Lossless Value Directed Compression of Complex User Goal States for Statistical Spoken Dialogue Systems

Paul A. Crook, Oliver Lemon

Interaction Lab,
School of Mathematical and Computer Sciences (MACS)
Heriot-Watt University, Edinburgh, UK
{p.a.crook, o.lemon} @hw.ac.uk

Abstract

This paper presents initial results in the application of Value Directed Compression (VDC) to spoken dialogue management belief states for reasoning about complex user goals. On a small but realistic SDS problem VDC generates a lossless compression which achieves a 6-fold reduction in the number of dialogue states required by a Partially Observable Markov Decision Process (POMDP) dialogue manager (DM). Reducing the number of dialogue states reduces the computational power, memory, and storage requirements of the hardware used to deploy such POMDP SDSs, thus increasing the complexity of the systems which could theoretically be deployed. In addition, in the case when on-line reinforcement learning is used to learn the DM policy, it should lead to, in this case, a 6-fold reduction in policy learning time. These are the first automatic compression results that have been presented for POMDP SDS states which represent user goals as *sets* over possible domain objects.

Index Terms: spoken dialogue systems, SDS, statistical dialogue management, POMDP, automatic belief compression, ABC, value directed compression, VDC, complex user goals.

1. Introduction

One of the main problems for a spoken dialogue system (SDS) is to determine the user's goal (*e.g.* plan suitable meeting times or find a good Indian restaurant nearby) under uncertainty, and thereby to compute the optimal next system dialogue action (*e.g.* offer a restaurant, ask for clarification). Recent research in statistical spoken dialogue systems has successfully addressed aspects of these problems through the application of Partially Observable Markov Decision Process (POMDP) approaches [1, 2]. However POMDP SDS are currently limited by an impoverished representation of user goals adopted to make systems computationally tractable.

Work in dialogue system evaluation, *e.g.* Walker et al. [3] and Lemon et al. [4], shows that real user goals are generally *sets of items*, rather than a single item. People like to explore possible trade offs between the attributes of items. Thus, as was pointed out by Crook and Lemon [5], a central challenge for the field of spoken dialogue systems is to develop realistic *large-scale* statistical approaches with an accurate, extended representation of user goals.

2. Background

2.1. POMDP for SDS DM

POMDPs are Markov Decision Processes where the system's state is only *partially observable*, *i.e.* there is *uncertainty* as to what the true state is. The ability to account for uncertainty is crucial for SDSs because their knowledge about the state is uncertain due to speech recognition errors and the fact that the user's goals are not directly observable. In POMDP models of spoken dialogue [6, 1, 2] the dialogue policy (what the system should say next) is based not on a single view of the current state of the conversation, but on a probability distribution over all possible states of the conversation (this is denoted as the system's *belief* b). The optimal POMDP SDS dialogue act thus automatically takes account of the uncertainty about the user's utterances and goals.

Formally, a POMDP is defined as a tuple $\langle S, A, O, T, \Omega, \mathcal{R} \rangle$ where S is the set of states that the environment can be in, A is the set of actions that the system can take, O is the set of observations which it can receive, T is a set of conditional transition probabilities which describe the likelihood of transitioning between states given a selected action (*e.g.* $P(s' \mid s, a)$ where $s, s' \in S$ and $a \in A$), Ω is a set of conditional observation probabilities which describe the likelihood of each observation occurring (*e.g.* $P(o' \mid s', a)$ where $o' \in O$), and $\mathcal{R}$ is the reward function ($\mathcal{R} : A, S \rightarrow \mathbb{R}$).

For SDS DM we say that the user's utterance after it has been rendered into the form of a semantic act (or the list of semantic acts[1]) is the observation o which the POMDP receives. We assume that the dialogue has a discrete number of states which it can be in and these are represented by the set of POMDP states S. Finally the DM action is equated to the POMDP act a. Now given a set of transition matrices, observations vectors, and an initial belief b_0 the POMDP DM can monitor and update its belief b over the possible states of the dialogue.

