Optimizing Dialogue Strategy in Large-Scale Spoken Dialogue System: A Learning Automaton Based Approach

G. Kumaravelan and R. Sivakumar

Abstract Application of statistical methodology to model dialogue strategy in spoken dialogue system is a growing research area. Reinforcement learning is a promising technique for creating a dialogue management component that accepts semantic of the current dialogue state and seeks to find the best action given those features. In practice, increase in the number of dialogue states, much use of memory and processing is needed and the use of exhaustive search techniques like dynamic programming leads to sub-optimal solution. Hence, this paper investigates an adaptive policy iterative method using learning automata that cover large state-action space by hierarchical organization of automaton to learn optimal dialogue strategy. The proposed approach has clear advantages over baseline reinforcement learning algorithms in terms of faster learning with good exploitation in its update and scalability to larger problems.

Keywords Human–computer interaction · Reinforcement learning · Learning automata · Spoken dialogue system

G. Kumaravelan (✉)
Department of Computer Science, Pondicherry Univeristy,
Karaikal Campus, Karaikal, India
e-mail: gkumaratcsbdu@gmail.com

R. Sivakumar
Department of Computer Science, AVVM Sri Puspam College,
Thanjavur, India
e-mail: rskumar.avvmspc@gmail.com

Mohan S. and S. S. Kumar (eds.), *Proceedings of the Fourth International Conference on Signal and Image Processing 2012 (ICSIP 2012)*, Lecture Notes in Electrical Engineering 222,
DOI: 10.1007/978-81-322-1000-9_2,

1 Introduction

Human–computer interfaces are now widely studied and have become one of the major interests among the scientific community. In particular, spoken dialogue system (SDS) is a natural language interface designed to make use of spoken language technology to accomplish a task between the user and a computer. Broadly, a SDS has three-modules, as shown in Fig. 1. The essential components are subsystems for input (conveying information from the user to the system), control (deciding how to react) and output (conveying information from the system back to the user) [1]. This paper is concern with the design of Dialogue Management (DM), the central component within the spoken dialogue systems to determine which communicative actions to take (i.e. what to say) given a goal and a particular set of observations about the dialogue history. In other words, they are responsible for controlling the flow of the interaction, sometimes referred to as dialogue strategy or policy in an efficient and natural way. This is a challenging task in most of the spoken dialogue systems wherein the dialogue strategy is handcrafted by a human designer which leads to errors, strenuous and non-portable.

Current research trends indicate attempts to find a way to automate the development of dialogue strategy using machine learning techniques. In practice, Reinforcement Learning (RL) techniques show appealing cognitive capabilities since they try to learn the appropriate set of actions to choose in order to maximize a scalar reward by following a trial and error interaction with an environment [2]. In this context, the dialogue strategy is regarded as a sequence of states with a reward for executing an action which in turn inducing a state transition in the conversational environment. The objective for each dialogue state is to choose such an action that leads to the highest expected long-term reward. For SDSs, these reward signals are associated with task completion and dialogue length. Hence, the system model covers the dynamics of Markov Decision Processes (MDPs) with a set of states S, a set of actions A, a state transition function, and a reward for each selected action. In this framework, a reinforcement learning agent aims at optimally mapping states to actions, i.e. "finding the optimal policy so as to maximize an overall reward" [3, 4].

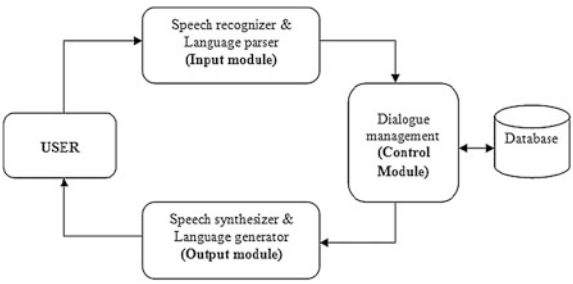

Fig. 1 General architecture of a spoken dialogue system

However, the practical application of RL to optimize dialogue strategy faces a number of technical challenges such as choosing an appropriate reward function, scalability, robustness, and portability [5]. Several approaches to deal with the problem of large state-action spaces have been proposed in recent years. One of the approaches is based on the idea that not all state variables are relevant for learning a dialogue strategy and the state-action space is reduced by carefully selecting a subset of the available state variables by function approximation and hierarchical decomposition. If the relevant variables are chosen, useful dialogue strategies can be learnt. This technique has been applied successfully in several recent studies [6–9]. In addition, eXtended Classifier System (XCS) model has been applied in dialogue strategy optimization to evolve and evaluate a population of rules/and RL algorithm is applied to assign rewards to the rules [10]. However, it mitigates the curse of dimensionality problem by using a more compact representation with regions of state-action, but it finds less optimal solutions compared to tabular value functions.

