

## Deliverable 4.2

### Decision Theory in Dialogue

Authors:	Stephen Pulman Oxford University Computing Laboratory sgp@clg.ox.ac.uk	Ramon Granell Oxford University Computing Laboratory ramon.granell@comlab.ox.ac.uk	Nigel Crook Oxford University Computing Laboratory nigc@comlab.ox.ac.uk
----------	---	---	--

---

Work-package: WP4.2 Integration of decision theoretical calculations into a Functionally Adequate Platform for a COMPANION

Type: Technical Report

Distribution: Public

Status: Final

Date: 29.10.2008

Deliverable Coordinator: Stephen Pulman, Oxford University Computing Laboratory

Reviewers:

Area Coordinator: Stephen Pulman, Oxford University Computing Laboratory

Project Coordinator: Yorick Wilks, University of Sheffield

EU Project Officer: Michel Brochard



## Contents

<b>1</b>	<b>Decision Networks: An Overview</b>	<b>5</b>
1.1	Graphical Models . . . . .	5
1.2	Bayesian Networks . . . . .	7
1.3	Reinforcement Learning . . . . .	8
1.4	Markov Decision Processes . . . . .	9
1.5	Partially Observable Markov Decision Processes . . . . .	10
<b>2</b>	<b>Dialogue simulation using probabilistic models</b>	<b>12</b>
2.1	Introduction . . . . .	12
2.1.1	Motivation . . . . .	12
2.1.2	Related work . . . . .	13
2.2	Knowledge representation in dialogue . . . . .	14
2.3	Probabilistic Models . . . . .	15
2.3.1	N-Grams using unsegmented turns . . . . .	15
2.3.2	N-Grams using segmented turns . . . . .	15
2.4	Corpus . . . . .	16
2.5	Experiments . . . . .	17
2.5.1	Generation experiments . . . . .	17
2.5.2	Unsegmented turn experiments . . . . .	20
2.5.3	Segmented turn experiments . . . . .	20
2.5.4	System turn prediction . . . . .	22
2.5.5	Some implementation details . . . . .	24
2.6	Discussion . . . . .	25
2.7	Future work . . . . .	25
<b>3</b>	<b>MDP</b>	<b>26</b>
3.1	Definition . . . . .	26
3.2	Q-learning algorithm . . . . .	27
<b>4</b>	<b>SDS-MDP</b>	<b>27</b>
4.1	Previous work . . . . .	28
<b>5</b>	<b>A DS-MDP for a train information task</b>	<b>28</b>
5.1	Components . . . . .	28
5.2	Implementation of Q-learning algorithm . . . . .	29
5.3	Experiments . . . . .	30
5.4	Discussion and future work . . . . .	31

<b>6</b>	<b>Integrating Decision Theoretic Mechanisms into the Senior Companion Demonstrator</b>	<b>31</b>
6.1	Dialogue Action Frames (DAFS)	32
6.2	The Integration of Decision Theoretic Mechanisms in the DAF Dialogue Manager	34

---

### ABSTRACT

Any dialogue consists (usually) of a sequence of turns taken by the participants, in our case, a user and a system. For each participant, when beginning the dialogue or responding to the previous turn, one question that has to be answered is ‘what do I say next?’. Factors influencing this decision include the overall purpose of the dialogue, the current state of the dialogue, the form and content of the previous utterances, and the participant’s immediate and future intentions. Different approaches to dialogue treat the decision problem in different ways: information state-based theories regard the decision as being largely if not completely determined by the effect of the previous utterance on the assumed information states of the participants. More recent statistically based models regard the decision as arising out of a complex interaction between a probability distribution over information or belief states, and a ‘policy’ resulting from some learning method which tries to work out from previous experience what the optimal decision is.

This deliverable has the aim of specifying how to incorporate decision theoretic models into different types of dialogue systems. We begin with a review of the relevant approaches to decision theory in dialogue. Many of these require training on large numbers of dialogues from the intended application domain. Real examples of such dialogues are seldom available, and are expensive, labour-intensive and difficult to obtain. One alternative is to produce artificial simulated dialogues, given a small number of real examples from which a simulated user can be built. These simulated dialogues, provided they are sufficiently realistic, can be used as input to methods for learning a decision policy. We describe some practical experiments which we have carried out on dialogue simulation. Since an annotated version of the Companions corpus has only recently become available to us, these experiments were carried out on the Dihana corpus: a collection of almost 1000 rail journey query dialogues in Spanish. Assessing the quality of simulated dialogues is not easy, and we also describe various experiments in which alternative methods of judging quality are compared.

The purpose of generating a large number of simulated dialogues is to have data with which to learn a good decision policy. We go on to describe experiments in which we construct a Markov Decision Process dialogue model for the Dihana application, and learn a policy for it from the simulated dialogues using Q-learning, a popular reinforcement learning method. The resulting system is evaluated along various dimensions.

Finally, we discuss ways in which some of these decision theoretic mechanisms might be incorporated into one of the current Companions prototype dialogue systems.

---

# 1 Decision Networks: An Overview

## 1.1 Graphical Models

In recent years graphical models have become increasingly important in the machine learning community. Their popularity stems partly from the fact that they provide clear, succinct, and intuitive ways of describing the relationships between the variables of a model, and partly from the solid theoretical grounding they have in statistics. These properties enable graphical models to deal with both the complexity and the uncertainty that plagues many machine learning problems [13]. In essence, graphical models are efficient representations of the joint probability distributions and conditional independence that exist between the variables of a model. Probabilistic inference in graphical models is achieved through message passing.

Graphical models are network representations in which the nodes correspond to model variables and the edges to the probabilistic relationships between them. The fact that conditional independence is represented implicitly by the absence of edges between nodes, leads to an efficient sparsely connected graphical representation of probabilistic models.

There are two kinds of graphical models: *undirected* and *directed*. Undirected graphs can also be described as *Markov networks* or *Markov random fields* [11]. Problem solving with Markov networks is concerned with identifying maximal *cliques* within the graph, where a clique is a subgraph in which every node is connected to every other node. Directed graphs can also be described as *belief networks* or *Bayesian networks*. The directed edges in these graphs can be interpreted as expressing causal relationships between the random variables of the model. It is possible to use the directed edges to represent non-causal relationships between variables, but this can often lead to sub-optimal graphical representations. The node at the start of an edge in a directed graph is often described as the *parent*, whilst the node at the destination of an edge is often described as the *child*. A *path* through a directed graph denotes a sequence of connected nodes. A path in which three nodes  $X_i$ ,  $X_j$  and  $X_k$  are connected in sequence such that  $X_i$  is the parent of  $X_j$  and  $X_j$  the parent of  $X_k$  is described as a *serial path* (Figure 1a). A *convergent path* is where one node is the child of two or more parent nodes (Figure 1b). A *divergent path* is where one node is the parent of two or more other nodes (Figure 1c).

Inference using directed graphs generally involves finding the values of nodes representing *latent* (or *hidden*) variables, given the values of nodes representing observed variables. There are two forms of inference using directed graphs: *predictive* and *diagnostic* inference. Predictive inference progresses in a forward direction along the edges between the nodes. In this case, the values of nodes representing causes are known and the values of nodes representing effects or symptoms are the predicted outcomes of those causes. Diagnostic inference, on the other hand, progresses in a backward direction along the edges from the nodes representing effects or symptoms back to nodes representing their potential causes. For many applications, the size of the graph-

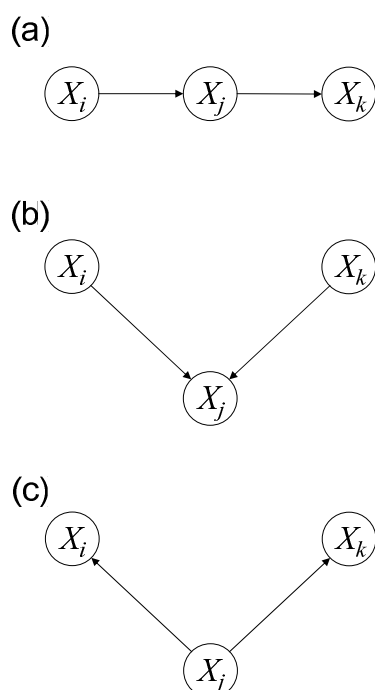


Figure 1: (a) serial, (b) convergent and (c) divergent paths in a graphical model

ical model makes exact inference computationally intractable. Consequently, several methods have been used for approximating the inference on large graphs, including *variational* methods such as *mean-field approximation* and *Markov Chain Monte Carlo* sampling methods.

Machine learning can be applied to graphical models to derive either the parameters of the model (e.g. probability distributions over node values) or the connectivity structure of the graph, or both. Different machine learning methods are required depending on whether or not all the variables of the model are observed.

