Grounding Simulation in Spoken Dialog Systems with Bayesian Networks

Stéphane Rossignol, Olivier Pietquin and Michel Ianotto

IMS Research Group, Supélec – Metz Campus, France, email: stephane.rossignol@supelec.fr

Abstract. User simulation has become an important trend of research in the field of spoken dialog systems because collecting and annotating real man-machine interactions with users is often expensive and time consuming. Yet, such data are generally required for designing and assessing efficient dialog systems. The general problem of user simulation is thus to produce as many as necessary natural, various and consistent interactions from as few data as possible. In this paper, is proposed a user simulation method based on *Bayesian Networks* (BN) that is able to produce consistent interactions in terms of user goal and dialog 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 dialogs in the TownInfo domain (a tourist information application). Experiments with a state-of-the-art dialog system (REALL-DUDE/DIPPER/OAA) have been realized and promising results are presented.

1 Introduction

Spoken dialog 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 are very expensive and time consuming. For this reason, user simulation has become an important trend of research during the last decade. User simulation can be used for performance assessment [4, 9] or for optimisation purposes [8, 13, 19], like when optimizing dialog strategies with reinforcement learning. User simulation should not be confused with user modelling. User modelling is generally used by a dialog system for internal purposes such as knowledge representation and user goal infering [5, 10] or natural language understanding simulation [14]. The role of user simulation is to generate a large amount of simulated interactions with a dialog system and the simulated user is therefore external to the system.

Dialog simulation can occur at several level of description. In this work, simulated interactions take place at the intention level (such as in [4, 8, 13, 19]) and not at the signal level as proposed in [9]. An intention is here defined as the minimal unit of information that a dialog participant can express independently. It will be modeled as dialog acts. Indeed, intention-based communication allows error modelling of every part of the system, including speech recognition and understanding [15, 14, 18]. Pragmatically, it is easier to automatically generate intentions or dialog acts compared with speech signals, as a large number of utterances can express the same intention.

The *Simulated User* (SU) presented in this paper 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 dialogs that are statistically consistent with each other. 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 paper relies on previous work [11, 13, 16] where BN are used for simulation purpose but emphasizes on two novel contributions. First, the model has been modified to generate grounding behaviours [3]. The grounding process will be considered here as the process used by dialog participants to ensure that they share the background knowledge necessary for the understanding of what will be said later in the dialog. In practice, that means that the simulated user will react automatically by providing again correct information to the dialog system if a problem in the information transmission is detected. The ultimate goal of this work being to train dialog policies that handle such grounding behaviours [12]. Second, the model is trained on actual human-machine dialogs data and tested on a state-of-the-art dialog system (the REALL-DUDE/DIPPER/OAA environment [7, 1, 2]).

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 three different slots ("food", "price_range" and "area") are considered. These slots can take respectively 3, 3 and 5 values (see table 1).

Table 1. Slots in the task, and corresponding possible values

Food	"italian"	"indian"	"chinese"		
Range price	"moderate"	"expensive"	"cheap"		
Area	"central"	"north"	"south"	"west"	"east"

The system acts in use are: <**hello**>, <**request**>, <**confirm**>, <**implicitconfirm_request**>, <**closingDialogue**>. The user acts in use are: <**inform**>, <**confirm**>, <**bye**>, (<**null**>).

The rest of this paper is organized as follows. In Section 2, the proposed model is described in details. In Section 3, experiments are presented. Finally, a conclusion and future works are provided in Section 4.

2 Description of the Model

2.1 Bayesian Network Model

The user simulation model described in this contribution is based on the probabilistic model of a man-machine dialog proposed in [11, 13]. The interaction between the user and the dialog manager is considered as a sequential transfer of intentions organized in

turns noted t thanks to dialog acts. At each turn t the dialog manager selects a system dialog act a_t conditionally to its internal state s_t. The user answers by a user act u_t which is conditioned by the goal g_t s/he is pursuing and the knowledge k_t s/he has about the dialog (what has been exchanged before reaching turn t). So, at a given turn, the information exchange can be modeled thanks to the joint probability $p(a, s, u, g, k)$ of all these variables. This joint probability can be factored as:

$$p(a, s, u, g, k) = p(u|g, k, a, s)p(g|k, a, s)p(k|s, a)p(a|s)p(s)$$

Given that :

- since the user doesn't have access to the SDS state, u, g and k cannot depend on s,
- the user's goal can only be modified according to his/her knowledge of the dialog,

this expression can be simplified :

$$p(a, s, u, g, k) = \underbrace{p(u|g, k, a)}_{\text{User act}} \overbrace{p(g|k)}^{\text{Goal Modif.}} \underbrace{p(k|a)}_{\text{Know. update}} \overbrace{p(a|s)}^{\text{DM Policy}} p(s)$$

This can be expressed by the Bayesian network depicted on figure 1-a).

