Simulation of Dialogue Management for Learning Dialogue Strategy Using Learning Automata

G.Kumaravelan
Research Scholar, Department of Computer Science
Bharathidasan University
Trichirappalli, India
gkumaratcsbdu@gmail.com

R. Sivakumar
Reader and Head, Department of Computer Science
AVVM Sri Pushpam College
Poondi, India
rskumar.avvmspc@gmail.com

***Abstract*: Modeling the behavior of the dialogue management in the design of a spoken dialogue system using statistical methodologies is currently a growing research area. This paper presents a work on developing an adaptive learning approach to optimize dialogue strategy. The problem of dialogue management can be formalized as a sequential decision making under uncertainty whose underlying probabilistic structure has a Markov Chain. A variety of data driven algorithms for finding the optimal dialogue strategy is available within Markov Decision Process which is based on reinforcement learning. However the local reward function is typically set as static and there exist a dilemma in engaging the type of exploration versus exploitation. Hence we present an online policy learning algorithm using learning automata for optimizing dialogue strategy which improves the naturalness of human-computer interaction that combines fast and accurate convergence with low computational complexity.**

Keywords- Adaptive learning; Dialogue management; Learning automata; Multi-Agent Reinforcement learning; Spoken dialogue system;

I. Introduction

Spoken Dialogue System (SDS) provides a natural language interface for the user to communicate with a computer to obtain information or engage in a goal oriented transaction. Such systems have covered a wide range of domains. Spoken dialogue technology has been an active research area for the past two decades. A number of academic and commercial systems have been developed, which demonstrates the potential of this technology [1], [2], [3]. In order to achieve the natural language interface there are a number of functions that must be carried out within a dialogue system, such as natural language processing, domain processing and Dialogue Management (DM). The design of an effective dialogue management component is the central aspect of dialogue system engineering [4]. Moreover, the dialogue management is the module that defines the interaction with the user and it is the window through which the user perceives the system's capabilities. This means dialogue strategies designed by human are prone to errors, labour-intensive and non portable. These facts motivate the topic of automatic dialogue strategy learning an attractive alternative.

Broadly speaking, three different approaches have been used in DM. First, the finite state-based approach represents the dialogue structure in the form of a network, where every node represents a question and the transitions between nodes represent all the possible dialogue [5] whose primary nature is system initiative. Secondly, the frame-based approach represents the dialogue structure in the form of frames that have to be filled by the user, where 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 [6]. Finally, agent-based approach incorporates a wide variety of approaches that use techniques from Artificial Intelligence to produce more intelligent systems [7] which incorporates mixed initiative interaction in which the user can change the current context and the dialogue history. These approaches typically involve authoring and complicating hand-crafted rules that require considerable deployment of time and cost. However they do not address the issue of how to develop the best possible dialogue strategy.

For these reasons, during the last decade many research groups have been attempting to find a way to automate the design of DM for learning dialogue strategy using machine learning technique such as Reinforcement Learning (RL) [8]. Reinforcement learning addresses the problem faced by an agent that learns behavior using trial and error interaction within a dynamic environment so as to maximize a scalar reward signal. The aforementioned factor influences dialogue management to be modeled as a Markov Decision Process (MDP) in which the state of the dialogue depends only on the previous state & its action and reinforcement learning is applied to find the optimal dialogue policy.

A number of reinforcement learning methods have been proposed in recent years to automate the design of dialogue strategy [9], [10], [11], [12], [13]. However, much research effort is being proposed for improving these techniques and in applying these techniques in various application domains. For practical deployment the local reward is typically set as static and it is perhaps the most hand-crafted aspect for dialogue management. On the other hand, increasing the size of the state space for RL has the danger of making the learning problem intractable (referred as "the curse of dimensionality"). Recent investigations employ function approximation [14], dialogue simulation [15] and prior knowledge [16] in order to find solutions on reduced state spaces.

