive when there is a corpus contains interactions with a dialog manager

which is believed to be optimal – for example, a human "wizard" who sees the output of the speech recognition, and controls the dialog manager in real time. This approach allows for the creation of more detailed dialog plans than could be written by a human designer. However, it unclear whether a human wizard could make good use of distribution over dialog states, in real time, while interacting with real users.

Other approaches choose actions as POMDPs do, but track state differently. For example, there has been a large amount of work which tracks a single hypothesis for the current dialog state, casting optimization as an MDP rather than a POMDP[74, 37, 39, 60, 46, 47, 18, 36, 68, 2, 67, 25, 49]. In this approach it is difficult to aggregate information across recognitions, and also to make use of multiple entries on the N-Best list. Experiments have shown than POMDP-based dialog systems are more robust than MDP counterparts [90, 71].

In addition, numerous other aspects of the spoken dialog system can be "learned" for a specific task, such as application-specific grammars [64, 65], prompt wording [16, 45], choice of text-to-speech audio [11], and others. Learning in these areas can certainly improve performance of a spoken dialog system, but is separate from the dialog management task.

In sum, POMDPs provide a principled approach to building spoken dialog systems. However, building a real-world POMDP-based dialog system presents a host of obstacles. These challenges, and the techniques developed to overcome them, are covered in the next section.

4 Real-world POMDP-based dialog systems

Early work applying POMDPs to SDSs made extensive use of toy problems like the one introduced in the previous section. Researchers at several labs initially followed in the POMDP tradition, creating abstract tasks akin to tiger [15], tiger-grid [40], chain-walk [32], or rock-sample [61]. In this formula, first a designer specifies a flat POMDP state, observation, and action set, and then asserts a transition, observation, and reward function in tabular form. Optimization is performed to produce a policy, and that policy is evaluated by measuring its average return on the same system dynamics. The POMDPs developed in early work had on the order of 1000 states, 10 actions and 10 observations, and were optimized with techniques such as the augmented POMDP [50], grid-based approximations [95, 96] point-based value iteration [85, 87, 48, 63], and heuristic search value iteration [62], among others.

These preliminary studies confirmed that as speech recognition errors increase, POMDPs outperform both the traditional commercial practice of handcrafting a policy, and other emerging research techniques such as MDP optimization. Analysis showed that POMDPs were realizing many of the behaviors predicted in the previous section. Spurred on by these preliminary results, the research community set out to scale POMDPs to real dialog systems.

Applying the SDS-POMDP model to real systems presented a series of substantial obstacles. After about 5 years of steady progress, researchers achieved the

aim of scaling the POMDP approach to real systems: today, statistical dialog systems are capable of handling a virtually unbounded number of possible dialog states, system actions, and observations, yet perform on-line inference in real-time, and perform off-line planning quickly. Methods have been developed for incorporating business rules into the policy, encoding structured domain knowledge into the state, and learning from a high-fidelity simulated user. These systems have been demonstrated to the research community [92, 31, 80, 73] and are available for public use [91, 75]. Recently, toolkits have been released to assist non-experts build statistical dialog systems [12, 83].

This chapter reviews five crucial advances which enabled the SDS-POMDP model to be scaled to real systems. First, two of these advances address the problem of *tracking* a distribution over dialog states quickly and accurately. These are discussed in Sections 4.1 and 4.2. The remaining three of these advances tackle the problem of *choosing actions* optimally, in very large state spaces. These are discussed in Sections 4.2–4.5.

4.1 Incorporating the N-Best list into the observation function

Early work cast the POMDP observation function as a single ASR hypothesis without a confidence score. However, as mentioned above in Section 2, real ASR outputs is an "N-Best" of N hypotheses $\tilde{\mathbf{u}} = (\tilde{u}_1, \ldots, \tilde{u}_N)$, each with a *local*, context-independent probability of correctness $P(u = \tilde{u}_n|o)$. For example, an N-Best list of length 2 with $\tilde{\mathbf{u}} = (\text{austin}, \text{boston})$ might assign $P(u = \text{austin}|o) = 0.7$ and $P(u = \text{boston}|o) = 0.1$. The residual mass can be used to determine the probability of correctness for items not on the N-Best list – for example, if there were 100 items in this grammar, then $P(u = \text{cleveland}|o) = (1.0 - 0.7 - 0.1)/(100 - 2) = 0.2/98 \approx 0.002$.

In practice, when the correct answer is not in the $N = 1$ position, about half the time it is further down the N-Best list – so making full use of the information on the N-Best list will produce more accurate belief state updates. Computing $P(u|o)$ can be done quite accurately by creating a regression model that takes features from the ASR and SLU process and yields a vector of probabilities [72, 84]. However incorporating $P(u|o)$ into the belief state update requires some care. The POMDP observation function calls for $P(o|u)$, where o is the *entire* N-Best list and all of the associated probabilities. In other words, the POMDP calls for a model of how the N-Best lists and their probabilities are *generated* $P(o|u)$, but the ASR is providing a model of how likely its entries are to be correct $P(u|o)$.

