CLASSiC

D3.2: Probabilistic user simulations for training in the (PO)MDP framework including the simulation of grounding in TownInfo dialogues

Olivier Pietquin, Stéphane Rossignol, Michel Ianotto, Paul Crook

Distribution: Public

CLASSiC
Computational Learning in Adaptive Systems for Spoken Conversation
216594 Deliverable 3.2

October 2009

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	216594
Project acronym	CLASSiC
Project full title	Computational Learning in Adaptive Systems for Spoken Conversation
Instrument	STREP
Thematic Priority	Cognitive Systems, Interaction, and Robotics
Start date / duration	01 March 2008 / 36 Months

Security	Public
Contractual date of delivery	M18 = August 2009
Actual date of delivery	October 2009
Deliverable number	3.2
Deliverable title	D3.2: Probabilistic user simulations for training in the (PO)MDP framework including the simulation of grounding in TownInfo dialogues
Type	Prototype
Status & version	final 1.0
Number of pages	21 (excluding front matter)
Contributing WP	3
WP/Task responsible	SUPELEC
Other contributors	UEDIN, UCAM
Author(s)	Olivier Pietquin, Stéphane Rossignol, Michel Ianotto, Paul Crook
EC Project Officer	Philippe Gelin
Keywords	User Modeling

The partners in CLASSiC are:

Heriot-Watt University	HWU
University of Cambridge	UCAM
University of Geneva	GENE
Ecole Superieure d'Electricite	SUPELEC
France Telecom/ Orange Labs	FT
University of Edinburgh HCRC	EDIN

For copies of reports, updates on project activities and other CLASSiC-related information, contact:

The CLASSiC Project Co-ordinator:
Dr. Oliver Lemon
School of Mathematical and Computer Sciences (MACS)
Heriot-Watt University
Edinburgh
EH14 4AS
United Kingdom
O.Lemon@hw.ac.uk
Phone +44 (131) 451 3782 - Fax +44 (0)131 451 3327

Copies of reports and other material can also be accessed via the project's administration homepage, `http://www.classic-project.org`

Contents

Executive summary

This document explains the work done in the framework of task 3.2 and the insights of the deliverable D3.2. This deliverable is a prototype of a new user simulation method based on dynamic Bayesian networks. The theory is first explained. Some experimental results are then described. The prototype is developed in C++ and has been interfaced with the DIPPER dialogue manager designed by the University of Edinburgh and written in JAVA. User simulation has become an important trend of research in the field of spoken dialogue systems because collecting and annotating real interactions with users is often expensive and time consuming. Yet, such data are generally required for designing and assessing efficient dialogue systems. The general problem of user simulation is thus to produce as many as necessary natural, varied and consistent interactions from as few data as possible. In this contribution, we propose a user simulation method based on Bayesian networks (BN) that is able to produce consistent interactions in terms of user goal and dialogue history but also to simulate the grounding process that often appears in human-human interactions. The BN is trained on a database of 1234 human-machine dialogues in the TownInfo domain (a tourist information application). Experiments with a state-of-the-art dialogue system (REALL-DUDE/DIPPER/OAA) have been conducted and promising results are presented.

Publications arising from this work are :

- Olivier Pietquin. Optimising spoken dialogue strategies within the reinforcement learning paradigm. In Mark Elshaw Cornelius Weber and Norbert Michael Mayer, editors, *Reinforcement Learning, Theory and Applications*, pages 239–256. I-Tech Education and Publishing, Vienna, Austria, January 2008.
- Olivier Pietquin, Stéphane Rossignol, and Michel Ianotto. Training bayesian networks for realistic man-machine spoken dialogue simulation. In *Proceedings of the 1rst International Workshop on Spoken Dialogue Systems Technology*, page 4 pages, Irsee (Germany), December 2009.
- Olivier Pietquin. Machine learning methods for spoken dialogue simulation and optimization. In Abdelhamid Mellouk and Abdennacer Chebira, editors, *Machine Learning*, pages 167–184. INTECH, January 2009.

A special session at the InterSpeech 2009 conference (Brighton, UK) has also been organized to promote the CLASSiC project and work done in spoken dialogue simulation.

Chapter 1

Theory

1.1 Introduction

Spoken dialogue systems are now widespread and are in use in many domains (from flight booking to troubleshooting services). Designing such a speech-based interface is usually an iterative process involving several cycles of prototyping, testing and validation. Tests and validation require interactions between the released system and real human users which is very expensive and time consuming. For this reason, user simulation has become an important trend of research during the last decade. User simulation can thus be used for performance assessment [1, 2] or for optimisation purpose [3, 4, 5], for example, when optimizing dialogue strategies with reinforcement learning. User simulation should not be confused with user modelling. User modelling is generally used by a dialogue system for internal purposes such as knowledge representation and user goal inference [6, 7] or natural language understanding simulation [8]. The goal of user simulation is to generate a large amount of simulated interactions with a dialogue system and the simulated user is, therefore, external to the system.

Dialogue simulation can occur at several levels of description. In this work, simulated interactions will take place at the intention level (such as in [1, 3, 4, 5]) and not at the signal level as proposed in [2]. An intention is here defined as the minimal unit of information that a dialogue participant can express independently. It will be modeled as dialogue acts. Indeed, intention-based communication allows error modelling of all the parts of the system, including speech recognition and understanding [9, 8, 10]. Pragmatically, it is easier to automatically generate intentions or dialogue acts compared with certainly speech signals, as a large number of utterances can express the same intention.

