Compact and Interpretable Dialogue State Representation with Genetic Sparse Distributed Memory

Chapter · December 2017

DOI: 10.1007/978-981-10-2585-3_3

3 authors:

Layla El Asri
Maluuba
18 PUBLICATIONS **68** CITATIONS
SEE PROFILE

Romain Laroche
Microsoft Maluuba
58 PUBLICATIONS **185** CITATIONS
SEE PROFILE

Olivier Pietquin
Google DeepMind
203 PUBLICATIONS **1,330** CITATIONS
SEE PROFILE

Compact and Interpretable Dialogue State Representation with Genetic Sparse Distributed Memory

Layla El Asri[1,2], Romain Laroche[1], and Olivier Pietquin[3,4,5]

[1] Orange Labs, Chtillon, France
[2] UMI 2958 (CNRS-GeorgiaTech), Metz, France
[3] University of Lille, CNRS, Centrale Lille, France
[4] UMR 9189 - CRIStAL, F-59000 Lille, France
[5] Institut Universitaire de France (IUF), France
layla.elasri@gmail.com, romain.laroche@orange.com,
olivier.pietquin@univ-lille1.fr

Abstract. User satisfaction is often considered as the objective that should be achieved by spoken dialogue systems. This is why, the reward function of Spoken Dialogue Systems (SDS) trained by Reinforcement Learning (RL) is often designed to reflect user satisfaction. To do so, the state space representation should be based on features capturing user satisfaction characteristics such as the mean speech recognition confidence score for instance. On the other hand, for deployment in industrial systems, there is a need for state representations that are understandable by system engineers. In this paper, we propose to represent the state space using a Genetic Sparse Distributed Memory. This is a state aggregation method computing state prototypes which are selected so as to lead to the best linear representation of the value function in RL. To do so, previous work on Genetic Sparse Distributed Memory for classification is adapted to the Reinforcement Learning task and a new way of building the prototypes is proposed. The approach is tested on a corpus of dialogues collected with an appointment scheduling system. The results are compared to a grid-based linear parametrisation. It is shown that learning is accelerated and made more memory efficient. It is also shown that the framework is scalable in that it is possible to include many dialogue features in the representation, interpret the resulting policy and identify the most important dialogue features.

Keywords: Spoken dialogue systems · Reinforcement learning · State space representation · Genetic algorithms · Sparse distributed memory

1 Introduction

Reinforcement Learning (RL) [1] is now a state of the art method to learn optimal policies for dialogue systems [2–6]). To do so, a reward function, describing how good a decision made by the system is, has to be designed. It encodes the goal of the system. On the other hand, a commonly used metric to assess dialogue management quality is user satisfaction [7, 8]. Since in RL, the reward function defines the task of the system, it is natural to have the rewards reflect user satisfaction. There has been extensive research on automatically estimating user satisfaction for a given dialogue [9–11]. These studies have shown that many dialogue features (duration, mean speech recognition scores, number of help requests,...) could play an important role in user satisfaction [12, 13]. Because RL relies on a representation of the dialogue state, if the rewards depend on dialogue features then the state space representation should also include these features[14, 15]. Some of these features are continuous and in this case it is not possible to learn by estimating the Q-function for each possible value, a parametrisation is needed.

In addition, if one wants to have his learning algorithm deployed in big industrial systems, there is a need for having dialogue state representations that are interpretable by a human designer. Therefore, if the representation is learnt, it should not transform the original features (no projection, mixing *etc.*). This might lead to intractable dimensions in the representation.

In the RL community, the parametric approximation (and especially linear ones) of the Q-function according to a set of basis functions is the most used framework because theoretical convergence properties can be proved [16]. Generally, the basis functions are assumed to be known in advance. Yet there is a parallel trend that learns basis functions from data. This trend is state aggregation which learns state prototypes by aggregating states into homogeneous clusters. Once the prototypes are learned, the Q-function is estimated as a linear function of their features. Linear representation are interpretable and come with theoretical guarantees [17–19]. However, linear representations are subject to the problem known as the *curse of dimensionality*: learning might become inconveniently slow as the number of dimensions of the state space increases [1]. This problem has already been addressed in dialogue [20, 21] but proposed approaches lead to features that can not be interpreted. Here, we proposed to perform state aggregation based on a Sparse Distributed Memory (SDM, [22]) so as to learn a compact and intepretable state representation.

