
Universidad
Carlos III de Madrid

e-Archivo
Institutional Repository

This is a postprint version of the following published document:

Griol, D., Carbo, J. & Molina, J. M. (2013). Bringing context-aware access to the web through spoken interaction. *Applied Intelligence*, 38(4), 620-640.
DOI: http://dx.doi.org/10.1007/s10489-012-0390-8

© 2013 Springer US

Bringing context-aware access to the web through spoken interaction

David Griol, Javier Carbo, José Manuel Molina

Group of Applied Artificial Intelligence (GIAA), Computer Science Department, Carlos III University of Madrid, Madrid, Spain
e-mail: david.griol@uc3m.es e-mail: javier.carbo@uc3m.es e-mail: josemanuel.molina@uc3m.es

Abstract The web has become the largest repository of multimedia information and its convergence with telecommunications is now bringing the benefits of web technology to handheld devices. To optimize data access using these devices and provide services which meet the user needs through intelligent information retrieval, the system must sense and interpret the user environment and the communication context. In addition, natural spoken conversation with handheld devices makes possible the use of these applications in environments in which the use of GUI interfaces is not effective, provides a more natural human-computer interaction, and facilitates access to the web for people with visual or motor disabilities, allowing their integration and the elimination of barriers to Internet access. In this paper, we present an architecture for the design of context-aware systems that use speech to access web services. Our contribution focuses specifically on the use of context information to improve the effectiveness of providing web services by using a spoken dialog system for the user-system interaction. We also describe an application of our proposal to develop a context-aware railway information system, and provide a detailed evaluation of the influence of the context information in the quality of the services that are supplied.

Keywords Context-aware systems, Spoken dialog systems, User adaptation, Web interfaces, Systems evaluation

1 Introduction

The Web is becoming more and more pervasive as an application platform and its convergence with telecommunications is now bringing the benefits of web technology to hand-held devices. The widespread use of new mobile technology implementing wireless communications enables a new type of advanced applications to access information services on the Internet.

As the trend to an increasing number of ubiquitous, connected devices continues to grow, the heterogeneity of client capabilities and the number of methods for accessing information services also increases. Thus, effectively bringing information and services to people can only be done by supporting pervasive computing, in which technical details are transparent to the user and the access to services is guaranteed independently of his location and computing devices. Contextual information can therefore be used to select the services and enhance them for a better adaptation to the communication channel, the user environment and device, and the user preferences and needs.

Additionally, consumers expect web services to be accessible from all these devices in a similar way. As personal devices continue to shrink in size yet expand their capabilities, the conventional GUI model becomes increasingly cumbersome to use. Speech and language technologies allow users to communicate in a natural, flexible, and efficient manner. In addition, they make it possible to access applications when traditional input interfaces cannot be used (e.g. in-car applications, access for disabled people, etc.). In order

to develop these interfaces, in the last years there has been an increasing interest in simulating human-to-human communication, employing spoken dialog systems [25, 30, 36, 40]. A dialog system can be understood as an automatic system which functionality is accessible through natural language, emulating human conversation.

Several authors [31, 35, 54] have highlighted the importance of standardizing and sharing a common base for context sensitivity and web services systems. However, most context-aware systems are closed, composed of highly coupled constituents, and generated ad-hoc for a specific domain [6, 54]. The same problem occurs when designing a dialog system. There is a high variety of applications in which dialog systems can be used, some of the most widespread are information retrieval from the web [11], database systems [28], and recommendation systems [9, 24]. However, these systems are also usually designed ad-hoc for their specific domain using rule-based models and standards in which developers must specify each one of the steps to be followed by the system. However, these systems are also usually designed ad-hoc for their specific domain using rule-based models and standards in which developers must specify each one of the steps to be followed by the system. This makes it difficult to adapt the resulting systems to new tasks or incorporate additional context information, as it would require modifying the hand-crafted design, which is very costly in terms of time and effort as this process cannot be automated [18, 38, 42]. In addition, although several works emphasize the importance of taking into account context information not only to solve the tasks presented to the dialog system by the user, but also to enhance the system performance in the communication task, this information is not usually considered when designing a dialog model [22, 49].

To facilitate this general-purpose behavior and reduce the effort required for both the implementation of a new system and the adaptation of systems to include a new task, we propose a statistical methodology for dialog management in which the dialog model is automatically learned from a dialog corpus. This way, the adaptation of the dialog system only supposes the acquisition of a dialog corpus for the new task, for which an automatic dialog simulation technique is proposed in the paper. In addition, the statistical dialog model is enriched with context information used to adapt the interaction and provide personalized, context-aware services through a natural language conversation. To this end, our dialog manager is implemented using a classifier based on neural networks, which takes the previous dialog history and the context information into account to carry out the selection of the next system action.

In this paper, we also provide a complete implementation of our architecture in a railway information system. This system is presented as a reference to describe how our proposal can be applied for the complete design of context-aware systems that use spontaneous speech to access the different web services. The architecture facilitates the inclusion of context information to provide a more natural and adapted service to each user, and also makes possible an easy adaptation to new domains. In addition, a set of new measures has been defined for an in-depth evaluation of this system and to assess the influence of context information in the quality of the acquired dialogs. The results of this evaluation show that context information not only allows a higher effectiveness in the provision of web services, but also increases their quality.

The remainder of the paper is organized as follows. Section 2 describes related research in the development of context-aware systems and the design of personalized dialog systems. Section 3 describes the main characteristics of our architecture for providing context-aware adaptable services using speech-based interfaces. Section 4 describes our approach to manage context information. Section 5 describes the methodology and measures employed to evaluate our proposal. Section 6 shows a practical implementation of our architecture to generate a context-aware railway information system and the results of its evaluation using these measures. Finally, our conclusions are presented.

2 Related work

The use of mobile devices, web services, new communication channels and pervasive environments is the basis of many important initiatives and projects like the Future Internet European Initiative.[1] However, some usability problems limit the usage of mobile communication devices for the end-user, mainly the limited screen size and input facilities of devices such as smartphones and PDAs. Voice-based interfaces work seamlessly with small devices, and allow users to easily invoke local applications or access remote information.

The adaptation capabilities of these interfaces are frequently restricted to static choices made by the users. However, adaptation can play a much more relevant role in speech applications. For example, users have diverse ways of communication. Novice users and experienced users may want the interface to behave completely differently, such as maintaining more guided vs. more flexible dialogs. As stated in [12, 49], processing context is not only useful to adapt the systems' behavior, but also to cope with the ambiguities derived from the use of natural language. For instance, context information can be used to resolve anaphoric references depending on the context of the dialog or the user location. The performance of a dialog system also depends highly on

[1] http://www.future-internet.eu/.

the environmental conditions, such as, for example, whether there are people speaking near the system or the noise generated by other devices.

Thus, context awareness is fundamental for building usable interfaces to web services. Additionally, it should also be employed to adapt the services provided to the user and the device. In the literature, there are several approaches developing mobile and context aware systems such as platforms, frameworks and applications for offering context-aware services. However, there is a lack for an integrated approach which combines the benefits of the main state-of-the-art approaches. Additionally, interfaces are usually conceived separately, usually following the GUI metaphor. In our proposal, we merge context-awareness with oral interfaces in order to obtain fully accessible and personalized web services and information in hand-held devices. This is one of the main features introduced in our architecture due to the reduced number of context-aware speech interfaces that can be found in the literature and its application to very specific domains. As described in the introduction, we propose a statistical dialog management technique to automatically learn the dialog strategy and facilitate the adaptation of the developed systems to new tasks. In order to do so, we present a novel architecture in which we have addressed the issues summarized in Table 1.

According to Dey and Abowd [8], "*any information that can be used to characterize the situation of an entity (...) relevant to the interaction between a user and an application, including the user and the applications themselves*" can be considered context, thus the first issue to support context-awareness is to study which information is relevant to provide adapted web services. Kang et al. [19] differentiate two types of context: *internal* and *external*. The former describes the user state (e.g. communication context and emotional state), whereas the latter refers to the environment state (e.g. location and temporal context).

Most of the studies in the literature focus only on external context. One of the most popular is location information. For example, the Akogrimo project [37] aims at supporting mobile users to access data, knowledge, and computational services on the Grid. It only concentrates on context that is related to situations of mobile users, such as user presence and location, and environmental information. Similarly, SMAUG [34] is a multi-agent context-aware system that allows tutors and pupils of a university to fully manage their activities. This system offers its users context-aware information from their environment and also provides a service that physically locates every user in the system. Also AmbieAgents [23] is an agent-based infrastructure for context-based information delivery for mobile users.

However, external and internal context are intimately related, as it happens in representative examples like service context and proactive systems [5, 39, 52]. Some systems combine external context with a static representation of internal context, such as considering specific age intervals for their users. For example, Afsarmanesh et al. [1] present an application for supporting virtual elderly assistance communities within the framework of the TeleCARE project. One of the most important contributions of our work is to combine both internal and external context information meaningfully, given that it is essential to provide a useful personalization of the web services and is of great interest to optimize the spoken interface.

With respect to context representation and modeling, a number of methods have been proposed in literature, from the simple key-value method (in which a variable contains the actual context), to tagged encoding approaches (which use context profiles to enable modeling and processing context recursively, and to employ efficient context retrieval algorithms), and object oriented models (which have the benefits of encapsulation and reusability). Along with the formalism employed, there are different languages which might be used to represent context, for example UML, XML, RDF, and OWL are widely used and are considered open and interoperable. In existing context-aware systems, XML is already used widely for modeling and implementing context information. For example, the Anyserver platform [16] utilizes various types of context information encoded in XML, such as device information, networks, and application type. In the Omnipresent context-aware location-based system [3], context information is modeled based on OWL, while Prezerakos et al. [41] use UML to model context and web services.