The simulated user (SU) presented in this contribution is based on Bayesian networks (BN). This model has been chosen for several reasons. First, BN are generative models and can, therefore, be used for inference as well as for data generation, which is of course required for simulation. Second, it is a statistical framework and it can thus generate a wide range of different dialogues, that are statistically consistent. Third, BN parameters can either be set by experts or trained on data. Given that data is often difficult to collect, the introduction of expert knowledge can be very helpful. Finally, a lot of efficient tools for inference and training for BN are freely available.

This contribution builds on previous work [11, 4] where Bayesian networks are used for simulation purpose but also contributes two novel approaches. First, the model has been modified to generate grounding behaviours [12]. The grounding process will be considered here as the process used by dialogue partic-

Table 1.1: Slots in the task, and corresponding possible values

Food	Price range	Area
"italian"	"moderate"	"central"
"indian"	"expensive"	"north"
"chinese"	"cheap"	"south"
		"west"
		"east"

ipants to ensure that they share the background knowledge necessary for the understanding of what will be said later in the dialogue. In practice, that means that the simulated user will react automatically by providing again correct information to the dialogue system if a problem in the information transmission is detected. The ultimate goal of this work is to train dialogue policies that handle such grounding behaviours [13]. Second, the model is trained on actual human-machine dialogues data and tested on a state-of-the-art dialogue system (the REALL-DUDE/DIPPER/OAA environment [14, 15, 16]).

The considered domain is the TownInfo domain, a tourist information task. The task consists in retrieving information about restaurants in a given city. This can be considered as a slot filling task where we consider three different slots : "food", "price_range" and "area". Possible values for these slots are provided in Table 1.1.

The rest of this chapter is organized as follows. In Section 1.2, the proposed model is described in details. In Section 2, experiments are presented. Finally, a conclusion and future works are provided in Section 2.4.

1.2 Description of the Model

The user model proposed here is based on a simplified version of the statistical description of man-machine spoken communication described in [4] and which we describe briefly here.

As depicted on figure 1.1, a task-oriented man-machine dialogue can be seen as a turn-taking process in which a human user and a Dialogue Manager (DM) exchange information through different channels processing speech inputs and outputs (ASR, TTS,...). At each turn t the DM chooses an action a_t according to its internal state s_t and its strategy π_t so as to accomplish the task it has been designed for. These actions can be greetings, spoken utterances (i.e. constraining questions, confirmations, relaxation, data presentation), database queries, dialogue closure etc. They result in a response from the DM environment (i.e. user speech input, database records), considered as an observation o_t, which usually leads to a DM internal state update ($s_t \rightarrow s_{t+1}$). During the interaction, the DM action has been transformed in synthesized speech sys_t if needed. This acoustic signal is mixed up with noise n_t before reaching the user. According to his/her goal (g_t), knowledge (k_t) and understanding of sys_t, the user will produce a spoken utterance u_t also mixed up with noise before reaching the ASR system. Given this description, the interaction can be described by the following probability:

$$P(s_{t+1}, o_t, a_t | s_t, n_t) = \underbrace{P(s_{t+1} | o_t, a_t, s_t, n_t)}_{\textit{Task Model}}.$$

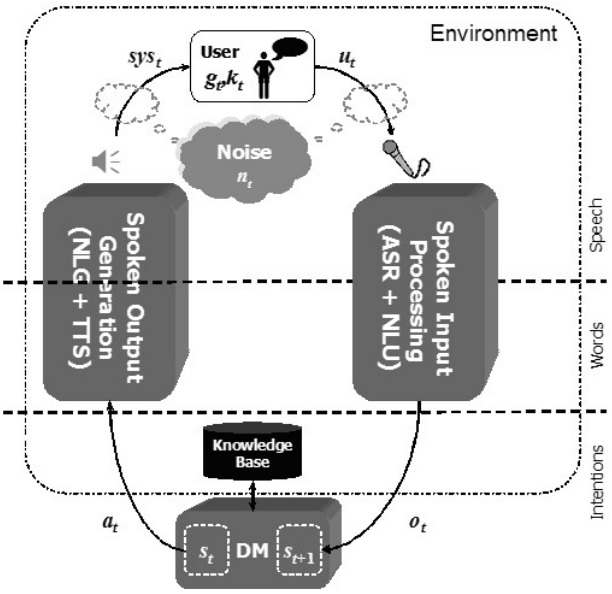

Figure 1.1: Spoken Dialogue Model

$$\underbrace{P(o_t|a_t,s_t,n_t)}_{Environment}.\underbrace{P(a_t|s_t,n_t)}_{DM} \tag{1.1}$$

The factorization of this joint probability includes a term related to the environment processing of the DM intention set (second term). Omitting the t indices, this term can in turn be factored as follows:

$$\begin{aligned} P(o|a,s,n) &= \sum_{sys,k,g,u} P(o,sys,k,g,u|a,s,n). \\ &= \sum_{sys,k,g,u} \underbrace{P(sys|a,s,n)}_{Output\ Processing}.\underbrace{P(o|u,g,sys,a,s,n)}_{Input\ Processing}. \\ &\qquad \underbrace{P(u,g,k|sys,a,s,n)}_{User\ Model} \end{aligned} \tag{1.2}$$

1.2.1 Probabilistic User Model

From the previous section, the user behavior can be probabilistically described by the following probability:

$$\begin{aligned} P(u,g,k|sys,a,s,n) &= \underbrace{P(k|sys,s,n)}_{Knowledge\ Update}.\underbrace{P(g|k)}_{Goal\ Modification} \\ &\quad .\ \underbrace{P(u|g,k,sys,n)}_{User\ Output} \end{aligned} \tag{1.3}$$

To obtain the last equality, the following assumptions were made:

- the user is only informed of the DM intentions a through the system utterance sys,
- if a goal modification occurs it is because the user's knowledge has been updated by the last system utterance.