One solution is to apply Bayes' rule:

$$P(o|u) = \frac{P(u|o)P(o)}{P(u)} \tag{8}$$

$$= k_1 \cdot \frac{P(u|o)}{P(u)} \tag{9}$$

$$\approx k_2 \cdot P(u|o). \tag{10}$$

During the update, $P(o)$ is constant; this is absorbed into the normalization constant k_1. The key assumption is that $P(u)$ is uniform over all actions, and can also be absorbed into the normalizing constant k_2 in Eq 10. Of course this is not strictly true, but in practice the error introduced is small since the full belief state update (Eq. 7) includes the term $P(u'|g', a, h)$, which is more informative than $P(u)$.

Overall this approach enables all of the information in the observed N-Best list to be used, which yields a more accurate belief state [84].

4.2 Scaling up belief tracking

Computing the belief state update itself (Eq 7) in real time is problematic for real-world dialog systems. To illustrate this, consider a small dialog task with four slots (e.g., origin city, destination city, time of travel, etc.), where each slot takes on 1000 values. This implies $O(1000^4) = O(10^{12})$ possible values for g, and similarly $O(10^{12})$ possible values for u. In a flat representation, the update iterates over each value of the 4 elements g, g', u, and u'; thus the whole update is $O((10^{12})^4) = O(10^{48})$ which is inconceivable in real time. The picture is even worse for more sophisticated dialog tasks.

One way of speeding up the belief state update is to *factor* the influence diagram further, assume conditional independences as appropriate, and apply *approximate* inference. For example, the user's goal g and user's action u could be decomposed into slots (such as origin city, destination city, time, etc.) [86, 89, 88, 13, 71]. It might then be assumed that the origin city is conditionally independent of the date of departure, or that a user's preference for characteristics of a restaurant or bar are independent of the location. With conditional independences in place, a speed-up over exhaustive enumeration can be realized by using *approximate* inference techniques to compute *marginal* distributions over each variable. For example, researchers in two different labs have applied particle filters [78], and loopy belief propagation [70]. Factoring approaches enable the number of hidden variables to be scaled, but their computation still grows with the number of values in each variable.

An alternative is to track only a handful of hidden states, and to track all the remaining states *en masse* without distinguishing between them. One implementation of this idea is to track hidden states in *partitions*, where each partition contains one or more user goals [94, 93, 90, 83]. At first there is one root partition that contains all user goals. Then partitions are sub-divided as necessary based on N-Best lists and system actions. For example, if a voice dialer system recognized "Jason", then the root partition would divide into two partitions: all listings with the first name equal to `jason`, and all listings with the first name *not* `jason` (denoted ¬`jason`). Then if "Williams" were recognized, each of these partitions would split, yielding four partitions: ¬`jason`∧¬`williams`, ¬`jason`∧`williams`, `jason`∧¬ `williams`, and `jason`∧`williams`. If the number of partitions grows larger than a threshold, then low-probability partitions can be *recombined*, summing their beliefs and ignoring the distinctions between them. This recombination ensures that the belief update can run in real-time

regardless of the length of the dialog. An example of the partitioning process is shown in Figure 11.

One key benefit of partitioning is that it allows a proper joint distribution to be tracked over all of the elements of the user's goal. This is important because there are often meaningful dependencies between slots. For example, business names and locations, people's first and last names, and flight departure and arrival cities are all highly coupled.

Another benefit of partitioning is that its update time is not dependent on the number of underlying user goals – in other words, it enables both the number of variables and the number of variable values to be scaled. However, partitioning makes several important assumptions – chiefly, that the user's goal is generally fixed, and can change in only highly regulated ways.

On the one hand, factoring has the benefit of allowing arbitrary changes in hidden variables, whereas partitioning does not. On the other hand, partitioning is capable of modeling a proper joint distribution and scales to arbitrary numbers of values per variable whereas factoring does not. Recognizing their complementary strengths, recent work has started to unify the two, for example by factoring with variables that may be partitioned [71]. However, at present no technique is able to scale to an arbitrary number of hidden values, handle user goal changes, and track a joint distribution over all goals.

In sum, factoring and partitioning enable POMDP-based dialog systems to track a distribution over a very large number of dialog states in real time, where complete enumeration is hopeless [78, 90, 71, 83]. However, both factoring and partitioning have implications for planning, discussed next.

4.3 Scaling up planning

Optimal planning in POMDPs is notoriously intractable. Indeed, early work verified that POMDP planners could not scale beyond toy tasks with a handful of user goals, even with the most sophisticated planning algorithms available. Moreover, the two advances discussed above present challenges for traditional POMDP planning. The main problem is that traditional POMDP optimization enumerates all possible hidden states, yet as discussed above, the scale of the dialog task renders this impossible for even the simpler problem of belief tracking. Performing traditional POMDP planning over all possible hidden states is completely hopeless.

The key to scaling up planning is the insight that planning can be done in a small *feature space* by exploiting properties of the dialog domain. The basic idea is to map the belief state and action into smaller feature spaces, perform planning and choose actions in feature space, and then map that action back to the full space [86, 89, 88].