In our proposal, XML files are used as a language to represent the context information. We combine aspects from the tagged encoding approach with the *dialog acts* (DA) formalism [51], which is employed to represent the information of the user interaction captured by the sensors in the environment. This way, we employ the same semantic representation for the user and system utterances (e.g. question, answer, response, etc.) in the conversational interface as well as the representation of internal and external context, thus building a unique structure of the complete "meaning" of the user queries.

Another issue is how context information is provided by sensors. Contextual information is usually measured by hardware or software-based sensors (such as GPS and monitoring programs), or provided by the users. Typically, sensors rely on low level communication protocols to send the collected context information or they are tightly coupled within their context-aware systems. Since sensing techniques are well developed, existing sensors utilize these techniques through instrumentation or polling mechanisms, and extend their capability by acquiring context information from existing systems. As it will be described in Sect. 3, we use a commercial platform that captures external context, then this information, together with the user profile, is used

Table 1 Main issues covered by our proposal

Alternatives in the literature	Our proposal
Type of systems	
Application domain: Domain specific [3, 16], Generic [4, 7, 20, 37, 53, 55] System type: Framework/Toolkits [4, 7, 20, 37, 53], Application [3] Mobility support [4, 16, 37, 53, 55] Level of Web service implementation: Full [3, 4, 7, 20, 53, 55], Partially [16, 37]	One of the main points of our proposal is to provide a more natural interaction with users by means of spoken dialog systems. We propose an architecture valid for slot-filling dialog tasks. Instead of using rule-based models to define the dialog model, we have developed our own statistical methodology for dialog management. This methodology is used for the automatically learning the dialog model and facilitates an easy adaptation to a new task. Mobility support is guaranteed by the use of speech as the communication modality, which allows to easily access the application by means of mobile devices. Web services are used to obtain the information that is task-dependent in order to provide an answer to the user
Context information	
External: location, device, network, calendar, device, network [4, 7, 16, 27, 37, 53] Internal: user peculiarities, preferences, needs [4, 7, 16, 53]	We merge both types of context in order to tailor the web services and the oral interface. External context is provided by means of the Appear IQ platform and internal context is considered by means of user profiles
Context representation	
Key-value, tagged encoding, object-oriented	We have developed an hybrid approach between key-value representation and dialog act semantics, in which the information which is vital for the oral interface is integrated into the dialog act and the rest of the contextual information is represented using XML
Context modeling	
UML [50, 53, 55], XML [4, 16, 20, 27], OWL/Ontology [7, 37, 53], tool specific [37]	We use XML files to transmit context information using the OASIS web services context specification for sharing and passing context information between services and clients
Context retrieval and reasoning	
Retrieval: Device, sensors, explicitly ask users Reasoning: Reasoning capability support [7, 53], Semantic-based reasoning [7, 53], Specific reasoning	By means of our architecture all the information is gathered in a way which is transparent to the user (using previous dialogs)
Context Sensor Techniques	
Mode: Automatic [7, 27, 37, 53, 55], Manual [7, 55] Sensing Techniques: Instrumentation [7, 27, 55], Polling [16, 37, 53, 55] Sensor interface: Web service [53, 55], Specific [7, 16, 37] Data retrieval and publishing: Query, Subscription [37], Push-Only [37]	By means of our architecture all the information is gathered in a way which is transparent to the user (by means of the Appear IQ platform to acquire information related to external context and using the previous dialogs to automatically annotate user preferences and update the user profiles)
Context storage	
Storage Model: Centralized [37, 53], Distributed [4, 53, 55] Storage Databases: Relational Databases [4, 37], XML [55], RDF/OWL [53], Others [37] Access Interface: Web service [4, 53, 55], Others Request Specification: SQL [4, 37], XPath/XQuery [55], SPARQL [53], Specific	We propose the use of relational databases for context storage
Context distribution	
Overlay network distribution: Centralized [4, 37, 53, 55], P2P [55] Direct transport distribution: SOAP extension [20, 53], Web proxy [27] Access mechanism: Query through Web service [4, 53, 55], Subscription through Web services Notification/Pubscribe [37], Subscription through WS call back [37, 55], Subscription through specific protocol [55]	Context information is transferred using SOAP messages according to the OASIS web services context specification. The SOAP message header to transfer the context information modeled following the OASIS specifications, and the management of the information is carried out in the dialog manager of the proposed dialog system

Table 1 (*Continued*)

Alternatives in the literature	Our proposal
Context Adaptation Techniques	
Adaptation purpose: Content adaptation [16, 37], Communication adaptation [27], Service and task selection [37, 53, 55], Information protection [56, 57], Others Adaptation specification: Modeling [27], Runtime [53] Adaptation layer: Unspecified, Middleware [16, 53, 56, 57], Application/Service [37, 55]	Context information is used for content, communication and service adaptation. In this way, context information is used to adapt content resulting from a request and to return the content in a form compatible with to the context of the requester, optimize the communication, and select the most suitable system action to perform actions given a dialog situation

in our architecture to provide web services that are adapted to the user location, geographical context, communication context preferences and needs.

Reasoning techniques can also be employed to infer new types of context information. When the context information is described by OWL and ontologies, typically reasoning techniques will follow a semantic approach, such as for example in SPARQL.[2] In our case, there is a module fully integrated into the dialog system architecture that is able to reason about the whole semantic of the application, that is, not only on the web services but also on the contextual information gathered from the user and the device, which also includes the semantic representation of each of the user interventions. As in other speech interfaces dealing with web contents, semantic knowledge is modeled in our architecture using frames [29]. A frame is a structure for representing a concept or situation which has several associated attributes (slots) and values [10]. In the semantic representation defined for our architecture, one or more concepts represent the intention of the utterance, and a sequence of attribute-value pairs contains the information about the actual values provided by the user.

Once the information is retrieved and analyzed, it must be conveniently stored. Relational databases are widely used to store context information in context-aware systems out of the web services domain [17, 32], even in the case of XML or ontology-based systems [21].

Regarding context distribution techniques, the main ones are direct transport protocols, techniques using overlay network protocols, and supporting access mechanisms. When using direct transport protocol, context information is transferred between two parties using SOAP messages (Simple Object Access Protocol).[3] OASIS proposed a web services context specification[4] that describes a mechanism and service structure for sharing and passing context information between services and clients. A context manager is provided to manage context sources and context information is represented as relations and defined by using context collectors and context information can be queried. We use the SOAP message header to transfer the context information modeled following the OASIS specifications, and the management of the information is carried out in the dialog manager of the proposed dialog system, as will be described in the next section.

Finally, adaptation based on context information is typically application-specific. Many context-aware middlewares allow the developer to specify actions that should be performed in particular contexts. In most cases, the middleware might just support the management and exchange of contextual information. Although the reasons for performing context adaptation are diverse, the main purposes are related to service selection and task adaptation (context information is used to select the most suitable service and task to perform actions given a situation), security and privacy control (context information is used to support adaptive control in security and privacy management), communication adaptation (context information is used to select communication protocols and optimize the communication), and content adaptation (context information is used to adapt content resulting from a request and to return system responses in a form suitable to the context of the requester). We propose the use of contextual information for both tasks: communication and content adaptation. This way, in our architecture, context information is used to adapt content resulting from a request and to return the system responses in a form suitable for the context of the requester, optimize the communication, and select the most suitable system action in each dialog situation.

3 Our context-aware architecture to provide web services

As stated in the introduction, we have developed a context-aware architecture that facilitates developing, discovering, providing and accessing adaptable web services through personalized speech-based interactions with a context aware

[2] http://www.w3.org/TR/rdf-sparql-query/.

[3] http://www.w3.org/TR/soap.

[4] http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.pdf.

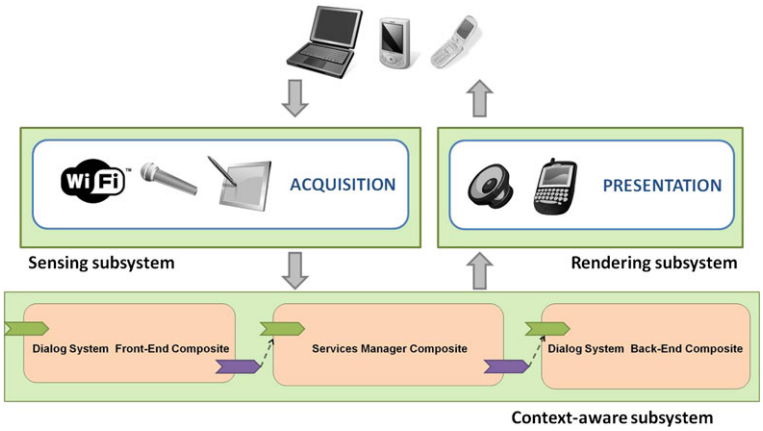


Fig. 1 Schema of the different subsystems in the architecture

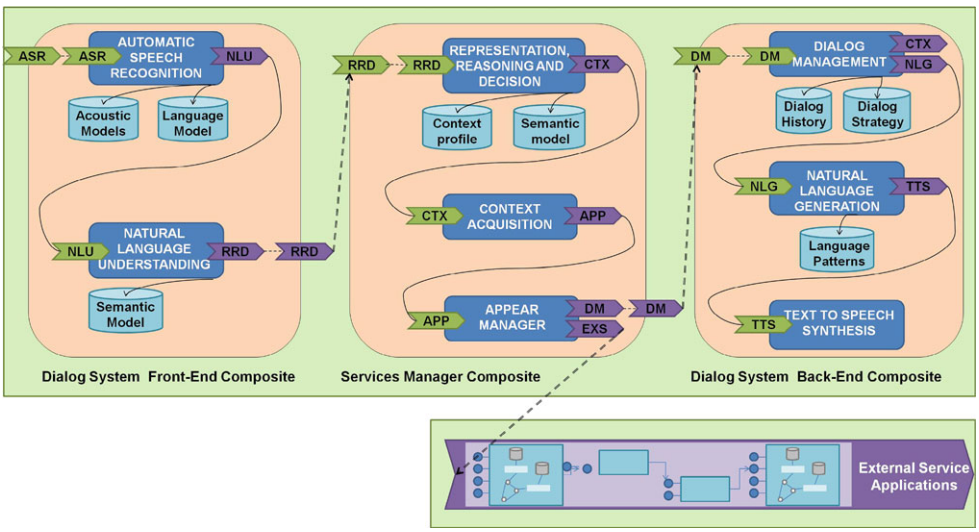


Fig. 2 Architecture of the context-aware subsystem

dialog system. As can be observed in Fig. 1, our architecture consists of three subsystems: (i) sensing subsystem; (ii) rending subsystem, and (iii) context-aware subsystem.