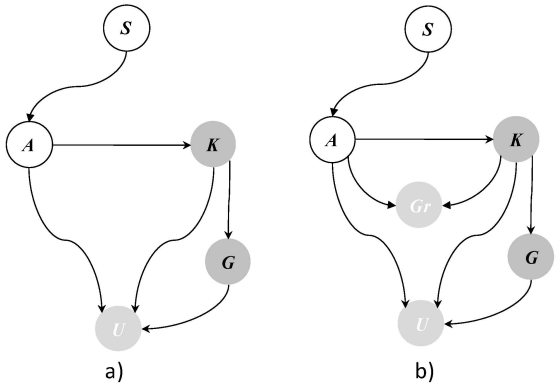

Fig. 1. Bayesian Network-based Simulated User

As explained in [13, 12], the practical use of this kind of bayesian network requires a tractable representation of the stochastic variables $\{a, s, u, g, k\}$. Therefore, the variable a is split into two subvariables $\{A_S, Sys\}$: the dialoge act type A_S (<**hello**>, <**request**>, <**confirm**>, <**implicitconfirm_request**>, <**closingDialogue**>) and the slots Sys on which it is applied (3 in this case, note Sys-SLOT(i)). The user act u is also divided into two subsets: the dialog act type u (<**inform**>, <**confirm**>, <**bye**>, (<**null**>)) and the values v of slots on witch they apply (*e.g.*. u = <**inform**>, v = "food_type = Italian"). A special variable C is added to simulate the closing of the dialog by the user. The knowledge k and the goal g are represented as sets of attribute-value pairs. There is a knowledge value for each of the slots in the task (3 in this case, note k-SLOT(i)). Three levels of knowledge (thus 3 possible values) are considered: *low*,

medium and *high*. These values represent the knowledge the user has about the fact that s/he previously provided the information about the corresponding slot 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* and it is more likely that the user will close the dialog if this slot is asked for one more time. The knowledge somewhat corresponds to a SU estimate of what is the dialog state. The user goal contains one value for each slot in the task. This ensures that the SU behaviour will be consistent according to a given goal. One extra possible value "*don't care*" is added to indicate that the user may not be interested in the value of one specific slot.

2.2 Grounding

Notice that, if the *Dialog Manager* (DM) asks confirmation for a slot for which the SU has a *low* knowledge value (i.e. a slot for which the SU has never provided information yet), it is likely that a grounding problem occurred. In other words, the DM and the user don't agree on the exchanged slots before reaching turn t. The SU described so far is designed to confirm (or not) the DM information it receives for this slot, with a certain probability, which can be low (or possibly to close the dialog). Yet, the knowledge value could be used to infer the occurrence of a grounding problem. More generally, the Bayesian network of Figure 1-a) shows that one could infer the most probable dialog state $\hat{s}_t$ at turn t given the observed DM act a_t and the user's knowledge k_t: $\hat{s}_t = \text{argmax}_s p(s|a_t, k_t)$ (diagnostic inference). The user's knowledge is required since it keep traces of the dialog history from the user's point of view. If this dialog state estimate $\hat{s}_t$ is very different from the user knowledge k_t, a grounding problem is likely to occur.

Instead of computing this state estimate and comparing it to the user's knowledge, we preferred to directly add a decision node in the network as shown on Figure 1-b). This node can only take boolean values, the *true* value meaning that a grounding problem occurred. In practice a grounding value can be obtained for each slot. In this implementation, if a grounding problem is detected for the slot i, the SU is forced to provide information concerning this slot. More sophisticated grounding strategies could be investigated but we restrict the study to this simple one in this paper.