2.2. The Need for State Space Compression

Even considering limited domains, POMDP state spaces grow very quickly. For example, consider finding a user's restaurant preference, which involves getting 4 pieces of information, *i.e.* food type, city area, price range, quality rating. Given 8 food types, 8 city areas, 3 price ranges and 3 quality ratings, coupled with 7 user actions and a 3^4 dialogue progress indicator[2] then the dialogue state space contains $8 \times 8 \times 3 \times 3 \times 7 \times 3^4 = 326,592$ states. A POMDP belief space is a probability distribution over all these dialogue states, *i.e.* a $326,592$ *dimensional* real valued ($\mathbb{R}$) space.

In order to render such large belief spaces tractable, the current state of the art in POMDP SDS uses a variety of *hand-crafted* compression techniques, such as making several types of independence assumption. For example, by assuming that users are only ever interested in one type of food or one location, and that their interests in food type, price range, quality,

[1] In the case of a system that considers N-best lists of ASR output.

[2] Whether each piece of information is filled, confirmed or unknown.

etc. are independent, the $326,592$ real valued state space can be reduced to a much smaller "summary space" [6] consisting of, say, $4 \times \mathbb{R}$ values[3].

3. Related Literature

The tight coupling between some dialogue states and actions (*e.g.* a user's goal state `travel-from-London` and system act `confirm-from-London`) has led some researchers to conclude that compression techniques, such as state aggregation, are not useful in the dialogue domain [7]. However, such tight coupling may not exist for all states, indeed VDC has already been applied to a toy spoken dialogue system problem [8] where it was shown that compressions could be found; losslessly compressing a test problem of 433 states to 31 basis functions. In addition, our introduction of sets to represent user goals should provide additional possibilities for compression.

4. Method

4.1. Sets of User Goals

Work to date on POMDP SDS has assumed that a users have a singular, fully constrained, fixed goal. To achieve a substantial gain in the naturalness and flexibility of SDS we need to allow user's goals that are *sets* of such goals. Maintaining beliefs over "sets of goals" allows a POMDP DM to refine its belief in the user's requirements (managing speech recognition errors) without forcing the user to specify a singular tightly constrained goal.

The type of SDS task that we focus on is a limited-domain query-dialogue, also known as a "slot filling" task. We consider a generic system that has knowledge about some set of *objects* where these objects have *attributes* and these attributes can take several *values*. An object can thus be *described* by a conjunction of attribute-value pairs. A dialogue progresses with the system obtaining requirements from the user which are specified in terms of attribute values. The system should eventually present objects (search results) based upon its understanding of the user's requirement. The dialogue ends when the user accepts one of the domain objects.

The POMDP state representation that we use is the *set* of possible user goal sets. Each state represents a set of objects that the user is willing to accept and can be expressed as a disjunction of object descriptions (conjunctions of attribute-value pairs). The full state space is the complete set of possible combinations of disjunctions of object descriptions, *e.g.* given two attributes u, v which can take the values $u1$, $u2$, $u3$ and $v1$ respectively, then the complete state space consists of $2^3 = 8$ states as listed in Table 1 below.

Table 1: *Example of complex user goal sets.*

state	user goal set
s_1	$\emptyset$ (empty set)
s_2	$u{=}u1 \wedge v{=}v1$
s_3	$u{=}u2 \wedge v{=}v1$
s_4	$u{=}u3 \wedge v{=}v1$
s_5	$(u{=}u1 \wedge v{=}v1) \vee (u{=}u2 \wedge v{=}v1)$
s_6	$(u{=}u1 \wedge v{=}v1) \vee (u{=}u3 \wedge v{=}v1)$
s_7	$(u{=}u2 \wedge v{=}v1) \vee (u{=}u3 \wedge v{=}v1)$
s_8	$(u{=}u1 \wedge v{=}v1) \vee (u{=}u2 \wedge v{=}v1) \vee (u{=}u3 \wedge v{=}v1)$

This representation is flexible enough to handle negative constraints, such as a user requesting "not $u1$", which can be represented in this state space as the disjunction of the remaining domain object descriptions that do not contain $u1$, *i.e.* $(u{=}u2 \wedge v{=}v1) \vee (u{=}u3 \wedge v{=}v1)$.

A simple example of the target dialogues we are aiming for is given in Table 2.

Table 2: *Example target dialogue. S=System, U=User.*

User goal: cheap central Italian or expensive Thai restaurant
S: Hello, how can I help you?
U: I'm looking for a central, cheap Italian restaurant.
S: Was that Italian?
U: Yes, or expensive Thai.
S: Okay, Ayutthaya is a great Thai restaurant ...
U: What else do you have?
S: Italian Connection serves cheap but wonderful food ...