The limitations of these contributions indicate that the exploration/exploitation trade-off in action selection strategy and curse of dimensionality in modeling the state space has to be solved completely. Most of the reinforcement learning based research attempt value iterative approach (Q-learning, SARSA) to find the optimal dialogue policy in action selection. However, when state-action spaces are small enough to represent in tabular form, Q-learning can be applied to generate a dialogue strategy. On the other hand, increasing the size of the state space for this algorithm has the danger of making the learning problem intractable referred to as "the curse of dimensionality." Another setback of the above baseline reinforcement learning algorithm is that it requires an update of the value function over the entire state space that is purely based on value iteration. In this case, one may get stuck on one iteration for a long time before any improvements in performance are made. Hence, tabular RL algorithms are designed to operate on individual state-action pairs with some practical limit to the size of the state-action table that can be implemented. Even with a relatively small number of state features and system actions, the size of the state-action space can grow very quickly. This constraint poses to be a problem for dialogue strategy developers. Hence, this paper proposes a scalable optimization approach which utilizes the policy iterative hierarchical structure learning automata algorithm of [10] to perform policy optimization over large state-action spaces.

2 Background

Reinforcement learning is a sub-area of Artificial Intelligence (AI) which considers how an autonomous agent acts through trial-and-error interaction with a dynamic unknown environment. Here, the agent refers to an entity that can perceive the state of the environment, and take actions to affect the environment's state. In turn, it receives a numerical signal called reinforcement from the

environment for every action it takes. Its goal is to maximize the total reinforcements it receives over time. In reinforcement learning, an environment is often modeled as MDP, where the history of the environment can be summarized in a sufficient statistic called state to solve sequential decision making problems.

2.1 Dialogue as a MDP

One of the key advantages of statistical optimization methods for dialogue strategy design is that the problem can be formulated as a precise mathematical model which can be trained on real data. The Markov Decision Process (MDP) model serves as a formal representation of human–machine dialogue and provides the basis for formulating strategy learning problems. Every MDP is formally described by a finite state space S, a finite action set A, a set of transition probabilities T and a reward function R. At each time step t the dialogue manager is in a particular state. It executes the discrete action $a_t \in A$, transitions into the next state s_{t+1} according to the transition probability $p(s_{t+1}|s_t,a_t)$ and receives a reward r_{t+1}. In this framework, a DM is a system aiming at optimally mapping states to actions, that is finding best strategy π^* so as to maximize an overall reward R over time, i.e. the policy that selects those actions that yield the highest reward over the course of the dialogue.

2.2 Learning Automata

Learning Automata (LA) are adaptive decision-making devices operating on unknown random environment, and are associated with a finite set of actions and each action has a certain probability (unknown to the automaton) of getting rewarded by the environment of the automaton [11]. The aim is to learn the ways to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system as shown in Fig. 2.

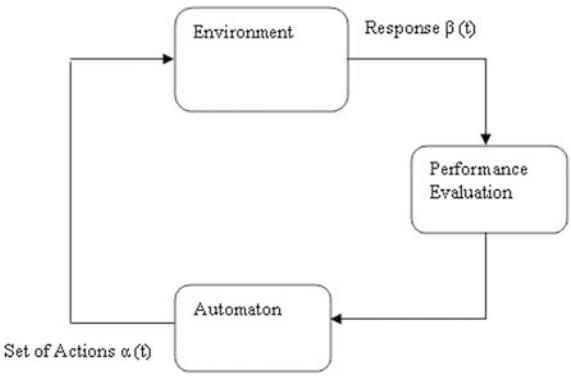

Fig. 2 Learning Automaton and its interaction with the environment

Formally, LA are represented by a triple $< \alpha, \beta, T >$, where α is the action set, β is the environment set and T is the learning algorithm. The learning algorithm is used to modify the action probability vector. The idea behind this update scheme T is that, when an action was successful, the action probability for the chosen action should be increased and all other action probabilities should be decreased appropriately. The general form is given by a following recurrence equations for the case where action $a(i)$ is selected at time step t (thus $a_t = a(i)$), the total number of actions is n and the reward obtained from the environment is r_{t+1}. The action probabilities are updated by the scheme given below:

$$p_{t+1}(i) = p_t(i) + \alpha r_{t+1}(1 - p_t(i)) - \beta(1 - r_{t+1})p_t \quad (1)$$

$$p_{t+1}(j) = p_t(i) - \alpha r_{t+1}p_t(j) + \beta(1 - r_{t+1})\left(\frac{1}{n-1} - p_t(j)\right) \forall j \neq i \quad (2)$$