Graphical models can be adapted to support decision making by incorporating additional nodes that denote the potential *utility* of certain actions or *decisions*. Inference methods can be applied to these so-called *influence graphs* to establish an optimal sequence of actions needed to maximise the expected utility. Figure 2 shows an example of a small influence graph to support decision making within a dialogue management context [16]. The graph models the decision between making a *checking* move (i.e. to check the previous dialogue move) or a *new topic* move (i.e. start a new topic in the dialogue). This decision is influenced by factors concerning the level of *noise*  $N$  in the environment in which the dialogue is being held and the importance of the *speed*  $S$  and *accuracy*  $A$  of the dialogue. The diagram in Figure 2 includes a square node labeled  $D$  that represents the *check/new move* decision, and a diamond node labeled  $U$  that represents the utility of that outcome of making that decision. The directed edges in the

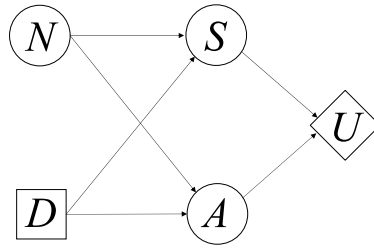


Figure 2: An influence graph modelling a simple dialogue move choice. The variables represent environmental noise ( $N \in \{yes, no\}$ ), dialogue speed ( $S \in \{fast, slow\}$ ), move decision ( $D \in \{check, new\}$ ), accuracy ( $A \in \{yes, no\}$ ) and decision utility ( $U$ )

graph show that both the noise variable  $N$  and the decision node  $D$  directly influence the variables representing the speed  $S$  and accuracy  $A$  of the dialogue. The resulting speed/accuracy ratio determines the value of the utility node  $U$ . Reinforcement Learning (see below) is commonly used to derive the parameters of influence graphs.

## 1.2 Bayesian Networks

Bayesian Networks are graphical models represented using Directed Acyclic Graphs (DAGs) and are capable of both predictive and diagnostic inference. The joint distribution of node values in a Bayesian Network is computed using the product of the conditional probabilities of the model variables. For a network with  $n$  nodes, each denoted by  $X_i$ , and with  $\Psi_i$  denoting the set of nodes that are the parents of node  $i$ , the joint probability distribution is given by:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | \Psi_i) \quad (1)$$

An important concept in Bayesian Networks is *d-separation*, which relates to the conditional independence between variables. Two nodes  $X_i$  and  $X_k$  of a network are said to be d-separated if every path between  $X_i$  and  $X_k$  includes an intermediate node  $X_j$  such that either:

- the path is serial (Figure 1a) or diverging (Figure 1c) and  $X_j$  is instantiated, or
- the path is converging (Figure 1b) and  $X_j$  (and none of its descendants) have been observed.

If  $X_i$  and  $X_k$  are d-separated, then any change in the value of  $X_i$  with not effect  $X_k$  and vice-versa.

Each node in a Bayesian Network has a finite set of mutually exclusive *states* assigned to it which correspond to the set of potential values of the associated random

variable. Each node also has a table of *potentials* assigned to it which contains the conditional probabilities  $P(X_i|\Psi_i)$  for the node.

### 1.3 Reinforcement Learning

Reinforcement Learning (RL) is an approach to learning that is centered on the *experiences* that the learner has of interacting with the environment (Figure 3). The concept of a reward is central to the reinforcement learning paradigm. As the learning agent interacts with the environment, each action elicits a *reward* from the environment. Reinforcement Learning methods attempt to learn which sequence of actions will maximise the reward given by the environment.

An RL system consists of:

- A finite state of states:  $S$
- A finite set of actions:  $A$
- A state transition function  $T$  that defines a probability of moving to state  $s'$  from state  $s$  having taken action  $a$ :  $T(s, a, s') = P(s'|a, s)$
- A reward function  $R$  which defines the expected reward for a given action  $a$  in state  $s$ :  $R(s, a)$
- A policy  $\pi$  which determines which action to perform in any given state  $s$  at time  $t$ :  $\pi_t(s)$

The reward policy  $\pi$  can either be *stationary* or *non-stationary*. A stationary policy is one that is dependent only on the current state  $\pi(s)$ , whilst a non-stationary policy is also dependent on time  $\pi_t(s)$ .

In general, RL needs to deal with three problems: credit assignment, generalisation, and the *exploration* versus *exploitation* trade off. The credit assignment problem is concerned with the issue of delayed reward: often the full reward from a sequence of actions are not received until later in the interactive sequence. It then becomes necessary to decide how to assign reward to earlier actions in the sequence. This is done using temporal difference learning.

In many applications, it is not possible for the learner to explore all possible action/state reward pairs. The issue then is, how does the learner generalise from what it does know of these rewards to previously unseen action/state pairs? For this, it is necessary to combine RL with generalisation methods, most frequently 'function approximation' - supervised learning (e.g. ANNs).

In RL, the learner gathers information about the reward for action/state pairs. As it does so, in each state it must choose either to apply an action which it knows from experience will produce a good reward (exploitation), or to try an action which for which the reward is unknown (exploration), leading it to potentially unexplored states.

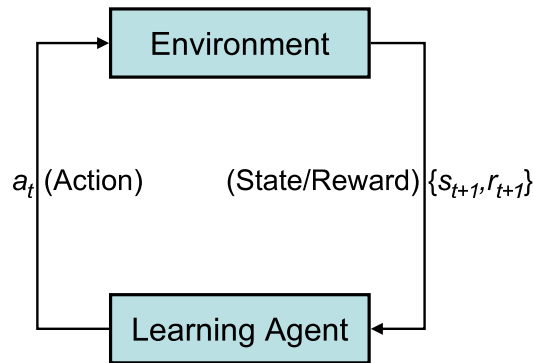


Figure 3: The interaction cycle between learning agent and environment in a reinforcement learning system

## 1.4 Markov Decision Processes

Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs - see below) are versions of *dynamic* Bayesian networks that satisfy the *Markov property*. That is, they are used to model systems that change state over time in which a new state depends only on the previous state and the action performed (Figure 4). The state of the system at a moment in time is represented by a node (or set of nodes), the nodes for consecutive states are connected to represent the temporal relationship between them.

A solution for an MDP is a policy  $\pi_V(s)$  which maps states onto actions based on the value  $V_\pi(s)$  of the subsequent states:

$$\pi_V(s) = \operatorname{argmax}_a \left[ R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V(s') \right] \quad (2)$$

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi s, s') V_\pi(s') \quad (3)$$

The solution involves iteratively finding the values of  $\pi_V(s)$  and  $V_\pi(s)$  until convergence is reached. Various algorithms have been developed for this including *Value Iteration* or *Backward Induction* (Bellman 1957) in which the value of  $\pi_V(s)$  is calculated only when needed. *Policy Iteration* offers an alternative method in which the value of  $\pi_V(s)$  is initially fixed whilst all the values of  $V_\pi(s)$  are calculated.  $\pi_V(s)$  is then updated based on  $V_\pi(s)$ . The new value of  $\pi_V(s)$  fixed again as  $V_\pi(s)$  is updated. This cycle continues until the values converge.

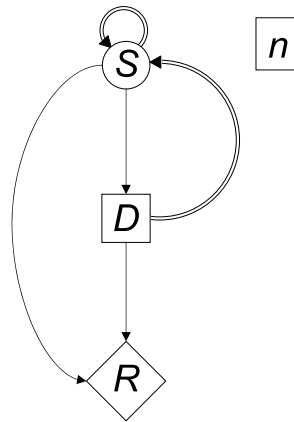


Figure 4: A graph representing a Markov Decision Process in which  $S$  denotes the state of the world,  $D$  denotes the decision or action of the system, and  $R$  denotes the reward. The double-lined arrows indicate a relation between the source at time  $t$  and the target at time  $t + 1$  (i.e. from  $S_t$  to  $S_{t+1}$  and from  $D_t$  to  $S_{t+1}$ ).  $n$  indicates that this subgraph is repeated for  $n$  time steps.

## 1.5 Partially Observable Markov Decision Processes

A significant drawback of MDPs is that they require complete knowledge of the state that the environment is in. In many real-world applications, it is difficult or impossible to observe the environmental state with absolute certainty, given, for example, the presence of measurement noise or reliability issues that are common with input devices. Partially Observable Markov Decision Processes (POMDPs) are a generalisation of MDPs designed to cope with precisely this issue (Figure 5).

POMDPs inherit all the components of MDPs, including a set of states  $S$ , a set of actions  $A$ , a set of state transition probabilities  $T$  and a reward function  $R$ . POMDPs additionally have a set of observations  $\Omega$ , and an observation function  $O(s', a, o)$ , which is the probability of observing  $o$  after taking action  $a$  and ending up in state  $s'$ . Instead of having a discrete state space, as with MDPs, POMDPs define a *belief* space  $b(S)$  which is a probability distribution over the set of states  $S$ , expressing the degree of belief over what state the environment is in at a given time.

POMDPs implementations typically include a *State Estimator*, which is responsible for updating the belief state based on the current observation  $o$ , the last action to be performed  $a$ , and the previous belief state  $b(S)$  (Figure 6). POMDPs use a policy  $\pi$  to map the current belief space to potential actions.

