A STOCHASTIC MODEL OF COMPUTER-HUMAN INTERACTION FOR LEARNING DIALOGUE STRATEGIES

Esther Levin and Roberto Pieraccini
AT&T Labs-Research, 180 Park Avenue, Floram Park, NJ 07932-0971, USA
(esther|roberto)@research.att.com

ABSTRACT

Recent progress in the field of spoken natural language understanding expanded the scope of spoken language systems to include mixed initiative dialogue. Currently there are no agreed upon theoretical foundations for the design of such systems. In this work we propose a stochastic model of computer-human interactions. This model can be used for learning and adaptation of the dialogue strategy and for objective evaluation.

1. INTRODUCTION

Man-machine interactions, from simple touch tone menus to more complex speech based systems are ubiquitous in today's world. In the natural language and AI research communities there are even more complex examples of dialogue systems that make use of natural ways of communication like speech and language [1, 2, 3, 6].

In this paper we show that such dialogue systems can be formally described in terms of their state space, action set and strategy. The *state* of a dialogue system represents all the knowledge the system has at a certain time about internal and external resources it interacts with (e.g. remote databases or machinery, user input, etc.). The *action set* of the dialogue system includes all possible actions it can perform, including different 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. The *dialogue strategy* specifies, for each state reached, what is the next action to be invoked. The strategy is usually devised by a designer that tries to predict all possible situations the dialogue system can get into (in terms of conditions on the dialogue state), and what are the appropriate actions to take in those situations. There exist no scientifically motivated guiding principles for the design of the strategy, and therefore this process of design can be considered as an art, rather than engineering or science. As a result, today there are no known methods for objective evaluation and comparison of dialogue systems, even those designed for the same application. There are no methods for automatizing the design of the strategy, neither for an automatic adaptation of the dialogue strategy in presence of interactions with users and their feedback.

Here we propose to state the problem of dialogue strategy design as an optimization problem. We assume that there is an implicit objective function (expected dialogue cost) that drives the design of the dialogue system. This objective function can be written as a sum of different terms, each representing the cost of a particular dialogue dimension. Some of this dimensions can be measured directly by the system, like dialogue duration, cost of internal processing, cost of accessing external databases or other resources, cost of ineffectiveness (e.g. number of errors the system made due to poor speech recognition); others quantify such abstract dimensions as user satisfaction (e.g. a simple happy/not happy-with-the-system feedback from the user at the end of dialogue, number of users hanging up before the completion of the dialogue goal, etc.). The actions taken by the system may affect some or all of the terms of the objective function, and therefore an optimal strategy is a result of a correct trade off between them. We illustrate this formalization in two examples. One is a simple form filling application for which an optimal strategy is derived analytically, and it quantifies a reasonable strategy adopted in real systems [5]. The other example refers to our research database retrieval dialogue system [6]. Due to the complexity of the system, the optimal strategy cannot be derived analytically in this case. Hence we propose to represent the dialogue system as a stochastic model known as Markov Decision Process (MDP) and to use reinforcement learning algorithms [4] to find the optimal strategy automatically.

2. FORMALIZATION

Here we introduce a formal model that describes, without loss of generality, any man-machine dialogue system in terms of state space, action set and strategy.

- s_t - the state of the dialogue system at time t that includes the current available information about internal and external processes controlled by the dialogue system.
- **S** - the space of the system states (finite or infinite). It includes two special states: s_I is an initial state, and s_F is a final state.
- a_t - the action performed by the system at time t. The next state of the dialogue, s_{t+1}, depends on the current state s_t and current action a_t. This dependence is in general not deterministic.
- **A** - the set of all system actions (usually is finite).
- π- the strategy of the system described in terms of a mapping between the state space **S** and the action set **A**. The strategy of a dialogue system specifies conditions on the current state upon which a certain action is invoked.

To illustrate these concepts we consider a very simple form filling application: the goal of the system consists in filling all the slots of a form asking the user the appropriate questions. We assume that the user will always answer the system questions obediently.