Contrary to [23–27] where states are aggregated according to their similarity or to [28–31] where states are aggregated according to the similarity of the optimal policy in those states, the proposed approach aggregates states according to the similarity of the Q-function in those states. To do so, a new parametrisation using an SDM is proposed. An SDM manipulates binary data vectors called *prototypes* and assesses the similarity between two prototypes with the Hamming distance. The contributions of this paper are an adaptation of the work on combining SDM and genetic programming for classification [32] to the RL problem and a novel method to incrementally build the set of prototypes in the SDM, according to the states observed by the learning agent. This representation is called Genetic Sparse Distributed Memory for Reinforcement Learning (GSDMRL). Advantages of GSDMRL over previous representations is that it can combine continuous, discrete and symbolic features, that it can be used in both an online and a batch learning setting and that it performs feature selection according to the value function returned by the chosen RL algorithm. GSDMRL is applied to the DINASTI (Dialogues with a Negotiating Appointment Setting Interface) corpus [33]. It is shown that learning is more efficient with GSDMRL and scales well with the number of dialogue features. Then, an analysis of the policy learnt with GSDMRL highlights some important features to take into account for decision making.

2 Background

2.1 The Reinforcement Learning Framework

Dialogue management is modelled as a Markov Decision Process (MDP) which is a five-tuple (S, A, T, R, γ). Elements of this tuple are respectively the state space, the action space, state transition probabilities, the reward function and a discount factor $\gamma \in [0, 1[$. The discounted cumulative reward at time t is the return: $r_t = \sum_{k \geq 0} \gamma^k R_{t+k}$ A deterministic policy π maps each state to one action. The Q-function for π is $Q^\pi(s, a) = E[r_t \mid s_t = s, a_t = a, \pi]$ The dialogue manager seeks an optimal policy π^*, which maximises the expected return starting from any state-action pair. The associated Q-function is: $Q^{\pi^*}(s, a) \in \operatorname{argmax}_\pi Q^\pi(s, a)$

2.2 The Sparse Distributed Memory Model

An SDM [22, 34] is defined on a n-dimensional binary space. It is initialised with a set of addresses. At each address, a vector of counters is stored. Each counter corresponds to a bit in the stored data and all counters are first set to 0. The writing process is illustrated in steps 1 and 2 in Figure 1. Let r be the address register corresponding to the data vector d to be written. The Hamming distance between r and the addresses in the memory is computed. The set of addresses for which the distance is below a given threshold Δ form the *selection set* (step 1).

The data vector d is then written in the vectors of counters linked to the addresses in the selection set (step 2): each counter of each vector is incremented if the corresponding bit in d is 1 and it is decremented otherwise. Reading from the memory with r as input is illustrated in step 3: the selection set is computed as before and the vectors of counters linked to these addresses are summed bitwise. If a counter of the resulting sum is positive, the returned bit is 1, otherwise it is 0.

The SDM was designed to store large binary vectors. It takes advantage of the fact that in large vector spaces, vectors tend to be orthogonal. When a data vector d is written with an address r, all the addresses in r's selection set receive a copy of d. Then, if an address r' close to r is presented to the memory for reading, virtually the same selection set will be computed and copies of d will be numerous in this set. Therefore, d will be output with high probability.

