13

Optimising Spoken Dialogue Strategies within the Reinforcement Learning Paradigm

Olivier Pietquin
Ecole Supérieure d'Electricité (Supélec)
France

1. Introduction

Human-Computer Interfaces are now widely studied and become one of the major interests among the scientific community. More and more electronic devices surround people in their day-to-day life. This exponential incursion of electronics in homes, cars and work places is not only due to its ability to ease the achievement of common and boring tasks or the continuously decreasing prices but also because the increasing "user-friendliness" of interfaces makes it easier to use.

Being studied for more than fifty years, speech and natural language processing knew major progresses during the two last decades. It is now feasible to build real Spoken Dialogue Systems (SDS) interacting with human users through voice-enabled interactions. Speech often appears as a natural way to interact for a human being and it provides potential benefits such as hand-free access to machines, ergonomics and greater efficiency of interaction. Yet, speech-based interfaces design has been an expert job for a long time. It necessitates good skills in speech technologies and low-level programming. Moreover, rapid design and reusability of previously designed systems are almost impossible. For these reasons, but not only, people are less used to interact with speech-based interfaces which are therefore thought as less intuitive.

Designing and optimizing a SDS is not only the combination of speech and language processing systems such as Automatic Speech Recognition (ASR) (Rabiner & Juang 1993), Spoken Language Understanding (SLU) (Allen 1998), Natural Language Generation (NLG) (Reiter & Dale 2000), and Text-to-Speech (TTS) synthesis (Dutoit 1997) systems. It also requires the development of dialogue strategies taking at least into account the performances of these subsystems (and others), the nature of the task (e.g. form filling (Pietquin & Dutoit 2006a), tutoring (Graesser *et al* 2001), robot control, or database querying (Pietquin 2006b)), and the user's behaviour (e.g. cooperativeness, expertise (Pietquin 2004)). The great variability of these factors makes rapid design of dialogue strategies and reusability across tasks of previous work very complex. Most often, such a design is a cyclic process composed of strategy hand-coding, prototype releases and expansive and time consuming user tests. In addition, there is also no guarantee that hand-crafted strategies developed by experts are anything close to optimal. Because it provides data-driven methods and objective clues about performances, statistical learning of optimal dialogue strategies became a leading domain of research (Lemon & Pietquin, 2007). The goal of such

Source: Reinforcement Learning: Theory and Applications, Book edited by Cornelius Weber, Mark Elshaw and Norbert Michael Mayer
ISBN 978-3-902613-14-1, pp.424, January 2008, I-Tech Education and Publishing, Vienna, Austria

approaches is to reduce the number of design cycles (Fig. 1).

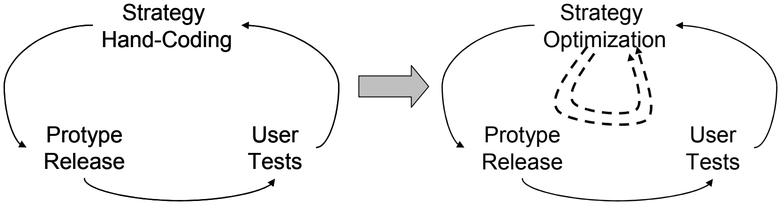


Fig. 1. Optimization for minimizing the number of design cycles

Supervised learning for such an optimization problem would require examples of ideal (sub)strategies which are typically unknown. Indeed, no one can actually provide an example of what would have objectively been the perfect sequencing of exchanges after having participated to a dialogue. Humans have a greater propensity to criticize what is wrong than to provide positive proposals. In this context, reinforcement learning using Markov Decision Processes (MDPs) (Levin *et al* 1998, Singh *et al* 1999, Scheffler & Young 2001, Pietquin & Dutoit 2006a, Frampton & Lemon 2006) and Partially Observable MDP (POMDPs) (Poupart *et al* 2005, Young 2006) has become a particular focus.

2. Formalism

2.1 Definitions

In the rest of this chapter, a *dialogue* will be referred to as an interaction between two *agents* based on sequential turn taking. In most of the cases, this interaction is *goal-directed* and both agents cooperate in order to achieve an aim (or accomplish a task). In the case of a man-machine dialog, one of the agents is a human *user* while the other is a computer (or system). In the particular case in which the interaction uses speech as the main communication mean, the computer implements a *Spoken Dialogue System* (SDS) while a system using several means of communication is referred to as a *Multimodal Dialogue System* (MMDS). When the man-machine dialog is dedicated to the realisation of a particular task (or set of tasks) it is called a *task-oriented* dialogue system. When one of the agents is an SDS, the dialogue consists of a sequence of *utterances* exchanged at each turn. A *spoken utterance* is the acoustic realisation of the *intentions* or *concepts* or *dialog acts* one of the agents wants to communicate to the other and is expressed as a *word* sequence.

2.2 Formal description of man-machine spoken dialog

A man-machine spoken dialog can be considered as a sequential process in which a human user and a Dialog Manager (DM) are communicating using speech through speech and language processing modules (Fig. 2). The role of the DM is to define the sequencing of spoken interactions and therefore to take decisions about what to do at a given time (providing information, asking for information, closing the dialog, etc.). A Spoken Dialog System is often meant to provide information to a user; this is why it is generally connected to a Knowledge Base (KB) through its DM. The dialog is therefore regarded as a turn-taking process in which pieces of information are processed sequentially by a set of modules and perform a cycle going from the DM to the user and back. At each turn $\boldsymbol{t}$ the DM generates a communicative act set $\boldsymbol{a_t}$ according to its internal state $\boldsymbol{s_t}$ and corresponding to its decision about what to do in that state. Dialogue communicative acts (shortly dialogue acts) can be of different kind such as greeting the user, ask a constraining question to the user, provide information to the user, ask for confirmation about some information to the user, query a database, close the dialogue. This act set is then transformed into a linguistic representation lt (generally a text) by a natural language processing module. The textual representation lt serves as an input to a text-to-speech synthesizer to produce a system spoken output syst. The TTS and the NLG modules are therefore spoken output generation modules. To this spoken solicitation, the user answers by a new spoken utterance ut according to what he could understand from syst, to his/her knowledge kt (about the task, the interaction history, the world in general) and to the goal gt s/he is trying to achieve by interacting with the system. Both spoken utterances syst and ut can be mixed with some background noise nt. The noisy user utterance is in turn processed by an automatic speech recognition system, which produces a written word sequence wt as a result and a confidence measure CLASR about this result. A natural language understanding module subsequently tries to extract communicative acts (or concepts) ct from wt (possibly helped by CLASR). The NLU module also provides some confidence measure CLNLU about this processing. The NLU and ASR sub-systems are speech input processing modules. The set {ct, CLASR, CLNLU} composes an observation ot which can be considered as the result of the processing of the DM communicative acts a_t by its *environment*. The dialogue manager then computes a new internal state $\boldsymbol{s_{t+1}}$ according to this observation.