Unlike many DM state spaces this state space does not include any state features which indicate dialogue progress, such as numbers of filled and confirmed slots. For this particular task we do not consider such state features are necessary as the belief distribution maintained over the set of states is sufficient to determine the dialogue progress and thus the next system act.

Of course this approach of using sets significantly expands the state space of possible user goals, with the number of goal sets being equal to 2 to the power of the number of object descriptions.

4.2. VDC Algorithm

We use the VDC algorithm *Krylov iteration for lossless compression* from Poupart [8] to compute the reduced state space. This algorithm is initiated by constructing a vector for each action a where that vector contains the reward associated with each state s. Of these initial vectors only those that are linearly independent[4] are retained. These form our initial basis vectors and Krylov iteration is then applied to generate further basis vectors. The algorithm can be understood as performing iteratively deepening projections of these initial vectors using the state transition and observation probabilities of the POMDP problem. At each iteration the newly generated vectors are tested to see if they are linearly independent of the existing set of basis vectors. If they are, they are added to the existing set and themselves projected in the next iteration. The algorithm halts when all the basis vectors have been projected forward one step and no new linearly independent vectors have been generated, or when the number of basis vectors equals the number of states s.

The number of basis vectors produced is the size of the compressed state space since the *value*[5] of being in any of the original states s can be constructed with no loss in precision[6] as the linear sum of the set of basis functions. The set of basis vectors can be used to project the POMDP reward, transition, and observation probabilities into the reduced state space, allowing the policy to be learnt and executed in this state space.

4.3. State Transition Probabilities

We consider a version of POMDP SDS where the transition update $P(s' \mid s, a)$ is simplified by making the assumption that users do not change their goal during the course of a dialogue. This is an assumption that other authors, *e.g.* Williams and Young [9], have made in the domain of SDSs.

[3] By considering only the maximum marginal likelihood for each of the attributes.

[4] Cannot be constructed as a weighted summation of the existing set of vectors.

[5] The sum of discounted future rewards.

[6] To the limit of accuracy of a digital computer.

Although we assume the user's goal set is fixed we are not assuming that it is necessarily consciously known by the user at the outset. The user may well only realise their complex goal set at the end of the dialogue. We simply assume that users will act largely in accordance with some consistent goal set.

This assumption is not as strict as it sounds as it is only a specification of the state transition dynamics model and provided there is some recognition uncertainty all goals will remain reachable. In addition it is further mitigated in this work by the representation of user goals not as singular targets but as *sets* of objects. Using this formulation a user who changed their mind would, unless they persistently negated a previous statement, be seen as browsing a set of acceptable objects.

4.4. Observation Probabilities

Given the above assumption with regard to the transition probabilities the observation probabilities become the prime driver in computing the belief state in response to the machine's actions. Ideally these would be derived from SDS corpora but for our investigation we generate frequency counts from an artificial but realistic set of frequency counts as shown in Table 3. In future work we will build these probabilities from real data.

For each given system act a and goal state s the rules in Table 3 assign a frequency count to each possible observation o. These counts are then normalised so that $\sum_{o \in O} P(o|s,a) = 1$.

The frequency counts presume co-operative users who can supply a single complete object description or one attribute value from the set of objects they are interested in during each dialogue turn. The counts assume that users tend to align with the system on which attributes they talk about when asked for an attribute or when an incorrect confirmation is attempted. They also encode that the response to correct confirmations will typically contain the provision of additional information.

As an example consider that a user's goal set consists of $(u{=}u2 \wedge v{=}v1) \vee (u{=}u3 \wedge v{=}v1)$ and the system attempts to incorrectly confirm the value $u1$ for attribute u. A cooperative user will typically respond with values for attribute u that are acceptable. This corresponds to the fourth row in Table 3, `confirm_value`, `value` $\notin$ `goal set`. Scanning along that row the largest count, 600, is associated with observations of the form `ans_no_provide_value` where the provided value is in the goal set and aligns with the attribute that the system tried to confirm. In this example that corresponds to the observations `ans_no_provide_`$u2$ or `ans_no_provide_`$u3$.

