# ASR SYSTEM MODELING FOR AUTOMATIC EVALUATION AND OPTIMIZATION OF DIALOGUE SYSTEMS

Olivier Pietquin
*Faculté Polytechnique de Mons – TCTS Lab*
*Parc Initialis – Av. Copernic, 1*
*B-7000 Mons - Belgium*
e-mail : pietquin@tcts.fpms.ac.be

Steve Renals
*University of Sheffield – Computer Science Dept.*
*Regent Court – 211 Portobello Street*
*Sheffield S1 4DP – United Kingdom*
e-mail : s.renals@dcs.shef.ac.uk

**ABSTRACT**

Though the field of spoken dialogue systems has developed quickly in the last decade, rapid design of dialogue strategies remains uneasy. Several approaches to the problem of automatic strategy learning have been proposed and the use of Reinforcement Learning introduced by Levin and Pieraccini is becoming part of the state of the art in this area. However, the quality of the strategy learned by the system depends on the definition of the optimization criterion and on the accuracy of the environment model.

In this paper, we propose to bring a model of an ASR system in the simulated environment in order to enhance the learned strategy. To do so, we introduced recognition error rates and confidence levels produced by ASR systems in the optimization criterion.

## 1. INTRODUCTION

In the last few years, research in the field of spoken dialogue systems has experienced increasing growth. The automation of dialogue strategy design is a leading domain of investigation, and the treatment of dialogue system design using the formalism of Markov Decision Processes (MDPs) and Reinforcement Learning (*RL*) was proposed by Pieraccini and Levin [1]. However, to obtain a fully automatic procedure, the learning agent needs either real interactions with a user through an automatic speech recognition (ASR) system, a large amount of corpus data or a sequence of simulated interactions with a virtual user [2]. The latter option is widely preferred, since several thousand dialogues may be necessary to train even a simple system. Moreover, to automatically learn the optimal strategy within the framework of MDPs using RL algorithms, it is necessary to express the strategy design as an optimization problem and then to define an optimization criterion. Such a criterion drives the design of the strategy and may be interpreted as a dialogue cost.

In an attempt to simulate a realistic scenario, in which the dialogue system user interacts with it via an imperfect speech recognizer, we have developed a simulated environment in which the recognizer is connected to an RL agent. This enables us to naturally introduce recognition error rate and confidence levels in the optimization criterion used in the learning algorithms.

Experiments in a simulated environment have indicated that incorporating recognition confidence levels into the reinforcement learning system results in an improved strategy in several cases. The results may also be used for objective evaluation of dialogue costs in order to compare strategies between them.

## 2. DIALOGUE AS A MARKOV DECISION PROCESS

A human-machine dialogue system may be expressed as a Markov Decision Process (MDP) in terms of states, actions and strategy if we assume the Markovian property which is met if the state $s_{t+1}$ of the system at time t+1 depends exclusively on the state $s_t$ at time t and on the action $a_t$ taken by the system when in state $s_t$ :

$$P(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = \boldsymbol{P_T}(s_{t+1} \mid s_t, a_t)$$

where $\boldsymbol{P_T}$ is the transition probability.

The general MDP formalism can be described as follow:

- $\underline{\mathbf{s}}_\mathbf{t}$ is the state of the system at time t. Each state is built so that it describes the information obtained thus far by the dialogue system. To avoid constraints linked to the Markov property, $s_t$ may contain data about the system's history. There are two special states : $s_I = s_0$ is the initial state of the system (at time $t_0$) and $s_F$ is the final state reached at $t = T_F$
- $\underline{\mathbf{a}}_\mathbf{t}$ is the action performed by the system at time t. Actions are taken from the finite action set $\underline{\mathbf{A}} = \{a_i\}$. Typically, $a_i$ are spoken utterances or database queries.

