BAYESIAN UPDATE OF DIALOGUE STATE FOR ROBUST DIALOGUE SYSTEMS

Blaise Thomson, Jost Schatzmann and Steve Young

Engineering Department, Cambridge University, CB2 1PZ, UK

{brmt2, js532, sjy}@eng.cam.ac.uk

ABSTRACT

This paper presents a new framework for accumulating beliefs in spoken dialogue systems. The technique is based on updating a Bayesian Network that represents the underlying state of a Partially Observable Markov Decision Process (POMDP). POMDP models provide a principled approach to handling uncertainty in dialogue but generally scale poorly with the size of the state and action space. The framework proposed, on the other hand, scales well and can be extended to handle complex dialogues. Learning is achieved with a factored summarising function that is applicable for many slot-filling type dialogues. The framework also provides a good structure from which to build hand-crafted policies. For very complex dialogues, this allows the POMDP's principled approach to uncertainty to be incorporated without requiring computationally intensive learning algorithms. Simulations show that the proposed framework outperforms standard techniques whenever errors increase.

Index Terms— Learning systems, Speech processing, Robustness

1. INTRODUCTION

One of the major problems facing current state-of-the-art dialogue systems is robustness. Speech recognition systems cannot be relied upon to give a correct word-level form for any given user utterance. This problem is exacerbated by the task of semantic decoding which is also error prone. The result is that a typical system allows only constrained modes of interaction and frequently needs to confirm the information it receives.

Recent systems accommodate for errors by maintaining beliefs about what the user has said rather than accepting the output from the recogniser. Researchers have, for example, built models that update a compressed representation of belief using the most likely last user utterance [1]. A more principled approach is to use the full set of possible user utterances to update a *distribution* over user goals [2]. This is theoretically more appealing, has a stronger mathematical basis and one would expect it to therefore give better results. The model used to do this type of update is the Partially Observable Markov Decision Process (POMDP).

Although the POMDP model provides an elegant solution, its update formulae are intractable for all but the simplest dialogues. This led to the Hidden Information State (HIS) model being suggested as a less computationally demanding approach [3]. The HIS model improves efficiency by grouping together user goals into partitions which are indistinguishable given the past user utterances. However, this approach can eventually become unwieldly if the dialogue is allowed to progress for a long time, or if the recogniser output contains a large number of alternative hypotheses. As the number of possible past utterances increases, the number of partitions will increase also.

This paper proposes an alternative approach to enabling tractable POMDP updates. The method uses a Bayesian Network to represent the state of the POMDP, where the network is factored according to the slots in the system. This has several advantages. It is computationally more efficient, scales better, is easier to implement, easier to allow for changes in the user goal and easier to extend to more complicated types of dialogue. The efficiency arises because more conditional independence assumptions are made. The ease of use is a result of the intuitive nature of Bayesian Networks and the highly structured form of the model.

The structure of this paper is as follows. The next section introduces the POMDP model and develops the Bayesian Update of Dialogue State framework. Section 3 discusses how policies may be designed with this approach. Both learned and hand-crafted policies are discussed. The section also discusses a new summary function technique which is required for learning because of the factored form of the belief network. Section 4 gives the results of a simulation experiment comparing the model against various baselines. Section 5 concludes the paper.

2. BAYESIAN UPDATE OF DIALOGUE STATE

Any POMDP contains a set of machine actions $\mathcal{A}$, a set of states $\mathcal{S}$ and a set of observations $\mathcal{O}$. In a dialogue system the actions denote the allowable machine utterances, $m \in \mathcal{A}$. A dialogue system state represents a combination of the user's goal, g, the user's last utterance, u, and some form of dialogue history, h. Hence the state is factored as $(g, u, h) = s \in \mathcal{S}$. States are hidden and must be inferred from the output of the recogniser, which forms the observation, $o \in \mathcal{O}$. This inference is done under the Markov assumption: states are conditionally dependent on only their previous value and the last machine action. Given an observation function $P(o|s)$ and a state transition function $P(s'|s, a)$, a belief distribution over states can then be updated using Bayes Theorem [2][1].

To make this update equation more tractable the Bayesian Update of Dialogue State (BUDS) framework makes several further assumptions. Most importantly, the system state is further factorised according to a set of slots, $i \in \mathcal{I}$. The goal becomes $g = (g_i)_{i\in\mathcal{I}}$, the user act $u = (u_i)_{i\in\mathcal{I}}$ and the dialogue history $h = (h_i)_{i\in\mathcal{I}}$. Next some conditional independence assumptions are taken. The user act for a slot depends only on that slot's user goal value and the last machine action. The slot history depends on the previous slot history and the current slot user act. The slot user goals depend on their previous value, the last machine action and may depend on

The work described in this paper was suported by a St John's College Benefactors' Scholarship and an EPSRC grant. Thanks to the anonymous reviewers and to Jason Williams for their comments.