The following paragraphs will use this description of a man-machine dialog as a base to build a probabilistic model.

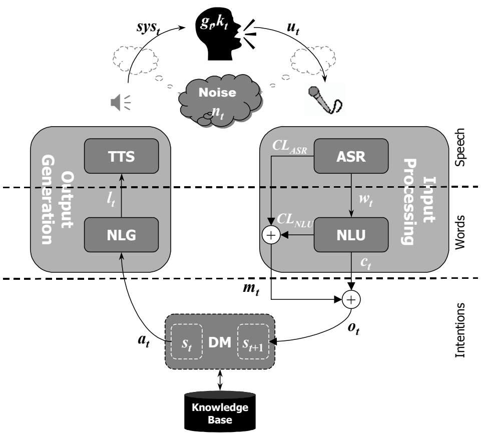


Fig. 2. Man-Machine Spoken Communication

2.3 MDP and reinforcement learning

In our vision of the MDP formalism, a discrete-time system interacting with its stochastic environment through actions is described by a finite or infinite number of states $\{s_i\}$ in which a given number of actions $\{a_j\}$ can be performed. To each state-action pair is associated a transition probability $\mathcal{T}$ giving the probability of stepping from state s at time t to state s' at time t+1 after having performed action a when in state s. To this transition is also associated a reinforcement signal (or reward) r_{t+1} describing how good was the result of action *a* when performed in state *s*. Formally, an MDP is thus completely defined by a 4-tuple {*S*, *A*, $\mathcal{T}$, $\mathcal{R}$} where *S* is the state space, *A* is the action set, $\mathcal{T}$ is a transition probability distribution over the state space and $\mathcal{R}$ is the expected reward distribution. The couple {$\mathcal{T}$, $\mathcal{R}$} defines the dynamics of the system:

$$\mathcal{T}_{ss'}^{a} = P(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{1}$$

$$\mathcal{R}_{ss'}^{a} = E[r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'] \tag{2}$$

These last expressions assume that the Markov property is met, which means that the system's functioning is fully defined by its one-step dynamics and that its behavior from state *s* will be identical whatever the path followed before reaching *s*. To control a system described as an MDP (choosing actions to perform in each state), one would need a *strategy* or *policy* π mapping states to actions: π(*s*) = P(*a* | *s*) (or π(*s*) = *a* if the strategy is deterministic). In this framework, a RL *agent* is a system aiming at optimally mapping states to actions, that is finding the best strategy π^* so as to maximize an overall return *R* which is a function (most often a discounted return is used i.e. a weighted sum of immediate rewards) of all the immediate rewards r_t.

$$R^{\pi} = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_t = \pi(s_t)\right] \tag{3}$$

$$\pi^* = \underset{\pi}{arg\,max}\left[E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_t = \pi(s_t)\right]\right] \tag{4}$$

If the probabilities of equations (1) and (2) are known, an analytical solution can be computed by dynamic programming, otherwise the system has to learn the optimal strategy by a trial-and-error process. RL is therefore about how to optimally map situations to actions by trying and observing environment's feedback. In the most challenging cases, actions may affect not only the immediate reward, but also the next situation and, through that, all subsequent rewards. Trial-and-error search and delayed rewards are the two main features of RL. Different techniques are described in the literature, in the following the Watkin's Q(λ) algorithm (Watkin 1989) will be used.

3. Human-machine dialogue and Markov decision process

From the point of view of the dialogue manager, the interaction can probabilistically be described by the joint probability of the signals a_t, o_t and s_{t+1} given the history of the interaction (Pietquin 2005):

$$\begin{aligned} P(s_{t+1}, o_t, a_t \mid s_t, n_t, a_{t-1}, s_{t-1}, n_{t-1}, \ldots, a_0, s_0, n_0) = & \underbrace{P(s_{t+1} \mid o_t, a_t, s_t, n_t, a_{t-1}, s_{t-1}, n_{t-1}, \ldots, a_0, s_0, n_0)}_{\text{Task Model}} \cdot \\ & \underbrace{P(o_t \mid a_t, s_t, n_t, a_{t-1}, s_{t-1}, n_{t-1}, \ldots, a_0, s_0, n_0)}_{\text{Environment}} \cdot \underbrace{P(a_t \mid s_t, n_t, a_{t-1}, s_{t-1}, n_{t-1}, \ldots, a_0, s_0, n_0)}_{\text{DM}} \end{aligned} \tag{5}$$

In (5), the first term stands for the *task model* that helps building a new dialogue manager internal state thanks to the perceived observation, the second term stands for the response of the environment to the dialogue manager stimulation, and the third stands for the dialogue manager decision process.

3.1 Markov property and random noise

In the case of a SDS, the Markov Property is met if the dialogue manager choice about the action a_t to perform at time t and the according transition probability for stepping to state s_{t+1} at time t+1 are only conditioned by the state s_t at time t and not of previous states and actions. From now on, the Markov Property will be assumed. It can anyway be met by a judicious choice of the DM state representation, which should embed the history of the interaction into the current state. Such a state representation is said *informational*.