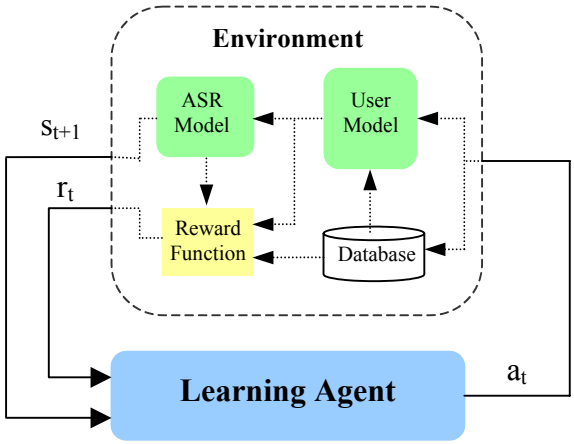


**Figure 1 : Learning Process**

- The reinforcement signal, $r_{t,}$ is the immediate cost or reward of having performed action $a_t$ when in state $s_t$. Its value indicates how good or bad performing an action in a given state is. The cost $\underline{\mathbf{R}}_d$ of a complete dialogue can be expressed as the sum of all the reinforcement signals obtained from the environment in the dialogue session overall:

$$R_d = \sum_{t=o}^{t=T_F} r_t$$

- $\underline{\pi}$ is the strategy of the system. It rules the system's behavior by creating a mapping between the state space $\underline{\mathbf{S}} = \{s_j\}$ (which can be finite or infinite) and the action set $\underline{\mathbf{A}}$. $\pi$ takes a single state as its argument: $\pi(s_j)$ defines which action $a_i$ is to be performed when in state $s_j$. An optimal strategy $\underline{\pi^*}$ is a strategy that minimizes the expected dialogue cost :

$$\overline{R_d} = E[R_d]$$

For example, in the case of a simple form filling application, the system's goal would be to fill all the fields of the form by asking questions to the user. In this case, we can describe the corresponding MDP as follows:

- **States** are represented by vectors $[f_1,\ldots,f_N]$ where N is the number of fields to fill and each $f_l$ is a boolean value {0,1} that indicates whether the $l^{th}$ field is filled or not. The size of the state space is thus $2^N$.
- **Actions** are questions about each of the N fields. Each action requires an ASR system to get the answer from the user. There are N different actions in the action set $A = \{A_1, \ldots A_N\}$.
- An obvious **strategy** would be to ask questions about unfilled fields until all fields are complete. Such a strategy is system-directed; the will of the user to participate actively to the dialogue is not taken into account. Hence this is a sub-optimal strategy.

## 3. SIMULATING THE ENVIRONMENT

As mentioned previously, training a system to find the optimal strategy of an MDP requires a large amount of data. Several thousand dialogues are usually necessary to reach this goal. To avoid the effort of running tests with real users, Pieraccini et al [2] proposed the creation of a virtual user to interact with the learning system. We go a step further by adding a model of an imperfect ASR system in our simulated environment (Figure 1).

Since dialogue systems are generally designed for voice-based consultation of databases, the virtual environment contains the database corresponding to the specific task to be learned. It also contains a virtual user and a reward building block in addition to the ASR system model.

### 3.1 Intention-based communication

In the simulated environment, communication takes place at the intention level rather than at the word sequence or speech signal level, as it would be in the real world. We regard an intention as the minimal unit of information that a dialogue participant can express independently. Intentions are closely related to concepts, speech acts or dialogue acts. For example, the sentence "I'd like to buy a desktop computer" is based on the concept buy(desktop).

It is unnecessary to model environment behavior at a lower level, because strategy optimization is a high level concept. Additionally, concept-based communication allows error modeling of all the parts of the system, including natural language understanding. More pragmatically, it is simpler to automatically generate concepts compared with word sequences (and certainly speech signals), as a large number of utterances can express the same intention.

### 3.2 User Model

In the real world, users display goal-directed behavior and prefer mixed-initiative to system-directed dialogues in which straight answers are required by the system. Starting from these observations, we built a partially stochastic simulated user (Figure 2) that allows mixed-initiative behavior and consistent confirmation sub-dialogues, with a main user-goal randomly defined for each dialogue session.

