Partially Observable Markov Decision Processes for Spoken Dialog Systems

Jason D. Williams[1]
AT&T Labs – Research

Steve Young
Cambridge University
Engineering Department

Abstract

In a spoken dialog system, determining which action a machine should take in a given situation is a difficult problem because automatic speech recognition is unreliable and hence the state of the conversation can never be known with certainty. Much of the research in spoken dialog systems centres on mitigating this uncertainty and recent work has focussed on three largely disparate techniques: parallel dialog state hypotheses, local use of confidence scores, and automated planning. While in isolation each of these approaches can improve action selection, taken together they currently lack a unified statistical framework that admits global optimization. In this paper we cast a spoken dialog system as a partially observable Markov decision process (POMDP). We show how this formulation unifies and extends existing techniques to form a single principled framework. A number of illustrations are used to show qualitatively the potential benefits of POMDPs compared to existing techniques, and empirical results from dialog simulations are presented which demonstrate significant quantitative gains. Finally, some of the key challenges to advancing this method – in particular scalability – are briefly outlined.

Keywords: Spoken dialog system, dialog management, planning under uncertainty, user modelling, Markov decision processes, decision theory

Corresponding author: Jason D. Williams
Email: jdw@research.att.com
Phone: +1 973 360 8138
Fax: +1 973.360.8092
Mailing address: AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07932
USA

[1] Work carried out while at Cambridge University Engineering Department.

Introduction

Spoken dialog systems (SDS) help people accomplish a task using spoken language. For example, a person might use an SDS to buy a train ticket over the phone, to direct a robot to clean a bedroom, or to control a music player in an automobile. Building SDSs is a challenging engineering problem in large part because automatic speech recognition (ASR) and understanding technology are error-prone. More specifically, speech recognition accuracy is relatively good for constrained speech limited to, for example, digits, place-names, or short commands, but accuracy degrades rapidly as the domain language becomes less constrained. Furthermore, as spoken dialog systems become more complex, not only do the demands on the speech recognition and understanding components increase, but also user behaviour becomes less predictable. Thus, as task complexity increases, overall there is a rapid increase in uncertainty, and principled methods of dealing with this uncertainty are needed in order to make progress in this research area.

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*	......

Table 1: Sample conversation with a spoken dialog system affected by misunderstandings in the CMU Communicator spoken dialog system, taken from (Bohus and Rudnicky, 2002). *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 an illustration of the effects of speech recognition errors, consider the example conversation shown in Table 1, taken from (Bohus and Rudnicky, 2002). The system shown here allows the user to take control of the conversation wherever reasonably possible. In turn 3, the machine asks "What's your full name?" and in turn 4, the user replies with their name, but is misrecognized as saying "Athens in Akron". Since the machine does not insist on knowing the user's name, it infers that the user is taking control of the conversation and is asking about a flight. Hence, the system interprets "Athens in Akron" as the starting point of a flight booking dialog. This choice of interpretation causes the whole conversation to go off track and it is not until turn 13, nine turns later, that the conversation is progressing again.

This interaction illustrates the motivation for the three main approaches that have been developed in order to minimise the effects of errors and uncertainty in a spoken dialog system.

First, systems can attempt to identify errors locally using a *confidence score*: when a recognition hypothesis has a low confidence score, it can be ignored to reduce the risk of entering bad information into the dialog state. In the example above, if "Athens in Akron" were associated with a poor confidence score, then it could have been identified as an error and the system might have recovered sooner.

Second, accepting that misrecognitions will occur, their consequences can be difficult for human designers to anticipate. Thus systems can perform *automated planning* to explore the effects of misrecognitions and determine which sequence of actions are most useful *in the long run*. Consider turn 5 in the example above: the hand-crafted dialog manager chose to disambiguate "Athens", but automated planning might have revealed that it was better in the long term to first confirm that the user really did say "Athens", even though in the short term this might waste a turn.

Finally, accepting that some bad information will be entered into the dialog state maintained by the system, it seems unwise to maintain just one hypothesis for the current dialog state. A more robust approach would maintain *parallel state hypotheses* at each time-step. In turn 4 in the example above, the system could have maintained a second hypothesis for the current state – for example, in which the user said their name but was not understood. The system could have later exploited this information when a non-understanding happened in turn 7.

These three methods of coping with speech recognition errors – local use of confidence scores, automated planning, and parallel dialog hypotheses – can lead to improved performance, and confidence scores in particular are now routinely used in deployed systems. However, these existing methods typically focus on just a small part of the system and rely on the use of *ad hoc* parameter setting (for example, hand-tuned parameter thresholds) and pre-programmed heuristics. Most seriously, when these techniques are combined in modern systems, there is a lack of an overall statistical framework which can support global optimization and on-line adaptation.

In this paper, we will argue that a partially observable Markov decision process (POMDP[2]) provides such a framework. We will explain how a POMDP can be developed to encompass a complete dialog system, how a POMDP serves as a basis for optimization, and how a POMDP can integrate uncertainty in the form of statistical distributions with heuristics in the form of manually specified rules. To illustrate the power of the POMDP formalism, we will show how each of the three approaches above represents a special case of the more general POMDP model. Further, we provide evidence of the potential benefits of POMDPs through experimental results obtained from simulated dialogs. Finally, we address scalability and argue that whilst the

[2] usually read as "Pom D P"

computational issues are certainly demanding, tractable implementations of POMDP-based dialog systems are feasible.

The paper is organized as follows. Section 1 begins by reviewing POMDPs and then shows how the state space of a POMDP can be factored to represent a spoken dialog system in a way which explicitly represents the major sources of uncertainty. Next section 2 shows how each of the three techniques mentioned above – parallel dialog hypotheses, local confidence scoring, and automated planning – are naturally subsumed by the POMDP architecture. Section 3 discusses the advantages of POMDPs using a combination of illustrative dialogs and experimental simulation, including simulations with user models estimated from real dialog data. Finally, section 3.4 concludes by highlighting the key challenge of scalability and suggests two methods for advancing POMDP-based spoken dialog systems.

1. Casting a spoken dialog system as a POMDP

In this section we will cast a spoken dialog system as a POMDP. We start by briefly reviewing POMDPs. Then, we analyze the typical architecture of a spoken dialog system and identify the major sources of uncertainty. Finally, we show how to represent a spoken dialog system as a POMDP. In this discussion extensive use is made of influence diagrams and Bayesian inference – readers unfamiliar with these topics are referred to texts such as (Jensen, 2001).

1.1. Review of POMDPs

Formally, a POMDP is defined as a tuple $\{S, A, T, R, O, Z, \lambda, b_0\}$ where *S* is a set of states describing the agent's world; *A* is a set of actions that an agent may take; *T* defines a transition probability $P(s' \mid s, a)$; *R* defines the expected (immediate, real-valued) reward $r(s,a)$; *O* is a set of observations the agent can receive about the world; and *Z* defines an observation probability, $P(o' \mid s', a)$; λ is a geometric discount factor $0 \le \lambda \le 1$; and b_0 is an initial belief state $b_0(s)$. A POMDP can be depicted as an influence diagram, as in Figure 1.

The POMDP operates as follows. At each time-step, the world is in some unobserved state $s \in S$. Since s is not known exactly, a distribution over states is maintained called a "belief state," *b*, with initial belief state b_0. We write $b(s)$ to indicate the probability of being in a particular state *s*. Based on *b*, the machine selects an action $a \in A$, receives a reward $r(s,a)$, and transitions to (unobserved) state s', where s' depends only on s and a. The machine then receives an observation $o' \in O$ which is dependent on s' and a. At each time-step, the belief state distribution *b* is updated as follows:

$$
\begin{aligned}
b'(s') &= p(s' \mid o', a, b) \\
&= \frac{p(o' \mid s', a, b)\, p(s' \mid a, b)}{p(o' \mid a, b)}
\end{aligned}
$$

$$
= \frac{p(o' \mid s', a) \sum_{s \in S} p(s' \mid a, b, s)\, p(s \mid a, b)}{p(o' \mid a, b)}
$$

$$
= \frac{p(o' \mid s', a) \sum_{s \in S} p(s' \mid a, s) b(s)}{p(o' \mid a, b)}. \qquad (1)
$$

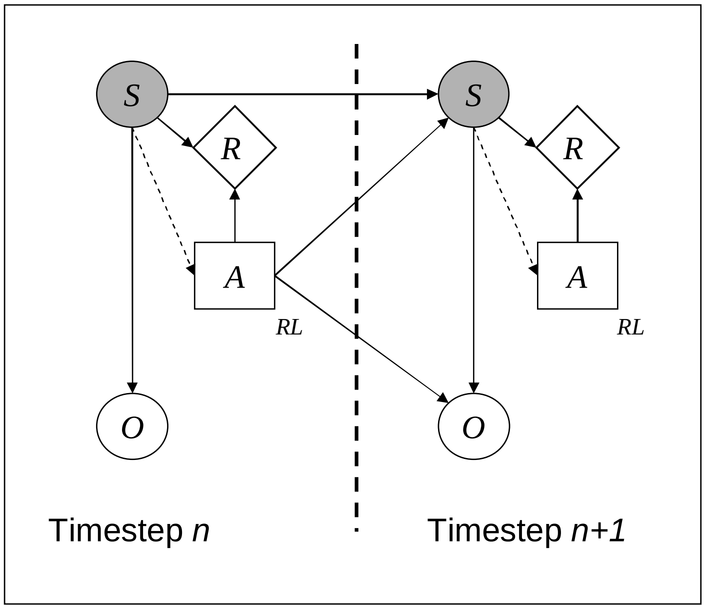


Figure 1: Influence diagram representation of a POMDP. Circles represent random variables; squares represent decision nodes; and diamonds represent utility nodes. Shaded circles indicate unobserved random variables, and un-shaded circles represent observed variables. Solid directed arcs indicate causal effect and dashed directed arcs indicate that a *distribution* is used (and not the actual unobserved value). The subscript *RL* indicates that actions are chosen using "Reinforcement learning," i.e., with the objective of maximizing the cumulative long-term reward.

The numerator consists of the observation function *Z*, transition matrix *T*, and current belief state *b*. The denominator is independent of s', and can be regarded as a normalization constant *k*; hence:

$$
b'(s') = k \cdot p(o' \mid s', a) \sum_{s \in S} p(s' \mid a, s) b(s)\,. \qquad (2)
$$

We refer to maintaining the value of *b* at each time-step as "belief monitoring." The value *b* has the useful property that it is a complete summary of the dialog history. More formally, for a given initial belief state b_0 and history $\{a_1, o_1, a_2, o_2, \ldots\}$, *b* provides a proper *sufficient statistic*: *b* is Markovian with respect to b_0 and $\{a_1, o_1, a_2, o_2, \ldots\}$. Thus, in effect, the update expressed in equation (2) is considering all possible (hidden) state transition histories when computing a new belief state, and planning algorithms need only consider *b* when choosing actions.

As mentioned above, at each time-step, the agent receives reward r_t. The cumulative, infinite-horizon, discounted reward is called the *return*:

$$\Theta = \sum_{t=0}^{\infty} \lambda^t r_t \tag{3}$$

where λ is the geometric discount factor, $0 \le \lambda \le 1$. The goal of the machine is to choose actions in such a way as to maximize the expected return $\mathrm{E}[\Theta]$ – i.e., to construct a *plan* called a *policy* which indicates which actions to take at each turn.[3] In general, a policy π can be viewed as a mapping from belief state to action $\pi(b) \in A$, and an *optimal policy* $\pi^*(b) \in A$ is one which maximizes $\mathrm{E}[\Theta]$.

In theory every belief point *b* could map to an arbitrary action $\pi(b)$, and for this reason finding optimal policies for POMDPs is in general intractable. In practice, $\pi^*(b)$ rarely maps to an arbitrary action and rather an optimal policy is a *partitioning* of belief space into a finite number of regions. Even so, exact algorithms such as the Witness algorithm (Kaelbling et al., 1998) rarely scale to problems with more than about 10 actions, states, and observations.[4] However, effective approximate solutions do exist. A review of POMDP optimization techniques is beyond the scope of this work; however, it should be noted that a family of approximate optimization techniques called *point-based value iteration* has been demonstrated to provide tractable solutions for a variety of real-world problems.[5] Standard (exact) value iteration computes a so-called value function *V(b)* which provides an estimate of the expected total reward that can be achieved from any point *b* in belief space given some policy π. Value iteration is a recursive process which leads to an estimate of *V*(b)*, the value function corresponding to the optimal policy π^*. Exact value iteration involves searching the whole of belief space; however, point-based value iteration heuristically selects a small set of representative belief points, and then iteratively applies value updates to just those points, achieving a significant speed-up (Pineau et al., 2003; Spaan and Vlassis, 2005).

In general, value iteration methods for POMDPs produce a collection of *n* vectors $\upsilon_n(s)$ each of dimensionality $|S|$ and an array of corresponding actions $\beta(n)$. Each vector $\upsilon_n(s)$ indicates the (long-term) value of taking a particular action $\beta(n) \in A$ in state *s*. By taking an expectation over belief space, we can find regions where action $\beta(n)$ is optimal – i.e., a policy can be produced from $\upsilon_n(s)$ and $\beta(n)$ as:

$$\pi(b) = \beta\left(\arg\max_{n} \sum_{s \in S} \upsilon_n(s)\, b(s) \right) \tag{4}$$

[3] In this work, we will assume that a planner has a model of the system dynamics – i.e., *T*, *R*, and *Z* are known or can be estimated from training data. In other words, we will focus on POMDPs which use *model-based* learning, as opposed to *experience-based learning.*

[4] Technically the method scales with the complexity of the optimal policy and not (necessarily) with the number of states, actions, and observations, but in practice the complexity of the optimal policy can not be predicted, and the number of states, actions, and observations is a useful heuristic.

[5] See (Murphy, 2000) for an overview of POMDP solution techniques.

Thus value iteration provides both a partitioning of belief space into regions corresponding to optimal actions as well as the expected return of taking that action.

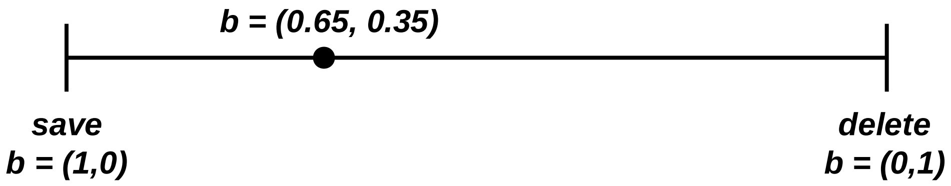


Figure 2: Belief space in a POMDP with two states, *save* and *delete*, which correspond to hidden user goals. At each time-step, the current belief state is a point on this line segment. The ends of the line segment represent certainty in the current state. The belief point shown is the initial belief state.

To illustrate how this POMDP framework is used in a spoken dialog system, an example will now be presented in some detail. This example concerns a very simple voicemail application which although very limited, nevertheless demonstrates the key properties of the POMDP approach. Later in the paper, we will consider the various issues which arise when scaling up the POMDP framework to handle more sophisticated applications.

In this example, users listen to voicemail messages and at the end of each message, they have two choices – *save* or *delete* the message. We refer to these as the user's goals and since the system does not *a priori* know which goal the user desires, they are *hidden* goals. For the duration of the interaction relating to each message, the user's goal is fixed and the POMDP-based dialog manager is trying to guess which goal the user has. Figure 2 shows a graphical depiction of belief space – since there are only two states, belief space can be shown as a line segment. In this depiction, the ends of the segment (in general called "corners") represent certainty. For example, *b = (1,0)* indicates certainty that *s* = *save*. Intermediate points represent varying degrees of certainty.

The machine has only three available actions: it can *ask* what the user wishes to do in order to infer his or her current goal, or it can *doSave* or *doDelete* and move to the next message. When the user responds to a question, it is decoded as either the observation $\overline{\textit{save}}$ or $\overline{\textit{delete}}$.[6] However, since speech recognition errors can corrupt the user's response, these observations cannot be used to deduce the user's intent with certainty. If the user says *save* then an error may occur with probability 0.2, whereas if the user says *delete* then an error may occur with probability 0.3. Finally, since the user wants *save* more often than *delete,* the *initial belief state* is set to indicate the prior (0.65, 0.35), and it is reset to this value after each *doSave* or *doDelete* action via the transition function.

