Available online at www.sciencedirect.com

ScienceDirect

Computer Speech and Language 28 (2014) 743–768

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

A domain-independent statistical methodology for dialog management in spoken dialog systems ☆

David Griol [a,*], Zoraida Callejas [b], Ramón López-Cózar [b], Giuseppe Riccardi [c]

[a] *Computer Science Department, Carlos III University of Madrid, Spain*
[b] *Department of Languages and Computer Systems, University of Granada, Spain*
[c] *Department of Information Engineering and Computer Science, University of Trento, Italy*

Received 3 October 2012; received in revised form 19 August 2013; accepted 8 September 2013
Available online 27 September 2013

Abstract

This paper proposes a domain-independent statistical methodology to develop dialog managers for spoken dialog systems. Our methodology employs a data-driven classification procedure to generate abstract representations of system turns taking into account the previous history of the dialog. A statistical framework is also introduced for the development and evaluation of dialog systems created using the methodology, which is based on a dialog simulation technique. The benefits and flexibility of the proposed methodology have been validated by developing statistical dialog managers for four spoken dialog systems of different complexity, designed for different languages (English, Italian, and Spanish) and application domains (from transactional to problem-solving tasks). The evaluation results show that the proposed methodology allows rapid development of new dialog managers as well as to explore new dialog strategies, which permit developing new enhanced versions of already existing systems.

Keywords: Spoken dialog systems; Dialog management; Statistical methodologies; User modeling; Dialog simulation; Systems evaluation

1. Introduction

Spoken dialog systems are computer programs that receive as input speech and generate as output synthesized speech, engaging the user in a dialog that aims to be similar to that between humans (Pieraccini, 2012; Heinroth and Minker, 2012; López-Cózar and Araki, 2005; McTear, 2004; Gibbon et al., 2000). Thus, these interfaces make technologies more usable, as they ease interaction (Hempel, 2008), allow integration in different environments (Heinroth and Minker, 2012; Minker et al., 2004), and make technologies more accessible, especially for disabled people (Vipperla et al., 2012; Beskow et al., 2009).

Usually, SDSs carry out five main tasks: Automatic Speech Recognition (ASR), Spoken Language Understanding (SLU), Dialog Management (DM), Natural Language Generation (NLG), and Text-To-Speech Synthesis (TTS). These tasks are typically implemented in different modules of the system's architecture.

☆ This paper has been recommended for acceptance by A. Potamianos.

* Corresponding author. Tel.: +34 916249116.
E-mail addresses: david.griol@uc3m.es, dgriol@inf.uc3m.es (D. Griol), zoraida@ugr.es (Z. Callejas), rlopezc@ugr.es (R. López-Cózar), riccardi@disi.unitn.it (G. Riccardi).

0885-2308/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.csl.2013.09.002

The goal of speech recognition is to obtain the sequence of words uttered by a speaker (Tsilfidis et al., 2013; O'Shaughnessy, 2008; López-Cózar and Callejas, 2008). It is a very complex task, as there can be a great deal of variation in the input the recognizer must analyze, for example, in terms of the linguistics of the utterance, inter and intra speaker variation, the interaction context and the transmission channel. Once the speech recognizer has provided an output, the system must understand what the user said. The goal of spoken language understanding is to obtain the semantics from the recognized sentence. This process generally requires morphological, lexical, syntactical, semantic, discourse and pragmatical knowledge (Wu et al., 2010; López-Cózar et al., 2010; Minker, 1999).

The dialog manager decides the next action of the system (Traum and Larsson, 2003; Williams and Young, 2007; Griol et al., 2008), interpreting the incoming semantic representation of the user input in the context of the dialog. In addition, it resolves ellipsis and anaphora, evaluates the relevance and completeness of user requests, identifies and recovers from recognition and understanding errors, retrieves information from data repositories, and decides about the next system's response. Natural language generation is the process of obtaining sentences in natural language from the non-linguistic, internal representation of information handled by the dialog system (Lemon, 2011; López et al., 2011). Finally, the TTS module transforms the generated sentences into synthesized speech (Dutoit, 1996).

In order to enable rapid deployment of these systems, markup languages such as VoiceXML[1] have been widely adopted as they reduce the time and effort required for system implementation. However, system development with this approach involves a very costly engineering cycle (Rojas-Barahona and Giorgino, 2009). As an attempt to reduce this cost and carry out rapid system prototyping, statistical approaches are gaining increasing interest. These approaches enable automatic learning of dialog strategies, thus avoiding the time-consuming process that hand-crafted dialog design involves. Statistical models can be trained from real dialogs, modeling the variability in user behaviors. Although the construction and parameterization of these models depend on expert knowledge about the task to be carried out by the dialog system, the final objective is to develop systems that are more robust for real-world conditions, and that are easier to adapt to different users and tasks (Schatzmann et al., 2006).

In this paper we present a statistical approach for the development of dialog managers, which is mainly based on the modelization of the sequences of the system and user dialog acts and the introduction of a partition in the space of all the possible sequences of dialog acts. Unlike other statistical approaches, our approach has the advantage of taking into account the data supplied by the user throughout the dialog, which makes the estimation of the statistical model tractable, without causing scalability problems. Our proposal is suitable to develop dialog managers regardless of the application domain and the interaction language. Moreover, it can be incrementally optimized to tackle complex tasks, as will be shown in the experiments.

After this introduction, the remainder of the paper is organized as follows. Section 2 describes existing approaches for the development of dialog managers, paying special attention to statistical approaches. Section 3 describes the proposed methodology for dialog management and the dialog generation technique employed to simulate dialogs. Section 4 describes the process and practical dialog systems used for evaluating our proposal. Section 5 presents the results of the evaluation of the dialog management methodology for the different systems using the proposed set of evaluation measures. Section 6 presents the conclusions and suggests some future work guidelines.

2. Related work

Although dialog management is only a part of the development cycle of spoken dialog systems, it can be considered one of the most demanding tasks given that this module encapsulates the logic of the speech application (Wilks et al., 2011). Traum and Larsson (2003) state that dialog management involves four main tasks: (i) updating the dialog context, (ii) providing a context for sentence interpretation, (iii) coordinating other modules and (iv) deciding the information to convey to the user and when to do it. Thus, the selection of a specific system action depends on multiple factors, such as the output of the speech recognizer (e.g., measures that define the reliability of the recognized information), the dialog interaction and previous dialog history (e.g., the number of repairs carried out so far), the application domain (e.g., guidelines for customer service), knowledge about the users, and the responses and status of external back-ends, devices, and data repositories. Given that the actions of the system directly impact users, the dialog manager is largely

[1] http://www.w3.org/TR/voicexml20/.

responsible for user satisfaction. This way, the design of an appropriate dialog management strategy is at the core of dialog system engineering.

The simplest dialog management strategy is programmatic dialog management, in which a generic program implements the application with an interaction model based on finite-state machines (Barnard et al., 1999). The user actions determine the transitions between the system responses, which are the nodes of the finite-state machine. Users' actions represent the user responses to the system prompts, which are usually coded in recognition grammars.

These early applications only supported strict directed dialog interaction, in which at each turn the system directs the user by proposing a small number of choices, which also result in a limited grammar or vocabulary at each turn. Directed dialog was efficient in terms of accuracy and cost of development. Although libraries and dialog modules are generally created in the form of sample code or templates, which could be reused and adapted to different applications, the weakest point of this approach is its lack of versatility (Acomb et al., 2007; Pieraccini et al., 2009).

Unlike the finite-state approach, frame-based dialog managers do not have a predefined dialog path but use a frame structure (Minsky, 1975) comprised of one slot per piece of information that the system can gather from the user (McTear, 2004). The core idea is that humans communicate to achieve goals and during the interaction the mental state of the speakers may change. Thus, frame-based dialog managers model dialog as a cooperation between the user and the system to reach common goals. Utterances are not considered text strings, but dialog acts in which the user communicates his intentions. In this approach, the system interprets speech in order to acquire enough information to perform a specific action. One of the main advantages is that it can capture several data at once and the information can be provided in any order (more than one slot can be filled per dialog turn and in any order), thus supporting mixed-initiative dialogs. This strategy is suitable for form-filling tasks in which the system asks the user a series of questions to gather information, and then consults an external knowledge source, such as the ones that can be developed with VoiceXML.

A related approach is the so-called "information state" dialog theory (Traum and Larsson, 2003). The information state of a dialog represents the information needed to uniquely distinguish it from all others. It comprises the accumulated user interventions and previous dialog actions on which the next system response can be based. The information state is also sometimes known as the conversation store, discourse context, or mental state. Following this theory, the main tasks of the dialog manager are to update the information state based on the observed user actions, and select the next system action.

Additionally, when it is necessary to execute and monitor operations in a dynamically changing application domain, an agent-based approach can be employed. The modular agent-based approach to dialog management makes it possible to combine the benefits of different dialog control models, such as finite-state based dialog control and frame-based dialog management (Chu et al., 2005). Similarly, it can benefit from alternative dialog management strategies, such as the system-initiative approach and the mixed-initiative approach (Walker et al., 1997).

Statistical approaches for dialog management present several important advantages. Rather than maintaining a single hypothesis for the dialog state, they maintain a distribution over many hypotheses for the correct dialog state. In addition, statistical methodologies choose actions using an optimization process, in which a developer specifies high-level goals and the optimization works out the detailed dialog plan.

Automating dialog management is useful for developing, deploying and re-deploying applications and also reducing the time-consuming process of hand-crafted design. In fact, the application of machine learning approaches to dialog management strategy design is a rapidly growing research area. Machine-learning approaches to dialog management attempt to learn optimal strategies from corpora of real human–computer dialog data using automated "trial-and-error" methods instead of relying on empirical design principles (Young, 2002). The main trend in this area is an increased use of data for automatically improving the performance of the system.

As described by Paek and Pieraccini (2008), there are three main categories of elements of the spoken dialog interaction where the use of massive amounts of data can potentially improve automation rate, and ultimately, the penetration and acceptance of speech interfaces in the wider consumer market. They are task-independent behaviors (e.g., error correction and confirmation behavior), task-specific behaviors (e.g., logic associated with certain customer-care practices), and task-interface behaviors (e.g., prompt selection). However, these three categories have in common today the lack of robust guiding principles validated by empirical evidence.

Statistical models can be trained with corpora of human–computer dialogs with the goal of explicitly modeling the variance in user behavior that can be difficult to address by means of hand-written rules (Schatzmann et al., 2006). Additionally, if it is necessary to satisfy certain deterministic behaviors, it is possible to extend the strategy learned from

the training corpus with handcrafted rules that include expert knowledge or specifications about the task (Suendermann and Pieraccini, 2012; Laroche et al., 2008; Torres et al., 2008; Young et al., 2013).

The goal is to build systems that exhibit more robust performance, improved portability, better scalability and easier adaptation to other tasks. However, model construction and parameterization is dependent on expert knowledge, and the success of statistical approaches is dependent on the quality and coverage of the models and data used for training (Schatzmann et al., 2006). Moreover, the training data must be correctly labeled for the learning process. The size of currently available annotated dialog corpora is usually too small to sufficiently explore the vast space of possible dialog states and strategies. Collecting a corpus with real users and annotating it requires considerable time and effort.

To address these problems, researchers have proposed alternative techniques that facilitate the acquisition and labeling of corpora, such as Wizard of Oz (Fraser and Gilbert, 1991; Lane et al., 2004), bootstrapping (Fabbrizio et al., 2008; Abdennadher et al., 2007), active learning (Liu and Shriberg, 2005; Cohn et al., 1994), automatic dialog act classification and labeling (O'Shea et al., 2012; Venkataraman et al., 2002), and user simulation (Schatzmann et al., 2006; López-Cózar et al., 2006).

Another relevant problem is how to deal with unseen situations, that is, situations that may occur during the dialog and that were not considered during training. To address this point it is necessary to employ generalizable models in order to obtain appropriate system responses that enable to continue with the dialog in a satisfactory way.

Another difficulty is in the design of a good dialog strategy, which in many cases is far from being trivial. In fact, there is no clear definition of what constitutes a good dialog strategy (Schatzmann et al., 2006; Lemon and Pietquin, 2012). Users are diverse, which makes it difficult to foresee which form of system behavior will lead to quick and successful dialog completion, and speech recognition errors may introduce uncertainty about their intention.

The most widespread methodology for machine-learning of dialog strategies consists of modeling human–computer interaction as an optimization problem using Markov Decision Processes (MDP) and reinforcement methods (Levin and Pieraccini, 1997; Singh et al., 1999; Levin et al., 2000). The main drawback of this approach is that the large state space of practical spoken dialog systems makes its direct representation intractable (Young et al., 2007). Partially Observable MDPs (POMDPs) outperform MDP-based dialog strategies since they provide an explicit representation of uncertainty (Roy et al., 2000). This enables the dialog manager to avoid and recover from recognition errors by sharing and shifting probability mass between multiple hypotheses of the current dialog state.