In addition, mostly all the popular RL algorithms (e.g., Q-Learning) used in learning dialogue strategies are model free in nature and require explicit tabular storage of agent Q-functions and possibly of their policies [17]. Hence when the state and action space contain a large number of elements, tabular storage of the Q-function becomes impractical and there exists a dilemma with exploration versus exploitation in choosing an optimal action. Consequently, online model-based policy learning algorithms hold great promise in this regard to overcome the above mentioned factors in learning the dialogue strategy within RL framework [18]. Alternatively, Learning Automata (LA) are valuable tools for current Multi Agent Reinforcement Learning (MARL) research [19] and their learning scheme (purely model-based) updates strictly on the basis of the response of the environment and not on the basis of any knowledge regarding other automata, i.e. neither their strategies, nor their feedback. As such LA agents are simple. Moreover, LA can be treated analytically. Convergence proofs do exist for a variety of settings ranging from a single automaton model acting in a random environment to a distributed automata model interacting in a complex environment. Therefore, in this paper we propose a design for dialogue strategy using a team of learning automaton in the context of decentralized control of MDP.

This paper is organized as follows: In the next section, a brief description of the Learning Automata employed to design the dialogue management for learning dialogue strategies is given. Section 3 describes our experimental design, where we provide details of the learning methodology. In section 4, we describe our experimental results. Finally, in section 5 we provide our conclusions and comment on future directions of this work.

II. Learning Automata

A learning automata is a precursor of a policy iteration type of reinforcement learning algorithm and has some roots in psychology and operations research [20]. Learning automata are adaptive decision-making devices operating on unknown random environments. Fig. 1 shows a typical learning system layout. A learning automata has 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. The objective is to learn for choosing the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction with the system. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in the selection of the optimal action. Learning Automata can be classified into two main families: Fixed Structure Learning Automata (FSLA) and Variable Structure Learning Automata (VSLA).

The VSLA guarantees robust behavior in the absence of complete knowledge of the solution space and has rigorous mathematical properties that can be used to create efficient algorithms. It can be viewed as a stochastic finite state machine with a set of actions and associated probabilities that help to learn the nature of an unknown environment. For every random action that is chosen, the environment that needs to be learned provides a response that is stochastically related to the chosen action. The iterative process of choosing random action and recording the response is continued until the solution space is fully explored.

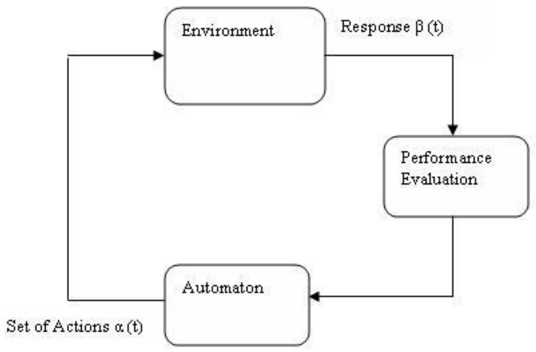

Figure 1. Learning automata system

In a Variable-Structure Learning Automata (VSLA), the probabilities of the various actions are updated on the basis of information the environment provides. A VSLA is a quadruple $< \alpha, \beta, p, T(\alpha, \beta, p) >$, where α, β, and p constitute an action set with r actions, an environment response set, and the probability set p containing r probabilities, each being the probability of performing every action in the current internal automata state, respectively. The function of T is the reinforcement scheme which modifies the action probability vector p with respect to the performed action and received response. If the response of the environment takes binary values, learning automata model is *P-model* and if it takes finite output set with more than two elements that take values in the interval [0,1], such a model is referred to as *Q-model*, and when the output of the environment is a continuous variable in the interval [0,1], it is referred to as *S-model*. Assuming $\beta \in [0,1]$, a general linear schema for updating action probabilities can be represented as follows:

$$\mathrm{P}i(t+1) = \mathrm{P}i(t) + a(1-\beta(t))(1-(\mathrm{P}i(t))) - b\beta(t)\mathrm{P}i(t) \quad (1)$$

if action α was taken at time step t

$$\mathrm{P}j(t+1) = \mathrm{P}j(t) - a\beta(t)\mathrm{P}j(t) + b(1-\beta(t))\lceil r-1 \rceil^{-1} - \mathrm{P}j(t) \quad (2)$$
$$\forall j \neq i$$

The constants a and b in the interval [0,1] are the reward and penalty parameters respectively and r the number of actions of the action set of the automata. When a=b the algorithm is referred to as linear reward–penalty ($L_{R\text{–}P}$), when b=0 it is referred to as linear reward–inaction ($L_{R\text{–}I}$) and when b is small compared to a it is called linear reward–є-penalty ($L_{R\text{–}\epsilon P}$).

