THE HIDDEN INFORMATION STATE APPROACH TO DIALOG MANAGEMENT

Steve Young, Jost Schatzmann, Karl Weilhammer, Hui Ye

Engineering Department, Cambridge University, CB2 1PZ, UK

ABSTRACT

Partially observable Markov decision processes (POMDPs) provide a principled mathematical framework for modelling the uncertainty inherent in spoken dialog systems. However, conventional POMDPs scale poorly with the size of state and observation space. This paper describes a variation of the classic POMDP called the Hidden Information State (HIS) model in which belief distributions are represented efficiently by grouping states together into partitions and policy optimisation is made tractable by using a master to summary space mapping. An implementation of the HIS model is described for a Tourist Information application and aspects of its training and operation are illustrated.

Index Terms— statistical dialog modelling; partially observable Markov decision processes (POMDPs)

1. INTRODUCTION

Conventional spoken dialog systems operate by finding the most likely interpretation of each user input, updating some internal representation of the dialog state and then outputting an appropriate response. Error tolerance depends on using confidence thresholds and where they fail, the dialog manager must resort to quite complex recovery procedures. Attempts have been made to optimise within this framework using MDPs [1, 2]. However, the lack of an explicit model for representing the inherent uncertainty in the users input and its subsequent interpretation severely limits what can be achieved.

Rather than MDPs, Partially Observable MDPs (POMDPs) potentially provide a much more powerful framework for modeling dialog systems since they provide an explicit represention of uncertainty [3, 4]. The structure of a POMDP-based dialog system is outlined in Fig 1. It is assumed that the machine's internal representation of the dialog state must capture the user's last input dialog act a_u, the user's goal s_u, and some record of the dialog history s_d. Since s_m can never be known with certainty, the dialog manager maintains a distribution over all possible values called a *belief state* $b(s_m)$. This belief state is updated every turn and its value is input to a policy which determines the next machine action a_m. By associating rewards with states and actions, this policy can be optimised to achieve the desired design criteria. Since the dialog manager is maintaining a distribution over all possible dialog states, it is straightforward to accommodate not just the most likely interpretation of a_u but a distribution over many possible a_u. Thus, the POMDP formalism provides a complete and principled framework for modelling the inherent uncertainty in a spoken dialog system and optimising its performance. Furthermore, it naturally accommodates N-best recognition outputs and associated confidence scores[5, 6].

The use of POMDPs for any practical system is, however, far from straightforward. Firstly, in common with MDPs, the state space of a practical SDS is very large and if represented directly, it would be intractable. Secondly, a POMDP with state space cardinality $n+1$ is equivalent to an MDP with a continuous state space $\boldsymbol{b} \in \Re^n$. Thus,

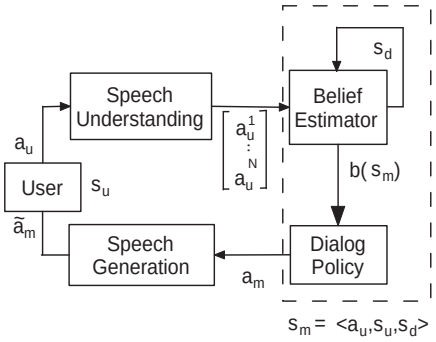

Fig. 1. Abstract view of a POMDP-based spoken dialog system

a POMDP policy is a mapping from partitions in n-dimensional belief space to actions. Not surprisingly these are extremely difficult to construct and whilst exact solution algorithms do exist [7], they do not scale to problems with more than a few states/actions.

This paper describes a form of POMDP which can be scaled to support practical dialog systems. It is inspired by the Information State (IS) update approach to dialog system implementation [8] in which the IS itself is hidden. Hence it is called the Hidden Information State (HIS) model. The practical implementation of the HIS system depends on two key ideas. Firstly, a belief distribution over an extremely large state space can be represented efficiently by grouping states together into partitions and then splitting partitions on demand as the dialog evolves. Secondly, efficient policy optimisation can be achieved by mapping between the full state space and a much smaller and more tractable summary space.