The proposed architecture was developed as a series of services which are assembled together in composite applications, this way it is possible to inter-connect components while preserving loose coupling. With this Service Oriented Architecture [33], services achieve integration through the exchange of messages. In the sensing subsystem, there is a set of sensors which capture the users' activity (position, voice, environmental noise). Location information is acquired by the web services manager layer which converts coordinates into zones, as well as the static user profile like: name, role, IP/MAC address, date/time. The rest of the profile information is then processed by the context-aware subsystem, which includes a spoken interface that converts spoken speech inputs into text, analyses the user requests during the interaction, decides which are the appropriate web services to be provided, and synthesizes a spoken output which is rendered to the user. Context information is used and updated throughout the entire cycle as will be explained in Sect. 4.

Figure 2 shows the architecture proposed for the context-aware subsystem, following the Service Component Architecture (SCA) specification,[5] which allows building coarse-grained components as assemblies of fine-grained ones. In our case, there are three such assemblies: the dialog system front and back end and the services management composites. Additionally, there are also external services to the system, which are applications provided by third-parties.

3.1 Web services manager

We have implemented the Web Services Manager using the Appear IQ Platform (AIQ).[6] Appear IQ is a solution that scales quickly to meet the changing needs of high-demanding applications. The platform features a distributed modular architecture that supports multiple network configurations and can be deployed as a distributed system. It

[5]http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications.

[6]http://www.appearnetworks.com.

consists of two main modules: the Appear Context Engine (ACE) and the Appear Client (AC).

The ACE implements a rules engine, where the domain-specific rules that are defined determine what should be available to whom, and where and when it should be available. These rules are fired by a context-awareness runtime environment, which gathers all known context information about a device and produces a context profile for that device. In our system, the context parameters defined for our application include physical location, date/time, device type, network IP address, and user language.

The ACE is installed in a server, while the ACs are included in the users' devices. Thus, the architecture is distributed, in our case communication is carried out through several proxies. The network management is carried out by the Appear Context Proxy (ACP), which eliminates unnecessary traffic, thus ensuring bandwidth for new user requests, and keeps a cache of active user sessions and most accessed services. When a wireless device enters the network, it immediately establishes the connection with a local proxy, which evaluates the position of the client device and initiates a remote connection with the server. Once the client is in contact with the server, it provides the set of applications the user can access depending on his physical position.

Therefore, the functionality of Appear depends mainly on the Appear Context Engine. The ACE is divided into three modules that collaborate to implement a dynamic management system that allows the administrator to control the capability of each device once they are connected to the wireless network. These modules are: the Device Management Module, the Push Provisioning Module, and the Synchronization Module.

The Push or Provisioning Module manages the automatic distribution of applications and content to hand-held devices. It pushes services on these devices using client-side intelligence when it is necessary to install, configure and delete user services.

The Device Management Module provides management tools to deploy control and maintain the set of mobile devices. The context-aware actions on the client side are: (i) The configuration of the different elements that describe the specific steps to be taken by the client. They are initially installed together with the client and then updated using the Synchronization Module; (ii) Context conditions: an associated condition to the current context is applied to determine if the action is applicable. It is made by the rule-engine of the client: (iii) Mirroring: it is a mechanism by which the client monitors file updates in a device. These updates are replicated to a secondary device as a storage card or a remote host using FTP/HTTP.

The Synchronization Module manages the exchange of files between corporate systems and mobile hand-held devices. The Device Management is continuously provided with updated versions of the configuration files. There are three steps in the Synchronization Module: (i) The Synchronization Module compiles contextual data to gain an understanding of the user's informational needs; (ii) Available data is filtered against the user's context to determine what information should be the most relevant; (iii) The Module automates synchronization, detecting files that have changed and synchronizing them. It is a dynamic synchronization of the profile based on User and Role, Location, Time, Device Status and Connectivity.

3.2 Dialog system

As stated in the introduction, a dialog system can be understood as an automatic system capable of emulating human conversation, with the aim that the system meets a certain functionality (usually providing information or performing a certain service). Discarding the simplest case, these applications require a sequence of interactions between the user and the system to achieve their final purpose. Therefore, the user's goal is gradually reached during several dialog turns.

Thus, it is necessary to endow the system with the abilities to reference information that has appeared previously during the dialog, take the initiative to recover the dialog after a failure, request information that is necessary to fulfill the objective, or require clarification if it is not confident about the information provided by the user.

During the communication process, the system initially generates a message to welcome and inform the user about the features and functionalities of the system. Then, the system must perform a basic set of actions that are cyclically repeated after each user utterance: recognize the sequence of words mentioned by the user; extract the meaning of these words (i.e. understand the information that is useful for the system domain), perform web services and database access operations to extract the information required by the user, and adapt the interaction to the context features described above; decide what action or actions should be performed after each user request; and generate a webpage and play a spoken message to provide a response to the user.

Given the number of operations that must be carried out, the scheme used for the development of these systems usually includes several generic modules that deal with multiple knowledge sources and that must cooperate to satisfy the user's requirements. With this premise, a dialog system can be described in terms of the following modules, which we have implemented as services. The *Automatic Speech Recognition module* (ASR) transforms the user utterance into the most probable sequence of words. The *Natural Language Understanding module* (NLU) provides a semantic representation of the meaning of the sequence of words generated by the ASR module. The *Dialog Manager* determines the next action to be taken by the system following a dialog

strategy. The traditional approach to do this is to handcraft a series of rules which determine such behavior. However, this design method is very time consuming and has the ever-increasing problem of dialog complexity. As an alternative, statistical models can be trained from real dialogs, modeling the variability in user behaviors.

Our dialog manager follows this paradigm and 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 [13]. This partition, which is defined taking into account the data supplied by the user throughout the dialog, makes the estimation of a statistical model from the training data manageable. In order to control the interactions integrating context information, our dialog manager represents the dialogs as sequences of pairs (A_i, U_i), where A_i is the output of the dialog system (the system answer) at time i, expressed in terms of dialog acts; and U_i is the semantic representation of the user input (the result of the understanding process of the user input) at time i. This way, each dialog is represented by:

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

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:

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

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

As the number of all possible sequences of states is very large, we define a data structure in order to establish a partition in the space of sequences of states (i.e., in the history of the dialog preceding time i). This data structure, that we call *Dialog Register* (*DR*), contains the information provided by the user throughout the dialog and the context information that is provided by the Context Manager.

For the dialog manager to determine the next system response, we have assumed that the exact values of the attributes are not significant. They are important for accessing the web service and for constructing the output sentences of the system. However, the only information necessary to determine the next system action is the presence or absence of concepts and attributes. Therefore, the information we used from the *DR* is a codification of this data in terms of three values, $\{0, 1, 2\}$, for each field in the *DR* according to the following criteria:

- 0: The concept is unknown, or the value of the attribute has not yet been provided by the user.
- 1: The concept or attribute is known with a confidence score that is higher than a certain threshold (between 0 and 1). The confidence score is calculated during the recognition and understanding processes and can be increased by means of confirmation turns.
- 2: The concept or attribute is activated with a confidence score that is lower than the given threshold.

After applying the above considerations and establishing the equivalence relation in the histories of dialogs, the selection of the best A_i is computed as:

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

This maximization can be solved by means of a classification process that takes into account the input pair (DR_{i-1}, S_{i-1}) and decides which is the best system action. We propose the use of multilayer perceptrons (MLP) [43] to define the classification function, where the input layer received the current situation of the dialog, which is represented by the term (DR_{i-1}, S_{i-1}). The values of the output layer can be viewed as the a posteriori probability of selecting each one of the system dialog acts (i.e., system prompts) defined for a specific task.

Once the dialog manager has selected which is the next system action, the *Natural Language Generator module* (NLG) transforms this action into an answer in natural language. The methodology that we have selected to incorporate context information in the natural language generation module is based on the use of a set of feature-based templates associated to the different system actions, in which the names of the different attributes are reflected. These names are replaced by the values obtained from the dialog register and the user profile to generate an answer for the user. Figure 3 shows different templates that have been defined for the case of a railway information system providing timetables to travel to a specific city on a given date. Finally, a *Text to Speech Synthesizer* (TTS) generates the audio signal transmitted to the user.

4 Our approach to context-awareness

To deal with context information and personalize web services, we have incorporated a new module in the architecture of a dialog system. This module, that we have called *Context Manager*, loads, updates and manages context information associated with each one of the users. Once the Context Manager receives the user's identification, it reads the context information from the user profile. The information stored in this data structure can be classified into three different groups:

- General user information. User's name and machine identifier, gender, preferred language, pathologies or speech disorders, age, current location, date, and time.