Equation 1.3 emphasizes the tight relation existing between the user's utterance production process and his/her goal and knowledge, themselves linked together. The user's knowledge can be modified during the interaction according to the system's speech outputs. Yet, such a modification of the knowledge is incremental (it is an update similar to the system state update) and it takes into account the last system utterance (which might be misunderstood, especially in presence of noise) and the previous user's knowledge state. This can be written as follows with k^- standing for k_{t-1}:

$$\begin{aligned} P(k|sys,s,n) &= \sum_{k^-} P(k|k^-,sys,s,n).P(k^-|sys,s,n) \\ &= \sum_{k^-} P(k|k^-,sys,n).P(k^-|s) \end{aligned} \tag{1.4}$$

Although the user's knowledge k^- is not directly dependent of the system state s, we kept this dependency in our description so as to be able to introduce a mechanism for user knowledge inference from system state because it is supposed to contain information about the history of the dialogue.

It is this mechanism that we will use in the following to introduce *grounding* [12] subdialogs in the interaction so as to obtain a good connection between the user's understanding of the interaction and the system view of the same interaction.

1.2.2 Variable Representation

In practice, the use of the proposed framework is difficult without a suitable representation of variables such as u, sys, g or k. According to the intention-based communication paradigm, these variables can be regarded as finite sets of abstract concepts, related to the specific task, that have to be manipulated along the interactions by the SDS and the user. In the CLASSiC project, we opted for an *Attribute-Value* (AV) pair variable representation. Each communicative act is then represented by a set of AV pairs. If we denote $\mathcal{A}$ the set of possible attributes (or slots when in a slot filling task such as the TownInfo task) according to the task, and by $\mathcal{V}$ the set of all possible values the different system and user utterances can be described as follows. The system utterances sys are then split into two variables : the dialog act A_s and a set of attributes denoted $Sys = \{sys^\sigma\} \subset \mathcal{A}$. The user's utterance u is modelled as a dialog act and a set of AV pairs in which attributes belong to $U = \{u^\upsilon\} \subset \mathcal{A}$ and the set of possible values for u^υ is $V^\upsilon = \{v_i^\upsilon\} \subset \mathcal{V}$. The ASR and NLU processes (Input processing) result in an error-prone set of AV pairs c which is part of the observation o. The user's goal $g = GOAL = \{[g^\gamma, gv_i^\gamma]\}$ and the user's knowledge $k = KWN = \{[k^\kappa, kv_i^\kappa]\}$ are also AV pair sets where g^γ and k^κ are attributes and where gv_i^γ and kv_i^κ are values. In the special case of a slot filling task such as the TownInfo task, the attributes are slots which will be denoted s_i.

1.2.3 Bayesian Networks

Equations 1.3 and 1.4 express the behaviour of the user as conditional probabilities. A standard framework to encode conditional (in)dependences is *Bayesian Networks* (BN). A BN also called *Belief Network*

is a directed acyclic graph encoding relationships among variables in a probabilistic framework [17]. In such a graph, nodes represent stochastic variables and arcs represent dependencies between variables. It creates an easy and readily way to encode uncertain expert knowledge of a given domain. A network so constructed also reflects implicit independencies among the variables by the lack of arcs. The network must be quantified by specifying a probability for each variable conditioned on all possible values of its immediate parents in the graph. In addition, the network must include a marginal distribution encoding unconditional probabilities for each variable that has no parent. This quantification is realised by associating a Conditional Probability Distribution (CPD) at each node. If the variables are discrete, this can be represented as a table (Conditional Probability Table: CPT), which lists the probability that the child node takes on each of its different parent set values. Together with the independence assumptions defined by the graph, this quantification defines a unique joint distribution over the variables in the network. Then, exploiting the independencies represented within the graphical structure, the probabilistic inference aiming at computing the probability of any event over this space can be realised. The most common task one wish to solve using BNs is probabilistic inference. That is finding the likelihood of a variable state given some evidence.

1.2.4 BN-based User Model

Roughly, this BN is inspired by the idea that the user's answer to a system dialogue act is influenced by this system act AS, the user goal GOAL and his/her knowledge KNW about the history of the dialogue. The user response can be of several types (INFORM or CONFIRM in this case), but including a particular action such as closing the dialogue. Figure 1.2 shows the details of the SU. For portability purpose and in order to integrate it into the system described in Section 2.1, it has been implemented in C++ using the Smile library (http://genie.sis.pitt.edu/about.html#smile). Smile is a library of C++ classes implementing graphical probabilistic and decision-theoretic models, such as Bayesian Networks, etc.

The node AS corresponds to the system act. The nodes SYSTEM correspond to the slots mentioned in the system act. The system act and the associated attributes are independent variables, thus the corresponding nodes have no parent. In interaction with a DM, the DM's output (in the form of a system act) will provide direct observations of these nodes.