The machine receives a large positive reward (+5) for getting the user's goal correct, a very large negative reward (-20) for taking the action *doDelete* when the user wanted *save* (since the user may have lost important information), and a smaller but still significant negative reward (-10) for taking the action *doSave* when the user wanted *delete* (since the user can always delete the message later). There is also a small negative reward for taking the *ask* action (-1), since all else being equal the machine

[6] The bar above *save* and *delete* indicates that these are *observations* – i.e., noisy, possibly erroneous indications of the user's goal.

should try to make progress to its goal as quickly as possible. The transition dynamics of the system are shown in Tables 2, 3 and 4.[7]

		s′	
a	***s***	*save*	*delete*
ask	*save*	1	0
	delete	0	1
doSave	*save*	0.65	0.35
	delete	0.65	0.35
doDelete	*save*	0.65	0.35
	delete	0.65	0.35

Table 2: Transition function $p(s' \mid s, a)$ for the example voicemail spoken dialog POMDP. The state *s* indicates the user's goal as each new voicemail message is encountered.

		o′	
a	***s′***	$\overline{save}$	$\overline{delete}$
ask	*save*	0.8	0.2
	delete	0.3	0.7
doSave	*save*	0.5	0.5
	delete	0.5	0.5
doDelete	*save*	0.5	0.5
	delete	0.5	0.5

Table 3: Observation function $p(o' \mid s', a)$ for the example voicemail application. Note that the observation o' only conveys useful information following an *ask* action

	s	
a	*save*	*delete*
ask	-1	-1
doSave	+5	-10
doDelete	-20	+5

Table 4: Reward function $r(s, a)$ for the example voicemail application. The values encode the dialog design criteria where it is assumed that deleting wanted messages should carry a higher penalty than saving unwanted messages, and where time wasting by repeatedly asking questions should be discouraged.

For a POMDP problem of this size, it is possible to produce an exact solution and here we have used the Witness algorithm (Kaelbling et al., 1998). Figure 3 shows the optimal

[7] Readers may recognize this as a variation of the *Tiger* problem cast into the spoken dialog domain (Cassandra et al., 1994).

policy. In the regions of belief space close to the corners (where certainty is high), the policy chooses *doSave* or *doDelete*; in the middle of belief space (where certainty is low) it chooses to gather information with the *ask* action. Further, since the penalty for wrongly choosing *doDelete* is worse than for wrongly choosing *doSave*, the *doDelete* region is smaller i.e., it requires more certainty than when the user's goal is *save*.

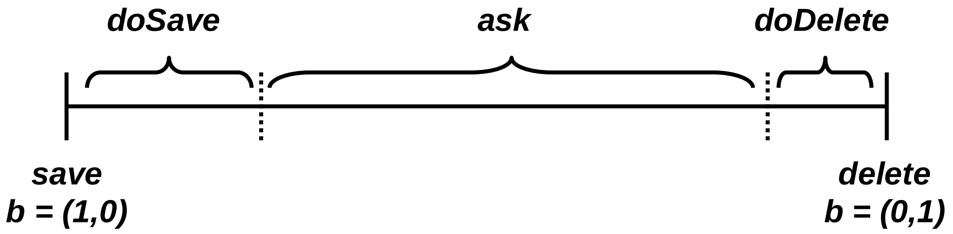


Figure 3: Optimal policy for the example voicemail spoken dialog system POMDP.

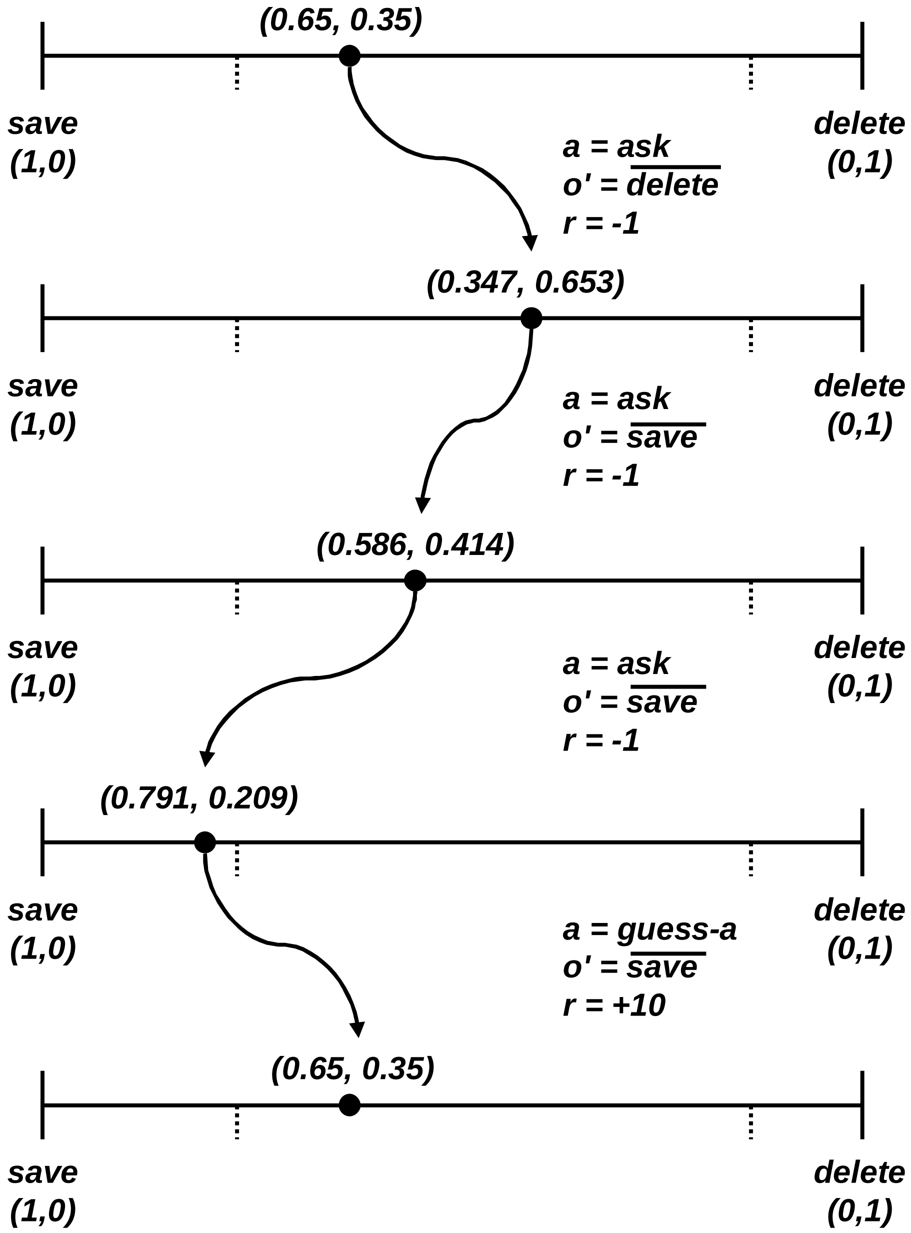


Figure 4: Evolution of the belief state in the example voicemail spoken dialog system POMDP. The dashed lines show the partition policy, given in Figure 3. At each time-step, the point *b* is updated using equation (2). Note that a recognition error is made after the first "ask" action.

Figure 4 shows an example conversation between the user and a machine executing the optimal policy. At each time-step, the machine action and the observation are used to update the belief state as in Eq. (2). Actions are selected depending on the partition which contains the current belief state. The first response is misrecognised as $\overline{\textit{delete}}$, moving the belief state towards the *delete* corner. However, since belief remains in the central region where uncertainty is high, the machine continues to ask the user what to do. After two successive (correct) $\overline{\textit{save}}$ observations, the belief state moves into the *doSave* region, the message is saved and the belief state transitions back to the prior state. The total reward for processing this message is +7.[8]

The key idea illustrated by this example is that the dialog system can never be certain of exactly what the user intends. This is true in human-human dialogs, but it is particularly true in human-machine dialogs where the existence of recognition errors greatly exacerbates the uncertainty. The sequence of machine actions dictated by the optimal POMDP policy guarantees that when averaged over a large number of dialogs, no other policy would achieve a greater reward. Hence, provided that the chosen reward function accurately reflects the dialog design criteria, the POMDP framework provides a principled approach to spoken dialog system design and optimisation.

Although the voicemail example illustrates the general approach to representing a spoken dialog system within the POMDP framework, it nevertheless sidesteps a number of important issues. In particular, models of how the user's goal evolves, how the user reacts, and how the speech recognition corrupts the user's actions need to be represented. In addition, some dialog history needs to be captured. To deal with this, the state space must be factored to allow the user's goal, the user's intention and relevant dialog history to be separated.

1.2. A factored state-space representation for spoken dialog systems

The architecture of a spoken dialog system is shown in Figure 5 (Young, 2000).[9] In this depiction, the user has some internal state S_u which corresponds to a goal that a user is trying to accomplish. Also, from the user's viewpoint, the dialog history has state S_d which indicates, for example, what the user has said so far, what the machine has confirmed, etc. Based on the user's goal prior to each turn, the user takes some *communicative action* (also called *intention*) A_u. A_u might correspond to a speech act, dialog act, or a parse structure. The user renders A_u as an audio signal Y_u by speaking. The *speech recognition and language understanding* component then takes the audio signal Y_u and produces two outputs: first, $\tilde{A}_u$, which is a noisy estimate of the user's *action* A_u; and *C* which represents a confidence score which provides an indication of the reliability of the recognition result $\tilde{A}_u$.[10] $\tilde{A}_u$ and *C* are then passed to the *dialog model*, which maintains an internal state S_m which tracks (from the machine's perspective) the state of the conversation.

[8] For simplicity we've ignored the geometric discount factor in this calculation, which would reduce this figure slightly.

[9] This figure makes several simplifications but conveys the concepts important to present purposes. Readers interested in the details of the recognition, understanding, generation, and text-to-speech components are referred to texts such as (Jurafsky and Martin, 2000) or survey articles such as (Glass 1999).

[10] In practice estimation of $\tilde{A}_u$ is usually performed in 2 stages – first a string of words is produced, then these words are parsed to extract $\tilde{A}_u$. This detail is not important for the purposes of this paper.

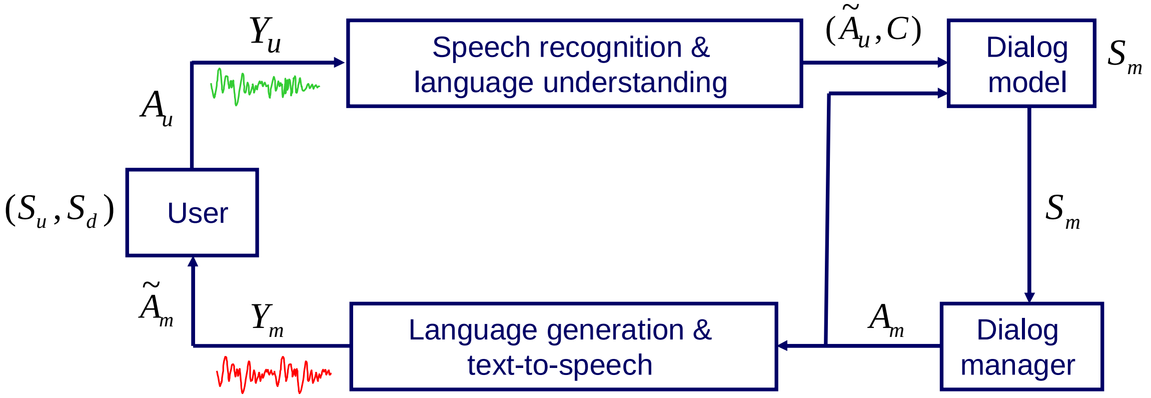


Figure 5: Typical architecture of a spoken dialog system

S_m is then passed to the *dialog manager*, which decides what action A_m the machine should take. A_m is converted to an audio response Y_m by the *language generation and text-to-speech* component, and it is also passed back to the dialog model so that S_m may track both user and machine actions. The user listens to Y_m, attempts to recover A_m, and as a result might update their goal state S_u and their interpretation of the dialog history S_d. The cycle then repeats.

One key reason why spoken dialog systems are challenging to build is that $\tilde{A}_u$ will contain recognition errors: i.e., it is frequently the case that $\tilde{A}_u \neq A_u$. As a result, the user's action A_u, the user's state S_u, and the dialog history S_d are not directly observable and can never be known to the system with certainty. However, $\tilde{A}_u$ and C provide *evidence* from which A_u, S_u, and S_d can be inferred.

We are now ready to cast a spoken dialog system as a POMDP. First, the machine action A_m will be cast as the POMDP action A. In a POMDP, the POMDP state S expresses the unobserved state of the world and the above analysis suggests that this unobserved state can naturally be factored into three distinct components: the user's goal S_u, the user's action A_u, and the dialog history S_d. Hence, the factored POMDP state S is defined as:

$$s = (s_u, a_u, s_d) \tag{5}$$

and the system state S_m becomes the belief state b over s_u, a_u, and s_d:

$$s_m = b(s) = b(s_u, a_u, s_d) \tag{6}$$

The noisy recognition result $\tilde{A}_u$ and the confidence score C will then be cast as the SDS-POMDP observation O:

$$o = (\tilde{a}_u, c) \tag{7}$$

We will henceforth refer to this factored form as the SDS-POMDP.

To compute the transition function and observation function, a few intuitive assumptions will be made. First, substituting (5) into the transition function and decomposing, we obtain:

$$p(s' \mid s, a) = p(s'_u, s'_d, a'_u \mid s_u, s_d, a_u, a_m) \tag{8}$$

$$p(s' \mid s, a) = p(s'_u \mid s_u, s_d, a_u, a_m) p(a'_u \mid s'_u, s_u, s_d, a_u, a_m) p(s'_d \mid a'_u, s'_u, s_u, s_d, a_u, a_m) \,. \tag{9}$$

We then assume conditional independence as follows. The first term in (9), which we call the *user goal model*, indicates how the user's goal changes (or does not change) at each time-step. We assume that the user's goal at each time-step depends only on the previous goal and the machine's action:

$$p(s'_u \mid s_u, s_d, a_u, a_m) = p(s'_u \mid s_u, a_m) \tag{10}$$

The second term, which we call the *user action model*, indicates what actions the user is likely to take at each time step. We assume the user's action depends on their (current) goal and the preceding machine action:

$$p(a'_u \mid s'_u, s_u, s_d, a_u, a_m) = p(a'_u \mid s'_u, a_m) \,. \tag{11}$$

The third term, which we call the *dialog history model*, captures relevant historical information about the dialog. We assume this component has access to the most recent value of all variables:

$$p(s'_d \mid a'_u, s'_u, s_u, s_d, a_u, a_m) = p(s'_d \mid a'_u, s'_u, s_d, a_m) \,. \tag{12}$$

Substituting (10), (11) and (12) into (9) then gives the SDS-POMDP transition function:

$$p(s' \mid s, a) = p(s'_u \mid s_u, a_m) p(a'_u \mid s'_u, a_m) p(s'_d \mid a'_u, s'_u, s_d, a_m) \,. \tag{13}$$

From (5) and (7), the observation function of the SDS-POMDP becomes:

$$p(o' \mid s', a) = p(\tilde{a}'_u, c' \mid s'_u, s'_d, a'_u, a_m) \,. \tag{14}$$

The observation function accounts for the corruption introduced by the speech recognition and language understanding process, so we assume that the observation depends only on the action taken by the user:[11]

$$p(\tilde{a}'_u, c' \mid s'_u, s'_d, a'_u, a_m) = p(\tilde{a}'_u, c' \mid a'_u) \,. \tag{15}$$

The two equations (13) and (15) represent a statistical model of a spoken dialog system. The transition function allows future behaviour to be predicted and the observation function provides the means for inferring the hidden system state from observations. The models themselves have to be estimated of course. The user goal model and the user action model (the first two components of Eq. (13)) will typically be estimated from a corpus of annotated interactions. For example, conditional distributions over user dialog acts can be estimated given a machine dialog act and a user goal. To appropriately cover all of the conditions, the corpus would need to include variability in the strategy

[11] This implicitly assumes that the same recognition grammar is always used. For systems where the grammar is switched at each turn, the dependence on a_m should be retained.

employed by the machine – for example, using a Wizard-of-Oz framework with a simulated ASR channel (Stuttle et al., 2004).

The dialog history model can either be estimated from data, handcrafted, or replaced by a deterministic function representing information state update rules as in for example (Larsson and Traum, 2000). Thus the SDS-POMDP system dynamics enable both probabilities estimated from corpora and hand-crafted heuristics to be incorporated. This is a very important aspect of the SDS-POMDP framework in that it allows deterministic programming to be incorporated in a natural way.