Fig. 3 Examples of the set of templates defined to take into account context information in the NLG module

System Dialog Act: (Answer:Departure-Hour)
<*User-Name*>, the trains to <*Destination-City*> on <*Date*>
<*User-Name*>, I inform you about the trains to travel from <*Origin-Station*> to <*Destination-City*> on <*Date*>
I inform you about the trains to travel from <*Origin-Station*> to <*Destination-City*> on <*Date*> at <*Time*>
I inform you about the trains to travel to <*Destination-City*> on <*Date*>

- Skill level: This level is estimated by taking into account variables like the number of previous sessions, dialogs and dialog turns, their durations, time that was necessary to access a specific web service, the date of the last interaction with the system, etc. A low, medium, high or expert level is assigned using these measures.
- Usage statistics: This set stores the counts of each action over the system that a user performs, and a mark of user clearance for each possible action. Users' preferences are automatically evaluated considering the user's most required services during the previous dialogs, date and hour of the previous interactions, most frequent objective, and preferred output modality.

This information is used by the push/provisioning, synchronization and device management context-aware modules in the ACE, that compare the current state of the user's device with the context-aware services set out for the device and order the adaptive client to update the corresponding services on the device. Context information is used throughout the entire life-cycle of the service: selection based on context, filtering of individual services and enhancement of services at boot or runtime. Then, the device receives an access to the different services that are available in the network taking into account the information stored in the ACE and including the context information. These services are provided in our architecture by means of spoken dialog systems. The dialog system is coupled in synchronous mode to a web service. Context information is received by the Context Manager in the dialog system included at the beginning of the interaction in order to adapt the behavior of the different modules in the dialog system taking this information into account.

The dialog system receives user utterances from any telephone, which are transmitted to the automatic speech recognition system and semantically analyzed by the natural language understanding module to obtain a frame-based representation including confidence scores. Then, the dialog manager takes into account not only the current user utterance but also the complete dialog history to decide the next system action. This can lead the system to require additional information to the user, to confirm or ask again for information already provided by the user, or to answer once the complete information that is required from the user has been retrieved. When an answer is generated, the information is sent to the Web Query Manager, which contacts the web service using a SOAP attachment. The web service then provides the result of the query. This is achieved in the specific application described in this paper, using a WS-Attachments protocol, proposed by Microsoft and called DIME (Direct Internet Message Encapsulation), instead of directly passing a URL to the web Service. This mechanism allows direct sending of an attachment (picture, text, sound, etc.) to a web service. We used C# language and Web Services Enhancements (WSE), within Visual Studio, for our developments. WSE is a .NET class library for building web services using the latest Web services protocols including WS-Security, WS-SecureConversation, WS-Trust, WS-Policy, WS-SecurityPolicy, WS-Addressing, and WS-Attachments.

The answer of the web service is transmitted, using again SOAP attachment, to the language generator module. This module takes into account the information received from both the dialog manager and the web service to generate a sentence in natural language to inform the user. This sentence is translated from text to speech by the Speech Synthesizer to inform the user. At the end of the interaction, the user profile is updated taking into account the information acquired during the dialog. The transmission of the context between modules is carried out by sending XML packages based on the OASIS Web Services Context Specification.

The complete process can be summarized as follows (Fig. 4):

1. *Device Detection*: Once the user is detected in the network by the ACP, this module evaluates the position of the client device and initiates a remote connection with the ACE. The ACE gathers all known information about the device, the user and his context, including physical location, date/time, device type, user roles, network IP address range, user locale and other customized context providers, e.g. temperature, available battery, etc. The context engine derives a description of the services that should be available on the device and passes it to the modules deployed on top of the ACE.

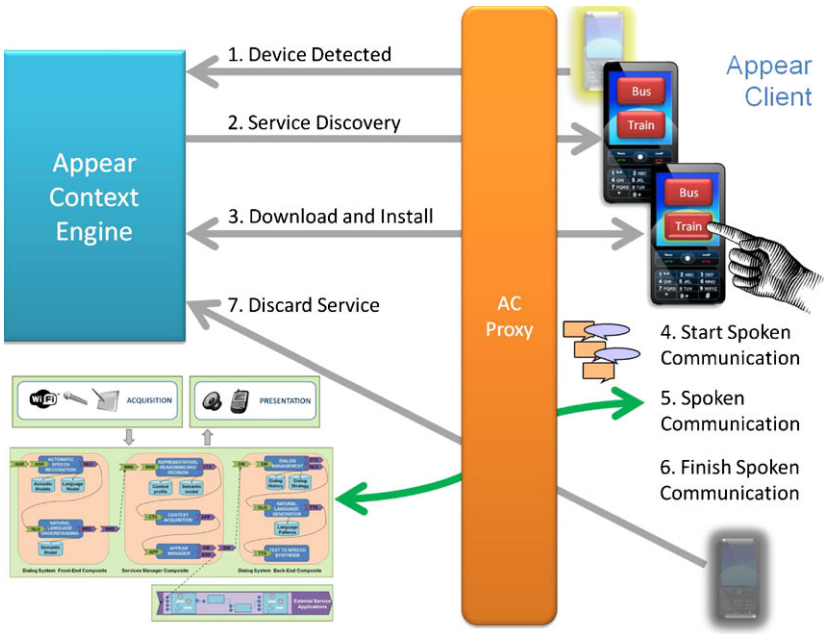


Fig. 4 Process followed in the proposed architecture to provide a context-aware web service

2. *Service Discovery*: the Context Profile is generated by the Context Engine and then transmitted to the client. This transmission is shown as icons on the hand-held device of the user interface. Then, the client decides which service to pull by clicking on the corresponding icon.
3. *Download and install*: the necessary resources right after service discovery. Once the resource is available on the device, the installation proceeds.
4. *Start spoken communication*: The user selects the spoken communication interface to receive the information. Immediately, the ACE sends an XML package to the Context Manager in the dialog system informing about its identification and current location. Using such information, the Context Manager selects the profile of the recognized user and communicates this information to the different modules of the dialog manager. Each module uses this knowledge to load its specific information and models.
5. *Spoken communication*: The user starts the interaction with the dialog system. Throughout the interaction, each module can update the active user profile. Depending on the information that is modified, the Context Manager sends the value of the new features only to the modules in the dialog system that require such information. Each module in the dialog system is able to adapt its specific models and characteristics by taking into account the contextual information that is provided by the Context Manager.
6. *Finish spoken communication*: At the end of the interaction, the user profile is updated using the information acquired during the last dialog session.
7. *Discard Service*: when a user leaves the network or if a context condition has changed for a service.

5 Evaluation methodology and measures

As the study and development of dialog systems becomes more complex, the task of defining n ew p rocedures and measures that will be unanimously accepted by the scientific c ommunity f or t he e valuation o f t his k ind o f systems presents more difficulties. A technique that has attracted increasing interest in the last decade is based on the automatic generation of dialogs between the dialog manager and an additional module, called the *user simulator*, which simulates user interactions with the dialog system [26, 45]. This technique makes possible to generate a large number of dialogs in a very simple way. Therefore, it reduces the time and effort that would be needed for the detailed evaluation of the quality of the services provided by a dialog system.

We have developed a methodology for the automatic generation of dialogs, which can be used for learning and evaluating statistical dialog models [14]. Our methodology is based on the interaction of a user simulator and a dialog manager simulator. Both modules 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, all the system answers are defined with the same probability. When a successful dialog is simulated, the probabilities of the answers selected by the dialog manager during that dialog are incremented before beginning a new simulation.

The user simulation simulates the user intention, that is, the simulator provides concepts and attributes that represent the intention of the user utterance. Therefore, the user simulator carries out the functions of the ASR and NLU modules. The semantics selected for the dialog manager is represented through the set of possible system answers defined

for a specific task. The selection of the possible user answers is carried out using the semantics defined for the NLU module. An error simulator module has also been designed to perform error generation and the addition of confidence measures. This information modifies the frames generated by the user simulator and also incorporates confidence measures for the different concepts and attributes. The number of errors that are introduced can be modified to adapt the error simulator module to the operation of any ASR and NLU modules.

The model employed for introducing errors and confidence scores is inspired in the one presented in [47]. Both processes are carried out separately following the noisy communication channel metaphor by means of a generative probabilistic model $P(c, a_u|\tilde{a}_u)$, where a_u is the actual incoming user dialog act, $\tilde{a}_u$ is the recognized hypothesis, and c is the confidence score associated with this hypothesis.

On the one hand, the probability $P(\tilde{a}_u|a_u)$ 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.

$$P(\tilde{a}_u|a_u) = \sum_{\tilde{w}_u} P(a_u|\tilde{w}_u) \sum_{w_u} P(\tilde{w}_u|w_u) P(w_u|a_u)$$

On the other hand, the generation of confidence scores is carried out by approximating $P(c|\tilde{a}_u, a_u)$ assuming that there are two distributions for c. These two distributions are defined manually, generating confidence scores for correct and incorrect hypotheses. These definitions are based on a sampling over the distributions found in the training data corresponding to our initial corpus.

$$P(c|a_w, \tilde{a}_u) = \begin{cases} P_{\text{corr}}(c) & \text{if } \tilde{a}_u = a_u \\ P_{\text{incorr}}(c) & \text{if } \tilde{a}_u \neq a_u \end{cases}$$

The dialog manager considers that the dialog is unsuccessful and decides to abort it when one of the following conditions take place:

- The dialog exceeds a maximum number of system turns, defined taking into account the requirements of the task.
- The answer selected by the dialog manager corresponds to a query not required by the user simulator.
- The web service informs about an error because the user simulator has not provided the information required.
- The answer generator provides a warning when the selected answer involves the use of a data not contained in the *DR*, that is, not provided by the user simulator.