III. METHODOLOGY

The current state-of-the art SDS are often mixed-initiative slot-filling system. This means that both the user and system may take the initiative to provide information or ask follow up questions in a dialogue session to jointly complete certain task. These kinds of SDSs are useful in domains where certain bits of information need to be elicited from the user, resulting in a set of slots to be filled, which are usually used to make a database query or update. However, automatically designing an efficient dialogue strategy to assist the user to quickly fill in these slots is never a trivial problem. The automatic learning of the dialogue strategy could be further described as an intelligent control problem that involves an agent learning to improve the performance of SDSs by interaction with underlying environment.

This section presents the general model of the DM strategy which has to be optimized and the DM will be the learning agent. At each turn the DM has to choose an action according to its interaction strategy with the environment so as to complete the task that it has been designed for. These actions can be greetings, request to constraint and confirm the value of attributes, perform retrieval operations in the database and to close the dialogue session. The response from the environment leads to the updation of the reward function and internal state of the learning agent which contains enough information about the history of the dialogue.

Figure 2 shows the Learning Automata model, Optimizes the learning strategy through interaction with the environment. The learning agent starts with a random policy in which a probability is associated with each state-action pair (i,a). This is the probability of selecting action *a* in state '*i*' in the simulator. Initially this probability is same for every action. The automaton through its interaction, updates probabilities until the optimal action(s) has the highest probability. Over time, with trial and error, the system learns the optimal action in each state. The response is actually the immediate reward earned by an action.

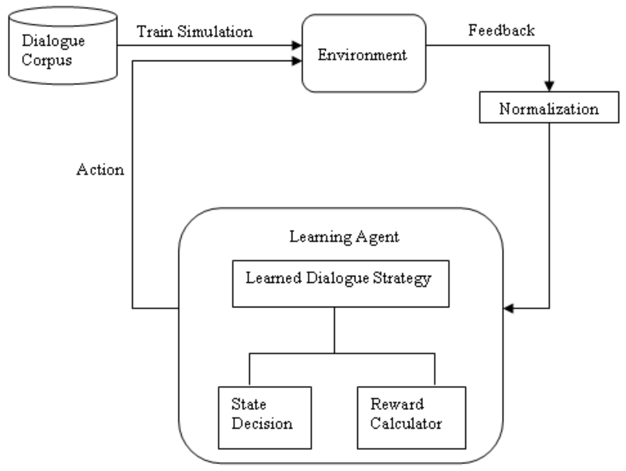

Figure 2. Mechanism of learning dialogue strategy using Learning Automata

A. Convergence Property

Let A_i denote the set of actions available at state '*i*'. Hence the union of A_i over '*i*' gives the action space. With every state-action pair, we associate a probability P*(i,a)* of taking action '*a*' in state '*i*'. Obviously $\Sigma_{a\epsilon A_i}$ P*(i,a)* = 1. In the beginning of the learning process, the policy is random and each action is equally likely. Hence P*(i,a)* = 1 / r_i where r_i = |*A* | is the number of possible actions in state '*i*'. If the performance of an action is good the probability of that action is increased and if the performance is poor, the probability is reduced. This is called updating of the state-action probabilities. However the updating scheme must always ensure that the sum of the probabilities of all the actions in a given state is 1.

B. Feedback Mechanism

The action chosen by the automaton is the input to the environment which responds with stochastic response or reinforcement. Higher values of the reinforcement signal are assumed more desirable. Whenever a state '*i*' in the system has been revisited the average reward is calculated by the total reward earned since the last visit to that state divided by the number of state transitions since the last visit to that state. This reward is called response of the system to the action selected in the last visit to the state '*i*'. Thus if '*i*' is under consideration, the response *S(i)* can be calculated as:

$$S(i) = R(i) / N(i) \tag{3}$$

where *R(i)* denotes the total reward earned since the last visit to state '*i*' and *N(i)* is the number of state transitions that have been taken place since the last visit to '*i*'. The response is then normalized to ensure that it lies between 0 and 1. The normalized response is called feedback represented by *β*. The normalization is performed via the following mechanism:

$$\beta(i) = S(i) - S_{min} \;/\; S_{max} - S_{min} \tag{4}$$

where S_{min} is the minimum response possible in the system and S_{max} is the maximum response possible in the system. The feedback is to update P*(i,a)* that is, the probability of selecting action '*a*' in state '*i*'. The conversion of the response to feedback itself is a research topic, more on this can be found in [20]. All schemes are designed to punish the bad actions and reward the good actions.