The value function for POMDPs must extend the MDP value function to accommodate the belief state:

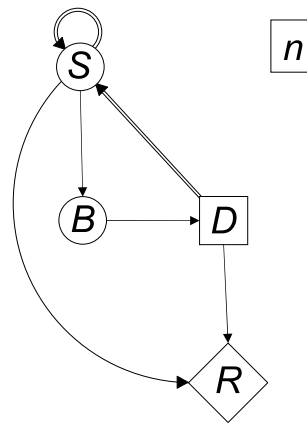


Figure 5: A graph representing a Partially Observable Markov Decision Process in which  $S$  denotes the (hidden) state of the world,  $B$  denotes the belief state of the system,  $D$  denotes the decision or action of the system, and  $R$  denotes the reward.

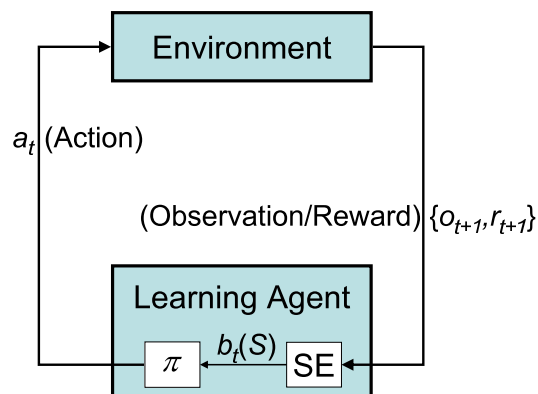


Figure 6: The interaction cycle between learning agent and environment in a POMDP. SE denotes the State Estimator which estimates the currently belief state  $b_t(S)$  that the policy  $\pi$  uses to determine which action to taken next.

$$V_{\pi}(b(S)) = \sum_{s \in S} b(s) V_{\pi}(s) \quad (4)$$

As with MDPs, the value function  $V_{\pi}(b)$  is computed using value iteration. POMDPs suffer from scalability problems, given the fact that the belief space  $b(S)$  is continuous. Various algorithms have been proposed to compensate for this, including the *Witness* algorithm. However, this is only feasible for relatively small state spaces. Most current implementations train an approximate policy using a reinforcement learning technique such as Q-learning. Intuitively, this technique begins with known reward values associated with final states in the POMDP. The reward values are recursively propagated back through earlier states so that if the state space has been sufficiently well sampled, all transitions will have appropriate rewards associated with them. In the best case Q-learning will converge to a stable solution, but for many large applications it is sufficient to run the algorithm for a sufficient number of iterations that an approximately optimal reward function is assigned. If possible, the sampling should be based on a corpus of actual or simulated dialogues.

## 2 Dialogue simulation using probabilistic models

### 2.1 Introduction

As mentioned earlier, collecting real dialogue data is an extremely time-consuming and expensive process. It has therefore become a common practice in the dialogue community to create artificial dialogues by building a simulated ‘user’. From a small sample of dialogues, it is usually possible to build a probabilistic system which will generate further dialogues of a similar type. In this section, we will be presenting some Machine Learning (ML) probabilistic techniques of dialogue simulation and experiments over a dialogue corpus. Because at this point in the project we do not have a real dialogue manager, the purpose of these experiments is to generate both user and system turns using some statistical models estimated from real data. The level of simulation is both at the intention level (dialogue act tags like ‘yes-no question’, or ‘confirmation’), and also at some semantic levels (frames and data attributes, such as names of cities, or times of trains). Using these two levels is motivated by the fact that the corpus with which the experiments were performed had been annotated at both levels.

#### 2.1.1 Motivation

There are two main goals for synthetically simulated dialogues:

- **generation of samples for developing ML dialogue managers.** In all data-driven approaches to problems of natural language processing the amount of training data is a critical aspect in order to obtain good performance. For example, only when

large parallel corpora (i.e. aligned sentences in both source and target languages) became available did machine translation (MT) based on ML begin to get interesting results. Dialogue corpora are hard and expensive to acquire, transcribe and afterwards annotate. Great amounts of resources are needed to get large annotated corpora, and the largest dialogue corpora that exist at the moment are only a few thousand dialogues, which is insignificant compared to MT corpora that may contain billions of sentences. If we mean to optimize dialogue managers using data driven approaches, really large amounts of data will be needed. In principle, simulation could be a way to achieve this. However, some caution is needed. This process may not produce good results if we are not careful about the simulation. If we are going to develop models from data that was synthetically generated by another model, it would be possible that these developed models will only estimate the generator models, i.e. be too close to them, without enough generalization. For this reason, in order to check their quality, independent criteria will be needed to evaluate these models.

- **inexpensive, automatic and fast way to evaluate the performance of different dialogue managers.** Evaluation of dialogue systems in general and dialogue managers especially are an open issue. However in many projects, real users are used to evaluate the models. This process is very expensive and slow since people must be found and usually paid. Furthermore, before getting a definite model it is common to have several provisional prototypes that need to be rapidly evaluated. Using user simulation can be a solution to the evaluation of these prototypes and an automatic comparison of final dialogue managers.

### 2.1.2 Related work

In the existing literature, some previous dialogue simulation works using machine learning methods can be found. Usually they are only simulating the user turns because they have a working dialogue manager to perform system turns. This requires the building of a 'user model' that will give behaviour approximately like that of a real user.

The first work that performed user simulation is [4] in which they experimented with different user model policies to evaluate robustness of dialogue systems. The user models are based on n-grams of intentions and the evaluation criterion that they used was the length of the dialogues. This approach was also followed by [25], using additionally some overall structure over the intentions of the user utterances in order to be consistent with a predefined user goal. They compared the average number of turns needed to achieve the goal, and the failure rate of the simulated dialogues, with real data. There is a third line of work that uses n-grams to simulate dialogues [6], comparing them with another method based on supervised learning using linear feature combinations. They measured the quality of the simulation as a function of filled and confirmed slots: in this kind of task oriented dialogue the aim of the system can be conceptualised as a process of filling a predetermined number of slots with appropriate values.

In [15] a complete probabilistic simulation of all the components of a dialogue system was performed. The user model relies on the Bayesian Network framework. Using these simulation techniques they try to train a reinforcement learning agent based on Markov Decision Processes (MDP). [17] investigated an agenda based used model. The user goal and user agenda “provide compact models of the dialogue context and user’s state of mind”. As the user agendas are not observable data they use an Expectation-Maximization training method to estimate the model parameters on data. Also, they have to use some constraints and a summary space representation - i.e. a representation based on a sample of the complete space - to reduce the intractable number of parameters that can exist. They compare dialogue length, completion rate, and performance of the trained model with real users against the same parameters with a handcrafted simulation policy. The trained model displays better behaviour.

## 2.2 Knowledge representation in dialogue

For our purpose, that is dialogue simulation at the intention level, a dialogue  $D$  can be described as an alternating series of user and system turns:

$$D = S_0, U_1, S_2, U_3, \dots, S_{T-2}, U_{T-1}, S_T$$

where  $S_i$  are turns performed by a system agent and  $U_j$  are turns by user agents. Each turn is formed by the words that the user/system utters. Obviously, at each step  $t$  of the dialogue, only information from previously generated turns is available.

At the same time, every turn is formed by one or more sequences of words, where each one describes a particular linguistic intention (speech act). That is, every sequence of words (utterance) is assigned with dialogue acts (DAs) labels, and depending on their definition they might also contain some semantic information. For example, in some applications it might be useful to subdivide the category of questions into more detailed acts like ‘ASK-TIME’, or ‘ASK-PLACE’. Each of the user and system turn can be composed of one or more DA labels of this predefined set, that are assumed to be the same for both agents.  $\Lambda_i = \Lambda_i^0, \Lambda_i^1, \dots, \Lambda_i^{|\Lambda_i|-1}$  where  $\Lambda_i \in \{S, U\}$  can be a user or system turn and  $\Lambda_i^j$  is the  $j$ -th DA of this  $i$ -th turn.

In task-oriented dialogues, i.e. those where the target of the dialogue is to carry out a specific well-defined task, such as gathering all the fields needed for a database query), other information that is available is the semantic data of the task that the turn contains. It can be the data contained in the user turn (or that the system believes is contained) or the information that the system gives in its turn (or data that the user believes the system gave).

$$D = \{S_0, V_{S_0}\}, \{U_1, V_{U_1}\}, \{S_2, V_{S_2}\}, \{U_3, V_{U_3}\}, \dots \\ \dots \{S_{T-2}, V_{S_{T-2}}\}, \{U_{T-1}, V_{U_{T-1}}\}, \{S_T, V_{S_T}\}$$

where  $V_{S_i}$  is the semantic data available for the system at turn  $i$ , and  $V_{U_j}$  is the semantic data available for the user at turn  $j$ .

## 2.3 Probabilistic Models

As some of the previously mentioned works do, the models that we are going to use are n-grams, i.e. sequences of words or even dialogue acts 1, 2 or n long. However, we have performed a systematic study of the effects of generating segmented or unsegmented turns and the relevance of using data state. By ‘segmented’ we refer to the fact that a particular utterance may realize several dialogue acts: for example, ‘thank you, and goodbye’.

