Back-off Action Selection in Summary Space-Based POMDP Dialogue Systems

M. Gašić, F. Lefèvre, F. Jurčíček, S. Keizer, F. Mairesse, B. Thomson, K. Yu, S. Young

Spoken Dialogue Systems Group, Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, UK
{mg436, frfl2, fj228, sk561, farm2, brmt2, ky219, sjy}@eng.cam.ac.uk

***Abstract*—This paper deals with the issue of invalid state-action pairs in the Partially Observable Markov Decision Process (POMDP) framework, with a focus on real-world tasks where the need for approximate solutions exacerbates this problem. In particular, when modelling dialogue as a POMDP, both the state and the action space must be reduced to smaller scale summary spaces in order to make learning tractable. However, since not all actions are valid in all states, the action proposed by the policy in summary space sometimes leads to an invalid action when mapped back to master space. Some form of back-off scheme must then be used to generate an alternative action. This paper demonstrates how the value function derived during reinforcement learning can be used to order back-off actions in an N-best list. Compared to a simple baseline back-off strategy and to a strategy that extends the summary space to minimise the occurrence of invalid actions, the proposed N-best action selection scheme is shown to be significantly more robust.**

I. Introduction

It has been suggested in recent years that modelling dialogue as a Partially Observable Markov Decision Process (POMDP) enables dialogue systems to be built which are more easily trainable, more natural in operation and more robust to recognition errors [1].

In the POMDP approach, the dialogue is characterised as a sequence of unobserved states with probabilistic transitions. The attractiveness of the POMDP lies in the fact that it maintains a probability distribution over all states, thus implicitly modelling the inherent uncertainty that occurs in spoken dialogue, especially that arising from speech recognition and understanding errors. However, algorithms that solve POMDPs exactly have exponential complexity and can be applied only to very simple problems. Even approximate solutions proved intractable for real-world problems with a large state and action space.

Several techniques have been developed to tackle this problem most of which are based on the idea of compressing the state space (the *master space*) into a reduced space (the *summary space*) so that learning can be tractably performed using approximate algorithms [2].

Two main issues arise when using the summary space method. Firstly, there is the problem of finding the optimal compression, *i.e.*, what information from the master space is crucial for learning and what can be omitted. The efficient selection of informative features is a current topic of research [3]. The second issue relates to mapping the summary action back to an appropriate master action, and what to do when such a mapping is not possible. The latter problem is addressed in this paper within the framework of grid-based learning algorithms. It is shown that the value functions learnt during policy optimisation can be used to associate N-best action lists with each grid point. Action selection then involves simply selecting the highest ranking valid action from the appropriate N-best list. This approach is shown to compare favourably with both a simple fixed back-off baseline and an alternative extended summary state scheme.

The paper is structured as follows. In Section II, the general problem of invalid state-action pairs in POMDPs is presented and why this problem becomes even more apparent with large state and action sets is explained. The three approaches mentioned above are then described. In Section III the Monte Carlo Control algorithm is explained in the framework of grid-based learning. For evaluation, the HIS dialogue system is used [4] and Section IV provides a brief overview of the relevant parts of this system especially the master and summary space and the learning process. In Section V, the implementation of the back-off strategies within the HIS framework is presented and the evaluation results are discussed. Finally, conclusions are given in Section VI.

II. The invalid action problem

It is a general property of Markov decision processes that not all actions are valid in every state. In a discrete observable MDP, this problem can be resolved simply by defining for each state a subset of actions that are possible. It is then straightforward during training and operation to ensure that the policy for any state only considers actions which are valid for that state.

In contrast, a POMDP belief state represents a probability over all environment states and since in principle any environment state is possible in any belief state, the policy must include the possibility of taking any action in any belief state. For example, in a dialogue system given some prior information on user preferences, there is nothing in principle to stop the system saying as its very first action: ”Please confirm that you want a Chinese restaurant?”, even though the user hasn’t yet said anything. This problem is exacerbated in a system which compresses actions into simple strategic decisions such as “ask”, “confirm”, etc. In this case, the fact that an action such as “confirm” in summary space is invalid, cannot be determined until it is mapped back into master space where it is discovered that there is in fact no information which can be confirmed.