Let's illustrate this on a train ticket booking system. When accessing such a system a customer can book a ticket by providing information about the cities of departure and arrival and a desired time of departure. Like in a 3-slot-filling application, three bits of information (sometimes called *attributes*) should therefore be transferred from the user to the system. A very simple way to build the state space is to represent the dialogue state as a vector of three Boolean values (e.g. [dep arr time]) set to *true* if the corresponding attribute is supposed to be known by the system and to *false* otherwise. An ideal dialogue for such an application and the corresponding dialogue state evolution is shown in Table 1.

Speaker	Spoken Utterance	Dialogue state
System	Hello, how may I help you?	[*false false false*]
User	I'd like to go to Edinburgh.	
System	What's your departure city?	[*false true false*]
User	I want to leave from Glasgow.	
System	When do you want to go from Glasgow to Edinburgh ?	[*true true false*]
User	On Saturday morning.	
System	Ok, seats are available in train n° xxx …	[*true true true*]

Table 1. Ideal dialogue in a train ticket booking application

To meet the Markov property with such a state representation, we have to assume that the system behaves the same whatever the order in which the slots where filled (and by the way, whatever the values of the attributes). The Markov assumption is also made about the environment; that is the user behaves the same whatever the filling order as well. These are of course strong assumptions but we will see later that they lead to satisfactory results.

Finally, most often the noise is considered as being random so as to have independence between n_t and $n_{t\text{-}1}$. Eq. (5) then simplifies as follow:

$$P(s_{t+1}, o_t, a_t \mid s_t, n_t) = \underbrace{P(s_{t+1} \mid o_t, a_t, s_t, n_t)}_{\text{Task Model}} \cdot \underbrace{P(o_t \mid a_t, s_t, n_t)}_{\text{Environment.}} \cdot \underbrace{P(a_t \mid s_t, n_t)}_{\text{DM}} \quad (6)$$

3.2 Dialogue management as an MDP

As claimed in paragraph 0 and depicted on Fig. 2, a task-oriented (or goal-directed) man-machine dialogue can be seen as a turn-taking process in which a human user and a dialogue manager exchange information through different channels processing speech inputs and outputs (ASR, TTS ...). In our problem, the dialogue manager's action (or dialogue act) selection *strategy* has to be optimized, the dialogue manager should thus be our learning *agent.*

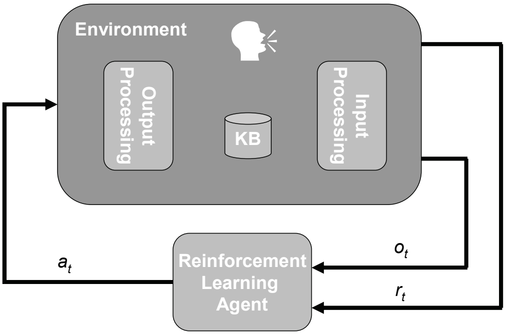


Fig. 3. Dialogue management as an MDP

As shown on Fig. 3, the *environment* modeled by the MDP comprises everything but the dialogue manager, i.e. the human user, the communication channels (ASR, TTS …), and any external information source (database, sensors etc.). In this context, at each turn t the dialogue manager has to choose an *action* a_t according to its interaction *strategy* so as to complete the task it has been designed for. The RL agent has therefore to choose an action among greetings, spoken utterances (constraining questions, confirmations, relaxation, data presentation etc.), database queries, dialogue closure etc. They result in a response from the DM environment (user speech input, database records etc.), considered as an observation o_t, which usually leads to a DM *internal state* update according to the task model (Eq. 6).

3.3 Reward function

To fit totally to the MDP formalism, a *reinforcement signal* r_t is required. In (Singh *et al* 1999) it is proposed to use the contribution of an action to the user's satisfaction. Although this seems very subjective, some studies have shown that such a reward could be approximated by a linear combination of the task completion (TC) and objective measures c_i related to the system performances (Walker *et al* 1997):

$$r_t = \alpha \cdot \mathcal{N}(TC) - \sum_i w_i \cdot \mathcal{N}(c_i), \tag{7}$$

where N is a Z-score normalization function that normalises the results to have mean 0 and standard deviation 1 and w_i are non-zero weights. This way, each weight (α and w_i) expresses the relative importance of each term of the sum in the performance of the system. The task completion can for example be measured by the kappa (κ) coefficient (Carletta 1996):

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}, \tag{8}$$

where $P(A)$ is the proportion of correct interpretations of user's utterances by the system and P(E) is the proportion of correct interpretations occurring by chance. One can see that κ

= 1 when the system performs perfect interpretation (P(A) = 1) and κ = 0 when the only correct interpretations were obtained by chance (P(A) = P(E)). In order to compute weights α and wi, a large number of users are asked to answer a satisfaction survey after having used the system while costs ci are measured during the interaction. The questionnaire comprises around 9 statements on a five-point Likert scale and the overall satisfaction is computed as the mean value of collected ratings. A Multivariate Linear Regression is then applied using the results of the survey as the dependent variable and the weights as independent variables. In practice, the significant performance measures ci are mainly the duration of the dialogue and the ASR and NLU performances.

3.4 Partial observability

If a direct mapping between observations and system (or dialogue) states exists, the process is completely observable and the task model (see Eq. 6) can easily be built. Yet, it is rarely the case that the observations are directly linked to the dialogue state. Indeed, the real dialogue state at time *t* is related to the information the user intended to transmit to the system until time *t* during the interaction. This information being transmitted through error prone statistical speech recognition and understanding systems, it can occur that the observation doesn't contain only the information meant by the user but a probability distribution over a set of possible bits of information. Indeed, the output of a speech recognition system is usually a list of N word sequences (named N-bests list), each of them being associated with a confidence level that can be considered as a probability of the word sequence being correct given the spoken utterance (and maybe the context). This N-bests list serves as an input to the natural language understanding module which in turn provides a list of concept sequences associated to confidence levels.