A user request for closing the dialog is selected once the system has provided the information defined in the objective(s) of the dialog. The dialogs that fulfill this condition before the maximum number of turns are considered successful.

5.1 Measures defined for the evaluation

It is very difficult to define new procedures and measures that will be unanimously accepted by the scientific community for the evaluation of voice-based systems. In fact, this field can be considered to be in an initial phase of development. In [48] and [45], a set of statistical measures to evaluate the quality of a simulated corpus is proposed. Three dimensions are defined: high-level features (dialog and turn lengths), dialog style (speech-act frequency; proportion of goal-directed actions, grounding, formalities, and unrecognized actions; proportion of information provided, reprovided, requested and rerequested), and dialog efficiency (goal completion rates and times). The simulation presented in [2, 44, 46] is evaluated by testing the similarity between real and simulated data by means of statistical measures (dialog length, task completion rate and dialog performance).

We have adapted the previously described measures considering the information that is available in the definition of the dialog system and context information included in the user profiles. Our proposed measures can be classified into three groups: task success/efficiency measures, high-level dialog features, and dialog style/cooperativeness measures.

- Task success/efficiency measures: These measures study the goal achievement rates and goal completion times for the services provided by the system.
- High-level dialog features: These features evaluate the duration of the dialogs, how much information is transmitted in individual turns, and how active the dialog participants are.
- Dialog style/cooperativeness measures: These measures analyze the frequency of different speech acts and reflect the proportion of actions that is goal-directed (i.e. not indexed in dialog formalities).

By means of task success/efficiency measures in our evaluation, we investigate the success rate and efficiency of the services provided by the system. We are particularly interested in goal achievement rates and goal completion times. The dialogs were only considered successful if they fulfilled the complete list of objectives that had been previously defined for it. We have also evaluated the influence in these measures of the different context information sources defined in a system.

Six high-level dialog features have been defined for the evaluation of the dialogs: the average number of turns per dialog, the percentage of different dialogs without considering the attribute values, the number of repetitions of the most

seen dialog, the number of turns of the most seen dialog, the number of turns of the shortest dialog, and the number of turns of the longest dialog. Using these measures, we tried to evaluate the success of the simulated dialogs as well as its efficiency and variability with regard to the different services.

For dialog style features, we define and count a set of system/user 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. Finally, we have measured the proportion of goal-directed actions (request and provide information) versus the grounding actions (confirmations) and rest of actions.

The previous measures evaluate the overall quality of the acquired dialogs and provided services as a whole. In addition, we have carried out a specific evaluation of the operation of our dialog management methodology. From our previous work on statistical dialog management [13], we propose four measures to evaluate the performance of the dialog manager. The first measure, which we call *% unseen*, makes reference to the percentage of unseen situations, i.e., the dialog situations that are present in the test partition but are not present in the corpus used for learning the dialog model. 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. These three measures are: (i) *% exact*: the percentage of answers provided by the dialog manager that are equal to reference answer in the corresponding turn of the training corpus; (ii) *% correct*: the percentage of answers provided by the dialog manager that are coherent with the current state of the dialog although they are not the same as the reference answer; (iii) *% error*: the percentage of answers provided by the dialog manager that would cause the failure of the dialog.

6 Implementation and evaluation of a context-aware railway information system

We have applied our context aware methodology to develop and evaluate an adaptive system in a railway information domain. The system provides information in natural language about train services, types, schedules, and fares [15]. The information offered to inform the user is extracted from a web service developed with the information available in the web of the Spanish National Railways.

We defined four concepts to represent the different queries that the user can perform (*Hour*, *Price*, *Train-Type*, *Trip-Time*, and *Services*), three task-independent concepts (*Affirmation*, *Negation*, and *Not-Understood*). The attributes needed by the system to answer the different user queries are *Origin*, *Destination*, *Departure-Date*, *Arrival-Date*, *Ticket-Class*, *Departure-Hour*, *Arrival-Hour*, *Train-Type*, *Order-Number*, and *Services-List*. An example of the semantic interpretation of a sentence is shown in Fig. 5.

The labeling of the system turns is similar to the labeling defined for the user turns. To do so, 30 task-dependent concepts were defined:

- Task-independent concepts (*Acceptance*, *Rejection*, *Not-Understood*, *New-Query*, *Waiting*, *Opening*, and *Closing*).
- Concepts used to inform the user about the result of a specific query (*Hour*, *Price*, *Train-Type*, *Trip-Time*, and *Services*).
- Concepts defined to require the user the attributes that are necessary for a specific query (*Origin*, *Destination*, *Departure-Date*, *Arrival-Date*, *Ticket-Class*, *Departure-Hour*, *Arrival-Hour*, *Train-Type*, *Order-Number*, and *Services-List*).
- Concepts used for the confirmation of the previous concepts and attributes.

A total of 51 system actions (different dialog acts) were defined taking into account the information that the system provides, asks or confirms. The *DR* defined for the task is a sequence of 19 fields, corresponding to the five concepts (*Hour*, *Price*, *Train-Type*, *Trip-Time*, *Services*), ten attributes defined for the task (*Origin*, *Destination*, *Departure-Date*, *Arrival-Date*, *Departure-Hour*, *Arrival-Hour*, *Ticket-Class*, *Train-Type*, *Order-Number*, *Services-List*) defined for the task, three task-independent concepts that users can provide (*Acceptance*, *Rejection* and *Not-Understood*), and a reference to the user profile.

A set of 30 scenarios were manually defined to cover the different queries for which the system should be able to respond, including different user requirements and profiles. Two main types of scenarios were defined. Type S1 scenarios defined only one objective for the dialog; i.e., the user must obtain information about only one type of the possible queries to the system (e.g., to obtain timetable information from an origin city to a destination for a specific date). Type S2 scenarios defined two objectives for the dialog (e.g., to obtain timetables and prices for a specific origin, destination and date). An example of a S2 scenario is as follows:

```
User name: José García
Location: Atocha Station
Date and Time: 2009-05-01, 9:00am
Device: PDAQ 00-18-41-32-0B-59
Objective: Timetables and prices to Granada
```

Fig. 5 An example of the labeling of a user turn in the railway information system

Input sentence:
[SPANISH] *Sí, me gustaría conocer los precios para ir mañana de Valencia a Madrid.*
[ENGLISH] *Yes, I would like to know the prices for tomorrow evening, leaving from Valencia to Madrid.*

Semantic interpretation:
(*Affirmation*)
(*Price*)
Origin: Valencia
Destination: Madrid
Departure-Date: Tomorrow
Departure-Hour: Evening

For each scenario, once context information is received by the Context Manager, it loads the specific context profile characteristics. This information is then checked by the rest of the modules in the dialog system to personalize the provided service. In the previous example, the context information in the user profile is the following:

```
Name: José García
-------------------
Gender: Male | Age: 29 | Language: Spanish | Skill level:
High | Pathologies: None...
-------------------
Preferences: Timetables, Talgo Train, Business Class,
Atocha Station, 10:00am...
-------------------
Current_Location: Atocha Station | Platform zone
```

To evaluate our architecture, we have acquired a set of dialogs for each of the scenarios defined for the railway information system, including or not considering the Context Manager in our architecture or not (context-aware system and context-unaware system). The dialog simulation technique described in the previous section was used to acquire a total of 1000 successful dialogs for the task (500 dialogs using the context-aware system and 500 dialogs using the context-unaware system). To compare the corpora acquired, we computed the mean value for each corpus with respect to each of the evaluation measures described in the previous section. We then used two-tailed t-tests to compare the means across the two corpora as described in [2]. All differences reported as statistically significant have p-values less than 0.05 after Bonferroni corrections.

As an example of the difference between the resulting dialogs when the Context Manager and user profiles are or are not included, Fig. 6 shows a dialog for the railway information domain acquired without incorporating the Context Manager, and a dialog of the same scenario acquired using our approach. Turns starting with S refer to system turns, and turns with U refer to user turns. As it can be observed, the context-aware system shows a tendency of providing the required services with higher agility and using more natural answers than the context-unaware system.

Using the codification previously described for the information in the *DR*, when a dialog starts (in the greeting turn of the system) all the values in the dialog register are initialized to "0". The information provided by the Context Manager at the beginning of the dialog and the information provided by the users in each dialog turn is employed to update the previous *DR* and obtain the current one, as shown in Fig. 7. If there is information available about the user gender, usage statistics and skill level, it is incorporated to a user profile that is addressed from the dialog register, as was explained in Sect. 4. This information is used not only for the selection of the next system action by the dialog manager, but also to include this data for the generation of the system prompts provided by the natural language generator module (as Fig. 6 shows).

In the example described in the figure, the semantic interpretation of a user's utterance provided by the NLU module can be observed. Confidence scores, also provided by this module to determine the reliability, are shown between brackets. 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. The attribute *Origin* is in this case provided by the Context Manager. As the input to the MLP is 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 probabilities assigned by the MLP for the different prompts defined for the task are in this case 0.63 to confirm the departure date, 0.22 to ask again for this piece of information, 0.09 to ask for the ticket class type, 0.06 to ask for the type of train, and a 0.00 probability to the rest of 47 system prompts defined for the task. This way, the output selected for the MLP in this case is confirming the departure date. This process is repeated to predict the next system response after each user turn.