As an illustration, consider a slot-filling dialog, in which the system's goal is to obtain the value of the user's goal for a set of *slots*. For simplicity first consider a dialog problem with a single slot – for example, a weather information service with N cities. The system can `ask` for the slot value ("What city do you want the weather for?"), `confirm` the slot's value ("The weather in Seattle, is

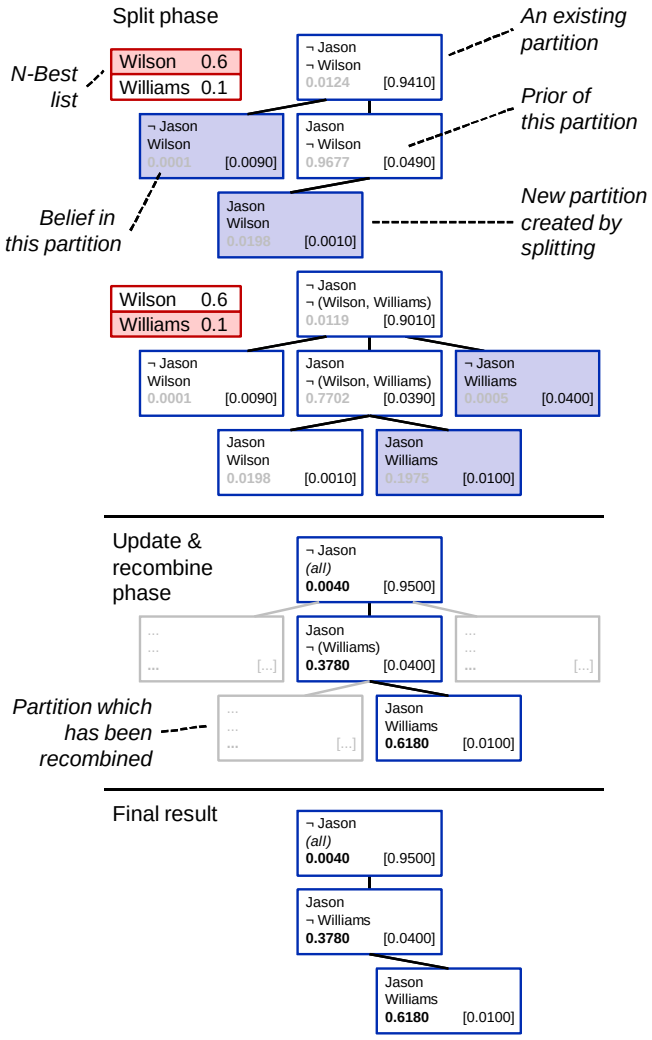

Figure 11: Illustration of belief monitoring with partitions. Initially there are two partitions (two upper right boxes in blue outline): `jason` and ¬`jason`. The number in the lower right of each partition indicates its prior probability; the number in the lower left is the belief (posterior). In this example, the N-Best list (shown in red outline) contains two entries: `wilson` and `williams`. In the *split* phase, partitions are sub-divided for each N-Best entry – first on `wilson` which increases the number of partitions from 2 to 4, then on `williams` which increases the number of partitions from 4 to 6. In the figure, new partitions are shown with blue shading. In the *update and recombine* phase, the belief in each partition is updated, and low probability partitions are re-combined. In this example the maximum number of partitions is limited to 3.

that right?"), or `submit` the slot's value ("The weather in Seattle today will be overcast and rainy."). There is a single `ask` action, and N `confirm` and `submit` actions, for a total of $1 + 2N$ actions. The key insight is that the `confirm` and `submit` actions can be limited *a priori* to act on only *the most likely slot value*, reducing the set of useful actions to 3. In other words, even though there are a very large set of *possible* actions available to the dialog system, only a small number of these are *useful* in a given dialog context. Further, to choose among these actions, it can be reasonably assumed that only the belief held by the most likely value (or the top M values) is relevant. In other words, the planner can act on a small set of *features* of the belief state, rather than considering the full belief simplex.

Recall that in real-world dialog systems, the belief state cannot be maintained explicitly, but is rather represented in a factored or partitioned representation. Fortunately, factored and partitioned belief states marry well with feature-based action selection. First, the designer creates a function which maps from the partitioned or factored belief state to feature vector $\hat{b}$. This feature vector might contain, for example, the amount of belief held by the most likely user goals. The designer also specifies a small set of *summary* actions $\{\hat{a}_1, \ldots, \hat{a}_N\}$, and a method for mapping these to fully-instantiated dialog actions. For example, the `confirm` summary action could be mapped to confirm the most likely slot value, such as "Boston, is that right?"

The aim of optimization is then to estimate $\hat{\pi}$ which maps from the feature vector $\hat{b}$ to a summary action $\hat{a}$, $\hat{\pi}(\hat{b}) = \hat{a}$. Unlike in traditional POMDP optimization, the learner does not have direct access to the transition function $\hat{P}(\hat{b}'|\hat{b}, \hat{a})$ or reward function $\hat{R}(\hat{b}, \hat{a})$, because in general these cannot be computed analytically from the underlying dynamics. Instead, the learner is presented with a simulator, or trajectories of the form $(\hat{b}_0, \hat{a}_1, r_1, \hat{b}_1, \hat{a}_2, r_2, \hat{b}_2, \ldots)$. Thus learning is now a general reinforcement learning problem, and can be accomplished in a variety of ways. Chapter 4 describes reinforcement learning in detail; in this domain, early work sampled trajectories through summary space $(\hat{b}_0, \hat{a}_1, r_1, \hat{b}_1, \hat{a}_2, r_2, \hat{b}_2, \ldots)$, used these to estimate a reward function $\hat{R}(\hat{b}, \hat{a})$ and grid-based transition function $\hat{P}(\hat{b}'|\hat{b}, \hat{a})$, and applied simple value iteration [89, 88, 81]. Subsequently, many other approaches have been explored, including eligibility traces [58], SARSA [24, 23], Monte Carlo methods [90], novel variants of Least-Squares Policy Iteration [38], and Natural Actor-Critic [71].