### 2.3.1 N-Grams using unsegmented turns

As was explained previously our aim is to generate both user and system turns for a dialogue. One possible approach to the problem is to generate directly the whole unsegmented turn (every turn is composed by one or more DA labels) according to the probability of each turn given information about previous turns. We look at this possibility first, so that in all the experiments reported in this section, the unit (word) of the n-gram will be the whole turn. The model using only turn labels will be  $\Pr(\Lambda_i | \Lambda_{i-1}, \Lambda_{i-2}, \Lambda_{i-3}, \dots, \Lambda_0)$ , where  $\Lambda_j \in \{S, U\}$ . Logically, the history must be limited using only  $n - 1$  previous turns, being the models estimated by n-grams.  $\Pr(\Lambda_i | \Lambda_{i-1}, \Lambda_{i-2}, \dots, \Lambda_{i-n+1})$ . Adding the semantic data of the task, a possible model will be the following:

$$\Pr(\Lambda_i | \{\Lambda_{i-1}, V_{i-1}\}, \{\Lambda_{i-2}, V_{i-2}\}, \dots, \{\Lambda_{i-n+1}, V_{i-n+1}\})$$

where  $V_j$  will be  $V_{S_j}$  if  $\Lambda_j = S_j$  and  $V_{U_j}$  if  $\Lambda_j = U_j$ .

Basically, the process of generation will be the following: an initial system turn is always deterministically generated and an initial data state is set. Afterward, in each turn, a whole turn will be randomly generated according to the distribution probabilities estimated from real samples. This process will be repeated until the system generates the turn that finishes the conversation.

### 2.3.2 N-Grams using segmented turns

The process is similar to the one using unsegmented turns, but the unit to generate is the DA label individually, i.e. the words of the n-gram are the labels of the segments. Therefore, it is possible to generate sequences of DAs that do not appear in the corpus: user or system turns that have not been seen in the training corpus. The probabilities can be represented:

$$\Pr(\Lambda_i^j | \Lambda_i^{j-1}, \dots, \Lambda_i^0, \Lambda_{i-1}^{|\Lambda_{i-1}|-1}, \dots, \Lambda_{i-1}^0, \dots, \Lambda_0^{|\Lambda_0|-1}, \dots, \Lambda_0^0)$$

Table 1: The set of dialogue act labels used in the Dihana corpus

Level	Labels
First	Opening, Closing, Confirmation, Undefined, Not-understood, Waiting, Consult, Acceptance, Rejection
Second	Departure-hour, Arrival-hour, Fare, Origin, Destination, Day, Train-Type, Service, Class, Trip-time
Third	Departure-hour, Arrival-hour, Fare, Origin, Destination, Day, Train-Type, Service, Class, Trip-time, Order-number, Number-trains, Trip-type

## 2.4 Corpus

In the dialogue framework as other Computational Linguistic areas the results of the experiments usually depend directly on specific features of the data with which experiments were performed. For this reason a detailed description of it is needed.

The corpus used for the experiments is the Dihana corpus [8] which is composed of 900 ‘system’-to-human dialogues. In order to limit the task domain, the queries were restricted to timetables, fares, and services for long-distance trains in Spanish. The acquisition of the corpus was carried out by means of an initial prototype and using the Wizard of Oz (WoZ) technique: this is a well known technique in the community in which an unseen human simulates the responses of a computer system unknown to the user.

A three-level annotation scheme of the Dihana corpus utterances was defined. The DA set used represents the general purpose of the utterance (first level), as well as more precise semantic information specific to each task (second and third levels). The second level indicates the repository of information implicit in the utterance (i.e. data used or modified according to the first level intention label), and the third level represents the specific data present in the utterance. An example of a label can be given: (M:Confirmation: Day: Day, Departure-hour). The M indicates that is a Machine (System) turn, the first level is Confirmation, the second level is Day and the third level is Day, Departure-hour. This label can correspond to the following system utterance of “*Do you want to leave on Saturday, May the 15th of 2004 at 7 pm?*”

Using this annotation scheme, dialogues were automatically labeled and manually corrected. After this process, there were 248 different labels (153 for the user, 95 for the system) using the three-level scheme. When considering only the first and second levels, there were 72 labels (45 for user, 27 for system). When considering only the first level, there were only 16 labels (7 for user, 9 for system).

Other important features of the dialogue annotation are:

- Every system or user turn can be formed by one or more labels, e.g. the labels (U:Question: Departure-hour: Origin), (U:Question: Departure-hour:Destination) correspond to the utterance turn: “*I want to*



Table 2: Values that appear at data vector. The order value indicates the position of the attribute or frame inside the vector.

<b>Order</b>	<b>Attributes</b>	<b>Frames</b>
1	Origin	Departure_time
2	Destination	Arrival_time
3	Interval_departure_dates	Fare
4	Interval_arrival_dates	Duration_type
5	Interval_departure_day	Train_type
6	Interval_arrival_day	Confirmation
7	Interval_departure_time	Affirmation
8	Interval_arrival_time	Negation
9	Train_type	Closing
10	Travel_type	Not_understanding
11	Class	
12	Relative_order_N_S	
13	Relative_order_N_S	
14	Fare	
15	Duration_trip	
16	Services	
17	Current_train	
18	Current_time	
19	Current_date	
20	Current_route	
21	Current_origin	
22	Current_destination	

Table 3: Description of the data vector. NC is the number of different vectors that appear in the corpus. PNC is the potential number of combinations that could exist. AvgO is the average number of occurrences of each different vector that appears. Avg10 indicates the number of combinations that appear at least 10 times

<b>Data Vector</b>	<b>NC</b>	<b>PNC</b>	<b>AvgO</b>	<b>Avg10</b>
Attributes	445	$2^{22}$	14.1	65
Frames	94	$2^{10}$	66.77	33
Attributes-Frames	1433	$2^{32}$	4.38	101

Table 4: Number of labels combining the annotated turns (3-level and 2-level labellings) with data vectors (attributes and frames). #Labels indicates the number of labels with the combination of data vector and #SLabels and #ULabels indicates the number of labels for system and user turns respectively.

Attributes	Frames	3-levels			2-levels		
		#Labels	#ULabels	#SLabels	#Labels	#ULabels	#SLabels
N	N	957	742	215	201	130	71
Y	N	2703	2486	215	1863	1792	71
N	Y	1809	1594	215	730	659	71
Y	Y	3598	742	215	2937	2866	71

Table 5: The percentage of different dialogues that end properly for 100000 simulated dialogues using unsegmented turns. The first value of each cell corresponds to dialogues using the first smoothing technique and the second one to dialogues using the Good-Turing discounting strategy.

Attribs	Frames	Percentage of different dialogues that end properly		
		2-grams	3-grams	4-grams
N	N	71.80/69.58	57.57/65.11	36.51/56.28
Y	N	80.40/72.16	25.24/63.39	9.66/58.68
N	Y	79.10/73.68	38.17/65.94	14.71/59.28
Y	Y	81.64/73.84	8.77/65.39	2.60/62.43

There are no clear generally agreed criteria for evaluating the quality of artificially generated dialogues, but some simple preliminary measures can be used to see if a dialogue is not valid. A necessary condition, but not sufficient, for a ‘good’ dialogue is that it ends properly (user emits (U:Closing: Nil: Nil) label and system replies with (M:Closing: Nil: Nil) label). We are also interested in not emitting duplicated dialogues. Another way to see the quality of the generated dialogues, is to check how many of the dialogues from the original corpus are re-generated. Additionally minimum edit distance can be calculated for each of the dialogues generated (only for these that end properly) compared with the original corpus (900 dialogues). (Edit distance is a commonly used method of comparison between two sequences of items. It counts how many items in one sequence would need to be deleted, inserted, or replaced to make it identical to the other sequence.)

### 2.5.2 Unsegmented turn experiments

In this section, the results that are presented have been obtained using unsegmented turns. That is, all the whole turn is generated in each step.

Experiments were done using three-level labels, and 100000 (100K) dialogues were generated, using the two different smoothing methods previously explained. All combinations of data state have been explored. We have the results shown in Table 5.

Analyzing only these generated dialogues that are different (not repeated) and end properly, Table 6 show the statistics that are obtained.

### 2.5.3 Segmented turn experiments

Experiments in this section have been performed using as the unit of generation individual DAs, that are obtained via the segmentation of the turns. Experiments were done

Table 6: The average length and standard deviation (in turns) for the different dialogues. Again, in each cell the first two values are the average and standard deviation for dialogues obtained using the first smoothing technique and the second two values using Good-Turing discounting strategy.

Attribs	Frames	Avg. length and standard deviation (in turns)			
		2-grams	3-grams	4-grams	Orig
N	N	12.34(12.88)	13.85(14.24)	15.67(16.00)	14.95
		/12.34(13.08)	/12.72(13.47)	/12.76(13.54)	(15.86)
Y	N	12.18(12.69)	15.49(15.82)	16.10(16.43)	14.95
		/11.83(12.65)	/12.51(13.37)	/12.38(13.27)	(15.86)
N	Y	12.17(12.69)	14.92(15.30)	15.67(16.01)	14.95
		/12.01(12.80)	/12.50(13.30)	/12.51(13.35)	(15.86)
Y	Y	11.92(12.41)	15.76(16.12)	15.31(15.71)	14.95
		/11.19(12.04)	/11.86(12.77)	/11.74(12.65)	(15.86)

Table 7: The average minimum edit distance and standard deviation from the original corpus (in turns). In each cell, values for dialogues using the two different smoothing techniques are separated by /.

