Learning Dialogue Strategies within the Markov Decision Process Framework

Esther Levin, Roberto Pieraccini, Wieland Eckert

AT&T Labs-Research,
180 Park Avenue, Florham Park, NJ 07932-0971, USA
(esther | roberto | eckert)@research.att.com

Abstract - In this paper we introduce a stochastic model for dialogue systems based on Markov decision process. Within this framework we show that the problem of dialogue strategy design can be stated as an optimization problem, and solved by a variety of methods, including the reinforcement learning approach. The advantages of this new paradigm include objective evaluation of dialogue systems and their automatic design and adaptation. We show some preliminary results on learning a dialogue strategy for an Air Travel Information System.

1. INTRODUCTION

Recent progress in the field of spoken natural language understanding [1] expanded the scope of spoken language systems to include mixed initiative [1-5, 7]. Currently there are no agreed upon theoretical foundations for the design of such systems. Looking at the history of speech recognition research and the tremendous progress due to the introduction of a computational model such as HMM, we believe that dialogue research could greatly benefit from a principled theoretical and computational description of the problem.

In this work we define a dialogue system as a system that tries to achieve an *application goal* in an *efficient* way through a series of interactions with the user. We show that by quantifying the terms *efficiency* and *achievement of application goal* in terms of an objective function, the dialogue system can be described as a known stochastic model - Markov Decision Process (MDP) - that can be used for learning the dialogue strategy for a given application.

The advantages of this new paradigm include objective evaluation of dialogue systems and their automatic design and adaptation.

We show some preliminary results on learning a dialogue strategy for an Air Travel Information System.

2. DIALOGUE SYSTEM AS A MARKOV DECISION PROCESS

In this section we will give a formal definition of a dialogue system. For clarity, we will illustrate it with a very simple tutorial example of *Day-and-Month Dialogue*,

0-7803-3698-4/97/$10.00 © 1997 IEEE.

where the goal of the system is to get the *correct* day and month values from the user through the *shortest* possible interaction.

We formalize a dialogue system by describing it in terms of a ***state space***, an ***action set***, and a ***strategy***.

The **state** of a dialogue system represents all the knowledge the system has about internal and external resources it interacts with (e.g. remote databases or machinery, user input, etc.). For our simple tutorial example, the state of the system includes only two entries: the day and the month, whose values can be either empty, or filled through interaction with the user. The total number of states is 411, including one empty initial state, 12 states for which the month is filled and the day isn't, 31 states in which the day is filled, but not the month, 366 states with complete dates, and a special final empty state.

The **action set** of the dialogue system includes all possible actions it can perform, such as interactions with the user (e.g. asking the user for input, providing a user some output, confirmations, etc.), interactions with other external resources (e.g. querying a database), and internal processing.

For our example, the action set include only four actions:

1. A question to the user asking for the value of the day.
2. A question to the user asking for the value of the month.
3. A more open-ended question asking for the value of the date (day and month).
4. A final action, closing the dialogue and submitting the form.

In actions 1, 2 and 3, the system asks the appropriate question, and activates a speech recognition system to obtain the user's answer.

When an action $\boldsymbol{a}$ is taken at state $\boldsymbol{s}$, the system's state changes to be $\boldsymbol{s'}$. For the day-and-month example, when the system is in an initial state and it asks the user for the month, the next state depends on the actual answer of the user as well as on the speech recognition performance of the system, and it can be any one among the 12 states in which the month is filled, but the day is not. The state transitions are modeled by transition probabilities $P_T(\, s(t+1) = s' \mid s(t) = s,\, a(t) = a\,)$.

A **dialogue session** corresponds to a path in the state space starting at the initial state and ending at a final state.

A **dialogue strategy** specifies, for each state reached, what is the next action to be invoked.

The next definition concerns with the main assumption of our model:

We assume that the goal of a dialogue system is to achieve an *application goal* in an *efficient* way through a series of interactions with the user.

Any dialogue system has an **application goal**: whether it is filling a form by obtaining information from a user (like in our tutorial example), or information retrieval, where the system is trying to provide some useful information to the user (like in the Air Travel example below). The **efficiency**, depending on application, represents dialogue duration, cost of internal processing, cost of accessing external databases or other resources, etc.