C. Reward-Inaction Learning Scheme

The scheme for updating the probabilities using feedback is a topic of research and several schemes have been suggested in literature. The scheme that gave the best result is known as the Reward-Inaction Scheme [20]. The goal of the automaton is to identify the optimal action. This is to be achieved through a learning algorithm that updates the action probability at each instant, using the most recent interaction with the environment.

When the system visits a new state '*i*', then the probabilities of the actions allowed in state '*i*' are first updated. For this *β(i)* is calculated and then according to the Reward-inaction scheme, the probabilities are updated via the rule given below:

$$P(i,a_i) \leftarrow P\,(i,a_i) + \eta\ \beta(i)\ I\,[\,D(i) = a_i\,] - \eta\ \beta(i)\ P(i,a_i) \tag{5}$$

where D(i) denote the action taken in state '*i' ϵ S* in its last visit , *η* denotes the learning rate, a_i the action whose probability of getting selected in state *'i'* is to be updated and *I [.]* equals 1 if the condition inside the brackets is satisfied and *I [.]* = 0 otherwise.

Figure 3 presents a very high level description of the proposed approach. In this scheme a good action automatically has a high value for *β* and therefore the scheme increases the probability of that action. Similarly an action that results in a poor reward has a low value for *β* and therefore the probabilities are not changed significantly. Thus, the objective of the learning scheme is to maximize the expected value of the reinforcement received from the environment. Hence, an equivalent way to characterizing the goal of an automaton can be defined as:

$$\text{Maximize } M^k(i) = E\,[\,\beta^k(i) \mid P^k\,] \qquad (6)$$

where $\beta^k(i)$ be the feedback received by the automaton in the i^{th} state at the k^{th} iteration of the algorithm and $M^k(i)$ to be its expected value. The feedback is associated with given values for the vector $P^k = (P^k(i,1)Pp^k(i,2),...Pp^k(i,r_i)$, where $P^k(i,a)$ denote the probability of selecting action *a* in state *i* in the k^{th} iteration of the algorithm. In this case the learning rate is fixed and need not be changed with iterations and MAX_STEPS specifies the termination of the algorithm until the most probable action in each state is achieved.

```
(1) Let $S$ denote the set of states and $A_j$ be the finite number of actions
(2) Initialize action probabilities $p(i,a) = 1/ri$ for all $i \varepsilon S$ and $a \varepsilon Aj$, cumulative reward $Cr(i)$, total reward earned $Tr$ for every $i \varepsilon S$, iteration count $m$ to zero and $S_{min}$ and $S_{max}$ according to the knowledge of the conversational system.
(3) While ($m$ < MAXSTEPS) do
(a) If state $i$ has been visited for the first time goto (c) else
        Set $R(i) = Tr - Cr(i),\ S(i) = R(i) / N(i)$ and
        $\beta(i) = S(i) - S_{min} / S_{max} - S_{min}$
(b) Update $p(i,a)$ using Reward-Inaction Learning scheme by equation (3)
(c) With probability $p(i,a)$ choose the action $a$ in $A_j$ and
        Set $Cr(i) = Tr$
(d) Observe immediate reward $r = g(i, a, j)$ from selecting action $a$ in state $i$ and simulate the chosen action $a$.
    (e) Compute $Tr = Tr + r$
    (f) Set i = j and m = m+1
```

Figure 3. The High level algorithm of the proposed approach

IV. EXPERIMENTAL RESULTS

A. Design

In this section a simple slot-filling dialogue system is presented based on railway reservation domain to verify the effectiveness of dialogue strategy learning by the proposed stochastic learning automaton model. The goal of the system is to acquire the values for four slots: departure city, destination city, date for the outward journey and date for the return journey. Each slot has three possible values (unknown, known, or confirmed), which gives a total of $3^4 = 81$ possible states. For each state three possible actions are associated (implicit, explicit, or repetitive question). The number of combinations of the pair (state and action) thus becomes $81^3 = 531441$. This corresponds to the maximal number that system could explore in order to identify an optimal strategy. This number can be reduced by eliminating non relevant actions in certain states, such as for example, at the start of the dialogue, there cannot be a repetitive or implicit question. However the aim is to exploit the learning capability of the agent in an uncertain environment and without prior knowledge.