[1] The notation here differs from previous papers to avoid subscripts. Common previous notation is $a_m = m$, $s_u = g$, $a_u = u$ and $s_d = h$

any combination of other current slot user goal values allowing a Bayesian Network structure. As usual, the observation depends on the user act. Figure 1 shows an example network for two time-slices of a two-slot system based on this idea.

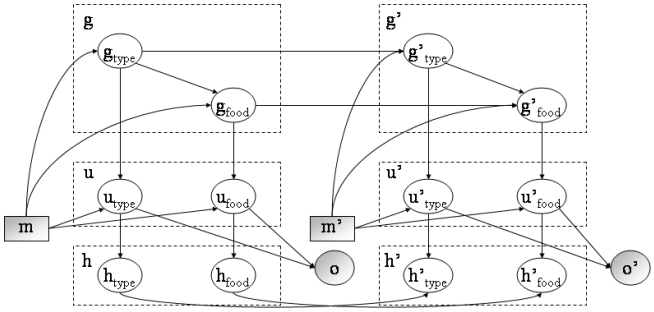

Fig. 1. The two time-slice network for updating the beliefs of two slots: type of venue and food. The food slot only applies if the type of venue is restaurant and is assumed to take the value "N/A" otherwise.

Belief updating is done with standard Dynamic Bayesian Network algorithms. The current implementation uses Loopy Belief Propagation but improvements could be made with other approximate approaches [4]. The history nodes never have observed children and can be omitted for updates of the user goal beliefs.

Implementing changes in the user goal is simplified by specifying a probability of change in each slot given the last machine act and assuming a constant distribution given a change. This allows a simple trade off between the emphasis placed on new information compared to old accumulated beliefs.

2.1. An example implementation

Further simplifications to the above framework can be made by assuming that the user goal network is a tree and that lower nodes in the tree are either applicable or not depending on their parent. Figure 2 shows the goal tree used for an implementation of this approach to building a Tourist Information System.

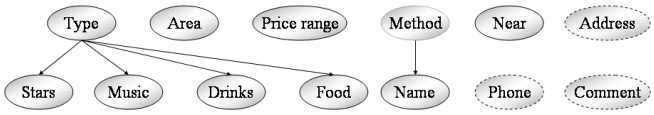

Fig. 2. A Bayesian Network showing the different slot-level user goal nodes and their dependencies for a Tourist Information System.

The task of the system is to aid tourists in finding a hotel, bar or restaurant and to give information about the venue. A user may ask about a venue by name or by specifying constraints. This is encoded in the "method" node which stores the probability that each of these approaches is being used. Although the system cannot ask about this node explicitly, its value is inferred just like any other node. The user can ask about the address, phone number or a comment for any venue. The corresponding nodes for these slots have uninformative user goals since only the dialogue history is important. The user is also allowed to explicitly inform a particular value for any of the remaining slots: type of venue, area, price range, nearness to a particular location, number of stars, type of music, type of drinks, type of food served and name. For these slots the system may request information about the user's constraint, confirm it or select between two options. The actions available to the system also include informing the user about a venue and informing the user that there is no venue matching a set of constraints.

3. POLICY DESIGN

There are two approaches that can be taken to policy design. The simpler approach is to hand-craft what the system should say as a function of the various slot probabilities. The alternative is to exploit the POMDP structure of the problem to allow automatic policy learning.

When hand-crafting a policy, the system designer has complete freedom in deciding which action to take for a particular belief distribution. Typically, the probability of the most likely value for each slot will be the most important feature. If the probability is high the slot value is accepted. The system then decides which action should be taken. It may try to find out more information about nodes it is unsure about or it may tell the user something based on the nodes it has accepted.

Policy learning is significantly more difficult [2]. First a reward must be specified for each pair of action and state and a discount rate γ is chosen. Next some form of learning algorithm is applied. Typically these algorithms make use of the value function, V^{π}, which is the expected future reward of following a policy π, given an initial belief state. Another important quantity is the Q function, which is the expected future reward of following a policy after taking a particular action [5].