We further assume that for each application we can measure the system performance by an objective function C,

$$C = \sum C_i, \quad (1)$$

where the costs *Ci* measure the distance to the achievement of the application goal, and the efficiency of the interactions. Therefore, the goal of dialogue system design is to build a system with a strategy that minimizes this objective function. It has been shown in [9] that also an abstract cost reflecting *user satisfaction* with the system can be measured experimentally and modeled as a linear combination of costs as in equation (1). In a real system, the user satisfaction cost can constitute one of the terms in (1). For our tutorial example, where the goal of the system is to obtain the correct day and month values through the shortest possible interactions, the objective function includes three terms:

(2) *C = Wi * <# interactions> + We * <# errors> + Wf * <# incomplete values>.*

The first term is the expected duration of the dialogue; the second corresponds to the expected number of errors in the obtained values (ranging from 0 to 2); and the third measures the expected distance from achieving our application goal (this distance is 0 for a complete date, 1 if either day or month value is missing, and 2 if both are incomplete).

In order to reflect this objective function in our dialogue model, we associate a cost *c* to the taken action *a* in a state *s*. The cost incurred with any of the first three actions in day-and–month dialogue system is *Wi + We * number of errors.* If we assume that the concept error rate for recognition of month or day values separately (for questions 1 and 2) is *p1*, and together (for question 3) is *p2, p2 > p1*, then the expected cost accumulated when actions 1 or 2 are taken is *Wi +We*p1*, while for question 3 is *Wi +2*We*p2*. For action 4 (closing the dialogue and submitting the obtained date) the cost depends deterministically on the state in which this action is taken and is *Wi + 2Wf* for an initial state, *Wi + Wf* for states in which either the day value or month value is unfilled, and *Wi* for the states in which both values are filled in.

In general, the costs in MDP are described by the conditional distributions
PC(c(t) = c | s(t) = s, a(t) = a).

If we define the *session cost* as a sum of all the costs experienced by the system during a dialogue, then the objective function (1) corresponds to the expected dialogue session cost.

This quadruple of state space, action set, transition probabilities, and cost distributions defines a Markov decision process.

Of course, different strategies for the same system result in different expected session costs. Figure 1 shows three different strategies and their costs for the day-and-month system. We define an *optimal strategy* as the one that minimizes the objective function. For example, in figure 4, strategy 1 (where the system does not even engage in dialogue, closing the dialogue as the first action) is optimal when the recognition error rate is too high: *p1 > (Wf –Wi)/We.*

In strategy 2, the system opens the dialogue by asking the open ended question number 3, fills out the day and the month slots with the values recognized from the user response, and closes the session. In strategy 3, the system first fills up the day and then the month by engaging in actions 1 and 2, and then closes the session. Strategy 3 is optimal when the difference in error rates justifies a longer interaction: $p2 - p1 > Wi / 2We$.

In the following section we address the problem of finding the optimal strategy automatically.

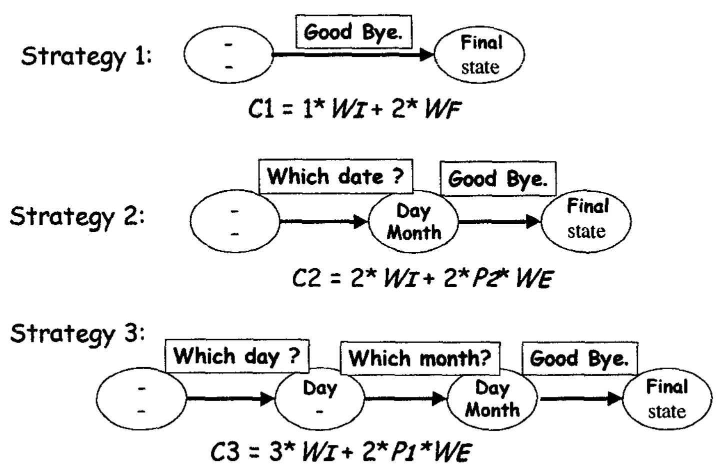

Figure 1: Three different strategies

2. TOWARDS AN AUTOMATIC DESIGN OF MAN-MACHINE DIALOGUE SYSTEM: THE REINFORCEMENT LEARNING APPROACH

Stating the problem of man-machine dialogue design as an optimization problem provides the following potential advantages:

Objective evaluation: It is possible now to grade different strategies for the same system just by comparing their expected cost. It is also possible to compare different systems that share the same objective function.

Automatic design: Since the problem of strategy design is cast as optimization problem, it is possible to devise methods for performing this optimization automatically.

Such automatic design procedure for finding the optimal strategy is the subject of the reinforcement learning discipline. For a tutorial on reinforcement learning look at

[6]. In this section we outline the computational issues involved in the problem of finding automatically the optimal strategy for MDP.