Another disadvantage of the POMDP methodology is that the optimization process is free to choose any action at any time. As a result, there is no obvious way to incorporate domain knowledge or constraints such as business rules. In addition, in the worst case spurious actions might be taken with real users, an especially serious concern if POMDP-based systems are going to handle financial or medical transactions. They are also limited to small-scale problems, since the state space would be huge and exact POMDP optimization is again intractable (Young et al., 2007).

An approach that scales the POMDP framework for implementing practical spoken dialog systems by the definition of two state spaces is presented in Young et al. (2005). Approximate algorithms have also been developed to overcome the intractability of exact algorithms but even the most efficient of these techniques such as Point-Based Value Iteration (PBVI) cannot scale to the many thousand states required by a statistical dialog manager (Williams et al., 2006). Composite Summary Point-Based Value Iteration (CSPBVI) has suggested the use of a small summary space for each slot where PBVI policy optimization can be applied. However, policy learning in this technique can only be performed offline, i.e. at design time, because policy training requires an existing accurate model of user behavior. An alternative technique for online training based on Q-learning is presented in Thomson et al. (2007), which allows the system to adapt to real users as new dialogs are recorded. This technique does not require any model of user behavior, so user simulation techniques are proposed to iteratively learn the dialog model.

Other authors have combined conventional dialog managers with a fully observable Markov decision process (Singh et al., 2002; Heeman, 2007), or proposed using multiple POMDPs and selecting actions using hand-crafted rules (Williams et al., 2006). In Williams (2008), the authors combine the robustness of the POMDP with the developer control afforded in conventional approaches: the (conventional) dialog manager and POMDP run in parallel, but the dialog manager is augmented so that it outputs one or more allowed actions at each time-step. The POMDP then chooses the best action from this limited set. Results from a real voice dialer application show that adding the POMDP machinery to a standard dialog system can yield a significant improvement (Williams, 2008).

Other interesting approaches for statistical dialog management are based on modeling the system by means of Hidden Markov Models (Cuayáhuitl et al., 2005), stochastic Finite-State Transducers (Hori et al., 2009; Hurtado et al., 2010; Planells et al., 2012), or using Bayesian Networks (Paek and Horvitz, 2000; Meng et al., 2003). Also Lee et al.

(2010) proposed a different hybrid approach to dialog modeling in which n-best recognition hypotheses are weighted using a mixture of expert knowledge and data-driven measures by using an agenda and an example-based machine translation approach respectively.

Our methodology for dialog management (Section 3) is based on the estimation of a statistical model from the sequences of the system and user dialog acts obtained from a set of training data. The automatic dialog generation technique, described in Section 3.3, facilitates the acquisition and also adaptation of the dialog system to deal with new domains. The next system response is selected by means of a classification process that considers the complete history of the dialog, which is one of the main advantages regarding the previously described statistical methodologies for dialog management. Another main characteristic is the inclusion of a data structure that stores the information provided by the user. The main objective of this structure is to easily encode the complete information related to the task provided by the user during the dialog history, then considering the specific semantics of the task and including this information in the proposed classification process.

3. Our proposed methodology for dialog management

As previously discussed, in most settings application developers, together with VUI designers, typically hand-craft dialog management strategies using rules and heuristics. As it is extremely challenging to anticipate every possible user input, hand-crafting dialog management strategies is an error-prone process that needs to be iteratively refined and tuned, which requires considerable time and effort. In this section we propose a methodology for statistical modeling dialog, which has the advantages of machine learning approaches to reduce the effort and time required by hand-craft dialog management strategies and, at the same time, makes it possible to isolate domain knowledge and then facilitate both to develop new dialog managers and to adapt them to deal with new domains (Section 3.1).

Additionally, as the success of statistical approaches depends on the quality of the data used to develop the dialog model, considerable effort is also necessary to acquire and label a corpus with the data necessary to train a good model. In addition, the usability of many spoken dialog systems is still limited, as developers frequently do not have the time to perform user tests during the development cycle. In order to avoid these negative effects, we also propose an approach to develop a user simulator and automatically acquire a dialog corpus (Section 3.3). This way, the user and dialog models are iteratively built from the interaction of both modules. Following our approach, a labeled corpus for the specific task is automatically obtained after the interaction, detecting whether a dialog is successful or not with the definition of a set of stop conditions.

3.1. Our approach for statistical dialog management

A conventional dialog manager maintains a state n such as a form or frame and relies on two functions for control, G and F. For a given dialog state n, $G(n) = a$ decides which system action to output, and then after observation o has been received, $F(n, o) = n_0$ decides how to update the dialog state n to yield n_0. This process is repeated until the dialog ends.

In a statistical approach, the conventional dialog manager is extended in three respects: firstly, its action selection function $G(n) = a$ is changed to output a set of one or more (M) allowable actions given a dialog state n, $G(n) = \{a_1, a_2, \ldots, a_M\}$. Next, its transition function $F(n, o) = n_0$ is extended to allow for different transitions depending on which of these actions was taken, $F(n, a, o) = n_0$.

In order to control the interactions with the user, our dialog manager represents dialogs as a sequence of pairs (A_i, U_i), where A_i is the output of the dialog system (the system answer) at time i, and U_i is the semantic representation of the user turn (the result of the understanding process of the user input) at time i; both expressed in terms of dialog acts (Griol et al., 2008). This way, each dialog is represented by:

$$(A_1, U_1), \ldots, (A_i, U_i), \ldots, (A_n, U_n)$$

where A_1 is the greeting turn of the system, and U_n is the last user turn. We refer to a pair (A_i, U_i) as S_i, the state of the dialog sequence at time i.

In this framework, we consider that, at time i, the objective of the dialog manager is to find the best system answer A_i. This selection is a local process for each time i and takes into account the previous history of the dialog, that is to say, the sequence of states of the dialog preceding time i:

$$\hat{A}_i = \underset{A_i \in \mathcal{A}}{\operatorname{argmax}} P(A_i | S_1, \ldots, S_{i-1}) \tag{1}$$

where set $\mathcal{A}$ contains all the possible system answers.

Following Eq. (1), the dialog manager selects the following system prompt by taking into account the sequence of previous pairs (A_i, U_i). The main problem to resolve this equation is regarding the number of possible sequences of states, which is usually very large. To solve the problem, we define a data structure in order to establish a partition in this space, i.e., in the history of the dialog preceding time i). This data structure, which we call *Dialog Register* (*DR*), contains the information provided by the user throughout the previous history of the dialog. After applying the above considerations and establishing the equivalence relation in the histories of dialogs, the selection of the best A_i is given by:

$$\hat{A}_i = \underset{A_i \in \mathcal{A}}{\operatorname{argmax}} P(A_i | DR_{i-1}, S_{i-1}) \tag{2}$$

Each user turn supplies the system with information about the task; i.e., the user asks for a specific concept and/or provides specific values for certain attributes. However, a user turn can also provide other kinds of information, such as task-independent information (for instance, *Affirmation*, *Negation*, and *Not-Understood* dialog acts). This kind of information implies some decisions which are different from simply updating the DR_{i-1}. Hence, for the selection of the best system response A_i, we take into account the *DR* that results from turn 1 to turn $i-1$, and we explicitly consider the last state S_{i-1}.

We propose to solve Eq. (2) by means of a classification process. This way, every dialog situation (i.e., each possible sequence of dialog acts) is classified taking into account a set of classes $\mathcal{C}$, in which a class contains all the sequences that provide the same set of system actions (responses). The objective of the dialog manager at each moment is to select a class of this set $c \in \mathcal{C}$, so that the system answer is the one associated with the selected class.

The classification function can be defined in several ways. We have evaluated six different definitions of such a function: a multinomial naive Bayes classifier, an n-gram based classifier, a decision tree classifier, a support vector machine classifier, a classifier based on grammatical inference techniques, and a classifier based on artificial neural networks (Griol et al., 2008, 2006). The best results were obtained using a multilayer perceptron (MLP) (Rumelhart et al., 1986) where the input layer holds the input pair (DR_{i-1}, S_{i-1}) corresponding to the dialog register and the state. The values of the output layer can be seen as an approximation of the a posteriori probability of the input belonging to the associated class $c \in \mathcal{C}$.

As stated before, the *DR* contains information about concepts and values for the attributes provided by the user throughout the previous history of the dialog. For the dialog manager to determine the next answer, we have assumed that the exact values of the attributes are not significant. They are important for accessing databases and for constructing the output sentences of the system. However, the only information necessary to predict the next action by the system is the presence or absence of concepts and attributes. Therefore, the codification we use for each field in the *DR* is in terms of three values, $\{0, 1, 2\}$, according to the following criteria: (0) The concept is unknown or the value of the attribute is not given; (1) the concept or attribute is known with a confidence score that is higher than a given threshold; (2) the concept or attribute has a confidence score that is lower than the given threshold.

To decide whether the state of a certain value in the *DR* is 1 or 2, the system employs confidence measures provided by the ASR and SLU modules (Torres et al., 2005). In complex scenarios, it might happen that the ASR provides a high confidence with respect to the recognized sentence (e.g. "eleven (0.9754)") but the SLU can decrease the confidence if it is vital for the system to obtain a more precise input (e.g. "eleven a.m."). Additional values for the codification of the information in the *DR* can also be introduced to cover exceptions for specific attributes.

The previously described process allows to model every task based only in the information provided by the user in the previous turns and its own model. In other dialog systems, the dialog manager generates the following system response taking also into account the information generated by the module that controls the application (which we denote as the *Application Manager*, *AM*). For example, the *AM* can validate restrictions, apply privacy policies or carry out computations that define the next system response (for instance, select a different system action depending on the

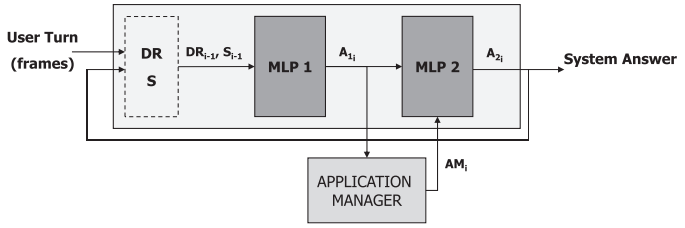

Fig. 1. Scheme of the complete architecture for the proposed dialog management methodology.

result of a query to the databases of the application). Thus, the output of this module has to be taken into account for the selection of the best system action.

For this reason, we have decided that for this kind of tasks, two phases are required for the selection of the next system turn. In the first phase, the information contained in the *DR* and the last state S_{i-1} are considered to select the best request to be made to the *AM* ($\hat{A}_{1_i}$):

$$\hat{A}_{1_i} = \underset{A_{1_i} \in \mathcal{A}_1}{\operatorname{argmax}} P(A_i | DR_{i-1}, S_{i-1}) \tag{3}$$

where $\mathcal{A}_1$ is the set of possible requests to the *AM*.

In the second phase, the system answer ($\hat{A}_{2_i}$) is generated taking into account $\hat{A}_{1_i}$ and the information provided by the *AM* (AM_i):

$$\hat{A}_{2_i} = \underset{A_{2_i} \in \mathcal{A}_2}{\operatorname{argmax}} P(A_i | AM_i, A_{1_i}) \tag{4}$$

where $\mathcal{A}_2$ is the set of possible system answers.

Fig. 1 shows the scheme proposed for the development of the dialog manager for this kind of tasks, detailing the two phases described for the generation of the system response. As stated above, we propose the use of two MLPs, each one to deal with the specific information defined for each phase.

Our methodology for dialog management can be used to model slot-filling tasks, which cover the most common application domains in which current dialog systems are involved, and for which defining a good dialog strategy can be difficult (Young, 2002). However, besides these tasks, the use of a second classification phase with the Application Manager (*AM*) makes it possible to apply our methodology to more complex tasks, as it will be described in Section 4.2. The *AM* makes it possible to consider specific requisites (e.g. special requirements, policies, or specific routines), which endow the system with a more sophisticated behavior that is different from only requiring information from the user and checking or updating a repository. This phase also makes possible for the systems to deal with specific cases for the different attributes, given that the exact values for each attribute are considered to access the data repositories.

In addition, the statistical dialog model allows introducing user adaptation, which makes it suitable for different application domains with varying degrees of complexity.

3.2. Example of a practical application

Fig. 2 shows an excerpt of a dialog for a system designed to provide railway information (Griol et al., 2008). The *DR* defined for the task consists of the possible queries that users can make (*Timetables*, *Fares*, *Train-Type*, *Trip-Time*, or *Services*) and ten attributes that they must provide to complete these queries (*Origin*, *Destination*, *Departure-Date*, *Arrival-Date*, *Departure-Hour*, *Arrival-Hour*, *Class*, *Train-Type*, *Order-Number*, and *Services*). Users can also provide three task-independent dialog acts (*Affirmation*, *Negation*, and *Not-Understood*).