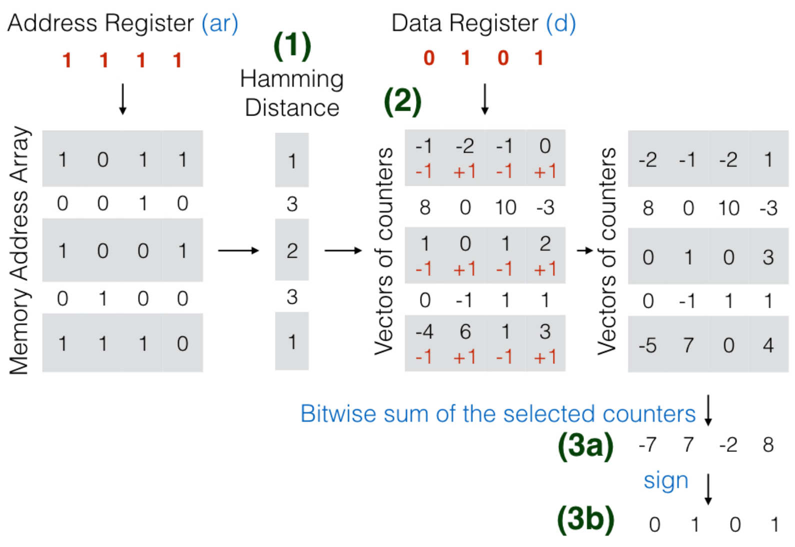

Fig. 1: A schematic display of the reading process in a sparse distributed memory.

Initialisation of the addresses is an issue. If the patterns stored in the memory are large binary vectors, a random initialisation implies to sample a great number of addresses. There exists a method to infer a set of addresses from data [35–38]. Yet they are not relevant for RL applications where only a small region of the state space is visited by most of the acceptable policies. A novel method for initialisation, based on the observations of the learning agent, is proposed in Section 3.

In the RL context, the addresses in the memory represent state prototypes and the stored data are no longer counters representing binary vectors but continuous values representing the Q-function. The memory is randomly initialised [39, 29]. Then, prototypes are re-engineered, for instance, according to visit frequencies. However, this can be problematic as some states might not be visited often but still be crucial like catastrophic states or even goal states [40].

2.3 Genetic Sparse Distributed Memory for Classification

Rogers and then Das and Whitley [41, 32, 42] used the SDM model for binary classification, combining SDM with genetic search. In this context, each address in the memory is linked to a unique counter where the class (0 or 1) is written as explained in Figure 1. Once enough data has been written in the memory, the value of the counter of an address indicates the probability that the class is 1 knowing that the address was selected. It is also possible to know if an address is relevant for class prediction or not. Each address in the memory has an *activation set* which is the set of addresses which would be selected if the address was fed to the memory (the activation set of an address in the memory is equivalent to the selection set of an address register presented to the memory). An address is relevant for classification if the probability that the class is 1 knowing the activation set is significantly different than the probability that the class is 1 given all the addresses in the memory. Based on this, Rogers defined a measure of the *fitness* of a given address. This fitness was then used to rank the addresses in the memory. The address with the lowest ranking was suppressed and replaced by the crossover of the two fittest addresses. The fitness scores were also used to weight the counters: when a new address was presented to the memory for reading, the counters of the addresses in the selection set were weighted according to their fitness so that more importance was given to the fittest addresses.

3 Genetic Sparse Distributed Memory for Reinforcement Learning (GSDMRL)

Let's start with some notations. A prototype p^j is composed of n bits $p^j = (p^j_1, ..., p^j_n)$. The Hamming distance between two prototypes p^1 and p^2 is $d(p^1, p^2) = \sum_{k=1}^{n}(1 - \delta_{p^1_k, p^2_k})$ where δ is the Kronecker symbol. A prototype p^2 belongs to the activation set (resp. selection set) of a prototype p^1 (resp. a state s) if $d(p^1, p^2) \leq \Delta$ (resp. if $d(p^1, s) \leq \Delta$). The notation $|S|$ will be used for the cardinal of a set S.

3.1 Building the Set of Prototypes

GSDMRL builds an SDM M with three layers: each prototype p^j is linked to a vector of counters $c(p^j) = c^j$ and an estimation of the Q-function for each action. The set of prototypes is built incrementally. GSDMRL starts with an empty memory. Each time a new state s_t is observed during learning, if the selection set X_t for this state is empty, the state is added to X_t and to the address set of M and linked to a vector of counters and Q-values initialised to 0:

$$\forall i \in 1, ..., |A| \; M(s_t)_i = \{\forall k \in 1, ..., n \; c(s_t)_k = 0, Q^{\pi}(s_t, a^i) = 0\}. \qquad (1)$$

The state s_t is then written in the counters corresponding to the prototypes in X_t, like in step 2 in Figure 1:

$$\forall p^j \in X_t, k \in 1, ..., n \; c(p^j)_k = c^j_k = c^j_k + (-1)^{(s_t)_k + 1}. \qquad (2)$$