The chosen system and user dialogue acts are summarized in Table 1. The system dialogue acts allow the system to request the user for the slot values or to confirm these values, either explicitly or implicitly and to restart or end the dialogue. Finally, the system can present the results of a user's database query. The user dialogue acts allow the user to provide slot information and to terminate the dialogue.

TABLE I. SYSTEM AND USER DIALOGUE ACTS

System Acts	***User Acts***
Greeting	Command (bye)
Request_Info(dep_value)	Provide_ Info(dep_value)
Request_Info(dest_value)	Provide_ Info(dest_value)
Request_Info(out_value)	Provide_ Info(out_value)
Request_Info(ret_value)	Provide_ Info(ret_value)
Database Results	

B. Results

After several thousand simulated dialogue sessions, the behavior of the Reward-Inaction Learning Scheme for the proposed approach with different settings of the parameter (learning rate) η, i.e. η = 0.1, η=0.5 and η=0.8 is plotted in Figure 4 shows that the bigger we set η, higher the probability of convergence to an optimal action. Table 2 shows the percentage of runs over which action converges to 0 as a function of η. From Table 2, it is clear that if one were to tolerate a 5% error, a suitable value for the learning rate would be 0.3, which would substantially increase the speed of convergence.

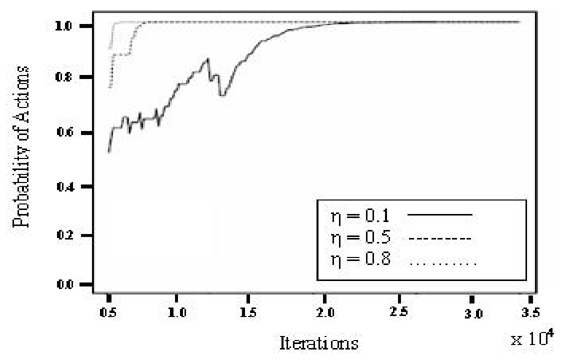

Figure 4. Convergence of the Reward-Inaction Learning scheme with learning parameter η

TABLE II. EFFICIENCY OF LEARNING PARAMETER

η	**% of wrong convergence**
0.9	22
0.5	12
0.3	5
0.1	0
0.01	0
0.001	0

C. *Evaluation*

In order to find the most significant parameters (i.e., *predictors*) of the proposed dialogue management performance, the PARADISE framework [21] was used. It maintains that the system's primary objective is to maximize user satisfaction, and it derives a combined performance metric for a dialogue system as a weighted linear combination of *task-success measures* and *dialogue costs*. To evaluate the dialogue strategies 60 undergraduate students (32 female, 28 male) with an average age of 21 have tested our system. It is important to mention that since our testers had no previous experience with a dialogue system, our experiments were performed with novice users.

To evaluate user satisfaction, users were given the user-satisfaction survey used within PARADISE framework, which asks to specify the degree to which one agrees with several questions about the behavior or the performance of the system (TTS Performance, ASR Performance, Task Ease, Interaction Pace, User Expertise, System Response, Expected Behavior, and Future Use). The answers to the questions were based on a five-class ranking scale from 1, indicating strong disagreement, to 5, indicating strong agreement. For our experiment, the mean *User Satisfaction* value was 33.32 as shown in Table 3. In this table we can observe some relevant positive results for the dialogue strategy: Task ease-of-use, user expertise, expected behavior and future use.

TABLE III. USER SATISFACTION SURVEY RESULT

Criteria	***SCALE VALUES***
TTS Quality	4.08
ASR Quality	4.20
Task ease	4.40
Interaction pace	2.68
User expertise	4.20
System response	4.16
Expected behavior	5.00
Future use	4.60
User Satisfaction	**33.32**

V. CONCLUSION

This paper presents a method for the development of model-based learning automata algorithm for dialogue management that can learn from training samples to generate the system actions. This representation allows the system to automatically generate specialized actions that takes into account the current situation of the dialogue depending on the use of expected cumulative reward. This approach is appealing due to the following benefits: a) faster learning, b) reduced computational demands, c) Knowledge transfer. Some experiments have been performed to test the behavior of the system with respect to PARADISE framework. The results show the satisfactory operations of the developed approach. An important area for future research could be in the area of reinforcement schemes, to analyze the finest configuration parameters for the environment. We hope that such a formulation would lead to a finer learning curve that would mimic more closely the behavior of the learning algorithm. Our tests are geared towards stochastic domain of interest; this may form an interesting avenue for further research.