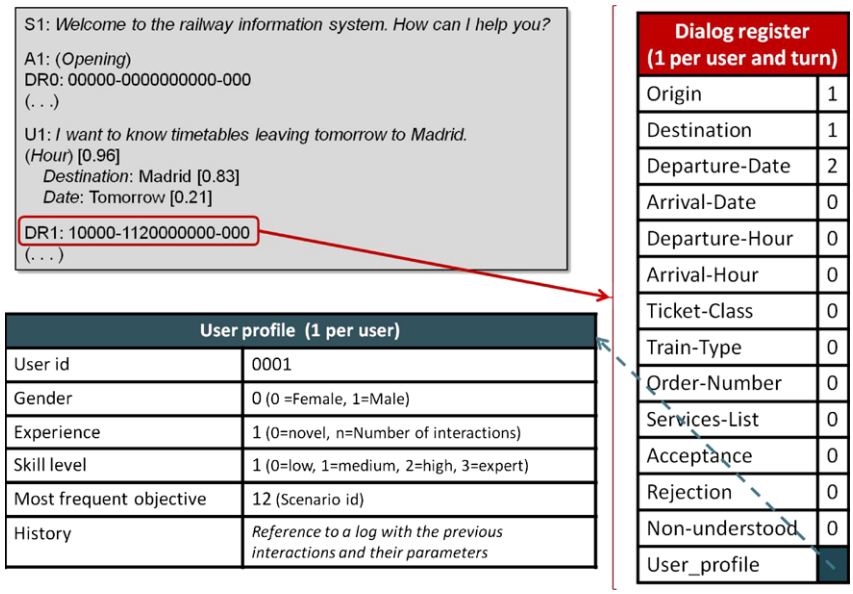
6.1 Evaluation results of the user simulator

First, we evaluated the user simulator that we developed. We were interested in assessing its efficiency and the dia-

Fig. 6 An example of a dialog for the railway domain using the context-unaware system (*above*) or the context-aware system (*below*)

S1: Welcome to the railway information system. How can I help you?	*No prompts personalization*
U1: I want to know timetables to Granada.	
S2: From which station do you want to leave?	*There is not information about the current location of the user*
U2: Atocha Station.	
S3: Please, tell me date.	*No date stored*
U3: Today.	
S4: There are five trains leaving from Atocha Station to Granada at 10:00 am, 11:30 am, 15:00 pm and 19:00 pm. Anything else?	*No information about the user preferences: timetables*
U4: Tell me the price of the first one.	
S5: Please, tell me the ticket class.	*No information about the user preferences: ticket class*
U5: Business class.	
S6: The price in business class is 35.5 euros. Anything else?	
U6: No, thank you.	
S7: Thank you for using the railway information system. Have a good trip!	
S1: Good morning José, I check trains information leaving from Atocha Station. Do you want timetables information?	*Prompt personalization including the user's name. The user location is known and his preference about consulting timetables*
U1: Yes, I want to know timetables to Granada.	
S2: The following two trains leave today at 10:00 am and 10:27 am from Atocha Station to Granada. Anything else?	*The system uses the current hour and the date to personalize the timetables service. In this case, there is not any information about other possible queries performed by the user in previous dialogs*
U2: Please, tell me the price of the first one.	
S3: The price in business class is 35.5 euros. Anything else?	*The system uses the ticket-class preferences included in the user profile*
U3: No, thank you.	
S4: Thank you José. Have a good trip!	*Prompt personalization including the user's name*

Fig. 7 Excerpt of a dialog with its correspondent user profile and dialog register for one of the turns



log success rate. Figure 8 shows the goal achievement rates and goal completion times for the different types of scenarios and systems (context-aware and context-unaware). The first important advantage of our proposal is a reduction in the number of dialogs that were necessary to simulate in order to obtain the 250 successful dialogs for the S1 and S2 scenarios in the context-aware and context-unaware systems. In other words, the percentage of dialogs in which the system successfully provides the complete list of services predefined for the corresponding scenario.

While only a 26.6 % of successful simulated dialogs is obtained for the S1 types using the context-unaware system, this percentage increases to 47.4 % for the same type of scenarios using the context-aware system. Regarding S2 scenarios, only a 12.1 % of successful dialogs are obtained using the context-unaware system. This percentage increases

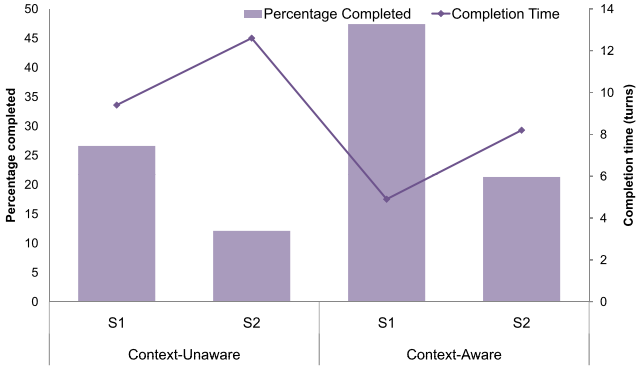


Fig. 8 Goal completion rates for the simulation process (percentage of successful dialogs with respect to the total amount of automatically generated dialogs) and completion times (in dialog turns)

to 21.3 % using the context-aware system. This is mainly due to the valuable context information that is directly provided to the system without the user's participation (e.g., current location and user's preferences). Given that this information is not provided by the user in most of the dialogs acquired with the context-aware system, recognition errors are less probable. Therefore, the context-aware user simulator provides more successful dialogs, which is a very important implication for the area of automatic generation of dialog corpora.

Our analysis shows that not only the dialogs of the context-aware system achieve their goals more frequently, but also their average completion time is shorter. As Fig. 8 shows, the average duration of the S1 dialogs acquired using the context-unaware system is 9.4 turns. This duration decreases to 4.8 turns using the context-aware system. With regard to the S2 dialogs, the average duration using the context-unaware system is 12.6 turns. This duration decreases to 8.2 turns using the context-aware system.

We have also evaluated the percentage of completed subgoals in the S2 type, that is, the percentage of services that are successfully provided regardless of the success of the dialog as a whole (the dialog is only considered successful when it provides the complete set of services that were required). This percentage is 54.3 % for the context-unaware system and 77.6 % for the context-aware dialogs. These results can also be explained by the introduction of context information. This information makes possible to directly incorporate values in the system that are needed to successfully achieve the required services without employing additional user and system turns. This not only reduces the information that the user must convey, but also the number of possible confirmations of these values.

Manual analysis of the dialogs also shows that the different phases of the dialogs (greeting, exchange of information, access to the services, ending) are fairly realistic when context information is introduced to select the different system actions and access the different services.

Table 2 Results of the evaluation of the dialog management methodology

	% unseen	% exact	% correct	% error
Context-unaware	21.77 %	88.11 %	96.45 %	3.55 %
Context-aware	17.68 %	91.15 %	98.23 %	1.77 %

6.2 Evaluation results for the dialog management methodology

Two dialog managers were developed using, respectively, each one of the two corpora of 500 dialogs acquired using the context-aware and the context-unaware systems. A 5-fold cross-validation process was used to carry out the evaluation of both managers. Each one of the acquired corpus was randomly split into five subsets (20 % of the corpus). Our experiment consisted of five trials. Each trial used a different test subset out of the five subsets, and the remaining 80 % of the corpus was used as the training set. A validation subset (20 %) was extracted from each training set. MLPs were trained using the backpropagation with momentum algorithm. The topology used was two hidden layers with 110 units each.

Table 2 shows the results of the evaluation of the different measures proposed. As can be observed, the number of unseen situations (not present in the training corpus) is reduced in the context-aware system, the variability of the different dialogs is reduced (given that the number of turns is also reduced, as shown in the dialog examples presented in Fig. 6). This is the main reason for obtaining better results for the rest of measures in the evaluation of the dialog manager developed for the context-aware system.

The results of the *% exact* and *% correct* measures show the satisfactory operation of the developed dialog manager for both systems. The codification developed to represent the state of the dialog and the good operation of the MLP classifier make it possible for the answer generated by the manager to agree with the reference answer for the corresponding turn by a percentage of 88.11 % and 91.15 %, respectively, for the context-unaware and context-aware systems.

Finally, the percentage of answers generated by the MLP that can cause the failure of the system is only 1.77 % for the context-aware system. An answer that is coherent with the current state of the dialog is generated in 98.23 % of cases for this system. These last two results also demonstrate the correct operation of the classification methodology developed to deal with the uncertainty of the complete set of possible situations during a dialog.

6.3 Evaluation results of the high-level dialog features

As stated in the previous section, the second group of experiments covers the following statistical properties: (i) Di-

Table 3 Results of the high-level dialog features defined for the comparison of the two kinds of dialogs for the railway information system

	Context-unaware		Context-aware	
	Type S1	Type S2	Type S1	Type S2
Percentage of different dialogs	92.9 %	98.3 %	71.9 %	83.7 %
Number of repetitions of the most seen dialog	5	3	12	7
Number of turns of the most seen dialog	7	9	5	7
Number of turns of the shortest dialog	5	7	5	7
Number of turns of the longest dialog	25	27	17	19

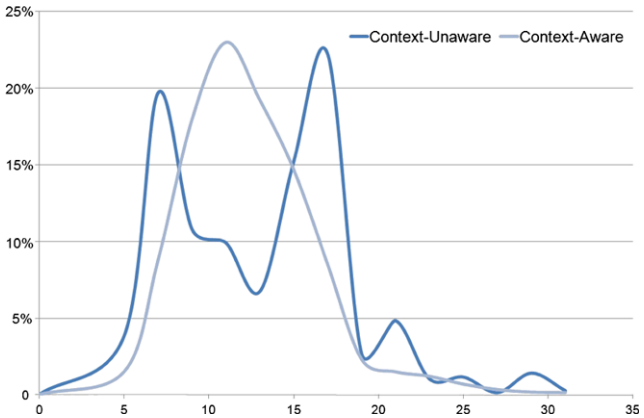


Fig. 9 Task length distribution

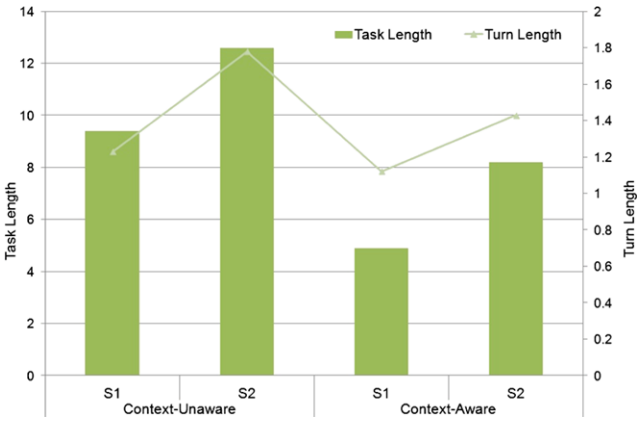


Fig. 10 Mean task and turn length

alog length, measured as the mean and shape of the distribution of the number of turns per task, and the number of turns of the shortest dialog, the number of turns of the longest dialog, and the number of turns of the most seen dialog; (ii) Different dialogs in each corpus, measured in the percentage of different dialogs and the number of repetitions of the most seen dialog; (iii) Turn length, measured by the number of actions per turn; (iv) Participant activity as a ratio of system and user actions per dialog. Table 3 shows the comparison of the different high-level measures for the context-aware and context-unaware systems.