To understand this issue in more depth, some formal notation is needed. A POMDP is described as a tuple of $\langle S, A, T, R, \Omega, O\rangle$, where S is a set of states, A is a set of actions, R is a set of rewards associated with each state-action pair, T is a transition probability function between states, Ω is a set of observations and O is the observation function $O(s', a, o)$ which is the probability $p(o|s', a)$ of observing o in state s' given that the previous system action was a. Since the state is unobservable, a probability distribution over S, is estimated at each step called the *belief state* b where $b(s)$ represents the probability of being in state s at that step. A policy $\pi : b \rightarrow a$ maps the current belief state into an action. The value function of a belief state b given policy π, $V_\pi(b)$, is the expected reward that can be obtained in belief state b by following policy π. Finding the optimal value function yields the best policy.

Although the belief state b is continuous, a policy can be represented as a tree where the nodes are actions, the branches are all possible observations and the depth is the number of steps needed to complete the process. The value function of a state s given a policy tree p, $V_p(s)$ is the expected reward that can be obtained in that state following policy tree p. The value function of the belief state then becomes $V_p(b) = \sum_{s\in S} b(s)V_p(s)$ – a linear function of $b(s)$. The optimal value function for each step is the upper surface of the value functions associated with all policy trees in that step [5].

As noted earlier, even if the set of valid actions is known for every state, the belief state can assign some probability mass to every state. A policy is a function of the belief and it proposes actions that give the highest expected reward assuming that all actions are valid. Thus, there will normally be a finite probability of issuing an invalid action from any belief state.

If we assume that once an action is proposed it is possible to determine whether or not it is valid, even if the actual environment state s remains unknown, this problem can be dealt with in a couple of ways. One way would be to expand the action set so that a default action which is guaranteed to be valid in any state is associated with every action. However, this can lead to suboptimal results. Consider for example, a simple POMDP with state space $S = \{s_1, s_2\}$ and action space $A = \{a_1, a_2, default\}$, action a_1 is not valid in state s_2 and the rewards are $r(s_1, a_1) = 5$, $r(s_1, a_2) = 2$, $r(s_1, default) = 1$, $r(s_2, a_2) = 3$ and $r(s_2, default) = 1$.

In order to allow action a_1 to be performed in any belief state b, it has to be given a valid interpretation in state s_2. This can be done by redefining action a_1 as $a_1 \rightarrow default$, meaning if action a_1 is not valid back-off to $default$ instead. However, as shown in Figure 1 this is clearly a suboptimal solution compared to $a_1 \rightarrow a_2$ since the optimal 1-step value function is associated with $a_1 \rightarrow a_2$ and not the sub-optimal 1-step value function associated with $a_1 \rightarrow default$.

In the general case, the optimal back-off for action a_i will be a sequence $a_i \rightarrow a_{i_1} \cdots \rightarrow a_{i_j} \cdots \rightarrow a_{i_{n-1}}$ where $n = |A|$ given the belief. Redefining each action to include all possible back-off actions would increase the cardinality of the action set to $n!$ and this is clearly not feasible.

An alternative way of finding the optimal back-off action is to utilise Q-values. The Q-value $Q(b, a)$ is the expected reward from taking action a in belief state b and following the policy thereafter. It can be calculated as $Q(b, a) = \sum_{s\in S} Q(s, a)b(s)$, where $Q(s, a)$ is the Q-value of taking action a in state s. For each action a, the set of states in which that action can be taken is known. Therefore, the learning algorithm can estimate $Q(b, a)$ by summing over only the $Q(s, a)$ values for which action a is valid in s. Ordering $Q(b, a)$ in a list for each belief b provides a sequence of possible back-off actions which can be searched until a valid action is found. This intuition is the motivation behind the N-best back-off action selection mechanism proposed in this paper.