The observation function can be estimated from a corpus or derived analytically using a phonetic confusion matrix, language model, etc. (Deng et al., 2003; Stuttle et al., 2004).

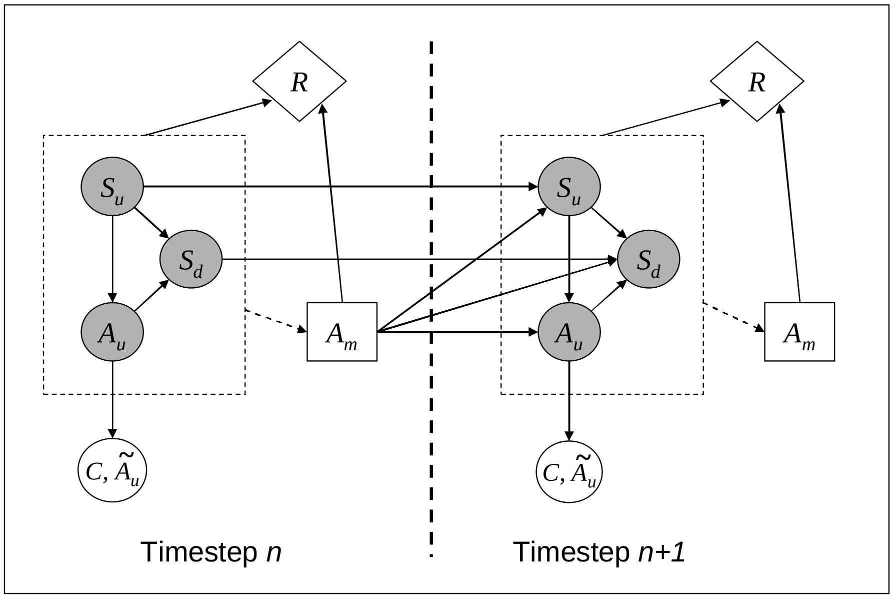


Figure 6: Influence diagram representation of the SDS-POMDP model. The dashed box indicates the composite state s which is comprised of three components, s_u, s_d, and a_u (see text for a complete definition of variables). The dashed line from the dashed box to a_m indicates that the action a_m is a function of the belief state $s_m = b(s_u, a_u, s_d)$.

The reward function is not specified explicitly since it depends on the design objectives of the target system. The reward function is well-suited to encoding a variety of objectives. Expressing simple, single optimization metrics is straightforward – for example, the chances of successful closure could be maximized by setting a positive reward for successful closure, and a zero reward for information gathering actions. Alternatively, the number of turns to completion could be minimized by setting a uniform negative reward for all information gathering actions, and a zero reward for closure actions.

Of course in a spoken dialog system, multiple competing criteria are important, and often a system should strive to maximize the chances of successful closure while also minimizing the number of turns required to do so. To combine multiple optimization criteria into one metric, *weightings* between the criteria are needed, and in a POMDP these weightings are naturally expressed in the reward function. For example, the reward function can include components for successful and unsuccessful closure, abandonment, and per-turn penalties, and the ratios between these reward components specifies the relative cost of longer dialogs, user abandonment, unsuccessful closure, etc. Moreover, the per-turn penalties can be used to encourage dialog "appropriateness", for example by setting a higher per-turn penalty for confirming an item which has not been discussed yet.

Finally, given the definitions above, the belief state can be updated at each time step by substituting equations (13) and (15) into (2), and simplifying:

$$\begin{aligned} b'(s'_u, s'_d, a'_u) = \quad & k \cdot p(\tilde{a}'_u, c' \mid a'_u) \cdot p(a'_u \mid s'_u, a_m) \cdot \\ & \sum_{s_u \in S_u} p(s'_u \mid s_u, a_m) \sum_{s_d \in S_d} p(s'_d \mid a'_u, s'_u, s_d, a_m) \sum_{a_u \in A_u} b(s_u, s_d, a_u). \end{aligned} \quad (16)$$

The summations over $s = (s_u, a_u, s_d)$ predict a new distribution for s' based on the previous values weighted by the previous belief. For each assumed value of a'_u, the leading terms outside the summation scale the updated belief by the probability of the observation given a'_u and the probability that the user would utter a'_u given the user's goal and the last machine output. Fig. 6 shows the influence diagram depiction of the SDS-POMDP, which clearly shows these dependencies. This figure will also be useful later for making comparisons between the SDS-POMDP representation and other approaches to dialog management.

	Standard POMDP	**SDS-POMDP**
State set	S	(S_u, A_u, S_d)
Observation set	O	$(\tilde{A}_u, C)$
Action set	A	A_m
Transition function	$p(s' \mid s, a)$	$p(s'_u \mid s_u, a_m) p(a'_u \mid s'_u, a_m) p(s'_d \mid a'_u, s_d, a_m)$
Observation function	$p(o' \mid s', a)$	$p(\tilde{a}'_u, c' \mid a'_u)$
Reward function	$r(s, a)$	$r(s_u, a_u, s_d, a_m)$
Belief state	$b(s)$	$b(s_u, a_u, s_d)$

Table 5: Summary of SDS-POMDP components.

For ease of reference, Table 5 summarises the expansion of terms in a standard POMDP to give the SDS-POMDP needed to characterise a spoken dialog system.

2. POMDPs and existing architectures

As described in the previous section, the SDS-POMDP model allows the dialog management problem to be cast in a statistical framework. It is therefore particularly well-suited to coping with the uncertainty inherent in spoken dialog systems. In this section, three existing techniques for handling uncertainty in an SDS will be reviewed: maintaining multiple dialog states, local use of confidence scores, and automated planning. In each case, it will be shown that the SDS-POMDP model provides an equivalent solution but in a more principled way which admits global parameter optimisation from data. Indeed, it will be shown that each of these existing approaches represents a simplification or special case of the SDS-POMDP model.

2.1. POMDPs and parallel state hypotheses

Traditional dialog management schemes maintain (exactly) one dialog state $s_m \in S_m$, and when a recognition error is made, s_m may contain erroneous information. Although designers have developed *ad hoc* techniques to avoid dialog breakdowns such as allowing a user to "undo" system mistakes, the desire for an inherently robust approach remains. A natural approach to coping with erroneous evidence is to maintain multiple hypotheses for the correct dialog state. Similar to a *beam search* in a hidden Markov model, maintaining many possible dialog states allows a system to explore many paths through a dialog, always allowing for the possibility that each piece of evidence is an error. In this section, we briefly review two techniques for maintaining multiple dialog hypotheses: greedy decision theoretic approaches and an M-Best list.

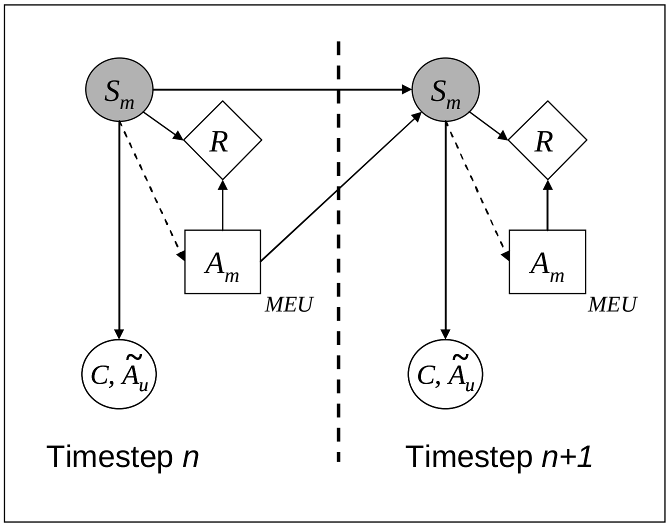


Figure 7: View of a spoken dialog system as a greedy decision theoretic process. Action A_m is selected to maximize the expected immediately utility *R*, indicated by the subscript *MEU* ("Maximum Expected Utility"). The dashed line indicates that A_m is a function of the *distribution* over S_m, rather than its actual (unobserved) value.

Greedy decision theoretic approaches construct an influence diagram as shown in Figure 7. The structure of the network is identical to a POMDP: the system state S_m is a belief state over hidden variables, such as s_u, a_u, and s_d. The dashed line in the figure from S_m to A_m indicates that A_m is chosen based on the *distribution* over S_m rather than its actual (unobserved) value. As with a POMDP, a reward (also called a utility) function is used to select actions – however, greedy decision theoretic approaches differ from a POMDP in *how* the reward is used to selection actions. Unlike a POMDP, in which machine actions are chosen to maximize the cumulative *long-term* reward, greedy decision theoretic approaches choose the action which maximizes the *immediate* *reward*.[12] In other words, the POMDP is performing planning, whereas the greedy decision theoretic approach is not. As such, action selection is certainly tractable for real-world dialog problems, and greedy decision theoretic approaches have been successfully demonstrated in real working dialog systems (Horvitz and Paek, 2000; Paek et al., 2000).

However, whether the dialog manager explicitly performs planning or not, a successful dialog must make progress to some long-term goal. In greedy decision theoretic approaches, a system will make long-term progress toward a goal only if the reward metric has been carefully crafted. Unfortunately, crafting a reward measure which accomplishes this is a non-trivial problem and in practice encouraging a system to make progress to long-term goals inevitably requires some hand-crafting resulting in the need for *ad hoc* iterative tuning.

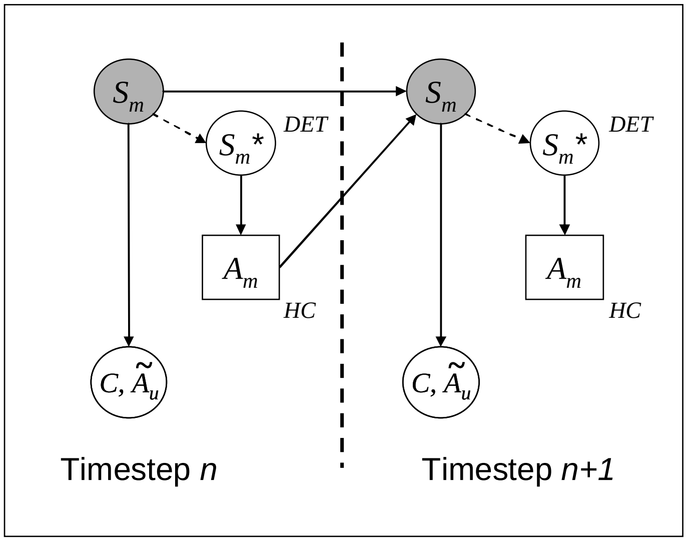


Figure 8: Influence diagram showing multiple state hypotheses. S_m^* takes the value of the state S_m with the highest probability mass at each time-step. The superscript *DET* indicates that the variable S_m^* is not random but is rather a *deterministic* function of its inputs.

An alternative to the greedy decision theoretic approach is to still maintain multiple dialog hypotheses but select actions by considering only the *top* dialog hypothesis, using a handcrafted policy as in conventional heuristic SDS design practice. This approach is referred to as the *M-Best list* approximation, and it is shown graphically in Figure 8. In this figure, the superscript *DET* indicates that the node S_m^* is not random but rather takes on a *deterministic* value for known inputs, and here S_m^* is set to the state S_m with the most probability mass. The *M-best list* approach has been used to build real dialog systems and shown to give performance gains relative to an equivalent single-state system (Higashinaka et al., 2003).[13]

[12] As such, a greedy decision theoretic method could also be classified as an "automatic action selection" method – the focus here is maintaining multiple dialog state hypotheses.

[13] This work makes two further approximations – first, for computational efficiency, a "beam" of approximately 30 states is maintained rather than all possible states. Second, a "scoring" mechanism is used as an approximation to a proper probability score.

The M-best approximation can be viewed as a POMDP in which action selection is hand-crafted, and based only on the most likely dialog state. When cast in these terms, it is clear that an M-best approximation makes use of only a fraction of the available state information since considering only the top hypothesis may ignore important information in the alternative hypotheses such as whether the next-best is very similar or very different to the best hypothesis. Hence, even setting aside the use of *ad hoc* hand-crafted policies, the *M-best list* approach is clearly sub-optimal. In contrast, since the SDS-POMDP constructs a policy which covers belief space, it naturally considers all alternative hypotheses.

2.2. POMDPs and local use of confidence scores

Most speech recognition engines annotate their output word hypotheses $\tilde{W}$ with confidence scores $p(\tilde{W} | Y_u)$ and modern systems can compute this measure quite accurately (Evermann and Woodland, 2000; Kemp and Schaff, 1997; Moreno et al., 2001). Subsequent processing in the speech understanding components will often augment this low level acoustic confidence using extra features such as parse scores, prosodic features, dialog state, etc (Bohus et al., 2001; Gabsdil and Lemon, 2004; Hirschberg et al. 2001; Krahmer et al., 1999, 2001; Litman et al., 2001; Pao et al., 1998).

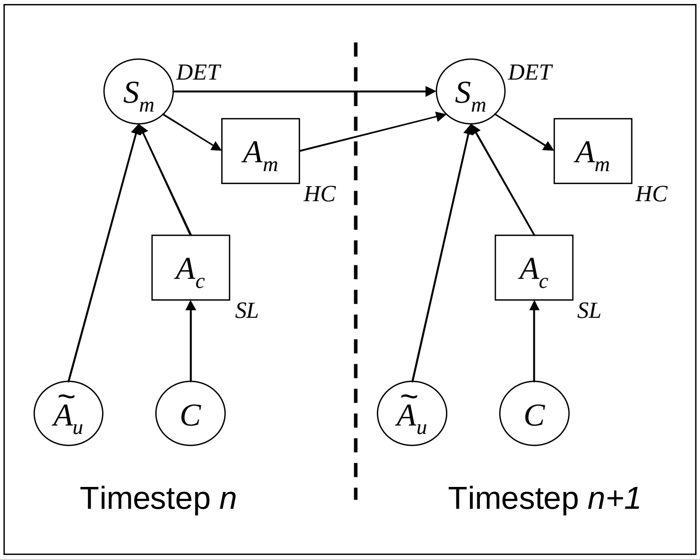


Figure 9: Influence diagram showing how a confidence score is typically incorporated into a spoken dialog system. Node *C* is a random variable representing confidence score. A_c may take on values such as *{hi, low}*, *{explicit-confirm, implicit-confirm, reject}*, etc.

For the purposes of a dialog system, the essential point of a confidence score is that it provides an overall indication of the reliability of the hypothesized user dialog act $\tilde{a}_u$. Traditional systems typically incorporate confidence scores by specifying a *confidence threshold* c_{thresh} which implements an *accept/reject* decision for an $\tilde{a}_u$: if $c > c_{thresh}$ then $\tilde{a}_u$ is deemed reliable and accepted; otherwise it is deemed unreliable and discarded. In practice any value of c_{thresh} will still result in classification errors, so c_{thresh} can be viewed as implementing a trade-off between the cost of a false-negative (rejecting an accurate $\tilde{a}_u$) and the cost of a false-positive (accepting an erroneous $\tilde{a}_u$).

Figure 9 shows how a spoken dialog system with a confidence score can be expressed in an influence diagram. A_c is a decision node that indicates the "confidence bucket" or action relative to the confidence score – for example, *{hi, low}* or *{accept, reject}*. A_c is typically trained using a corpus of examples and supervised learning, indicated by the subscript *SL* on the node A_c.[14] This "confidence bucket" is then incorporated into the dialog state using hand-crafted update rules – i.e., $S'_m = f(S_m, A_m, A'_c, \tilde{A}'_u)$. As above, the superscript *DET* on the node S_m indicates that S_m takes on a *deterministic* value – i.e., for a known set of inputs, it yields exactly one output. Based on the updated dialog state S_m, the policy determines which action to take. The dialog manager is implemented with hand-crafted rules, indicated by the subscript *HC* on the A_m decision node.

Figure 9 also highlights key differences between a traditional system with a confidence score and the SDS-POMDP model. In both models, $\tilde{A}_u$ and *C* are regarded as observed random variables. However, in traditional approaches, a *hard* and *coarse* decision is made about the validity of $\tilde{A}_u$ via the decision A_c. The decision implemented in A_c is non-trivial since there is no principled way of setting the confidence threshold c_{thresh}. In practice a developer will look at expected accept/reject figures and use intuition. A slightly more structured approach would attempt to assign costs to various outcomes (e.g., cost of a false-accept, cost of a false reject, etc.) and choose a threshold accordingly. However, these costs are specified in immediate terms, whereas in practice the decisions have long-term effects (e.g., subsequent corrections) which are difficult to quantify, and which vary depending on context. Indeed, when long-term costs are properly considered, there is evidence that values for optimal confidence thresholds are not at all intuitive: one recent study found that for many interactions, the optimal confidence threshold was *zero* – i.e., any recognition hypothesis, no matter how poorly scored, should be accepted (Bohus and Rudnicky, 2005b).