Using the previously described codification for the *DR*, when a dialog starts (in the greeting turn) all the values in the dialog register are initialized to "0". The information provided by the users in each dialog turn is employed to update the previous *DR* and obtain the current one, as Fig. 2 shows.

This figure shows the semantic interpretation and confidence scores (in brackets) for a user's utterance provided by the SLU module. In this case, the confidence score assigned to the attribute *Date* is very low. Thus, a "2" value is added in the corresponding position of the DR_1. The concept (*Hour*) and the attribute *Destination* are recognized with a high confidence score, adding a "1" value in the corresponding positions of the DR_1. As the input to the MLP is

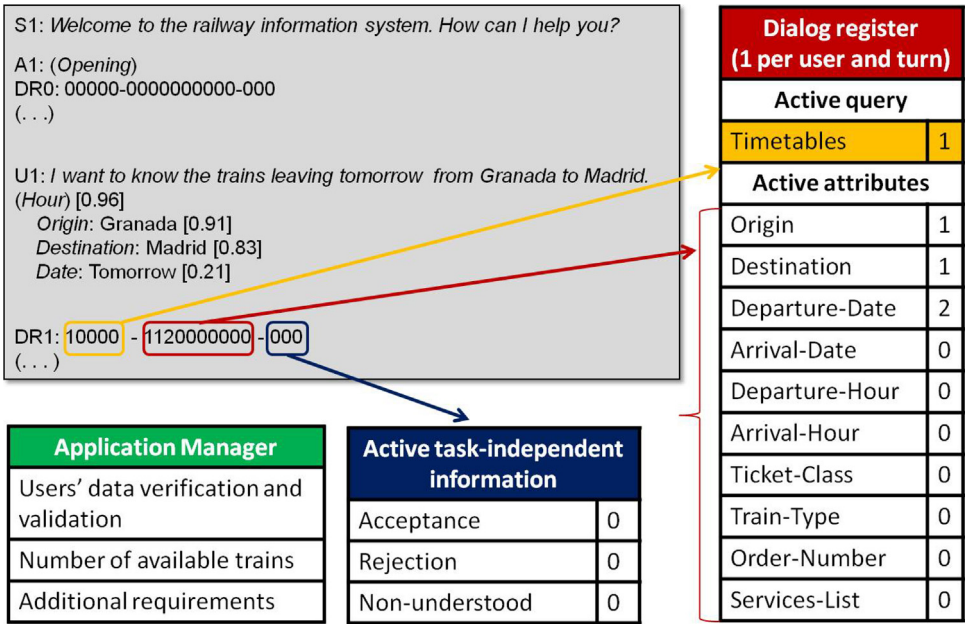

Fig. 2. Excerpt of a dialog with its correspondent Dialog Register and active task-independent information for one of the turns.

generated using DR_1, the codification of the labeling of the last system turn (A_1), and the task-independent information provided in the last user turn (none in this case), the dialog manager selects to confirm the departure date. This process is repeated to predict the next system response after each user turn.

As described in the previous subsection, the dialog manager requires the second phase to take into account the response generated by the Application Manager module. For instance, in order to book a ticket, the dialog manager would select the next system prompt taking into account the response of the *AM*, which can be either providing the response generated in the first phase (e.g., when the dialog manager selects the confirmation of an attribute, to ask the user for additional information, or to finish the dialog) or participating actively to produce a different answer, e.g., if the *AM* informs that there is no train that fulfills the user requirements, if there is only one available train and the reservation can now be completed, if more than one train is available and the user must select one of them, or the *AM* informs that the user query cannot be completed because there is an error (e.g., the origin city is the same that the destination city).

3.3. Automatic acquisition of the dialog corpus

As described in Section 2, the success of statistical approaches is tied to the quality of the models and the quality of the data used for its training. Different valid alternatives to the acquisition and manually annotation of the training data for our proposed dialog management technique have been described in the previous section. In this work, we propose to use a dialog simulation technique in order to automatically acquire the corpus that is required to learn the dialog model for the statistical dialog manager.

Our approach is based on the interaction of a user simulator and a dialog manager simulator (Griol et al., 2009a). Both modules initially use a random selection of one of the possible answers defined for the semantics of the task (user and system dialog acts). At the beginning of the simulation, the responses are defined as equiprobable. When a successful dialog is simulated, the probabilities of the responses selected by the dialog manager simulator and the user simulator during that dialog are incremented before starting a new simulation. This way, the described statistical dialog model is gradually learned with the incorporation of the successfully simulated dialogs. Although this approach converges to collaborative dialogs in the long term, it does not relay in predefined rules or user profiles, which would produce a more restricted behavior.

This technique simulates the user intention level, i.e., the user simulator provides concepts and attributes that represent the intention of the user utterance in terms of frames (Minsky, 1975). Therefore, the user simulator carries out the functions of the ASR and SLU modules. The semantics selected for the dialog manager simulator is represented through the set of possible system answers defined for the task. The selection of the possible user answers is carried out using the semantics defined for the SLU module.

The space of task-dependent responses for the user simulator and the dialog simulator can be defined considering the possibility of providing one or more dialog acts at each turn depending on the requisites of the dialog systems. It is very useful to let the user simulator choose several dialog acts to build the current response, as it allows modeling the situations in which users provide data for which they were not explicitly prompted emulating mixed dialog initiatives. Then, each of such combinations is considered as another valid alternative in the search space of the simulator, and there is no special reward or punishment method for the responses comprised of multiple acts. In this work, the semantics for the tasks were previously defined for each of the dialog systems used in the experimental setup (Section 4.2). This way, we could directly employ these semantics for the user simulator and the dialog manager simulator.

The task-independent user dialog acts (*Affirmation*, *Negation*, and *Not-Understood*) are generated in a slightly different way. The generation of *Affirmation* and *Negation* dialog acts is carried out automatically after the system has asked for confirmation. This way, if the data to be confirmed matches the predefined as correct for the simulation, the user simulator provides *Affirmation*, and in any other case, *Negation*. These dialog acts can be produced in isolation or in combination with others, for example: *Negation* + providing the correct value, *Affirmation* + reiteration of the correct value, *Negation* or *Affirmation* + other acts related to the task.

The *Not-Understood* dialog act is used to simulate the user communicating a misunderstanding. To generate it, we replicate the situations in which real users provide a *Not-Understood* dialog act. Thus, it is usually selected when the system poses a complex requirement or when the possible answers are not clear to the user. In the cases when an initial corpus is not available, a random or a statistical approach using a predefined complexity tag for each system act can be used. In the first case, *Not-Understood* can be selected randomly as a response to any system-act. In the second case, more complex system dialog acts will produce a higher probability of selecting *Not-Understood*, but each possible dialog act must be tagged with a complexity value inside a predefined range. In this paper, these situations have been extracted from the initial corpus available for the different tasks. In addition, the way in which the dialog manager simulator responds to the user's *Not-Understood* dialog act replicates the behavior of the initial practical dialog systems by repeating the previous system prompt. In a more advanced level of user adaptation, the system should clarify the previous turn trying to solve the misunderstanding more effectively.

An error simulation module has been implemented to include semantic errors in the generation of dialogs. Once the user simulator has selected the information to be provided to the dialog manager simulator, the error simulator modifies the frames created and provides a confidence score for each concept and attribute in the semantic representation of the user turn.

In the particular case of error simulation, we processed an initial corpus for each system to replicate the functioning of the ASR and NLU modules for the systems as accurately as possible. The model employed for introducing errors and confidence scores is inspired in the one presented by Schatzmann et al. (2007). Both processes are carried out separately following the noisy communication channel metaphor by means of a generative probabilistic model $P(c, U_a|\tilde{U}_a)$, where U_a is the true incoming user dialog act, $\tilde{U}_a$ is the recognized hypothesis, and c is the confidence score associated with this hypothesis.

On the one hand, the probability $P(\tilde{U}_a|U_a)$ is obtained by maximum-likelihood using the initial labeled corpus acquired with real users. To compute it, we consider the recognized sequence of words w_U and the actual sequence uttered by the user $\tilde{w}_U$. This probability is decomposed into a component that generates the word-level utterance that corresponds to a given user dialog act, a model that simulates ASR confusions (learned from the reference transcriptions and the ASR outputs), and a component that models the semantic decoding process.

$$P(\tilde{U}_a|U_a) = \sum_{\tilde{w}_U} P(U_a|\tilde{w}_U) \sum_{w_U} P(\tilde{w}_U|w_U) P(w_U|U_a)$$

Confidence score generation is carried out by approximating $P(c|\tilde{a}_u, a_u)$ assuming that there are two distributions for c. These two distributions are handcrafted, generating confidence scores for correct and

incorrect hypotheses by sampling from the distributions found in the training data corresponding to our initial corpus.

$$P(c|U_a, \tilde{U}_a) = \begin{cases} P_{corr}(c) & \text{if} \quad \tilde{U}_a = U_a \\ P_{incorr}(c) & \text{if} \quad \tilde{U}_a \neq U_a \end{cases}$$

A maximum number of turns per dialog is considered for acquiring a corpus using our user simulator, taking into account the requirements of the task for real users. A user request for closing the dialog is selected once the system has provided the information defined in the objective(s) of the dialog. When this happens before the maximum number of turns, the dialog is considered successful. The dialog manager considers that the dialog is unsuccessful and decides to abort it when the following conditions take place: (i) the dialog exceeds the maximum number of user turns; (ii) the answer selected by the dialog manager corresponds to a query not made by the user simulator; (iii) the *AM* generates an error because the user simulator has not provided the mandatory data to carry out the query; (iv) the answer generator generates an error because the selected answer involves the use of data not provided by the user simulator.

As described previously, to cover a wider user population, we do not only train the dialog model with optimal dialogs but also simulate recognition and understanding errors to make the dialog manager learn how to react to these erroneous situations. Additionally, to train the system with dialogs which are longer than the optimal, we increase the threshold that indicates the maximum number of dialog turns after which the simulation is aborted. However, regardless of these steps, it may happen that real dialogs contain situations not observed during the training, as also happens with rule-based approaches. Thus, our objective is not to cover all the possible situations in the training, but to develop a dialog manager that selects a response that allows continuing the dialog as best as possible in all scenarios. Then, the systems generated with the technique can be gradually improved by considering additional features acquired during their use, such as user preferences, beliefs, capabilities, and satisfaction with the system behavior.

4. Experiments and evaluation measures

In this section we describe the process and measures defined for the evaluation of the proposed dialog management technique (Section 4.1), and the different dialog systems for which it has been applied and evaluated (Section 4.2). As it will be described, these systems have been developed for domains with a very different nature and complexity, with different definitions for the semantics of the task (frames, dialog acts, attribute-values, predicate arguments), interaction languages (English, Italian, Spanish, Catalan, and Basque), dialog initiatives (system initiative, mixed initiative, and user initiative), confirmation and error handling and correction techniques (explicit confirmations, mixed confirmations, definition of confidence scores), interaction modalities (unimodal and multimodal), underlying technologies (from VoiceXML-based platforms to generic high-level programming languages), and different techniques for acquiring the initial dialog corpus (human–human, human–machine, Wizard of Oz technique).

4.1. Process followed

We have carried out two experiments for each system considered, as summarized in Figs. 3 and 4. The first setup is described in Fig. 3. As can be observed, several experts developed an initial, handcrafted dialog strategy for the different systems, resembling the behavior of experienced human operators who provide the same information and services that the corresponding systems. After implementing this strategy, a corpus of 300 dialogs was collected from spontaneous telephone calls of real users to each system. A first dialog manager was trained with 200 of such dialogs. The other 100 were used for testing.

Then, we employed the simulation technique described in Section 3.3 and generated 100,000 successful dialogs for each task. To do so, a set of different scenarios was created with the same goals as those of the real users in the initial corpus for each system. We developed a second dialog manager, called, which was trained using the proposed statistical technique with the successful simulated dialogs and the 200 dialogs of the initial corpus. Both and dialog managers were evaluated with the 100 dialogs test partition.