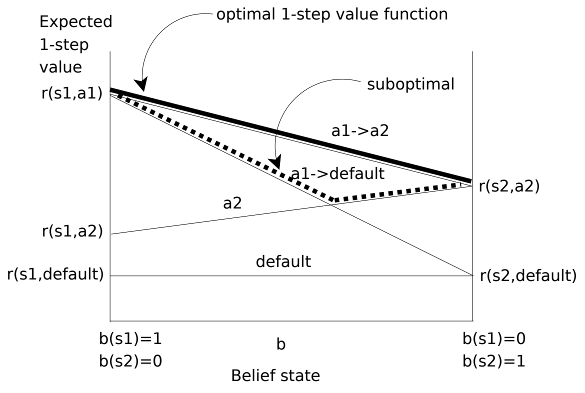

Fig. 1. Two state POMDP with an impossible action-state pair

A major problem with POMDPs is the intractability of exact learning algorithms and hence the need for approximate solutions. A POMDP can be viewed as a continuous MDP [5] and the approximation would then be discretising the continuous space to form a finite grid. The result is a discrete MDP problem which can be solved using standard methods [6].

A detailed algorithm for implementing grid-based learning with back-off action selection is given in Figure 2. Before that however, it should be noted that discretisation of the belief space further exacerbates the invalid action problem since the effect of the discretisation is to represent all of the belief points in a neighbourhood by a single representative grid point. Thus, even if there are no belief points within the neighbourhood with invalid actions, the merging into a single grid point will generate the union of all neighbouring actions and this union might have invalid actions. One possibility to avoid this is to increase the information transferred from master to summary space with the subset of actions that are valid in that summary state. This possibility is straightforward to implement and it is explored further in the experimental section below. However, since it significantly expands summary space, it is likely to lead to some loss in robustness.

III. The Monte Carlo Control Algorithm for grid-based learning

The above discussion has highlighted the problem of invalid actions in POMDP-based systems and identified an approach to solving it based on ranking the alternative actions for each point in belief space into an N-best list ordered by Q-value. It has also been noted that for practical real world systems, it is mandatory to compress the master belief space into a much smaller summary space. This summary space will typically consist of a set of features extracted from the master space. These features can be a mixture of discrete and continuous variables. All that is required is that a distance metric is defined over the resulting space.

In this section, a specific grid-based learning algorithm is described called the Monte Carlo Control algorithm. This is a standard model-free approach to on-line policy optimisation [7] which is especially suited to episodic processes of which dialogue is a prime example. The basic idea is simple. The POMDP summary space is represented by a number of discrete grid points. The system interacts with a user [1] and Q-values are estimated for each grid point and each action. The action with the highest Q-value at each grid point then forms the policy. Extending this algorithm to include N-best action selection is then simply a matter of storing not just the highest Q-value at each grid point but a rank-ordered list.

The main issue in grid-based learning is how to generate grid-points efficiently. The approach used here is to start with a single grid point and then add new points as required during the exploration phase of training. The learning starts by arbitrarily assigning values to the Q-values for each action a associated with the initial grid point $\hat{b}_0$[2]. In addition to the Q values, a counter $N(\hat{b}, a)$ is associated with each grid point and initialised to zero. This counter records the number of times that each action is taken in that grid point. Each learning episode is conducted ϵ-greedily *i.e.*, using the current best policy ($\pi(\hat{b}) = argmax_a Q(\hat{b}, a)$) except that with probability ϵ a random action is taken instead of the action proposed by the policy. In the standard algorithm, the policy consists of one action per grid point – this is the action that has the highest Q-value [7]. For the N-best action selection, the algorithm is modified so that the policy consists of a list of actions per grid point ordered by their Q-values. In a similar way, when exploring instead of generating one random action, a random ordering of actions is generated. In all cases, the reward obtained for each turn is assigned to the Q-value for the action that was actually taken. Every time a new belief state is visited, it is mapped to a summary state and then to the nearest grid point. If there is no nearby grid point a new one is created and added to the set of grid points. Thus, during training grid points are created with their respective lists of Q- and N-values. The complete algorithm is given in Figure 2.

[1]A simulator is often used for this to allow efficient training over a large number of dialogues.