As can be observed, there is also a reduction in the number of turns of the longest, shortest and most seen dialogs for the context-aware system. The number of different dialogs is also lower using the context-aware system, due to the reduction in the number of turns, as can be observed in the number of repetitions of the most seen dialog. This is because users have more variability in order to provide the different information that is needed to access the different services in the context-unaware system.

In fact, the shape of the distributions for the task length for the total of dialogs acquired with or without context information (Fig. 9) shows that the context-unaware dialogs have the largest standard deviation given that the task length of these dialogs is more disperse. Context-aware dialogs have minimum deviation since the successful dialogs are usually those which require the minimum number of turns to achieve the objective(s) predefined for both kinds of scenarios.

Figure 10 shows that there is a slight reduction in the mean values of the turn length for the context-aware dialogs. These dialogs are statistically shorter, as they provide 1.1 actions per user turn instead of the 1.2 actions provided by the context-unaware dialogs for the S1 scenarios, and 1.4 actions instead of 1.8 for the S2 scenarios. This is again because the users have to explicitly provide more information in the context-unaware system.

In the rest of the paper we have grouped the outcomes of scenarios S1 and S2 into a single result. This way, regarding the dialog participant activity, Fig. 11 shows the ratio of user versus system actions. Context-aware dialogs have a higher proportion of system actions, as less information requires confirmation in the context-aware system.

6.4 Evaluation results for the dialog style and cooperativeness

The experiments described in this subsection cover the following statistical properties: frequency of different user and system actions (dialog acts), and proportion of goal-directed actions (request and provide information) versus grounding actions (confirmations) and rest of actions.

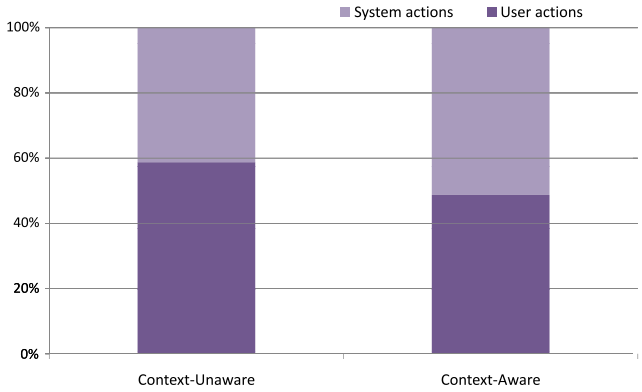


Fig. 11 Ratio of user vs. system actions

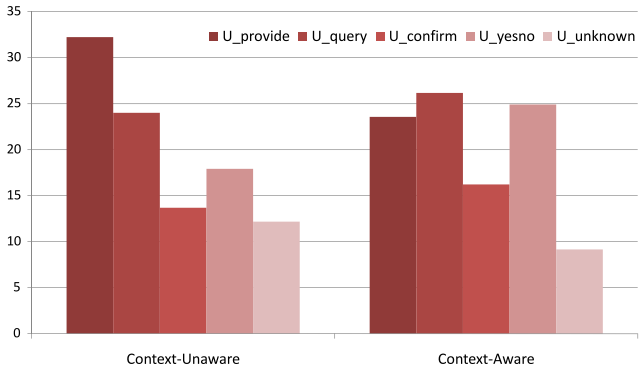


Fig. 12 Histogram of user dialog acts

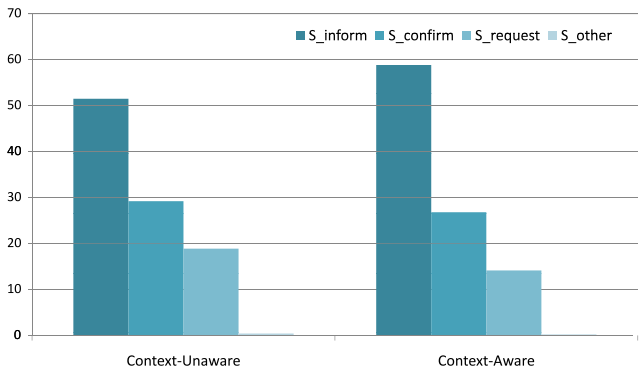


Fig. 13 Histogram of system dialog acts

The histograms in Figs. 12 and 13, respectively, show the frequency of the most dominant user and system dialog acts in the context-aware and context-unaware dialogs. On the system side, *S_request*, *S_confirm*, and *S_inform* indicate actions through which the system respectively requests, confirms, or provides information. *S_other* stands for other types of system prompts (e.g, *Waiting* and *Not-Understood* dialog acts). On the user side, *U_provide*, *U_query*, *U_confirm*, and *U_yesno*, respectively, identify actions by which the user provides, requests, confirms information or gives a yes/no answer, while *U_unkown* represents all other user actions (e.g, dialog formalities or out of task information).

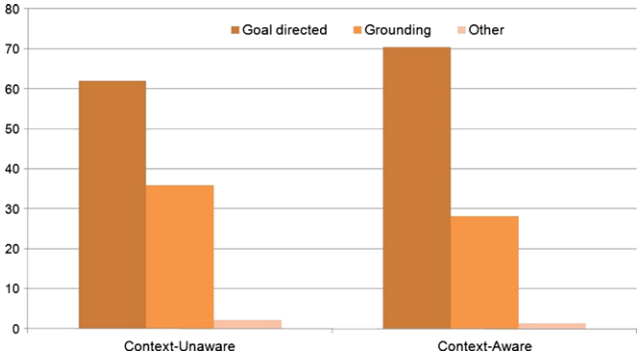


Fig. 14 Proportions of dialog spent on-goal directed actions, ground actions and other possible actions

In both cases, it can be observed that there are significant differences in the distribution of dialog acts. On the one hand, it can be observed that users need to provide less information using the context-aware architecture. This explains the higher proportion for the rest of user actions in the context-aware system. A higher proportion of yes/no actions for the context-aware dialogs is also observed. These actions are mainly used to confirm that the specific services have been correctly provided using context information.

On the other hand, there is a reduction in the system requests when the context-aware architecture is used. This explains the higher proportion of the inform and confirmation system actions in the context-aware system.

Finally, we grouped all user and system actions into three categories: “goal directed” (actions to provide or request information), “grounding” (confirmations and negations), and “rest”. Figure 14 shows a comparison between these categories. As can be observed, the dialogs provided by the context-aware system have a better quality, as the proportion of goal-directed actions is higher.

6.5 Evaluation with real users

Finally, we evaluated the behavior of our system with real users using the same set of type S1 and S2 scenarios designed for the user simulation. A total of 150 dialogs were recorded from interactions of six users employing the context-aware and context-unaware systems. The evaluation was carried out by students and lecturers in our department following the types of scenarios described in the paper in different settings with their own devices. An objective and subjective evaluation were carried out. We considered the following measures for the objective evaluation:

1. Dialog success rate (% success). This is the percentage of successfully completed tasks. In each scenario, the user has to obtain one or several items of information, and the dialog success depends on whether the system provides

Table 4 Results of the objective evaluation of the context-aware and context-unaware systems with real users

	% success	nT	% confirm	% ECR	nCE	nNCE
Context-unaware	84 %	12.4	34 %	82 %	0.84	0.18
Context-aware	93 %	7.2	26 %	91 %	0.88	0.09

correct data (according to the aims of the scenario) or incorrect data to the user.

2. Average number of turns per dialog (nT).
3. Confirmation rate (% confirm) was computed as the ratio between the number of explicit confirmations turns (nCT) and the number of turns in the dialog (nCT/nT).
4. Average number of corrected errors per dialog (nCE). This is the average of errors detected and corrected by the dialog manager. We have considered only those errors that modify the values of the attributes and thus could cause the failure of the dialog. The errors are detected using the confidence scores provided by the automatic speech recognizer and the speech understanding module. Implicit and explicit confirmations are employed to confirm or again require values detected with low reliability.
5. Average number of uncorrected errors per dialog (nNCE). This is the average of errors not corrected by the dialog manager. Again, only errors that modify the values of the attributes are considered.
6. Error correction rate (% ECR). The percentage of corrected errors, computed as $\text{nCE}/(\text{nCE} + \text{nNCE})$.