State space: A state s of the system is represented by $k_1, \ldots, k_N$ where N is the number of slots to be filled and

$$k_i = \begin{cases} 1 & \text{if slot } i \text{ is filled} \\ 0 & \text{otherwise} \end{cases} . \qquad (1)$$

S consists, in this case, of 2^N states.

Action set: $\mathbf{A} = \{A_0, A_1, \ldots, A_N\}$, where $A_i, i = 1, \ldots, N$ represents a question about the i-th slot. Generally, an action corresponding to a system's question requires the use of a speech recognizer, an understanding system, or other devices for different input modalities, in order to collect the answer from a user.

A_0 corresponds to the action of ending the dialogue and

submitting the form.
State transitions: In this system the state transitions are deterministic: the system starts in the initial state where $k_i = 0, i = 1, \ldots, N$; an action $A_i, i = 1, \ldots, N$ deterministically changes the current state to the next one, for which $k_i = 1$; A_0 brings the system to the final state.
Strategy: Common sense will suggest the following *reasonable strategy* for this system: for each state ask a question about one of the slots that are not filled yet, and submit the form when all the slots are filled.

3. A QUANTITATIVE MODEL

As was explained in the previous section, the system design for new application consists of determining the state space **S**, the action set **A**, and finding a good strategy, that is usually the most time consuming part of the design. Since the state space in most of the applications is very big or even infinite, strategy design involves an iterative process that consists in testing the system, finding states in which the current strategy results in an unreasonable action, correcting it, and so on, until a reasonably stable strategy is found. In what follows we propose a model for quantifying the concept of good strategy. The advantages of this quantitative model include the possibility of objective evaluation and an automatic way of designing an optimal strategy, or learning it from real or simulated data.

3.1. Markov Decision Process

The quantitative dialogue model relies on the description of dialogue system in terms of Markov Decision Process (MDP) [4].

Formally a Markov Decision process is a stochastic model that consists of:
• State space **S**, including initial and final states s_I and s_F. At each discrete time step only one state is active, the active state at $t = 0$ being the initial state s_I;
• Set of actions **A**;
• Transition probabilities $P_T(s_t|s_{t-1}, a_{t-1})$, describing the probability of the next active state given the previous one and the previous action. The Markovian property of this model assumes that:

$$P(s_t|s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = P_T(s_t|s_{t-1}, a_{t-1}). \quad (2)$$

• Each action in a given state is associated with a reinforcement (cost or reward) $c \in \Re$. An MDP is characterized by cost probabilities $P_C(c_t = C|s_t = S, a_t = A)$, describing the probability of getting a reinforcement C when executing an action A in state S.
In a generative mode, the system starts at state s_I, chooses an action from a possible set of actions **A**, receives a reinforcement signal c drawn randomly from an appropriate cost probability P_C, and reaches the next state according to P_T. This process will continue until the system reaches the final state s_F. Assuming that the system reached a final state, the cost of such session (path through the state space) is the sum of all the cost involved:

$$C_D = \sum_{t=0}^{t=T_F} c_t, \quad (3)$$

where $s_{T_F} = s_F$. The expected cost $\overline{C_D}$ is the expectation of session cost with respect to the two probabilities, P_T, and P_C and it depends on the particular sequence of actions chosen during the session.

A strategy π of an MDP is a mapping between states and actions, indicating which action the system should take in each state. An *optimal* strategy π^* is a strategy that minimizes the expected cost $\overline{C_D}$,

3.2. Dialogue system as an MDP

To use MDP for modeling man-machine dialogue requires the following assumptions:
• The transitions between the states of the dialogue are characterized by a stochastic Markov process $P_T(s_t|s_{t-1}, a_{t-1})$.
• There are costs c associated to dialogue actions in each state of the dialogue distributed according to $P_C(c_t = C|s_t = S, a_t = A)$.
• The optimal strategy for a dialogue system results from a minimization of an expected dialogue session cost.
With this assumptions we pose the problem of finding a good strategy for a dialogue session as an optimization problem of minimizing an expected cost of a dialogue session. In general, the expected cost can be written as a sum of multiple terms,

$$\overline{C_D} = \sum_i \lambda_i C_i, \quad (4)$$

where the terms C_i represent the expected value of important dimensions of quality of a dialogue session, i.e., the duration of a dialogue, number of errors in recognition or understanding of the user, cost of accessing external databases, distance from the user goal, etc., and λ_i are positive weights. The λ's measure the actual cost per unit of respective dimensions, and control their relative importance.