[2]The hat on variables denotes elements the summary space.

```
 1: Let $Q(\hat{b}, a)$ = expected reward of taking action $a$ at grid
    point $\hat{b}$
 2: Let $N(\hat{b}, a)$ = number of times action $a$ is taken at grid
    point $\hat{b}$
 3: Let $\mathcal{B}$ be a set of grid points
 4: Let $\pi : \hat{b} \rightarrow a_{i_1}, \cdots, a_{i_n}; \forall \hat{b} \in \mathcal{B}$ be a policy
 5: repeat
 6:     $t \leftarrow 0$
 7:     $a_{i_1^0} \cdots a_{i_n^0}$ initial random order of actions
 8:     $b = b_0$ initial grid point

        Generate episode using $\epsilon$-greedy policy
 9:     repeat
10:         $t \leftarrow t + 1$
11:         Update belief state $b$
12:         $\hat{b}_t \leftarrow$ GridPoint(SummaryState($b$))
13:         $a_{i_1^t}, \cdots, a_{i_n^t} \leftarrow \begin{cases} \text{RandomlyOrderedActions} \\ \pi(\text{Nearest}(\hat{b}, \mathcal{B})) \end{cases}$
14:         record $\langle \hat{b}_t, a_{i_j^t} \rangle$, $T \leftarrow t$, where $a_{i_j^t}$ is the
             taken action
15:     until episode terminates with reward $R$

        Scan episode and update $\mathcal{B}$, $Q$ and $N$
16:     for $t = T$ downto $1$ do
17:         if $\exists b_k \in \mathcal{B}, |\hat{b}_t - \hat{b}_k| < \delta$ then $\leftarrow$ update pt in $\mathcal{B}$
18:             $Q(\hat{b}_k, a_{i_j^t}) \leftarrow \frac{Q(\hat{b}_k, a_{i_j^t}) * N(\hat{b}_k, a_{i_j^t}) + R}{N(\hat{b}_k, a_{i_j^t}) + 1}$
19:             $N(\hat{b}_k, a_{i_j^t}) \leftarrow N(\hat{b}_k, a_{i_j^t}) + 1$
20:         else                                    $\leftarrow$ create new pt
21:             add $\hat{b}_t$ to $\mathcal{B}$
22:             $Q(\hat{b}_t, a_{i_j^t}) \leftarrow R$, $N(\hat{b}_t, a_{i_j^t}) \leftarrow 1$
23:         end if
24:         $R \leftarrow \gamma R$                       $\leftarrow$ discount the reward
25:     end for

        Update the policy
26:     for all $\hat{b}_k$ with updated $Q(\hat{b}_k, a)$ for any $a$ do
27:         $\pi(\hat{b}_k)$ = actions ordered by $Q(\hat{b}_k, a)$
28:     end for
29: until converged
```

Fig. 2. Monte Carlo Control Policy Optimisation Algorithm extended for N-best back-off action selection

The described approaches are evaluated in the framework of the HIS Dialogue manager which is explained in the next section.

IV. HIS Dialogue Manager

The experimental evaluation of the N-best back-off action selection method has been conducted using the Hidden Information State (HIS) dialogue system. The pertinent features of this system are briefly described here. A full description is given in [4].

A. System Architecture

The HIS dialogue system consists of an ATK speech recogniser [8], an SVM-based semantic tuple classifier, a POMDP dialogue manager, a natural language generator, and an HMM speech synthesiser. When interacting with the system, the user's speech is recognised by the recogniser and, in the form of a scored N-best list, passed to the semantic classifier. The semantic classifier outputs an N-best list of dialogue acts. A dialogue act is a semantic representation of the user action which contains the user intention (such as `inform`, `request`, etc) and a list of slot-value pairs (e.g. `type=hotel, area=east`). The N-best list of dialogue acts is used by the dialogue manager to update the dialogue state. Based on the state and the policy, a system action is produced, again in the form of a dialogue act. The system action is then passed to the natural language generator that converts it to text, which is finally synthesised. The HIS dialogue system is currently implemented for the Town-Info domain. It provides tourist information for an imaginary town. However, no specific domain knowledge is incorporated in the dialogue manager that would not be applicable for any query-driven dialogue.

B. Dialogue Manager

The unobserved dialogue state of the HIS dialogue manager consists of the user goal, the dialogue history and the user action. The user goal is represented as a tree structure – a *partition*, which is built according to a domain ontology. Slots and values are both represented as nodes in the tree. An example of a partition is given in Figure 3. When querying the data base using the partition, a set of matching entities is produced. The dialogue history consists of the grounding states of nodes in the partition, generated using a finite state machine and the previous user and system action. The combination of a partition, a user action from the last N-best input and the respective set of grounding states forms a *hypothesis*. A distribution over all hypotheses (the *belief state*) is maintained throughout the dialogue. Taking into account that any real-world problem would have a non-trivial ontology, it is clear that this belief space will be extremely large, and must therefore be reduced to a smaller scale summary space. An example of the summary space mapping and action selection process is given in Figure 4. In the top part of the diagram, it is shown how belief b in master space is mapped to a summary point $\hat{b}$ that contains only five components.