5. Experiment

We consider a SDS task which has three attributes. One of the attributes can take three different values, the other two attributes can just take two values. This results in 12 possible object descriptions ($3 \times 2 \times 2$) and 2^{12} or 4,096 goal sets (states).

The system has 23 actions it can choose; `greet`, 3 `ask_attribute`, 7 `confirm_value`, 12 `present_obj` and 49 observations; `answer_yes`, `answer_no`, `ask_something_else`, `silence/out-of-domain`, 12 `provide_obj_des`, 12 `ans_no_provide_obj_des`, 7 `provide_value`, 7 `ans_yes_provide_value`, 7 `ans_no_provide_value`.

The reward structure used was -10 for `present_obj` where the object `obj` description lies outside the user goal set, +10 for `present_obj` where `obj` description lies in the user goal set, -1 for any other system action. By penalising each move this reward structure encourages shorter dialogues whilst the larger penalty for incorrect presentations should prevent the system from adopting a style of repeatedly guessing.

6. Results and Discussion

Using Krylov iteration the 4096 state problem described above compressed to 630 states which is a compression of approximately 6.5 times. The degree of compression obtained is directly related to the number of original states which have similar values, and is thus indirectly related to the observation matrix of the system.

When the basis vectors are composed together in a matrix form, each row forms a mapping from its associated state s to that state's compressed representation. By comparing rows we can thus arrive at some idea of how the problem has been compressed [8]. For example identical rows would indicate that the corresponding states have been simply merged. In this case only unique rows were found indicating that the compression does not consist of simple state aggregation. It is not obvious that the POMDP problem that we have set up will compress. For example, the equivalent task using two values per attribute and thus 8 object descriptions and 256 states failed to compress.

7. Conclusions

We show a lossless 6 fold reduction in the state space of a small but fairly realistic POMDP SDS task. This is the first time compression has been demonstrated for states representing complex user goals. We suspect that tasks with larger numbers of features may well exhibit even greater compression than has been demonstrated here.

VDC was initially developed to allow large scale POMDPs to be solved but has generally been eclipsed by other POMDP solution techniques because the computational load in compressing the POMDP task is often similar to that of simply solving the orginal task. Where VDC is of benefit is that for SDSs we are looking not to just solve POMDP tasks but to run the resulting systems in real time. We have found that storage is the most pressing constraint when developing POMDP SDS DMs. Reducing the number of states by, say, 6-fold results in a similar reduction in the storage required. Depending on the size of the initial problem and the device that it is to be deployed to, this and similar techniques could significantly increase the sophistication of tasks that can be deployed. There is a one-off cost in computing the compression but this can be carried out off-line and then the reduced state system can be deployed and run on-line.

A further advantage is that when on-line reinforcement learning is used to learn the DM policy, it should lead to, in this case, a 6-fold reduction in policy learning time. This is because the number of states that need to be sampled is reduced thus speeding convergence. Also in terms of off-line POMDP policy learning this reduced problem should be solvable in relatively short time as Spaan and Vlassis [10] have demonstrated POMDP policy solving with Perseus on a robot TAG game problem with 850 states in 28 minutes.

VDC and other automated compressed techniques reduce the human design load by automating part of the current POMDP SDS design process. This lowers the benchmark and knowledge required when building such statistical systems and may make them more easier for industry to deploy.

Such compression approaches are not only applicable to SDSs but should be equally relevant for multi-modal interaction systems where several modalities are being combined to try compute the user's state.

Table 3: *Rules for generating observation frequency counts. "provide_obj_des" is the provision of an object description, i.e. a conjunction of the complete set of attribute-value pairs that describe an object, "obj ∈ goal set" and "obj_des ∈ goal set" both mean that the set of conjunctions which describes the object exists in the users goal set, "∗" is a wild card (matches any observation) and "-" indicates a logically impossible entry.*

system act	condition re. system act	frequency counts																								
greet	none	900	0	100	-	-	-	-	-	100	-	-	-	-	-	100	-	-	-	-	-	0	0	0	40	0
ask_attribute	none	30	0	-	500	200	-	-	-	-	250	100	-	-	-	-	400	100	-	-	-	0	0	0	40	0
confirm_value	value ∈ goal set	30	9	-	-	-	60	40	100	-	-	-	90	60	150	-	-	-	60	120	30	150	0	0	40	0
	value ∉ goal set	30	45	-	-	-	-	40	100	-	-	-	-	12	30	-	-	-	-	600	150	0	375	0	40	0
present_obj	obj ∈ goal set	0	0	0	-	-	-	-	-	0	-	-	-	-	-	0	-	-	-	-	-	500	0	100	40	0
	obj ∉ goal set	30	150	0	-	-	-	-	-	0	-	-	-	-	-	100	-	-	-	-	-	0	0	0	40	0