This is typically what happens in partially observable environments where a probability distribution is drawn over possible states given the observations. For this reason, emerging research is focused on the optimization of spoken dialogue systems in the framework of Partially Observable Markov Decision Processes (POMDPs) (Poupart *et al* 2005, Young 2006)

4. Learning dialogue policies using simulation

Using the framework described previously, it is theoretically possible to automatically learn spoken dialogue policies allowing natural conversation between human users and computers. This learning process should be realised online, through real interactions with users. One could even imagine building the reinforcement signal from direct queries to the user about his/her satisfaction after each interaction (Fig. 4).

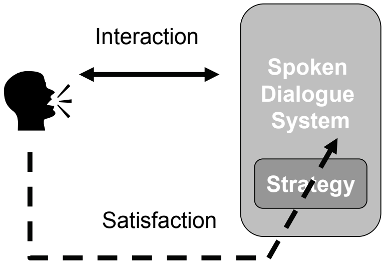


Fig. 4. Ideal learning process

For several reasons, direct learning through interactions is made difficult. First, a human user would probably react badly to some of the exploratory actions the system would choose since they might be completely incoherent. Anyway a very large number of interactions are required (typically tens of thousands of dialogues for standard dialogue systems) to train such a system. This is why data driven learning as been proposed so as to take advantage of existing databases for bootstrapping the learning process. Two methods were initially investigated: learning the state transition probabilities and the reward distribution from data (Singh *et al*, 1999) or learning parameters of a simulation environment mainly reproducing the behaviour of the user (Levin *et al* 2000). The second method is today preferred (Fig. 5). Indeed, whatever the data set available, it is unlikely that it contains every possible state transitions and it allows exploring the entire state and policy space. Dialogue simulation is therefore necessary for expanding the existing data sets and learning optimal policies. Most often, the dialogue is simulated at the intention level rather than at the word sequence or speech signal level, as it would be in the real world. An exception can be found in (Lopez Cozar *et al* 2003). Here, 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 considered as 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 (Pietquin & Renals 2002, Pietquin & Dutoit 2006b). 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 while it should not influence the dialogue manager strategy. Table 2 describes such a simulation process. The intentions have been expanded in the last column for comprehensiveness purposes. The signals column refers to notations of section 0.

Signals	Intentions	Expanded Intentions
sys_0	greeting	*Hello! How may I help you?*
u_0	arr_city = ‘Paris’	I’d like to go to Paris.
sys_1	const(arr_time)	*When do you prefer to arrive?*
u_1	arr_time = ‘1.00 PM’	I want to arrive around 1 PM.
sys_2	rel(arr_time)	*Don’t you prefer to arrive later?*
u_2	rel = *false*	No.
sys_3	conf(arr_city)	*Can you confirm you want to go to Paris?*
u_3	conf = *true*	Yes !
…	…	…
…	…	…

Table 2. Simulated dialogue at the intention level (‘const’ stands for constraining question, ‘rel’ for relaxation and ‘conf’ for confirmation)

This approach requires modelling the environment of the dialogue manager as a stochastic system and to learn the parameters of this model from data. It has been a topic of research since the early 2000’s (Levin et al 2000, Scheffler & Young 2001, Pietquin 2004). Most of the research is now focused on simulating the user (Georgila et al 2005, Pietquin 2006a, Schatzmann et al 2007a) and assessing the quality of a user model for training a reinforcement learning agent is an important track (Schatzmann et al 2005, Rieser & Lemon 2006, Georgila et al 2006). Modelling the errors introduced by the ASR and NLU systems is also a major topic of research (Scheffler & Young 2001, Lopez Cozar et al 2003, Pietquin & Beaufort 2005, Pietquin & Dutoit 2006b).

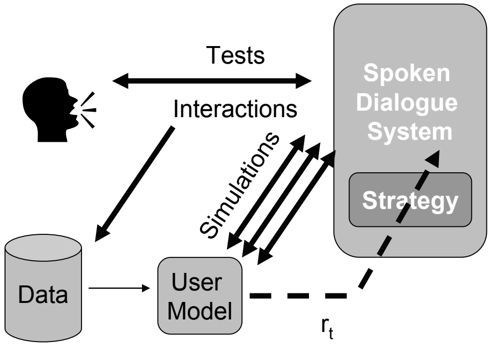


Fig. 5. Learning via simulation

5. Speech-based database querying

We will illustrate reinforcement learning based dialogue optimization on the particular task of a speech-based database querying system. In such an application, the user wants to extract from a database one or a list of records selected according to specific features provided by a user through speech-based interactions.

In the following, several experiments made on a database containing 350 computer configurations are described. The database is split into 2 tables (for notebooks and desktops), each of them containing 6 fields: pc_mac (pc or mac), processor_type, processor_speed, ram_size, hdd_size and brand. The goal of the dialogue system is therefore to extract one or a short list of computer configurations from the database and to present it the user. To do so, the system will have to gather information about which computer features the user wants. In the following, the application is described in terms of actions, states and rewards so as to be mapped to the Markov decision processes paradigm.

5.1 Action set

The task involves database querying. Therefore possible systems actions do not only imply interactions with the user (such as spoken questions, confirmation requests or assertions) but also with the database (such as database querying). The action set contains 8 generic actions:

- greet: greeting (e.g. “How may I help you ?”).
- constQ(*arg*): ask to constrain the value of *arg*.
- openQ: ask an open ended question.
- expC(*arg*): ask to confirm the value of *arg*.
- allC: ask for a confirmation of all the arguments.
- rel(*arg*): ask to relax the value of *arg*.
- dbQ([*args*]): perform a database query thanks to retrieved information.
- close: present data and close the dialogue session.

The value of *arg* may be the table's type (notebook or desktop) or one of the 6 table fields. Notice that there is no data presentation action because it will be considered that the data presentation is included in the 'close' action. This means that, when the dialogue is closed by the dialogue manager, it systematically presents to the user the content of the last retrieved recordset.