By contrast, the SDS-POMDP is a generative model in which confidence score is modelled as a continuous observed random variables. Note how in Figure 9, the confidence score is viewed as a functional input, whereas in the POMDP (Figure 6), it is viewed as an observed output from a distribution. In this way, the SDS-POMDP never makes hard accept/reject decisions about evidence it receives, but rather uses the confidence score to perform inference over all possible user actions A_u. Further, the explicit machine dialog state S_m used in traditional approaches is challenged to maintain a meaningful confidence score history since typically if a value of $\tilde{A}_u$ is rejected, that information is discarded.[15] By contrast, the SDS-POMDP aggregates all information

[14] A_c could also be handcrafted – the key point that confidence score is quantized.

[15] A small body of work has attempted to identify "good dialogs" by looking at features over multiple turns, but the classification scheme – good dialog vs. bad dialog – is even coarser than accept/reject decisions (Litman and Pan, 2000), (Langkilde et al., 1999). D.

over time *including conflicting evidence* via a belief state, properly accounting for the reliability of each observation in cumulative terms. Finally, whereas accept/reject decisions in a traditional system are taken based on local notions (often human intuitions) of utility, in the SDS-POMDP actions are selected based on expected long-term reward – note how Figure 6 explicitly includes a reward component, absent from Figure 9.

In summary, local use of confidence scores in traditional hand-crafted SDSs does add useful information, but acting on this information in a way which serves long-term goals is non-trivial. A traditional SDS with a confidence score can be viewed as an SDS-POMDP with a number of simplifications: one dialog state is maintained rather than many; accept/reject decisions are used in place of parallel dialog hypotheses; and actions are selected based on a hand-crafted strategy rather than selected to maximize a long-term reward metric.

2.3. POMDPs and automated action selection

Choosing which action a_m a spoken dialog system should take in a given situation is a difficult task since it is not always obvious what the long-term effect of each action will be. Hand-crafting dialog strategies can lead to unforeseen dialog situations, requiring expensive iterative testing to build good systems. Such problems have prompted researchers to investigate techniques for choosing actions automatically and in this section, the two main approaches to automatic action selection will be considered: supervised learning, and Markov decision processes.

As illustrated graphically in Figure 10, supervised learning attempts to estimate a direct mapping from machine state S_m to action A_m given a corpus of training examples. It can be thought of as a simplification of the SDS-POMDP model in which a single state is maintained, and in which actions are learnt from a corpus. Setting aside the limitations of maintaining just one dialog state and the lack of explicit forward planning, using supervised learning to create a dialog policy is problematic since collecting a suitable training corpus is very difficult for three reasons.

Firstly, using human-human conversation data is not appropriate because it does not contain the same distribution of understanding errors, and because human-human turn-taking is much richer than human-machine dialog. As a result, human-machine dialog exhibits very different traits than human-human dialog (Doran et al., 2001; Moore and Browning, 1992). Secondly, while it would be possible to use a corpus collected from an existing spoken dialog system, supervised learning would simply learn to approximate the policy used by that spoken dialog system and an overall performance improvement would therefore be unlikely. Thirdly, a corpus could be collected for the purpose, for example, by running Wizard-of-Oz style dialogs in which the wizard is required to select from a list of possible actions at each step (Bohus and Rudnicky, 2005a; Lane et al., 2004) or encouraged to pursue more free-form interactions (Skantze, 2003; Williams and Young, 2004). However, in general such collections are very costly, and tend to be orders of magnitude too small to support robust estimation of generalized action selection.

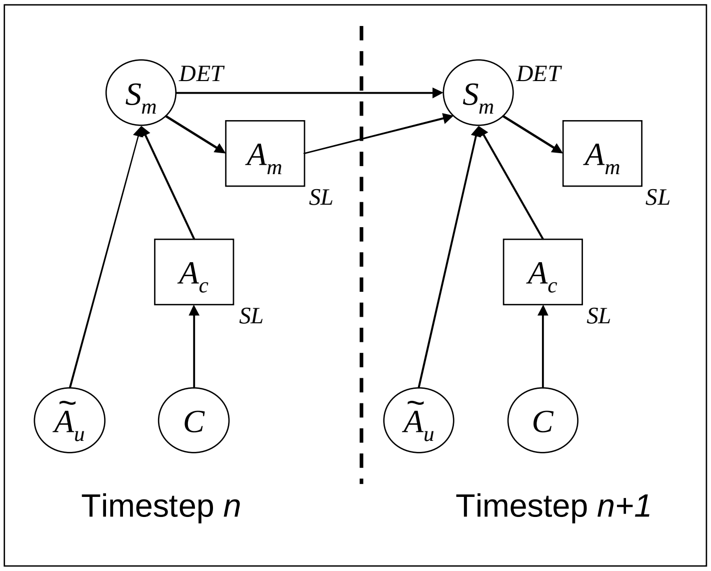


Figure 10: Supervised learning for action selection. The node A_m has been trained using supervised learning on a corpus of dialogs (indicated with the *SL* subscript). The *DET* superscript on S_m indicates that this node is deterministic.

Fully-observable Markov decision processes (usually just called Markov decision processes, or MDPs) take a very different approach to automated action selection. As their name implies, a Markov decision process is a simplification of a POMDP in which the state is fully observable. This simplification is shown graphically in Figure 11. In an MDP, $\tilde{A}_u$ is again regarded as a random observed variable and S'_m is a deterministic function of S_m, A_m, $\tilde{A}'_u$, and A'_c. Since at a given state s_m a host of possible observations $\tilde{a}_u$ are possible, planning is performed using a transition function – i.e. $P(s'_m | s_m, a_m)$. Like POMDPs, MDPs choose actions to maximize a long-term cumulative sum of rewards: i.e., they perform planning. Unlike POMDPs, the current state in an MDP is known, so a policy is expressed directly as a function of state *s*: $\pi(s) \in A$. This representation is discrete (a mapping from discrete states to discrete actions), and as a result, MDPs are usually regarded as a more tractable formalism than POMDPs. Indeed, MDPs enjoy a rich literature of well-understood optimization techniques and have been applied to numerous real-world problems (Sutton and Barto, 1998).

By allowing designers to specify rewards for desired and undesired outcomes (e.g., successfully completing a task, a caller hanging up, etc) without specifying explicitly how to achieve each required goal, much of the tedious “handcrafting” of dialog design is avoided. Moreover, unlike the supervised learning approach to action selection, MDPs make principled decisions about the long-term effects of actions, and the value of this approach has been demonstrated in a number of research systems. For example, in the ATIS Air Travel domain, Levin et al. constructed a system to optimize the costs of querying the user to restrict (or broaden) their flight search, the costs of presenting too many (or too few) flight options, and the costs of accessing a database (Levin and Pieraccini, 1997; Levin et al., 1998, 2000). In addition, researchers have sought to find optimal initiative, information presentation, and confirmation styles in real dialog systems (Singh et al., 2002; Walker et al., 1998). MDP-based spoken dialog systems have also given rise to a host of work in user modelling and novel training/optimization techniques (Denecke et al., 2004; Goddeau and Pineau, 2000; Henderson et al., 2005; Pietquin 2004; Pietquin and Renals, 2002; Scheffler and Young, 2002).

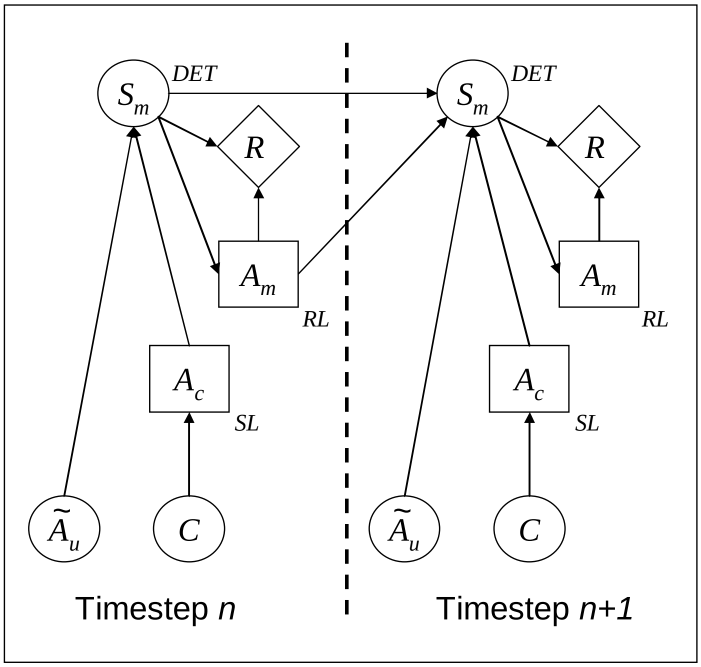


Figure 11: Depiction of an MDP used for dialog management. The action A_m is chosen to maximize the sum of rewards *R* over time.

A key weakness of MDPs is that they assume that the current state of the world is known exactly and this assumption is completely unfounded in the presence of recognition errors. The impact of this becomes clear when the MDP transition function is calculated:[16]

$$P(s'_m \mid s_m, a_m) = \sum_{\tilde{a}'_u} P(\tilde{a}'_u \mid s_m, a_m) P(s'_m \mid s_m, a_m, \tilde{a}'_u) \qquad (17)$$

To compute the transition function properly, an estimate of $P(\tilde{a}_u \mid s_m, a_m)$ is required, but in reality $\tilde{A}_u$ depends critically (via A_u) on S_u. Dialog designers try to ensure that S_m closely models S_u, but as errors are introduced and the two models diverge, the effects of the dependence of $\tilde{A}_u$ on a hidden variable increasingly violate the Markov assumption expressed in $P(s'_m \mid s_m, a_m)$, compromising the ability of the MDP to produce good policies. While there exist sophisticated learning techniques (such as *eligibility traces*) which attempt to partially overcome the fact that the user's state is not fully observable (Scheffler and Young, 2002), in simple terms, as speech recognition errors become more prevalent, theory predicts that POMDPs will perform better than MDPs by an increasing

[16] In this calculation, a'_c has been omitted for clarity.

margin. As will be shown below, the results of simulation studies support this theoretical prediction.

In summary, from a theoretical standpoint, maintaining multiple dialog hypotheses, confidence scoring, and automated planning can all be viewed as special cases or simplifications of a POMDP. Of course, contemporary spoken dialogue systems may employ more than one of these techniques, but a POMDP is unique in providing a unified statistical framework that supports global optimization. For example, an MDP may include a confidence bucket in its state space, but there is no straightforward way to search for optimal confidence threshold settings (i.e., those which maximize expected return), save a brute-force search. Further, some combinations of these techniques have only been demonstrated with a POMDP – for example, as far as the authors are aware, the only systems in the literature which both maintain multiple hypotheses for the dialog state and perform forward planning are POMDPs.

In the next section, we illustrate the benefits of these theoretical advantages concretely through example dialogs and experimental simulations.

3. Empirical support for the SDS-POMDP framework

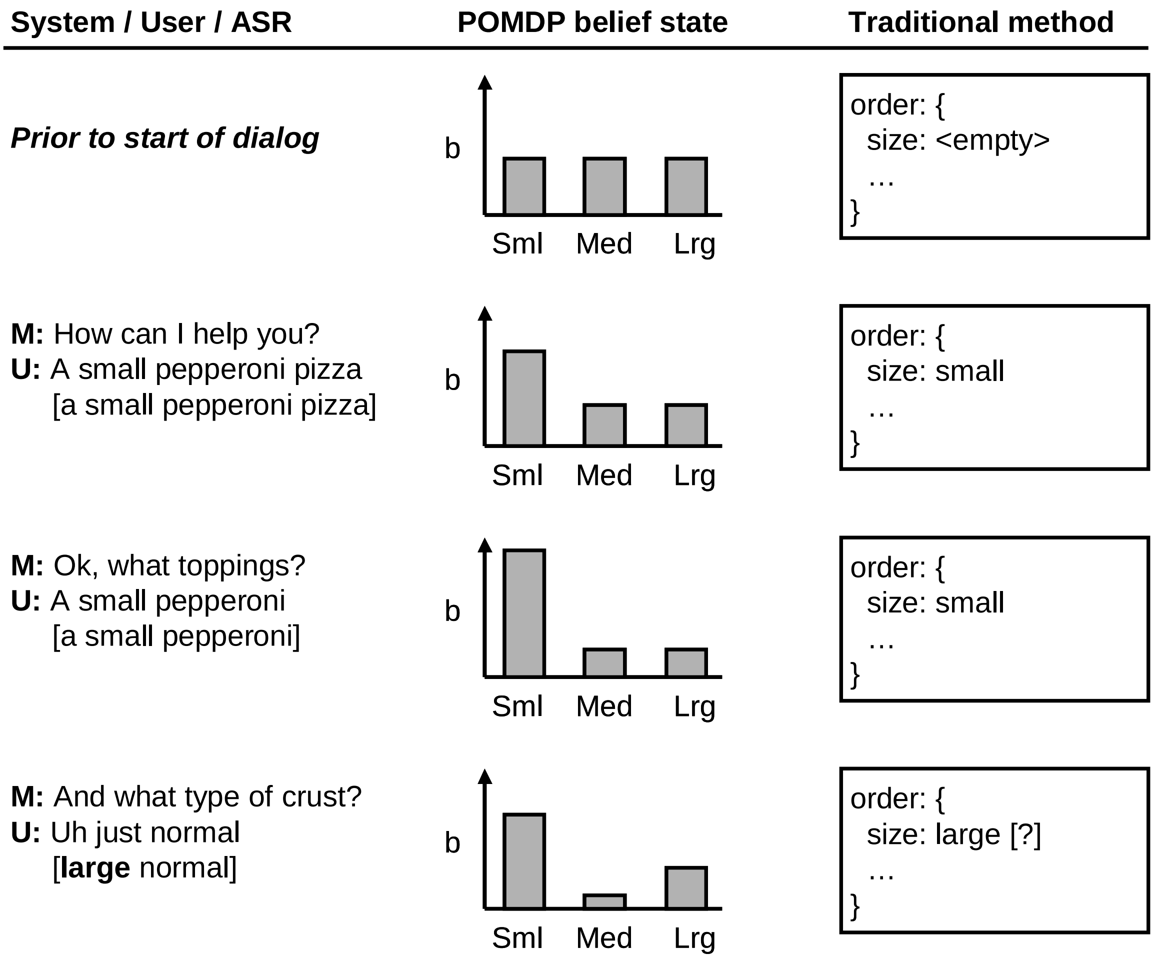
Section 1 has shown how POMDPs can be viewed as a principled theoretical approach to dialog management under uncertainty and section 2 has demonstrated that existing approaches to handling uncertainty are subsumed and generalised by the SDS-POMDP framework. In this section, the practical advantages of utilising the SDS-POMDP framework are demonstrated through example interactions and simulation experiments.

3.1. Benefits of parallel state hypotheses

A central claim of this paper is that because POMDPs maintain parallel dialog state hypotheses, they are able to cope better with speech recognition errors. In this section, we will first discuss how multiple dialog hypotheses add robustness to speech recognition errors. In doing so, we will also explain how the SDS-POMDP model takes proper account of a user model.

To begin illustrating this claim, consider a spoken dialog system with no confidence scoring and which makes speech recognition errors with a fixed error rate. For this example, which is in the pizza ordering domain, it is assumed that all cooperative user actions are equally likely: i.e., there is no effect of a user model. An example conversation with such a system is shown in Figure 12. In this figure, the first column shows interactions between the user and the machine. Text in brackets shows the recognized text (i.e., $\tilde{A}_u$). The middle column shows a portion of a POMDP representation of the user's goal. The last column shows how a traditional dialog model might track this same portion of the dialog state with a frame-based representation

This conversation illustrates how multiple dialog hypotheses are more robust to errors by properly accounting for conflicting evidence. In this example, the frame-based representation must choose whether to change its value for the *size* field or ignore new evidence; by contrast, the POMDP easily accounts for conflicting evidence by shifting belief mass. Intuitively, a POMDP naturally implements a "best two out of three" strategy.