The nodes KNW correspond to the knowledge the SU has concerning the history of the dialogue. There is a knowledge value for each of the slots in the task (3 in this case). Three levels of knowledge (thus 3 possible values) are considered: low, medium and high. These values represent the knowledge of the user about the fact the information about the corresponding slot has been provided to the system. For example, if the user once provided information about the type of food s/he wants, the knowledge about this slot goes from *low* to *medium*. If this information has been provided several times, the knowledge becomes *high*. The knowledge somewhat corresponds to a SU estimate of the dialogue state.

The nodes GOAL correspond to the user goal. They indicate which slots are present in the user goal. The GOAL VALUES indicate the value given to each slot in the goal. This ensures that the SU behaviour will be consistent according to a given goal.

The nodes INFORM and INFORM VALUES correspond to the user act "inform" and its attributes. INFORM correspond to the slots transmitted in the SU utterance, and INFORM VALUES correspond to the set of values associated to the transmitted slots. The nodes CONFIRM and CONFIRM VALUES correspond to the user act "confirm" or "negate". CONFIRM correspond to the slots transmitted in the SU utterance, and CONFIRM VALUES correspond to the set of values (that is to say answer_yes or answer_no) associated to

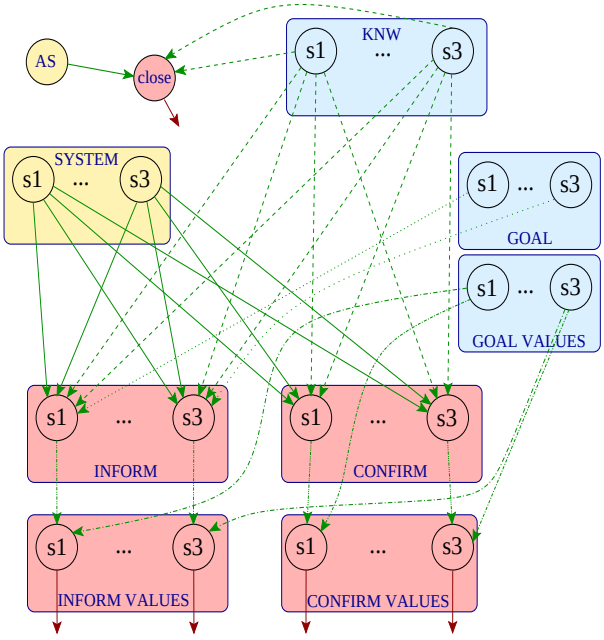

Figure 1.2: Bayesian Network-based Simulated User

the transmitted slots. The node CLOSE indicates whether the Simulated User decides or not to close the dialogue. These nodes are the SU's output variables.

Notice that, if the Dialogue Manager (DM) asks for a slot for which the SU has already a medium or high knowledge value (i.e. a slot for which the SU has already provided information), it is likely that a grounding problem occurred. The main SU described so far is designed to send a value for this slot, with a certain probability, which can be low (or possibly to close the dialogue). Yet, the knowledge value could be used to infer the occurrence of a grounding problem. This is done by adding the grounding component of Figure 1.3 where KNW and GOAL nodes are the internal variables of the Simulated User as explained in the next section.

The whole set of probabilities in the conditional probability tables (CPTs) can not be trained, as this concerns thousands of them (2151, more precisely, here). However, we defined a set of 25 probabilities that generale well and could be learned from a database comprising more than 1000 dialogues (see Section 2.3.3). Those generic probabilities can be of several type :

- a prior probability distribution over attributes representing the probability of each attribute to be provided after a greeting prompt;
- a set of probabilities of the form $p(u^{\upsilon}|s^{\kappa},k,g)$ denoting the probability of the attribute u^{υ} being provided given that the attribute s^{κ} was asked for by the system and given the user's knowledge and the goal;
- a set of probabilities of the form $p(yes|s^{\kappa}=v^{\kappa},k,g)$ denoting the probability of the user confirming the value $s^{\kappa}=v^{\kappa}$ given the user's knowledge and the goal;
- a set of probabilities of the form $p(close|s^{\kappa},k,g)$ denoting the probability of the user closing the

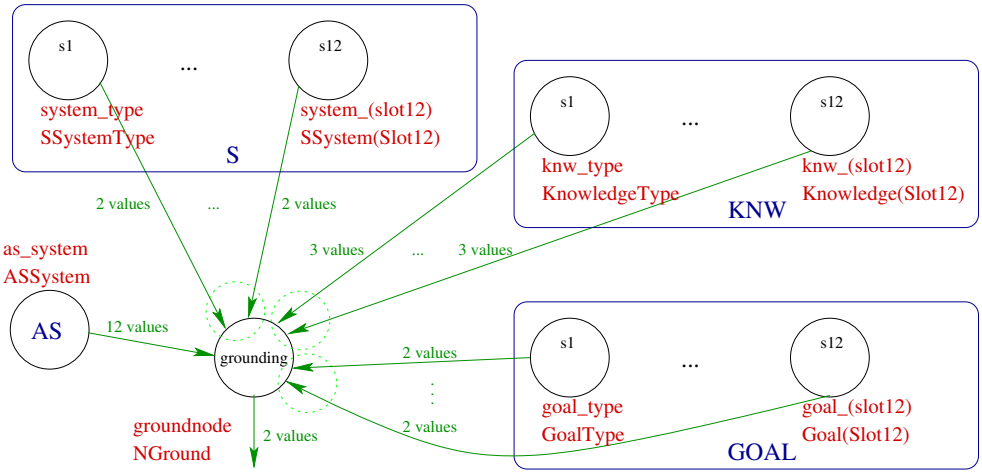

Figure 1.3: The grounding component

dialogue given that the attribute s^{κ} was asked for by the system and given the user's knowledge and the goal.

