Practical dialogue manager development using POMDPs

Trung H. Bui, Boris van Schooten, and Dennis Hofs
University of Twente
PO Box 217 Enschede, The Netherlands
{buith, schooten, hofs}@ewi.utwente.nl

Abstract

Partially Observable Markov Decision Processes (POMDPs) are attractive for dialogue management because they are made to deal with noise and partial information. This paper addresses the problem of using them in a practical development cycle. We apply factored POMDP models to three applications. We examine our experiences with respect to design choices and issues, and compare performance with hand-crafted policies.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) are attractive for dialogue management in the cases where the dialogue manager has to make choices which depend on statistical information. They can determine optimal strategies in the face of error and partial information. POMDPs can take advantage of statistical information about behavior or error to the fullest extent, and take into account extensive hidden information.

Using POMDPs for spoken dialogue management has been examined thoroughly in (Williams and Young, 2007). Current POMDP-based dialogue managers model a complete slot-filling dialogue including all slots with all values. Large numbers of slots and values lead to a large state space, which is not tractable for current POMDP solvers. Usually, this restricts us to toy problems. Recent effort to scale up POMDP-based models is reported in (Williams and Young, 2007; Bui et al., 2007).

It is not yet clear enough how to employ POMDPs in a systematic development cycle. A number of practical issues with POMDPs has not really been addressed yet. How do you obtain the user model and the probability distributions? How do you test and debug POMDPs? How do you tweak reward values? How do you evaluate and compare performance of the POMDP policy which other approaches? We address these questions by using the factored POMDP models (Williams and Young, 2007; Bui et al., 2007) as a basis, and applying them to three dialogue management systems.

2 Methodology

Design guidelines. The state space represents the user's state and action. It is defined as a set of features. We should keep it compact. This can be done by specifying only features which are relevant in selecting the system action and by pruning all unreachable states. For example, when analyzing the Williams's 1945-state travel problem (Williams and Young, 2007), we found that could increase tractability by pruning 1626 states[1], leaving only 319 reachable states. The system actions are not only the actions toward the user but also actions for other dialogue manager tasks such as querying the database. Similar to the state space, the observation space is also defined as a set of observation features such as user's action with noise (from the ASR) and observed user's emotional state.

Designing a reward model that leads to a good policy is a very challenging task. The typical parameters used to design a reward model are task success, the number of turns, and dialogue act appropriateness (for example, the system should not confirm a value if it has not yet been provided by the user). The precise numerical values used may have significant impact on the policy and convergence behaviour.

Evaluation setup and toolset. From the literature, the typical approach is first to test the quality of the POMDP-based dialogue policy with a simulated user. The real-user evaluation is considered at the

---

[1] For example, the states which the user's goal feature is ab and the user's action feature is c

final step. An advantage of modeling dialogue as a POMDP is that we can use the POMDP environment model itself as a simulated user model. The probability distributions of the simulated user (testing model) might be varied with the ones of the dialogue manager (training model). The probability distributions of all the user models used in our three applications are handcrafted. We have developed a software toolkit to conduct our experiments, which includes a factored POMDP to flat POMDP translator, and an interactive simulator for both the user and the system. The POMDP problem is first solved with a POMDP solver (we used Perseus (Spaan and Vlassis, 2005) and ZMDP (Smith and Simmons, 2005)). The generated alpha file is then used to carry out the performance test with simulated user models. Section 3 shows our test results on three different problems. We conducted a large number of dialogue episodes ($\geq 10,000$) to guarantee the statistical significance.

3   Evaluation

Ritel QA dialogue system.   Ritel (Galibert et al., 2005) is a telephone-based question answering (QA) dialogue system.   Dialogue functionality includes confirmation of key phrases and the type of the answer sought, and handling follow-up questions. In our model, we focus on confirmation, modeling key phrases and answer type as slots. In the real system, there are thousands of possible key phrases, but answer type only has a few possible values. To make it tractable, we simplified the model to one slot with between 3 and 10 values, suitable at least for modeling answer type fully.

The POMDP state space consists of the user goals and the user actions ($S = G_u \times A_u$). The user goals are the different questions or question types that the user may ask, $G_u = q_1, ..., q_n$. The user actions are composed of the questions, plus positive and negative feedback, a 'bye' utterance, and a 'hang-up' signal, $A_u = q_1, ..., q_n, pos, neg, bye, null$. The observation set Z is the same as Au. The system actions consist of confirming each question, answering it, and the 'ask' action, asking an open question to the user, $A = confirm_{q_i}, answer_{q_i}, ask$. When the system answers the correct question, the user poses a new question, otherwise the user either repeats or

gives negative feedback. The user may hang up in any dialogue turn, with a fixed probability 0.1.