Prior to start of dialog

```
order: {
  size: <empty>
  …
}
```

M: How can I help you?
U: A small pepperoni pizza
[a small pepperoni pizza]

```
order: {
  size: small
  …
}
```

M: Ok, what toppings?
U: A small pepperoni
[a small pepperoni]

```
order: {
  size: small
  …
}
```

M: And what type of crust?
U: Uh just normal
[**large** normal]

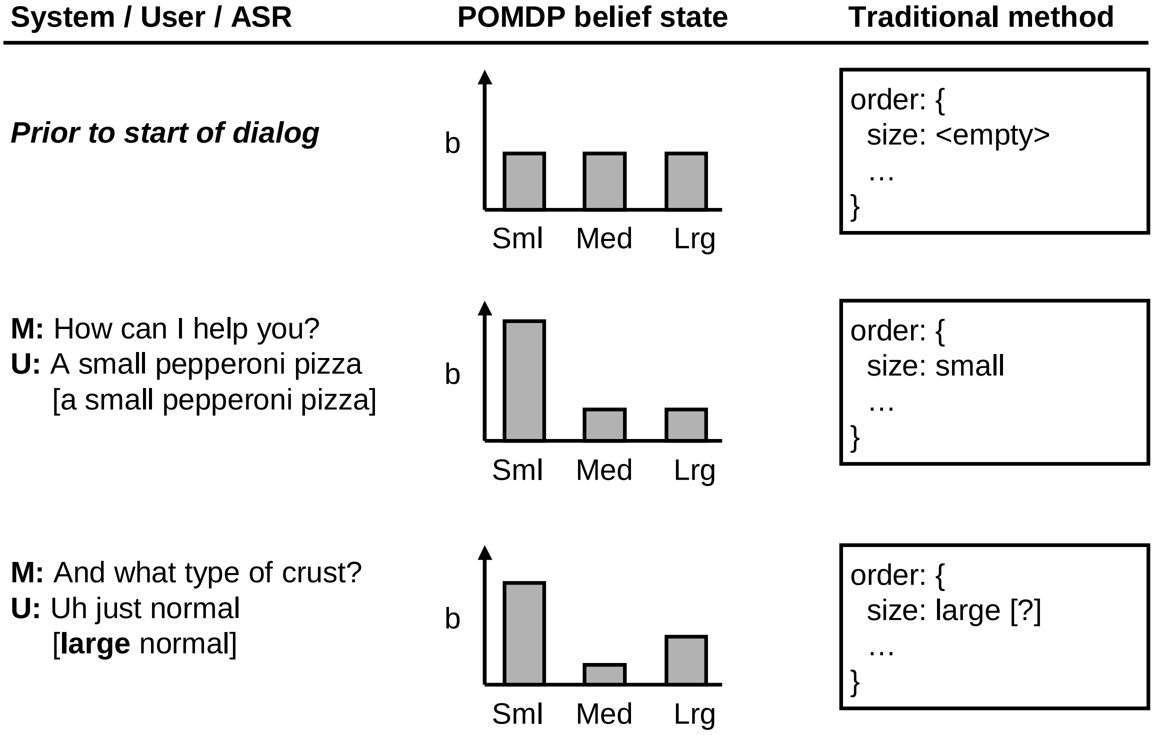
```
order: {
  size: large [?]
  …
}
```

Figure 12: Example conversation with a spoken dialog system illustrating the benefit of maintaining multiple dialog state hypotheses. This example is in the pizza ordering domain. The left column shows the machine and user utterances, and the recognition results from the user's utterance is shown in brackets. The center column shows a portion of the POMDP belief state; *b* represents the belief over a component of the user's goal (pizza size). The right-hand column shows a typical frame-based method which is also tracking this component of the user's goal. Note that a speech recognition error is made in the last turn – this causes the traditional method to absorb a piece of bad information, whereas the POMDP belief state is more robust. In this example no account is taken of which user actions are more or less likely, or of confidence score – see below for illustrations of these elements.

A POMDP is further improved with the addition of a user model which indicates how a user's goal S_u changes over time, and what actions A_u the user is likely to take in a given situation. For example, consider the dialog shown in Figure 13. In this figure, a user model informs the likelihood of each recognition hypothesis $\tilde{A}_u$ given S_u and A_m.

In this example, the machine asks for the value of one slot, and receives a reply. The system then asks for the value of a second slot, and receives a value for that slot *and* an *inconsistent* value for the first slot.

In the traditional frame-based dialog manager, it is unclear how this evidence should be incorporated – should the new information replace the old information, or should it be ignored? If the frame is extended to allow conflicts, how can they be resolved? Finally, how can the fact that the new evidence is less likely than the initial evidence be incorporated? By contrast, in the SDS-POMDP the belief state update is scaled by the likelihood predicted by the user model. In other words, the POMDP takes minimal (but non-zero) account of very unlikely user actions it observes, and maximal account of very likely actions it observes.



Prior to start of dialog

order: {
  size: <empty>
  …
}

M: How can I help you?
U: A small pepperoni pizza
[a small pepperoni pizza]

order: {
  size: small
  …
}

M: And what type of crust?
U: Uh just normal
[**large** normal]

order: {
  size: large [?]
  …
}

Figure 13: Example conversation with a spoken dialog system illustrating the benefit of an embedded user model. In the POMDP, for the first recognition, the observed user's response is very likely according to the user model. The result is a large shift in belief mass toward the *Sml* value. In the second recognition, providing information about the size is predicted as being less likely; as a result, the observed response *Lrg* (which happens to be a speech recognition error) is given less weight, and the final POMDP belief state has more mass on *Sml* than *Lrg*. By contrast, the traditional method must choose whether to update the state with *Sml* or *Lrg.*

To test these intuitions experimentally, a test-bed dialog simulation experiment was created (Williams et al., 2005a). The goal of the experiment was to quantify the benefits of multiple dialog hypotheses and the embedded user model, and explore the effects of different speech recognition errors rates. This assessment is made by comparing the performance of a POMDP to an MDP which (as described in section 3.3) does not maintain multiple hypotheses.

The test-bed simulation is in the travel domain. A simulated user is trying to buy a ticket to travel from one city to another city. The machine asks the user a series of questions, and then "submits" the ticket purchase request, ending the dialog. The machine may also choose to "fail," abandoning the dialog. To make the system relatively straightforward to optimize, there are just three cities in the test-bed problem. The machine has 16 actions available, including *greet, ask-from/ask-to, confirm-to-x/confirm-from-x, submit-x-y,* and *fail.* The user's goal specifies the user's desired itinerary, and the dialog history s_d indicates (from the user's perspective) whether the *from* place and *to* place have not been specified, are unconfirmed, or are confirmed. The user's action and the speech recognition result are drawn from the set *x*, *from-x*, *to-x*, *from-x-to-y*, *yes*, *no*, and *null*, where in all cases *x* and *y* indicate cities. These state components yield a total of 1945 states.

a_m	s'_u	**Description**	a'_u	$p(a'_u \mid s'_u, a_m)$
greet	from x to y	User wasn't paying attention	*null*	0.100
		User says both places	*from-x-to-y*	0.540
		User says just "from" place	*from-x*	0.180
		User says just "to" place	*to-y*	0.180
		All other user actions	(all others)	0.000
ask-from	from x to y	User wasn't paying attention	*null*	0.100
		User says just the name of the place	*x*	0.585
		User says the name of the place preceded by "from"	*from-x*	0.225
		User says both places	*from-x-to-y*	0.090
		All other user actions	(all others)	0.000
confirm-to-y	from x to y (NB - the system has the right hypothesis)	User wasn't paying attention	*null*	0.100
		User says just "yes"	*yes*	0.765
		User says the item that was being confirmed	*y*	0.101
		User says the item being confirmed, with the "to" preposition	*to-y*	0.034
		All other user actions	(all others)	0.000

Table 6: Extracts from the hand-crafted user model employed in simulation experiments.

In the test-bed problem the user has a fixed goal for the duration of the dialog, and we define the *user goal model* accordingly. We define the *user action model* to include a variable set of responses – for example, the user may respond to *ask-to/ask-from* with *x*, *to-x/from-x*, or *from-x-to-y*. The probabilities in the user action model were chosen such that the user provides cooperative but varied responses, and sometimes does not respond at all. The probabilities were handcrafted, selected based on experience performing usability testing with slot-filling dialog systems. A portion of the user model parameters is given in Table 6.

We define the observation function to encode the probability of making a speech recognition error to be p_{err}, and define the observation function as:

$$p(\tilde{a}'_u \mid a'_u) = \begin{cases} 1 - p_{err} & if\ \tilde{a}'_u = a'_u \\ \dfrac{p_{err}}{|A_u| - 1} & if\ \tilde{a}'_u \neq a'_u \end{cases} \qquad (18)$$

Below we will vary p_{err} to explore the effects of speech recognition errors.

The reward measure includes components for both task completion and dialog “appropriateness” and reflects the intuition that behaving inappropriately or even abandoning a hopeless conversation early are both less severe than submitting the user's goal wrong. The reward assigns -3 for confirming a field before it has been referenced by the user; -5 for taking the *fail* action; +10 or -10 for taking the *submit-x-y* action when the user’s goal is *(x,y)* or not, respectively; and -1 otherwise. This reward function expresses how trade-offs should be made between the system’s competing objectives of speed and accuracy – for example, this reward function indicates that a dialog which requires 15 turns to arrive at the correct answer (and receives $-1 \cdot 15 + 10 = -5$) obtains the same reward as one in which the system immediately abandons the interaction via the *fail* action (and receives -5). Thus if the planner determines that successful completion would require more than 15 turns, it will instead choose to immediately abandon the conversation and not waste the user’s time.[17]

POMDP optimization was performed with a variant of point-based value iteration called *Perseus* (Spaan and Vlassis, 2005).

An MDP was constructed to assess performance of a model which does not track multiple dialog states, and which does not make use of an explicit user model. The MDP was patterned on systems in the literature, for example (Pietquin, 2004)). The MDP was trained and evaluated through interaction with a model of the environment, which was formed from the POMDP transition, observation, and reward functions. This model of the environment takes an action from the MDP as input, and emits an observation and a reward to the MDP as output.

List of MDP states		
u-u	*o-u*	*c-u*
u-o	*o-o*	*c-o*
u-c	*o-c*	*c-c*
dialog-start		*dialog-end*

Table 7: The 11 MDP states used in the test-bed simulation. In the items of the form *x-y,* the first item refers to the *from* slot, and the second item refers to the *to* slot. *u* indicates *unknown*; *o* indicates *observed* but not confirmed; *c* indicates *confirmed.*

The MDP state contains components for each field which reflect whether, *from the standpoint of the machine,* a value has not been observed, a value has been observed but not confirmed, or a value has been confirmed. Two additional states – *dialog-start* and *dialog-end* – which were also in the POMDP state space, are included in the MDP state space for a total of 11 MDP states, shown in Table 7. The MDP was optimized using Watkins Q-Learning (Watkins, 1989).

Figure 14 shows the average return (i.e. total cumulative reward) for the POMDP and MDP solutions vs. the recognition error rate p_{err} ranging from 0.00 to 0.65. The (negligible) error bars for the MDP show the 95% confidence interval for the estimate of

[17] For clarity, this illustration has assumed that the discount factor γ is equal to 1.

the return assuming a normal distribution.[18] The POMDP and MDP perform equivalently for $p_{err} = 0$, and the return for both methods decreases consistently as p_{err} increases but the POMDP solution consistently achieves the larger return. Thus, in the presence of perfect recognition accuracy, there is no advantage to maintaining multiple dialog states, however, when errors do occur, the POMDP solution is always better and furthermore the difference in performance increases as p_{err} increases. This result confirms that the use of multiple dialog hypotheses and an embedded user model enable higher recognition error rates to be tolerated compared to the conventional single-state approach. A detailed inspection of the dialog transcripts confirmed that the POMDP is better at interpreting inconsistent information, agreeing with the intuition shown in Figure 12.

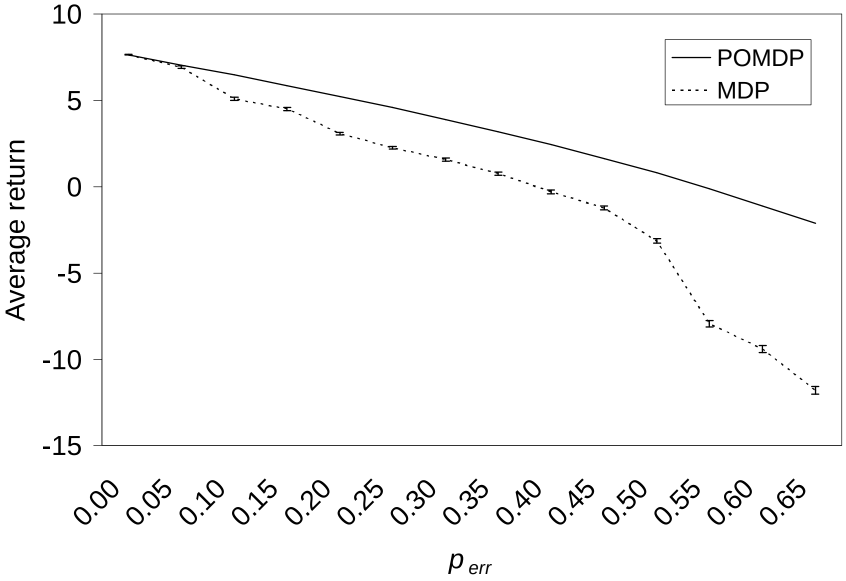


Figure 14: Expected or average return of the POMDP policy and the MDP baseline. Error bars show the 95% confidence interval

Although there is little related work in the literature, experiments by Roy et al also showed performance gains compared to a conventional MDP, using a simpler *Augmented MDP* in which planning is performed considering only the (discrete) best state, and the entropy of the belief state, (Roy et al., 2000).

3.2. Benefits of the POMDP approach to confidence scoring

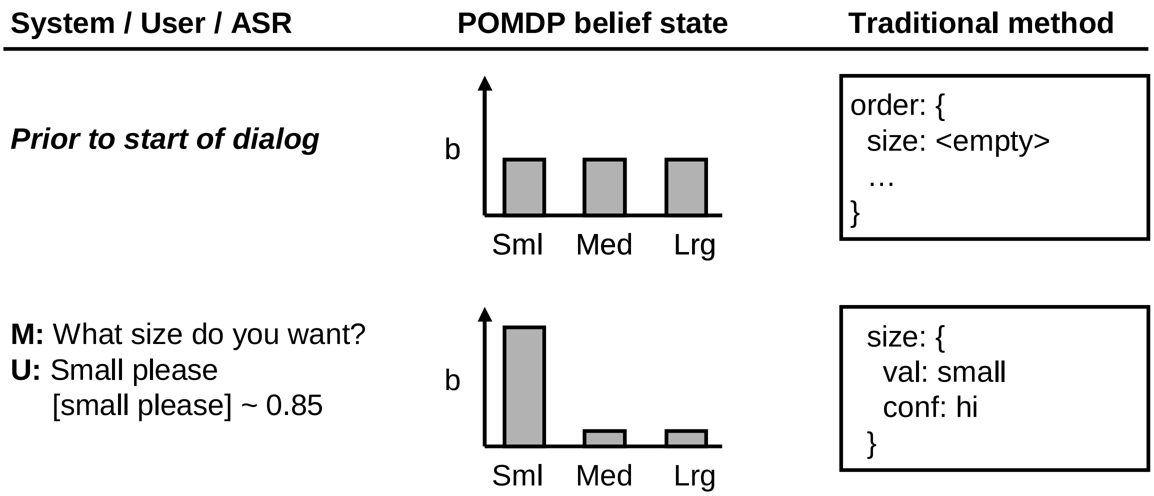
A second central claim of this work is that POMDPs provide a principled approach to confidence scoring.

To illustrate this claim, consider a spoken dialog system which makes use of a per-utterance confidence score which ranges from 0 to 1. Assume that all cooperative user actions are equally likely so that the effects of a user model can be disregarded. In the traditional version of this system with three confidence buckets {*reject, low, hi*}, suppose that a good threshold between *reject* and *low* has been found to be 0.4, and a good threshold between *low* and *hi* has been found to be 0.8.

[18] The POMDP value is exact and hence error bars aren't shown.

An example conversation is shown in Figure 15 in which the machine asks a question and correctly recognizes the response. In the traditional method, the confidence score of 0.85 is in the *high* confidence bucket, hence the utterance is accepted with “hi” confidence and the dialog state is updated accordingly. In the POMDP, the confidence score is incorporated into the magnitude of the belief state update.