1.2.5 Grounding Nodes

The grounding components are described in Figure 1.3.

A grounding value is obtained for each slot i by the following process. The attributes and the values of the nodes KNW and GOAL for the slot i are recursively copied in the nodes KNW GROUNDING and GOAL GROUNDING. The inference engine is used afterwards to infer the value for the grounding node, using the system act to complete the evidence. If a grounding problem is detected for the slot i, the SU is forced to provide information concerning this slot.

1.2.6 Inference

User simulation is achieved using probabilistic inference into the BN-based user model. A lot of open source tools are available to perform probabilistic inference given a BN such as the Smile library already mentionned earlier. Yet, we provide here a simple example so as to give the reader a clear understanding on how the user model is practically used.

Let's simulate the user's answer to the greeting. The following evidence will be inserted in the inference engine to obtain the distribution over the attributes and values the user will provide to the system.

As	(KNW)(s_1)	(KNW)(s_2)	(GOAL)(s_1)	(GOAL)(s_2)
greet	*low*	*low*	gv^1	gv^2

Table 1.2: Evidence for the greeting system act

Inference will provide distributions over the possible attributes and values. To simulate a user answer, actual attributes and values will be drawn according to those distributions. Doing so, several behaviours will be possibly simulated for a same system dialog act. These behaviours should be statistically consistent with those seen in the data.

Chapter 2

Experiments

2.1 Interaction of the Simulated User with a Dialogue Manager

Firstly, the Simulated User developed here has been tested using a straightforward DM, this in order to validate the implemented grounding components (see Section 2.2). DM builds its own dialogue state representation, following a simple stereotype-based model.

Secondly, the Simulated User has been interfaced with the spoken dialogue system provided in the REALL-DUDE/DIPPER/OAA environment (see [14], [15] and [16]) (see Section 2.3). This environment primarily aims at training policies by reinforcement learning. Yet, the dialogue policy used for our experiments has been trained independently of the SU presented here and is used for testing purpose only. Dialogues similar to the examples provided in Table 2.1 are obtained. The SU accompanies each hypothesis it sends to the DM with a probability simulating the confidence score which would be provided among other things by the speech recogniser (ASR). The DM can decide to ask for a confirmation if this probability is too low. This explains the "<syst act> confirm(Food='Italian')" and "<user act> affirm" turns in the second dialogue example in Table 2.1. It can be noticed that the SU correctly answered to this system act.

2.2 Grounding Problem Solving – Interaction with a simple DM

A 3-slots task is considered here. The grounding problem solver presented in Figure 1.3 is integrated into the Simulated User system. Twenty thousand dialogues have been simulated, considering two configurations:

- The grounding problem is detected, and nothing is done.
- The grounding problem is detected, and therefore the Simulated User tries to solve it. In other words, if it detects a grounding problem for the slot *i*, it is forced to send its information concerning this slot.

Table 2.1: Dialogue examples, obtained using the Simulated User integrated within the REALL-DUDE/DIPPER/OAA spoken dialogue system environment

First dialogue example:

```
User goal:
  Food:   indian
  rPrice: cheap
  Area:   west

<syst act> hello(Food)
<user act> inform(slot_1='indian')
<syst act> request(Area)
<user act> inform(slot_3='west')
<syst act> request(rPrice)
<user act> inform(slot_2='cheap')
<syst act> close
```

Second dialogue example:

```
User goal:
  Food:   italian
  rPrice: expensive
  Area:   central

<syst act> hello(Food)
<user act> inform(slot_1='italian')
<syst act> confirm(food='italian')
<user act> affirm()
<syst act> request(rPrice)
<user act> inform(slot_2='expensive')
<syst act> request(Area)
<user act> inform(slot_3='central')
<syst act> close
```

	Number of dialogues with grounding problems	number of turns	mean
nothing is done	13449	127808	9.503160
the SU reacts	13391	116952	8.733627

Table 2.2: The mean number of turns necessary to reach the end of the dialogues is provided, this is when nothing is done if a grounding problem is detected (first row) and when the SU reacts to the grounding problem (second row) – whole set of turns taken into account

	Number of dialogues with grounding errors	number of turns	mean
nothing is done	13449	47114	3.503160
the SU reacts	13391	36606	2.733627

Table 2.3: The mean number of turns necessary to reach the end of the dialogues is provided, this is when nothing is done if a grounding problem is detected (first row) and when the SU reacts to the grounding problem (second row) – only informative set of turns taken into account

Finally, the number of turns obtained (this is only considering the dialogues for which a grounding problem occurred), is computed. The dialogue length is a metric for success in this kind of applications and is, therefore, domain specific (in throubleshooting tasks this metric would not be relevant). The results are given in Table 2.2.

These results show that the detection of grounding problems notably reduces the number of turns.. The amount of reduction is 8.1%. Furthermore, if one only takes into account the informative set of turns (i.e. no greetings or closures) and with no inforced six turn minimum (all slots confirmed), a drop of 22% is observed (see Table 2.3.

In that case, the amount of turns drops with 22 %, thanks to the grounding component.

Several other results still need to be examined, for example, the ratio of dialogues for which the three slots have been filled, the number of dialogues for which the final state is different from the original user goal, etc..