3 Methodology

A given dialogue task is decomposed into a root subtask and set of dialogue goals. Then, each dialogue goal is decomposed according to the nature of the slot filling strategy. Therefore, each dialogue sub-task in the hierarchy is represented with an MDP, and the hierarchy is denoted by $M = M_j^i$. The proposed method follows frame-based approach in modeling the dialogue structure in the form of frames that have to be filled by the user. Each frame contains slots that guide the user through the dialogue. In this approach, the user is free to take the initiative in the dialogue. The large body of transcribed and annotated conversation forms the basis for task identification, DA recognition, and form filling for task completion. In particular, this representation scheme classifies system actions in terms of their conversational domain, speech act, and task. For example, one possible system action is <about task, request info, dest city>, which corresponds to a system utterance such as What is your destination city?. The high-level structure of the dialogue manager is illustrated in Fig. 3, which includes several interconnected LA organised in hierarchical modules to handle such dialogue sub-tasks.

The dialogue policy is defined by a hierarchy of dialogue sub-tasks $M = M_j^i$ and that each sub-task can apply state abstraction to compress the state space. The indexes i and j only identify a subtask in a unique way in the hierarchy and they do not specify the execution sequence of subtasks because it is learnt by the learning automaton. Algorithm 1, elucidate the procedural form of Hierarchical Learning Automata (HLA) approach for handling knowledge-rich and knowledge-compact state space. HLA approach in practice receives dialogue subtask M_j^i and knowledge base k (based on domain artifacts) used to initialize the automaton at each level in the hierarchy and performs primitive action selections. But for composite actions it invokes recursively with a child subtask. When the subtask is completed

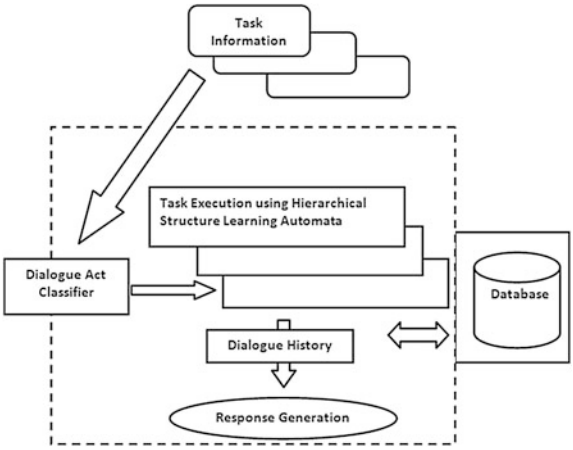

Fig. 3 Task hierarchy mechanism of the proposed method

with α time step it returns an average reward $R(a_t^l(h))$ and continues its execution until finding a leaf state for the root subtask M_0^0. The algorithm is iterated until convergence occurs in optimal context-independent policies.

Algorithm 1 HLA-learning algorithm with knowledge compact states

Procedure HLA (Knowledge base k, Sub task M_j^i) return *averageReward* $R\langle a_t^l(h)\rangle$
Initialisation
  For all the learning automata: Initialize action probabilities:
$$\forall\, a \in A : p(a) = \frac{1}{|A|};$$
  The estimates of all the actions $\forall\, a \in A : R\langle a_t^l(h)\rangle = 0;$
  The estimates for the learning automata at level n
$$\forall\, a \in A : L\langle a_t^l(h)\rangle = 0;$$
for each trial do
 Activate the top LA of the hierarchies
  for each level "l" in the hierarchy "h" do
   The active LA selects action $a_t^l(h)$ probabilistically
   Perform joint-action selection $a = [a_t^l(1), \ldots, a_t^l(h)]$
   Observe immediate reward r_t
   Compute $R_t = r_t + \gamma\, L\langle a_t^l(h)\rangle)$
  end for
  for each level "l" in hierarchy "h "do
    Compute combined reward $R\langle a_t^l(h)\rangle = R\langle a_t^l(h)\rangle + \rho[R_t - R\langle a_t^l(h)\rangle]$
    Update action probability p using $R\langle a_t^l(h)\rangle$ as the reward for the L_{RI} scheme
    Propagates r_{t+1} to the parent
  end for
end for
end procedure HLA