Now consider the conversation in Figure 16, in which each of the recognitions is again correct, but the confidence scores are lower. In the traditional method, each confidence score falls into the “reject” confidence bucket, and nothing is incorporated into the dialog frame. In the POMDP-based system, however, the magnitude of the confidence score is incorporated into the belief update as above, although this time since the score is lower, each update shifts less belief mass.



Prior to start of dialog

```
order: {
  size: <empty>
  …
}
```

M: What size do you want?
U: Small please
[small please] ~ 0.85

```
size: {
  val: small
  conf: hi
}
```

Figure 15: Example conversation with a spoken dialog system illustrating a high-confidence recognition. The POMDP incorporates the magnitude of the confidence score by scaling the belief state update correspondingly. The traditional method quantizes the confidence score into a “bucket” such as *{reject, low, hi}*.

This second example illustrates two key benefits of POMDPs. First, looking *within* one time-step, whereas the traditional method creates a finite set of confidence buckets, the POMDP in effect utilizes an infinite number of confidence buckets and as a result the POMDP belief state is a lossless representation of a *single* confidence score. Second, looking *across* time-steps, whereas the traditional method is challenged to track aggregate evidence about confidence scores over time, a POMDP effectively maintains a cumulative confidence score over user goals. For the traditional method to approximate a cumulative confidence score, a policy which acted on a *historical* record of confidence scores would need to be devised, and it is quite unclear how to do this.

Moreover, the incorporation of confidence score information and user model information are complementary since they are separate product terms in the belief update equation (16). The probability $p(\tilde{a}'_u, c' \mid a'_u)$ reflects the contribution of the confidence score and the probability $p(a'_u \mid s'_u, a_m)$ reflects the contribution of the user model. The belief term $b(s_u, s_d, a_u)$ records the dialog history and provides the memory needed to accumulate evidence. This is in contrast to traditional approaches which typically have a small number of confidence score "buckets" for each recognition event, and typically log only the most recently observed "bucket". POMDPs have in effect infinitely many confidence score buckets *and* they aggregate evidence properly over time as a well-formed distribution over dialog states (including user goals).

To test these intuitions experimentally, the dialog management problem presented in Section 3.1 was extended to include a confidence score (Williams et al., 2005b). In the POMDP, the confidence score c is regarded as a continuous component of the observation, and in the MDP, the confidence score is quantized into "buckets" as is customarily done (Pietquin, 2004).

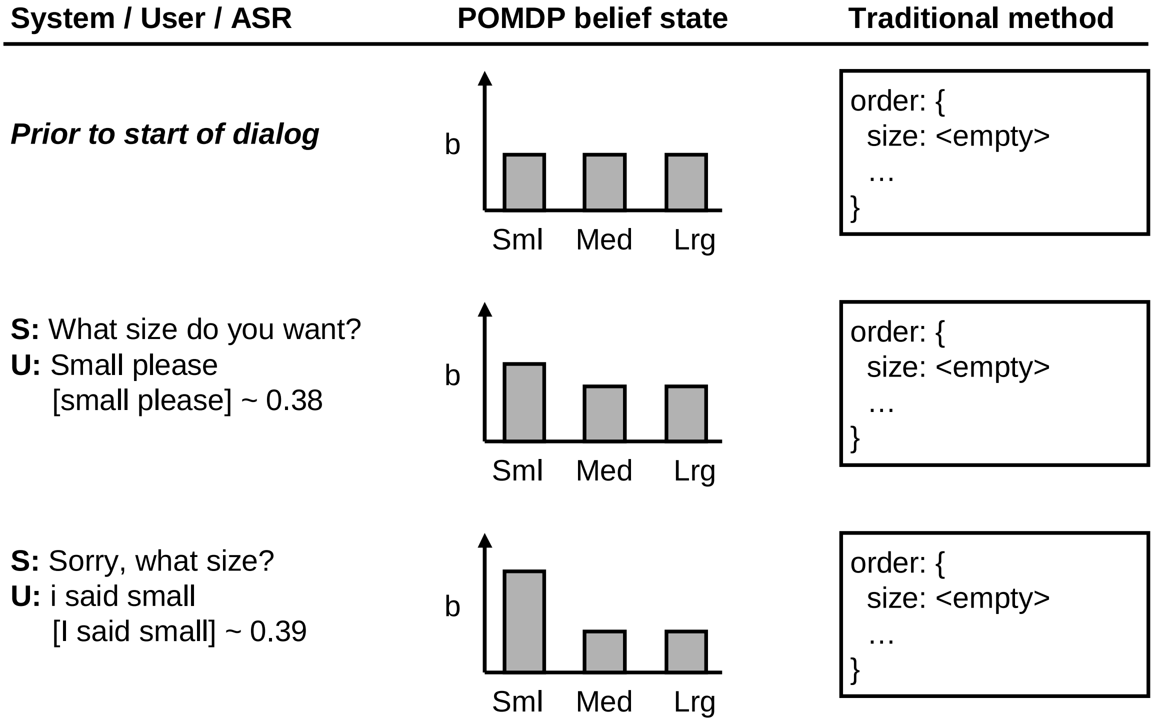


Figure 16: Example conversation with a spoken dialog system illustrating two successive low-confidence recognitions. In this example, both recognitions are correct. The POMDP incorporates the confidence score in the same way as shown in Figure 15, accumulating weak evidence. For the traditional method, both confidence scores are below the threshold of 0.40, and thus they are both ignored. In effect, the traditional method is ignoring possibly useful information.

In the POMDP, the observation function $p(\tilde{a}'_u, c' \mid a'_u)$ is in practice impossible to estimate directly from data, so it is decomposed into two distributions – one for "correct" recognitions and another for "incorrect" recognitions. In the test-bed problem we assume that all confusions are equally likely and occur with probability p_{err}, yielding:

$$p(\tilde{a}'_u, c' \mid a'_u) = \begin{cases} p_h(c') \cdot (1 - p_{err}) & if\ \tilde{a}'_u = a'_u \\ p_h(1 - c') \cdot \dfrac{p_{err}}{|A_u| - 1} & if\ \tilde{a}'_u \neq a'_u \end{cases} \tag{19}$$

where c' is defined on the interval [0,1], and $p_h(c')$ is an exponential probability density functions with slope determined by a parameter h. When $h = 0$, $p_h(c')$ is a uniform density and conveys no information; as h approaches infinity, $p_h(c')$ provides complete and perfect information. POMDP policy optimization was performed with a technique which admits continuous observations (Hoey and Poupart, 2005).

The MDP baseline was similarly extended to include M confidence buckets, patterned on systems in the literature, such as (Pietquin, 2004). Ideally the thresholds between confidence buckets would be selected so that they maximize average return; however, it is not obvious how to perform this selection – indeed, this is one of the weaknesses of the "confidence bucket" method. Instead, a variety of techniques for setting confidence score threshold were explored, and it was found that dividing the probability mass of the confidence score c evenly between buckets produced the largest average returns.

List of MDP-2 states						
u-u	u-o(l)	u-o(h)	u-c(l,l)	u-c(l,h)	u-c(h,l)	u-c(h,h)
o(l)-u	o(l)-o(l)	o(l)-o(h)	o(l)-c(l,l)	o(l)-c(l,h)	o(l)-c(h,l)	o(l)-c(h,h)
o(h)-u	o(h)-o(l)	o(h)-o(h)	o(h)-c(l,l)	o(h)-c(l,h)	o(h)-c(h,l)	o(h)-c(h,h)
c(l,l)-u	c(l,l)-o(l)	c(l,l)-o(h)	c(l,l)-c(l,l)	c(l,l)-c(l,h)	c(l,l)-c(h,l)	c(l,l)-c(h,h)
c(l,h)-u	c(l,h)-o(l)	c(l,h)-o(h)	c(l,h)-c(l,l)	c(l,h)-c(l,h)	c(l,h)-c(h,l)	c(l,h)-c(h,h)
c(h,l)-u	c(h,l)-o(l)	c(h,l)-o(h)	c(h,l)-c(l,l)	c(h,l)-c(l,h)	c(h,l)-c(h,l)	c(h,l)-c(h,h)
c(h,h)-u	c(h,h)-o(l)	c(h,h)-o(h)	c(h,h)-c(l,l)	c(h,h)-c(l,h)	c(h,h)-c(h,l)	c(h,h)-c(h,h)
	dialog-start				dialog-end	

Table 8: The 51 states in the "MDP-2" simulation. In the items of the form *x-y*, the first item refers to the *from* slot, and the second item refers to the *to* slot. *u* indicates *unknown*; *o* indicates *observed* but not confirmed; *c* indicates *confirmed*. *o(l)* means that the value was observed with *low* confidence; *o(h)* means that the value was observed with *high* confidence. *c(l,l)* means that both the value itself and the confirmation were observed with *low* confidence; *c(l,h)* means that the value was observed with low confidence and the confirmation was observed with high confidence, etc.

The MDP state was extended to include this confidence "bucket" information. Because the confidence bucket for each field (including its value and its confirmation) is tracked in the MDP state, the size of the MDP state space grows with the number of confidence buckets. For $M=2$, the resulting MDP called *MDP-2* has 51 states; this is shown in Table 8.[19] Watkins Q-learning was again used for MDP optimization.

Figure 17 shows the average returns for the POMDP and *MDP-2* solutions vs. p_{err} ranging from 0.00 to 0.65 for $h = 1$. The error bars show the 95% confidence intervals for the return assuming a normal distribution. Note that return decreases consistently as p_{err} increases for all solution methods, but the POMDP solutions attain larger returns than the MDP method at all values of p_{err}.[20]

We next explored the effects of varying the informativeness of the confidence score. Figure 18 shows the average returns for the POMDP method and the *MDP-2* method vs.

[19] For reference, *M=1* produces an MDP with 11 states, and *M=3* produces an MDP with 171 states.

[20] The *MDP-3* system was also created but we were unable to obtain better performance from it than we did from the MDP-2 system.

h for p_{err} = 0.3. The error bars show the 95% confidence interval for return assuming a normal distribution. The POMDP method outperforms the baseline MDP method consistently for a range of confidence score measures. This trend was also observed for a range of other values of p_{err} (Williams et al., 2005a). Note that increasing h increases the average return for all methods.

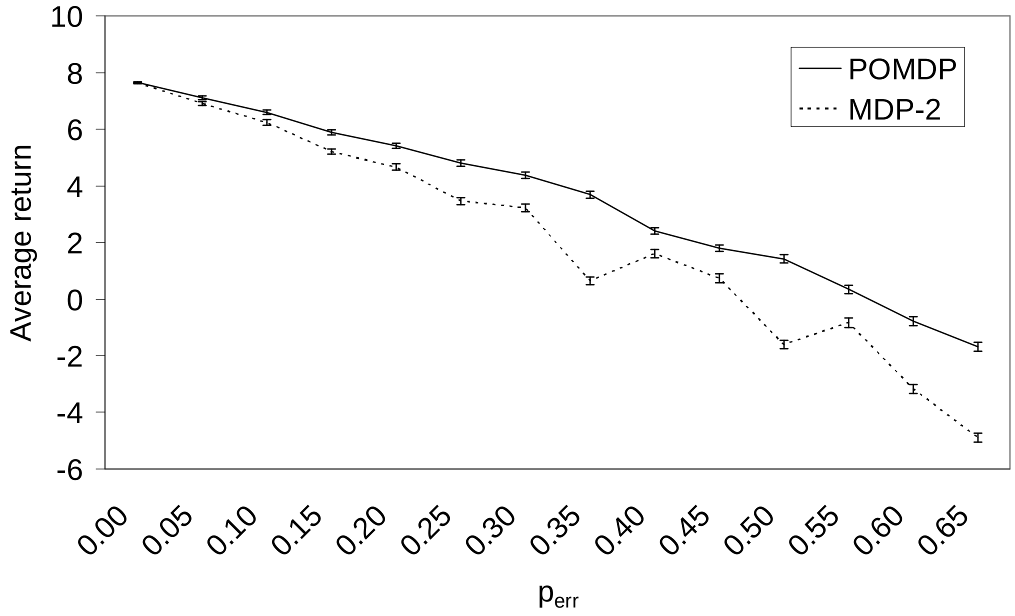


Figure 17: Average return for the *POMDP* and *MDP-2* methods for *h* = 1.

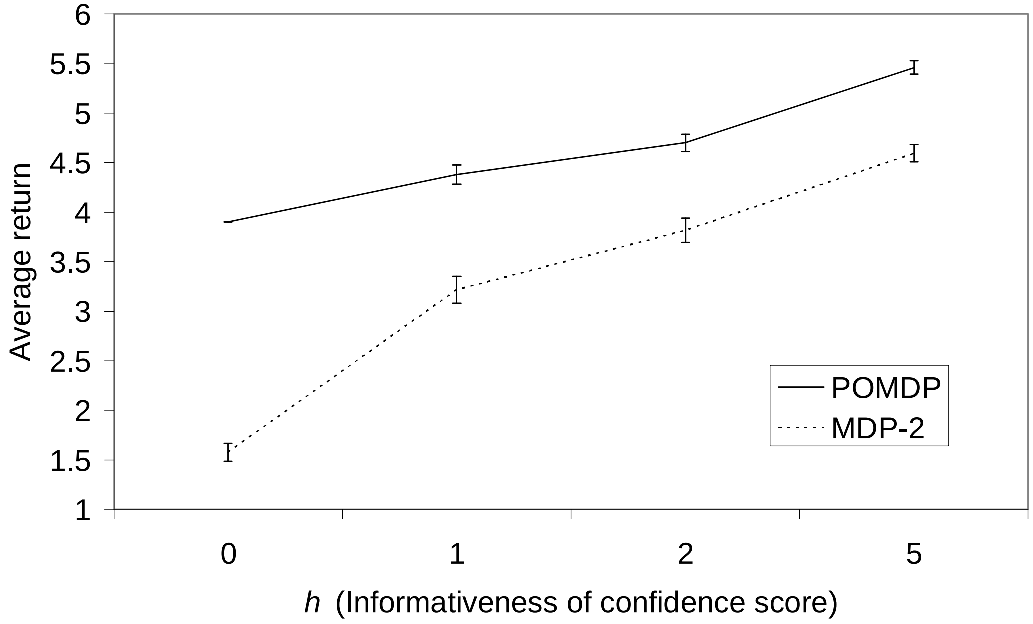


Figure 18: Average return vs. *h* (informativeness of confidence score) at p_{err} = 0.30 for the POMDP and *MDP-2* methods.

3.3. Benefits of automated planning

A third central claim of this work is that POMDPs provide a principled framework for automated planning. In this section we support this claim with two discussions. First, since there exists considerable expertise in *hand-crafting* spoken dialog systems, it is important to make comparisons with hand-crafted strategies. We show how these comparisons can be made and demonstrate the relative gains of POMDPs. Second, the benefits of planning (vs. not planning) for *automatically generated* dialog mangers are also addressed.

To compare a POMDP policy with a hand-crafted policy, first the *form* of POMDP policies must be considered. In the previous sections, we relied on the representation of a POMDP policy produced by value iteration – i.e., a value function, represented as a set of N vectors each of dimensionality $|S|$. A second way of representing a POMDP policy is as a "policy graph" which is a finite state controller consisting of N nodes and some number of directed arcs. Each controller node is assigned a POMDP action, and $\hat{\pi}(n)$ indicates the action associated with the *nth* node. Each arc is labelled with a POMDP observation, such that all controller nodes have exactly one outward arc for each observation. $l(n,o) \in N$ denotes the successor node for node n and observation o.

A policy graph is a general and common way of representing handcrafted dialog management policies (Pieraccini and Huerta, 2005). More complex handcrafted policies – for example, those created with rules – can usually be compiled into a (possibly very large) policy graph. A policy graph does not make the expected return associated with each controller node explicit; however, as pointed out by (Hansen, 1998), the expected return associated with each controller node can be found by solving a system of linear equations in υ:

$$\upsilon_n(s) = r(s,\hat{\pi}(n)) + \gamma \sum_{s' \in S} \sum_{o \in O} p(s' \mid s,\hat{\pi}(n))\, p(o \mid s',\hat{\pi}(n))\, \upsilon_{l(n,o)}(s') \qquad (20)$$

Solving this set of linear equations yields a set of vectors – one vector $\upsilon(s)$ for each controller node, $\upsilon_n(s)$. In words, Eq. (20) sets the value of a node equal to the immediate reward of taking that node's action $r(s,\hat{\pi}(n))$ plus the discounted expected future reward.