2. THE HIDDEN INFORMATION STATE MODEL

2.1. POMDP Basics

Formally, a Partially Observable MDP is defined as a tuple $\{S_m, A_m, T, R, O, Z, \lambda, \boldsymbol{b}_0\}$ where S_m is a set of machine states; A_m is a set of machine actions; T is a transition probability $P(s'_m|s_m, a_m)$; R defines the expected (immediate, real-valued) reward $r(s_m, a_m)$; O is a set of observations; Z is an observation probability $P(o'|s'_m, a_m)$; λ is a geometric discount factor $0 \leq \lambda \leq 1$; and $\boldsymbol{b}_0$ is an initial belief state.

A POMDP operates as follows. At each time-step, the machine is in some unobserved state $s_m \in S_m$. Since s_m is not known exactly, a distribution over states is maintained called a belief state such that the probability of being in state s_m given belief state $\boldsymbol{b}$ is $b(s_m)$. Based on the current belief state $\boldsymbol{b}$, the machine selects an action $a_m \in A_m$, receives a reward $r(s_m, a_m)$, and transitions to a new (unobserved) state s'_m, where s'_m depends only on s_m and a_m. The machine then receives an observation $o' \in O$ which is dependent on s'_m and a_m. Finally, the belief distribution $\boldsymbol{b}$ is updated based on o' and a_m as follows:

$$b'(s'_m) \quad = \quad k \cdot P(o'|s'_m, a_m) \sum_{s_m \in S_m} P(s'_m|a_m, s_m) b(s_m) \quad (1)$$

where k is a normalisation constant[7]. The first term on the RHS of (1) is called the *observation model* and the term inside the summation is called the *transition model*. Maintaining this belief state as the dialog evolves is called *belief monitoring*.

At each time step t, the machine receives a reward $r(b_t, a_{m,t})$ based on the current belief state b_t and the selected action $a_{m,t}$. The cumulative, infinite horizon, discounted reward is called the *return* and it is given by:

$$R = \sum_{t=0}^{\infty} \lambda^t r(b_t, a_{m,t}) = \sum_{t=0}^{\infty} \lambda^t \sum_{s_m \in S_m} b_t(s_m) r(s_m, a_{m,t}). \quad (2)$$

Each action $a_{m,t}$ is determined by a policy $\pi(\boldsymbol{b}_t)$ and building a POMDP system involves finding the policy π^* which maximises the return.

2.2. HIS Belief Monitoring

In a spoken dialog system, the observation o is the estimate of the user dialog act output by the speech understanding component. In the general case, this will be an N-best list of hypothesised user acts, each with an associated probability, i.e.

$$o \quad = \quad [(a_u^1, p_1), (a_u^2, p_2), \ldots, (a_u^N, p_N)] \quad (3)$$

As indicated in the introduction, the machine state s_m in a spoken dialog system can be factored into three components $s_m = [s_u, a_u, s_d]$. Substituting this factored form into the first term in (1) and making reasonable independence assumptions gives

$$P(o'|s'_m, a_m) = P(o'|s'_u, a'_u, s'_d, a_m) = P(o'|a'_u) \quad (4)$$

This is the HIS observation model and it can be approximated as $P(o|a_u) = p_i$ where $a_u = a_u^i$ in the N-best list. To guard against very poor recognition causing the correct value of a'_u to be dropped from the observation altogether, a *null* action is always included representing all of the user acts not in the N-best list.