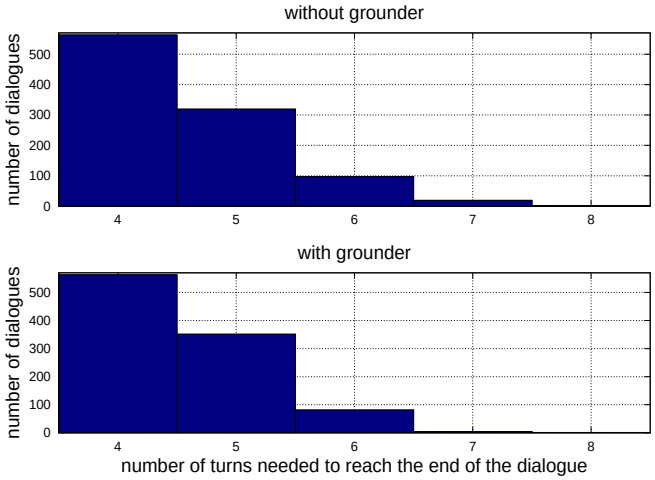

Figure 2.1: Histogram of the number of turns requested to reach the end of the dialogue – top: no grounding problem solver in use – bottom: grounding problem solver in use

Table 2.4: Statistics computed on dialogues obtained with the SU interacting with the REALL-DUDE/DIPPER/OAA spoken dialogue system environment – mean number of turns requested to reach the end of the dialogues; percentage of dialogues for which the end is reached in more than five turns – the BN without grounding component (second row) or the BN with grounding component (third row) is used

	mean	>5 turns
no grounding problem solver	4.578	11.80 %
grounding problem solver	4.529	8.60 %

2.3 Grounding Problem Solving – Interaction with REALL-DUDE/DIPPER/OAA

2.3.1 Statistics

In this section, we present statistics computed on dialogues obtained using the SU described above and the REALL-DUDE/DIPPER/OAA environment. As the task comprises of three slots and as the SU is configured to send no more than one single slot per turn, the minimum number of turns necessary to reach the end of the dialogue is four: one per slot, plus the "Close" turn. Notice that a turn is defined here as a couple <syst act>/<user act> (except for the "Close" act, as it is the DM which decides to stop a dialogue).

Results given on Figure 2.1 and Table 2.4 are promising. The longer dialogues are less numerous. Specifically, the amount of dialogues longer than 5 turns drops by 27.1%.

Table 2.5: Statistics computed on dialogues obtained with the SU interacting with the REALL-DUDE/DIPPER/OAA spoken dialogue system environment – mean number of turns requested to reach the end of the dialogues; percentage of dialogues for which the end is reached in more than five turns – the BN without grounding problem solver (second row) or the BN with a grounding problem solver (third row) is used; and only the dialogues with grounding problems are considered

	mean	> 5 turns
no grounding problem solver	5.254	29.22 %
grounding problem solver	5.069	20.29 %

If one considers only the dialogues with grounding problems, the number of dialogues longer than 5 drops further by 30.6% (see Table 2.5).

2.3.2 Dialogue Example

Out of 1000 dialogues, 496 grounding problems have been detected. In Table 2.6, dialogue examples with a grounding problem are shown. In the first case, the SU detected the grounding problem, and solved it; in the second case, a grounding problem occurred, but the SU did not act and the dialogue consequently deteriorated.. When the DM asks for the second time for the range price, the SU decided to send the third slot (that is to say, the area), as it has already sent once the range price, and not yet the area. However, this action confused the DM.

2.3.3 Statistics on Simulated Dialogues

In this section, are presented statistics computed on dialogues obtained using two versions of the SU presented in this work and the REALL-DUDE/DIPPER/OAA environment. As previously mentioned, the task comprises three slots and as the SU is configured to send no more than one slot per turn, the minimum number of turns necessary to reach the end of the dialogue is four.

The first version of the SU is obtained using heuristically determined values for the BN parameters. The second version is obtained considering learned parameters. The database used for training contains 1234 dialogues and parameters are learned by a maximum likelihood method (user goals are provided). It provides much more complex dialogues than the proposed task. For instance, twelve slots in total are considered. The values for some of these slots are requested by the DM, such as “type”, “food”, etc., implying thus a “Request” system act and an “Inform” user act; others are requested by the User, such as “address”, “phone”, etc.; the “price range” can be requested by both sides. Furthermore, more than ten different system and user acts are considered (see [18] for an exhaustive list). In the current task, less acts are considered, as noticed in Section 1.2. Finally, more than one slot and more than one act can be presented during each turn, both considering the DM turns and the SU turns. Thus, the database needed to be carefully labelled in order to provide useful data for the purpose of this approach. The database has been described in more details in [19] and [20], where it has been used for training dialogue management strategies.

Table 2.6: Dialogue examples, obtained using the Simulated User integrated within the REALL-DUDE/DIPPER/OAA spoken dialogue system environment; top: a grounding problem is detected and solved; bottom: there is a grounding problem, which is not solved by the SU

First dialogue example:

```
 User goal:
   Food:   italian
   rPrice:   cheap
   Area:   central

<syst act> hello(Food)
<user act> inform(slot_1='italian')

<syst act> confirm(Area='central')
=> grounding error detected for the slot_3 (the DM,
   understood something wrong); problem solved by the SU
<user act> inform(slot_3='central')

<syst act> request(rPrice)
<user act> inform(slot_2='cheap')

<syst act> close
```