5.2 State space

The way the state space is built is very important and several state variables can be envisioned for describing the same task. Yet, some general considerations might be taken into account:

1. The state representation should contain enough information about the history of the dialogue so as to assume the Markov property to be met.
2. State spaces are often considered as informational in that sense that they are built thanks to the amount of information the DM could retrieve from the environment until it reached the current state.
3. The state representation must embed enough information so as to give an accurate representation of the situation to which an action has to be associated (it is not as obvious as it sounds).
4. The state space must be kept as small as possible since the reinforcement learning algorithms converge in linear time with the number of states of the underlying Markov decision process.

According to these considerations and the particular task of database querying, two slightly different state spaces where built to describe the task as an MDP so as to illustrate the sensitivity of the method to the state space representation. In the first representation, referred to as S_1 in the following, each state is represented by two features.

- A vector of 7 boolean values (one for each value of *arg*). Like in the example of paragraph 0, each of these values is set to *true* if the corresponding value of *arg* is known (for example if the user specified to search in the notebooks table, the fist value is set to *true*). This is a way to meet the Markov property (informational state).
- Information about the Confidence Level (CL) of each value set to *true*. The confidence level is usually a real number ranging between 0 and 1 computed by the speech and/or language analysis subsystems (ASR and NLU) and providing information about the confidence of the system in the result of its processing. To keep the size of the state space reasonable, we only considered 2 possible values for the confidence level: *High* or *Low* (i.e. *High* means CL ≥ 0.8 and *Low* means CL < 0.8).

Notice that 'dbQ' 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: {value = *false*, *CL* = *undef*}, {value = *true*, *CL* = *Low*}, {value = *true*, *CL* = *High*}. This leads to 3^7 possible states.

The second way to represent the state space is built on the same basis but an additional state variable *NDB* is added to take the number of records returned by the last database query into account. This variable can also take only two values (*High* or *Low*) and is set according to the comparison of the query result size and a predefined threshold. If no 'dbQ' action has been performed, the *NDB* variable is initialized with the *High* value (an empty query would provide the whole database as a result). This state space representation will be referred to as S_2 in the following.

5.3 Reward function

Once again, there is not a unique way to build the reward function and slight differences in the choices can result in large variations in the learned strategy. To illustrate this, some simple functions will be described in the following. According to (Walker *et al*, 1997), the reward function (which is here a cost function that we will try to minimize) should rely on an estimate of the dialogue time duration (*D*), the ASR performances (*ASR*) and the task completion (*TC*) so as to approximate the user's satisfaction using objective measures:

$$R = w_D \cdot D - w_{ASR} \cdot ASR - w_{TC} \cdot TC \qquad (9)$$

In this last expression, the w_x factors are positive weights. Considering the estimate of the time duration, two values are actually available: the number of user turns $D = N_U$ (the number of turns perceived by the user) and the number of system turns $D = N_S$ (including database queries as well).

On another hand, the task completion is not always easy to define. The *kappa* coefficient defined in (Carletta 1996) and section 0 is one possibility but didn't always prove to correlate well with the perceived task completion. For the purpose of this experiment, two simple task completion measures will be defined:

$$TC_{max} = \max_{R_i}\left(\#\left(G_U \cap R_i\right)\right) \qquad (10)$$

$$TC_{av} = average\left(\#\left(G_U \cap R_i\right)\right) \qquad (11)$$

In these last expressions, $\#(G_U \cap R)$ is the number of common values in the user's goal G_U (the user goal is supposed to have the same structure as an existing database record and is set before the dialogue begins) and one of the records *R* presented to the user at the end of a dialogue. When a value is not present in the user goal it is considered as common (if a field is not important to the user, it is supposed to match any value). The first task completion measure TC_{max} indicates how close the closest record in the presented results is. The second TC_{av} measures the mean number of common values between the user's goal and each presented record.

Finally, the ASR performance measures will be provided by the confidence levels (*CL*) computed by the ASR system after each speech recognition task.

6. Experiments

The number of required interactions between a RL agent and its environment is quite large (10^4 dialogues at least in our case). So, it has been mandatory to simulate most of the dialogues for reasons explained in section 0. An intention-based simulation environment has therefore been built as described in (Pietquin & Dutoit 2006a). It simulates ASR errors using a constant Word Error Rate (WER). It generates confidence levels as real numbers ranging between 0 and 1 according to a distribution measured on a real system. If the system has to recognize more than one argument at a time, the *CL* is the product of individual *CL*s obtained for each recognition task (so it usually decreases). Other ASR simulation models can be considered (Pietquin & Beaufort 2005) but it is out of the scope of this introduction to the technique.

Several experimental results obtained with different settings of the state space and the reward function will be exposed in the following. These settings are obtained by combining in three different ways the parameters S_1, S_2, N_U, N_S, TC_{max}, TC_{av} mentioned before. Results are described in terms of average number of turns (user and system turns), average task completion measures (TC_{max} and TC_{av}) for the performance and action occurrence frequency during a dialogue session to get a clue about the learned strategy. These results are obtained by simulating 10,000 dialogues with the learned strategy.

6.1 First experiment: S_1, N_U, TC_{max}

The first experiment is based on the smaller state space S_1 (without any information about the number of retrieved records). The dialogue cost is computed thanks to the number of user turns N_U as a measure of the time duration and the TC_{max} value as the task completion measure. Results are shown in the following tables.

N_U	N_S	TC_{max}	TC_{av}
2.25	3.35	6.7	1.2

Table 3. Performances of the learned strategy for the {S_1, N_U, TC_{max}} configuration

greet	constQ	openQ	expC	AllC	rel	dbQ	close
1.00	0.06	0.0	0.14	0.0	0.05	1.10	1.00

Table 4. Learned strategy for the {S_1, N_U, TC_{max}} configuration

When looking at the three first columns of the performance table (Table 4), the learned strategy doesn't look so bad. It actually has a short duration in terms of user turns as well as in system turns and has a very high task completion rate in terms of TC_{max} measure. Yet the TC_{av} shows a very low mean value.