3 Experiments

3.1 Learning BN parameters

The whole set of probabilities in the conditional probability tables (CPTs) can not be learned, as this concerns thousands of them (2151, more precisely, in this paper). For most of the probabilities, it would not be possible to get a good estimate of their values anyway since only a small amount of similar situations can be found in the database. In Tables 2, 3, 4, 5 and 6, are summarized the probabilities that we suppose to well generalize. For instance, the slots are considered as equivalent and their actual values not to influence the user's answer to confirmations for example. Furthermore, when

considering the probability for the user to close the dialog (C variable mentioned before) we consider that the four system acts "hello", "request", "confirm", and "implicit-confirm_request" are equivalent. Only the system act "closingDialog" has been treated differently. Finally, the variables coming from the A_S node and from the k nodes are supposed independent. Therefore, for the probability of closing the dialog, it is assumed than providing $2 + 3 = 5$ probabilities is enough to well generalise the parameters for the corresponding variable C. The same kind of reasoning has been performed for each node, leading us to a set of only 25 probabilities that need to be learned or heuristically fixed.

Table 2. Expert versus trained probabilities – closing probability variable C

	expert	learned
from A_S: the system act is not "closingDialog"	0.90	0.997
from A_S: the system act is "closingDialog"	0.99	0.944
from k-SLOT(i); the knowledge is low, thus the SU rather does not close (*low* value) the dialog	0.01	0.016
from k-SLOT(i); the knowledge is medium, thus the SU rather closes the dialog	0.90	0.104
from k-SLOT(i); the knowledge is high, thus the SU rather closes the dialog (probability higher than above)	1.00	0.773

The heuristically determined values for these probabilities, and their corresponding learned values are provided in the next five tables (Tables 2, 3, 4, 5 and 6). For instance, the first probability in Table 2 corresponds to the probability that the Simulated User decides to close the dialog if the system act coming from the DM is something else than a "closingDialog" act. Furthermore, in Table 2, the fact that the trained value for the probability of closing the dialog when the knowledge has a *high* value is quite low (0.773) compared to the corresponding expert probability value (1.00), indicates that human users tend to avoid to close the dialog even when they should think that they already provided the whole requested pieces of information. Therefore, dialogs generated with the heuristically parameterized BN-based simulator would probably tend to be shorter than those obtained with human users or with the trained BN-based Simulated User. The values in the seventh row and in the last row of Table 3 are going in the same direction. This indicates as well that the heuristic BN has been especially designed to allow the full system to reach as fast as possible the end of the dialog, that is to say in as few turns as possible.

3.2 Interaction with REALL-DUDE/DIPPER/OAA

The Simulated User has been interfaced with the spoken dialog system provided in the REALL-DUDE/DIPPER/OAA environment (see [7], [1] and [2]). This environment originally aims at training policies by reinforcement learning. Yet, the dialog policy used for our experiments has been trained independently of the SU presented here and is used for testing purpose only. Dialogs similar to the examples provided in Tables 7 and 8 are obtained. The SU accompanies each hypothesis it sends to the DM with

Table 3. Expert versus trained probabilities – $\{u, v\}$=INFORM SLOT(i)

	expert	learned
from A_S: the system greets, or asks for a slot, thus the SU decides to enter the slot(i)	1.00	0.32
from A_S: the system asks for a confirmation, thus the SU decides rather not to enter the slot(i) (low value)	0.01	0.0314
from A_S: the system closes the dialog, thus the SU decides rather to enter the slot(i) (low value)	0.35	$9.6e^{-5}$
from Sys-SLOT(i): the slot(i) is concerned by the system act, thus the SU decides to enter the slot(i)	1.00	0.848
from Sys-SLOT(i): the slot(i) is not concerned by the system act, but the SU decides to enter the slot(i) anyway	0.95	0.727
from Sys-SLOT(i): the slot(i) is concerned by the system act, and the SU decides to enter the slot(j)	0.95	0.47
from k-SLOT(i): the knowledge for the slot(i) is low, thus the SU decides to enter the slot(i)	1.00	0.968
from k-SLOT(i): the knowledge for the slot(i) is medium, thus the SU hesitates (probability smaller than above) to enter the slot(i)	0.8	0.768
from k-SLOT(i): the knowledge for the slot(i) is high, thus the SU decides rather not to enter the slot(i) (low value)	0.001	0.361