General POMDP algorithms are well known to be intractable for large state spaces. In a dialogue system cast as a POMDP at least one state is needed for every user goal. This number grows exponentially with the number of slots in the system. In the Hidden Information State model, learning was implemented by summarising the full belief distribution according to the beliefs in the most likely partitions [3]. An algorithm for online learning of the Q function was then implemented using a grid-based approximation. This type of approach will not work in the BUDS framework since there is no concept of a partition. Grid based approximations to a summary containing the probability of each slot require too many grid points and are ill-suited to the task. Instead, some form of tiling or more general function approximation must be used.

An algorithm that performs well within the proposed framework is episodic Natural Actor Critic [6]. For each action, a, a summary function $\boldsymbol{\phi}_a(\boldsymbol{b})$ is defined. This maps belief distributions, $\boldsymbol{b}$, to a summary vector that the system designer believes are important in making choices between actions. The task of the algorithm is to find parameters, $\boldsymbol{\theta}$, such that $\boldsymbol{\theta} \cdot \boldsymbol{\phi}_a(\boldsymbol{b})$ is a good measure of how appropriate taking action a is. Given the parameters a stochastic policy π is defined which gives the probability of taking action a from belief state $\boldsymbol{b}$. During a dialogue the action to be taken at any point will be sampled from the distribution given by π. This probability distribution is given by the following equation:

$$\pi(a|\boldsymbol{b},\boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}\cdot\boldsymbol{\phi}_a(\boldsymbol{b})}}{\sum_{a'} e^{\boldsymbol{\theta}\cdot\boldsymbol{\phi}_{a'}(\boldsymbol{b})}} \quad (1)$$

Learning of the $\boldsymbol{\theta}$ parameters is done via natural gradient descent. This technique has been shown to avoid local maxima during optimisation [6]. It is also covariant in the choice of parameters so

rescaling of parameters does not distort learning. If G is the Fisher Information matrix and α is an arbitrary scalar, the update equation is:

$$\triangle \boldsymbol{\theta} = \alpha G^{-1} \nabla_{\boldsymbol{\theta}} V^{\pi_{\boldsymbol{\theta}}} \tag{2}$$

As shown in [6], these gradient vectors can be estimated using ordinary least squares. For each dialogue, two statistics are needed,

$$\boldsymbol{\psi}_t = \sum_{k=1}^{n_t} \gamma^k \nabla_{\boldsymbol{\theta}} \log \pi(a_k | \boldsymbol{b}_k, \boldsymbol{\theta}) \tag{3}$$

$$y_t = \sum_{k=1}^{n_t} \gamma^k r_k \tag{4}$$

where t denotes which dialogue is under consideration, n_t is the number of turns in the dialogue, a_k is the k^{th} action taken and $\boldsymbol{b}_k$ is the k^{th} observed belief state. A vector $\boldsymbol{w}$ and scalar J are then chosen to minimise the sum of squares error:

$$\sum_t (\boldsymbol{w} \cdot \boldsymbol{\psi}_t + J - y_t)^2 \tag{5}$$

Adjustments can be made to reduce the weighting of earlier dialogues, since they will have used unoptimised parameter values. After a suitable number of iterations, the parameters are updated by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \boldsymbol{w} \tag{6}$$

3.1. Learning with BUDS

In a dialogue system, learning over the full action set is unnecessary since some actions will always be suboptimal. For example, if a system confirms the value of a particular slot it should always confirm the most likely value. It will never be optimal to confirm less likely values. This prior information is included in the learning process by grouping such actions together. The groups are called *summary actions* and play a similar role to the summary actions of the HIS model [3].

Typical summary actions will be to "request", "confirm" the most likely value or to "select" between the top two values for a particular slot. An example of such a summary action would be to "confirm the most likely food type". Similarly, offers of database entities matching the users constraints can be combined to form a single summary "offer" action. The details of what happens when taking such a summary action can then be hand-crafted by the system designer. Formally, the POMDP's action set is reduced and the system becomes a continuous state Markov Decision Process with fewer actions available. Assuming one "offer" action and "request", "confirm" and "select" actions for each slot, the number of available actions to the system will be three times the number of slots plus 1. This approach of grouping together actions is the first simplification made while learning in the BUDS model.