3.2 Q-function Parametrisation

We use a linear representation for the Q-function: $Q_\theta(s_t, a^i) = \sum_{j=1}^{|X_t|} \theta_{j,i}\phi_j(s_t, a^i)$. The ϕ_j are basis functions and the $\theta_{j,i}$ will be updated by the reinforcement learning algorithm. The basis functions are defined as follows:

$$\phi_j(s_t, a^i) = \frac{w^{j,i}}{\sum_{k=1}^{|X_t|} w^{k,i}} \qquad Q_\theta(s_t, a^i) = \sum_{j=1}^{|X_t|} \theta_{j,i} \frac{w^{j,i}}{\sum_{k=1}^{|X_t|} w^{k,i}}. \tag{3}$$

The Q-function is thus estimated as the weighted average of the values of the weights at $(p^j) \in X_t$. This weight is denoted by $w^{j,i}$. Following the same idea as the genetic sparse distributed memory proposed by Rogers, the prototypes in X_t are weighted according to their relevance to estimate the Q-values. The weight $w^{j,i}$ is a function of the fitness $f^{j,i}$ of (p^j, a^i). Rogers [43] showed that the weight which should be given to (p^j, a^i) is $w^{j,i} = \frac{f^{j,i}}{(1-f^{j,i})^2}$.

The fitness score measures the relevance of p^j with respect to the regression of the Q-function. It is computed by looking at the prototypes in the activation set of p^j and checking whether the confidence intervals for the Q-function estimated at action a^i overlap with the confidence interval for $Q^\pi(p^j, a^i)$, noted $CI(Q^\pi(p^j, a^i))$. Confidence intervals are defined as [44]:

$$\begin{aligned} CI(Q^\pi(p^j, a^i)) = CI^{j,i} &= [Q^\pi(p^j, a^i) - \epsilon^{j,i}, Q^\pi(p^j, a^i) + \epsilon^{j,i}] \\ \epsilon^{j,i} &= t_{\alpha/2}^{n^{j,i}-1} \frac{\sigma^{j,i}}{\sqrt{n^{j,i}}}. \end{aligned} \tag{4}$$

In this equation, $n^{j,i}$ is the number of visits to the pair (p^j, a^i), $\sigma^{j,i}$ is the standard deviation of the returns observed after visiting (p^j, a^i) and $t_{\alpha/2}^{n^{j,i}-1}$ is Student's t function with $n^{j,i} - 1$ degrees of freedom. Parameter α sets the confidence with which observed returns will be comprised in the interval. Since the prototypes in M are added only if their distance to all the other prototypes is greater than Δ, the activation set of p^j, $AS(p^j)$ only contains p^j. Nevertheless, two distant prototypes might be activated by the same states and they should predict the same Q-values. The vectors of counters which constitute the second layer of M are used as averages of the states which activate the prototypes. Indeed, every state which activates a prototype is written in the prototype's vector of counters according to Equation 2. So, if two prototypes are linked to similar vectors of counters, it means these prototypes tend to be jointly activated and they should be linked to similar Q-values. The fitness $f^{j,i}$ of (p^j, a^i) is computed by first finding the activation set of p^j's vector of counters c^j. Then, c^j is compared to the other vectors of counters in Hamming distance and the vectors closer than Δ form the activation set $AS(p^j)$. The fitness of (p^j, a^i) is the ratio of the number of confidence intervals overlapping on the size of $AS(p^j)$:

$$f^{j,i} = \frac{|\{p^k \in AS(p^j)\ ,\ (CI^{j,i} \cap CI^{k,i}) \neq \emptyset\}|}{|AS(p^j)|} \tag{5}$$

Besides weighting the prototypes, the fitness scores are also used to re-engineer the prototypes in M.