4 Experiments and Results

In this section, a slot-filling dialogue system based on the travel domain to verify the effectiveness of a dialogue strategy in a simulated learning environment for the proposed model is presented. The experimental design is based on the open agent architecture (OAA).The state space illustration of the chosen application has six slots representing all the currently available information regarding internal and external processes controlled by the dialogue system i.e., the knowledge of the concerned domain. For information seeking tasks, a common approach is to specify state variables on the number of slots that need to be filled and grounded. For example, a particular slot has not yet been stated (unknown), stated but not grounded (known), or grounded (confirmed). In addition, A record of the number of times each slot was asked for or confirmed by the system was necessary to indicate that a dialogue was not progressing sufficiently, perhaps due to the persistent (simulated) misrecognition of a particular slot. With these factors, the list of state variables and their possible values are given in Table 1.

The preferred system and user DAs which comprise action space for modeling dialogue strategies are summarized in Table 2. The system dialogue acts allow the system to request the user for the slot values and to restart or end the dialogue. Finally, the system can then present the results of a user's database query. The user dialogue acts allow the user to provide slot information, allow the user to terminate the dialogue, ask for help, and start the dialogue from the beginning.

Hence with 2,916,000,000 unique states and 10 system actions, the selected size of the state-action space explored by the experimental setup is 2.9×109. For tractable hierarchical learning, the state-action representation is decomposed into 7 sub-tasks. Figure 4 illustrates the sub-task hierarchy of the aforementioned application domain and Table 3 describes the state variables actions per subtask. It has been clearly emphasized that the each sub-tasks undergoes a state abstraction procedure by ignoring irrelevant variables.

Table 1 State variables representation in travel planning SDS

State variable	Possible values	State space size
dep_city_confidence	unknown, known, confirmed	3
dest_city_confidence	unknown, known, confirmed	9
date_confidence	unknown, known, confirmed	18
time_confidence	unknown, known, confirmed	324
brand_confidence	unknown, known, confirmed	972
location_confidence	unknown, known, confirmed	2,916
dep_city_times_asked	1..10	29,160
dest_city_times_asked	1..10	291,600
date_times_asked	1..10	2,916,000
time_times_asked	1..10	29,160,000
brand_times_asked	1..10	291,600,000
location_times_asked	1..10	2,916,000,000

Table 2 Common system and user dialogue acts used in travel planning SDS

System acts	User acts
Greeting	Command(bye)
Goodbye	Command(request_help)
Restart	Command(restart)
Request_Info(dep_city)	Provide_ Info(dep_city)*
Request_Info(dest_city)	Provide_ Info(dest_city)*
Request_Info(date)	Provide_ Info(date)*
Request_Info(time)	Provide_ Info(time)*
Request_Info(brand)	Provide_ Info(brand)*
Request_Info(location)	Provide_ Info(location)*
Database Results	Answer(yes); Answer(no)

* Multiple slot values can be provided in a single utterance

Table 3 State variables and actions of the subtask hierarchy in the travel planning system

Subtask	State variables	Actions
M_0^0	GIF, SAL, F0,H0	M_1^1, M_1^2, greeting(),
M_1^1	MAN, OPT	M_2^1, M_2^2, greeting(), restart(), database_results()
M_1^2	MAN, OPT	M_2^3, M_2^4, greeting(), restart(), database_results()

M_2^1	C00,C01,C02,C03	request_Info(dep_city) + imp_confirmation + exp_confirmation, provide_ Info(dep_city);
		request_Info(dest_city) +imp_confirmation + exp_confirmation, provide_ Info(dest_city);
		request_Info(date) + imp_confirmation + exp_confirmation, provide_ Info(date);
		request_Info(time) + imp_confirmation + exp_confirmation, provide_ Info(time);

M_2^2	C04	command(request_help); answer(yes);answer(no);
M_2^3	C05,C06	request_Info(brand) + imp_confirmation + exp_confirmation, provide_ Info(brand); request_Info(location) + mp_confirmation + exp_confirmation, provide_ Info(location)
M_2^4	C07	command(request_help)); answer(yes);answer(no);

The values of state variables includes Goal In Focus (GIF) = {0 = flight details, 1 = hotel details}, Salutation (SAL) = {0 = null, 1 = greeting, 2 = goodbye}, {MAN, OPT, F0, H0} ← {0 = unfilled sub-task, 1 = filled sub-task, 2 = confirmed sub-task}, slot in focus (Cij) ← {0 = unknown, 1 = known, 2 = confirmed}.