Since the summary state space does not encapsulate all the information that master actions can contain, these actions cannot be learnt directly. For that reason, the action space is also reduced to a summary action set. Figure 4 shows how the policy π determines on the basis of summary point $\hat{b}$ which of the ten possible summary actions to take.

The optimal policy is obtained using reinforcement learning in interaction with an agenda-based simulated user [9]. At the end of each dialogue, a reward is given to the system based on the completion of the user goal and the efficiency of the dialogue (20 is given for a successful completion and

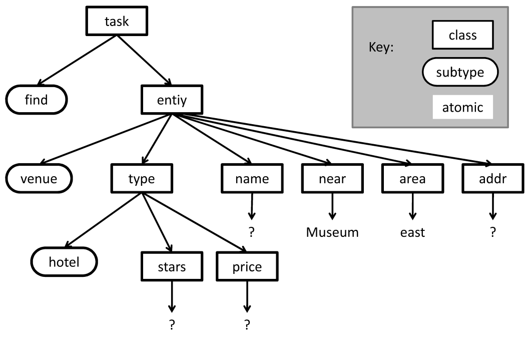

Fig. 3. Example of a partition built according to the domain ontology for the goal `find(type=hotel, area=east, near=Museum)`

-1 for each turn). Policy optimisation used the Monte Carlo Control algorithm described above in Section III. The mapping from summary action back to a master action use heuristics to add the appropriate information from the master space. For example, if the proposed summary action is `Confirm`, then the value pair relating to the node from the partition of the top hypothesis that is not grounded will be confirmed together with its slot. For instance, if the node relating to the value `hotel` is not grounded the action would be `confirm(type=hotel)`, meaning *You are looking for a hotel, right?*

The invalid action problem occurs when the proposed action cannot be mapped to the master space. In the previous example of the summary action `Confirm`, if all goal tree nodes are grounded, there is nothing left to confirm, so this action cannot be performed and an alternative back-off selected.

V. Experiments

This section presents results for the proposed N-best back-off action selection method and compares it with a simple fixed back-off baseline and an alternative based on extending the summary space with features designed to minimise the occurrence of invalid actions. In all cases, the system was trained and tested using a user simulator which incorporates an error model to allow a range of noise levels to be simulated. That is a hand-crafted error model that creates semantic level confusions and confidence scores at different error rates. Item-level Cross Entropy (ICE) of the resulting N-best list decreases as N increases. This indicates that simulated confidence scores are informative [10]. However, in the future we plan to use a statistical error model as in [11].

A. Fixed Back-off

The simplest solution to the back-off problem is to use a single global default action as the back-off action. The requirement for this action is that it can be performed for any point in the belief space. In the HIS system, the only appropriate candidates are the `Repeat` action where the system asks the user to repeat the last input, or the `Bye` action where the system ends the dialogue. In the case of `Repeat`, users typically repeat the last dialogue act or hang up if the