An alternative for planning using summary actions which *does* admit traditional POMDP planning is the *Permutable POMDP* [19]. Here states are dynamically re-ordered by their belief, allowing planning in master space using summary-style actions. While elegant, this approach requires that the state variables are enumerated – as discussed above, this is impossible with the large state spaces required in real-world dialog problems.

4.4 Adding more sophisticated simulated users

In traditional POMDPs it is usually assumed that the model of the environment $P(s'|s, a)$ is correct. In spoken dialog systems, the "environment" is the users

of the system and the ASR/SLU input channel. The behavior of real people is highly complex, and modeling this behavior is an unsolved problem and active area of research in its own right. Studies have found that modeling user behavior well requires non-trivial data structures – for example, modeling persistent goals over long dialog segments requires a dynamic stack of intentions [52, 57, 69, 54, 56]. Implementations of these models generally contain complex series of deterministic and random decisions, and this complexity makes it extremely difficult to incorporate them into an influence diagram. In addition, research has shown that optimizing a dialog system with an overly simplistic simulated user may yield promising results when tested on the same simplistic simulated user, but very poor results on more realistic simulated users (or real users!) [53, 30]. Similarly, other studies have found that the most reliable ASR/SLU simulations rely on sequences of operations which would be problematic to encode in an influence diagram [55].

In sum, it would be beneficial to make use of a high-fidelity simulator during the optimization process, but that simulator can't be encoded into the transition function. As a result, state-of-the-art POMDP-based dialog systems make use of *two* distinct user models: a *simplified, internal* simulated user which can readily be incorporated into an influence diagram, and a *complex, external* simulated user which runs in a purely generative mode. The internal model is used to perform inference to compute the belief state; the external model is used to simulate dialogs for policy learning [90, 71].

4.5 Adding business rules and domain knowledge to planning

In many dialog tasks, dialog designers know a great deal about the structure of the optimal policy. For example, it is easy to articulate rules such as "don't attempt to confirm a value before it has been requested." In addition, in commercial settings, dialog systems must be guaranteed to follow certain business rules, such as always verifying a password before allowing a caller to transfer funds. Thus there are constraints available on when actions can be taken.

One solution is to apply a *partial program* [7] to dialog systems [79, 21]. A hand-crafted dialog manager *with its own internal state* runs in parallel with the belief state. At each time-step, the hand-crafted dialog manager outputs a vector of features, and nominates a set of one or more *allowed* actions; the planner chooses which action is *optimal* from this limited set based on the feature vector.

Experiments have shown that this approach not only enables expert knowledge and business rules to be encoded in the policy, but also that optimization runs faster and more reliably as compared to standard exhaustive exploration [79, 21]. The intuition is that the expert knowledge avoids taking spurious actions and focuses planning effort on the set of plausible plans.

From an engineering standpoint, this formulation provides an important bridge from current commercial practices to statistical approaches. Dialog designers currently write computer programs that output a single action at each

time-step, possibly with the help of a dialog design tool (for example, [6, 5, 3, 4]). It is relatively straightforward to extend this model to output a small set of actions at each time-step. In effect, this approach captures the strengths of both human expertise (in providing high-level structure and cohesion) and the robustness and fine-grained control afforded by statistical techniques.

Crucially, the feature vector can contain elements drawn from either the belief state or the hand-crafted dialog manager state, and the features themselves can take any form, including categorical, binary, integral, or real-valued elements. For example, consider a voice dialer in which a caller says the name of a listing they want to call [80, 81]. In this problem, the hidden state enumerates all the user's possible goals (the listings), and the belief state maintains a distribution over these. There are two probability features: one for the belief in the top listing, and another in the belief in the top type of phone (office or cell). In addition, the planner needs information about whether the most likely listing has an office phone and/or cellphone available, whether there are multiple listings with the same name in the directory, and how many times the caller has been asked each question. Unlike the user's goal (the listings), there is no uncertainty in these items. Thus the feature vector includes categorical features for how many phone types are available for the most likely listing ("none","one","two"), and binary features ("true" or "false") for whether the most likely listing is ambiguous and whether confirmation has been requested by the system. As this example illustrates, a heterogeneous feature vector is useful because there are often fully observable features useful for planning, and tracking a distribution over these adds additional computation with no benefit.

4.6 Section summary

This chapter has described five key advances for decision-theoretic approaches to dialog control: incorporating the N-Best list into the observation function, speeding up the belief state update, scaling up planning, employing more sophisticated user simulations, and injecting business rules and domain knowledge to planning. Each of these has played an important role in moving the POMDP approach closer to commercial readiness. Although state-of-the-art dialog systems are implemented rather differently to a traditional POMDP, it is important to realize that all of their machinery is in service of the two fundamental ideas of POMDPs: tracking a distribution over multiple states, and choosing actions to maximize a sum of rewards. It is these two fundamental ideas which are responsible for the gains in performance achieved by POMDP-based dialog systems over competing approaches.

Research into statistical approaches to dialog remains an active area, and a number of international research teams are currently working on this topic. Although researchers at different labs have made good progress in the past five years, important open questions remain, and there is still interesting work ahead. The next section concludes by discussing suggesting some opportunities.