Table 4. Expert versus trained probabilities – $\{u, v\}$ = CONFIRM SLOT(i)

	expert	learned
from A_S: the system greets or asks for a slot, but the SU decides to confirm the slot(i) (probability low)	0.01	0.117
from A_S: the system asks for a confirmation for the slot(i), thus the SU decides to confirm the slot(i)	1.00	0.709
from Sys-SLOT(i): the slot(i) is concerned by the system act, but the SU decides to confirm the slot(i)	0.99	0.873
from Sys-SLOT(i): the slot(i) is not concerned by the system act, thus the SU decides to confirm the slot(i) (low value)	0.001	0.0832
from k-SLOT(i): the slot(i) is concerned by the system act and the knowledge for the slot(i) is low, and the SU decides to confirm the slot(i) (low value?)	0.01	0.906
from k-SLOT(i): the slot(i) is concerned by the system act and the knowledge for the slot(i) is medium, thus the SU rather decides to confirm the slot(i)	0.98	0.915
from k-SLOT(i): the slot(i) is concerned by the system act and the knowledge for the slot(i) is high, thus the SU rather decides to confirm the slot(i)	0.99	0.833

Table 5. Expert versus trained probabilities – $\{u, v\}$ = INFORM VALUE SLOT(i)

	expert	learned
from INFORM SLOT(I): INFORM SLOT(I) not selected (the system decided not to provide information about the slot(i)), thus the SU decides to enter the value for the slot(i) (low value)	0.02	0.0213
from VALUE GOAL SLOT(I): the SU decides to enter the value for the slot(i) it has in its goal	0.99	0.979

Table 6. Expert versus trained probabilities – $\{u, v\}$ = CONFIRM VALUE SLOT(i)

	expert	learned
from u = confirm, v = SLOT(i): CONFIRM SLOT(I) not selected (the system decided not to confirm the slot(i)), but the SU decides to confirm the value for the slot(i) (low value)	0.05	0.00443
from VALUE GOAL SLOT(I): the SU decides to confirm the value for the slot(i) it has in its goal	0.99	0.980

a probability simulating the confidence score which would be provided among other things by the *Automatic Speech Recogniser* (ASR) system. The DM can decide to ask for a confirmation if this probability is too low. This explains the "<syst act> confirm(Food)" and "<user act> confirm(food=yes)" turns in the second dialog example (in Table 8). It can be noticed that the SU correctly answered to this system act.

Table 7. First dialog example, obtained using the Simulated User integrated within the REALL-DUDE/DIPPER/OAA spoken dialog system environment

```
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
```

3.3 Heuristic versus Trained BN-based SU – Statistics

In this Section, are presented statistics computed on dialogs obtained using two versions of the SU presented in this paper and the REALL-DUDE/DIPPER/OAA environment. The first version of the SU is obtained considering heuristically determined values for the BN parameters. The second version is obtained considering trained parameters. The database used for training contains 1234 dialogs. It provides much more complex dialogs 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., appealing 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 used (see [23] for an exhaustive list). In the current task, less acts are considered, as noticed in the introduction. 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. The database has been described in more details in [22] and [20], where it has been used for training dialog management strategies.

In Table 9, are presented the results obtained using the *heuristic BN* (h-BN) and the *trained BN* (t-BN). A thousand of dialogs have been simulated for each of them. On Figure 2, is presented the histogram of the number of turns required to reach the end

of the dialogs. Considering the h-BN, it can be seen that most of the dialogs indeed are carried out into four turns, as expected (93.4 %). This is due to the fact that it has been designed to obtain as short dialogs as possible.