When looking to the average frequency of actions in table, one can see that the only action addressed to the user that happens frequently during a dialogue is the greeting action. Others almost never occur. Actually, the learned strategy consists in uttering the greeting prompt to which the user should answer by providing some arguments. Then the system performs a database query with the retrieved attributes and presents the results to the user. Sometimes, the user doesn't provide any attribute when answering to the greeting prompt or the value is not recognized at all by the ASR model (very low CL value), so the strategy is to perform a constraining question (and not an open-ended question) that will provide an argument with a better CL. Sometimes the provided arguments have still a poor CL and an explicit confirmation is requested. Sometimes the provided arguments don't correspond to any valid record in the database so the strategy is to ask for relaxation of one argument (this also explains why the number of database queries is greater than 1). The value of TC_{max} is not maximal because sometimes the dialogue fails.

This results in presenting almost the whole database when the user only provides one argument when prompted by the greeting. This is why there is a so big difference between TC_{max} and TC_{av}. The desired record is actually in the presented data (TC_{max} is high) but is very difficult to find (TC_{av} is low). The learned strategy is definitely not suitable for a real system, specially if the record set have to be presented vocally. An example of dialogue is shown in Table 5, where the signal column refers to signals used on Fig. 2 and in section 0.

Signals	Intentions	Expanded Intentions
$a_0 \rightarrow sys_0$	greeting	*Hello! How may I help you?*
u_0	Table= 'Notebook'	I'd like to buy a Notebook.

o_0	Table = 'Notebook'
	CL = *high*	

a_1	dbQ	
o_1	DB = 97 (*high*)	
$a_2 \rightarrow sys_2$	As = close	*Ok, here are the computers corresponding to your request: (proposes the 97 Notebooks in the DB) …*

Table 5. Dialogue sample for the {S_1, N_U, TC_{max}} configuration

6.2 Second experiment: S_2, N_U, TC_{av}

Here, the same settings are used except that the *NDB* variable is added to the state variables and the task completion is measured with TC_{av}.

N_U	N_S	TC_{max}	TC_{av}
5.75	8.88	6.7	6.2

Table 6. Performances of the learned strategy for the {S_2, N_U, TC_{av}} configuration

greet	constQ	openQ	expC	AllC	rel	dbQ	close
1.00	0.87	1.24	0.31	1.12	0.21	3.13	1.00

Table 7. Learned strategy for the {S_2, N_U, TC_{av}} configuration

Results shows that TC_{max} and TC_{av} are close to each other, so the presented results are more accurate but the number of turns has increased. The number of system turns particularly exhibits higher values. This is obviously explained by the increase of database queries. Looking at Table 7 one can see that the learned strategy tries to maximize the TC_{av} value while minimizing the number of user turns and maximizing recognition performance. To do so, it systematically performs a database query after having retrieved information from the user. The number of results being among the state variables, the agent learned not to present the results when in a state with a high *NDB* value. If this value is too high after the greeting, the learner tries to reach a state where it is lower. Thus it almost systematically performs an 'openQ' action after the greeting so as to get as much information as possible in a minimum of turns (this explains the 1.24 value). Yet, this often results in lower CL values, thus it also performs a confirmation of all the fields before presenting any result. Sometimes, more information is provided after the greeting and only a constraining question is required to gather enough information to reach a state with less result. A constraining question is preferred in this case because it leads to better recognition results.

The mean number of user turns shows that only 5.75 turns are usually needed to reach an accurate result set because the computer configurations are sufficiently different so as not to need too much attributes in the database query to provided accurate results. Thus, the system doesn't ask for all the attribute values to the user. Further investigations would show that the system takes advantage of the structure of the database and asks for attributes allowing extracting the desired records as fast as possible.

Signals	Intentions	Expanded Intentions
$a_0 \rightarrow sys_0$	greeting	*Hello! How may I help you?*
u_0	Table = 'Notebook'	I'd like to buy a Notebook.

o_0	Table = 'Notebook'
	CL = *high*	

a_1	dbQ	
o_1	DB = *high*	
$a_2 \rightarrow sys_2$	openQ	*Do you have any other preference?*

u_2	pc_mac = 'PC'
	proc_type = 'Pentium III'
		I'd rather like a PC with a Pentium III processor.

o_2	pc_mac = 'PC'
	proc_type = 'Pentium III'
	CL = *high*	

a_3	dbQ	
o_3	DB = *high*	
$a_4 \rightarrow sys_4$	constQ(ram)	*How much memory would you like?*
u_4	ram = 128	128 MB.
a_5	dbQ	
o_5	DB = *low*	
$a_6 \rightarrow sys_6$	allC	*You asked for a PC Notebook with a Pentium III processor and 128 MB memory.*
u_6	conf_table = *true* …	Yes.
$a_7 \rightarrow sys_7$	close	*Ok, here are the computers corresponding to your request: (proposes the 3 results of the DB query) …*

Table 8. Dialogue sample for the {S_2, N_U, TC_{av}} configuration

6.3 Third experiment: S_2, N_S, TC_{av}

The same experiment as the previous one has been performed but replacing the N_U measure of time duration by the N_S measure. It actually makes sense since in a real application, the database could be much larger than the one used here. Thus, the database queries could be much more time consuming.

N_U	N_S	TC_{max}	TC_{av}
6.77	7.99	6.6	6.1

Table 9. Performances of the learned strategy for the {S_2, N_S, TC_{av}} configuration

greet	**constQ**	**openQ**	**expC**	**AllC**	**rel**	**dbQ**	**close**
1.00	1.57	1.24	0.33	1.32	0.31	1.22	1.00

Table 10. Learned strategy for the {S_2, N_S, TC_{av}} configuration

This obviously results in a decrease of the number of database queries involving a proportional decrease of the number of system turns N$_S$. Yet, an increase of the number of user turns N_U is also observed. By examining the action frequencies, one can notice that the number of constraining questions increased resulting in an increase of N_U. Indeed, the learned strategy implies gathering enough information from the user before performing a database query. This explains why the systems ask more constraining questions.