5 Conclusions and open problems

Spoken dialog systems are a classic example of planning under uncertainty: dialog is a control problem where speech recognition errors render the true state of the world only partially observable, yet the agent must choose actions to make progress toward a long-term goal. Difficult trade-offs exist between actions which gather information but prolong the dialog, and actions which can conclude the dialog but have high cost if taken in error.

The dialog management problem is well modeled as a POMDP. In a POMDP, the state of the world isn't observed directly, so a *distribution* over many dialog states is maintained. Actions are then chosen to maximize the expected *sum* of rewards until the end of the interaction.

In early work, toy dialog problems were solved using traditional POMDP optimization techniques. To apply POMDP-based dialog systems to real-world dialog problems, numerous extensions have been made to tailor the POMDP approach to the dialog domain. Several POMDP-based dialog systems have been trialed with paid usability subjects [90, 71], and currently POMDP-based dialog systems are being fielded with real users in the 2010 Spoken Dialog Challenge [8].

Despite this progress, several important open questions remain:

- The analysis in Section 4 suggests that first-order "lifted" POMDPs would be a natural fit with SDSs. As discussed above, lifted inference (in the form of partitions) is required to perform the belief update in real time. To scale, planning ought to operate on the same representation of state. Although first-order POMDPs are in their infancy today [51], perhaps spoken dialog systems will provide a fertile testbed for emerging first-order POMDP work.

- There is a clear opportunity for a POMDP-based dialog system to refine its models as it interacts with users. While there has been some work exploring learning models of user behavior from logs of dialog data [66, 69], there are many issues remaining to explore: for example, there is an interdependence between the user's actual behavior, the system's model of the user's behavior, and the system's current policy. More work is needed to understand how to jointly optimize the models of the users behavior and the system's policy.

- Currently, it is not well understood how to set the reward function used to guide action selection. In practice, practitioners try different values until optimization yields a reasonable dialog policy. There is a clear need for specific guidance on how to set the reward function. One possibility is to examine existing (hand-designed) dialog systems to infer what reward function would make their policy optimal. This process is known as *inverse reinforcement learning*, and algorithms have been suggested for inferring reward functions from expert trajectories [42], including in the partially observable case [17].

- Substantial work is needed to make POMDP-based SDSs accessible to commercial practitioners. Tools are needed in general-purpose programming language such as Python or Java which encapsulate common functions, such as policy optimization, model representation and estimation, simulated dialog execution, etc.

Although interesting research questions such as these remain, POMDP-based dialog systems are rapidly maturing, and it seems promising that they will be applied in industry in the near future.

6 Additional reading

For readers interested in reading more about speech recognition and dialog systems in general, see:

- Bruce Balentine. 2001. *How to Build a Speech Recognition Application: Second Edition: A Style Guide for Telephony Dialogues.* Enterprise Integration Group; second edition.
- Michael H. Cohen, James P. Giangola, and Jennifer Balogh. 2004. *Voice User Interface Design.* Addison-Wesley Professional.
- Frederick Jelinek. 1998. *Statistical Methods for Speech Recognition (Language, Speech, and Communication).* The MIT Press.
- Kristiina Jokinen and Michael McTear. 2010. *Spoken Dialogue Systems.* Morgan and Claypool Publishers.
- Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing.* Prentice Hall; second edition.
- Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing.* The MIT Press.

References

[1] Conversation as action under uncertainty. In *Proc Conf on Uncertainty in Artificial Intelligence (UAI), Stanford, California*, pages 455–464, 2000.

[2] Comparing user simulation models for dialog strategy learning. In *Proc Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), Rochester, New York, USA*, 2007.

[3] Audium studio application design tool. `http://www.audiumcorp.com/Products/`, 2010.

[4] Nuance openspeech dialog design tool. http://www.nuance.com/dialog/, 2010.

[5] Speechdraw application design tool. http://www.speechvillage.com/home/, 2010.

[6] Voiceobjects application design tool. http://www.voiceobjects.com, 2010.

[7] David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *Proc National conference on Artificial Intelligence, Edmonton, Alberta, Canada*, pages 119–125, 2002.

[8] Alan W Black, Susanne Burger, Brian Langner, Gabriel Parent, and Maxine Eskenazi. Spoken dialog challenge 2010. In *Proc Workshop on Spoken Language Technologies (SLT), Spoken Dialog Challenge 2010 Special Session, Berkeley, CA*, 2010.

[9] Dan Bohus and Eric Horvitz. Models for multiparty engagement in open-world dialog. In *Proc SIGdial Workshop on Discourse and Dialogue, London, UK*, 2009.

[10] Dan Bohus and Alexander I Rudnicky. Integrating multiple knowledge sources for utterance-level confidence annotation in the CMU Communicator spoken dialog system. Technical Report CMU-CS-02-190, Carnegie Mellon University, 2002.

[11] Cedric Boidin, Verena Rieser, Lonneke van der Plas, Oliver Lemon, and Jonathan Chevelu. Predicting how it sounds: Re-ranking dialogue prompts based on TTS quality for adaptive spoken dialogue systems. In *Proc INTERSPEECH, Special Session on Machine Learning for Adaptivity in Spoken Dialogue, Brighton, UK*, 2009.

[12] Trung H. Bui, Dennis Hofs, and Boris van Schooten. POMDP toolkit for spoken dialog systems. http://wwwhome.ewi.utwente.nl/~hofs/pomdp/index.html.