Fig. 4 shows the second experiment, in which we evaluated the behavior of the dialog managers for each system with recruited users using the same set of scenarios designed for the user simulation. A total of 600 dialogs were recorded from the interactions of 150 users (75 users employed DM_1 and 75 users employed DM_2, each user acquired

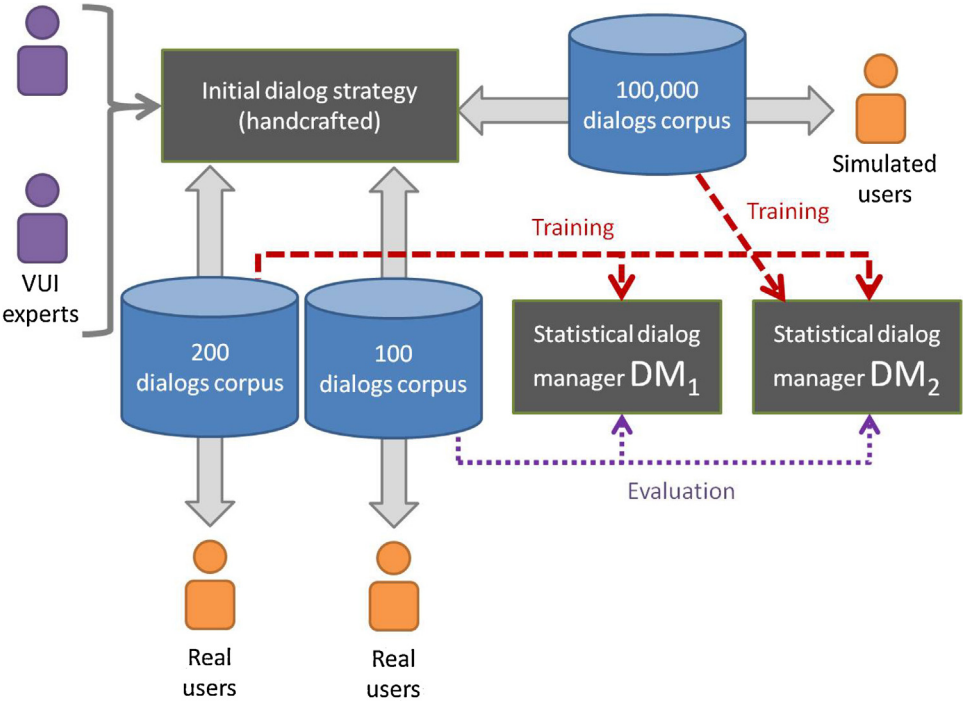

Fig. 3. Setup for the first experiment.

four dialogs). In this case, DM_1 was trained with the initial corpus of 300 dialogs, and DM_2 with the same 300 and the 100,000 simulated dialogs. The 300 dialogs acquired with each one of the dialog managers were used for their evaluation.

We carried out a detailed study of the dialogs obtained with both dialog managers using the set of quantitative evaluation measures proposed in (Ai et al., 2007). We computed the mean value for the defined evaluation measures, which we extracted from different studies (Ai et al., 2007; Schatzmann et al., 2006). We then used two-tailed *t*-tests to compare the means across the different types of scenarios and users as described in (Ai et al., 2007). The significance of the results was computed using the SPSS software with a significance level of 95%. Additionally, these users were asked to fill in a questionnaire with their opinion about several aspects of the interaction.

4.2. Experimental dialog systems

We have employed four dialog systems for evaluating our proposal: DI@L-log, UAH, LUNA, and EDECAN. Table 1 summarizes the main characteristics of the systems.

DI@L-log is a spoken dialog system which acts as a voice logbook to collect home monitored data from patients suffering from Type-2 diabetes (Black et al., 2005). The data collected by the system are the patient's weight, blood pressure (systolic and diastolic values) and sugar levels. The system validates and analyzes the data, providing some immediate feedback to the patients regarding their current progress as well as communicating the results to doctors who are able to review the patient's progress graphically and deal with any alerts generated by the system concerning abnormal developments.

The definition of the semantics for the task was carried out considering the information that is required to monitor the patients and inform them about their evolution. For the diabetes task, the *DR* is a sequence of four fields related to the information that the system requires to the user (*Weight*, *Sugar*, *Systolic-Pressure*, and *Diastolic-Pressure*).

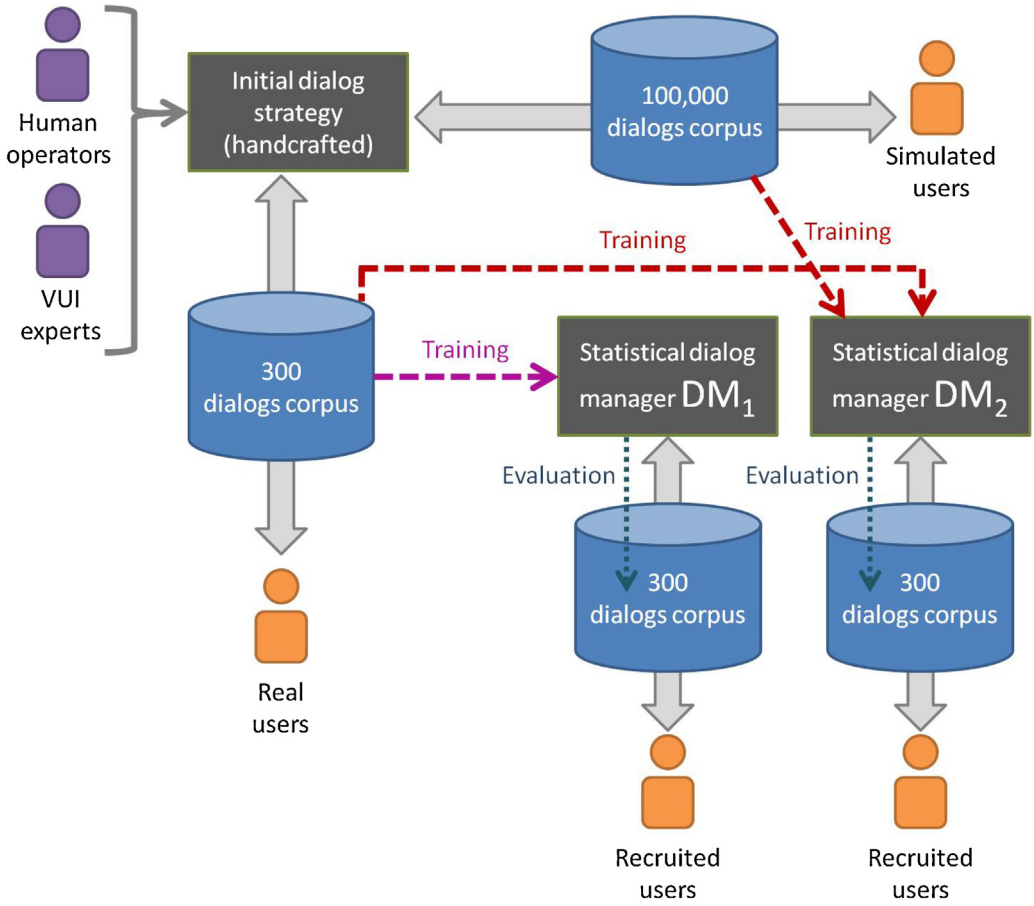

Fig. 4. Setup for the second experiment.

The architecture of the application used for the experimental setup includes a VoiceXML-compliant platform (Voxeo Evolution[2]), which also provides the ASR and TTS modules, and telephone access. Static VoiceXML files and grammars are stored in this voice server. These files were simplified by generating a VoiceXML file for each specific system prompt. Each file contains a reference to a grammar that defines the valid user's inputs for the corresponding system prompt. The dialog manager, implemented in Python and PHP, selects the next system prompt (i.e. VoiceXML file) by verifying the probabilities assigned by the statistical dialog manager to each system prompt given the current state of the dialog. This module is stored in an external web server and includes the data structure corresponding to the Dialog Register and the trained neural networks. The result generated by the statistical dialog manager informs the IVR platform about the most probable system prompt to be selected for the current dialog state. Then, the platform selects the corresponding VoiceXML file and reproduces it to the user.

The main purpose of the experiments carried out with the DI@L-log system was to evaluate the performance of our methodology in a system with restricted requirements, provided that the interaction is system-directed and thus it is the dialog system which prompts the user, whose role is just to answer the system questions. Taking the requirements of the task into account, an initial strategy was designed for the dialog manager. A corpus of 300 dialogs was acquired using this strategy. Although the original rule-based system was evaluated with patients suffering from diabetes (Black et al., 2005), the new version developed with our approach has been evaluated with recruited elderly users. Just two types of scenarios were created for the acquisition of dialogs. The first one allowed users to freely provide the different values and the second one informed them about the specific values that must be provided.

[2] http://evolution.voxeo.com/.

Table 1
Main characteristics and modules of the dialog systems used in the evaluation.

	Initiative	Error handling	Initial corpus collection	Initial DM	Domains
Di@l-log	System	No	Human–computer	Rule based	Questionnaire answering. Healthcare domain
UAH	Mixed	Yes	Human computer	Rule based	Information provision
LUNA	Mixed	Yes	Human–human and WOz technique	Not available	Problem solving. Help-desk domain
EDECAN	Mixed	Yes	Human–computer	Rule based	Information provision and booking operations

	ASR	SLU	NLG	TTS
Di@l-log	Loquendo	Grammars	VoiceXML prompts in a database	Loquendo
UAH	Verbio	Grammars	Dynamic VoiceXML prompts based in templates	Verbio
LUNA	Loquendo VoxNauta platform	Statistical approach based on a multilevel semantic and dialog annotation	Templates	Loquendo VoxNauta platform
EDECAN	CMU Sphinx-II with specific language and acoustic models developed in the project	Stochastic SLU approach	Dynamic templates	Loquendo

Fig. 5 shows an example of a simulated dialog for the DI@L-log task. The objective defined for the dialog was to collect the weight, sugar and pressure values. The values defined in the scenario are 12, 11, 160 and 80, respectively. A sentence in natural language, that is equivalent to the labeled system turn selected by the dialog manager, has been incorporated to clarify the explanation.

In this dialog, the system begins asking the user about his weight. As a low confidence score is introduced for the value provided by the user simulator in U1, the system decides to confirm this value in S2. Then, the system asks for the sugar value. The user simulator provides this value in U3 and a high confidence score is assigned. Therefore, this value has not to be confirmed by the system. The system asks for the systolic pressure in S4. An error is introduced in the value provided by the user simulator for this parameter (it changes 160 for 150) and a low confidence score is assigned to this value. Then, the system asks the user to confirm this value. The user simulation rejects this value in U5 and the system decides to ask for it again. Finally, the system asks for the diastolic pressure. This value is correctly introduced by the user simulator and the user simulator also assigns a high confidence score. Then, the system has the data required from the patient, analyzes the evolution of the patient, and informs him.

LOGIN PHASE

S1: (*Weight*)
Tell me your weight in stones.
U1: *Weight*: 12 [0.35]
S2: (*Confirmation-Weight*)
I have understood 12 stones. Is it correct?
U2: (*Acceptance*) [0.83]
S3: (*Sugar*)
What is your blood sugar?
U3: *Sugar*: 11 [0.90]
S4: (*Systolic*)
Tell me your blood systolic pressure.
U4: (*Systolic*): 150 [0.21]
S5: (*Confirmation-Systolic*)
I have understood 150. Is it correct?
U5: (*Rejection*) [0.89]
S6: (*Systolic*)
Tell me your systolic pressure.
U6: *Systolic*: 160 [0.98]
S7: (*Diastolic*)
And what is the lower number?
U7: *Diastolic*: 80 [0.89]

DATA ANALYSIS - FINAL ANSWER

Fig. 5. An example of a dialog extracted from the simulated corpus of the DI@L-log task.

S1: Opening Welcome to the UAH system. **U1:** (*Lecturers*) [0.15] I want to know information about lecturers. **S2:** (*Confirmation:Lecturers*) Do you want to know information about lecturers? **U2:** *Affirmation* [0.98] *Yes* **S3:** (*Question:Lecturer_Name*) Tell me the name of the lecturer. **U3:** *Lecturer_Name*: Ramón López-Cózar [0.23] Ramón López-Cózar.	**S4:** (*Confirmation:Lecturer_Name*) Do you want to know the information about Ramón López-Cózar? **U4:** *Affirmation* [0.86] Yes. **S5:** (*Answer:Lecturers*) (*New-Query*) {*Lecturer information*} Anything else? **U5:** (*Question:Registration*) [0.87] *Degree: Computer Science* [0.91] I want to know the registration information in the Computer Science degree. **S6:** (*Answer:Registration*) (*New-Query*) {*Registration information*} Anything else? **U6:** *Negation* No. **S7:** (*Closing*) Thank you for using the UAH system.

Fig. 6. An example of a dialog acquired for the UAH task by means of the simulation technique.

Universidad Al Habla (UAH – University on the Line) is a spoken dialog system developed to provide spoken access to academic information about the Department of Languages and Computer Systems in the University of Granada, Spain (Griol et al., 2012; Callejas and López-Cózar, 2008). The information that the system provides can be classified in four main groups: subjects, professors, doctoral studies, and registration. The *DR* defined for the system is a sequence of four fields related to the four task-dependent concepts that users can provide (*Subject*, *Lecturers*, *Doctoral studies*, *Registration*) and eight attributes (*Subject-Name*, *Degree*, *Group-Name*, *Subject-Type*, *Lecturer-Name*, *Program-Name*, *Semester*, and *Deadline*).