Attribs	Frames	Avg. min. edit distance and st. dev.		
		2-grams	3-grams	4-grams
N	N	5.49(6.10)	5.30(5.79)	5.09(5.56)
		/5.38(6.10)	/5.49(6.22)	/5.58(6.35)
Y	N	4.99(5.51)	5.11(5.57)	4.33(4.87)
		/5.39(6.16)	/5.47(6.24)	/5.50(6.29)
N	Y	5.13(5.68)	5.21(5.70)	4.39(4.91)
		/5.34(6.08)	/5.37(6.13)	/5.43(6.22)
Y	Y	7.78(5.29)	4.38(4.97)	2.85(3.67)
		/5.34(6.13)	/5.44(6.25)	/5.43(6.25)

Table 8: Percentage of the original corpus generated in this 100k dialogues. In each cell, values for dialogues using the two different smoothing techniques are separated by /.

Attribs	Frames	Percentage orig. corpus generated		
		2-grams	3-grams	4-grams
N	N	3.22/4.44	16.44/7.22	37.00/5.11
Y	N	5.22/4.33	49.22/4.22	66.55/3.77
N	Y	5.22/4.44	30.22/7.00	51.66/5.44
Y	Y	6.440/3.66	60.22/4.44	66.22/2.55

Table 9: The percentage of different dialogues using segmented turns that end properly. In each cell, values for dialogues using the two different smoothing techniques are separated by /.

At	Fr	Percentage of different dialogues that end properly				
		2-grams	3-grams	4-grams	5-grams	6-grams
N	N	83.82/84.33	88.12/87.51	84.32/86.62	79.52/84.45	59.01/82.54
Y	N	84.14/81.02	82.55/82.15	65.47/80.92	52.94/80.08	17.18/80.41
N	Y	85.65/84.53	86.57/86.38	76.31/84.86	64.65/82.95	29.69/82.761
Y	Y	83.97/77.38	78.93/78.79	55.32/77.89	42.47/77.70	6.22/77.61

using three-level labels, and 100000 (100K) dialogues were generated. Several values of  $n$  for the  $n$ -gram have been explored and for each set-up, two different smoothing techniques have been used. In the first one, as described earlier, models were not really smoothed: if there was not any  $n$ -gram with a particular history, the  $n$  is simply decreased by one. At the second one, backoff using the Good-Turing discounting strategy has been used. Additionally, different combinations of the data state have been explored. We have the results shown in Table 9.

Analyzing only these generated dialogues that are different (not repeated) and end properly, the statistics shown in Table 10 are obtained.

#### 2.5.4 System turn prediction

A second set of experiments were done to see if models can predict the behaviour of the system turn. They were only done for system turns (unsegmented turns) and we studied the influence of the data vector, which occur only in user turns. Basically, the experiments were done for each dialogue of the test set and each system turn of the

Table 10: The average length and standard deviation (in turns). In each cell, values for dialogues using the two different smoothing techniques are separated by /.

At	Fr	Avg. length and stand.deviation (in turns)					
		2-grams	3-grams	4-grams	5-grams	6-grams	Orig
N	N	14.23(15.91)	14.29(15.55)	14.67(15.66)	14.77(15.53)	15.68(16.36)	14.95
		/13.53(15.24)	/13.94(9.32)	/14.30(9.68)	/14.70(10.21)	/15.03(10.64)	(15.86)
Y	N	12.94(14.30)	13.73(14.69)	14.99(15.79)	15.30(15.96)	17.36(18.09)	14.95
		/10.44(11.77)	/11.91(13.51)	/12.07(13.72)	/12.22(13.91)	/12.35(14.07)	(15.86)
N	Y	13.80(15.32)	14.07(15.10)	14.56(15.37)	14.76(15.43)	16.44(17.15)	14.95
		/12.09(13.65)	/13.13(14.76)	/13.38(15.09)	/13.70(15.51)	/13.70(15.51)	(15.86)
Y	Y	12.97(13.72)	13.64(14.63)	15.01(15.81)	15.44(16.14)	17.30(18.15)	14.95
		/9.09(10.15)	/10.18(11.50)	/10.31(11.67)	/10.39(11.76)	/10.44(11.84)	(15.86)

Table 11: The average minimum edit distance and standard deviation from the original corpus (in turns). In each cell, values for dialogues using the two different smoothing techniques are separated by /.

Attribs	Frames	Avg. min. edit distance and st. dev.				
		2-grams	3-grams	4-grams	5-grams	6-grams
N	N	8.36(9.54)	7.60(9.32)	7.33(9.68)	6.90(10.22)	6.86(10.65)
		/8.07(9.54)	/7.94(9.32)	/8.23(9.68)	/8.67(10.22)	/9.04(10.65)
Y	N	7.22(7.49)	6.82(8.35)	7.08(8.58)	6.70(8.82)	6.71(8.95)
		/6.43(7.48)	/7.09(8.35)	/7.27(8.58)	/7.46(8.82)	/7.57(8.94)
N	Y	7.85(8.53)	7.10(8.90)	6.89(9.23)	6.53(9.70)	6.57(9.73)
		/7.24(8.54)	/7.55(8.91)	/7.80(9.24)	/8.17(9.70)	/8.21(9.74)
Y	Y	6.89(6.67)	6.77(7.37)	6.98(7.53)	6.61(7.61)	6.07(7.67)
		/5.84(6.68)	/6.34(7.36)	/6.46(7.53)	/6.54(7.61)	/6.58(7.67)

Table 12: Percentage of the original corpus generated in this 100k dialogues. In each cell, values for dialogues using the two different smoothing techniques are separated by /.

Attribs	Frames	Percentage orig. corpus generated				
		2-grams	3-grams	4-grams	5-grams	6-grams
N	N	2.77/2.55	4.77/4.00	8.77/5.11	13.33/5.00	33.00/3.55
Y	N	2.77/1.44	7.44/3.22	19.44/3.00	29.00/2.88	81.44/2.33
N	Y	3.00/2.44	5.77/3.88	13.55/4.11	24.66/3.22	64.88/3.11
Y	Y	3.00/1.44	8.11/2.22	23.77/2.33	33.44/2.11	92.00/1.66

Table 13: Average of the percentage of correctly predicted system turns for cross-validation experiments.

Attributes	Frames	3-levels			2-levels		
		2-grams	3-grams	4-grams	2-grams	3-grams	4-grams
N	N	36.2	36.0	33.6	49.4	51.3	49.5
Y	N	29.5	28.6	27.9	46.7	46.0	44.7
N	Y	33.6	32.8	32.3	49.6	49.9	48.0
Y	Y	24.5	24.1	22.5	42.9	41.8	39.7

dialogue, the intention being, given the real history, to try to guess the following turn. The formula to predict the turn is:  $\arg \max_{s_i \in \Sigma} \Pr(S_i | U_{i-1}, S_{i-2})$ , where  $\Sigma$  is the set of possible system turns. This formula is only for 3-grams, with equivalent ones for different n-grams when  $n \neq 3$ .

Cross-validation experiments using 5 partitions were performed, obtaining the results shown in Table 13 in accuracy for 3-level and 2-level labellings.

### 2.5.5 Some implementation details

Generally, in experiments where state information has been used (when the vector of data is available) this value was added at the end of the original turn as if it was another dialogue label. Therefore, for implementation details, some of the n-grams will include also the state information (all the user turns but not most of the closing ones).

The toolkit that was used to calculate the n-grams is the CMU-Cambridge Statistical Language Modelling Toolkit v2 [2].

## 2.6 Discussion

In this work some experiments of dialogue simulation using n-grams have been performed, testing two different ways of dealing with the data. On the one hand independent dialogue acts have been used as units of the n-grams. On the other hand, each whole turn forms the unit of the n-gram by itself. Additionally, information of data state has been also used to contribute to the knowledge representation of the model.

It is known that statistical models such as n-grams need a large amount of data to estimate properly their parameters. The larger the size of vocabulary the less the number of samples per class and the worse the estimation of the models. As we are always limited by the small size of the corpus (900 dialogues), models with whole turns with data state, where there are a lot of combinations (3598 words with the most complex information state), will be worse estimated than these ones formed by DAs as words and less information in data state (248 words without any information state). This effect can be clearly seen as we compare the number of dialogues that end properly (Tables 5 and 9). Another important consequence of using independent DAs as units of n-gram is that we need to use a large  $n$  for giving information of previous turns, because many turns are composed by several DAs.

However, the most similar simulated dialogues to the original corpus have been obtained using the models with more knowledge information sources, (see Tables 8 and 12). This feature may be seen positive, because we want to obtain similar data to the original one as we know that they are real dialogues, but we do not want identical dialogues. Otherwise, these models with data state do not help to improve the prediction of system turns (Table 13).