The next simplification comes in the design of the summary vector functions, $\boldsymbol{\phi}$. Since the dialogue state is factored according to the slots in the system, the summary functions are also factored accordingly. Any further information can be encoded by including an extra summary factor. Formally, the summary function for summary action a is split according to the slot factor beliefs $\boldsymbol{b}_i$ which record the belief distribution over (g_i, h_i), and an auxiliary summary $f(\boldsymbol{b})$:

$$\boldsymbol{\phi}_a(\boldsymbol{b}) = \sum_{i \in \mathcal{I}} \boldsymbol{\phi}_{a,i}(\boldsymbol{b}_i) + \boldsymbol{\phi}^*_a(f(\boldsymbol{b})) \tag{7}$$

Each of the factor summaries, $\boldsymbol{\phi}_{a,i}$, summarise the information relevant for a particular slot into a vector. Although not required, it is preferable that the summaries do not interact so $\boldsymbol{\phi}_{a,i}$ and $\boldsymbol{\phi}_{a,j}$ are chosen to never have nonzero values at the same vector index when $i \neq j$. Similarly, the auxiliarly function should only take nonzero values at indices where the slot summaries are zero.

In the example of a Tourist Information System, the $\boldsymbol{\phi}_{a,i}$ are evaluated by splitting into five cases determined by how the summary action a relates to slot i. Either the slot is being requested, confirmed or selected by action a, an offer of a venue is made, or another slot is talked about. The $\boldsymbol{\phi}_{a,i}$ are then defined as selector functions using this information along with a grid-based summary of the belief. The resulting vector will have zeroes everywhere except for a 1 corresponding to the relevant grid region and action case. The slot belief space is split into seven fixed regions based on the probabilities of the two most likely values for the slot. In the current implementation, the grid groups together all slot beliefs where the most likely value is less than 0.4. Other belief points are mapped to the closest point in the set $\{(1,0), (0.8,0), (0.8,0.2), (0.6,0), (0.6,0.2), (0.6,0.4)\}$. The first element in these tuples is the probability of the most likely value and the second is the probability of the second most likely value. Note that a grid based approximation is possible at the slot level because the dimensionality is so much smaller. At the overall level this summary causes a form of tiling [5].

Venue offers are grouped together into a single offer action. First the system goes through all slots checking if their probability is higher than a threshold. If the set of such slots allows no venues, then an offer action will tell the user that there is no such venue. Otherwise the system will give the user information about the venue that most likely matches the system beliefs. The exact information that is given is handcrafted and can use the history beliefs to give the most likely requested slots.

In order to help learn when to offer, the auxiliary summary of belief $f(\boldsymbol{b})$ stores how close in probability the venues are to one another. There are three possible cases: one venue is much more likely than any others, there is a group of two or three venues that are much more likely or there are a large number of venues that are approximately equally likely. Depending on which of these cases is observed and whether an offer is made or further information is requested, the $\boldsymbol{\phi}^*$ function will set an appropriate index in the vector to 1 and leave all others 0.

4. EVALUATION

The proposed framework was evaluated against competitive baseline systems on a Tourist Information System task by using semantic level simulations. An agenda-based user simulator was developed [7] and a simulated error channel using random substitution, deletion, and insertion errors was used to test robustness. The same simulator was used in both training of learned policies and for testing of the various approaches.

The error channel assumes a fixed N-best list which represents the word-level output of a speech recogniser. Each item in the list is scrambled randomly with probability e and is decoded to the true semantic user act with probability $1-e$. From the N-best list a confidence for each act is computed by counting the number of repetitions of each act and using Bayes Theorem.

The BUDS belief update framework was implemented as in Section 2.1. A hand-crafted policy (BUDS-HDC) for the framework was built by accepting the most likely value for a slot if its probability was greater than 0.8. If sufficient information for a venue recommendation was available, the hand-crafted system offered the information. Otherwise the system tried to implicitly confirm, confirm, select or request (in that order) one of the low probability slots.

Slots where the most likely value had probability above 0.5 were confirmed or implicitly confirmed. Slots where this probability was below 0.4 were requested. If the probability of the most likely value was between 0.4 and 0.5, the user was asked to select between the most likely two values. As a variation on this approach, the evaluation also included a system that uses the same approach but only uses the most likely user act for any turn (BUDS-HDC-MLUA).

As described in Section 3.1, policy learning was implemented using a factored summary function with the episodic Natural Actor Critic algorithm (BUDS-LEARNED). The confusion rate e was set to 0.4. Parameters were intitialised to zero and were updated every 500 dialogues. Figure 3 shows how the policy improves over time and eventually scores slightly higher than the hand-crafted policy. The system was rewarded with 20 points if the system gave a venue that matched all the users constraints and also gave all the requisite information (phone, address, etc). No points were given when this was not achieved. One point was removed for every dialogue turn.