4. EXAMPLE 1: SIMPLE FORM FILLING

In this example we assume that the user input is noisy (e.g. due to errors introduced by a speech recognizer), and the goal of the system consists in filling the form *correctly*. We assume that the error rate p of user answers depends on the type of question the system asked.
State space: as in the example in section 2.
Action set: $\mathbf{A} = \{A_0, A_1, \ldots, A_N, A_{1,2}, \ldots, A_{N,N-1}\}$, where A_0 corresponds to the action of ending the dialogue and submitting the form; $A_i, i = 1, \ldots, N$ represents a question about the i-th slot; $A_{i,j}, i, j = 1, \ldots, N$ is a compound question about i-th and j-th slot (an example of a compound question is asking the user for a date, instead of asking separately for month and day number). We assume that the error rate for user's answer is p_1 for single-slot questions $A_i, i = 1, \ldots, N$, and $p_2 > p_1$ for compound questions.
State transitions: Each question modifies the current state similarly to the example of section 2 by deterministically changing the values of k_i to 1 for those entries i the user was asked for.
Reasonable Strategy: A strategy in this example depends on the actual values of p_1 and p_2. If $p_2 \approx p_1$, a reasonable strategy will prefer compound questions to reduce the duration of the dialogue. When p_2 is much larger than p_1 and compound questions will result in an unreasonable error rate, single-slot questions should be preferred.
Expected dialogue cost: Here the goal of the system is to fill *correctly* the form slots with shortest dialogue, and therefore the dialogue cost is

$$\overline{C_D} = \lambda_Q N_Q + \lambda_U N_U + \lambda_E N_E, \quad (5)$$

where:
• N_Q is the expected number of questions in the session,
• N_U is the expected number of unfilled slots in the submitted form,
• N_E is the expected number of erroneously filled slots in the form determined by the error rate for the kind of question used to fill the slots.

Cost distributions: The costs in this application are deterministic:

$$c(a_t, s_t) = \begin{cases} \lambda_Q + \lambda_E p_1 & \text{if } a_t = A_i, i = 1, \ldots, N \\ \lambda_Q + \lambda_E p_2 & \text{if } a_t = A_{ij}, i, j = 1, \ldots, N \\ \lambda_U N_U & \text{if } a_t = A_0 \end{cases} . \quad (6)$$

Optimal strategy: The optimal strategy π^* minimizing the expected cost (5) quantifies the *reasonable* strategy by specifying the exact conditions on error rates p_2 and p_1 for choosing the right actions:

- Don't ask any questions and submit an empty form if $\lambda_U < \lambda_Q + \lambda_E p_1$.
- Else, if the error rate for a composite question is much larger than the one for a single question, $p_2 - p_1 > \frac{\lambda_Q}{2\lambda_E}$, ask only single-slot questions without repetitions, and submit the form when filled.
- Else, ask composite questions about unfilled slots in the form, and submit the form when filled.

5. EXAMPLE 2: AMICA

AMICA [6] dialogue system is a research mixed initiative spontaneously spoken input dialogue system that was initially developed for the ARPA ATIS task. The application here is an intelligent interface between a user and a relational database.

State space: The current state of the system is represented by $s = (M, Q, N_D, C, R)$, where M is a meaning template representing the current user request (obtained by a speech understanding module); Q is a database query; N_D is the number of data tuples obtained by retrieving data from the database according to Q, with $N_D = -1$ if no retrieval was attempted yet; C is a template of additional constraints; R is a template of constraints to relax.