REFERENCES

[1] Mctear, M, Spoken Dialogue Technology: Toward the Conversational User Interface. Springer, London, 2004.

[2] Möller, S, Quality of Telephone-based Spoken Dialogue systems. Springer, London, 2004.

[3] López-cózar, R., Araki, M, Spoken, Multilingual and Multimodal Dialogue Systems. Development and Assessment. John Wiley & Sons Publishers, 2005.

[4] Pieraccini, R., Huerta, J, "Where do we go from here? Research and Commercial Spoken Dialog Systems," In: Proceedings of 6th SIGdial workshop on dialogue and discourse, Lisbon, Portugal, pp 1–10, 2005.

[5] Mctear, M, "Modelling Spoken Dialogues with State Transition Diagrams: Experiences with the CSLU Toolkit," In: Proc of ICSLP, Sidney, Australia, pp. 1223-1226, (1998).

[6] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and I. S. Busayapongcha, "A Form-Based Dialogue Manager for Spoken Language Applications," in Proc of ICSLP, Philadelphia, USA, pp. 701-704, 1996.

[7] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent ,"Towards conversational human–computer interaction," AI Magazine, 22(4), pp. 27–38, 2001.

[8] Sutton, and A. Barto, "Reinforcement learning: An introduction," MIT Press, 1998.

[9] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human–machine interaction for learning dialog strategies," IEEE Trans Speech Audio Processing 8(1), pp. 11–23, 2000.

[10] S. Singh, D. Litman, M. Kearns, and M. Walker, “Optimizing dialogue management with reinforcement learning: experiments with the NJFun system,” Journal of Artificial Intelligence Research 16: PP. 105–133, 2002.

[11] K.Scheffler, and S. Young, “Corpus-based dialogue simulation for automatic strategy learning and evaluation, in Proc of the NAACL Workshop on Adaptation in Dialogue Systems, 2001.

[12] O. Pietquin, and T. Dutoit, “A Probabilistic Framework for Dialog Simulation and Optimal Strategy Learning,” IEEE Transactions on Audio, Speech and Language Processing, 14(2): pp. 589–599, 2006.

[13] O. Pietquin, “A Framework for Unsupervised Learning of Dialogue Strategies, Preses Universitaries de Louvain, ISBN:2-930344-63-6, 2004.

[14] J.Henderson, O. Lemon, and K. Georgila, “Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data,” in Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI), 2005.

[15] J. Schatzmann, K. Weilhammer, M. N. Stuttle, and S. Young. “A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies,” in Knowledge Engineering Review, Cambridge University Press, 21, pp. 97-126, 2006.

[16] H. Cuay´ahuitl, S. Renals, O. Lemon, and H. Shimodaira, “Reinforcement Learning of Dialogue Strategies Using Hierarchical Abstract Machines”, in Proc. of IEEE/ACL SLT, 2006.

[17] C. J. C. H. Watkins, and P. Dayan, “Q-learning,” Machine learning, 8(3), pp. 229-256, 1992.

[18] T. Peak, R. Pieraccini, “Automating spoken dialogue management design using machine learning: an industry perspective,” Speech Communication, 50, pp. 716-729, 2008

[19] A. Nowé, K. Verbeeck, and M. Peeters, “Learning Automata as a Basis for Multi Agent Reinforcement Learning,” Learning and Adaption in Multi-Agent, ISSN 0302-9743, pp. 71-85, 2006.

[20] Thathachar, M., and Sastry, P., “Varieties of learning automata: An overview,” IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 32(6) 2002, 711–722.

[21] M. Walker, D. Litman, C. Kamm, A. Abella, “PARADISE:A framework for evaluating spoken dialogue agents,” in Proc of the 5th annual meeting of the association for computational linguistics(ACL-97), pp 271–280, 1997.