Finally, we cannot definitely state which models are better for simulating dialogues as we think the results and evaluation measures are not completely discriminative. Nevertheless, we believe that there must be a trade-off between the size of vocabulary (combinations of labels) and the sources of knowledge information used.

## 2.7 Future work

As in other works that address the same problem, an evaluation criteria that measures the failure or success of the generated dialogues at the specific task (giving information about train times and fares) must be used.

Other important factor to check the quality of the proposed models is to use them as training data for some Reinforcement Learning technique aimed at learning a policy for a MDP or POMDP. Then, these models can act as the system agent and the n-gram models act as the user agent.

Last but not least, extension of these experiments to other corpora is needed, and especially to the Companions corpus. A part of this corpus has only recently become available to us: when the experiments reported here were being carried out the acquisition of the Companions corpus was still in progress. At the present moment some dialogues are transcribed but not yet accurately annotated at the dialogue level.

From them, it is possible to see what kind of dialogues we are going to work with. They seem to be as long as Switchboard ones [10] (in terms of number of turns per dialogue). However, they are also initially going to be task limited with a defined set of attributes (photographer, people who appear in the photographs, place, time, etc) which makes them more similar to some task orientated corpora such as Communicator [23] or Dihana.

## 3 MDP

### 3.1 Definition

A Markov Decision Process, as we saw briefly in the introductory section, models the synchronous interaction of an agent with a world, providing a mathematical framework where the output of the agent (actions) depends on the state of that world. The effects of the agent's actions are uncertain, but not the current state of the world, which in the MDP framework is always unique and known. Formally, MDPs are characterized by the tuple

- $S$  is the state space which describes the state of the world.
- $A$  is the action space, the possible decisions that the agent may take.
- $T : S \times A$  defines the probability of state transitions, i.e. that action  $a \in A$  in state  $s \in S$  at time  $t$  will lead to state  $s' \in S$  at time  $t + 1$ ,  $\Pr(s_{t+1} = s' | s_t = s, a_t = a)$ .
- $R$  defines the expected (immediate, real-valued) reward for performing action  $a$  in state  $s$ ,  $r(s, a)$ .

According to these definitions, the probability of a state transition only depends on the previous state and the action performed at this stage. This is called the 'Markovian property'. Markov systems have no 'memory' of previous states (although a limited memory effect can often be achieved by enlarging the state space.)

$$\Pr(s_{t+1} = s' | s_t = s, a_t = a) = \Pr(s_{t+1} = s' | s_t = s, a_t = a, s_{t-1} = s'', a_{t-1} = a'', \dots, s_0 = s''', a_0 = a''')$$

The goal of the agent is to choose actions which fulfill its task as well as possible. A policy is a function,  $\pi : S \rightarrow A$ , that specifies, for each state an action to be taken, if the strategy is deterministic. Therefore, we need to find the optimal policy  $\pi^*(s)$  that selects the best action (the one that will optimize the long-term reward) to perform at each state  $s$ .

For achieving this objective, we need to measure the long-term reward received. Considering an infinite lifetime for the agent, we can use the value function,  $Q$  (function of state-action pairs that estimate how good it is to perform a given action in a given

state).  $Q^\pi(s, a)$  is the state-value function for a policy  $\pi$  and indicates the expected return starting from  $s$ , taking action  $a$  and after that following policy  $\pi$ .

$$Q^\pi(s, a) = E\pi\{R_t | s_t = s, a_t = a\} = E\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\}$$

where  $\gamma, 0 \leq \gamma \leq 1$ , is a geometric discount factor that is used to penalize rewards received by future actions. The further ahead the action is performed the smaller the reward taken account of to estimate the  $Q$  function.

There are exact solutions to the problem of computing an optimal policy, but they become intractable for MDPs with more than a few states. There are however various reinforcement learning methods for finding approximations to optimal policies  $\pi^*$ , that can be derived from the optimal value function,  $Q^*$ . They are usually based on trial-and-error search and estimating these delayed rewards.

### 3.2 Q-learning algorithm

The Q-learning [19] is a reinforcement algorithm that directly approximates  $Q^*$ , the optimal action-value function, independent of the policy being followed. Its simplest form is defined by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

The Q-learning algorithm goes as follows:

Initialize  $Q(s, a)$  arbitrarily

Repeat (for each episode):

    Initialize  $s$

    Repeat (for each step of episode):

        Choose  $a$  from  $s$  using policy derived from  $Q$

        Take action  $a$ , observe  $r, s'$

$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

$s \leftarrow s'$

    until  $s$  is terminal

The algorithm assumes that meaningful rewards are associated with transitions to final states (e.g. 100 for a successful ending, -100 for failure). All other transitions have some other value, typically zero. The effect of the Q-learning algorithm is to gradually spread the rewards backwards through the transitions until the distribution becomes stable (i.e. no further changes happen, or changes fall below some threshold.)

## 4 SDS-MDP

We can specialize the MDP framework in order to model a task-oriented spoken dialogue system (SDS). At each dialogue turn  $t$  the dialogue manager has to choose an

action  $a_t$  (speech/dialogue act) according to its current policy  $\pi$  and its internal state  $s_t$ , trying to achieve its predetermined task. The dialogue state is built in such a manner as to reflect the amount of information that the system has at this moment. Following the action will be an update to the next state  $s_{t+1}$ . A more completed and detailed example for a spoken dialogue system using MDPs will be shown in the next section.

## 4.1 Previous work

In the last ten years, there have been many attempts to model a spoken dialogue system as a MDP, for instance [12, 18, 15]. Moreover, some hybrid models have been developed, [7], in which reinforcement learning and supervised learning have been applied to find an optimal policy, tasks such as the flight booking domain.

For dialogue optimization on large spaces, multiple Semi-MDP [20] and hierarchical reinforcement learning methods [3] have been proposed. They are based on decomposing the dialogue manager into sub-tasks.

Finally, there are other systems that use reinforcement learning as Elvis [18] and NJFUN, [22]. In the first project, an agent for real time spoken language access to email from phone was developed. They applied Q-learning to try to compute the utility of the actions in a given state. In NJFUN, the telephone system provides information to the user about activities in New Jersey. A reward function, with positive values for actions that help to fulfill the task, is used for the MDP.

## 5 A DS-MDP for a train information task

We carried out an experimental implementation of a Dialogue System in the MDP framework using the Dihana corpus already mentioned. An extended explanation of the properties of this corpus was given earlier in the section on dialogue simulation. We will explain in detail how we model a Dialogue System using a MDP, using the components previously defined.

### 5.1 Components

**States** ( $S$ ). We can define the states in two different ways:

- formed by combining DAs performed in the last user turn with the vector of data information available at this moment (see SECTION X). Using this definition of state, there are 2733 distinct states observed in the corpus, e.g. encoded state 100\_156\_1184063, corresponds to the situation of dialogue when user has performed in the last turn the following DAs:

`(U:Afirmation:Departure-hour,Train-type:nil),`

`(U:Question:Departure-hour,Train-type:nil)` and the data

state is 1184063 that in binary is 0100100001000100111111, where 1 indicates that there is the presence of a slot-filling piece of data. There will be also the possibility that this state is formed by the second part of data vector or combining with the first.

- formed only by data information that is available at the moment. With this definition there are 446 states. e.g 1184063.

**Actions** ( $A$ ), formed by DAs performed in the system turn. There are 215 distinct actions, e.g. `_37_60_40` that corresponds to  $(S:Waiting:Nil:Nil)$ ,  $(S:Answer:Price:Clase,Price)$ ,  $(S:Consult:Nil:Nil)$

**Transition function** ( $\Pr(s_{i+1} = s' | s_i = s, a_i = a)$ ), function that defines the probability to move from a specific state that dialogue is at a moment  $i$  to a new state at the moment  $i + 1$  by performing action  $a$ . We estimate it in the following way:

$$\#(s, a, s') / \sum_{s''} \#(s, a, s'')$$

where  $\#(s, a, s')$  is the number of times that from state  $s$  has moved to state  $s'$  when appears action  $a$ . This function can be non-deterministic, for example, using the first definition of state, from state 101\_1048831, when action 23 appears, the dialogue moves to state 114\_225\_3145983 once and twice to state 161\_1048831. Then,  $\Pr(s_{i+1} = 114_225_3145983 | s_i = 101_1048831, a_i = 23) = 1/3$  and  $\Pr(s_{i+1} = 161_1048831 | s_i = 101_1048831, a_i = 23) = 2/3$

**Initial states** this set is composed by the states that appear at the beginning of the dialogue, as our model starts with a user turn (dialogues really always start with the same deterministic system turn). This set is needed by the algorithm used to estimate the value function.

**Reward function** ( $E(r_{i+1} | s_i = s, a_i = a, s_{i+1} = s')$ ), is the expected value of the next reward when at state  $s$  is performed action  $a$  and the next state is  $s'$ . The way to estimate it is explained in the following section.

## 5.2 Implementation of Q-learning algorithm

The general algorithm was explained in the previous section. Here we will go into some specific details of the implementation showing some practical examples.