We made the reward model as simple as possible: give a reward of 1 for answering the right question, -1 for answering a wrong one, zero otherwise. We found that modelling dialogue state was not necessary, and it increases state space to intractable levels. This model yields the desired behaviour, though like Williams et al., we found that the system starts confirming even when the user has not yet said anything. This can be remedied by rewarding the *ask* action with a reward slightly more than 0. Note that it is not necessary to give an explicit penalty for dialogue length. The problem can be translated as: answer as many questions as possible before the user hangs up. The results of Perseus were not useable, so the experiments were done with ZMDP only. Convergence was good up till nine slot values. We observed that, when the ASR error becomes high, 0.7 or above, the system actually wants to hear a question multiple times in a row before answering it. The policy was compared to a hand-crafted policy (figure 1), similar to the actual Ritel policy, which is based on counting the number of times a particular keyword was heard. It was optimised to each particular problem by determining the optimal number of times a question should be heard before confirmation is sufficient, just as the POMDP does.
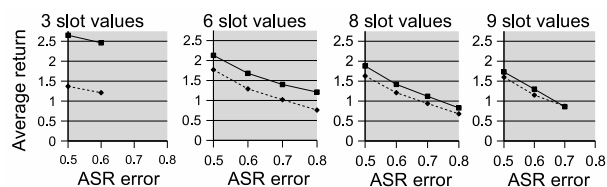

Figure 1: Performance comparison of POMDP and optimised hand-crafted models for different problem sizes and ASR error rates. The solid line is the POMDP, the dashed line is the hand-crafted model. For three values, an error more than 0.6 would result in the probability of hearing the wrong question being higher than the right one. For nine values and error=0.8, no sensible policy could be calculated.

ICIS route navigation system.   In the ICIS project[2], we are developing a multimodal human-

---

[2]http://www.icis.decis.nl/

216

computer framework for crisis management (Fitrianie, 2007). A subtask of the system is to assist rescuers to find a route description to evacuate victims from an unsafe tunnel. This task has been implemented as a multimodal route navigation dialogue system (Bui et al., 2007).

The simplified POMDP for this problem (one slot case) is represented by $S = \langle G_u \times A_u \times E_u \times D_u \rangle = \langle \{v_0, ..., v_m\} \times \{v_0, ..., v_m, yes, no\} \times \{stress, nostress\} \times \{notstated, stated\} \rangle$, $A = \{ask, confirm\text{-}v_0, ..., confirm\text{-}v_m, ok\text{-}v_0, ..., ok\text{-}v_m, fail\}$, and $Z = \langle OA_u \times OE_u \rangle = \langle \{v_0, ..., v_m, yes, no\} \times \{stress, nostress\} \rangle$. The full flat-POMDP model is composed of $(4m^2 + 8m + 1)$ states (including a special `end` state), $(2m + 2)$ actions, and $(2m + 4)$ observations.

The transition and observation models are generated from the two time-slices Dynamic Decision Network (Bui et al., 2007). We assume that the observed user's action only depends on the true user's action (i.e. $P(oa_u|a_u) = (1 - p_{oa})$ if $oa_u = a_u$, otherwise $P(oa_u|a_u) = 1/(m+1) \times p_{oa}$). The observed user's emotional state is computed in a similar way. The reward model is defined as follows: if the system `confirms` when dialogue state is `notstated`, the reward is `-2`, the reward is `-5` for action `fail`, the reward is `10` for action `ok-x` where $g_u = $ `x` ($x \in v_0, ..., v_m$), otherwise the reward is `-10`. The reward for any action taken in the absorbing `end` state is `0`. The reward for any other action is `-1`.

We set different values for parameters m, p_e, p_{oa}, p_{oe}[3] and use two POMDP solvers Perseus and ZMDP to compute the near-optimal policy. Previous research showed that the optimal policy depends on the user's stress level in case $p_e > 0$ and the POMDP policy outperforms hand-crafted policies (Bui et al., 2007). The size of the state space of POMDP model increases as the square of slot numbers and computing the optimal policy is not possible when the number of slot values is greater than 30 because the POMDP parameter file size rapidly increases (for example with $m = 30$, the size is bigger than 200MB). Therefore, the POMDP solver got stuck in initializing the problem.

An alternative solution is to use DDN-POMDP (Bui et al., 2007) or summary POMDP (Williams and Young, 2007). However, when the number of slot values is greater than 100, the belief update task is not tractable. Therefore, a further research on the POMDP problem representation is necessary. A practical issue is that ZMDP is more suitable for the more complex problem ($10 \leq m \leq 30$). This is because ZMDP is able to handle a larger state space by more effective use of sparsity (Smith and Simmons, 2005). On the other hand, Perseus solves small problems very well. The reason for this was not theoretically indicated in the Perseus paper, but they found the same result when testing with the standard POMDP problems from the literature.