observation (recognised user act)	conditions relating to observation
provide_obj_des	obj_des ∈ goal set
ans_no_provide_obj_des	
provide_value	value ∈ goal set & system act = greet/present
	value ∈ goal set & ask_attribute = provide attribute
	value ∈ goal set & ask_attribute ≠ provide attribute
	value ∈ goal set & value = confirm_value
	value ∈ goal set & value ≠ confirm_value & provide attribute = confirm attribute
	value ∈ goal set & provide attribute ≠ confirm attribute
ans_yes_provide_value	value ∈ goal set & system act = greet/present
	value ∈ goal set & ask_attribute = provide attribute
	value ∈ goal set & ask_attribute ≠ provide attribute
	value ∈ goal set & value = confirm_value
	value ∈ goal set & value ≠ confirm_value & provide attribute = confirm attribute
	value ∈ goal set & provide attribute ≠ confirm attribute
ans_no_provide_value	value ∈ goal set & system act = greet/present
	value ∈ goal set & ask_attribute = provide attribute
	value ∈ goal set & ask_attribute ≠ provide attribute
	value ∈ goal set & value = confirm_value
	value ∈ goal set & value ≠ confirm_value & provide attribute = confirm attribute
	value ∈ goal set & provide attribute ≠ confirm attribute
answer_yes	
answer_no	
ask_something_else	
silence/out_of_domain	
∗	above observation conditions not met

7.1. Future Work

This is a first step in using automatic compression approaches for developing more flexible statistical SDSs and multi-modal interaction systems. There remains a wealth of areas to investigate. For example: examining the variation in compression with increasing numbers of attributes and values; extending the observation matrix to allow for user utterances containing more complex conjunctions and disjunctions of values; optimising the computation time of the Krylov iteration approach to VDC (especially with regard to specialising it for the compression of such limited-domain query-dialogue SDS tasks); investigating alternative methods of generating minimal sets of bases, some of which may produce more human-interpretable compressions, *e.g.* non-negative matrix factorisations.

8. Acknowledgements

The research leading to these results has received funding from the Engineering and Physical Sciences Research Council, UK (EPSRC) project no. EP/G069840/1, the European Commission's 7th Framework Programme (FP7) under grant agreement no. 216594 (CLASSiC project `www.classic-project.org`) under grant agreement no. 270435 (JAMES project), and under grant agreement no. 270019 (SpaceBook project).

9. References

[1] B. Thomson and S. Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech and Language*, vol. 24, no. 4, pp. 562–588, 2010.

[2] S. Young, M. Gašić, S. Keizer, F. Mairesse, B. Thomson, and K. Yu, "The Hidden Information State model: a practical framework for POMDP based spoken dialogue management," *Computer Speech and Language*, vol. 24, no. 2, pp. 150–174, 2010.

[3] M. Walker, S. Whittaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy, "User tailored generation in the match multimodal dialogue system," *Cognitive Science*, vol. 28, pp. 811–840, 2004.

[4] O. Lemon, K. Georgila, and J. Henderson, "Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation," in *IEEE/ACL Spoken Language Technology*, 2006.

[5] P. A. Crook and O. Lemon, "Representing uncertainty about complex user goals in statistical dialogue systems," in *proceedings of SIGdial*, September 2010.

[6] J. D. Williams and S. Young, "Scaling Up POMDPs for Dialog Management: The "Summary POMDP" Method," in *Proc. ASRU*, 2005.

[7] J. Williams and S. Young, "Scaling POMDPs for spoken dialog management," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2116 –2129, Sept 2007.

[8] P. Poupart, "Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes," Ph.D. dissertation, Dept. Computer Science, University of Toronto, 2005.

[9] 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.

[10] M. T. J. Spaan and N. Vlassis, "Perseus: randomized point-based value iteration for pomdps," *J. Artif. Int. Res.*, vol. 24, pp. 195–220, August 2005.