Algorithm 1 Crossover process

```
Choose random crossover size $s$ in $1, ..., n$
Choose random crossover point $c$ in $1, ..., n - s$
$p^{\text{cross}} = \text{concat}(c^{+,1}_{1,...,d}, c^{+,2}_{d+1,...,d+1+s}, c^{+,1}_{d+s+2,...,n})$
Add $p^{\text{cross}}$ to the set of addresses in $M$
Remove $p^{-}$ from $M$
```

3.3 Re-engineering the Prototypes

We define the set-policy π_s for a prototype p^j as:

$$\pi_s(p^j) \in \underset{a}{\operatorname{argmax}}\, Q^{\pi}(AS(p^j), a^i) = \frac{\sum_{k=1}^{|AS(p^j)|} w^{k,i}\theta_{k,i}}{\sum_{k=1}^{|AS(p^j)|} w^k}$$

The re-engineering rule is only based on the confidence interval for the set-policy of each prototype. The fitness score of (p^j, π_s) measures the relevance of p^j for predicting $Q^{\pi}(s, \pi_s(p^j))$ where s is a state activating p^j. A prototype with a low fitness adds noise to learning. Therefore, the prototype p^- with the lowest fitness score for its set-policy is suppressed and replaced by the result of a random crossover between $c^{+,1}$ and $c^{+,2}$ which are respectively the vectors of counters for the two fittest prototypes $p^{+,1}$ and $p^{+,2}$. The crossover process is detailed in Algorithm 1.

Once the radii of the confidence intervals (ϵ in Equation 4) have reached a low value, the confidence intervals in an activation set will no longer overlap. Therefore, a parameter β is added to the algorithm as a lower bound for the confidence intervals. When the radius of a confidence interval goes under β, it is set to β.

4 Experiments with the NASTIA dialogue system

NASTIA (Negotiating Appointment Setting Interface) is a French-speaking SDS. Its task is an appointment scheduling for landline maintenance. It was tested on 1734 scenario-based dialogues with 385 volunteers who interacted at most 5 times with the system [45]. After each dialogue, the user was asked to fill an evaluation questionnaire and rate the dialogue on a scale of 1 to 10. This corpus is named DINASTI (Dialogues with a Negotiating Appointment Setting Interface) [33].

NASTIA has five dialogue phases where decisions should be taken. We want to use the ratings to infer a state space for each of NASTIA's dialogue phases with GSDMRL. First the system chooses a negotiation strategy. If User Initiative (UI) is chosen, the user is asked: "When would you like to book an appointment ?". System Initiative (SI) asks a sequence of questions in three different dialogue turns, filling successively the day, week and half-day slots. The third option consists of proposing directly a List of Availabilities (LA) to the user, waiting for her/him to interrupt the list as soon as an appropriate appointment has been proposed. A second dialogue phase is the help phase, it is encountered either because the user has requested help or because the system decided to play a help message. Three lengths of messages are available. The third phase is visited after the user has proposed a time slot. NASTIA chooses between three confirmation strategies. The fourth dialogue phase is visited after a rejection from speech recognition or a user time out. In this case, NASTIA may prompt a help message or inform the user that s/he was not understood or heard. The last phase is visited after an appointment setting failure or after the user has expressed some constraints. The system can decide to provide information about its availabilities based on what it has understood of the user's constraints.

Two experiments were conducted using Fitted-Q Iteration [17, 46]. First we compared GSDMRL to a grid-based state representation. For GSDMRL, a dialogue state is the concatenation of 32-bit vectors, each vector being the binary representation of one of the 120 features in the corpus (see the list in [33]). We set $\beta = 0.01$ and $\gamma = 0.99$. The grid was built by performing entropy-based discretisation on the dialogue feature [47, 48]. With this kind of representation, it is not possible to use all the 120 features because the size of the state space, defined as a Cartesian product of the intervals, would become inconveniently high. Therefore, to compare these two approaches, the first experiment only used two features, which are well-known to be important for user satisfaction: the number of dialogue turns and the mean speech recognition confidence score. GSDMRL was thus composed of 64-bit vectors and Δ was set to 15. The second experiment highlights the scalable and interpretable powers of GSDMRL. To do this, the original 120 features of the DINASTI corpus were included in the state space and the threshold Δ was set to 150.