Before each simulated dialogue, a goal template based on the database's architecture (e.g. its tables and columns) is automatically built. According to this template, the model produces concept arrays to answer the system's questions. The stochastic part can simulate mixed-initiative behavior as it contains probabilities to answer multiple questions or non-asked arguments as well as probabilities to relax constraints, and to confirm or refute the propositions of the system. The simulated user may also stop the dialogue before its natural end after a fixed number of turns and express its unhappiness regarding the quality of the system.

The goal template may also be used to simulate the user's satisfaction as it can define whether or not the user's aim has been reached at the end of each dialogue session.

### 3.3 ASR System Model

Until now, even if previous reinforcement-learning-based systems took into account the imperfections of the ASR system that links the human user to the dialogue manager to optimize their strategy [1][3], few of them really used the ASR word error rate. Those systems that did take ASR errors into account, typically only implemented a single error rate, and did not use the specificity of each speech recognition task in order to

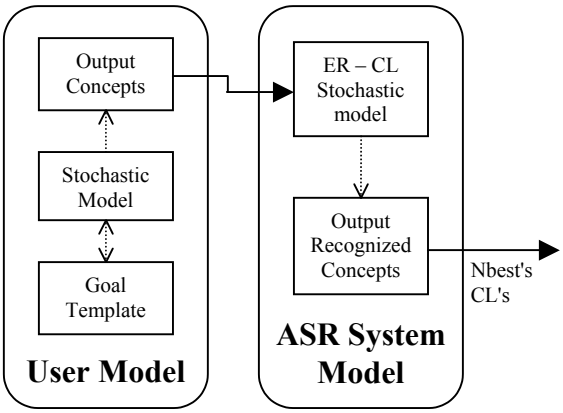


**Figure 2 : Models of the user and the ASR System (ER and CL respectively stand for Error Rate and Confidence Level)**

improve the learned strategy.

In this work, we implemented a more complex ASR system model (Figure 2) with different confidence level and error rate distributions for a finite number of recognition task types in order to approximate as well as possible the real case. Indeed, recognition performances vary from task to task and several recognition task types may be distinguished, including digits, numbers, dates, and unrestricted continuous speech. For each specific task, the ASR system model knows the average error rate and the Confidence Level (*CL*) distribution. The CL is a number between 0 and 1, based on acoustic measurements and defining how sure the system is to have performed a good recognition [4]. Its distribution is composed of two distinct curves (as shown on Figure 3) respectively for good and bad recognition results. As those curves cover each other, it is unavoidable to reject some well-recognized utterances as well as to accept few bad recognition results by defining a single CL threshold.

Figure 3 represents a CL distribution output from a real ASR system, obtained using some of its training data (isolated words). Thus these results are rather optimistic as the curves would be flatter in reality. Since the learning process is based on recognition performances for specific tasks relative to each other, absolute individual results are not so important.

In the complete simulated environment the ASR system model is connected to the user model, which transmits concept lists to it according to the goal template, the dialogue history and the last utterance of the system. When a concept list is received, the ASR system model splits it into individual concept elements, and performs the following algorithm:

```
- For each concept of the list
  {
    ➢ Choose randomly a number R between 0 and 1
    ➢ If R < RER(current concept)    // ASR error
        {
          • Substitute the current concept with
            another of the same nature (simulate
            an ASR error)
          • Produce a partial confidence level
            consistent with the "bad recognition"
            curve of the corresponding CL
            distribution
        }
```