This last strategy is actually optimal for the considered simulation environment (constant word error rate for all tasks) and is suitable for using with this simple application.

7. Conclusion

This chapter described a formal description of a man-machine spoken dialogue suitable to introduce a mapping between man-machine dialogues and (partially observable) Markov decision processes. This allows data-driven optimization of a dialogue manager's interaction strategy using the reinforcement learning paradigm. Yet, such an optimization process often requires tenths of thousands of dialogues which are not accessible through real interactions with human users because of time and economical constraints. Expanding existing databases by means of dialogue simulation is a solution to this problem and several approaches can be envisioned as discussed in section 0. In this context, we described the particular task of speech-based database querying and its mapping into the MDP paradigm in terms of actions, states and rewards. Three experiments on a very simple task have shown the influence of parameterization of the MDP on the learned conversational strategy. From this, one can say first that the state space representation is a crucial point since it embeds the knowledge of the system about the interaction. Second, the reward function is also of major importance since it measures how well the system performs on the task by simulating the perception of the dialogue quality from the users' point of view. Performance measure is a key of RL. The three experiments described in the last section showed the influence of these parameters on the learned strategy and concluded that a correctly parameterized RL algorithm could result in an acceptable dialogue strategy while little changes in the parameters could lead to silly strategies unsuitable for use in real conditions.

8. Future works

Data-driven optimization of spoken dialogue strategies is an emerging area of research and lots of problems still remain. One of the first is to find tractable algorithms to train real size dialogue systems. Indeed, the standard RL algorithms are suitable for small tasks as described in section 0 but real applications can exhibit up to several million of states, possibly with continuous observations (Williams *et al* 2005). The curse of dimensionality is therefore of particular interest in the area of spoken dialogue systems. Several attempts to tackle this problem in the framework of spoken dialogue systems can be found in the literature by scaling up MDPs using supervised learning (Henderson *et al* 2005) and hierarchical learning (Cuayáhuitl *et al* 2007). Also algorithms for tractable solutions to the optimization of spoken dialogue systems via the POMDP paradigm can be found in (Poupart *et al* 2005, Young 2006). This preliminary work in the field of generalisation and hierarchical learning shows the interest of the community in these techniques. Another problem to tackle is the development of realistic user models, easily trainable from data and suitable for training RL-based dialogue managers. Different approaches are being studied such as the recently proposed agenda-based user model (Schatzmann *et al* 2007b) that can be trained by an Expectation-Maximisation algorithm from data, or user models based on dynamic Bayesian networks (Pietquin & Dutoit 2006a). Assessing such user models in terms of quality of the trained strategies and similarity with real user behavior is of course primordial (Schatzmann *et al* 2005, Georgila *et al* 2006, Rieser & Lemon 2006). On another hand, it might be interesting to see how to use learned strategies to help human developers to design optimal strategies. Indeed, the solution may be in computer-aided design more than fully automated design (Pietquin & Dutoit 2003). Finally, designing a complete dialogue system using an end-to-end probabilistic framework, from speech recognition to speech synthesis systems automatically trained on real data, is probably the next step (Lemon & Pietquin 2007).

9. Acknowledgement

The research presented in this chapter has been funded by the 'First Europe' program of the Belgian Walloon Region, the SIMILAR European Network of Excellence and the French Lorraine Region.

10. References

Allen, J. (1994) *Natural Language Understanding*, Benjamin Cummings, 1987, Second Edition, 1994.

Carletta J. (1996), Assessing Agreement on Classification Tasks: the Kappa Statistic. *Computational Linguistics*, 22(2), 1996, 249-254.

Cuayáhuitl, H.; Renals, S.; Lemon, O. and Shimodaira, H. (2007) Hierarchical Dialogue Optimization Using Semi-Markov Decision Processes, in *Proceedings of International Conference on Speech Communication (Interspeech'07)*, Anvers (Belgium), 2007.

Dutoit, T., *An Introduction to Text-To-Speech Synthesis*. Kluwer Academic Publishers, Dordrecht, ISBN 0-7923-4498-7, 1997.

Frampton, M. & Lemon O. (2006). Learning more effective dialogue strategies using limited dialogue move features, in *Proceedings of ACM*, 2006.

Georgila, K.; Henderson, J. and Lemon, O. (2005). Learning User Simulations for Information State Update Dialogue Systems, in *Proceedings of International Conference on Speech Communication (Interspeech'05)*, Lisbon (Portugal) 2005.

Georgila, K.; Henderson, J. and Lemon, O. (2006) User simulation for spoken dialogue systems: Learning and evaluation, in *Proceedings of International Conference on Speech Communication (Interspeech'06)*, Pittsburgh, 2006.

Graesser, A.; VanLehn, K.; Rosé, C.; Jordan, P. & Harter, D. (2001) Intelligent Tutoring Systems with Conversational Dialogue. in *AI Magazine* vol. 22(4) , 2001, pp. 39-52.

Henderson, J.; Lemon, O. and Georgila, K. (2005) Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data, in *Proceedings of the IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2005, pp. 68–75.

Lemon, O. & Pietquin, O. (2007). Machine learning for spoken dialogue systems, in *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, Anvers (Belgium), August 2007.

Levin, E.; Pieraccini, R. & Eckert, W. (1997). Learning dialogue strategies within the Markov decision process framework, in *Proceedings of the International Workshop on Automatic Speech Recognition and Understanding* (ASRU'97), December 1997.

Levin, E.; Pieraccini, R. and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies, in *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2000.

Lopez-Cozar, R.; de la Torre, A.; Segura, J. and Rubio, A. (2003) Assesment of dialogue systems by means of a new simulation technique, in *Speech Communication*, vol. 40, no. 3, pp. 387–407, May 2003.