[13] Trung H Bui, Mannes Poel, Anton Nijholt, and Job Zwiers. A tractable DDN-POMDP approach to affective dialogue modeling for general probabilistic frame-based dialogue systems. In *Proc Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Intl Joint Conf on Artificial Intelligence (IJCAI), Hyderabad, India*, pages 34–37, 2007.

[14] Trung H Bui, Mannes Poel, Anton Nijholt, and Job Zwiers. A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems. *Natural Language Engineering*, 15(2):273–307, 2009.

[15] Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *Proc Conf on Artificial Intelligence, (AAAI), Seattle*, 1994.

[16] John Chen, Srinivas Bangalore, Owen Rambow, and Marilyn A. Walker. Towards automatic generation of natural language generation systems. In *Proc Intl Conf on Computational Linguistics (COLING), Taipei*, 2002.

[17] Jaedeug Choi and Kee-Eung Kim. Inverse reinforcement learning in partially observable environments. In *IJCAI2009*, pages 1028–1033, 2009.

[18] Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. Reinforcement learning of dialogue strategies with hierarchical abstract machines. In *Proc Workshop on Spoken Language Technologies (SLT), Aruba*, pages 182–185, 2006.

[19] Finale Doshi and Nicholas Roy. The permutable POMDP: fast solutions to POMDPs for preference elicitation. In *Proc International Joint Conference on Autonomous Agents and Multiagent Systems, Estoril, Portugal*, pages 493–500, 2008.

[20] Finale Doshi and Nicholas Roy. Spoken language interaction with model uncertainty: an adaptive humanrobot interaction system. *Connection Science*, 20(4):299318, 2008.

[21] Milica Gasic, Fabrice Lefevre, Filip Jurcicek, Simon Keizer, Francois Mairesse, Blaise Thomson, Kai Yu, and Steve Young. Back-off action selection in summary space-based POMDP dialogue systems. In *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Merano, Italy*, 2009.

[22] David Griol, Lluìs F. Hurtado, Encarna Segarra, and Emilio Sanchis. A statistical approach to spoken dialog systems design and evaluation. *Speech Communication*, 50(8-9):666–682, 2008.

[23] James Henderson and Oliver Lemon. Mixture model POMDPs for efficient handling of uncertainty in dialogue management. In *Proc Association for Computational Linguistics Human Language Technologies (ACL-HLT), Columbus, Ohio*, 2008.

[24] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid reinforcement/supervised learning for dialogue policies from Communicator data. In *Proc Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Intl Joint Conf on Artificial Intelligence (IJCAI), Edinburgh*, pages 68–75, 2005.

[25] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511, 2008.

[26] Ryuichiro Higashinaka, Mikio Nakano, and Kiyoaki Aikawa. Corpus-based discourse understanding in spoken dialogue systems. In *Proc Association for Computational Linguistics (ACL), Sapporo, Japan*, 2003.

[27] Chiori Hori, Kiyonori Ohtake, Teruhisa Misu, Hideki Kashioka, and Satoshi Nakamura. Statistical dialog management applied to wfst-based dialog systems. In *Proc Intl Conf on Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taiwan*, pages 4793–4796, 2009.

[28] Eric Horvitz and Tim Paek. A computational architecture for conversation. In *Proc 7th International Conference on User Modeling (UM), Banff, Canada*, pages 201–210, 1999.

[29] Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, and Preetam Maloor. MATCH: An architecture for multimodal dialogue systems. In *Proc Association for Computational Linguistics (ACL), Philadelphia, USA*, 2002.

[30] Dongho Kim, Hyeong Seop Sim, Kee-Eung Kim, Jin Hyung Kim, Hyunjeong Kim, and Joo Won Sung. Effects of user modeling on POMDP-based dialogue systems. In *Proc INTERSPEECH, Brisbane, Australia*, 2008.

[31] Kyungduk Kim and Gary Geunbae Lee. Multimodal dialog system using hidden information state dialog manager. In *Proceedings of the ninth international conference on multimodal interfaces (ICMI 2007) Demonstratino Session, Nagoya*, 2007.

[32] Daphne Koller and Ronald Parr. Policy iteration for factored mdps. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford, California*, pages 326–334, 2000.

[33] Mae Kowalke. DMG: Recession won't hamper IVR market growth. `http://www.tmcnet.com/channels/ivr/articles/44089-dmg-recession-wont-hamper-ivr-market-growth.htm`, October 2008.

[34] Cheongjae Lee, Sangkeun Jung, Kyungduk Kim, and Gary Geunbae Lee. Hybrid approach to robust dialog management using agenda and dialog examples. *Computer speech and language*, Accepted for publication, 2009.

[35] Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5):466–484, May 2009.

[36] Oliver Lemon, Kallirroi Georgila, and James Henderson. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the TALK TownInfo evaluation. In *Proc Workshop on Spoken Language Technologies (SLT), Aruba*, pages 178–181, 2006.

[37] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialogue strategies. *IEEE Trans on Speech and Audio Processing*, 8(1):11–23, 2000.

[38] Lihong Li, Jason D Williams, and Suhrid Balakrishnan. Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In *Proc INTERSPEECH, Brighton, UK*, 2009.

[39] Diane J Litman, Michael S Kearns, Satinder B Singh, and Marilyn A Walker. Automatic optimization of dialogue management. In *Proc Association for Computational Linguistics (ACL), Hong Kong*, 2000.