The system must ask the user for different pieces of information before producing a response following in most cases a mixed initiative. This way, the system acts in a more difficult domain that the Di@-log system. A set of 300 dialogs was acquired with an initial version of the UAH system by means of its interaction with students and professors of the University of Granada. The acquisition process resulted in a spontaneous Spanish speech dialog corpus with 60 different speakers, with small dialectal variants. A rule-based strategy for dialog management was designed for this initial system based on VoiceXML files (Callejas and López-Cózar, 2008).

The architecture of the system used for the experimental setup is similar to the one described for the DI@L-log system. We used the ASR and TTS modules provided by the Verbio VoiceXML toolkit from ATLAS.[3] The ASR module processes each user utterance provided by an Intel Dialogic D/41JCT-LS telephony card. An external web server stores the statistical dialog manager and the set databases of the application. An additional module, called GAG, was developed to automatically create dynamic ASR grammars (Callejas et al., 2007).

A set of 40 scenarios was defined to consider the different queries that may be performed by users. Two types of scenarios were specified. Type-1 scenarios defined only one objective for the dialog (e.g. to obtain timetable information of a specific subject). Type-2 scenarios defined two objectives for the dialog (e.g. to obtain timetables of a specific subject and registration deadlines for the corresponding degree). The recruited users to evaluate the system were university students. A web page was created providing details about the system, which was publicly available on the telephone so that users employed it spontaneously.

Fig. 6 shows an example of a simulated dialog for the UAH task with two objectives (to know timetables of a given lecturer and subject, and information about the registration in a specific degree). As in the example described in Fig. 5, sentences in natural language, that are equivalent to the labeled system and user turns, have been incorporated to clarify the explanation. It can also be seen that confidence measures are used with the dialog manager to require a confirmation each time a low level measure is assigned to a piece of information (e.g, the query about lectures in turn U1 and the name of the lecturer in turn U3).

The main objective of the LUNA system (Dinarelli et al., 2009) is to advance the state of the art in understanding conversational speech in spoken dialog systems. Three aspects of SLU are of particular concern in LUNA: generation

[3] http://www.verbio.com/.

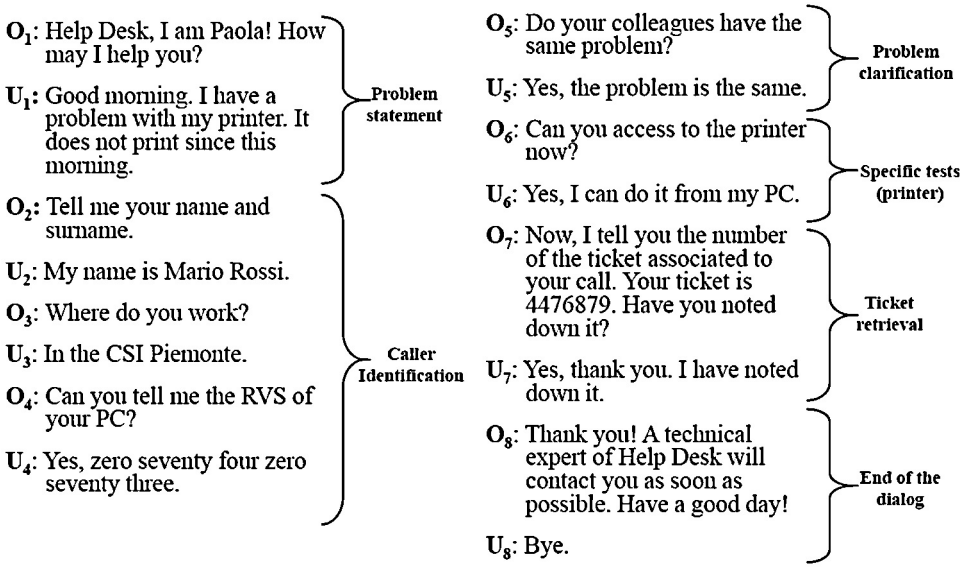

Fig. 7. An example of a dialog acquired for the LUNA task by means of the simulation technique.

of semantic concept tags, semantic composition into conceptual structures, and context sensitive validation using information provided by the dialog manager. In order to train and evaluate SLU models, different corpora of spoken dialogs in multiple domains (from transactional tasks to problem-solving) and languages (French, Italian and Polish) have been acquired. The dialogs were collected in the following application domains: stock exchange, hotel reservation and tourism inquiries, customer support service/help-desk and public transportation. The corpus for the IT help-desk system is currently being annotated with the aim to collect human–machine dialogs (HM) and human–human dialogs (HH). HH dialogs refer to real user conversations with agents engaged in a problem solving task in the domain of software/hardware repairing. HM dialogs are acquired with a Wizard of Oz approach (WoZ).

A corpus of 300 labeled dialogs (150 HH and 150 HM dialogs) has been used for the experiments shown in this paper. To the best of our knowledge this is the first SDS corpus (HM and HH dialogs) annotated with a multilayer approach to the annotation of lexical, semantic and dialog features (Griol et al., 2009b; Dinarelli et al., 2009). These features include dialog acts, attribute-values, and predicate argument structures.

The main purpose of the experiments with the LUNA system is to extend the experimentation described for the DI@L-log and UAH dialog systems to demonstrate that the proposed statistical methodology is applicable in a more complex domain for both human–machine (HM) and human–human (HH) conversations. For the LUNA task, the *DR* is a sequence of 15 fields related to the specific task information. The sequence of fields is *Name*, *Organization*, *Telephone*, *Address*, *Brand-Model*, *Machine-Code* and one field for each possible problem (*Printer*, *Network_Connection*, *PC_going_slow*, *Monitor*, *Keyboard*, *Mouse*, *Virus*, *CD-DVD_player*, and *Power-Supply*).

The practical system used for the evaluation required the adaptation of the design of the VoxNauta Platform[4] including the Loquendo ASR and TTS modules. The ASR output is sent to the SLU module using the HTK format for Word Lattice representation. Regarding the SLU module, models used in machine translation and models for predicate/argument extraction from partial parses were designed and tested (Coppola et al., 2009; Mosso et al., 2008). A external web server was also used to store the proposed statistical dialog manager. Ten different dialog scenarios inspired from the real services were defined considering ten possible problems (printer, PC network, slow computer, screen/video card, keyboard, mouse, office network, virus issues, CD-ROM reader, or PC power). The acquisition with recruited users was carried out by students and staff of our university.

Fig. 7 shows an example of a simulated dialog for the LUNA task. As it can be observed, the basic structure of the dialogs included in the LUNA corpora is usually composed by the sequence of the following tasks: *Opening*, *Problem-statement*, *User-identification*, *Problem-clarification*, *Problem-resolution*, *Ticket-retrieval*, and *Closing*. The

[4] http://www.loquendo.com/en/products/speech-platforms/.

shared plan is represented as a data register that encapsulates the task structure, dialog act structure, attribute-values and predicate-argument structure of utterances. During the *Problem-statement* task, the caller explains the reasons why he/she calls the help-desk. In the *User-identification* task, the operator asks for additional information regarding the identity of the caller. Once the caller has described the problem, the operator can ask for additional information to clarify it during the *Problem–clarification* task. During the *Problem–resolution* task, the operator asks the user to perform specific tests. In the *Ticket–retrieval* phase, the operator assigns a ticket number for the current call. The user must take note of this number and inform about this to the operator. Finally, the dialog participants end the dialog during the *Closing* phase.

EDECAN (Hurtado et al., 2010) is a multilingual, multimodal, mixed-initiative information system for booking sport facilities in the Technical University of Valencia, Spain. Users can ask for the availability, the booking or cancellation of facilities and the information about their current bookings. The languages involved in the system are Spanish, Catalan and Basque. The definition of the semantics of the EDECAN-SPORT task was carried out considering the different functionalities required for the booking system and the information to complete them. A set of 300 dialogs recorded at the telephone sport service of the University with 150 real users was labeled in terms of dialog acts to make this definition.

The main purpose of the experiments carried out with the EDECAN system has been to demonstrate that our methodology can be adapted to complex tasks in which an application manager with a complex regulation is required in the architecture of the dialog system. For the EDECAN task, the *DR* is a record structure of 10 fields, corresponding to the four task-dependent concepts (*Availability*, *Booking*, *Booked*, *Cancellation*) and six attributes (*Sport*, *Hour*, *Date*, *Court-Type*, *Court-Number*, and *Order-Number*) defined for the task.

The dialog system used for the practical experiments uses the CMU Sphinx-II speech recognition software.[5] The acoustic training was performed using a corpus acquired in a previous research project (Griol et al., 2008). The SLU module is based on stochastic models estimated by means of automatic learning techniques (Segarra et al., 2002). The statistical dialog manager and the *AM* were implemented in Python with the access to different PostgreSQL databases. The NLG module is based on the use of a set of templates associated to the different system dialog acts. The system integrates the Loquendo TTS system.

The recruited users were university students, who were provided with a brochure describing the scenarios that they were asked to complete, main functionalities of the system, and specific regulation defined by the sports center of the University.

Different levels of complexity were proposed to define a set of 15 scenarios for the acquisition. For instance, the first and the last of them have been coded as follows: Scenario-1 (<Availability> Sport [Court-Type] [Date] [Hour]); scenario-15 (<Booked> <Cancellation> [Sport] [Date] [Hour] [<Availability>] <Booking> Sport [Court-Type] Date Hour). Scenario-1 consists of a query about availability on a certain sport, allowing the user to specify date, hour, and court-type. Scenario-15 is a complex scenario, and it can be decomposed into three phases: (1) the user has to obtain the list of his/her booked courts; (2) the user has to cancel some courts of the previous list, and s/he can optionally provide the sport, the date, or the hour, to specify the booked court whose cancellation s/he wants; and (3) the user has to book some courts providing the sport, the date, and the hour, and s/he can supply the court-type, or can make an availability query. Thus, dialogs of complex scenarios are dialogs composed by sequences of sub-dialogs, and there are sub-dialogs that share information among them. This circumstance occurs between the BOOKED and CANCELLATION sub-dialogs, and also between the AVAILABILITY and BOOKING sub-dialogs.

Fig. 8 shows an example of a simulated dialog for the EDECAN task. The objective defined for the dialog was the booking of a tennis court for March the 14th at 8 o'clock. The dialog begins with a greeting turn generated by the system (S1). In the first turn, the simulated user provides the *Sport*, the *Date* and the *Hour*. The user simulator introduces an error value in the *Date* slot (it changes day 14 for day 15) and assigns confidence scores to the different slots. In this case, a low confidence is assigned to this erroneous value. In the second system turn, a confirmation for the *Date* is selected. Considering the information defined in the objective of the scenario, the user simulator selects a *Negation* dialog act and provides the correct value for the *Date* according to the objective (U2). In this turn, the user simulator assigns a high confidence value to the information provided by the user. In the following system turn (S3), the dialog manager selects to make a booking query to the database. There is only one court available. The user

[5] http://cmusphinx.sourceforge.net/.

S1: (*Opening*)
Welcome to the sport service. How can I help you?
U1: (*Booking*) [0.89]
Sport: Tennis [0.96]
Date: 03-15-2010 [0.12]
Hour: 08.00-09.00 [0.85]
I want to book a tennis court for 14th March at 8'clock in the morning.
S2: (*Confirmation-Date*)
Do you want to play on the 15th March?
U2: (*Negation*) [0.79]
Date: 03-14-2010 [0.94]
No, I want to play on the 14th March.
S3: (*Confirmation-Booking*) *Sport*: Tennis *Date*: 2010-03-14 *Hour*: 08.00-09.00
{*One court available*} Do you want to book the tennis court number 2?
U3: (*Affirmation*)
Yes.
S4: (*Booking*) *Sport*: Tennis *Date*: 2008-03-14 *Hour*: 08.00-09.00 (*New-Query*)
The tennis court number 2 has been booked. Do you want anything else?
U4: (*Negation*) [0.96]
No.
S5: (*Closing*)
Thank you for using the sport service. Goodbye.

Fig. 8. An example of a dialog extracted from the simulated EDECAN corpus.

simulator confirms the booking in U3. Once the objective of the dialog is achieved, the user simulator selects a request for closing the dialog in the following turn (U4).

For illustration purposes, Fig. 9 shows two dialogs respectively acquired with the LUNA and EDECAN dialog systems using the described scenarios for the acquisition with recruited users.

5. Discussion of results

As described in Section 4.1, the experimental setup can be divided into two fundamental experiments. In the first experiment we have used the initial and the simulated corpora for each task to analyze the quality of the responses provided by the dialog manager (Section 5.1). The second experiment is based on an objective and subjective assessment with recruited users (Section 5.2).