2.1 Computing an Optimal Strategy when the Model is Known

There exist in literature several techniques for computing the optimal strategy given the correct model parameters (the transition probabilities and the cost distributions), including *value iteration*, *policy iteration*, etc. These techniques are based on dynamic programming that can be used due to the Markovian nature of this model. They rely on the following definition:

Optimal value ***V(s)*** of a state ***s*** is the lowest expected cost incurred after the system left state ***s*** and until it reached the final state. The optimal value function is unique and can be defined as the solution to the simultaneous equations

$$V(s) = \min_{a} \left(\langle C(s,a) \rangle + \sum_{s'} P_T(s' \mid s,a)\, V(s') \right), \quad (3)$$

where $\langle C(s,a) \rangle$ is the expected cost for action a in state s.

This equation states that the optimal value of state ***s*** is a sum of expected instantaneous cost plus the optimal value of the next state, using the best available action $a^*(s)$. Given the optimal value function, the optimal strategy can be computed simply as

$$a^*(s) = \operatorname{argmin}_{a} \left(\langle C(s,a) \rangle + \sum_{s'} P_T(s' \mid s,a)\, V(s') \right). \quad (4)$$

Techniques like value iteration, policy iteration, and others iteratively solve equations (3) and (4) for computing the optimal strategy.

2.2 Learning the Optimal Policy: Model-Free Methods

However, in most of the interesting cases it is not possible to use the model-based methods above because of one or more of the following reasons:

- *The state space is very large* (or infinite). The number of equations in (3) is equal to the number of states, and it is impossible not only to solve them, but even to store in memory the optimal strategy.
- *Some of the MDP parameters are not known* in advance. As it is illustrated by the tutorial example above, some of the model parameters reflect the probability of user's response given the system question and the state of the dialogue. Other parameters can reflect the properties of external resources, such as a remote database, that are also unknown in advance.
- *Delayed reinforcement*. Some costs (e.g. total user satisfaction measured by a score that a user gives at the end of the dialogue) depend on actions taken during the session, but are available only at the end of the session.

Reinforcement learning (RL) is primarily concerned with obtaining the optimal strategy in such cases when the model is not known in advance, too large to compute, or reinforcement is delayed. In the reinforcement learning paradigm the system learns the optimal policy from interactions with the user. If during a dialogue session the system chose an action ***a*** in state *s*, and accumulated a total cost ***C*** until it reached the finite state, the estimated value of state ***s*** for action ***a*** is updated from this experience. The details of an actual update differ between the different RL algorithms, whether it is Q-learning, Monte Carlo style algorithms, on and off policy estimation algorithms, etc.. Note that unlike in an automatic speech recognition, where the estimation of HMM parameters constitutes the learning of the model, in RL the MDP parameters are not estimated, but rather the optimal policy is learned directly from interaction.

3. USING RL FOR LEARNING THE OPTIMAL STRATEGY FOR THE ATIS DOMAIN.

We used a Monte Carlo style reinforcement learning algorithm to learn the optimal strategy for a dialogue system based on the ATIS task. The possible actions of the system in this case include: *greeting* the user with an open ended question (i.e. *How can I help you*?); *asking* the user to provide information about a specific attribute of the task (e.g. origin, airline, departure time, etc.); *retrieving* data from the database according to the current user request (this action does not involve interaction with the user); *output* the retrieved data to the user; asking the user to *release* a constrain; and *closing* the dialogue.

We chose a very simple state description in order to simplify the learning. The state included three *templates* (a template is a set of keyword-value pairs that we used in our ATIS understanding system [3] as a meaning representation). The user template represents the meaning of user request interpreted in context; the data template includes the number of data tuples retrieved from the database according to the query based on user template; and the system template includes a keyword *OUTPUT* only if the action *output* was used in the past and data was output to the user.

The objective function for this application has four terms:

$$\textit{Objective function} = W1 * \#\,\textit{interactions} + W2 * \textit{cost of data retrieval} + W3 * \textit{cost of data presentation} + W4 * \textit{cost of user dissatisfaction}, \tag{5}$$

where the cost of data retrieval is linear with the number of tuples retrieved, cost of data presentation is zero for a reasonable number of tuples (very few, if the system has to verbalize the data, more if it can use a display), and rapidly increasing with the number of tuples thereafter, and the user dissatisfaction cost that penalizes with a fixed cost dialogues that fail to provide flight information to the user.

The system started with no knowledge of the task, i.e., each action had the same probability of being selected by the system at any step. A typical dialogue with an untrained system will look like the following:

System: RELEASE AIRLINE:	*Do you want to choose another airline?*
User:	*What?*
System: RETRIEVAL	
System: CONSTRAIN DEPARTURE TIME:	*When do you want to leave?*
User:	*Uh? I want to go to Boston*
System: OUTPUT DATA:	*There are 12000 flights...*
User:	*I don't understand, can you please show me*

	the flights from San Francisco to Boston?
System: CLOSE DIALOGUE:	*Thank you for using AT&T*

Of course the total cost for this dialogue is quite high, especially due to the high cost of retrieval (all the database was retrieved here) and data output (12000 flights).
After training the system learned the following optimal strategy: start the dialogue by greeting (the probability of getting more information from the user is higher with greeting rather than a specific question), ask constraining questions until the origin, destination, and airline are specified, and retrieve data from the database. After the retrieval, if the resulting data set is empty (because the query was over-constrained), release the airline constraint and retrieve again. If there are too many flights in the data set, ask to constrain the departure time and retrieve again. If at any point during the dialogue the retrieved data set has a reasonable number of flights, the data is output and the dialogue is closed.