4.1 Comparing GSDMRL to a Grid-Based Representation

The comparison of GSDMRL and the grid is based on the expected cumulative reward per dialogue. Each dialogue starts with deciding the negotiation strategy so they start with the same state s_0. This state s_0 is the first prototype to be written in GSDMRL and it is added to the grid. For both representations, it is ensured that this state would only be visited (or selected in the case of GSDMRL) at the beginning of each dialogue. The reward function gives $\frac{P_i}{\gamma^{t_f-1}}$ at the end of each dialogue D_i where P_i is the performance rating for D_i and t_f is the last dialogue turn (Dialogue turns start at time $t_0 = 0$). For the other intermediate dialogue turns, the reward is equal to 0. These rewards were designed so that the value of the Q-function at the initial state is an estimate of the expected performance for the dialogue:

$$Q^\pi(s_0, a) = E[r_t = \gamma^{tf-1} \times \frac{P_i}{\gamma^{t_f-1}} \mid \mathcal{D}, s_0, a, \pi]$$

$$Q^\pi(s_0, a) = E[P_i \mid \mathcal{D}, s_0, a, \pi] \tag{6}$$

Both representations are compared on $\hat{r}_0$, the expected cumulative reward starting from state s_0 and following the distribution of actions in the corpus,

Table 1: Comparison of GSDMRL and a grid-based representation. All values are averages on 500 runs. The results in bold are statistically significant (p-value under 0.01 for Student's t-test).

Training Corpus	Test Corpus	#states Grid	#states GSDMRL	$\hat{r}_0^{\pi_{Grid}}$	$\hat{r}_0^{\pi_{GSDMRL}}$
531	1200	427.07	**63.9**	7.867	7.878
731	1000	282.23	**66.67**	7.868	**7.885**
831	900	271.99	**66.85**	7.869	**7.891**
931	800	266.84	**62.58**	7.872	**7.900**

that is to say:

$$\hat{r}_0^{\pi} = \sum_a \frac{n(s_0, a)}{\sum_a n(s_0, a)} Q^{\pi}(s_0, a), \tag{7}$$

where $n(s_0, a)$ is the number of visits to (s_0, a). In order to organize a fair comparison, the policies learnt with both representations were evaluated on the same state space representation. Two policies π_{GSDMRL} and π_{Grid} were learnt respectively with the GSDMRL and the grid representation. Then the expected cumulative reward for both policies was computed by projecting the policies on the simple state space representation composed of NASTIA's dialogue phases. Thus, $\hat{r}_0^{\pi}$ was computed as the expected cumulative reward given the policy π_{GSDMRL} or π_{Grid} and given the probability transitions between the dialogue phases. This evaluation tests which mapping from dialogue phases to a higher-dimensional space is the most efficient.

4.2 Scalability and Interpretability of GSDMRL

In the second experiment, a policy is learnt with 120 features. It is possible to easily analyse the policy, based on the prototypes. To achieve this, for each action a, all the prototypes whose optimal action is a are grouped together. Then, the mean feature-wise Hamming distance is computed between the prototypes of one group and the prototypes of the other groups. These distances are normalised by the variances of the values in the corpus. The highest distances are representative of the features which play an important role in choosing action a. For instance, for the strategy negotiation, if the mean speech recognition score of the prototypes choosing the LA action is, in normalised distance, far from the mean recognition scores of the prototypes choosing another action, then it means that the mean speech recognition score is important when it comes to choosing LA. Prototypes within a range of recognition scores will tend to have LA as optimal action. From this, the policy can be analysed by identifying how features influence the system's behaviour.

4.3 Results

The results of experiment 1 are presented in Table 1. First, the number of states varies considerably less with GSDMRL than with the grid. The number of states in GSDMRL is also significantly lower with this value of Δ. Even with a smaller number of states, the average expected cumulative reward with GSDMRL is at least as good as the one with the grid and even a little higher. This shows that GSDMRL enables to build a small set of states upon which a good policy will be learnt in a more efficient way than it is possible to do with a grid-based representation.

In experiment 2, a total number of 899 prototypes were added to the memory. An analysis of these prototypes enables to highlight the most important features in the policy NASTIA learnt with GSDMRL.These features and their values explain the circumstances behind the system's decision.