5.1. Evolution of the dialog strategy

From our previous work on statistical dialog management (Griol et al., 2008), we propose four measures to evaluate the performance of the dialog manager. The first measure, which we call *Unseen Situations*, makes reference to the percentage of dialog states that are present in the test partition but are not present in the corpus used for learning the dialog manager. As described in Section 3, dialog states are represented as a pair < *DR, last system response*>, which corresponds to the input to the first classifier of the dialog manager. For instance, a dialog state in the EDECAN system could be:

- *DR*: The user has indicated that he/she wants to make a booking for a certain sport.
- Last response of the system: Require booking date.

It is important to note that we do not take into account the exact values provided by the user, but only the codification of the *DR* (whether each piece of information has been provided or not and if it was recognized with low or high confidence).

The other three measures are calculated by comparing the answer automatically generated by the dialog manager for each input in the test partition with regard to the reference answer annotated in the corpus. This way, the evaluation is carried out turn by turn. Thus, the aim is not to evaluate the complete dialog as a unit, but to assess the appropriateness of the dialog manager response for each sample in the test partition. Nevertheless, the dialog history is taken into

Scenario: Book a basketball court for a specific date. Cancel a previous booking after accessing the list of booked courts.

S1: Welcome to the sport service. How can I help you?
U1: I want to book a basketball court.
S2: You can see the available courts on the screen.
U2: I want a court for November 12th in the afternoon.
S3: Sorry, do you want a court from 15 to 21 hours?
U3: Yes.
S4: There are several courts available on Friday November 12th. Please select one of them.
U4: The one at 7pm.
S5: There is only one court with the required features, do you want to book it?
U5: Yes.
S6: The basketball court has been booked. Anything else?
U6: Yes, I want to know my list of booking courts.
S7: You can see the courts that you have booked on the screen. Anything else?
U7: Yes, cancel the one at 12 am.
S8: The specified court has been released. Anything else?
U8: No, thank you.
S9: Thank you for using the sport service. Goodbye.

Scenario: Inform about a specific problem with your printer.

S1: Help Desk, I am Paola! How may I help you?
U1: Hello I have a problem with the printer. It can no longer print. Probably it is a the toner discharge.
S2: Where do you work?
U2: For CSI Piemonte.
S3: Tell me your name and surname.
U3: My name is Ugo Foscolo.
S4: I have not heard you well. Please repeat trying to talk louder.
U4: Ugo Foscolo.
S5: Can you tell me your phone number?
U5: Zero eleven forty fifty sixty-three.
S6: Can you tell me the brand and model?
U6: Epson EPL nine thousand.
S7: Is the printer on now?
U7: It is on.
S8: Can you tell me the message on the printer display?
U8: No network connection is written.
S9: Can you see the active printer from the control panel of your PC?
U9: It is active.
S10: Now, I tell you the number of the ticket associated to your call. Your ticket number is 2609192. Have you noted down it?
U10: Yes.
S11: Thank you! A technical expert of Help Desk will contact you as soon as possible. Have a nice day!

Fig. 9. Examples of the dialogs acquired with recruited users and the EDECAN (top, translation from Spanish to English) and LUNA (bottom, translation from Italian to English) dialog systems.

account for the specification of the dialog state, and this state is used to compare the response provided by the dialog manager with the one annotated in the test partition.

The three measures used for the evaluation described are:

- *Matching*: the percentage of responses provided by the dialog manager that are equal to the reference answer in the corresponding turn of the test corpus;
- *Coherence*: the percentage of answers provided by the dialog manager that are coherent with the current state of the dialog although they are not necessarily the same that the reference answer;
- *Error*: the percentage of answers provided by the dialog manager that would cause the failure of the dialog.

The measure *Matching* is automatically calculated, evaluating whether the answer provided by the dialog manager is the same that the reference answer in the corpus. If a dialog situation is observed with several associated responses

Table 2
Evaluation results obtained with the DMs trained for the different systems.

	Unseen	Matching	Coherence	Error
DI@L-log				
DM_1	1.16%	95.25%	98.29%	1.71%
DM_2	0.14%	97.12%	99.85%	0.15%
UAH				
DM_1	3.54%	93.16%	94.71%	5.29%
DM_2	1.12%	81.12%	97.52%	2.48%
LUNA				
DM_1	7.50%	90.56%	92.21%	7.79%
DM_2	5.05%	78.89%	96.12%	4.16%
EDECAN				
DM_1	12.23%	87.36%	91.29%	8.71%
DM_2	4.27%	81.23%	96.09%	3.48%

in the training corpus, we consider that the current and the observed situation match if the dialog manager's response matches one of these associated responses.

The calculation of the *Coherence* and *Error* measures requires expert annotation of the corpus. Thus, to decide about coherence of system responses, we asked each annotator to answer the following question: "Given the current dialog state and the current response generated by the Application Manager: does it make sense that the system generates this prompt?" An example of a system responses annotated as coherent for the EDECAN system consisted of the system asking about the date or the hour once the user has required the booking of a court for a specific sport.

The answers labeled as *Error* correspond to those that have not been considered coherent. For example, in the EDECAN system, there were cases in which the system decided to book a facility without one or more required items (for example, date, hour, sport, and court type).

Table 2 shows the results of the evaluation. The results for the *Unseen* measure show that the simulation technique reduces the initial values for each system. The results for the rest of measures show the satisfactory operation of the statistical dialog management methodology for the different systems. This way, the values obtained for the coherence measure range between 99.85% for the dialog manager developed for the DI@L-log task and 96.09% for the dialog manager developed for the EDECAN task. The percentage of system responses that would cause the failure of the dialog is also very small and ranges between 0.15% for the DM_2 dialog manager developed for the DI@L-log task and 3.48% for the DM_2 dialog manager developed for the EDECAN task). For all systems, the DM_2 dialog manager improves the results obtained for the DM_1 dialog manager.

With regard the DI@L-log task, the reduced number of possible dialog situations explains the small differences between the DM_1 and the DM_2 dialog managers, given that the structure of the dialogs in the initial corpus acquired with real users is almost the same in the complete set of dialogs. For this reason, similar values are obtained for the four measures defined for the evaluation. The use of the automatic dialog generation technique to acquire the simulated corpus allows reducing the percentage of unseen situations to only 0.14% for the DM_2, increases the coherence measure to 97.52%, and reduces the number of erroneous answers to only 0.15%. This shows that even for a simple task with a easy definition of the dialog strategy, the proposed methodology improves the initial results.

For the UAH, LUNA and EDECAN tasks, the values obtained for the matching and coherence measures show that the DM_2 dialog manager deviates from the initial dialog model and provides new valid paths to achieve each one of the required objectives defined in each task. This way, the values obtained for the matching measure in the three systems are reduced in the dialog manager whereas the coherence measure is increased. An in-depth study of the system responses provided by these dialog managers showed that the enhanced dialog models allow the dialog managers to tackle new situations and generate new coherent answers for the situations already present in the initial corpus. After the learning process, the dialog manager can ask for the required information using different orders, confirm these information items taking into account the confidence scores, reduce the number of system turns for the different kinds of dialogs, automatically detect different valid paths to achieve each of the required objectives, etc. Moreover, the codification developed to represent the state of the dialog and the good operation of the MLP classifier make it possible for the

Table 3
High-level dialog measures obtained for DM_1 and DM_2. Dialog success rate (M_1), Average number of turns per dialog (M_2), Percentage of different dialogs (M_3), Repetitions of the most observed dialog (M_4), Average number of actions per turn (M_5), Number of user turns of the most observed dialog (M_6), Number of user turns of the shortest dialog (M_7), Number of user turns of the longest dialog (M_8), Confirmation rate (M_9), Error correction rate (M_{10}).

	DI@L-log		UAH		LUNA		EDECAN	
	DM_1	DM_2	DM_1	DM_2	DM_1	DM_2	DM_1	DM_2
M_1	95.0%	97.0%	86.0%	93.0%	87.0%	94.0%	89.0%	96.0%
M_2	12.1	11.8	14.4	11.2	22.5	18.2	23.1	17.3
M_3	26.6%	33.7%	66.7%	81.5%	74.8%	83.3%	84.8%	89.2%
M_4	16	11	7	4	8	5	5	3
M_5	1.0	1.2	1.3	1.6	1.2	1.6	1.3	1.7
M_6	7	6	6	5	11	9	12	8
M_7	4	3	7	4	8	5	9	7
M_8	11	8	15	10	17	11	15	12
M_9	38%	33%	34%	26%	31%	25%	39%	36%
M_{10}	0.87%	0.89%	0.87%	0.98%	0.87%	0.96%	0.89%	0.95%

number of answers that cause the failure of the system to be only 3.48% for the DM_2 dialog manager developed for the EDECAN system.

5.2. Evaluation with real users

Finally, we evaluated the dialog managers with recruited users. As explained in Section 4.1, a total of 600 dialogs were recorded from the interactions of 150 users, 75 users employed DM_1 and 75 users employed DM_2 as described in Fig. 4. An objective (Section 5.2.1) and subjective (Section 5.2.2) evaluations were carried out.

5.2.1. Objective evaluation

We considered the following high-level measures for the objective evaluation: (i) dialog success rate; (ii) dialog length: average number of turns per dialog, number of turns of the shortest dialog, number of turns of the longest dialog, and number of turns of the most observed dialog; (iii) different dialogs: percentage of different dialogs with respect to the total number of dialogs, and number of repetitions of the most observed dialog; (iv) turn length: average number of actions per turn; (v) participant activity: number of turns in the most observed, shortest and longest dialogs; (v) confirmation rate, computed as the ratio between the number of explicit confirmation turns and the total number of turns in the dialog; and (vi) error correction rate, computed as the number of errors detected and corrected by the dialog manager divided by the total number of errors.

Table 3 presents the results of the objective evaluation. As can be observed, both dialog managers could interact correctly with the users in most cases for the four systems. However, the DM_2 dialog manager obtained a higher success rate, improving the initial results by a value ranging between 2% absolute for the DI@L-log system and 7% absolute for the LUNA and EDECAN systems. Using the DM_2 dialog manager, the average number of required turns is also reduced from 12.1 to 11.8 in the DI@L-log system, 14.4 to 11.2 in the UAH system, 22.5 to 18.2 in the LUNA system, and 23.1 to 17.3 in the EDECAN system. These values are slightly higher for both systems regarding the results obtained with the user simulator as in some dialogs the recruited users provided additional information which was not mandatory or asked for additional information that was not specified in the scenarios.

For all systems, it can also be observed that when DM_2 was used, there was a reduction in the average number of turns and in the number of turns in the longest, shortest and most observed dialogs. These results show that improving the dialog strategy made it possible to reduce the number of necessary system actions to attain the dialog goals for the different tasks. In addition, the results show a higher variability in the dialogs generated with DM_2 as there was a higher percentage of different dialogs and the most observed dialog was less repeated. There was also a slight increment in the mean values of the turn length for the dialogs collected with DM_2 due to the better selection of the system actions in the improved strategy. As in the previous section, the most significant differences were observed for the dialog models learned for the UAH, LUNA, and EDECAN dialog systems.

Table 4
Percentages of user and system dialog acts using DM_1 and DM_2.

	DI@L-log		UAH		LUNA		EDECAN	
	DM_1	DM_2	DM_1	DM_2	DM_1	DM_2	DM_1	DM_2
U_Request	9.08	10.44	34.42	35.13	34.75	38.81	43.19	47.42
U_Provide	36.24	37.42	22.15	26.04	26.31	27.15	25.70	25.07
U_Confirm	17.26	16.48	9.47	7.03	10.73	8.06	8.53	7.14
U_Yes/No	37.21	35.39	32.93	30.84	26.67	24.83	21.15	19.36
U_Other	0.21	0.27	1.03	0.96	1.54	1.15	1.43	1.01
S_Confirm	38.04	33.23	34.43	26.14	30.71	25.33	39.02	36.27
S_Request	38.06	40.30	17.24	18.51	29.17	28.02	20.10	18.26
S_Inform	23.83	26.38	48.02	55.16	39.33	46.04	40.45	45.21
S_Other	0.07	0.09	0.31	0.19	0.79	0.61	0.43	0.26

The confirmation and error correction rates were also improved by using DM_2 as it required less data from the user, thus reducing the number of ASR errors. A problem occurred when the user input was misrecognized but it had high confidence score, in which case it was forwarded to the dialog manager. However, as the success rate shows, this problem did not have a remarkable impact on system performance.

The results are better than the usually reported in commercial systems. This may be because in most cases the users were undergraduate students of our universities. Thus, they had a good technical background and a positive inclination to technology, whereas in commercial systems the age range is wider and the technological knowledge and affinity varies. Nevertheless, as described in our previous work (Callejas et al., 2012) in some cases the differences between these users are only significant in the extreme cases (e.g., technophobic old users and technophile young users).