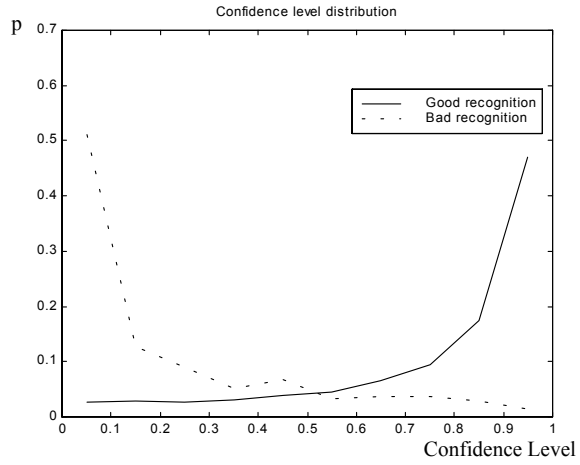


**Figure 3: Confidence Level distribution for distinct words**

```
    ➢ Else           // Good recognition
        {
          • Transmit the current concept without
            substitution
          • Produce a partial confidence level
            according to the "good recognition"
            curve of the corresponding CL
            distribution
        }
  }
- Transmit the new concept list to the system
-  Generate a global confidence level for the
   list by multiplying all partial confidence
   levels.
```

This algorithm may be iterated if we want to simulate the production of the N best recognized utterances. This is useful to generate natural confirmation sub-dialogues for example, giving the user the choice between concepts associated with similar confidence levels.

### 3.4 Reinforcement Signal

The reinforcement signal $\mathbf{r_t}$ (or reward function) gives information about the immediate cost of a system's action. In our case, at each turn, the system gets a reinforcement signal as the sum of several weighted terms :

$$\mathbf{r_t} = W_t N_t + W_{dba} N_{dba} + W_{pr} N_{pr} - W_{cl} CL - W_s f(U_s)$$

with :

- $N_t$ : set to 0 if $s_{t+1} = s_F$, 1 otherwise.
- $N_{dba}$ : Number of database accesses
- $N_{pr}$ : Number of presented data
- CL : Confidence Level of the current user's utterance
- $f(U_s)$ : Function of the modeled user's satisfaction ($U_s$)
- $W_x$ : Adjustable positive weights

Minimisation of the expected cost (obtained by summing the reinforcement signal) will result in a strategy that trades off the length of the session (as $N_t$ is a punishment for not reaching the final state), the number of database accesses, the amount of information transmitted to the user, the user's satisfaction and the confidence level over the entire session (ie, the likelihood of the dialogue).

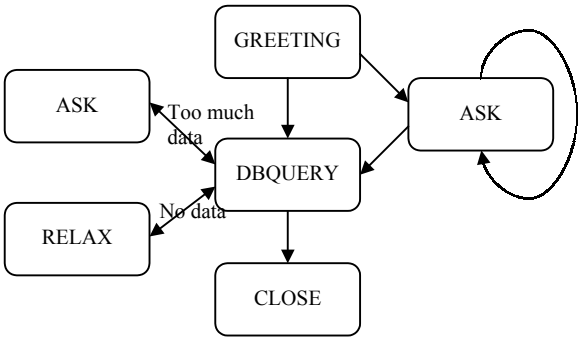


**Figure 4 : Learned strategy. After the GREETING, the system either has enough information (with a HIGH CL) to perform a DBQUERY action either it has to ASK for more data from the user. If the DBQUERY result is empty, it has to RELAX some constrains. If the DBQUERY result is too large, it has to add some constrains (ASK).**

## 4. EXPERIMENT

The dialogue application that we tried to optimize was an automatic computer dealing system. The database contains 350 different computer configurations split into 2 tables (for notebooks and desktops) containing 6 fields each : pc_mac (pc or mac), processor_type , processor_speed, ram_size, hdd_size and brand. We built a Java implementation, keeping in mind that the final dialogue application would use VoiceXML technology to interact with real users (no optimization of help sub-dialogues).

The system's MDP is described as follows :

**Action Set** : The action set contains 6 generic actions :
- GREETING : e.g. "How may I help you ?"
- ASK(**arg**) : ask to constrain the value of **arg**.
- CONF(**arg**) : ask to confirm the value of **arg**.
- RELAX(**arg**) : ask to relax the value of **arg**.
- DBQUERY : perform a database query.
- CLOSE : present data and close the dialogue session.

where **arg** may be the table's type (notebook or desktop) or one of the 6 tables fields. The action set's size is 24.