Table 8. Second dialog example, obtained using the Simulated User integrated within the REALL-DUDE/DIPPER/OAA spoken dialog system environment

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

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

Table 9. Mean number of turns requested to reach the end of the dialogs; max number of turns; min number of turns; percentage of dialogs for which the end is reached in 4 turns; percentage of dialogs for which the end is reached in less than 9 turns

	mean	max	min	4 turns	< 9 turns
h-BN	4.969	21	4	93.20 %	93.40 %
t-BN	4.577	9	4	58.00 %	99.90 %

Using the t-BN, the dialogs are quite longer than when using the h-BN. This cannot be seen considering the average number of turns, but considering the percentage of dialogs which needed exactly four turns to reach their end: this percentage drops from 93.2 % to 58.0 %. However, very promisingly, the mean number of turns and the percentage of dialogs which needed less than nine turns to reach their end are quite better using the t-BN. Furthermore it can be noticed that the very long dialogs (more than nine turns requested), indicating some deep misunderstanding between the DM and the SU, have completely disappeared. This indicates that the dialogs obtained with the t-BN are much more natural, at least from a DM point of view, than the dialogs obtained with the h-BN. The distribution is actually more in agreement with the data which shows again the naturalness of the simulated dialogs.

Table 10 presents the number of turns needed per slot, respectively when the h-BN is used the longest dialogs being kept, when the h-BN is used the longest dialogs not being kept (they only reflect the DM stopping condition), when the t-BN is used, and considering the database. Clearly, the t-BN provides more realistic dialogs, in terms of number of turns needed for each slot.

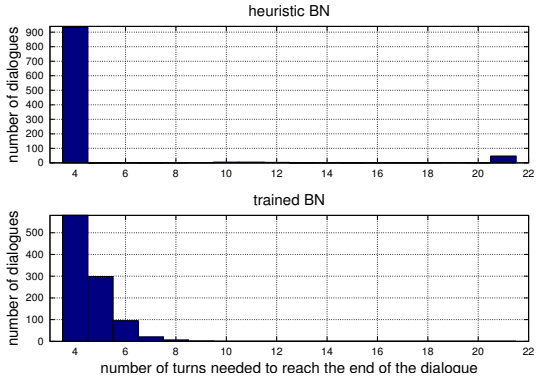

Fig. 2. Histogram of the number of turns requested to reach the end of the dialog – top: the heuristic BN is used – bottom: the trained BN is used

Table 10. Number of turns needed per slot; h-BN and longest dialogs being kept, h-BN used and longest dialogs not being kept, t-BN used, and considering the database

h-BN	h-BN without long dialogs	t-BN	database
1.6563	1.3986	1.5257	1.5661

3.4 Grounding process – Dialog Examples and Statistics

On 1000 simulated dialogs, 496 grounding problems have been detected. In Tables 11 and 12, dialog 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 consequently and the dialog went worst afterwards. When the DM asks confirmation for the third slot, the SU answered to it even if it had never sent information about this slot before. The SU is confused, and the DM is likely to be confused afterwards as well.

Statistics computed on dialogs obtained using the *grounding-enabled* SU are presented as well, the grounding problem solver being used or not. As the task features three slots and as the SU is configured to send information about no more than a single slot per turn, the minimum number of turns necessary to reach the end of the dialog is four: one per slot, plus the "closingDialog" turn. Notice that a turn is defined here as a couple <syst act>/<user act> (except for the "closingDialog" act, as it is the DM which is actually able to stop a dialog).

Considering only the dialogs with a grounding problem, and not the whole set of dialogs, results in Table 13 are obtained. They are quite promising. The amount of dialogs longer than 5 turns drops of 30.6 %.

4 Conclusion and Future Works

In this paper, a user simulation model based on Bayesian Networks is proposed to simulate realistic human-machine dialogs at the intention level, including grounding

Table 11. Dialog example, obtained using the Simulated User integrated within the REALL-DUDE/DIPPER/OAA spoken dialog system environment; a grounding problem is detected and solved (the grounding problem solver is used)

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

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

<syst act> confirm(Area)
=> grounding error detected for the slot_1 (the DM understood
   something wrong); problem solved by the SU
<user act> inform(slot_1='italian')

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

Table 12. Dialog example, obtained using the Simulated User integrated within the REALL-DUDE/DIPPER/OAA spoken dialog system environment; a grounding problem occurs and it is not solved by the SU (the grounding problem solver is not used)

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

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

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

<syst act> confirm(Area)
=> grounding error detected for the slot_1 (the DM understood
   something wrong); the SU does not respond correctly to
   this problem
<user act> confirm(area=no)