For dialog style features, we measured the balance between different types of dialog acts. On the system side, we have measured the confirmation of concepts and attributes, questions to require information, and system answers generated after a database query. On the user side, we have measured the percentage of turns in which the user carries out a request to the system, provides information, confirms a concept or attribute, the yes/no answers, and other answers not included in the previous categories. The experimental results are set out in Table 4. It can be observed that there are significant differences for both dialog models in the different tasks. On the one hand, users needed to employ less confirmation turns with DM_2, which explains the higher proportion of dialog acts to request and provide information. On the other hand, using DM_2 there was an increment in the number of system turns providing information to the user, which is consistent with the fact that the task completion rate is higher using this dialog manager in the different systems.

In addition, we grouped all user and system actions into "goal directed" (actions to provide or request information) and "grounding" actions (dialog formalities, unrecognized actions, confirmations, and negations). The results of the evaluation (Table 5) show that the dialogs acquired with DM_2 are better as the proportion of goal-directed actions increases for all systems.

5.2.2. Subjective evaluation

One of the ultimate goals of designing and building spoken dialog systems is the optimization of the caller experience. This way, we also asked the users to complete a questionnaire to assess their subjective opinion about the system performance. The questionnaire had six questions: (i) Q1: *How well did the system understand you?*; (ii) Q2: *How*

Table 5
Percentages of goal directed and grounding actions using DM_1 and DM_2.

	DI@L-log		UAH		LUNA		EDECAN	
	DM_1	DM_2	DM_1	DM_2	DM_1	DM_2	DM_1	DM_2
Goal-directed actions	71.47	74.41	65.48	72.07	64.31	71.88	73.33	78.97
Grounding actions	28.53	25.59	34.52	27.93	35.69	28.12	26.67	21.03

Table 6
Results of the subjective evaluation with recruited users (1 = lowest, 5 = highest).

	DI@L-log		UAH		LUNA		EDECAN	
	DM_1	DM_2	DM_1	DM_2	DM_1	DM_2	DM_1	DM_2
Q1	4.8	4.8	4.6	4.7	4.7	4.7	4.6	4.7
Q2	4.5	4.6	3.6	3.9	4.1	4.3	4.3	4.5
Q3	4.5	4.6	3.8	4.3	3.9	4.6	4.1	4.7
Q4	4.1	4.2	3.4	4.2	3.8	4.5	3.9	4.6
Q5	4.2	4.3	3.2	3.3	3.6	3.8	3.8	3.9
Q6	4.3	4.6	4.1	4.6	4.0	4.5	4.3	4.7

well did you understand the system messages?; (iii) Q3: *Was it easy for you to get the requested information?*; (iv) Q4: *Was the interaction with the system quick enough?*; (v) Q5: *If there were system errors, was it easy for you to correct them?*; (vi) Q6: *In general, are you satisfied with the performance of the system?* The possible answers for each one of the questions were the same: *Never/Not at all*, *Seldom/In some measure*, *Sometimes/Acceptably*, *Usually/Well*, and *Always/Very Well*. All the answers were assigned a numeric value between one and five (in the same order as they appear in the questionnaire). Table 6 shows the average results of the subjective evaluation using the described questionnaire.

It can be observed that using either DM_1 or DM_2 the users perceived that the system understood them correctly. Moreover, they expressed a similar opinion regarding the easiness for correcting system errors. However, users said that it was easier to obtain the information specified for the different objectives using DM_2, and that the interaction with the system was more adequate with this dialog manager. Finally, the users were more satisfied with the system employing DM_2.

6. Conclusions and future work

In this paper, we have presented a corpus-based methodology for the development of statistical dialog managers and the optimization of dialog strategies. The methodology is based on the estimation of a statistical model from the sequences of system and user dialog acts obtained from a set of training data. The selection of the following system answer is based on a classification process that takes into account the history of the dialog. We have defined a codification of this information to facilitate the correct operation of the classification function. This representation allows the system to automatically generate a specialized answer that takes the current situation of the dialog into account.

As the dialog model is learned from a corpus of training samples, the performance of the dialog manager depends on the quality and size of the corpus used to learn the model. We have presented a technique for automatically acquiring a dialog corpus in which the simulated dialogs are automatically generated in the labeling format defined for the task. Therefore, the effort necessary to manually acquire and label this high number of dialogs, and learn a dialog manager is considerably reduced. In addition, error detection and correction techniques have also been developed. These techniques, which are based on the use of confidence scores and the definition of different kinds of confirmations, allow to distinguish error-prone situations and to make the necessary corrections to satisfactorily complete the task.

Task-dependent information is isolated from the model so the methodology can be applied to develop dialog managers for any application domain. This is done by defining a dialog register (*DR*) that takes into account whether the user has provided a given piece of information related to the task and also the confidence scores that the recognition and understanding modules have assigned to this piece of information. This allows not only to cope with the situations observed the training corpus, but also to manage unseen situations by selecting the most convenient system action.

We have applied the proposed methodology to develop a dialog manager for a number of different tasks, using resources already available for existing dialog systems (DI@L-log, UAH, LUNA, and EDECAN). We have evaluated the performance of the proposed technique applying it for the implementation of dialog managers for these systems. Firstly, we have carried out a comparative evaluation focused on the quality of the answers generated by the initial (baseline) and the proposed statistical dialog managers working in each system. To do this, a set of statistical measures

has been defined to measure the evolution of the dialog strategy, overall quality of the dialogs in terms of high-level dialog measures, and the proportion of different user and system dialog acts.

The experiments show that the number of coherent responses provided by the statistical dialog managers increases with respect the baselines, while the number of responses that lead to dialog failure decreases. Thus, although their implementation required less effort, they outperformed the baselines. The greatest differences were observed for the LUNA and EDECAN systems, whereas the improvement for simpler systems (DI@L-log and UAH) was smaller.

These results have been corroborated by means of objective and subjective evaluation with real users. The results obtained for the UAH, LUNA and EDECAN systems show that the proposed methodology can be used not only to develop new dialog managers, but also to explore new dialog strategies, as they allowed a higher variability in the generated dialogs. This indicates that the method was useful to find new valid paths to achieve the objectives, which permits developing enhanced versions of the already existing systems. Hence, the advantage of the methodology is clear, because implementing new dialog managers or testing existing ones using a non-statistical approach would require a considerable effort and time, which sometimes is not affordable.

With regard to the high-level dialog features, for the four experimental systems the improved dialog strategies led to a reduction in the average number of dialog turns and in the number of system actions required to attain the dialog goals for the different tasks. This improvement was achieved because our methodology enabled better selection of system actions. Regarding the dialog style and cooperativeness, the most important conclusion is the higher proportion of goal-directed actions with regard to grounding actions achieved with the new dialog strategies.

In the case of the LUNA system we have extended the evaluation to assess the performance of the statistical dialog manager by learning the dialog models from dialogs collected by means of different methodologies (real users and WOZ technique). The results show that the dialog manager developed for this system using our methodology can be adapted to human–human dialogs and human–machine dialogs. In addition, the results indicate that the dialog manager created using human–machine dialogs can also infer the structure of task-related aspects that are present in human–human dialogs.

The dialog managers created using the proposed technique achieved higher success rates, reduced the average number of dialog turns and improved the confirmation and error correction rates for the different tasks. Also, users reported higher satisfaction and found easier to obtain information using the enhanced systems. For simple tasks as the ones described for the DI@L-log and UAH systems, the advantages introduced by our method are less relevant; as it is possible to cost-efficiently handcraft very efficient rule-based managers, as described in the previous work for these systems. For more complex tasks, as the ones described for LUNA and EDECAN, for which the initial dialog manager had not been developed or a considerable effort had been invested in its development, our approach substantially helps the development process and allows the automatic generation of robust dialog models that efficiently covers the possible dialog situations for each task.

For future work, we plan to extend the evaluation presented in this paper by carrying out a study of the relationships between the subjective opinions and the values of the different statistical measures defined in this work. Also, very interesting proposals in areas related to dialog management could be employed along with our contribution to potentially improve the *DR* and the *AM* module (Acomb et al., 2007; Pieraccini et al., 2009). For instance, different approaches to SLU are described that allow overcoming the difficulties derived from vague or imprecise user inputs. Also, we could consider to include in the *DR* information regarding the user, such as their profile, emotional state and current position. We could include in the *AM* module additional information concerning the specific task the system has been designed for, for example, the time it would take for a general user to complete a troubleshooting step. Another interesting topic for future work is to replicate the evaluation of the dialog manager considering the impact that each of the parameters of the automatic dialog generation technique has in each of the results, or even determine the number of simulated dialogs required for the dialog manager to achieve a certain performance. Finally, we also want to evaluate our systems with wider populations using crowdsourcing approaches, as discussed in Suendermann and Pieraccini (2013).

Acknowledgements

We want to thank the Signals and Interactive Systems Lab at the University of Trento (Italy), the Group of Natural Language Engineering and Pattern Recognition at the Technical University of Valencia (Spain), the Computer Science Research Institute at University of Ulster (Northern Ireland), and the Spoken and Multimodal Dialog Systems Group at

University of Granada (Spain) for their valuable help during the last two years providing the described dialog systems and facilitating the evaluation of our proposal.

We also thank the reviewers for their comments, suggestions and corrections that have significantly improved the quality of this paper.

References

Abdennadher, S., Aly, M., Bühler, D., Minker, W., Pittermann, J., 2007. Becam tool – a semi-automatic tool for bootstrapping emotion corpus annotation and management. In: Proc. of the International Conference on Spoken Language Processing (Interspeech'2007), Antwerp, Belgium, pp. 946–949.

Acomb, K., Bloom, J., Dayanidhi, K., Hunter, P., Krogh, P., Levin, E., Pieraccini, R., 2007. Technical support dialog systems: issues, problems, and solutions. In: Proc. of NAACL-HLT-Dialog'07 Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies, Rochester, NY, USA, pp. 25–31.

Ai, H., Raux, A., Bohus, D., Eskenazi, M., Litman, D., 2007. Comparing Spoken Dialog Corpora Collected with Recruited Subjects versus Real Users. In: Proc. of the 8th SIGdial Workshop on Discourse and Dialogue, Antwerp, Belgium, pp. 124–131.

Barnard, E., Halberstadt, A., Kotelly, C., Phillips, M., 1999. A Consistent Approach To Designing Spoken-dialog Systems. In: Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'99), Keystone, CO, USA, pp. 1173–1176.

Beskow, J., Edlund, J., Granström, B., Gustafson, J., Skantze, G., Tobiasson, H., 2009. The monami reminder: a spoken dialogue system for face-to-face interaction. In: Proc. of the International Conference on Spoken Language Processing (Interspeech'2009), Brighton, UK, pp. 296–299.

Black, L.-A., McTear, M.F., Black, N.D., Harper, R., Lemon, M., 2005. Appraisal of a conversational artefact and its utility in remote patient monitoring. In: Proc. of the 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05), Dublin, Ireland, pp. 506–508.

Callejas, Z., Griol, D., Engelbrecht, K., López-Cózar, R., 2012. A clustering approach to assess real user profiles in spoken dialogue systems. In: Proc. of IWSDS 2012: International Workshop on Spoken Dialog Systems: Towards a Natural Interaction with Robots, Knowbots and Smartphones, pp. 9–18.

Callejas, Z., López-Cózar, R., 2007. Automatic creation of ASR grammar rules for unknown vocabulary applications. In: Proc. of the 8th International workshop on Electronics, Control, Modelling, Measurement and Signals (ECMS'07), Liberec, Czech Republic, pp. 50–55.

Callejas, Z., López-Cózar, R., 2008. Relations between de-facto criteria in the evaluation of a spoken dialogue system. Speech Communication 50, 646–665.

Chu, S., O'Neill, I., Hanna, P., McTear, M., 2005. An approach to multistrategy dialogue management. In: Proc. of the 9th International Conference on Spoken Language Processing (Interspeech'05-Eurospeech), Lisbon, Portugal, pp. 865–868.

Cohn, D.A., Atlas, L., Ladner, R., 1994. Improving generalization with active learning. Machine Learning 15, 201–221.

Coppola, B., Moschitti, A., Riccardi, G., 2009. Shallow Semantic Parsing for Spoken Language Understanding. In: Proc. of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT'09), Boulder, CO, USA, pp. 85–88.

Cuayáhuitl, H., Renals, S., Lemon, O., Shimodaira, H., 2005. Human–Computer Dialogue Simulation Using Hidden Markov Models. In: Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'05), San Juan, Puerto Rico, pp. 290–295.

Dinarelli, M., Tonelli, S., Moschitti, A., Riccardi, G., 2009. Annotating Spoken Dialogs: from Speech Segments to Dialog Acts and Frame Semantics. In: Proc. of the EACL 2009 Workshop on Semantic Representation of Spoken Language, Athens, Greece, pp. 34–41.