Substituting the factored form of s_m into the transition model and making reasonable independence assumptions yields

$$\begin{aligned} P(s'_m|s_m, a_m) &= P(s'_u, a'_u, s'_d|s_u, a_u, s_d, a_m) \\ &= P(s'_u|s_u, a_m) P(a'_u|s'_u, a_m) P(s'_d|s'_u, a'_u, s_d, a_m) \end{aligned} \quad (5)$$

In the HIS model, a user goal is deemed to be the specific entity that the user has in mind. For example, in a tourist information system, the user might be wishing to find "a moderately priced restaurant near the theatre". The user would interact with the system, effectively refining his or her query until an appropriate establishment was found. The duration of a dialog is therefore defined as being the interaction needed to satisfy a single goal. Hence by definition, the transition function for s_u in (5) simplifies trivially to a delta function, i.e.

$$P(s'_u|s_u, a_m) = \delta(s'_u, s_u). \quad (6)$$

To further simplify belief updating, the HIS model assumes that at any time t, the space of all user goals S_u can be divided into a number of equivalence classes where the members of each class are tied together and are indistinguishable. These equivalence classes are called *partitions*. Initially, all states $s_u \in S_u$ are in a single partition p_0. As the dialog progresses, this root partition is repeatedly split into smaller partitions. This splitting is binary i.e. $p \rightarrow \{p', p - p'\}$ with probability $P(p'|p)$. Since multiple splits can occur at each time step, this binary split assumption places no restriction on the possible refinement of partitions from one turn to the next.

Given that user goal space is partitioned in this way, beliefs can be computed based on partitions of S_u rather than on the individual states of S_u. Initially the belief state is just $b_0(p_0) = 1$. Whenever a partition p is split, its belief mass is reallocated as,

$$b(p') = P(p'|p)b(p) \quad \text{and} \quad b(p - p') = (1 - P(p'|p))b(p) \quad (7)$$

Note that this splitting of belief mass is simply a reallocation of existing mass, it is not a belief update, rather it is *belief refinement*.

Substituting (4), (5), (6), into (1) and summing over partitions leads to the update equation for the HIS model [9]

$$\begin{aligned} b'(p', a'_u, s'_d) \quad = \quad k \cdot \quad & \underbrace{P(o'|a'_u)}_{\substack{\text{observation} \\ \text{model}}} \quad \underbrace{P(a'_u|p', a_m)}_{\substack{\text{user action} \\ \text{model}}} \\ \cdot \sum_{s_d} & \underbrace{P(s'_d|p', a'_u, s_d, a_m)}_{\text{dialog model}} \quad \underbrace{P(p'|p)b(p, s_d)}_{\text{belief refinement}} \end{aligned} \quad (8)$$

where p is the parent of p'.

As shown by the labelling in (8), the probability distribution for a'_u is called the *user action model*. It allows the observation probability that is conditioned on a'_u to be scaled by the probability that the user would speak a'_u given the goal s'_u and the last system prompt a_m. In the current implementation of the HIS system, user dialog acts take the form $act(a = v)$ where act is the dialog type, a is an attribute and v is its value [for example, *request(food=chinese)*], the user action model is then approximated by

$$P(a'_u|p', a_m) \approx P(\mathcal{T}(a'_u)|\mathcal{T}(a_m)) P(\mathcal{M}(a'_u)|p') \quad (9)$$

where $\mathcal{T}(\cdot)$ denotes the *type* of the dialog act and $\mathcal{M}(\cdot)$ denotes whether or not the dialog act *matches* the current partition p'. The first term on the RHS of (9) is estimated from a dialog corpus, the second term is set to 1 if the act matches and zero otherwise.

The dialog model is a deterministic encoding based on a simple grounding model. It yields probability one when the updated dialog hypothesis (ie a specific combination of p', a'_u, s_d and a_m) is consistent with the history and zero otherwise.

2.3. Summary Space Mapping and Optimisation

Although the use of state partitioning makes belief monitoring tractable for practical dialog systems, the state space itself must be reduced to make policy optimisation tractable. The solution to this lies in the observation that most reasonable system responses will focus on just the most likely states. This suggests maintaining two coupled state spaces: the full space called the *master state space* and a much simpler space called the *summary state space*[10]. The summary state space consists of the top 1 or 2 user goal states (s_u) from master space and a simplified encoding of the user action a_u and dialog history s_d. The summary action space consists of a list of high level abstractions of possible machine responses. A dialog turn then consists of first updating the belief state by evaluating (8) in master space. The updated belief state $\boldsymbol{b}$ is then mapped into a summary state $\hat{\boldsymbol{b}}$ where an optimised dialog policy is applied to compute a new summary machine action $\hat{a}_m$. The summary machine action is then mapped back into master space where it is converted to a specific machine dialog act a_m and a response is output to the user.

Policy optimisation in the HIS model utilises a grid-based discretisation of summary belief space and on-line batch ϵ-greedy policy iteration. Given an existing policy π, dialogs are executed and

entity	→	venue(name,type,area)	1.0
type	→	bar(drinks,music)	0.4
type	→	restaurant(food,pricerange)	0.3
area	=	(central|east|west| . . .)	
food	=	(Italian|Chinese| . . .)	

Table 1. Example Ontology Rules

machine actions generated according to π except that with probability ϵ a random action is generated. The system maintains a set of belief points $\{\hat{\boldsymbol{b}}_i\}$. At each turn in training, the nearest stored belief point $\hat{\boldsymbol{b}}_k$ to $\hat{\boldsymbol{b}}$ is located using a distance measure. If the distance is greater than some threshold, $\hat{\boldsymbol{b}}$ is added to the set of stored points and $\hat{\boldsymbol{b}}_k = \hat{\boldsymbol{b}}$. The sequence of points $\hat{\boldsymbol{b}}_k$ traversed in each dialog is stored in a list. Associated with each $\hat{\boldsymbol{b}}_i$ is a function $Q(\hat{\boldsymbol{b}}_i, \hat{a}_m)$ whose value is the expected total reward obtained by choosing summary action $\hat{a}_m$ from state $\hat{\boldsymbol{b}}_i$. At the end of each dialog, the total reward is calculated and added to an accumulator for each point in the list, discounted by λ at each step. On completion of a batch of dialogs, the Q values are updated according to the accumulated rewards, and the policy updated by choosing the action which maximises each Q value. The whole process is then repeated until the policy stabilises. Since even the summary state space is very large, around 10^5 dialogs are required for policy convergence and learning using real users is not practical. Hence, a user simulator is used for training.

3. AN IMPLEMENTATION

To demonstrate the practical application of the HIS model, a complete working system has been built for the Tourist Information Domain which can supply information about hotels, restaurants, bars and amenities in a (fictitious) town. Inputs and outputs to the dialog manager are in the form of dialog acts which consist of an act type such as "inform", "request",etc. and one or more attribute value pairs. For example, an utterance such as "I'd like to find a Chinese restaurant on the east side of town" would get mapped by the semantic decoder into the user dialog act "request(restaurant,food=Chinese,area=east)".

3.1. Partition Splitting

The space of all user goals is described by a set of simple ontological rules of the form illustrated in Table 1. These rules describe the hierarchical structure of the data and the specific values which can be assigned to terminal nodes.[1] Since non-terminal nodes can be expanded in different ways, node expansion rules (indicated by →) have an associated prior probability corresponding to the partition split probability $P(p'|p)$ described above.

Partitions of user goal space are represented by a forest of trees where each tree represents a single partition. This forest of trees is stored in such a way that no partition is duplicated and the sum of the probability of all partitions is always unity. At the start of a dialog, there is just one partition represented by a single root node with belief mass unity. Each incoming user act is matched against each partition in turn. If there is no match, the ontology rules are consulted and the system attempts to create a match by expanding the tree. This expansion will result in partitions being split and their belief mass redistributed between the original partition and the new partition as in equation (7). This is illustrated in Fig 2 in which a partition representing a generic "venue" is split as the result of the user requesting a "bar". The original "type" node had a probability mass of 1.0 and this is redistributed according to the prior in the corresponding ontology rule, 0.4 to the new partition and 0.6 remains with the original. If the user subsequently mentioned another type of venue, this remaining mass of 0.6 would be split again.

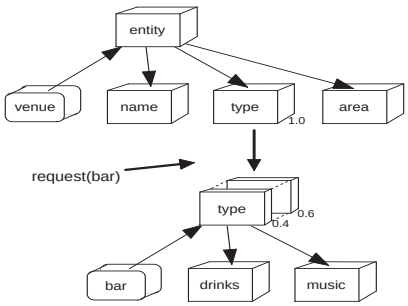

Fig. 2. Illustration of Partition Splitting

3.2. The Dialog Cycle

The overall operation of the prototype HIS system is summarised in Fig 3. Each user utterance is decoded into an N-best list of dialog acts. Each incoming act plus the previous system act are matched against the forest of user goals and partitions are split as needed. Each user act a_u is then duplicated and bound to each partition p. Each partition will also have a set of dialog histories s_d associated with it. The combination of each p, a_u and updated s_d forms a new dialog hypothesis h_k whose beliefs are evaluated using (8).

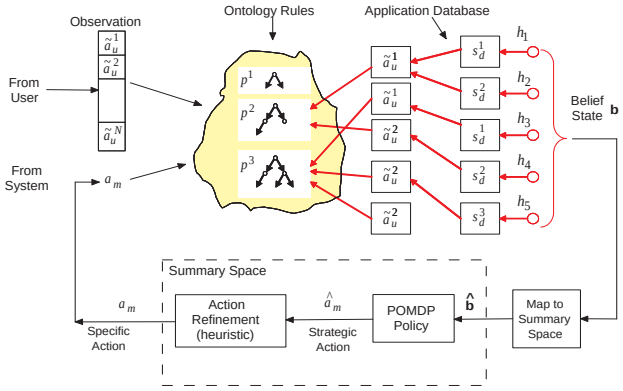

Fig. 3. Overview of Prototype HIS Dialog Manager

Once all dialog hypotheses have been evaluated and any duplicates merged, the master belief state $\boldsymbol{b}$ is mapped into summary space $\hat{\boldsymbol{b}}$ and the nearest policy belief point is found. The associated summary space machine action $\hat{a}_m$ is then mapped back to master space and the machine's actual response a_m is output. The cycle then repeats until the user's goal is satisfied.

3.3. Training

Training follows the Q-learning approach described in section 2.3. Each policy iteration uses a batch size of 5000 dialogs, the discount factor is 0.95 and $epsilon$ is held constant at 0.1. The reward function returns -1 per system turn and $+20$ if the system recommends a venue that matches all the constraints in the user's goal. In all cases, the initial policy is random. A user simulator is used to generate responses to system actions. It has two main components: a *User Goal* and a *User Agenda*. At the start of each dialog, the goal is randomly initialised with requests such as "name", "addr", "phone" and constraints such as "type=restaurant", "food=Chinese", etc. The agenda

[1] It should be noted that apart from the database itself, there is no other application dependent data or code in the dialog manager.

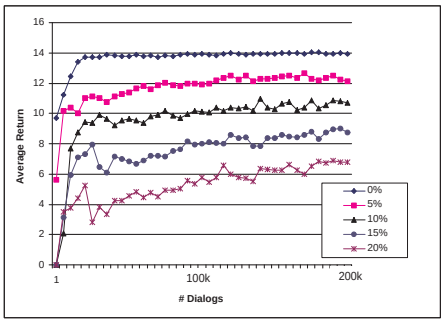

Fig. 4. Average return for various user act error rates vs number of training dialogs

stores the dialog acts needed to ellicit this information in a stack-like structure which enables it to temporarily store actions when another action of higher priority needs to be issued first. This enables the simulator to refer to previous dialog turns at a later point. To generate a wide spread of realistic dialogs, the simulator reacts wherever possible with varying levels of patience and arbitrariness.

Speech understanding errors are simulated at the dialog act level. The user action is fed through a *Scrambler* which uses a set of confusion matrices to generate an N-best list of parsed recognition hypotheses with associated confidence scores at a given error rate.

3.4. Results

Fig. 4 shows the average return achieved by the HIS system at differing user act error rates when tested against the user simulator as a function of the number of dialogs used for training. As can be seen learning increases rapidly at first and then asymptotes. At higher error rates, learning is slower and the asymptotic return reduces. A typical dialog containing a number of recognition errors is shown in Fig. 5. Initially, the system wrongly believes that the user is looking for a hotel. Around turn 4, the systems belief in the user's need for a hotel vs a restaurant is similar, it therefore asks the user to choose. However, the user's decoded response is again ambiguous so the system confirms again before proceeding. Meanwhile, confidence in "Russian food" has been accumulating and by turn 6, a request for an expensive restaurant serving Russian food is the dominant top-ranked hypothesis. So the system proceeds to make a recommendation.

S1	Hello, how can I help you?
U1	I'm looking for a restaurant.
S2	You want a hotel? Whereabouts?
U2	I want a restaurant.
S3	Do you want a hotel?
U3	I want a restaurant.
S4	Do you want a hotel or a restaurant?
U4	A restaurant with Russian food.
S5	You want a restaurant?
U5	Yes, with Russian food.
S6	You want a restaurant serving Russian food?
U6	Yes, somewhere nice and expensive.
S7	The Siberian Tiger is very good.
U7	Ok, where is it?
S7	It's on West Loop.
U7	Ok, thank you goodbye.

Fig. 5. Example Dialog (User Act Err Rate $\approx 15\%$)

4. CONCLUSIONS

This paper has outlined a new Hidden Information State (HIS) approach to statistical dialog management which adapts the POMDP formalism in order to scale to real world problems. The HIS approach provides a number of potential advantages. It naturally integrates N-best recognition hypotheses and confidence measures without setting thresholds or requiring explicit strategies for exploring options. It is robust to recognition errors and because it maintains multiple recognition hypotheses, it does not require elaborate dialog strategies to recover from errors. For database enquiry type applications it is entirely application independent. Finally, by logging conversations and retraining the internal models, it should be capable of adaptively improving over time.

A working prototype system has been implemented, trained and evaluated using a simulator and through informal live testing. Starting from a random policy, the system can learn a competitive strategy without any manual intervention. Furthermore, the system accumulates evidence for each possible user goal over time, making it resilient to errors without explicit programming of recovery procedures. A user trial is planned for the fall as part of the EU Talk Project. The system will then be benchmarked against a hand-crafted system and an MDP-based system. We look forward to reporting the results of this trial in a future paper.

5. REFERENCES

[1] E Levin, R Pieraccini, and W Eckert, "A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies," *IEEE Trans Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2000.

[2] S Singh, DJ Litman, M Kearns, and M Walker, "Optimizing Dialogue Management with Reinforcement Learning," *J Artificial Intelligence Research*, vol. 16, pp. 105–133, 2002.

[3] N Roy, J Pineau, and S Thrun, "Spoken Dialogue Management Using Probabilistic Reasoning," in *Proc ACL*, 2000.

[4] SJ Young, "Talking to Machines (Statistically Speaking)," in *Int Conf Spoken Language Processing*, Denver, Colorado, 2002.

[5] JD Williams, P Poupart, and SJ Young, "Factored Partially Observable Markov Decision Processes for Dialogue Management," in *4th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Edinburgh, 2005.

[6] JD Williams, P Poupart, and SJ Young, "Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management," in *SIGDIAL*, Lisbon, 2005.

[7] LP Kaelbling, ML Littman, and AR Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[8] S Larsson and D Traum, "Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit," *Natural Language Engineering*, pp. 323–340, 2000.

[9] SJ Young, JD Williams, J Schatzmann, MN Stuttle, and K Weilhammer, "The Hidden Information State Approach to Dialogue Management," Tech. Rep. CUED/F-INFENG/TR.544, Cambridge Univ. Engineering Dept, 2005.

[10] JD Williams and SJ Young, "Scaling up POMDPs for Dialogue Management: the Summary POMDP Method," in *IEEE workshop on Automatic Speech Recognition and Understanding (ASRU2005)*, Puerto Rico, 2005.