To assert the impact of the proposed method, different sizes of datasets (dialogue corpora) which represent the problem space are required. However, it is not possible to collect large amounts of data. Therefore, a user simulation technique [12] has been used to generate different datasets. The task of the user simulator is to provide examples of how a real user would behave while interacting with the system. It provides user actions at a dialogue act level and uses two main

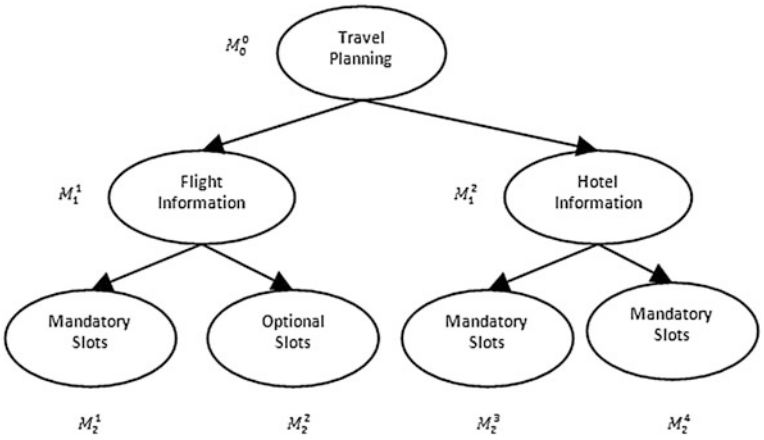

Fig. 4 The hierarchy of LA induced for the dialogue sub-tasks in the travel planning system

components user goal and user agenda. At the start of each dialogue, the goal describes the full set of constraints that the user requires to satisfy such as departure city, destination city, date and time. The agenda stores an ordered list of dialogue acts that the user is planning to use in stack like structure in order to complete its task. At the end of every dialogue episode the system receives a -1 penalty for every action it takes, a final reward of +20 in case of successful dialogue when all the necessary information has been obtained. In the case where no flight detail matches the attribute-slot values, the dialogue is deemed successful and a suitable alternative is offered. Since a characteristic dialogue will require about six or seven turns to complete, this implies that the achievable average reward has an upper bound of 14.

In each experiment, dialogue strategies were allowed to evolve over a fixed number of dialogues. The goal of the system is to acquire the values for the slots (attributes) with concern to each subtask (flight and hotel booking). In this case, the state-action space representation follows 2 hierarchies of 4 levels, with 10 actions per automaton. This gives a total of $(2^4)^{10} = 1.09 \times 10^{12}$ solution paths. Each action in the respective hierarchy is selected based on $L_{R\text{-}I}$ scheme as stated in Eqs. (1) and (2). The average reward is normalized to the interval [0–1] which determines the probability of action selection in each automaton in the hierarchy. The accuracy of the proposed approach is tested in function of the learning rate as shown in Fig. 5.

Figure 6 shows the reward value plotted against the number of iterations averaged over 10 training runs of 2000 episodes (or dialogues) by various approaches. With one time step of history, the baseline RL and HLA methods both appear to converge after about 500 dialogues. However the baseline RL method show clear signs of instability whereas HLA does not. This is due to the fact that the non-mobile LA defined in the levels of hierarchy does not move around the