Virtual Guide application. The Virtual Guide is a character in a Virtual Reality model of the Music Centre in Enschede (Hofs et al., 2003). The character can help users find their way in the building. It encompasses a multimodal dialogue system that allows users to refer to locations and objects with spoken or written language or by pointing at a location on a map. The system uses clarification questions and implicit confirmations. The user can continue a dialogue with follow-up questions.

It is currently impossible to create a tractable POMDP model for the system. In our simplified models the user can only ask for the route between two objects, and the world is limited to three or eight objects. Moreover we made a closed model where follow-up questions are not allowed. We have fit the problem into a POMDP dialogue model of Williams. A reward is given when the system gives the correct route and the user provided both locations.

Evaluations were performed for four models. For each of them we compared the solutions of Perseus and ZMDP and an adapted hand-crafted system. The solvers were run for ten minutes, when convergence was usually slowing down, although it had not always reached a desirable level. We then ran dialogues with an automatic user simulation based on the user model of the POMDP.

The first model stops after giving any answer, has observation error 0.2, and three locations. We varied the observation error of the simulator. The results in figure 2 show that with increasing errors the POMDP solutions produced higher returns than the

---

[3]p_e is the probability of the user's action error being induced by stress. p_{oa} and p_{oe} are the probabilities of the observed user's action and observed user's stress errors.
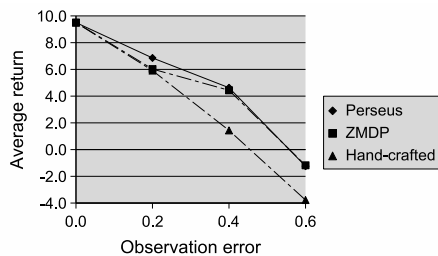
hand-crafted system.

Figure 2: Average returns for simulation with different observation errors

For the second model, we increased the observation error to 0.6. The Perseus solution contained a state from which the dialogue never ended. ZMDP did not converge acceptably. Therefore its solution performed worse than the hand-crafted system.

The third model has observation error 0.2 again, but the dialogue only stops after giving a correct answer. The average returns for Perseus, ZMDP and the hand-crafted system were 8.08, 6.84 and 6.69 (higher than the first model, because of a reward for an extra system action).

In the last model we increased the number of locations to eight, resulting in 729 POMDP states instead of about 80. Perseus was not able to load this problem. The average returns obtained with ZMDP and the hand-crafted system were 5.08 and 4.04.

4 Conclusions

Although our experiments indicate that POMDP-based dialogue systems can perform better than hand-crafted ones, we identified several problems with modelling them. One of the major problems remains tractability. It is not possible to obtain useful solutions for any but strongly simplified models, which may bear little relation to the original problem. For example, when reducing the number of slot values, the strategy of trying them one by one can be employed, something that may not have been feasible for the original number of values. Another example was the need to simplify an open model, where the end of a dialogue is determined by the user, to a closed model.

The definition of a good reward model is another hard problem. While the reward model models psychological factors such as user satisfaction, which

cannot easily be quantified precisely, the POMDPs proved very sensitive to small changes in the reward model, in particular the relative magnitude of different types of reward. In practice we had to experiment with different reward values.

The POMDP policies sometimes came up with surprising strategies. For example, some policies decided to confirm multiple times in a row, something which our original hand-crafted models did not. We could significantly improve performance of the hand-crafted policies by adapting them according to the strategies found by the POMDP policies. This shows how POMDPs could be used to improve hand-crafted systems.

References

T.H. Bui, M. Poel, A. Nijholt, and J. Zwiers. 2007. A tractable ddn-pomdp approach to affective dialogue modeling for general probabilistic frame-based dialogue systems. In *Proc. of the 5th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 34–37.

S. Fitrianie. 2007. A multimodal human-computer interaction framework for research into crisis management. In *Proc. of the Intelligent Human Computer Systems for Crisis Response and Management*, pages 149–158.

O. Galibert, G. Illouz, and S. Rosset. 2005. Ritel: an open-domain, human-computer dialog system. In *Interspeech 2005*, pages 909–912.

D. Hofs, R. op den Akker, and A. Nijholt. 2003. A generic architecture and dialogue model for multimodal interaction. In *Proc. of the 1st Nordic Symposium on Multimodal Communication*, pages 79–92.

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

M.T.J. Spaan and N. Vlassis. 2005. Perseus: Randomized point-based value iteration for pomdps. *Journal of Artificial Intelligence Research*, 24:195–220.

J.D. Williams and S. Young. 2007. Partially observable markov decision processes for spoken dialogue systems. *Computer Speech and Language*, 21:393–422.