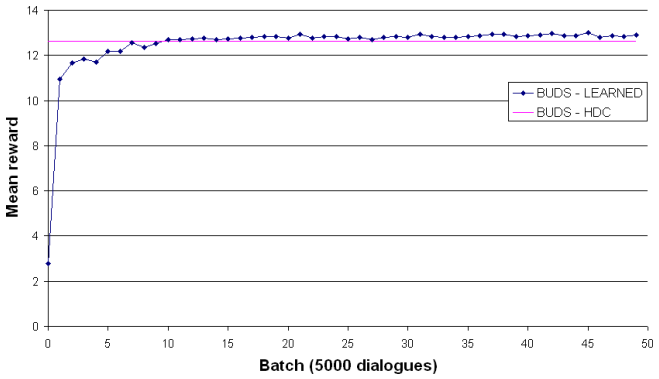

Fig. 3. A plot of the mean reward for the BUDS learned policy during training.

The traditional approach to building dialogue managers is to use either a finite state Markov Decision Process (MDP) model, or to hand-craft a policy (HDC). Both these approaches take only the most likely output from the recogniser, since it is usually impractical to use more. Two such systems were built by maintaining a set of slots, where the slots are either unknown, known or confirmed. At each turn, the slots are updated for the top user act hypothesis. Based on this information the hand-crafted policy chooses to offer information about a particular venue, or to request or confirm a slot. The MDP policy uses the same information but implements standard reinforcement learning techniques to optimise the choice of action [5].

Figure 4 shows the results of a comparison of the various systems at different error levels. As can be seen from the graph, system performance is comparable at low error rates while at higher rates differences develop. Firstly, the use of a full probabilistic framework significantly outperforms using only the most likely hypothesised user act. This is particularly evident when comparing the BUDS hand-crafted policy with a full distribution of possible acts against the same policy using only the most likely act. Secondly, the BUDS framework shows good improvements over traditional approaches, even when the same information is used. This is a consequence of using a systematic approach to handling conflicting evidence from different dialogue turns. Finally, the learned BUDS policy is able to improve on the hand-crafted policy at all error levels. This suggests that the learning has been effective and that if a suitable learning technique is applied, performance gains can be realised.

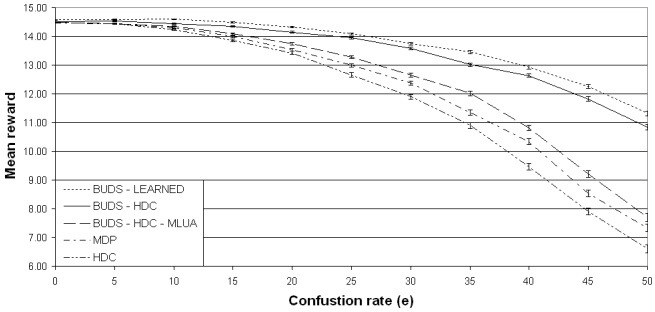

Fig. 4. A comparision of the performance of five different systems. Each point represents the mean after 5000 dialogues. Error bars show the estimated standard deviation of the mean.

5. CONCLUSIONS

This paper has introduced a new Bayesian Update of Dialogue State approach to maintaining beliefs in dialogue systems. Based on the POMDP model, the system has a strong mathematical basis, but because of the conditional independence assumptions taken, updating can be done in a tractable way. Policies may be hand-crafted or learned using reinforcement learning algorithms.

An implementation of this framework was trained and tested using simulated interactions. The framework outperforms traditional approaches even when using exactly the same information. Furthermore, the framework naturally incorporates N-best outputs and significantly improves when given a full distribution over user acts. A novel summary function for learning was also designed for the framework. The resulting learned policy was shown to have the best performance of all systems tested. It is a well-documented problem that performance with simulations and real users are not always comparable [8]. Future work will need to evaluate the performance of the system with real users.

6. REFERENCES

[1] D. Bohus and A Rudnicky, "Constructing accurate beliefs in spoken dialog systems," in *Proc. of ASRU*, 2005.

[2] J. Williams and S. Young, "Partially Observable Markov Decision Processes for Spoken Dialog Systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 231–422, 2007.

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

[4] F. Jensen, *Bayesian Networks and Decision Graphs*, Statistics for Engineering and Information Science. Springer, 2001.

[5] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 1998.

[6] J. Peters, S. Vijayakumar, and S. Schaal, "Natural actor-critic," in *Proc. of ECML*. 2005, pp. 280–291, Springer.

[7] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-based user simulation for bootstrapping a pomdp dialogue system," in *Proc. of HLT/NAACL*, 2007.

[8] J. Schatzmann, M. N. Stuttle, K. Weilhammer, and S. Young, "Effects of the user model on simulation-based learning of dialogue strategies," in *Proc. of ASRU*, 2005.