**State Space :** Each state is represented by two features.
- A vector of 7 boolean values [$f_x$] (one for each value of **arg)**. Each of these $f_x$ is set to *true* if the corresponding value of **arg** is known (for example if the user specified to search in the notebooks table, $f_0$ is set to true).
- Information about the confidence level of each $f_x$ set to true. In this case we only considered 2 possible values for the confidence level (HIGH or LOW) in order not to increase too much the size of the state space. Note that DBQUERY actions will only include values with a HIGH confidence level.

For each value of **arg**, there are 3 different possibilities for the corresponding slot in the state representation : {$f_x$ = *false*, CL = *undef*}, {$f_x$ = *true*, CL = LOW}, {$f_x$ = *true*, CL = HIGH}. This leads to $3^7$ possible states.

**RL algorithm** : We chose the "Exploring Starts Montecarlo" algorithm [5] as the learning agent interacts with a simulated environment and so, is in a pure learning process. Thus it doesn't have to follow any consistent strategy but has to evaluate all the state-actions pairs as fast and as many times as possible.

## 5. RESULTS

After several thousand simulated dialogue sessions, the system adopts a stationary strategy, roughly described in figure 4, which appears to be optimal (we have not shown confirmation processes in order to keep the figure readable). One can say that this strategy is similar to the one we would obtain without simulating ASR system's behavior and including confidence levels into the reinforcement signal. Nevertheless, we noticed significant differences mainly standing in the order in which constraining questions (ASK) occurs. Indeed, it appeared that only 4 **arg** values are generally necessary to obtain a satisfactory database query result. Without using the ASR system model, the dialogue manager would ask any of the four values of **arg** with the same probability. In order to increase the overall confidence level and to reduce the average length of a dialogue session (by avoiding useless confirmation sub-dialogues) our system asks questions about values that present better recognition results as numbers for example and prefers asking for a RAM size than a computer brand.

## 6. CONCLUSIONS

By introducing a simulated ASR system in the environment model of a reinforcement-learning agent, we managed to take into account the specificity of different recognition tasks in order to learn an optimal dialogue strategy. Such a model of an ASR system brings the simulated environment closer to reality. Following the same idea, we can imagine the introduction of a function of the ASR vocabulary size in the reinforcement signal of such a learning system and thus generalize our approach of the ASR system model.

Another improvement would be the introduction of intention-level confidence in the reinforcement signal instead of (or in addition to) word-level confidence. In this manner, the learning agent would try to maximize the likelihood of the dialogue at the semantic and pragmatic level and would normally lead to more consistent dialogues. Nevertheless, context tracking is strongly task-dependent and some generalization of techniques like *centering* [6] to build stochastic models like our ASR system model seems difficult.

## 7. REFERENCES

[1] R. Pieraccini, E. Levin, W. Eckert 'A Stochastic Model of Human Machine Interaction for Learning Dialog Strategies'. *IEEE Transactions on Speech and Audio Processing, Vol. 8 pp 11-23. 2000.*

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

[3] S. Young, K. Sheffler, 'Probabilistic Simulation of Human-Machine Dialogues' *Proc. ICASSP, Istanbul, Turkey, 2000.*

[4] E. Mengusoglu, C. Ris , 'Use of Acoustic Prior Information for Confidence Measure in ASR Applications', *Eurospeech 2001 Scandinavia, Aalborg, September 2001.*

[5] R. S. Sutton, A.G. Barto 'Reinforcement Learning : An Introduction' *MIT Press 1998*

[6] M. A. Walker. 'Centering, Anaphora Resolution, and Discourse Structure'. *In Marilyn A. Walker, Aravind K. Joshi, and Ellen F. Prince, editors,* Centering in Discourse. *Oxford University Press, 1998*