Concerning the negotiation phase, the LA action is linked to short dialogues (35.23 seconds *vs.* 67.45), with few rejections from speech recognition (1.26 *vs.* 4.68), and where, in average, less than one confirmation occurred. This strategy should thus rather be chosen at the beginning of a dialogue. As expected, the SI strategy should be chosen if the dialogue is problematic. In average, SI should be chosen after 5.43 speech recognition rejections *vs.*. 3.92 for UI and 1.26 for LA. UI seems more fit after a list of availabilities was proposed with LA but none of the propositions suited the user. As for the phase recovering from ASR rejections and user inactivities, the two most important features are the number of user dialogue turns and dialogue duration. It is better to ask the user to repeat after dialogue length has gone above a certain threshold (183.42 seconds *vs.* 52.97). The implicit confirmation strategy is chosen after dialogue duration has gone above 200 seconds and when, in average, there has been less than 1 speech recognition rejection. The explicit strategy should be chosen if more than 2.5 help messages have been played and the number of user turns is high. This means that the dialogue might be problematic and it is better to choose a more conservative strategy. Information about the system's calendar should not be provided to the user if the system has already tried the LA strategy at least once. This reflects the fact that the user is already partially aware of system availabilities and might be annoyed by a reminder. On the other hand, information is given after, in average, 0.47 dialogue acts notifying the user that a slot is not available have been observed. This means that after UI or SI, if the user proposes unrealisable constraints, the system should always provide information about its calendar. Finally, indications of a long dialogue with several unsuccessful negotiation rounds should push to choose the shortest help message.

5 Conclusion

This paper proposed a new model for state aggregation in reinforcement learning-based dialogue management. The model, Genetic Sparse Distributed for Reinforcement Learning (GSDMRL) incrementally builds a set of prototypes. GSDMRL can include features from speech recognition, natural language understanding and dialogue management. Compared to a grid-based representation, GSDMRL was shown to be more efficient both in terms of memory and learning. In addition, GSDMRL is scalable and can handle many dialogue features. Finally, the set of prototypes provides a valuable insight on the system's learning and is easily interpretable.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning. An introduction. MIT Press (1998)
2. Levin, E., Pieraccini, R., Eckert, W.: Learning dialogue strategies within the markov decision process framework. In: Proc. of IEEE ASRU (1997)
3. Singh, S., Kearns, M., Litman, D., Walker, M.: Reinforcement learning for spoken dialogue systems. In: Proc. of NIPS (1999)
4. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. Computer Speech and Language 21, 231–422 (2007)
5. Laroche, R., Putois, G., Bretier, P.: Optimising a handcrafted dialogue system design. In: Proc. of Interspeech (2010)
6. Daubigney, L., Geist, M., Chandramohan, S., Pietquin, O.: A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimisation. IEEE Journal of Selected Topics in Signal Processing 6(8), 891–902 (2012)
7. Dybkjaer, L., Bernsen, N.O., Minker, W.: Evaluation and usability of multimodal spoken language dialogue systems. Speech Communication 43, 33–54 (2004)
8. Lemon, O., Pietquin, O.: Data-Driven Methods for Adaptive Spoken Dialogue Systems. Springer (2012)
9. Walker, M., Hindle, D., Fromer, J., Fabbrizio, G., Mestel, C.: Evaluating competing agent strategies for a voice e-mail agent. In: Proc. of EuroSpeech (1997)
10. Schmitt, A., Schatz, B., Minker, W.: Modeling and predicting quality in spoken human-computer interaction. In: Proc. of SIGDIAL (2011)
11. El Asri, L., Khouzaimi, H., Laroche, R., Pietquin, O.: Ordinal Regression for Interaction Quality Prediction. In: Proc. of ICASSP (to be published) (2014)
12. Larsen, L.B.: Issues in the evaluation of spoken dialogue systems using objective and subjective measures. In: Proc. of IEEE ASRU. pp. 209–214 (2003)
13. Walker, M.A., Langkilde-Geary, I., Hastie, H.W., Wright, J., Gorin, A.: Automatically training a problematic dialogue predictor for a spoken dialogue system. Journal of Artificial Intelligence Research 16, 293–319 (2002)
14. Paek, T., Pieraccini, R.: Automating spoken dialogue management design using machine learning : An industry perspective. Speech Communication 50 (2008)
15. Paek, T., Chickering, D.M.: The markov assumption in spoken dialogue management. In: Proc. of SIGdial Workshop on Discourse and Dialogue. pp. 35–44 (2005)
16. Geist, M., Pietquin, O.: Algorithmic survey of parametric value function approximation. IEEE Trans. Neural Netw. Learning Syst. (2013)
17. Gordon, G.J.: Stable function approximation in dynamic programming. In: Proc. of ICML (1995)
18. Tsitsiklis, J., Van Roy, B.: An analysis of temporal-difference learning with function approximation. Automatic Control, IEEE Transactions on (1997)
19. Gordon, G.J.: Reinforcement learning with function approximation converges to a region. In: Proc. of NIPS (2001)
20. Li, L., Williams, J.D., Balakrishnan, S.: Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In: Proc. of Interspeech (2009)