1.  $r(s, a, s')$  is initialized in the following way:

$$r(s, a, s') = \begin{cases} X & \text{if } s' \text{ is a final state} \\ 0 & \text{if } s' \text{ is not a final state} \end{cases}$$

where  $X > 0$ . Only the  $r(s, a, s')$  where  $(s, a, s')$  appears in the corpus will have a value: the rest are not defined. E.g. in our corpora, all the values  $r(s, a, s') = 0$  but

not  $r(105\_ - 2, 1, -1\_ - 2), r(106\_ - 2, 1, -1\_ - 2), r(106\_ - 2, 1, -1\_ - 2), r(97\_ - 2, 1, -1\_ - 2), r(-1\_ - 2, 1, -1\_ - 2)$  than are equal to 100. 105 corresponds to (U:Closing:Nil:Nil), 106 is (U:Undefined:Nil:Nil), 97 is (U:Affirmation:Day:Nil), the state of data vector  $-2$  corresponds to a special final state and  $-1$  is a special final DAs that indicates the end of dialogue.

2. Initialization of  $Q(s, a) = 0.0, \forall s, \forall a$ , selection of  $\gamma$  (e.g.  $\gamma = 0.9$ ) and number of dialogues (episodes) to perform (e.g 50.000).
3. For each episode, a starting state  $s$  is selected randomly between the possible initial states (e.g  $s = 106\_255$ )
4. Repeat (until we get to a final state -e.g. final state in our model is  $-1\_ - 2$ ):
  - an action  $a$  and state  $s'$  are randomly selected from all  $(s, a, s')$  that appears at corpus. (e.g.  $a = 21, s' = 101\_255$ )
  - estimation of  $\alpha = 1/(visits(s, a) + 1)$ , where  $visits(s, a)$  is the number of times that state  $s$  and  $a$  have been visited. Later 'visits' is updated  $visits(s, a) + 1$ . (e.g.  $alpha = 1/(1+visits(106\_255, 21)) = 1/1(1+0) = 1/visits(106\_255, 21) + 1 = 2$ )
  - maximum value of the state-value function is selected from current values of  $Q$  and starting from the new state achieved  $s'$ .  $m = \max_{a'} Q(s', a')$  (e.g.  $m = \max(Q(101\_255, 41)) = 0$ , in this example there is only this transition)
  - $Q$  is updated as follows:

$$Q(s, a) = (1 - \alpha) * Q(s, a) + \alpha(r(s, a, s') + m)$$

$$(e.g. Q(106\_255, 21) = (1 - 1)Q(106\_255) + 1(r(106\_255, 21, 101\_255) + 0) = 0$$

### 5.3 Experiments

Evaluating the performance of a Dialogue System is always a complicated task. For this reason, following the previous implementation, some preliminary experiments have been performed, using different configurations for states.

- the MDP models have been tested in the task of prediction of the next system turns, using real data (the Dihana corpus). Experiments using cross validation with 80% training and 20% test have been performed and the results have been compared with smoothed n-gram models. For each user turn the system turn with the highest reward for MDP and the one with highest probability for n-gram was chosen using information acquired in previous turns. In both models (n-grams and MDP) this previous information is gathered from the real corpus data. For these models, states were a combination of DAs and data information that is composed by attributes and frames, see in 2.

Table 14: Average of the percentage of correctly predicted system turns for cross-validation experiments.

Attributes	Frames	2-grams	3-grams	4-grams	MDP
Y	N	29.5	28.6	27.9	22.87
N	Y	33.6	32.8	32.3	22.65
Y	Y	24.5	24.1	22.5	23.11

## 5.4 Discussion and future work

A Markov Decision Process Model has been developed to work as a dialogue manager. The distributions of the model were estimated from real data using the Q-learning algorithm. Different combinations of representation of the knowledge information (data state) were used.

A preliminary cross-validation experiment of the evaluations have been performed trying to predict the system's turn. Results have been compare with n-grams models. Clearly, the MDP models are in disadvantage with 3-grams and 4-grams models because they use a longer history than the MDP model (only previous turn). Additionally, the robustness and the smoothing of the n-grams can be the reason for their superior performance.

Future work on this must include a more exhaustive evaluation of the models. Previous approaches have commonly compared the average reward function of the policy obtained using RL techniques with one hand made policy [24]. However, for doing this experiment automatically, the user can be simulated using of the techniques described previously.

It is important that this probabilistic model be integrated in a dialogue system prototype in the near future. In this framework it will then be possible to evaluate by real users, that is one of the principal criticism that some experts [14] have about this Reinforcement Learning models applied to dialogue systems. Additionally, more powerful models such as POMDP can be also used. The main problem will be estimate the large number of parameters that these models have.

## 6 Integrating Decision Theoretic Mechanisms into the Senior Companion Demonstrator

In previous sections we have described the current state of the art in using decision theoretic mechanisms to make dialogue decisions. However, the vast majority of this work has been concerned with simple task-oriented dialogues in which a fixed number of pieces of information - e.g. time and date of travel, origin city, destination city - are required from the user in order to send a request to a database and successfully complete

the dialogue. The dialogues that the Senior Companion demonstrator are intended to deal with, however, differ from these in various ways. It is true that one aspect of the dialogues (as seen in the Companions corpus) is concerned with gathering information (SC) about particular photographs (e.g. who is in the picture, where was it taken, etc.) and thus can be seen as a type of task-oriented dialogue, but the conversation can also vary topic, e.g. by retrieving news items relevant to the current discussion, or in an intended extension, by retrieving other kinds of relevant information from the web. Thus the techniques we have been discussing may need adaptation and elaboration, if this is possible, for us to incorporate them into the demonstrator. In this section we sketch ways in which this might be approached.

### 6.1 Dialogue Action Frames (DAFS)

In the SC demonstrator the dialogue mechanism is based on Dialogue Action Frames (DAFs) [1, 5]. An example of a DAF is shown in Figure 7. Individual DAFs are described as a form of Augmented Transition Network, i.e. a Recursive Transition Network (RTN) with arbitrary tests to enable transitions between states. Tests are usually based on some property of the user input. As well as a test, there is an action associated with each transition. Actions can result in the system producing an utterance, or advancing the dialogue state in some other way, or an action can invoke another DAF, pushing the next state onto a stack so that control returns to this state when this other DAF is completed. This is the recursive part of an RTN. Also associated with each transition is a priority score, ranging between 0, the default, and 10. If in a particular state there is more than one possible transition going out (i.e. more than one test returns 'true') this score is used to decide which transition to take. Notice that this already constitutes a simple kind of decision theory mechanism, although it is hand-coded, and the priorities are fixed ahead of time instead of being able to change in response to the progress of the dialogue.

As an informal illustration, consider the task-oriented component of the current SC dialogues: obtaining information about photographs. There will be a 'photo' DAF which will contain arcs corresponding to the subject, date, and location of the photograph. Each of these arcs will invoke more specialised DAFs 'ask for subject', 'ask for date' etc. These DAFs may of course also form components of other DAFs on different topics.

As well as DAFs having the ability to invoke other DAFs, the whole process is governed by a Control Structure which takes care of the stack mechanism, and also checks at every utterance for a possible change of topic. This is currently done by looking for keywords associated with individual DAFs: if these keywords indicate that the current DAF is not the most appropriate one, control is switched to that associated with the keywords.

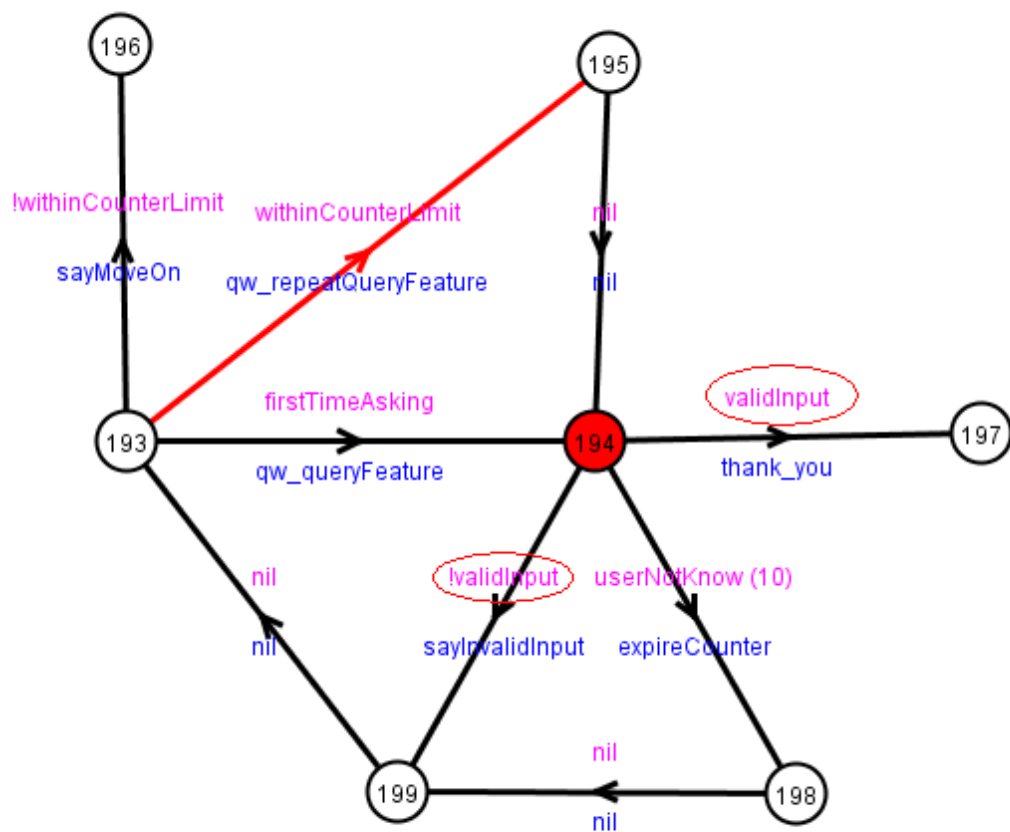


Figure 7: General DAF for validity checking (from [1])