...
```

Table 13. Mean number of turns requested to reach the end of the dialogs; percentage of dialogs for which the end is reached in more than 5 turns – the BN without grounding problem solver or the BN with a grounding problem solver is used

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

behaviours. Our goal was first to show the interest of training the parameters of the Bayesian Networks using a database of actual human-machine dialogs, and second to show the interest of simulating the grounding process often occuring in human-human dialogs. This is done by comparing the number of turns required to reach the end of a dialog using different configuration of the proposed simulation framework. Several directions for future works are furthermore envisioned.

First, one of the goal of developing this Simulated User is the training of the dialog management policies within a reinforcement learning paradigm. The developed Simulated User will allow training the policies in use in the POMDP engines implemented in the REALL-DUDE/DIPPER/OAA environment. Second, some preliminary results concerning the interaction of the presented Simulated User with an independently developed Dialog Manager are shown. It is planned in the very next future to interact with the spoken dialog system provided in the REALL-DUDE/DIPPER/OAA environment in a much more extensive and systematic way. This will allow comparing the number of turns obtained with the BN-based Simulated Users to the number of turns obtained with human users and analysing the corresponding task completion scores etc. Furthermore, refined metrics for the evaluation of User Simulations ([21],[17],[6]) are under study. Third we would like to use the ability of Bayesian Networks to learn online their parameters so as to improve the simulated dialogs naturalness as users are really interacting with the system and to be able to retrain policies between real interactions.

Acknowledgements

This work is supported by the CLASSIC (Computational Learning in Adaptive Systems for Spoken Conversation) european project, Project Number 216594 funded under EC FP7, Call 1 (http://www.classic-project.org/). The authors also want to thank Oliver Lemon and Paul Crook for their help in using the REALL-DUDE/DIPPER/OAA framework.

References

1. 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.
2. A. Cheyer and D. Martin. The Open Agent Architecture. In *Journal of Autonomous Agents and Multi-agent Systems*, number 4, pages 143–148, 2001.
3. H. Clark and E. Schaefer. Contributing to discourse. *Cognitive Science*, 13:259–294, 1989.
4. 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.
5. 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 *Proc. of the 14th Conference on Uncertainty in Artifical Intelligence*, July 1998.
6. S. Janarthanam and O. Lemon. A Two-Tier Simulation Model for Reinforcement Learning of Adaptative Referring Expression Generation Policies. In *Proceedings of 10th SIGDIAL*, pages 120–123, 2009.

7. 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; BRANDIAL06*, 2006.
8. 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.
9. R. López-Cózar, Z. Callejas, and M. F. McTear. Testing the performance of spoken dialogue systems by means of an artificially simulated user. *Artificial Intelligence Review*, 26(4):291–323, 2006.
10. 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.
11. O. Pietquin. A Probabilistic Description of Man-Machine Spoken Communication. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, July 2005.
12. O. 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.
13. O. Pietquin and T. 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.
14. 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)*, May 2006.
15. O. Pietquin and S. 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)*, Orlando, (USA, FL), May 2002.
16. O. Pietquin, S. Rossignol, and M. Ianotto. Training Bayesian Networks for Realistic Man-Machine Spoken Dialogue Simulation. *Proceedings of the 1rst International Workshop on Spoken Dialogue Systems Technology*, December 2009.
17. J. Schatzmann, K. Georgila, and S. Young. Quantitative Evaluation of User Simulation Techniques for Spoken Dialogue Systems. In *Proceedings of the 6th SIGDIAL*, pages 45–54, 2005.
18. 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)*, Kyoto (Japan), 2007.
19. 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.
20. B. Thomson, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, K. Yu, and S. Young. User Study of the Bayesian Update of Dialogue State Approach to Dialogue Management. In *Proceedings of Interspeech*, 2008.
21. L. Vuurpijl, L. ten Bosch, S. Rossignol, A. Neumann, N. Pfleger, and R. Engel. Evaluation of multimodal dialog systems. In *Proceedings of the LREC Workshop on Multimodal Corpora*, 2004.
22. J. D. Williams and S. Young. Partially Observable Markov Decision Processes for Spoken Dialog Systems. In *Computer Speech and Language*, volume 21, pages 231–422, 2007.
23. S. Young. CUED Standard Dialogue Acts. Technical report, Cambridge University Engineering Dept, October 2007.