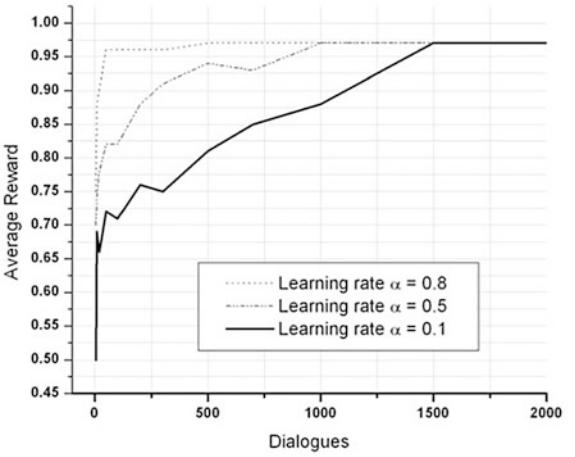

Fig. 5 The average reward using HLA method for various learning rates

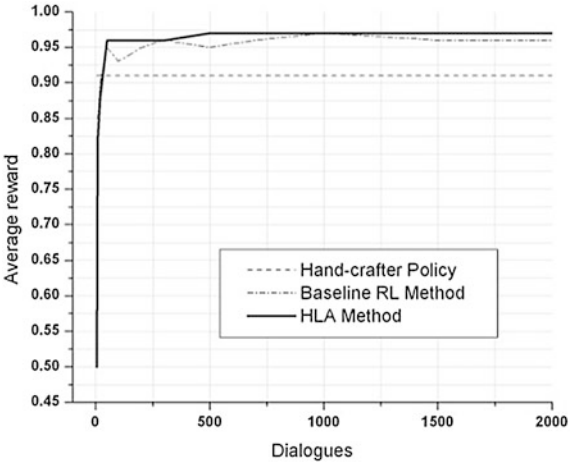

Fig. 6 Dialogue strategy learning by different approaches

state space but stay in their own state to get active and learn to take actions only in their own state of the MDP. However, in the case of baseline hierarchical learning approach using Q-learning algorithm, each Q-learner learns individually the associated Q-values with their own action rather than joint actions.

As a consequence the tabular approaches are completely independent and have no knowledge of the other Q-learner acting in the environment and influencing their reward. However, when the system has a unique limiting distribution over the state space, both independent LA and independent Q-learners are able to find a strategy for optimal action. Although the latter needs good exploration settings, it may take a very long time before convergence. In addition, the proposed approach appears quite stable and converges consistently with improved exploitation in its updates compared to the baseline tabular approach. This is one of the advantages

of the HLA method when developing a new system or adapting it to changes in the tendency in the data.

5 Conclusion

This paper proposes learning dialogue strategies using policy iterative hierarchical structure learning automata that cover large state-action spaces under the formalism of Markov Decision Process in the simulated environment. In the analysis, it is found that the developed methodology is capable of learning optimal dialogue strategy in the context of large state-action spaces, application to travel planning domain. The hierarchical representation allowed the system to automatically generate specialised action that takes into account the current situation of the dialogue depending on the use of expected cumulative reward. Faster learning has become possible when the state space is being divided among multiple interconnected automata that can work independently to provide better convergence and knowledge transfer which provides the solution to learn about previous problems that can be re-used in new problems. For future work, we believe that our approach can also be extended in the direct context of Partially Observable MDPs which account for approximately linear running time of learning algorithms when the state space is continuous and directly incorporates uncertainty imposed to a noisy channel.

References

1. McTear M (2004) Spoken dialog technology: toward the conversational user interface. Springer, New York
2. Sutton RS, Barto AG (1998) Reinforcement learning an introduction. MIT Press, Cambridge
3. Levin E, Pieraccini R, Eckert R (2000) A stochastic model of human-machine interaction for learning dialog strategies. IEEE Trans Speech Audio Process 8(1):11–23
4. Singh S, Litman D, Walker M (2002) Optimizing dialogue management with reinforcement leaning: experiments with the NJFun system. J Artif Intell 16:105–133
5. Paek T, Pieraccini R (2008) Automating spoken dialogue management design using machine learning: an industry perspective. Speech Commun 50(8-9):716–729
6. Pietquin O, Dutoit T (2006) A probabilistic framework for dialog simulation and optimal strategy learning. IEEE Trans Audio Speech Lang Process 14(2):589–599
7. Henderson J, Lemon O, Georgila K (2008) Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. Comput Linguist 34(4):487–512
8. Cuayáhuitl H, Renals S, Lemon O, Shimodaira H (2010) Evaluation of a hierarchical reinforcement learning spoken dialogue system. Comput Speech Lang 24(2):395–429
9. Toney D, Moore J, Lemon O (2006) Evolving optimal inspectable strategies for spoken dialogue systems. In: Proceedings of HLT, pp 173–176
10. Baba N, Mogami Y (2006) A relative reward-strength algorithm for the hierarchical structure learning automata operating in the general nonstationary multiteacher environment. IEEE Trans Syst Man Cybern- Part B Cybern 36:781–794

11. Thathachar MAL, Sastry PS (2004) Networks of learning automata: techniques for online stochastic optimization. Kluwer, Norwell
12. Schatzmann J, Weilhammer K, Stuttle MM, Young S (2006) A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. The Knowl Eng Rev 21(02):97–126

http://www.springer.com/978-81-322-0999-7

Proceedings of the Fourth International Conference on Signal and Image Processing 2012 (ICSIP 2012)
Volume 2
S, M.; S Kumar, S. (Eds.)
2013, XVI, 631 p., Hardcover
ISBN: 978-81-322-0999-7