The results presented in Table 4 show that both systems could interact correctly with the users in most cases. However, the context-aware system obtained a higher success rate, improving the context-unaware results by 9 % absolute. Using the context-aware system, the average number of required turns is also reduced from 12.4 to 7.2. These values are slightly higher for both systems due to the fact that in some dialogs the real users provided additional information which was not mandatory for the corresponding scenario or asked for additional information not included in the definition of the scenario once its objectives were achieved.

The confirmation and error correction rates were also improved by the context-aware system, as context information makes it possible to require less information from the user, reducing the probability of introducing ASR errors. The main problem detected was that when there was a user input misrecognized with a very high ASR confidence, this erroneous information was forwarded to the dialog manager. However, as the success rate shows, this fact did not have a considerable impact on the system operation.

In addition, we asked the users to complete a questionnaire to assess their subjective opinion about the system performance. The questionnaire had five questions: (i) Q1: *How well did the system understand you?*; (ii) Q2: *How well did you understand the system messages?*; (iii) Q3: *Was it easy for you to get the requested information?*; (iv) Q4: *Was the interaction rate adequate?*; (v) Q5: *Was it easy for you to correct the system errors?*. The possible answers for each one of the questions were the same: *Never*, *Seldom*, *Sometimes*, *Usually*, and *Always*. All the answers were assigned a numeric value between one and five (in the same order as they appear in the questionnaire). Table 5 shows the average results of the subjective evaluation.

Table 5 Results of the subjective evaluation of the context-aware and context-unaware systems with real users (0 = worst, 5 = best evaluation)

	Q1	Q2	Q3	Q4	Q5
Context-unaware	4.6	3.6	3.8	3.4	3.2
Context-aware	4.7	3.9	4.3	4.2	3.3

From the results, it can be observed that both systems are considered to correctly understand the different user queries and obtain a similar evaluation regarding the facility of correcting errors introduced by the ASR module. However, the context-aware system has a higher evaluation rate regarding the facility of obtaining the data required to fulfill the complete set of objectives of the scenario and the suitability of the interaction rate during the dialog.

7 Conclusions

Context-aware systems in combination with mobile devices offer new opportunities in the areas of natural language processing and intelligent information retrieval from the web. In this paper, we have presented a system architecture that combines different aspects from these important research areas to provide context aware adaptable web services. This allows us to deal with the increasing complexity that the design of these kinds of systems require, including sensing changes in the user environment, its integration in hand-held mobile devices to access web services from almost everywhere, and the use of the most natural communication between the user and the system. We have described our context model and provided a detailed description of the architecture's main components.

In our architecture, the different functionalities are distributed into specific modules, taking profit of the web to guarantee the most efficient, natural and adapted service to the user. Three subsystems have been defined that carry out sensing and rendering functions, context-aware web services management, and communication with the user. Context information is captured, updated, managed and stored by means of the interaction between the different modules in the three layers. The Appear IQ Platform has been integrated for the web services management, dealing with the different processes that are required to implement this layer.

One of the most important contributions of our architecture is that it provides a natural and intelligent interaction to access web services by means of the integration of context-aware spoken dialog systems in the communication layer. This way, users can access these services using their voice; this allows the system to provide the most natural communication with the user. On the other hand, it also allows the system to be employed in environments in which traditional interfaces are not supported and also facilitates access for people with disabilities.

A Context Manager has been introduced in the architecture of the dialog system to facilitate the provisioning of personalized services using spoken interfaces. This module deals with context information used to adapt the different modules of the dialog system. To store this context information, we have defined a data structure that manages this information into user profiles. These profiles include not only external context, but also information about the user's preferences and needs.

We have also defined our own methodology for dialog management in dialog systems. This methodology is based on the use of a classification process to select the system answer by taking into account the previous history of the dialog. We have adapted this methodology to build a context-aware dialog manager by using a data structure that stores not only the information provided by the user regarding the task, but also the context information that is provided by the context manager.

In addition, we have provided a complete implementation of our architecture in a system that provides personalized services in a railway information system. To develop this system we have defined the complete requirements for the task and develop the different modules, and the necessary information to be incorporated in the user profiles.

For its evaluation, we have employed a simulation technique that makes it possible to acquire a great number of dialogs and then carry out a detailed evaluation of the quality of both the dialogs and the services. A total of 1000 successful dialogs have been acquired using this methodology from which we have studied the influence of context information on the quality of the services that are provided by the system.

To compare the dialogs acquired using a context-aware and a context-unaware version of the system, we have defined a set of measures adapted to the main characteristics of our proposed architecture. Using these measures, we evaluate the success of the dialogs and services provided, as well as their efficiency and variability with regard to the different objectives specified in the set of dialog scenarios.

The results of the evaluation show that context information not only allows a higher success rate in the provision of web services, but also its use increases the quality of the provided services. Using context information, the time required to provide a service can be reduced by 50 % in several cases. The quality of the interaction between the user and the system is increased, due to the fact that context-aware dialogs present a better ratio of goal-directed actions selected by the system to successfully provide the different services. This way, actions that might discourage users (e.g., confirmations or re-request of information) are reduced.

We have currently adapted the implementation of our architecture for the railway information domain to allow the complete interaction by means of our website. As future work, we intend to carry out a study of the relationships between this subjective opinion and the values of the different statistical measures defined in this work. We also want to carry out a detailed study of the user rejections of system-hypothesized actions using the values extracted from the user profile and extending the evaluation with a higher number of users.

Finally, we also want to apply our context-aware architecture to deal with more complex tasks. The methodology that we have developed for dialog management permits an easy modelization of dialog management in slot-filling tasks, which are very common in dialog systems. For more difficult domains, a previous plan recognition phase would be necessary. Information regarding the task is centralized in our approach in the *DR*. Thus, the adaptation to new tasks consists of adapting the structure of this register and the user profile to the requirements of the new task.

Acknowledgements Research funded by projects CICYT TIN2011-28620-C02-01, CICYT TEC 2011-28626-C02-02, CAM CONTEXTS (S2009/TIC-1485), and DPS2008-07029-C02-02.

References

1. Afsarmanesh H, Masís VG, Hertzberger L (2003) Virtual community support in telecare. In: Proc of the 4th IFIP working conference on virtual enterprises (PRO-VE'03), pp 211–220
2. 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, pp 124–131
3. Almeida DD, Baptista CDS, Silva ED, Campelo C, Figueiredo HD, Lacerda Y (2006) A context-aware system based on service-oriented architecture. In: Proc of the 20th int conference on advanced information networking and applications (AINA'06), pp 205–210
4. Athanasopoulos D, Zarras A, Issarny V, Pitoura E, Vassiliadis P (2008) CoWSAMI: interface-aware context gathering in ambient intelligence environments. Pervasive Mob Comput 4(3):360–389
5. Bressan S, Goh C, Levina N, Madnick S, Shah A, Siegel M (2000) Context knowledge representation and reasoning in the context interchange system. Int J Appl Intell 13:165–180
6. Brown B, Randell R (2004) Building a context sensitive telephone: some hopes and pitfalls for context sensitive computing. Comput Support Coop Work 13:329–345
7. Chen I, Yang S, Zhang J (2006) Ubiquitous provision of context aware web services. In: Proc of IEEE int conference on services computing, pp 60–68

8. Dey A, Abowd G (2000) Towards a better understanding of context and context-awareness. In: Proc of the 2000 conference on human factors in computer systems (CHI'00), pp 304–307
9. Felfernig A, Friedrich G, Isak K, Shchekotykhin K, Teppan E, Jannach D (2009) Automated debugging of recommender user interface descriptions. Int J Appl Intell 31:1–14
10. Fikes R, Kehler T (1985) The role of frame-based representation in knowledge representation and reasoning. Commun ACM 28:904–920
11. Glass J, Flammia G, Goodine D, Phillips M, Polifroni J, Sakai S, Seneff S, Zue V (1995) Multilingual spoken-language understanding in the MIT voyager system. Speech Commun 17:1–18
12. González-Rodríguez M, Manrubia J, Vidau A, González Gallego M (2009) Improving accessibility with user-tailored interfaces. Int J Appl Intell 30:65–71
13. Griol D, Hurtado LF, Segarra E, Sanchis E (2008) A statistical approach to spoken dialog systems design and evaluation. Speech Commun 50(8–9):666–682
14. Griol D, Callejas Z, López-Cózar R (2009) Acquiring and evaluating a dialog corpus through a dialog simulation technique. In: Proc of the 9th SIGdial workshop, pp 326–332
15. Griol D, Sanchez-Pi N, Carbó J, Molina J (2009) Context-aware approach for orally accessible web services. In: Proc of IEEE/WIC/ACM international joint conference on web intelligence and intelligent agent technology, pp 171–174
16. Han B, Jia W, Shen J, Yuen M (2008) Context-awareness in mobile web services. In: Proc of the 2nd int symposium on parallel and distributed processing and applications (ISPA 2004), pp 519–528
17. Henricksen K, Indulska J, Rakotonirainy A (2002) Modeling context information in pervasive computing systems. In: Proc of the 1st int conference on pervasive computing, pp 167–180
18. Iglesias A, Martínez P, Aler R, Fernández F (2009) Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. Int J Appl Intell 31:89–106
19. Kang H, Suh E, Yoo K (2008) Packet-based context aware system to determine information system user's context. Expert Syst Appl 35:286–300
20. Keidl M, Kemper A (2004) Towards context-aware adaptable web services. In: Proc of the 13th international world wide web conference (WWW'04), pp 55–65
21. Kerschberg L, Weishar D (2000) Conceptual models and architectures for advanced information systems. Int J Appl Intell 13:149–164
22. Ko J, Murase F, Mitamura T, Nyberg E, Tateishi M, Akahori I (2006) Context-aware dialog strategies for multimodal mobile dialog systems. In: Proc of AAAI international workshop on modeling and retrieval of context, pp 7–12
23. