Training a real-world POMDP-based Dialogue System

Blaise Thomson, Jost Schatzmann, Karl Weilhammer, Hui Ye and Steve Young
Cambridge University Engineering Department
Trumpington Street, Cambridge, CB21PZ, United Kingdom
{brmt2, js532, kw278, hy216, sjy}@eng.cam.ac.uk

Abstract

Partially Observable Markov Decision Processes provide a principled way to model uncertainty in dialogues. However, traditional algorithms for optimising policies are intractable except for cases with very few states. This paper discusses a new approach to policy optimisation based on grid-based Q-learning with a summary of belief space. We also present a technique for bootstrapping the system using a novel agenda-based user model. An implementation of a policy trained using this system was tested with human subjects in an extensive trial. The policy gave highly competitive results, with a 90.6% task completion rate.

1 Introduction

Recent work on statistical models for dialogue systems has argued that Partially Observable Markov Decision Processes (POMDPs) provide a principled mathematical framework for modeling the uncertainty inherent in human-machine dialogue (Young, 2006). Briefly speaking, POMDPs extend the traditional (fully-observable) Markov Decision Process (MDP) framework by maintaining a *belief state*, ie. a probability distribution over dialogue states. This enables the dialogue manager to avoid and recover from recognition errors by sharing and shifting probability mass between multiple hypotheses of the current dialogue state. The framework also naturally incorporates n-best lists of multiple recognition hypotheses coming from the speech recogniser.

Due to the vast space of possible belief states, however, the use of POMDPs for any practical system is far from straightforward. Exact algorithms for solving POMDPs do exist, but have been shown to be intractable except for domains limited to a few states (Kaelbling et al., 1998). In a practical dialogue system the minimum number of dialogue states is typically determined by the number of possible user goals, and this number usually far exceeds the limits of exact solution algorithms.

Approximate algorithms have been developed to overcome the intractibility of exact algorithms but even the most efficient of these techniques such as Point Based Value Iteration (PBVI) cannot scale to the many thousand states required by a statistical dialogue manager (Williams, 2006; Pineau et al., 2003). Previous work by Williams and Young (2006) on Composite Summary Point Based Value Iteration (CSPBVI) has suggested the use of a small *summary space* for each slot where PBVI policy optimisation can be applied. This has shown to give good results on synthetic data but remains untested within real dialogue systems.

One potential problem with the CSPBVI technique is that policy learning can only be performed *offline*, ie. at design time, because policy training requires an existing accurate model of user behaviour. In this paper, an alternative technique for *online* training based on Q-learning is presented. Online training allows the system to adapt to real users as new dialogues are recorded.

The learning algorithm presented here does not require any model of user behaviour so initial dialogues may well be incoherent. In fact, the system requires several thousand dialogues before convergence to a suitable policy begins. This means that in practice the model needs to be bootstrapped via a user simulator. Further adapatation can then be done with real users.

The paper is organised as follows. Section 2 provides an introduction to the POMDP model and explains the Summary POMDP framework that is used in the remainder of the paper. A new online method for policy optimisation is presented in Section 3 and a novel agenda-based user model for bootstrapping the system is introduced in Section 4. Section 5 discusses an evaluation of a sample implementation built for a Tourist Information System and tested with human subjects. The system per-

Bridging the Gap: Academic and Industrial Research in Dialog Technologies Workshop Proceedings, pages 9–16,
NAACL-HLT, Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

formed competitively with 90.6% of tasks successfully completed despite a mix of native and non-native speakers. The paper concludes with a summary and some directions for future work.

2 Background

2.1 POMDP Basics

A POMDP is defined in much the same way as an MDP, except that the states are not observable and instead have to be estimated from observations. Formally, a POMDP is 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 actions that the machine may take
- O is a set of possible observations
- T defines the transition probability such that $T(s_m, a_m, s'_m) = P(s'_m|s_m, a_m)$
- R defines the immediate reward obtained from taking a particular action in a particular state to be $r(s_m, a_m)$
- Z defines the probability of a particular observation given the state and machine action $P(o'|s'_m, a_m)$
- λ is a geometric discount factor $0 \leq \lambda \leq 1$
- $\boldsymbol{b}_0$ is an initial belief state.

When the POMDP operates, it also makes use of a policy $\pi : \Pi(S) \longrightarrow A_m$ that chooses an action given a point in belief space. Here $\Pi(S)$ is the set of all possible probability distributions over S_m (an $|S_m| - 1$ dimensional simplex). $\pi(\boldsymbol{b})$ gives the action to take when the POMDP is in belief state $\boldsymbol{b}$.

The sequence of events in the POMDP follows a cycle. At each time step, the machine is in some unobserved state $s_m \in S_m$. Since the true state is unknown, the machine maintains a probability distribution over the states $\boldsymbol{b}$, which is called the belief state. Based on this belief state and the policy π being followed, the system takes an action $a_m = \pi(\boldsymbol{b})$. The machine is rewarded with $r(s_m, a_m)$ and the state transtitions to a new unobserved state s'_m with probability $T(s_m, a_m, s'_m)$. The machine then receives an observation $o' \in O$, with probability dependent only on the new state s'_m and the machine action a_m. The belief state is updated based on the events of the turn and the cycle repeats. The belief state update is computed as

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

where k is a normalisation constant(Kaelbling et al., 1998). Maintaining this belief state as the dialog evolves is called *belief monitoring*.

Figure 1 shows a graphical representation of a POMDP based dialogue system. When the user utters a user act a_u

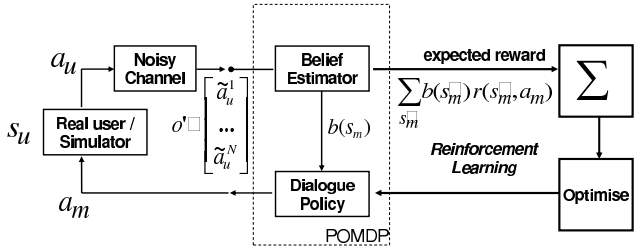

Figure 1: Training a POMDP with a simulated user

it is transmitted via speech to the dialogue system. From the machine's point of view, the speech acts as a noisy channel so that the observation received, o, is not necessarily the true dialogue act. Instead it typically describes an n-best list of hypothesised user acts. Based on this, the POMDP belief state is updated and a machine dialogue act, a_m, is selected. As can be seen from the diagram, it is quite trivial to replace a real user with a simulated one so that training can be performed less laboriously.

Given a particular policy, the infinite horizon expected reward as a function of belief state is called the value function. It is calculated as:

$$V^\pi(\boldsymbol{b}) = \sum_{t=0}^{\infty} \lambda^t r(\boldsymbol{b}_t, a_{m,t}) \quad (2)$$

$$= \sum_{t=0}^{\infty} \lambda^t \sum_{s_m \in S_m} b_t(s_m) r(s_m, a_{m,t}) \quad (3)$$

$$= \sum_{t=0}^{\infty} \lambda^t \sum_{s_m \in S_m} b_t(s_m) r(s_m, \pi(\boldsymbol{b}_t)) \quad (4)$$

The goal of POMDP policy optimisation is to find the policy that maximises the value function at every point $\boldsymbol{b}$. It can be shown that such a function always exists and is both continuous and convex.

In the context of policy optimisation it is also useful to define the concept of a Q function(Sutton and Barto, 1998). This is a function of both belief state $\boldsymbol{b}$ and action a_m and is simply the expected reward obtained by first taking action a_m and then following the policy π.

The Bellman Optimality Equation states that a policy is optimal if and only if

$$\pi(\boldsymbol{b}) = \arg \max_{a \in S_m} Q^\pi(a, \boldsymbol{b}) \quad (5)$$

2.2 The Summary POMDP

As discussed in the introduction, directly optimising POMDPs for dialogue systems is completely impractical. Instead, the belief state and actions are mapped down to a summarised form where optimisation becomes tractable. In this context, the original belief space and actions are

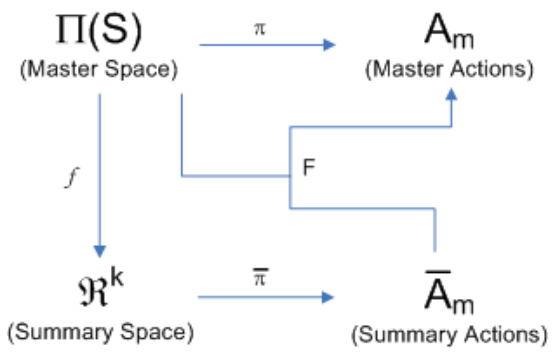

Figure 2: The Summary POMDP framework

called *master* space and *master* actions, while the summarised versions are called *summary* space and *summary* actions.

Action selection in the full model would be a mapping from a belief state $\boldsymbol{b} \in \Pi(S_m)$ to an action $a_m \in A_m$. The summary POMDP splits this up as follows. The model initially extends the standard POMDP with a set of summary actions $\bar{A}_m$ and a mapping from summary actions to master actions F. This function should be allowed access to the master belief space so that the summary can be as brief as possible (formally, $F : \bar{A}_m \times \Pi(S_m) \longrightarrow A_m$). Next, a summarising function f is defined from master belief space $\Pi(S_m)$ to summary belief space $\mathbb{R}^k$. Finally, a summary policy $\bar{\pi}$ is defined as a mapping from summary space $\mathbb{R}^k$ to summary actions $\bar{A}_m$. A policy in master space is composed from the above three functions by first mapping to summary space via f, using policy $\bar{\pi}$ to find an appropriate summary action $\bar{a}_m$ and then obtaining a master action with F. The full process is shown graphically in Figure 2. Algebraically the master policy is defined by:

$$\pi(\boldsymbol{b}) = F(\bar{\pi}(f(\boldsymbol{b})), \boldsymbol{b}) \tag{6}$$

Further explanation of the Summary POMDP method can be found in (Williams and Young, 2005). Note that the formalism introduced above may be used for both the summary methods previously used for dialogue systems as well as belief compression techniques used in a more general setting (Roy et al., 2005)

At the summary level, policies may not have enough information to act truly optimally. Hence defining an optimal summary policy is not so obvious. If f is chosen well, however, then one could hope that the optimal action is dependent only on $f(\boldsymbol{b})$. If this is true then a summary policy is called optimal when the following equation holds for every $\boldsymbol{b}$ such that $f(\boldsymbol{b}) = (\boldsymbol{x})$:

$$\bar{\pi}(\boldsymbol{x}) = \arg \max_{\bar{a} \in \bar{A}_m} Q(F(\bar{a}, \boldsymbol{b}), \boldsymbol{b}) \tag{7}$$

3 Summarised Q-learning

Q-learning is a technique for online learning traditionally used in an MDP framework. It is an iterative Monte-Carlo style algorithm where a sequence of sample dialogues are used to estimate the Q functions for each state and action. Inspired by grid-based methods, the summarised Q-learning algorithm discretises *summary space* and uses Q-learning on the resulting MDP-like grid.

Operation of the algorithm proceeds by simply engaging the dialogue manager with either a real user or a user model. At each point where the system must choose an action, the master belief space is mapped down to the summary level as described in Section 2.2. The nearest summary point in the grid is found and the optimal summary action given by that point is chosen.

At the end of the dialogue, the discounted future reward is known for each stage where a choice was taken. This value is recorded along with the grid point where the decision was made, and the action chosen. This is a sample of the discounted future reward obtained by taking the particular action and then following the current policy - i.e. the Q-function evaluated at this grid point. If sufficient dialogues are done the mean of these values will give a good estimate of the true Q-value.

In order to enable learning, an exploration paramater ϵ is selected so that a random summary act will be chosen with probability ϵ. After a batch of dialogues have been completed, the estimates of the Q-functions are updated with the new dialogue scores. The optimal action is then chosen for each point p by

$$\bar{a}_p = \arg\max_{\bar{a}} \hat{Q}(a, p) \tag{8}$$

The selection of which points to put into the grid is a crucial part of the algorithm as one would like the most accuracy at points that will be visited often. As a result, this algorithm uses a variable grid method (Brafman, 1997; Bonet, 2002). During operation, when a point is reached that is further away from any other point than some threshold paramater, the point is added to the grid. This ensures that points are only included in the grid if needed.

Grid-based methods are often criticised because they do not scale well to large state spaces (Pineau et al., 2003). However, when using the Summary POMDP method the state space is reduced significantly before the grid is applied. Although there are no convergence guarantees for this method in the context of Summary POMDPs, Q-learning does guarantee convergence to the optimal policy for standard MDPs. As can be seen from Figure 4, in practice the method does converge to a high performing policy after several thousand dialogues.

4 Agenda-Based Simulation

4.1 User Simulation-Based Training

As described in the introduction to this paper, *online* methods for training statistical dialogue managers allow the dialogue policy to be adapted and improved at runtime, ie. through interaction with real users. During the initial development phase however, many thousand training dialogues are needed to bootstrap the dialogue policy, and this is generally too time-consuming and expensive to be done with real users.

A number of research groups (Levin et al., 2000; Scheffler and Young, 2002; Pietquin and Dutoit, 2005; Georgila et al., 2005; Rieser and Lemon, 2006) have thus investigated the use of user simulation tools for training the dialogue manager (DM). The simulation-based approach typically involves two steps. Firstly, a statistical user model (such as an n-gram or a graphical model) is trained on a limited amount of dialogue data. The model is then used to simulate dialogues with the interactively learning DM (see Schatzmann et al. (2006) for a literature review). Simulation is usually done at a semantic dialogue act level to avoid having to reproduce the variety of user utterances at the word- or acoustic level.

The simulation-based approach assumes the presence of a small corpus of suitably annotated in-domain dialogues (Lemon et al., 2006). For the experiments presented in this paper, no such data was available for training the user model. Hence, it was necessary to develop a model which was simple enough for the model parameters to be handcrafted and yet capable of producing user behaviour realistic enough for training a prototype system. A similar approach has been previously taken by (Levin et al., 2000; Pietquin and Dutoit, 2005) but the performance of the learned dialogue policies was not evaluated using real users.

4.2 User Simulation at a Semantic Level

Human-machine dialogue can be formalised on a semantic level as a sequence of state transitions and dialogue acts[1]. At any time t, the user is in a state s_u, takes action a_u, transitions into the intermediate state s'_u, receives machine action a_m, and transitions into the next state s''_u where the cycle restarts.

$$s_u \rightarrow a_u \rightarrow s'_u \rightarrow a_m \rightarrow s''_u \rightarrow \cdots \quad (9)$$

Assuming a Markovian state representation, user behaviour can be decomposed into three models: $P(a_u|s_u)$ for action selection, $P(s'_u|a_u, s_u)$ for the state transition into s'_u, and $P(s''_u|a_m, s'_u)$ for the transition into s''_u.

[1]In this paper, the terms *dialogue act* and *dialogue action* are used interchangeably. The notation *act(a=x, b=y,...)* is used to represent a dialogue act of a given type *act* (such as *inform* or *request* with items $a = x$, $b = y$, etc.

4.3 Goal- and Agenda-Based State Representation

Inspired by agenda-based methods to dialogue management (Wei and Rudnicky, 1999) the approach described here factors the user state into an agenda A and a goal G.

$$s_u = (A, G) \qquad \text{and} \qquad G = (C, R) \quad (10)$$

During the course of the dialogue, the goal G ensures that the user behaves in a consistent, goal-directed manner. G consists of constraints C which specify the required venue, eg. a centrally located bar serving beer, and requests R which specify the desired pieces of information, eg. the name, address and phone number (cf. Fig. 3).

The user agenda A is a stack-like structure containing the pending user dialogue acts that are needed to elicit the information specified in the goal. At the start of the dialogue a new goal is randomly generated using the system database and the agenda is populated by converting all goal constraints into *inform* acts and all goal requests into *request* acts. A *bye* act is added at the bottom of the agenda to close the dialogue.

As the dialogue progresses the agenda is dynamically updated and acts are selected from the top of the agenda to form user acts a_u. In response to incoming machine acts a_m, new user acts are pushed onto the agenda and no longer relevant ones are removed. The agenda thus serves as a convenient way of tracking the progress of the dialogue as well as encoding the relevant dialogue history. As can be seen in Fig. 3 (turns 1-3), user acts can also be temporarily stored when actions of higher priority need to be issued first, hence providing the simulator with a simple model of user memory.

4.4 Action Selection

At any time during the dialogue, the updated agenda of length N contains all dialogue acts the user intends to convey to the system. Since the agenda is ordered according to priority, with $A[N]$ denoting the top and $A[1]$ denoting the bottom item, selecting the next user act simplifies to popping n items off the top of the stack. Hence, letting $a_u[i]$ denote the ith item in the user act a_u

$$a_u[i] := A[N - n + i] \quad \forall i \in [1..n], 1 \leq n \leq N. \quad (11)$$

and the action selection model becomes a Dirac delta function

$$P(a_u|s_u) = P(a_u|A, G) = \delta(a_u, A[N-n+1..N]) \quad (12)$$

where $A[N - n + 1..N]$ is a Matlab-like shorthand notation for the top n items on A and the random variable n corresponds to the level of initiative taken by the simulated user. In a statistical model the probability distribution over integer values for n should be conditioned on A and learned from dialogue data. For the purposes of

$$C_0 = \begin{bmatrix} type = bar \\ drinks = beer \\ area = central \end{bmatrix}$$

$$R_0 = \begin{bmatrix} name = \\ addr = \\ phone = \end{bmatrix}$$

Sys 0 Hello, how may I help you?

$$A_1 = \begin{bmatrix} inform(type = bar) \\ inform(drinks = beer) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 1 I'm looking for a fine beer bar.

Sys 1 Ok, a wine bar. What pricerange?

$$A_2 = \begin{bmatrix} negate(drinks = beer) \\ inform(price = cheap) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 2 No, beer please!

Sys 2 You are looking for a beer bar, correct?

$$A_3 = \begin{bmatrix} affirm() \\ inform(price = cheap) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 3 Yeah something cheap in the town centre.

Sys 3 Murphy's on Main Square serves cheap beer.

$$A_4 = \begin{bmatrix} request(phone) \\ bye() \end{bmatrix}$$

Usr 4 Ok, and what's the phone number?

Sys 4 The number is 796 69 94.

$$A_5 = [\ bye()\ ]$$

Usr 5 Thanks, goodbye!

Figure 3: Sample dialogue and agenda sequence

bootstrapping the system, n can be assumed independent of A and any distribution $P(n)$ that places the majority of its probability mass on small values of n can be used.

4.5 State Transition Model

The factorisation of s_u into A and G can now be applied to the state transition models $P(s'_u|a_u, s_u)$ and $P(s''_u|a_m, s'_u)$. Letting A' denote the agenda after selecting a_u (as explained in the previous subsection) and using $N' = N - n$ to denote the size of A', we have

$$A'[i] \quad := \quad A[i] \quad \forall i \in [1..N']. \tag{13}$$

Using this definition of A' and assuming that the goal remains constant when the user executes a_u, the first state transition depending on a_u simplifies to

$$\begin{aligned} P(s'_u|a_u, s_u) &= P(A', G'|a_u, A, G) \\ &= \delta(A', A[1..N'])\delta(G', G). \end{aligned} \tag{14}$$

Using $s_u = (A, G)$, the chain rule of probability, and reasonable conditional independence assumptions, the second state transition based on a_m can be decomposed into *goal update* and *agenda update* modules:

$$\begin{aligned} &P(s''_u|a_m, s'_u) \\ &= \underbrace{P(A''|a_m, A', G'')}_{\text{agenda update}} \underbrace{P(G''|a_m, G')}_{\text{goal update}}. \end{aligned} \tag{15}$$

When no restrictions are placed on A'' and G'', the space of possible state transitions is vast. The model parameter set is too large to be handcrafted and even substantial amounts of training data would be insufficient to obtain reliable estimates. It can however be assumed that A'' is derived from A' and that G'' is derived from G' and that in each case the transition entails only a limited number of well-defined atomic operations.

4.6 Agenda Update Model

The agenda transition from A' to A'' can be viewed as a sequence of push-operations in which dialogue acts are added to the top of the agenda. In a second "clean-up" step, duplicate dialogue acts, *null()* acts, and unnecessary *request()* acts for already filled goal request slots must be removed but this is a deterministic procedure so that it can be excluded in the following derivation for simplicity. Considering only the push-operations, the items 1 to N' at the bottom of the agenda remain fixed and the update model can be rewritten as follows:

$$\begin{aligned} &P(A''|a_m, A', G'') \\ &= P(A''[1..N'']|a_m, A'[1..N'], G'') \qquad (16) \\ &= P(A''[N'+1..N'']|a_m, G'') \\ &\quad \cdot \ \delta(A''[1..N'], A'[1..N']). \qquad (17) \end{aligned}$$

The first term on the RHS of Eq. 17 can now be further simplified by assuming that every dialogue act item in a_m triggers one push-operation. This assumption can be made without loss of generality, because it is possible to push a *null()* act (which is later removed) or to push an act with more than one item. The advantage of this assumption is that the known number M of items in a_m

now determines the number of push-operations. Hence $N'' = N' + M$ and

$$P(A''[N'+1..N'']|a_m, G'') = P(A''[N'+1..N'+M]|a_m[1..M], G'') \quad (18)$$

$$= \prod_{i=1}^{M} P(A''[N'+i]|a_m[i], G'') \quad (19)$$

The expression in Eq. 19 shows that each item $a_m[i]$ in the system act triggers one push operation, and that this operation is conditioned on the goal. This model is now simple enough to be handcrafted using heuristics. For example, the model parameters can be set so that when the item *x=y* in $a_m[i]$ violates the constraints in G'', one of the following is pushed onto A'': *negate()*, *inform(x=z)*, *deny(x=y, x=z)*, etc.

4.7 Goal Update Model

The goal update model $P(G''|a_m, G')$ describes how the user constraints C' and requests R' change with a given machine action a_m. Assuming that R'' is conditionally independent of C' given C'' it can be shown that

$$P(G''|a_m, G') = P(R''|a_m, R', C'')P(C''|a_m, R', C'). \quad (20)$$

To restrict the space of transitions from R' to R'' it can be assumed that each request slot (ag. *addr,phone,etc.*) is either filled using information in a_m or left unchanged. One can further assume that the value of any slot depends on its value at the previous time step, the value provided by a_m and that the transition needs to be conditioned on whether the information given in a_m matches the goal constraints. Using $R[k]$ to denote the k'th request slot we can approximate

$$P(R''|a_m, R', C'') = \prod_k P(R''[k]|a_m, R'[k], \mathcal{M}(a_m, C'')). \quad (21)$$

To simplify $P(C''|a_m, R', C')$ we assume that C'' is derived from C' by either adding a new constraint, setting an existing constraint slot to a different value (eg. *drinks=dontcare*), or by simply changing nothing. The choice of transition does not need to be conditioned on the full space of possible a_m, R' and C'. Instead it can be conditioned on simple boolean flags such as "Does a_m ask for a slot in the constraint set?", "Does a_m signal that no item in the database matches the given constraints?", etc. The model parameter set is then sufficiently small for handcrafted values to be assigned to the probabilities.

5 Evaluation

5.1 A scalable POMDP-based system

The summary Q-learning algorithm and agenda-based user model were tested by implementing a POMDP-based dialogue system for a Tourist Information Domain. Users are assumed to have arrived in a town unknown to them and must find a bar, a hotel or a restaurant in the town subject to some constraints (eg. a cheap, Chinese restaurant in the centre of town). The town used was fictitious so that users could not know any of the venues.

The speech recognition was implemented using the Application Toolkit for HTK (ATK) with a vocabulary of about 2000 words. A simple keyword-spotting semantic decoder was used to extract meaning representations (dialogue acts) from the output of the recogniser. The dialogue manager is based on the Hidden Information State (HIS) model (Young et al., 2007), which gives an efficient way of implementing the belief state update in a POMDP-based dialogue system.

In the implementation used for testing, the town included approximately 40 possible venues. Eight different variables are used by the system in deciding which venue to recommend: type of venue; pricerange; area; proximity to a particular place; stars; drinks; food and music. Additionally, the user could ask for the average price, the phone number, the address or a comment on a particular venue.

The model allows for a rich structure in possible user goals via simple ontology rules. For example, venues can only have a *food* concept if their type is *restaurant*. Hence, one would expect that most information retrieval type dialogues could be modeled in a similar manner.

5.2 System Training

The HIS manager factors the machine state of the POMDP into three parts: the user's goal, the dialogue state and the last machine act. An important feature of the system is that indistinguishable user goals are grouped together into *partitions*. For example, if the user is trying to obtain information about restaurants and has not mentioned what type of food they would like, then restaurants will be grouped together regardless of the type of food they serve. In the HIS model, a *hypothesis* refers to the grouping of a partition with a dialog state.

The splitting of the machine state into separate hypotheses provides for a simple mapping to summary state for the Q-learning algorithm, where only information from the top two hypotheses is included. The summary state used has five components: the probabilities of the two most probable hypotheses along with three summary features. These enumerate the possibilities for how many database items fall into the partition, a summary of the dialog state and the type of the last dialog act.

Rewards were given based on task completion and the number of turns in the dialogue. The system was given 20 points for a successful dialogue and 0 for an unsuccessful one. One point was subtracted for each dialogue turn. This encourages the system to be sure of the user's goal,

while penalising inefficient system behaviour.

Training was done using the agenda-based user model described in Section 4. Initially, batches of 1000 dialogues were performed with no error-modeling, updating Q-values and optimal actions at the end of each batch. Figure 4 shows the average reward obtained from each of these policies over 1000 sample dialogues. The policy score converged after approximately 25 000 dialogues and reached an average score of around 14. This illustrates that the Q-learning approach described above does in fact converge in practice. Further training was then done including simulated errors by changing random concept words with a probability of 0.05.

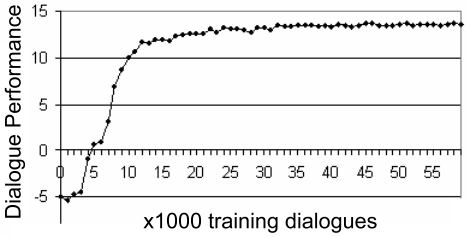

Figure 4: Training a POMDP system

5.3 User testing

The prototype dialogue system described above was evaluated as part of the EU FP6 Talk Project. No online learning was performed during the evaluation. 40 human subjects took part, each of whom completed 4 dialogues. In total, 160 dialogues were recorded, with 6452 turns and 21667 words. Subjects were selected to include a mix of native and non-native speakers and the average Word-Error-Rate was 29.8%.

The performance of the system was measured based on the recommendation of a correct venue, ie. a venue matching all constraints specified in the given task (all tasks were designed to have exactly one solution). Based on this definition, 145 out of 160 dialogues (90.6%) were completed successfully, and the average number of turns to completion was 5.59 (if no correct venue was offered the full number of turns was counted). An extract from an example dialogue is given as an illustration in Fig. 5.

Turn 1:	
System:	Hello and welcome to our dialogue system! Are you looking for a hotel, a restaurant or a bar?
User:	I'm looking for a five star hotel, preferably in the East side of town, near the park
ASR:	I'm looking for a five star hotel preferably use side of town near the park
DA:	request(hotel,stars=5,near=Park)
Turn 2:	
System:	What part of town do you have in mind? For example something in the centre of town or in the north or on the riverside
User:	In the East side of town
ASR:	In the East side of town
DA:	inform(area=east)
Turn 3:	
System:	Did you say you are looking for a hotel in the east part of town?
User:	Yes
ASR:	Yes
DA:	affirm()
Turn 4:	
System:	The Royal Hotel is a great hotel in the east part of town
User:	I was wondering whether I could get the telephone number for that town... for that hotel?
ASR:	was wondering about could get the telephone number of the town for a hotel
DA:	request(phone,hotel)
Turn 5:	
System:	The Royal Hotel has 5 stars and is near the Park
User:	Could I have the phone number?
ASR:	I have the phone number
DA:	request (phone)
Turn 6:	
System:	No problem the phone number of the Royal Hotel is 7027003

Figure 5: Sample test dialogue along with most likely output from the automatic speech recognition (ASR) and decoded dialogue act (DA). The dialogue act format is described in (Young et al., 2005).

6 Summary and Future Work

This paper has investigated a new online method for training a real-world POMDP-based dialogue system. A novel agenda-based user simulation technique was also presented for bootstrapping the dialogue manager without access to training data. Although there is no guarantee of convergence, evaluation results suggest that in practice the training algorithm does converge to a suitable policy. The results also show that the learned policy works well for human users. Future work will focus on finding better mappings to summary space, on training the simulated

user with recorded data and on investigating the effectiveness of online training with real users.

Acknowledgements

This research was partly funded by the EU FP6 TALK Project (http://www.talk-project.org). The system evaluation was conducted in collaboration with Oliver Lemon, Kalliroi Georgila and Jamie Henderson at Edinburgh University and their work is gratefully acknowledged. We would also like to thank the anonymous referees for their helpful comments.

References

B Bonet. 2002. An e-Optimal Grid-based Algorithm for Partially Observable Markov Decision Processes. In *Proceedings of the Nineteenth International Conference on Machine Learning(ICML 2002),*, Sydney, Australia.

RI Brafman. 1997. A Heuristic Variable Grid Solution Method for POMDPs. In *AAAI*, Cambridge, MA.

K. Georgila, J. Henderson, and O. Lemon. 2005. Learning user simulations for information state update dialogue systems. *Proc. of Eurospeech*. Lisbon, Portugal.

LP Kaelbling, ML Littman, and AR Cassandra. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99–134.

O. Lemon, K. Georgila, and J. Henderson. 2006. Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation. In *Proc. of IEEE/ACL SLT*, Palm Beach, Aruba.

E. Levin, R. Pieraccini, and W. Eckert. 2000. A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Trans Speech and Audio Processing*, 8(1):11–23.

O. Pietquin and T. Dutoit. 2005. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Speech and Audio Processing, Special Issue on Data Mining of Speech, Audio and Dialog*.

Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. 2003. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025 – 1032, August.

V. Rieser and O. Lemon. 2006. Cluster-based User Simulations for Learning Dialogue Strategies. In *Proc. of ICSLP*, Pittsburgh, PA.

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. 2005. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40.

J. Schatzmann, K. Weilhammer, M.N. Stuttle, and S. Young. 2006. A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies. *Knowledge Engineering Review, Cambridge University Press*, 21(2):97–126.

K. Scheffler and S. J. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. of NAACL/HLT*. San Diego, CA.

RS Sutton and AG Barto. 1998. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass.

X Wei and AI Rudnicky. 1999. An agenda-based dialog management architecture for spoken language systems. In *Proc. of IEEE ASRU*. Seattle, WA.

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

J. Williams and S. Young. 2006. Scaling pomdps for dialog management with composite summary point-based value iteration (cspbvi). In *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, Boston.

Jason D. Williams. 2006. *Partially Observable Markov Decision Processes for Spoken Dialogue Management*. Ph.D. thesis, University of Cambridge, April.

SJ Young, JD Williams, J Schatzmann, MN Stuttle, and K Weilhammer. 2005. The hidden information state approach to dialogue management. Technical report, Cambridge Univ. Engineering Dept.

S. Young, J. Schatzmann, K. Weilhammer, and H. Ye. 2007. The Hidden Information State Approach to Dialog Management. In *Proc. of ICASSP*, Honolulu, Hawaii.

S. Young. 2006. Using POMDPs for Dialog Management. In *Proc. of IEEE/ACL SLT*, Palm Beach, Aruba.