21. Chandramohan, S., Geist, M., Pietquin, O.: Sparse approximate dynamic programming for dialog management. In: Proc. of SIGDIAL (2010)
22. Kanerva, P.: Associative Neural Memories: Theory and Implementation. Oxford University Press (1993)
23. Broomhead, D., Lowe, D.: Multivariable functional interpolation and adaptive networks. Complex Systems (1988)
24. Albus, J.S.: A theory of cerebellar function. Mathematical Biosciences (1971)
25. Singh, S., Sutton, R.S.: Reinforcement learning with replacing eligibility traces. In: Machine Learning (1996)
26. Forbes, J.R.: Reinforcement Learning for Autonomous Vehicles. Ph.D. thesis, University of California at Berkeley (2002)
27. Mahadevan, S., Maggioni, M., Guestrin, C.: Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. Journal of Machine Learning Research (2006)
28. Bernstein, A., Shimkin, N.: Adaptive aggregation for reinforcement learning with efficient exploration: Deterministic domains. In: Proc. of COLT (2008)
29. Wu, C., Meleis, W.: Adaptive fuzzy function approximation for multi-agent reinforcement learning. In: Proc. of IEEE/WIC/ACM IAT (2009)
30. Baumann, M., Buning, H.K.: State aggregation by growing neural gas for reinforcement learning in continuous state spaces. In: Proc. of ICMLA (2011)
31. Baumann, M., Klerx, T., Büning, H.K.: Improved state aggregation with growing neural gas in multidimensional state spaces. In: Proc. of ERLARS (2012)
32. Rogers, D.: Weather prediction using a genetic memory. Tech. rep., NASA (1990)
33. El Asri, L., Laroche, R., Pietquin, O.: DINASTI: Dialogues with a Negotiating Appointment Setting Interface. In: Proc. of LREC (to be published) (2014)
34. Kanerva, P.: Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. Cognitive computation (2009)
35. Hely, T.A., Willshaw, D.J., Hayes, G.M.: A new approach to kanerva's sparse distributed memory. IEEE Transactions on Neural Networks (1997)
36. Rao, R.P.N., Fuentes, O.: Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. Machine Learning (1998)
37. Anwar, A., Dasgupta, D., Franklin, S.: Using genetic algorithms for sparse distributed memory initializations. In: Proc. of GECCO (1999)
38. Hart, E., Ross, P.: Exploiting the analogy between immunology and sparse distributed memories: A system for clustering non-stationary data. In: Proc. of ICAIS (2002)
39. Kostiadis, K., Hu, H.: KaBaGe-RL: Kanerva Based Generalisation and Reinforcement Learning for Possession Football. In: Proc. of IEEE IROS (2001)
40. Ratitch, B., Precup, D.: Sparse distributed memories for on-line value-based reinforcement learning. In: Proc. of ECML (2004)
41. Rogers, D.: Statistical prediction with kanerva's sparse distributed memory. Tech. rep., NASA (1989)
42. Das, R., Whitley, D.: Genetic sparse distributed memory. In: Proc. of COGANN (1992)
43. Rogers, D.: Using data-tagging to improve the performance of the sparse distributed memory. Tech. rep., NASA (1988)
44. Kaelbling, L.P.: Learning in Embedded Systems. Ph.D. thesis (1990)

45. El Asri, L., Lemonnier, R., Laroche, R., Pietquin, O., Khouzaimi, H.: NASTIA: Negotiating Appointment Setting Interface. In: Proc. of LREC (to be published) (2014)
46. Chandramohan, S., Geist, M., Pietquin, O.: Optimizing Spoken Dialogue Management with Fitted Value Iteration. In: Proc. of Interspeech (2010)
47. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. of UAI. pp. 1022–1027 (1993)
48. Rieser, V., Lemon, O.: Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets. Computational Linguistics 37 (2011)