Action set :

$$\mathbf{A} = A_Q \bigcup \mathbf{A_C} \bigcup \mathbf{A_R} \bigcup A_O \quad (7)$$

where:

- A_Q is an action that forms a query Q (i.e. appending the additional constraints from C to M, and removing the relaxed constraints specified in R), and retrieving the data from the database according to Q.
- $\mathbf{A_C}$ is a set of actions that correspond to asking the user for an additional constraint. This set is parameterized by the attribute the system suggests to constrain.
- $\mathbf{A_R}$ is a set of actions that correspond to asking the user to relax a constrain. This set is parametrized by the attribute the system suggests to relax.
- A_O is the action of showing or verbalizing the retrieved data to the user.

State transitions: The state transitions in this case are deterministic for all actions, except the actions of asking the user for constraining/relaxing the query, where the next state is determined by the user's answer to the question with the appropriate probability: the system starts in an initial state where M is set to the initial user's request, C, Q, R are empty, and $N_D = -1$; actions in $\mathbf{A_C}$ modify the C template in the current state by appending or not the appropriate constraints according to the user's answer; actions in $\mathbf{A_R}$ modify R template in the current state by appending or not the appropriate constraints according to the user's answer; A_Q sets Q and N_D; A_O brings the system to the final state.

Reasonable Strategy: A strategy that we found useful and reasonable for this system is as follows: the system starts in an initial state in which an initial user's query was specified. If this query is under-constrained (we specified a set of conditions on the query when we believe that it might result in too large of a data set to be retrieved from the database) the system will generate a question asking for additional constraints. Then a query is formed according to the user's answer, and the data is retrieved. There are three possible situations: if there is no data to match the request ($N_D = 0$), the system will ask the user to relax a constraint, form a new query, and retrieve the data; if the number of tuples retrieved is too large (e.g. $N_D > 3$) the system will ask the user for additional constraints; otherwise (e.g. $0 < N_D < 3$), the system will output the data.

Expected dialogue cost:

$$\overline{C_D} = \lambda_Q N_Q + \lambda_R \overline{N_D} + \lambda_0 \overline{f(N_D)}, \quad (8)$$

where N_Q is the expected number of questions measuring the length of the dialogue, $\overline{N_D}$ is the expected number of tuples retrieved from the database during the session measuring the cost for data retrieval and $\overline{f(N_D)}$ is the expected channel cost associated to the number of tuples the system showed the user by the end of the session. It reflects the preference of users for short and concise, but non-empty, outputs. In our case:

$$f(N_D) = \begin{cases} 0 & 1 \le N_D \le 3 \\ C_1 & N_D \le 0 \\ C_2 & N_D \ge 3 \end{cases} , \quad (9)$$

Referring to the system description in [6] we can map specific modules to cost terms of equation (8): the minimal information module tries to minimize the database access cost N_D; the constraining and relaxation modules try to minimize the cost of the output channel $f(N_D)$; and all of them contribute to the duration cost N_Q.

Cost distributions: The costs are random variables as follows:

$$c(a_t, s_t) = \begin{cases} \lambda_Q & a_t \in \mathbf{A_C} \bigcup \mathbf{A_R} \\ \lambda_R N_D & a_t = A_Q \\ \lambda_O f(N_D) & a_t = A_O \end{cases} , \quad (10)$$

Optimal strategy: The optimal strategy minimizing the expected cost, trades off the cost components of equation (8). We cannot derive the optimal strategy analytically. Currently we are experimenting with a reinforcement learning algorithm for learning the optimal strategy from interactions [4].

6. 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, but it does not provide a quantitative analysis of dialogue system qualities. 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. This paradigm also allows us to objectively evaluate and compare different strategies and different systems for the same application.

REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] Pieraccini, R., Levin, E., "AMICA: the AT&T Mixed Initiative Conversational Architecture," Eurospeech 97, Rhodes (Greece), Sept. 1997