Dutoit, T., 1996. An Introduction to Text-To-Speech Synthesis. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Fabbrizio, G.D., Tur, G., Hakkani-Tür, D., Gilbert, M., Renger, B., Gibbon, D., Liu, Z., Shahraray, B., 2008. Bootstrapping spoken dialogue systems by exploiting reusable libraries. Natural Language Engineering 14, 313–335.

Fraser, M., Gilbert, G., 1991. Simulating speech systems. Computer Speech and Language 5, 81–99.

Gibbon, D., Mertins, I., Roger, K. (Eds.), 2000. Handbook of Multimodal and Spoken Dialogue Systems: Resources, Terminology and Product Evaluation. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Griol, D., Callejas, Z., López-Cózar, R., 2009a. A Comparison between Dialog Corpora Acquired with Real and Simulated Users. In: Proc. of the 10th Annual SIGDIAL Meeting on Discourse and Dialogue, London, UK, pp. 326–332.

Griol, D., Hurtado, L., Segarra, E., Sanchis, E., 2008. A Statistical Approach to Spoken Dialog Systems Design and Evaluation. Speech Communication 50, 666–682.

Griol, D., Hurtado, L.F., Segarra, E., Sanchis, E., 2006. Managing unseen situations in a Stochastic Dialog Model. In: Proc. of AAAI Workshop Statistical and Empirical Approaches for Spoken Dialogue Systems, Boston, USA, pp. 25–30.

Griol, D., Molina, J.M., Callejas, Z., 2012. Bringing together commercial and academic perspectives for the development of intelligent AmI interfaces. Journal of Ambient Intelligence and Smart Environments 4, 183–207.

Griol, D., Riccardi, G., Sanchis, E., 2009b. Learning the structure of human–computer and human–human dialogs. In: Proc. of the International Conference on Spoken Language Processing (Interspeech'2009), Brighton, UK, pp. 703–706.

Heeman, P., 2007. Combining reinforcement learning with information-state update rules. In: Proc. of the 8th Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'07), Rochester, NY, USA, pp. 268–275.

Heinroth, T., Minker, W., 2012. Introducing Spoken Dialogue Systems into Intelligent Environments. Kluwer Academic Publishers/Springer-Verlag, New York, USA.

Hempel, T., 2008. Usability of Speech Dialog Systems: Listening to the Target Audience. Springer, Berlin Heidelberg, Germany.

Hori, C., Ohtake, K., Misu, T., Kashioka, H., Nakamura, S., 2009. Recent Advances in WFST-based Dialog System. In: Proc. of the International Conference on Spoken Language Processing (Interspeech'2009), Brighton, UK, pp. 268–271.

Hurtado, L., Planells, J., Segarra, E., Sanchis, E., Griol, D., 2010. A Stochastic Finite-State Transducer Approach to Spoken Dialog Management. In: Proc. of the International Conference on Spoken Language Processing (Interspeech'2010), Makuhari, Chiba, Japan, pp. 3002–3005.

Lane, I., Ueno, S., Kawahara, T., 2004. Cooperative dialogue planning with user and situation models via example-based training. Proc. of Workshop on Man–Machine Symbiotic Systems , Kyoto, Japan, pp. 2837–2840.

Laroche, R., Putois, G., Bretier, P., Young, S., Lemon, O., 2008. Requirements Analysis and Theory for Statistical Learning Approaches in Automaton-Based Dialogue Management. Technical Report School of Informatics. Edinburgh University, Edinburgh, UK.

Lee, C., Jung, S., Kim, K., Lee, G.G., 2010. Hybrid approach to robust dialog management using agenda and dialog examples. Computer Speech and Language 24 (4), 609–631.

Lemon, O., 2011. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. Computer Speech & Language 25, 210–221.

Lemon, O., Pietquin, O., 2012. Data-Driven Methods for Adaptive Spoken Dialogue Systems. Computational Learning for Conversational Interfaces. Springer, New York, USA.

Levin, E., Pieraccini, R., 1997. A stochastic model of human–machine interaction for learning dialog strategies. In: Proc. of the European Conference on Speech Communications and Technology (Eurospeech'97), Rhodes, Greece, pp. 1883–1896.

Levin, E., Pieraccini, R., Eckert, W., 2000. A stochastic model of human–machine interaction for learning dialog strategies. IEEE Transactions on Speech and Audio Processing 8 (1), 11–23.

Liu, Y., Shriberg, E., 2005. Does active learning help automatic dialog act tagging in meeting data. In: Proc. of the International Conference on Spoken Language Processing (Interspeech'2005), Lisbon, Portugal, pp. 2777–2780.

López-Cózar, R., Araki, M., 2005. Spoken, Multilingual and Multimodal Dialogue Systems. John Wiley & Sons Publishers, Chichester, West Sussex, England.

López-Cózar, R., Callejas, Z., 2008. ASR post-correction for spoken dialogue systems based on semantic, syntactic, lexical and contextual information. Computer Speech and Language 50, 745–766.

López-Cózar, R., Callejas, Z., Griol, D., 2010. ASR post-correction for spoken dialogue systems based on semantic, syntactic, lexical and contextual information. Knowledge-Based Systems 23, 471–485.

López-Cózar, R., Callejas, Z., McTear, M., 2006. Testing the performance of spoken dialogue systems by means of an artificially simulated user. Artificial Intelligence Review 26, 291–323.

López, V., Eisman, E., Castro, J., Zurita, J., 2011. A case based reasoning model for multilingual language generation in dialogues. Expert Systems with Applications 39, 7330–7337.

McTear, M.F., 2004. Spoken Dialogue Technology: Towards the Conversational User Interface. Springer, London, UK.

Meng, H.H., Wai, C., Pieraccini, R., 2003. The use of belief networks for mixed-initiative dialog modeling. IEEE Transactions on Speech and Audio Processing 11, 757–773.

Minker, W., 1999. Design considerations for knowledge source representations of a stochastically-based natural language understanding component. Speech Communication 28, 141–154.

Minker, W., Haiber, U., Heisterkamp, P., Scheible, S., 2004. The SENECA spoken language dialogue system. Speech Communication 43, 89–102.

Minsky, M., 1975. A Framework for Representing Knowledge. In: The Psychology of Computer Vision. McGraw-Hill, New York, USA, pp. 211–277.

Mosso, S., Mori, R.D., Bechet, F., Hahn, S., Gubrynowicz, R., Damnati, G., Riccardi, G., 2008. Deliverable 6.3 – LUNA functionalities description. Technical Report LUNA Project – Spoken language understanding in multilingual communication systems.

O'Shaughnessy, D., 2008. Automatic speech recognition: history, methods and challenges. Pattern Recognition 41, 2965–2979.

O'Shea, J., Bandar, Z., Crockett, K., 2012. A multi-classifier approach to dialogue act classification using function words. Lecture Notes in Computer Science 7270, 119–143.

Paek, T., Horvitz, E., 2000. Conversation as action under uncertainty. In: Proc. of the 16th Conference on Uncertainty in Artificial Intelligence, San Francisco, USA, pp. 455–464.

Paek, T., Pieraccini, R., 2008. Automating spoken dialogue management design using machine learning: an industry perspective. Speech Communication 50, 716–729.

Pieraccini, R., 2012. The Voice in the Machine: Building Computers that Understand Speech. The MIT Press, Cambridge, MA, USA.

Pieraccini, R., Suendermann, D., Dayanidhi, K., Liscombe, J., 2009. Are we there yet? Research in commercial spoken dialog systems. Lecture Notes in Computer Science 5729, 3–13.

Planells, J., Hurtado, L., Sanchis, E., Segarra, E., 2012. An Online Generated Transducer to Increase Dialog Manager Coverage. In: Proc. of the International Conference on Spoken Language Processing (Interspeech'2012), Portland, USA.

Rojas-Barahona, L., Giorgino, T., 2009. Adaptable dialog architecture and runtime engine (adarte): A framework for rapid prototyping of health dialog systems. International Journal of Medical Informatics 78, 56–68.

Roy, N., Pineau, J., Thrun, S., 2000. Spoken dialogue management using probabilistic reasoning. In: Proc. of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00), Hong Kong, China, pp. 93–100.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning internal representations by error propagation. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1. MIT Press, Cambridge, MA, USA, pp. 319–362.

Schatzmann, J., Thomson, B., Young, S., 2007. Error simulation for training statistical dialogue systems. In: Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'07), Kyoto, Japan, pp. 526–531.

Schatzmann, J., Weilhammer, K., Stuttle, M., Young, S., 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. Knowledge Engineering Review 21, 97–126.

Segarra, E., et al., 2002. Extracting semantic information through automatic learning techniques. International Journal on Pattern Recognition and Artificial Intelligence 16, 301–307.

Singh, S., Kearns, M., Litman, D., Walker, M., 1999. Reinforcement learning for spoken dialogue systems. In: Proc. of Neural Information Processing Systems (NIPS'99), Denver, USA, pp. 956–962.

Singh, S., Litman, D., Kearns, M., Walker, M., 2002. Optimizing dialogue management with reinforcement leaning: experiments with the NJFun system. Journal of Artificial Intelligence 16, 105–133.

Suendermann, D., Pieraccini, R., 2012. One year of contender: what have we learned about assessing and tuning industrial spoken dialog systems? In: Proc. of NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data (SDCTD'12), Montreal, Canada, pp. 45–48.

Suendermann, D., Pieraccini, R., 2013. Crowdsourcing for industrial spoken dialog systems. In: Crowdsourcing for Speech Processing: Applications to Data Collection, Transcription and Assessment. Wiley, Stroudsburg, PA, USA.

Thomson, B., Schatzmann, J., Weilhammer, K., Ye, H., Young, S., 2007. Training a real-world POMDP-based Dialog System. In: Proc. of NAACL-HLT-Dialog'07 Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies, Rochester, NY, USA, pp. 9–16.

Torres, F., Hurtado, L., García, F., Sanchis, E., Segarra, E., 2005. Error handling in a stochastic dialog system through confidence measures. Speech Communication 45, 211–229.

Torres, F., Sanchis, E., Segarra, E., 2008. User simulation in a stochastic dialog system. Computer Speech and Language 22, 230–255.

Traum, D., Larsson, S., 2003. The information state approach to dialogue management. In: Current and New Directions in Discourse and Dialogue. Kluwer, Dordrecht, The Netherlands, pp. 325–353.

Tsilfidis, A., Mporas, I., Mourjopoulos, J., Fakotakis, N., 2013. Automatic speech recognition performance in different room acoustic environments with and without dereverberation preprocessing. Computer Speech & Language 27, 380–395.

Venkataraman, A., Stolcke, A., Shriberg, E., 2002. Automatic dialog act labeling with minimal supervision. In: Proc. of the 9th Australian International Conference on Speech Science & Technology, Sydney, Australia.

Vipperla, R., Wolters, M., Renals, S., 2012. Spoken dialogue interfaces for older people. In: Advances in Home Care Technologies. IOS Press, Amsterdam, The Netherlands, pp. 118–137.

Walker, M., Hindle, D., Fromer, J., Fabbrizio, G.D., Mestel, C., 1997. Evaluating Competing Agent Strategies for a Voice Email Agent. In: Proc. of the European Conference on Speech Communications and Technology (Eurospeech'97), Rhodes, Greece, pp. 2219–2222.

Wilks, Y., Catizone, R., Worgan, S., Turunen, M., 2011. Some background on dialogue management and conversational speech for dialogue systems. Computer Speech and Language 25, 128–139.

Williams, J., 2008. The best of both worlds: Unifying conventional dialog systems and POMDPs. In: Proc. of the International Conference on Spoken Language Processing (InterSpeech'08), Brisbane, Australia, pp. 1173–1176.

Williams, J., Poupart, P., Young, S., 2006. Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management. In: Recent Trends in Discourse and Dialogue. Springer, Dordrecht, The Netherlands, pp. 191–217.

Williams, J., Young, S., 2007. Partially observable markov decision processes for spoken dialog systems. Computer Speech and Language 21 (2), 393–422.

Wu, W.-L., Lu, R.-Z., Duan, J.-Y., Liu, H., Gao, F., Chen, Y.-Q., 2010. Spoken language understanding using weakly supervised learning. Computer Speech & Language 24, 358–382.

Young, S., 2002. The Statistical Approach to the Design of Spoken Dialogue Systems. Technical Report Cambridge University Engineering Department.

Young, S., Gasic, M., Thomson, B., Williams, J., 2013. Pomdp-based statistical spoken dialogue systems: a review. In: Proc. of the IEEE, Montreal, Canada, pp. 1–18).

Young, S., Schatzmann, J., Weilhammer, K., Ye, H., 2007. The Hidden Information State Approach to Dialogue Management. In: Proc. of the 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Honolulu, Haway, USA, pp. 149–152.

Young, S., Williams, J., Schatzmann, J., Stuttle, M., Weilhammer, K., 2005. The Hidden Information State Approach to Dialogue Management. Technical Report Department of Engineering. University of Cambridge, Cambridge, UK.