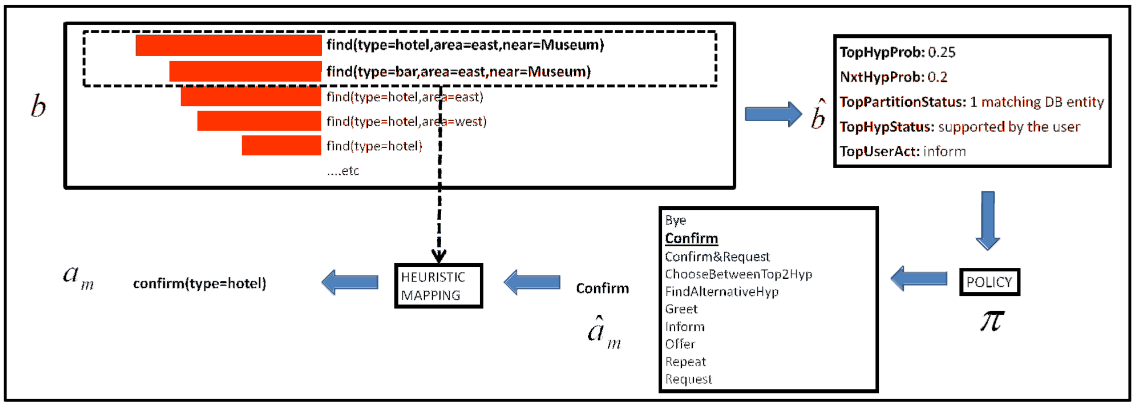

Fig. 4. Action selection process in the HIS framework

system has already asked this several times. By asking the user to repeat the last act, the system can potentially obtain a better estimation of the current user state, but it can also waste time leading to a lower reward. The `Bye` action would only be appropriate for a system which could divert to an operator, otherwise its use would be unacceptable. Hence, the `Repeat` action is used for the fixed back-off baseline.

Training of the fixed back-off system starts in low noise and then incrementally increases it, as in [12]. Approximately $1,000,000$ dialogues were used for training and the total number of grid points was 400. The evaluation on the user simulator performing 5000 dialogues at each of 11 error rates is shown in Figure 5.

B. Extension of the Summary Space

The second approach to the problem of invalid actions is to extend the summary space with explicit information about which action can be taken. In this approach, each summary grid point was augmented with a binary flag to indicate whether or not each summary action is valid for that grid point. Since two belief points that have different subsets of plausible actions cannot be mapped to the same grid point and since there are 11 possible summary actions, this can potentially increase the summary space by a factor of 2^{11}. In practice, however, due to the nature of the problem some actions are always possible and there are also dependencies between them, so in total the extended summary space resulted in a policy with 2000 grid points. The training scheme used for this system was similar to the previous one, with the difference that training had ten times more dialogues to compensate for the increased number of grid points. The performance of this system is shown in Figure 5.

C. N-best Back-off

Q values from the Monte Carlo Control algorithm produce a list of actions ranked by the policy preference associated with each summary state (see Section III). Utilising this N-best list for back-off action selection, the policy was trained using the same training scheme as in the fixed back-off strategy. The performance results for this system are given in Figure 5.

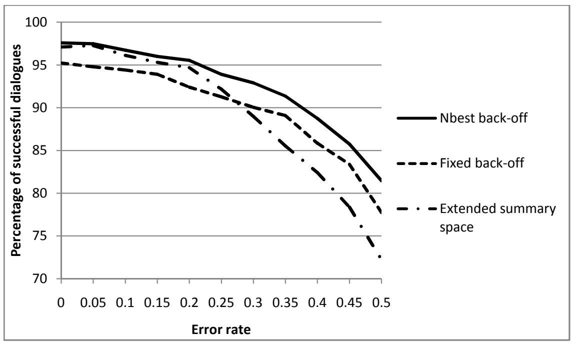

Fig. 5. Comparison of the percentage of the successfully completed simulated dialogues between the fixed back-off strategy, the strategy with extended summary space and the N-best back-off strategy on different error rates

As shown, in Figure 5, the N-best back-off outperforms both alternative strategies across all error rates. Extension of the summary space improved the performance on low error rates. However, due to increased fragmentation of summary space, performance degraded rapidly in noise and increasing the number of training dialogues by a factor of 10 was not able to compensate for this. Hence, it appears that extending summary space with features whose only purpose is to avoid invalid actions degrades the policy overall and results in a system with poor robustness to noise.

It is also interesting to note that the frequency with which the top proposed action is not taken differs in the N-best and fixed back-off strategies. The policy obtained using the N-Best back-off strategy backs off more often (see Figure 6). This suggests that this policy has more liberty when choosing the action, since there is a whole list of back-off actions to try if the top action fails. Not only does the N-best strategy back off more, but the percentage of backing-off increases more dramatically than with the fixed back-off strategy. This may be ascribed to the difficulty of correctly determining which actions are