An example of dialogue performed with a trained system is as follows:

System: GREETING:	This *is the AT&T flight information. How may I help you?*
User:	*I want to go to Boston*
System: CONSTRAIN ORIGIN:	*Where do you want to leave from?*
User:	*San Francisco*
System: CONSTRAIN AIRLINE:	*Which airline?*
User:	*Delta*
System: RETRIEVAL (30 Flights retrieved)	
System: CONSTRAIN DEPARTURE TIME:	*When do you want to leave?*
User:	*In the late afternoon.*
System: RETRIEVAL (3 flights)	
System: OUTPUT DATA:	*Flight ... leaves at..., flight .. leaves at, ...*
User:	*Thanks*
System: CLOSE DIALOGUE:	*Thank you for using AT&T*

Rather than conducting thousands of dialogues with the system in order to train it, we used a *user model* that is described in a companion paper [8]. The user model is a stochastic dialogue system that generates a reasonable user response to system actions. Different parameters of the user model will result in different learned strategies. The strategy described above was obtained by interactions with a user model that has a very high degree of compliance (i.e. very high probability of producing proper answers to system's questions).

4. Summary

In this paper we propose a formal quantitative model for man-machine dialogue systems. First, we introduce a general formalization of such systems in terms of their state space, action set and strategy. With this formalization we can describe any dialogue system without loss of generality. Then, we proceed with the main assumption that a good strategy for a dialogue system is minimizing an objective function that reflects the costs of all the important dialogue dimensions. With this assumption we can model any man-machine dialogue system using a Markov decision process, a stochastic model commonly used today for control, games, and other applications, and use reinforcement learning algorithms for designing the optimal strategy automatically. We used reinforcement learning algorithm to learn an optimal strategy for air travel based dialogue system, and showed that a system that started without any initial knowledge converged to a very reasonable strategy. This paradigm also allows us to objectively evaluate and compare different strategies and different systems for the same application.

References

[1] Proc. of *1995 ARPA Spoken Language Systems Technology Workshop* , Austin Texas, Jan. 1995.

[2] Glass, J. et al., "The MIT Atis System: December 1994 Progress Report", Proc. of *1995 ARPA Spoken Language Systems Technology Workshop*, Austin Texas, Jan. 1995.

[3] Levin, E., Pieraccini, R., Di Carlo, A., "User Initiated Mixed Initiative Dialogue for Database Information Retrieval," to appear in LuperFoy, S. (editor) *Automated Spoken Dialogue Systems*, MIT Press.

[4] Sadek, M.D., Bretier, P., Cadoret, V., Cozannet, A., Dupont, P., Ferrieux, A., & Panaget, F., "A Cooperative Spoken Dialogue System Based on a Rational Agent Model: A First Implementation on the AGS Application," Proceedings of the *ESCA/ETR Workshop on Spoken Dialogue Systems*, Hanstholm, Denmark, 1995.

[5] Stallard, D., "The BBN ATIS4 Dialogue System," Proc. *of 1995 ARPA Spoken Language Systems Technology Workshop*, Austin Texas, Jan. 1995.

[6]Kaelbling, L. P., Littman, M. L., Moore, A. W., "Reinforcement Learning: A Survey," *in Journal of Artificial Intelligence Research*, No. 4, pp. 237-285, May 1996.

[7]Marcus, S. M., Brown, D. W., Goldberg, R. G., Schoeffler, M. S., Wetzel, W. R., and Rosinski, R. R. "Prompt Constrained Natural Language - Evolving the Next Generation of Telephony Services," Proc. of *ICSLP '96*, Philadephia (PA), October 1996.

[8] Eckert, W., Levin, E., Pieraccini, R., "User Modeling for Spoken Dialogue Systems," in Proc. *IEEE ASR Workshop*, Santa Barbara, 1997.

[9] Walker, M. A., Littman, D. J., Kamm, C. A., Abella, A., "PARADISE: A Framework for Evaluation of Spoken Dialogue Agents," in Proc. *35-th Annual Meeting of the Association for Computational Linguistics,* Madrid, Spain, 1997.