To find the expected value $V_n(b)$ of starting the controller in node n and belief state b we compute:

$$V_n(b) = \sum_{s \in S} \upsilon_n(s) b(s) \qquad (21)$$

Note that a human designer is free to define the controller however they wish: the controller may have any number of nodes, and its size is not linked to the size of the POMDP state space.

To illustrate policy graph evaluation, three handcrafted policies called *HC1*, *HC2*, and *HC3* were created for the spoken dialog problem presented above. Each of these policies encode strategies typically used by designers of spoken dialog systems. All of the handcrafted policies first take the action *greet*. *HC1* takes the *ask-from* and *ask-to* actions to fill the *from* and *to* fields, performing no confirmation. If no response is detected, *HC1* re-tries the same action. If *HC1* receives an observation which is inconsistent or nonsensical, it re-tries the same action. Once *HC1* fills both fields, it takes the corresponding *submit-x-y* action. A flow diagram of the logic used in *HC1* is shown in Figure 19.[21] *HC2* is identical to *HC1* except that if the machine receives an observation which is inconsistent or nonsensical, it immediately takes the *fail* action. *HC3* employs a similar strategy to *HC1* but extends *HC1* by confirming each field as it is collected. If the user responds with "*no*" to a confirmation, it re-asks the field. If the user provides inconsistent information, it treats the new information as "correct" and confirms the new information. Once it has successfully filled and confirmed both fields, it takes the corresponding *submit-x-y* action.

Figure 20 shows the expected return for the handcrafted policies and the optimized POMDP solution vs. the recognition error rate p_{err}. The optimized POMDP solution outperforms all of the handcrafted policies for all values of p_{err}. On inspection, conceptually the POMDP policy differs from the handcrafted policies in that it tracks conflicting evidence rather than discarding it. For example, whereas the POMDP policy can interpret the "best 2 of 3" observations for a given slot, the handcrafted policies can maintain only 1 hypothesis for each slot. As expected, the additional representational power of the automated solution is of no benefit in the presence of perfect recognition – note that where $p_{err} = 0$, HC1 and HC2 perform identically to the POMDP policy. It is interesting to note that *HC3,* which confirms all inputs, performs least well for all values of p_{err}. For the reward function used in the test-bed system, requiring 2 consistent recognition results (the response to *ask* and the response to *confirm*) gives rise to longer dialogs which outweigh the benefit of the increase in accuracy.

[21] Only the logic of HC1 is shown for clarity: the full controller uses actual city name values instead of the variables *X* and *Y,* resulting in a controller with 15 nodes. This type of expansion is typical of the "compilation" process mentioned above.

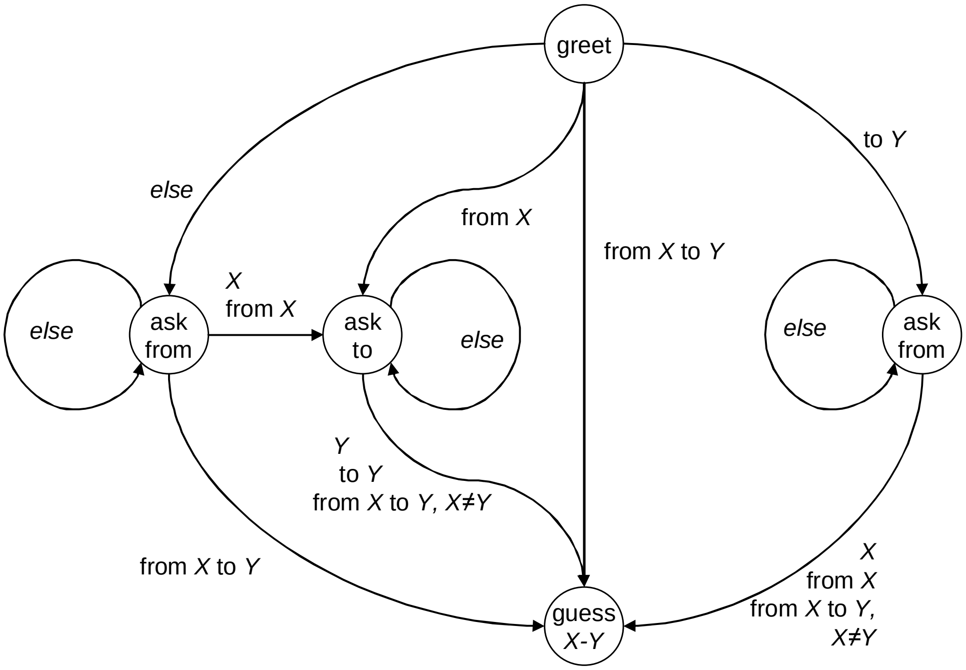


Figure 19: *HC1* handcrafted policy represented as a finite state controller. Node labels show the POMDP action to take for each node, and arcs show which POMDP observations cause which transitions. Note that the nodes in the diagram are entirely indepenent of the POMDP states.

Finally, we consider whether *planning* is beneficial to automatically generated dialog managers by comparing the performance of the POMDP to a greedy decision theoretic dialog manager (section 3.1) on the dialog problem described in section 4.1. This greedy dialog manager always takes the action with the highest expected immediate reward – i.e., unlike a POMDP, it is not performing planning. Both dialog managers were evaluated by simulating conversations and finding the average reward gained per dialog. Results are shown in Figure 21. The POMDP outperforms the greedy method by a large margin for all error rates. Intuitively, the POMDP is able to reason about the future and determine when gathering information will reap larger gains in the long term *even if* it incurs an immediate cost. More specifically, in this example, the POMDP gathers more information than the greedy approach. As a result, dialogs with the POMDP dialog manager are longer but the resulting increased cost is offset by correctly identifying the user's goal more often. In general, POMDPs are noted for their ability to make effective trade-offs between the (small) cost of gathering information, the (large) cost of acting on incorrect information, and rewards for acting on correct information (Cassandra et al., 1994).

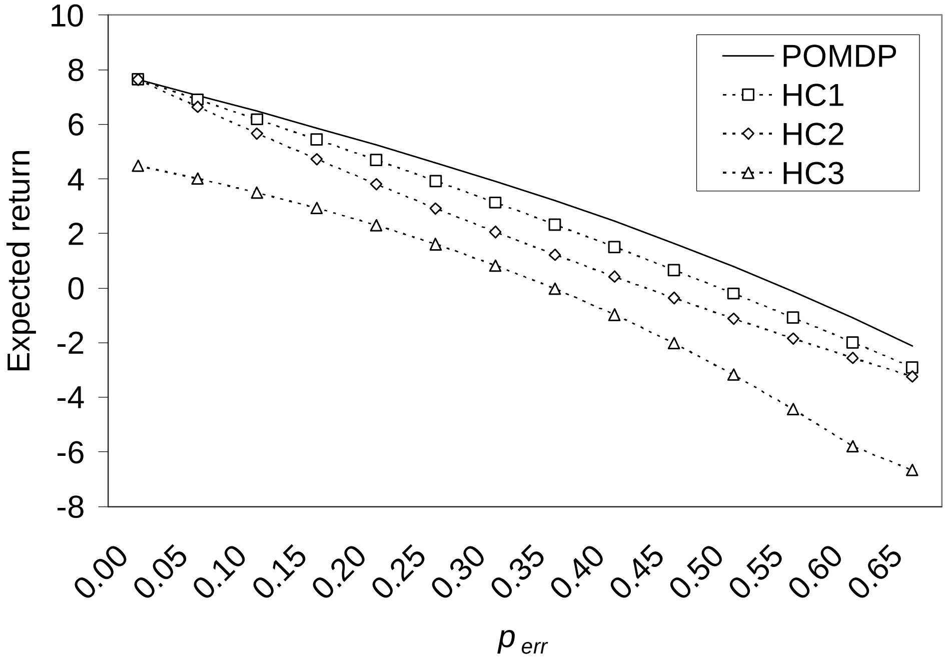


Figure 20: Expected return vs. p_{err} for optimized POMDP policy and 3 handcrafted policies.

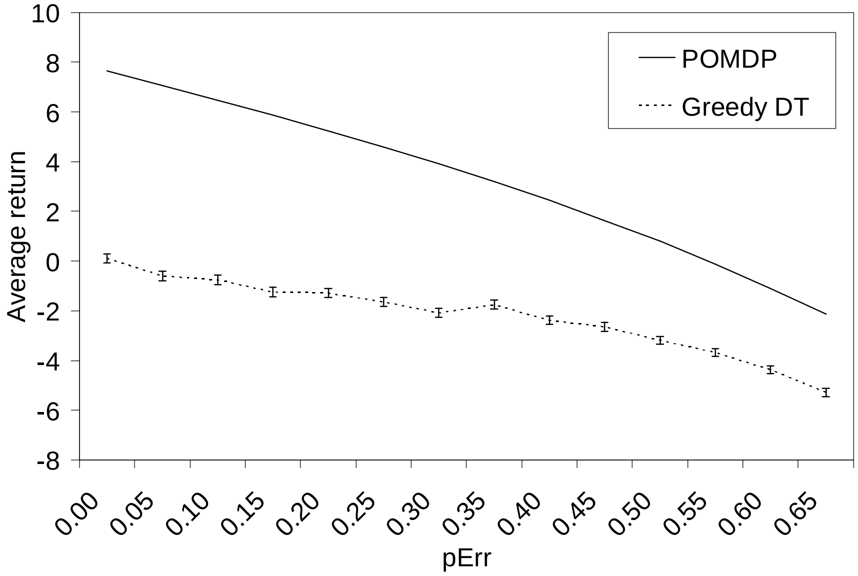


Figure 21: Average concept error rate (*pErr*) vs. average return for the POMDP and greedy decision theoretic ("Greedy DT") dialog managers.

3.4. Illustration with real dialog data

All of the above simulations employed a hand-crafted model of the user. To assess the impact of this, a final experiment was conducted using a dialog manager optimized with a user model estimated from real dialog data, and then evaluated on a second user model estimated from held-out data.

In this experiment, we employed real dialog data from the SACTI-1 corpus (Williams and Young., 2004). The SACTI-1 corpus contains 144 human-human dialogs in the travel/tourist information domain using a "simulated ASR channel", which introduces errors similar to those made by a speech recognizer (Stuttle et al., 2004). One of the subjects acts as a tourist seeking information (analogous to a user) and the other acts as an information service (analogous to a spoken dialog system). The corpus contains a variety of word error rates, and the behaviors observed of the subjects in the corpus are broadly consistent with behaviors observed of a user and a computer using a real speech recognition system (Williams and Young, 2004).

				Training	**Testing**
a_m	s'_u	**Description**	a'_u	$p(a'_u \mid s'_u, a_m)$	$p(a'_u \mid s'_u, a_m)$
greet	from x to y	User wasn't paying attention	*null*	0.013	0.025
		User says both places	*from-x-to-y*	0.573	0.630
		User says just "from" place	*from-x*	0.207	0.173
		User says just "to" place	*to-y*	0.207	0.173
		All other user actions	(all others)	0.000	0.000
ask-from	from x to y	User wasn't paying attention	*null*	0.013	0.025
		User says just the name of the place	*x*	0.444	0.419
		User says the name of the place preceded by "from"	*from-x*	0.399	0.349
		User says both places	*from-x-to-y*	0.144	0.207
		All other user actions	(all others)	0.000	0.000
confirm-to-y	from x to y (NB - the system has the right hypothesis)	User wasn't paying attention	*null*	0.013	0.025
		User says just "yes"	*yes*	0.782	0.806
		User says the item that was being confirmed	*y*	0.108	0.092
		User says the item being confirmed, with the "to" preposition	*to-y*	0.097	0.077
		All other user actions	(all others)	0.000	0.000

Table 9: Training and Testing user models estimated from disjoint data in the SACTI-1 corpus.

Wizard/User turn pairs which broadly matched the types of action in the test-bed dialog problem were annotated. The corpus was then segmented into a *training* sub-corpus and a *testing* sub-corpus, each composed of an equal number of dialogs, the same mix of word error rates, and disjoint subject sets. One user model $p(a'_u \mid s'_u, a_m)$ was then estimated from each sub-corpus, shown in Table 9. Due to data sparsity in the SACTI-1 corpus, the user actions *yes* and *no* were grouped into one class, so probabilities for these actions are equal (with appropriate conditioning for the sense of *yes* vs. *no*).

To conduct the simulations, first policy optimization was performed on the test-bed dialog problem with the training user model using *Perseus*. Then the testing user model was installed, and 10,000 dialog turns were run with the policy created from the training user model. This process was repeated for values of p_{err} from 0.00 to 0.65.Figure 22 shows results for a range of values of p_{err}. The Y-axis shows average return per dialog. Error bars indicate the 95% confidence interval for the performance on the testing user model. As speech recognition errors increase, the average reward decreases, consistent with the findings in the previous sections. For all values of p_{err}, the performance on the testing user model is very close to the performance on the training user model, and in some cases it is slightly higher. This is possible because, in some situations, the testing user model provides slightly more information than the training user model, and this enables the policy to perform better on the testing user model at certain error rates. For example, when asked the *greet* question or asked for the *from* or *to* places, the testing user model is more likely than the training model to reply with both the *from* and *to* places. Overall, the results in Figure 22 demonstrate that the POMDP policy estimation is reasonably robust to variations in user behaviour, or stated alternatively, that errors in the estimation of the user model can be tolerated.

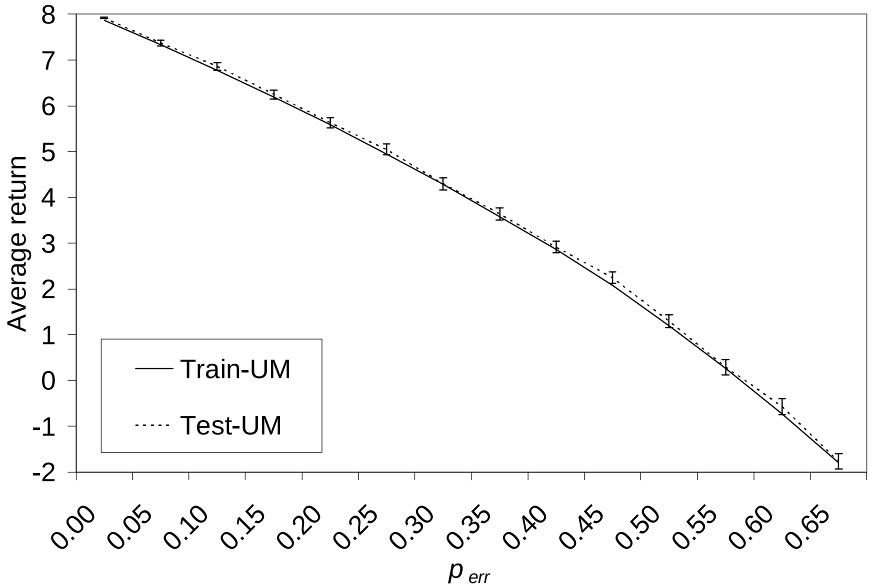


Figure 22: p_{err} vs. average return per dialog for a dialog manager optimized on the training user model, and evaluated on the same model (*Train-UM*) and the testing user model (*Test-UM*).

3.5. Conclusions and future work

Despite the advances made in recent years, the design of robust spoken dialog systems continues to be a major research challenge. The key problem is that the uncertainty caused by speech recognition errors makes it extremely difficult to accurately track the state of the dialog. Typically, these errors lead to false assumptions which in turn lead to spurious dialogs. This paper has argued that by modelling a spoken dialog system as a partially observable Markov decision process (POMDP), significant improvements in robustness can be achieved. Furthermore, it has been shown that the ideas underlying existing techniques to improving robustness – maintaining multiple state hypotheses, using local confidence scores to validate user input, and automating action selection and planning – are all just special cases of the POMDP formalism. Thus, the POMDP approach provides a basis for both improving the performance of these existing techniques and unifying them into a single framework supporting global optimisation. The paper has explained how the various benefits of POMDPs can be exploited in the form of an SDS-POMDP, and presented empirical results from simulation experiments – including experiments trained on real dialog data and evaluated on held-out dialog data – as supporting evidence.

Even so, despite the clear potential of POMDPs, several key challenges remain. Most crucially, *scaling* the model to handle real-world problems remains a significant challenge: the complexity of a POMDP grows with the number of user goals, and optimization quickly becomes intractable. The POMDPs described in this paper and in the literature (Roy et al., 2000; Zhang et al., 2001) have been artificially small problems consisting of a limited set of user goals, yet real systems have thousands or millions of user goals for which optimization is intractable, even using the latest approximate optimization techniques.