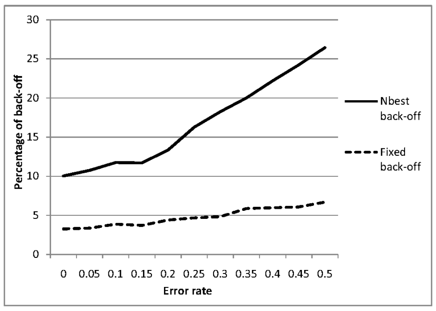

Fig. 6. Percentage of backed-off actions for the Fixed back-off strategy and the N-best back-off strategy for different error rates

valid in noisy states. Therefore, the N-best strategy tries the actions that, if valid, achieve the best reward, whereas the fixed back-off chooses the actions that rarely lead to back-off. Figure 7 shows the percentage of the first-, the second- and the third-best action taken. The results were obtained from 2500 dialogues for each error rate. It shows that on average 82% of the time the top action can be mapped to a master action, but there is a significant tail when the system backs off to the second- and the third-best action.

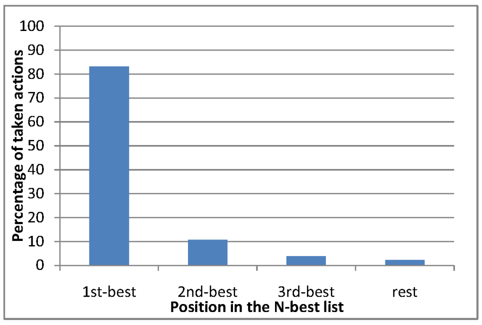

Fig. 7. Percentage of actions on different positions in the N-best list taken in the N-best back-off strategy

VI. Conclusion

Invalid state-action pairs are an intrinsic feature of POMDPs. Available solutions try to back off to a default action that is defined everywhere, but this leads to sub-optimal performance. Alternative solutions to this problem have been empirically examined in the framework of a real-world summary space-based POMDP dialogue system with the goal of overcoming the sub-optimality problem. In a baseline back-off strategy, one summary action is chosen to be the back-off action. In another strategy, the summary space was extended to include information about the subset of actions which are plausible in that particular state. However, this resulted in fragmentation of the space, increased demand for training dialogues and poor robustness to noise.

On the other hand, the proposed strategy of associating an N-best list of actions ranked by Q-value with each grid point worked very well. This approach provided the best performance at all error rates and, importantly, it did not need any more training data than the fixed back-off strategy.
Apart from showing how dialogue performance can be improved by the correct choice of back-off actions, these results also show that fragmentation of the summary space can lead to reduced robustness. Therefore future work will consider both improvements to the design of the master to summary space mapping and to the overall management of the summary space structure.

Acknowledgements

This research was partly funded by the UK EPSRC under grant agreement EP/F013930/1 and by the EU FP7 Programme under grant agreement 216594 (CLASSIC project: www.classic-project.org).

References

[1] S. Young, "Talking to Machines (Statistically Speaking)," in *Int Conf Spoken Language Processing*, Denver, Colorado, 2002.
[2] J. Williams and S. Young, "Scaling POMDPs for Spoken Dialog Management," *IEEE Audio, Speech and Language Processing*, vol. 15, no. 7, pp. 2116–2129, 2007.
[3] L. Li, J. Williams, and S. Balakrishnan, "Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection," in *Interspeech*, Brighton, 2009.
[4] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management," *Computer Speech and Language, In press*, 2009.
[5] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.
[6] R. Brafman, "A Heuristic Variable Grid Solution Method for POMDPs," in *AAAI*, Cambridge, MA, 1997.
[7] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 1998.
[8] S. Young, "ATK: An Application Toolkit for HTK," 2005. [Online]. Available: http://mi.eng.cam.ac.uk/research/dialogue/atk_home
[9] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System," in *HLT/NAACL*, Rochester, NY, 2007.
[10] B. Thomson, K. Yu, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, and S. Young, "Evaluating semantic-level confidence scores with multiple hypotheses," in *Interspeech*, Brisbane, Australia, 2008.
[11] J. Schatzmann, B. Thomson, and S. Young, "Error Simulation for Training Statistical Dialogue Systems," in *ASRU*, Kyoto, Japan, 2007.
[12] M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, K. Yu, and S. Young, "Training and evaluation of the HIS-POMDP dialogue system in noise," in *9th SIGdial Workshop on Discourse and Dialogue*, Columbus, Ohio, 2008.