Second dialogue example :

```
 User goal:
   Food:   italian
   rPrice:   cheap
   Area:   central

<syst act> hello(Food)
<user act> inform(slot_1='italian')

<syst act> confirm(Area='central')
=> grounding error detected for the slot_3 (the DM,
   understood something wrong); the SU does not detect
   it and thus sends its information about a slot
<user act> inform(slot_2='cheap')

<syst act> request(rPrice)
<user act> inform(slot_3='central')

<syst act> confirm(rPrice)
<user act> bye

...
```

Table 2.7: Statistics computed on dialogues obtained with the SU interacting with the REALL-DUDE/DIPPER/OAA spoken dialogue system environment – mean number of turns requested to reach the end of the dialogues; maximum number of turns; minimum number of turns; percentage of dialogues for which the end is reached in four turns; percentage of dialogues for which the end is reached in less than nine turns – the heuristic BN (second row) or the trained Bayesian Network (third row)

	mean	max	min	4 turns	< 9 turns
heuristic BN	4.969	21	4	93.20 %	93.40 %
trained BN	4.577	9	4	58.00 %	99.90 %

Table 2.7 (second row) shows the results obtained using the heuristic BN. One thousand of dialogues have been simulated. Figure 2.2 (left side)gives the histogram of the number of turns required to reach the end of the dialogues. It can be seen that most of the dialogues indeed are carried out into four turns, as expected (93.4 %). This is due to the fact that the heuristic BN has been especially designed to obtain dialogues that are as short as possible.

Table 2.7 shows the results obtained using the trained BN. One thousand of dialogues have been simulated. Figure 2.2 (right side) gives the histogram of the number of turns requested to reach the end of the dialogues. The dialogues are significantly longer than when using the heuristic BN. This can not be seen considering the mean number of turns, but considering the percentage of dialogues which needed exactly four turns to reach their end: this percentage drops from 93.2 % to 58.0 %. However, it is promising to observe that the mean number of turns and the percentage of dialogues which needed less than nine turns to reach their end have improved using the trained BN. However, very promisingly, the mean number of turns and the percentage of dialogues which needed less than nine turns to reach their end are significantly better using the trained BN, and furthermore it can be noticed that the very long dialogues (more than nine turns requested), indicating some deep misunderstanding between the DM and the SU, have completely disappeared. This indicates that the dialogues obtained with the trained BN are much more natural, at least from a DM point of view, than the dialogues obtained with the heuristic BN. This distribution is actually more in agreement with the data which shows again the naturalness of the simulated dialogues.

In addition, it can be noticed as well, in Figure 2.2, that the REALL-DUDE/DIPPER/OAA environment prevents dialogues to be longer than 21 turns. Table 2.8 presents the number of turns needed per slot, respectively when the heuristic BN is used keeping the longest dialogues, when the heuristic BN is used not keeping the longest dialogues as they only reflect the DM stopping condition, when the trained BN is used, and considering the database. Clearly, the trained BN provides more realistic dialogues, in terms of number of turns needed for each slot.

2.4 Conclusion and Future Work

In this contribution, a user simulation model based on Bayesian networks is proposed to simulate realistic human-machine dialogues at the intention level, including grounding behaviours. Our goal was to determine that the grounding process occurs frequently and attempt to simulate it. This is done by comparing

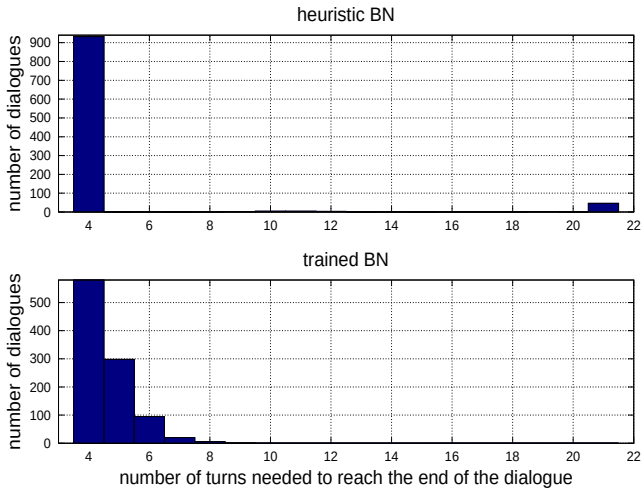

Figure 2.2: Histogram of the number of turns requested to reach the end of the dialogue – top: the heuristic BN is used – bottom: the trained BN is used

Table 2.8: Number of turns needed per slot, respectively when the heuristic BN is used the longest dialogues being kept, when the heuristic BN is used the longest dialogues being not kept, when the trained BN is used, and considering the database

	heuristic BN with longest dialogues	heuristic BN without the longest dialogues	trained BN	database
mean number of turns per slot	1.656333	1.398672	1.525667	1.566179

the number of turns required to reach the end of a dialogue using different configurations of the proposed simulation framework. Several directions for future work are furthermore envisioned.

Firstly, one of the goals of developing this Simulated User is training dialogue policies within a reinforcement learning paradigm. Secondly, some preliminary results concerning the interaction of the presented SU with an independently developed DM are shown. It is planned in the near future to have the SU interact with the spoken dialogue system provided in the REALL-DUDE/DIPPER/OAA environment in a much more extensive and systematic way. This will allow a comparision of the number of turns obtained with the BN-based Simulated Users to the number of turns obtained with human users and analysis the corresponding task completion scores, etc.. It will also allow the trained policies to be used in the POMDP engines implemented in the REALL-DUDE/DIPPER/OAA environment. Thirdly, we would like to enable the BNs to dynamically learn their parameters, thus improving the quality and naturalness of the simulated dialogues as users are really interacting with the system and to be able to retrain policies between real interactions.