Pietquin, O. and Renals, S. (2002). Asr system modelling for automatic evaluation and optimization of dialogue systems, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'02)*, Orlando, (USA, FL), May 2002.

Pietquin, O. and Dutoit, T. (2003). Aided Design of Finite-State Dialogue Management Systems, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2003)*, Baltimore (USA, MA), 2003.

Pietquin, O. (2004). *A Framework for Unsupervised Learning of Dialogue Strategies*, Presses Universitaires de Louvain, ISBN : 2-930344-63-6, 2004.

Pietquin, O. (2005). A probabilistic description of man-machine spoken communication, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'05)*, Amsterdam (The Netherlands), July 2005.

Pietquin, O., Beaufort, R. (2005). Comparing ASR Modeling Methods for Spoken Dialogue Simulation and Optimal Strategy Learning. In *Proceedings of Interspeech/Eurospeech 2005*, Lisbon, Portugal (2005)

Pietquin, O. (2006a) Consistent goal-directed user model for realistic man-machine task-oriented spoken dialogue simulation, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'06)*, Toronto, Canada, July 2006.

Pietquin, O. (2006b). Machine learning for spoken dialogue management : an experiment with speech-based database querying, in *Artificial Intelligence : Methodology, Systems and Applications*, J. Euzenat and J. Domingue, Eds., vol. 4183 of Lecture Notes in Artificial Intelligence, pp. 172–180. Springer Verlag, 2006.

Pietquin, O. & Dutoit, T. (2006a). A probabilistic framework for dialog simulation and optimal strategy learning, in *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 2, pp. 589–599, March 2006.

Pietquin, O. and Dutoit, T. (2006b). Dynamic Bayesian networks for NLU simulation with applications to dialog optimal strategy learning, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'06)*, May 2006.

Poupart, P.; Williams, J. & Young, S. (2006). Partially observable Markov decision processes with continuous observations for dialogue management, in *Proceedings of the SigDial Workshop (SigDial'06)*, 2006.

Rabiner, L. & Juang, B.H. (1993). *Fundamentals of Speech Recognition*, Prentice Hall, Signal Processing Series, 1993.

Reiter, E. & Dale, R. (2000) *Building Natural Language Generation Systems*, Cambridge University Press, Cambridge, 2000.

Rieser, V. and Lemon, O. (2006) Cluster-based user simulations for learning dialogue strategies and the super evaluation metric, in *Proceedings of Interspeech/ICSLP*, 2006.

Schatzmann, J.; Georgila, K. and Young, S. (2005) Quantitative evaluation of user simulation techniques for spoken dialogue systems, in *Proceedings of the SIGdial'05 Workshop*, September 2005.

Schatzmann, J.; Weilhammer, K.; Stuttle, M. and Young, S. (2007a) A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies, in *Knowledge Engineering Review* 21(2): 97-126, 2007.

Schatzmann, J.; Thomson, B. and Young., S. (2007b). Statistical User Simulation with a Hidden Agenda. In *Proceedings of the 8th SigDIAL Workshop*, Antwerp, 2007.

Scheffler, K. & Young, S. (2001). Corpus-based dialogue simulation for automatic strategy learning and evaluation, in *Proc. NAACL Workshop on Adaptation in Dialogue Systems*, 2001.

Singh, S.; Kearns, M.; Litman, D. & Walker, M. (1999), Reinforcement learning for spoken dialogue systems, in *Proceedings of NIPS'99*, 1999.

Young, S. (2006). Using POMDPs for dialog management, in *Proceedings of the 1st IEEE/ACL Workshop on Spoken Language Technologies (SLT'06)*, 2006.

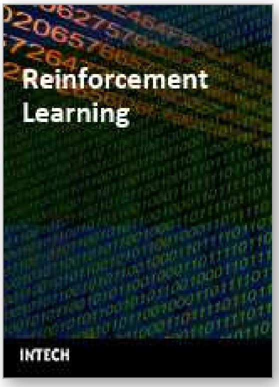
Young, S.; Schatzmann, J.; Weilhammer, K. & Ye, H. (2007). The hidden information state approach to dialog management, in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'07)*, April 2007.

Walker, M.; Litman, D.; Kamm, C. & Abella, A. (1997). PARADISE: A Framework for Evaluating Spoken Dialogue Agents. in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain (1997) 271-280.

Watkins, C. (1989). *Learning from delayed rewards*. PhD thesis, Psychology Department, Cambridge University, Cambridge, England, 1989.

Williams, J. & Young, S. (2005). Scaling up POMDPs for dialogue management: the summary POMDP method, in *Proceedings of the IEEE workshop on Automatic Speech Recognition and Understanding (ASRU'05)*, 2005.

Williams, J.; Poupart, P. and Young, S. (2005). Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management, in *Proceedings of the 6th SigDial Workshop*, Lisbon (Portugal), 2005.



Reinforcement Learning

Edited by Cornelius Weber, Mark Elshaw and Norbert Michael Mayer

ISBN 978-3-902613-14-1

Hard cover, 424 pages

Publisher I-Tech Education and Publishing

Published online 01, January, 2008

Published in print edition January, 2008

Brains rule the world, and brain-like computation is increasingly used in computers and electronic devices. Brain-like computation is about processing and interpreting data or directly putting forward and performing actions. Learning is a very important aspect. This book is on reinforcement learning which involves performing actions to achieve a goal. The first 11 chapters of this book describe and extend the scope of reinforcement learning. The remaining 11 chapters show that there is already wide usage in numerous fields. Reinforcement learning can tackle control tasks that are too complex for traditional, hand-designed, non-learning controllers. As learning computers can deal with technical complexities, the tasks of human operators remain to specify goals on increasingly higher levels. This book shows that reinforcement learning is a very dynamic area in terms of theory and applications and it shall stimulate and encourage new research in this field.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821