[40] Michael Littman, Anthony Cassandra, and Leslie Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning, San Francisco, CA*, pages 362–370. Morgan Kaufmann, 1995.

[41] Alex Mihailidis, Jennifer N Boger, Marcelle Candido, and Jesse Hoey. The COACH prompting system to assist older adults with dementia through handwashing: An efficacy study. *BMC Geriatrics*, 28(8), 2008.

[42] Andrew Y Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proc Intl Conf on Machine Learning (ICML), Stanford, California*, 2000.

[43] Tim Paek and Eric Horvitz. Grounding criterion: Toward a formal theory of grounding. Technical Report MSR-TR-2000-40, Microsoft Research, 2000.

[44] Tim Paek and Eric Horvitz. On the utility of decision-theoretic hidden subdialog. In *Proc ISCA Workshop on Error Handling in Spoken Dialogue Systems, Chateau-d'Oex-Vaud, Switzerland*, pages 95–100, 2003.

[45] Taghi Paksima, Kallirroi Georgila, and Johanna Moore. Evaluating the effectiveness of information presentation in a full end-to-end dialogue system. In *Proc SIGdial Workshop on Discourse and Dialogue, London, UK*, 2009.

[46] Olivier Pietquin. *A framework for unsupervised learning of dialogue strategies*. PhD thesis, Faculty of Engineering, Mons (TCTS Lab), Belgium, 2004.

[47] Olivier Pietquin and Thierry Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):589–599, 2006.

[48] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: an anytime algorithm for POMDPs. In *Proc Intl Joint Conf on Artificial Intelligence (IJCAI), Acapulco, Mexico*, pages 1025–1032, 2003.

[49] Verena Rieser. *Bootstrapping Reinforcement Learning-based Dialogue Strategies from Wizard-of-Oz data*. PhD thesis, Saarland University, 2008.

[50] Nicholas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialog management for robots. In *Proc Association for Computational Linguistics (ACL), Hong Kong*, pages 93–100, 2000.

[51] Scott Sanner. First-order models for sequential decision-making. `http://videolectures.net/ilpmlgsrl09_sanner_fomsdm/`, July 2009.

[52] Jost Schatzmann, Kallirroi Georgila, and Steve Young. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proc SIGdial Workshop on Discourse and Dialogue, Lisbon, Portugal*, pages 178–181, 2005.

[53] Jost Schatzmann, Matthew N Stuttle, Karl Weilhammer, and Steve Young. Effects of the user model on simulation-based learning of dialogue strategies. In *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), San Juan, Puerto Rico, USA*, 2005.

[54] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Proceedings of Human Language Technologies / North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, 2007.

[55] Jost Schatzmann, Blaise Thomson, and Steve Young. Error simulation for training statistical dialogue systems. In *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Kyoto, Japan*, pages 526–531, 2007.

[56] Jost Schatzmann, Blaise Thomson, and Steve Young. Statistical user simulation with a hidden agenda. In *Proc SIGdial Workshop on Discourse and Dialogue, Antwerp, Belgium*, pages 273–282, 2007.

[57] Jost Schatzmann, Karl Weilhammer, Matthew N Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126, June 2007.

[58] Konrad Scheffler and Steve Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc Human Language Technologies (HLT), San Diego, USA*, pages 12–18, 2002.

[59] Candice L Sidner and Christopher Lee. *Conversational Informatics: An Engineering Approach*, chapter Attentional Gestures in Dialogues between People and Robots. Wiley and Sons, 2007.

[60] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement leaning: experiments with the NJFun system. *Journal of Artificial Intelligence*, 16:105–133, 2002.

[61] Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, July 2004.

[62] Trey Smith and Reid Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc Conference on Uncertainty in Artificial Intelligence*, pages 542–549, 2005.

[63] Matthijs T J Spaan and Nikos Vlassis. Perseus: randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[64] Bradford Starkie, Greg Findlow, Khanh Ho, Alvaro Hui, Lawrence Law, Liron Lightwood, Simon Michnowicz, and Christian Walder. Lyrebird [tm]: Developing spoken dialog systems using examples. In *Grammatical Inference: Algorithms and Applications; 6th International Colloquium, ICGI, Amsterdam*, pages 354–358. Springer-Verlag Lecture Notes in Computer Science, 2002.

[65] David Suendermann, Jackson Liscombe, Krishna Dayanidhi, and Roberto Pieraccini. A handsome set of metrics to measure utterance classification performance in spoken dialog systems. In *Proceedings of the SIGDIAL 2009 Conference*, pages 349–356, London, UK, September 2009. Association for Computational Linguistics.

[66] Umar Syed and Jason D Williams. Using automatically transcribed dialogs to learn user models in a spoken dialog system. In *Proc Association for Computational Linguistics Human Language Technologies (ACL-HLT), Columbus, Ohio*, 2008.

[67] Joel R Tetreault, Dan Bohus, and Diane J Litman. Estimating the reliability of MDP policies: A confidence interval approach. In *Proc Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), Rochester, New York, USA*, 2007.

[68] Joel R Tetreault and Diane J Litman. Using reinforcement learning to build a better model of dialogue state. In *Proc European Association for Computational Linguistics (EACL), Trento, Italy*, 2006.

[69] Blaise Thomson, Jost Schatzmann, Karl Welhammer, Hui Ye, and Steve Young. Training a real-world POMDP-based dialog system. In *Proc Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT) Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies, Rochester, New York, USA*, pages 9–17, 2007.