Appendix A : Implementation - JNI Interface for DIPPER

Implementation for DIPPER (JNI + OAA) is available on SUPELEC's foundry : http: foundry.supelec.fr

System actions	User actions	Slot values
- greet - askASlot - implConfAskASlot - explicitConfirm - closeDialogue	- provide_info - answer_yes - answer_no - bye - null	- slot_1 : Indian, Chinese, Italian. - slot_2 : moderate, cheap, expensive. - slot_3 : central, north, east, south, west.

Table 9: Actions

The BayesianUserSimulation receives system actions from the dialogue manager and sends it user actions with slot values. Table 9 contains the set of system actions, user actions and slot values.

JAVA methods implemented by the BayesianUserSimulation OAA module are depicted in Table 10 and the corresponding C++ functions are shown in Table 11.

JAVA methods defined in the file `BayesianUserSimulation.java`			
Method name	**Arguments**	**Return value**	**Comments**
`BayesianUserSimulation`	-	`void`	The constructor
`initialiseOAA`	-	`boolean`	Connect to the OOA facilitor.
`processMessage`	• `IclTerm goal` • `IclList params` • `IclList answers`	`boolean`	Process a message from the OAA facilitor
`processGetUserAnswer`[1]	• `IclStruct goal` • `IclList answers`	`boolean`	Get the answer from the simulated user
`processGetGoal`[2]	• `IclStruct goal` • `IclList answers`	`boolean`	Get the goal from the simulated user

Table 10: Java Methods

C functions defined in the file `ProcessGetUserAnswerBN.cpp`			
Method name	**Arguments**	**Return value**	**Comments**
`GetUserAnswer`	• `Jstring j_system_action` • `jstring j_num_slot` • `jstring j_slot_value` • `jint j_nb_answers` • `jobjectArray listUserAct` • `jobjectArray listNumSlot` • `jobjectArray listSlotValues` • `jobjectArray listProbValues` • `jintArray nbSlots`	`void`	Return the answer from the Bayesian network
`GetUserGoal`	• `jobjectArray jlistSlotGoal` • `jobjectArray jlistValueGoal`	`void`	Return the goal of the simulated user

Table 11: C++ functions

Bibliography

[1] W. Eckert, E. Levin, and R. Pieraccini. User Modeling for Spoken Dialogue System Evaluation. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 80–87, 1997.

[2] Ramn López-Cózar, Zoraida Callejas, and Michael F. McTear. Testing the performance of spoken dialogue systems by means of an artificially simulated user. *Artificial Intelligence Review*, 26(4):291–323, 2006.

[3] E. Levin, R. Pieraccini, and W. Eckert. A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. In *IEEE Transactions on Speech and Audio Processing*, volume 8, pages 11–23, January 2000.

[4] Olivier Pietquin and Thierry Dutoit. A Probabilistic Framework for Dialog Simulation and Optimal Strategy Learning. In *IEEE Transactions on Audio, Speech, and Language Processing*, volume 14, pages 589–599, 2006.

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

[6] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the 14th Conference on Uncertainty in Artifical Intelligence*, July 1998.

[7] H. Meng, C. Wai, and R. Pieracinni. The Use of Belief Networks for Mixed-Initiative Dialog Modeling. In *Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP)*, October 2000.

[8] O. Pietquin and T. Dutoit. Dynamic Bayesian Networks for NLU Simulation with Applications to Dialog Optimal Strategy Learning. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'06)*, May 2006.

[9] Olivier Pietquin and Steve Renals. ASR system modeling for automatic evaluation and optimization of dialogue systems. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2002)*, Orlando, (USA, FL), May 2002.

[10] J. Schatzmann, B. Thomson, and S. Young. Error simulation for training statistical dialogue systems. In *Proceedings of the International Workshop on Automatic Speech Recognition and Understanding (ASRU'07)*, Kyoto (Japan).

[11] O. Pietquin. A probabilistic description of man-machine spoken communication. In *Proc. ICME'05*, July 2005.

[12] H. Clarck and E. Schaefer. Contributing to discourse. *Cognitive Science*, 13:259–294, 1989.

[13] Olivier Pietquin. Learning to Ground in Spoken Dialogue Systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages 165–168, 2007.

[14] O. Lemon, X. Liu, D. Shapiro, and C. Tollander. Hierarchical Reinforcement Learning of Dialogue Policies in a Development Environment for Dialogue Systems: REALL-DUDE. In *10th SemDial Workshop on the Semantics and Pragmatics of Dialogue; BRANDIAL*, 2006.

[15] J. Bos, E. Klein, O. Lemon, and T. Oka. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124, 2003.

[16] A. Cheyer and D. Martin. The Open Agent Architecture. In *Journal of Autonomous Agents and Multi-agent Systems*, number 4, pages 143–148, 2001.

[17] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1988.

[18] Steve Young. CUED Standard Dialogue Acts. Technical report, Cambridge University Engineering Dept, October 2007.

[19] Jason D. Williams and Steve Young. Partially Observable Markov Decision Processes for Spoken Dialog Systems. In *Computer Speech and Language*, volume 21, pages 231–422, 2007.

[20] B. Thomson, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, K. Yu, and Steve Young. User Study of the Bayesian Update of Dialogue State Approach to Dialogue Management. In *Proceedings of Interspeech*, 2008.