To illustrate why POMDPs scale poorly for dialog management, consider an SDS-POMDP in the travel domain which attempts to gather the name of a single city from a user. The machine is aware of 1000 cities, and since the POMDP maintains a distribution over all user goals, it must include one user goal for each of the 1000 cities. Further, the POMDP includes (among other actions) distinct actions to "confirm" and "submit" each city. Finally, the POMDP includes an observation for each city name. Thus, in general, the number of states, actions, and observations all grow with the number of distinct user goals, and adding models for the user actions and dialog history further exacerbates this growth.

Two strands of recent work have begun to address scalability. First, the *Summary POMDP* method provides a way to scale up the SDS-POMDP model for the so-called *slot-filling* class of spoken dialog systems (Williams and Young, 2005).[22] In a Summary POMDP, exact belief monitoring is performed, but planning is done in a compressed space called *summary space*. For a given slot, summary space expresses the probability mass of the highest-ranking value but disregards the value itself. Continuing the example above with 1000 cities, suppose that at a certain time-step, $\max(b(s_u)) = 0.8$ and $\arg\max_{s_u}(b(s_u)) = london$. The summary POMDP performs planning by considering the vector $p(\tilde{s}_u) = [0.8, 0.2]$, whereas a standard formulation considers a vector over all 1000 cities. As a result, the Summary POMDP method can scale to much larger problems. This is demonstrated by Figure 23 which shows the expected reward of the optimal policy computed using both the full POMDP model and the Summary model as the number M of slot values increases. As can be seen the baseline model fails to find

[22] "Slot-filling" dialogs seek to elicit values for N variables – or "slots" – from a user. This construction makes it possible to enumerate all possible user goals by constructing a vector of all possible values for each slot. Slot-filling dialogs are generally regarded as a useful class of dialogs but they are limited in expressiveness and can't account for more complex dialog behaviours like negotiation, complex information exchange, stack-like behaviour, etc..

acceptable solutions for *M* greater than 20 slots, whereas the performance of the Summary model is unaffected by *M*. Subsequent work has extended this technique to scale to a large number of slots by performing planning myopically for each slot, and then combining each slot's policy together using a simple heuristic (Williams and Young, 2006). The optimization techniques employed are described in detail in (Williams, 2006).

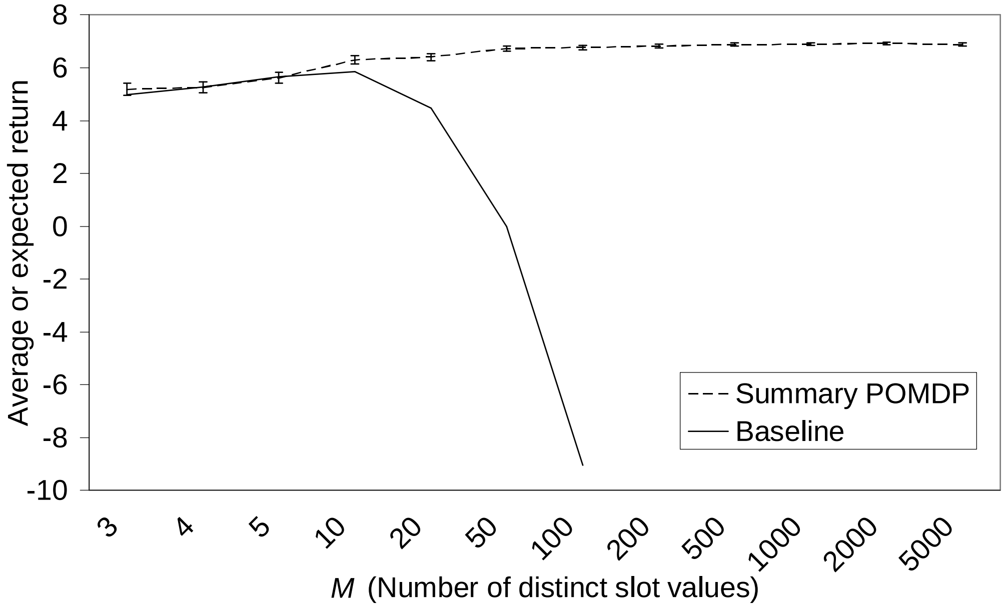


Figure 23: *M* (number of distinct slot values) vs. average or expected return for a simplified 1-slot dialog problem. The baseline is a direct solution of the fully-enumerated POMDP. Note that at about 20 slot values, the direct optimization is no longer able to produce good policies, but the performance of the Summary POMDP is relatively constant. Taken from (Williams and Young, 2005).

The Summary POMDP performs belief monitoring by enumerating all possible user goals, and while this enumeration is reasonable for comparatively simple dialog models such as slot-filling, it is not directly applicable to more complex applications such as those tackled by *Information State Update* systems which represent the dialog state by a large and complex hierarchical data structure (Larsson and Traum, 2000). To deal with these very large state spaces, a second promising method is to divide the space of user goals into a hierarchy of equivalence classes or *partitions*. Belief updating is then performed on partitions rather than states. By successively splitting partitions, the system can use the incoming evidence to gradually focus-in on the underlying states of interest without having to needlessly consider large numbers of low probability states. A specific implementation of this idea is the *Hidden Information State* dialog model which uses probabilistic context-free rules to describe the partition hierarchy. In effect, these rules form an ontology of the application domain and they enable user goals to be expressed in a top-down manner which directly reflects the order in which sub-topics are typically visited in conversation (Young et al., 2006).

In addition to scaling issues, several other interesting questions remain concerning the uses of POMDPs in dialog. In particular, the choice of appropriate reward functions and their relationship to established metrics of user performance such as the PARADISE scheme remain to be resolved (Walker et al., 1997). There is also the related question of how models of user behaviour should be created and evaluated. Ultimately, the definitive test of a POMDP-based dialog system must be evaluation using real users, and the next step is clearly to build such systems and gather the necessary empirical data. In the meantime, the SDS-POMDP is unique in providing a complete mathematical framework for designing and building spoken dialog systems. This framework allows all of the key components to be trained from data and it supports global optimisation. We believe that POMDPs have clear potential to advance the state-of-the-art in spoken dialog systems and as such merit serious further investigation.

4. Acknowledgments

The authors thank Pascal Poupart for many helpful discussions and comments.

This work was supported in part by the European Union "Tools for Ambient Linguistic Knowledge (TALK)" project.

5. References

Bohus D, Carpenter P, Jin C, Wilson D, Zhang R, Rudnicky AI. Is this conversation on track? Proc Eurospeech, Aalborg, Denmark; 2001.

Bohus D, Rudnicky AI. Integrating multiple knowledge sources for utterance-level confidence annotation in the CMU communicator spoken dialog system. Carnegie Mellon University, Technical Report CS190; 2002.

Bohus D, Rudnicky AI. Sorry, I didn't catch that! - An investigation of non-understanding errors and recovery strategies. Proc SIGdial Workshop on Discourse and Dialogue, Lisbon; 2005a.

Bohus D, Rudnicky AI. A principled approach for rejection threshold optimization in spoken dialog systems. Proc Eurospeech, Lisbon; 2005b.

Cassandra AR, Kaelbling LP, Littman ML. Acting optimally in partially observable stochastic domains. Proc Conf on Artificial Intelligence, (AAAI), Seattle; 1994.

Denecke M, Dohsaka K, Nakano M. Learning dialogue policies using state aggregation in reinforcement learning. Proc Intl Conf on Speech and Language Processing (ICSLP), Jeju, Korea; 2004.

Deng Y, Mahajan M, Acero A. Estimating speech recognition error rate without acoustic test data. Proc Eurospeech, Geneva; 2003.

Doran C, Aberdeen J, Damianos L, Hirschman L. Comparing several aspects of human-computer and human-human dialogues. Proc SIGdial Workshop on Discourse and Dialogue, Aalborg, Denmark; 2001.

Evermann G, Woodland P. Posterior probability decoding, confidence estimation and system combination. Proc Speech Transcription Workshop, Baltimore; 2000.

Gabsdil M, Lemon O. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. Proc Association for Computational Linguistics (ACL), Barcelona; 2004.

Glass J. Challenges for spoken dialogue systems. Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Colorado, USA; 1999.

Goddeau D, Pineau J. Fast reinforcement learning of dialog strategies. Proc IEEE Intl Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Istanbul; 2000.

Hansen EA. Solving POMDPs by searching in policy space. Proc Uncertainty in Artificial Intelligence (UAI), Madison, Wisconsin; 1998.

Henderson J, Lemon O, Georgila K. Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data. Proc Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Intl Joint Conf on Artificial Intelligence (IJCAI), Edinburgh, UK; 2005.

Higashinaka H, Nakano M, Aikawa K. Corpus-based discourse understanding in spoken dialogue systems. Proc Association for Computational Linguistics (ACL), Sapporo, Japan; 2003.

Hirschberg J, Litman D, Swerts M. Detecting misrecognitions and corrections in spoken dialogue systems from `aware' sites. Proc Prosody in Speech Recognition and Understanding, New Jersey, USA; 2001.

Hoey J, Poupart P. Solving POMDPs with continuous or large observation spaces. Proc Intl Joint Conf on Artificial Intelligence (IJCAI), Edinburgh, Scotland; 2005.

Horvitz E, Paek T. DeepListener: harnessing expected utility to guide clarification dialog in spoken language systems. Proc Intl Conf on Spoken Language Processing (ICSLP), Beijing; 2000.

Jensen, F. Bayesian networks and decision graphs. New York: Springer Verlag; 2001.

Jurafsky D, Martin J. Speech and language processing. Englewood, NJ: Prentice-Hall; 2000.

Kaelbling L, Littman ML, Cassandra AR. Planning and acting in partially observable stochastic domains. Artificial Intelligence; 1998: 101.

Kemp T, Schaff T. Estimating confidence using word lattices. Proc Eurospeech, Rhodes, Greece; 1997, 827–830.

Krahmer E, Swerts M, Theune M, Weegels M. Problem spotting in human-machine interaction. Proc Eurospeech, Budapest; 1999: 1423-1426.

Krahmer E, Swerts M, Theune M, Weegels M. Error detection in spoken human-machine interaction. Intl Journal of Speech Technology; 2001: 4 (1), 19-30.

Lane IR, Ueno S, Kawahara T. Cooperative dialogue planning with user and situation models via example-based training. Proc Workshop on Man-Machine Symbiotic Systems, Kyoto, Japan; 2004, 2837-2840.

Langkilde I, Walker MA, Wright J, Gorin A, Litman D. Automatic prediction of problematic human-computer dialogues in "How May I Help You?" Proc IEEE Workshop on Automatic Speech Recognition and Understanding Workshop (ASRU), Colorado, USA; 1999, 369-372.

Larsson S, Traum D. Information state and dialogue management in the TRINDI dialogue move engine toolkit. Natural Language Engineering; 2000: 5(3–4), 323–340.

Levin E, Pieraccini R. A stochastic model of computer-human interaction for leaning dialog strategies. Proc Eurospeech, Rhodes, Greece; 1997.

Levin E, Pieraccini R, Eckert W. Using Markov decision process for learning dialogue strategies. Proc Intl Conf on Acoustics, Speech, and Signal Processing (ICASSP), Seattle; 1998, 201-204.

Levin E, Pieraccini R, Eckert W. A stochastic model of human-machine interaction for learning dialogue strategies. IEEE Trans on Speech and Audio Processing; 2000: 8 (1), 11-23.

Litman DJ, Hirschberg J, Swerts M. Predicting user reactions to system error. Proc Association for Computational Linguistics (ACL), Toulouse, France; 2001.

Litman DJ, Pan S. Predicting and adapting to poor speech recognition in a spoken dialogue system. Proc Conf on Artificial Intelligence (AAAI), Austin, Texas, USA; 2000, 722-728.

Moore R, Browning S. Results of an exercise to collect "genuine" spoken enquiries using woz techniques. Proc Institute of Acoustics; 1992: 14 (6).

Moreno PJ, Logan B, Raj B. A boosting approach for confidence scoring. Proc Eurospeech, Aalborg, Denmark; 2001.

Murphy K. A survey of POMDP solution techniques. Technical Report, U. C. Berkeley; 2000.

Paek T, Horvitz E, Ringger E. Continuous listening for unconstrained spoken dialog. Proc Intl Conf on Speech and Language Processing (ICSLP), Beijing; 2000.

Pao C, Schmid P, Glass J. Confidence scoring for speech understanding systems. Proc Intl Conf on Speech and Language Processing (ICSLP), Sydney; 1998.

Pieracinni R, Huerta J. Where do we go from here? Research and commercial spoken dialog systems. Invited talk, SigDial Workshop on Discourse and Dialogue, Lisbon; 2005.

Pietquin O. A framework for unsupervised learning of dialogue strategies, PhD Thesis, Faculty of Engineering, Mons (TCTS Lab), Belgium; 2004.

Pietquin O, Renals S. ASR modelling for automatic evaluation and optimisation of dialogue systems. Proc IEEE Intl Conf on Acoustics, Speech and Signal Processing (ICASSP), Florida; 2002.

Pineau J, Gordon G, Thrun S. Point-based value iteration: an anytime algorithm for POMDPs. Proc Intl Joint Conf on Artificial Intelligence (IJCAI), Acapulco, Mexico; 2003.

Roy N, Pineau J, Thrun S. Spoken dialog management for robots. Proc Association for Computational Linguistics (ACL), Hong Kong; 2000.

Scheffler K, Young SJ. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. Proc Human Language Technologies (HLT), San Diego, USA; 2002.

Singh S, Litman DJ, Kearns M, Walker MA. Optimizing dialogue management with reinforcement leaning: experiments with the NJFun system. Journal of Artificial Intelligence; 2002: 16, 105-133.

Skantze G. Exploring human error handling strategies: implications for spoken dialogue systems. Proc Error Handling in Spoken Dialogue Systems, ISCA Tutorial and Research Workshop, Château d'Oex, Vaud, Switzerland; 2003.

Spaan MTJ, Vlassis N. Perseus: randomized point-based value iteration for POMDPs. Journal of Artificial Intelligence Research; 2005: 24, 195–220.

Stuttle M, Williams JD, and Young, SJ. A framework for wizard-of-Oz experiments with a simulated ASR channel. Proc Intl Conf on Spoken Language Processing (ICSLP), Jeju, South Korea, 2004.

Sutton RS, Barto AG. Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press; 1998.

Walker MA, Litman DJ, Kamm CA, Abell A. PARADISE: A framework for evaluating spoken dialogue agents. Proc Association for Computational Linguistics (ACL), Madrid, Spain; 1997.

Walker MA, Fromer JC, Narayanan S. Learning optimal dialogue strategies: a case study of a spoken dialogue agent for email. Proc Association for Computational Linguistics and Intl Conf on Computational Linguistics (ACL/COLING), Montreal; 1998

Watkins CJCH. Learning from delayed rewards. Ph.D. thesis, Cambridge University; 1989.

Williams JD. Partially Observable Markov Decision Processes for Spoken Dialogue Management. Ph D thesis, Cambridge University; 2006.

Williams JD, Poupart P, Young SJ. Factored partially observable Markov decision processes for dialogue management. Proc Workshop on Knowledge and Reasoning in Practical Dialog Systems, Intl Joint Conf on Artificial Intelligence (IJCAI), Edinburgh; 2005a.

Williams JD, Poupart P, Young SJ. Partially observable Markov decision processes with continuous observations for dialogue management. Proc SigDial Workshop on Discourse and Dialogue, Lisbon; 2005b.

Williams JD, Young SJ. Characterizing task-oriented dialog using a simulated ASR channel. Proc Intl Conf on Speech and Language Processing (ICSLP), Jeju, South Korea; 2004.

Williams JD, Young SJ. Scaling up POMDPs for dialog management: the "Summary POMDP" method. Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), San Juan, Puerto Rico; 2005.

Williams JD, Young SJ. Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI). Proc AAAI Workshop on Statistical and Empirical Approaches to Spoken Dialog Systems, Boston; 2006.

Young SJ, Williams JD, Schatzmann J, Stuttle M, Weilhammer K. The hidden information state approach to dialogue management. Cambridge University Engineering Department, Technical Report CUED/F-INFENG/TR.544; 2006.

Young SJ. Probabilistic methods in spoken dialogue systems. Philosophical Trans of the Royal Society (Series A); 2000: 358 (1769), 1389-1402.

Zhang B, Cai Q, Mao J, Chang E, Guo B. Spoken dialogue management as planning and acting under uncertainty. Proc Eurospeech, Aarlborg, Denmark; 2001.