[70] Blaise Thomson, Jost Schatzmann, and Steve Young. Bayesian update of dialogue state for robust dialogue systems. In *Proc Intl Conf on Acoustics, Speech, and Signal Processing (ICASSP), Las Vegas, USA*, 2008.

[71] Blaise Thomson and Steve Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24:562–588, 2010.

[72] Blaise Thomson, Kai Yu, Milica Gasic, Simon Keizer, Francois Mairesse, Jost Schatzmann, and Steve Young. Evaluating semantic-level confidence scores with multiple hypotheses. In *Proc INTERSPEECH, Brisbane, Australia*, 2008.

[73] Sebastian Varges, Silvia Quarteroni, Giuseppi Riccardi, Alexei V. Ivanov, and Pierluigi Roberti. Combining POMDPs trained with user simulations and rule-based dialogue management in a spoken dialogue system. In *ACL-IJCNLP Demonstrations*, 2009.

[74] Marilyn A Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Articial Intelligence Research*, 12:387–416, 2000.

[75] Jason D Williams. AT&T voice dialer demonstration. `http://www.research.att.com/people/Williams_Jason_D`.

[76] Jason D Williams. *Partially Observable Markov Decison Processes for Spoken Dialogue Management*. PhD thesis, Cambridge University, 2006.

[77] Jason D Williams. Applying POMDPs to dialog systems in the troubleshooting domain. In *NAACL-HLT Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies, Rochester, New York, USA*, pages 1–8, 2007.

[78] Jason D Williams. Using particle filters to track dialogue state. In *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Kyoto, Japan*, 2007.

[79] Jason D Williams. The best of both worlds: Unifying conventional dialog systems and POMDPs. In *Proc INTERSPEECH, Brisbane, Australia*, 2008.

[80] Jason D Williams. Demonstration of a POMDP voice dialer. In *Proc Demonstration Session of Association for Computational Linguistics Human Language Technologies (ACL-HLT), Columbus, Ohio*, 2008.

[81] Jason D Williams. Integrating expert knowledge into POMDP optimization for spoken dialog systems. In *Proc AAAI Workshop on Advancements in POMDP Solvers, Chicago*, 2008.

[82] Jason D Williams. Spoken dialogue systems: challenges, and opportunities for research. In *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Merano, Italy*, 2009.

[83] Jason D Williams. Incremental partition recombination for efficient tracking of multiple dialog states. In *Proc Intl Conf on Acoustics, Speech, and Signal Processing (ICASSP), Dallas, USA*, 2010.

[84] Jason D Williams and Suhrid Balakrishnan. Estimating probability of correctness for ASR N-best lists. In *Proc SIGdial Workshop on Discourse and Dialogue, London, UK*, 2009.

[85] Jason D Williams, Pascal Poupart, and Steve Young. Factored partially observable Markov decision processes for dialogue management. In *Proc Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Intl Joint Conf on Artificial Intelligence (IJCAI), Edinburgh*, 2005.

[86] Jason D Williams and Steve Young. Scaling up POMDPs for dialog management: The "summary POMDP" method. In *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), San Juan, Puerto Rico, USA*, pages 177–182, 2005.

[87] Jason D Williams and Steve Young. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422, 2007.

[88] Jason D Williams and Steve Young. Scaling POMDPs for spoken dialog management. *IEEE Trans. on Audio, Speech, and Language Processing*, 15(7):2116–2129, 2007.

[89] Jason D Williams and Steve J Young. Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI). In *Proc American Association for Artificial Intelligence (AAAI) Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems, Boston*, 2006.

[90] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174, April 2010.

[91] Steve Young, Simon Keizer, Kai Yu, Franois Mairesse, Filip Jurccek, Blaise Thomson, Milica Gaic, and Fabrice Lefvre. Tourist information system for cambridge. `http://mi.eng.cam.ac.uk/research/dialogue/demo.html`.

[92] Steve Young, Jost Schatzmann, Blaise Thomson, Karl Weilhammer, and Hui Ye. The hidden information state dialogue manager: A real-world POMDP-based system. In *Proc Demonstration Session of Human Language Technologies: The Annual Conference of the North American*

Chapter of the Association for Computational Linguistics (NAACL-HLT), Rochester, New York, USA, 2007.

[93] Steve Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye. The hidden information state approach to dialog management. In *Proc Intl Conf on Acoustics, Speech, and Signal Processing (ICASSP), Honolulu, Hawaii, USA*, pages IV149–IV152, 2007.

[94] Steve Young, Jason D Williams, Jost Schatzmann, Matthew N Stuttle, and Karl Weilhammer. The hidden information state approach to dialogue management. Technical Report CUED/F-INFENG/TR.544, Cambridge University Engineering Department, 2006.

[95] Bo Zhang, Qingsheng Cai, Jianfeng Mao, Eric Chang, and Baining Guo. Spoken dialogue management as planning and acting under uncertainty. In *Proc INTERSPEECH, Aalborg, Denmark*, pages 2169–2172, 2001.

[96] Bo Zhang, Qingsheng Cai, Jianfeng Mao, and Baining Guo. Planning and acting under uncertainty: A new model for spoken dialogue systems. In *Proc Conf on Uncertainty in Artificial Intelligence (UAI), Seattle, Washington*, pages 572–579, 2001.