## 6.2 The Integration of Decision Theoretic Mechanisms in the DAF Dialogue Manager

How could we incorporate some decision theoretic mechanisms into a DAF-based dialogue manager? We pointed out that the ‘priority score’ is already a simple, hand-coded, form of decision mechanism. So one possibility would be to try to assign these scores automatically as the result of some overall policy optimisation mechanism, like Q-learning. Of course, to do that, we would need training data, real or simulated. Furthermore, there is a technical issue to be resolved: Q-learning in the context of MDPs, or other mechanisms that are effectively finite state, is well understood. But RTNs are not finite state: taking a transition can involve entering and then exiting a subnetwork, perhaps recursively. In this situation it is not clear that a reward can be consistently assigned to a transition which invokes a subnetwork, because the effects may be different each time this transition is taken. Perhaps an alternative would be for each subnetwork to return the total reward accrued while it is being traversed: this would be the (variable) reward associated with the relevant transition. But then an adjustment would have to be made to the Q-learning algorithm so that these transitions could be correctly handled during the learning phase.

At present, we have only a small amount of SC training data: probably not enough to be able to build a user model and therefore generate simulated dialogues. An alternative, therefore, ignoring the RTN complication for the time being, is to start with hand-assigned priority scores, and to adjust them using a version of an on-line reinforcement learning procedure ([21]) while the dialogue system is actually in use. One way of implementing this would be as follows. It assumes that there is some way of evaluating a state: in Q-learning, this evaluation function (E) would be the reward associated with the highest valued transition out of that state, but it could in principle be almost any property of that state.

In state X, choose the action A with the highest priority score S, go to the indicated state Y, and get an immediate reward R. Update the priority score of A according to a version of the temporal difference learning rule:

$$Priority(A) \leftarrow Priority(A) + \alpha(R + \gamma E(Y) - Priority(A))$$

where  $\gamma$  is between 0 and 1 and is the usual discounting function for future actions and  $\alpha$  is a positive learning-rate parameter.

Choosing an action A is, in our context, producing the next system utterance, and getting an immediate reward we can associate with some property of the next user utterance. States are states of the DAF. There are various choices for E and R: perhaps the simplest would be to use the Q-learning rule for E (i.e. the value of highest priority score out of that state). Possible choices for R might include the information content of the utterance (does it fill a slot needed for our task?), or some measure of the ‘positivity’

of the user towards the entities referred to, or a measure of the emotional state of the user (happy, sad) as suggested by the utterance. Of course, when this kind of online reinforcement learning mechanism is usually used (e.g. in robotics) it is assumed that there will be many thousands of experiences leading eventually to a rational overall allocation of priority scores. It is not clear how well in practice the mechanism might perform with fewer numbers of trials.

It is easier to see how to add a similar decision theoretic step to the Control Structure. This currently checks each utterance to see whether a change of topic has occurred. It would be easy to extend the range of properties that were checked: for example, if it is possible to detect the emotional state of the user, that would again be useful information that might also suggest a change of topic, or a change of register. Similarly, detecting 'sentiment' strongly positive or negative attitudes towards an entity referred to during the dialogue might trigger a change of topic. Knowing how confident the speech recogniser or the natural language processing component of the dialogue system was about the input just processed would also be useful: low confidence would increase the value of confirming or checking. However, it is less easy to see how an online reinforcement learning method could be applied here, since there is no clear notion of a change of state, unless we regard 'being somewhere in DAF X' and 'changing to DAF Y' as states. Nor is there a clear notion of reward, unless it can be immediately detected whether changing the topic was the right thing to do. If it is possible to do both of these things, then we could use our online reinforcement learning method to induce a what would effectively be a Markov Decision Process whose states corresponded to whole DAFS. Alternatively, it might be simpler to just treat this as a probabilistic choice, conditioned on these various parameters:

$$P(\text{changetopic} \mid \text{keywords}, \text{sentimentscore}, \text{emotionalscore})$$

The probabilities could be initially estimated and then refined online in the light of experience.

## References

- [1] R. Catizone, S. Worgan, Y. Wilks, and W. Cheng. A multimodal conversational companion for reminiscing about images. In *Proc. of Fourth International Workshop on Human-Computer Conversation*, Bellagio, Italy, 6th to 7th October 2008.
- [2] P. Clarkson. Statistical language modeling using the cmu-cambridge toolkit. In *Proceedings of ESCA Eurospeech 1997*, pages 2707–2710, 1997.
- [3] H. Cuayahitl, S. Renals, O. Lemon, and H. Shimodaira. Hierarchical dialogue optimization using semi-markov decision processes. In *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, Antwerp, Belgium, 2007.

- [4] W. Eckert, E. Levin, and R. Pieraccini. User modelling for spoken dialogue system evaluation. In *Proceedings of 1997 IEEE automatic speech recognition and understanding workshop*, pages 80–87, Santa Barbara, California, 1997.
- [5] D. Field, S. Worgan, N. Webb, M. Hepple, and Y. Wilks. Automatic induction of dialogue structure from the companions dialogue corpus. In *Proc. of Fourth International Workshop on Human-Computer Conversation*, Bellagio, Italy, 6th to 7th October 2008.
- [6] K. Georgila, J. Henderson, and O. Lemon. Learning user simulations for information state update dialogue systems. In *Interspeech/Eurospeech: the 9th biennial conference of the International Speech Communication Association*, Lisbon, Portugal, 2005.
- [7] J. Henderson, O. Lemon, and K. Georgila. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Edinburgh, Scotland, 1st August 2005.
- [8] J.M.Benedí, E.Lleida, A. Varona, M.J.Castro, I.Galiano, R.Justo, I. López, and A. Miguel. Design and acquisition of a telephone spontaneous speech dialogue corpus in spanish: Dihana. In *Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 1636–1639, Genova, Italy, May 2006.
- [9] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.
- [10] Daniel Jurafsky. Johns hopkins lvcsr workshop-97 \*switchboard\* \*discourse\* \*language\* modeling project final report.
- [11] Murphy K. An introduction to graphical models. Technical report, 2001.
- [12] E. Levin, R. Pieraccini, and W. Eckert. Learning dialogue strategies within the markov decision process framework. In *Proceedings IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 72–79, Santa Barbara, California, 17th December 1997.
- [13] Jordan M. *Learning in Graphical Models*. MIT Press, 1998.
- [14] T. Paek. Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *Proc. of Interspeech-06 Workshop on Dialogue on Dialogues - Multidisciplinary Evaluation of Advanced Speech-based Interactive Systems*, Pittsburgh, Pennsylvania, 17th to 21st September 2006.

- [15] O. Pietquin and Dutoit T. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):589–599, 2006.
- [16] S.G. Pulman. Conversational games, belief revision and bayesian networks. In *CLIN VII: Proc. of 7th Computational Linguistics in the Netherlands meeting*, pages 1–25, Eindhoven, 15th November 1996.
- [17] J. Schatzmann, B. Thomson, and S. Young. User modelling for spoken dialogue system evaluation. In *Proceedings of 8th SIGDial Workshop on Discourse and Dialogue*, Antwerp, Belgium, 2007.
- [18] S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Artificial Intelligence*, 16:105–133, 1999.
- [19] R. Sutton. *Reinforcement Learning*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.
- [20] R. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.
- [21] R.S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224, Austin, Texas, 21st to 23rd June 1990. Morgan Kaufmann.
- [22] M. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Artificial Intelligence Research*, 12:387–416, 2000.
- [23] M. Walker, B. Pellom, J. Polifroni, A. Potamianos, P. Prabhu, A. Rudnicky, S. Seneff, and D. Stallard. Darpa communicator dialog travel planning systems: The june 2000 data collection. In *Eurospeech 2001*, pages 1371–1374, Aalborg, Denmark, 3rd to 7th September 2001.
- [24] J. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.*, 21(2):393–422, 2007.
- [25] Steve Young. Probabilistic simulation of human-machine dialogues. In *Proc. ICASSP*, pages 1217–1220, Istanbul, 2000.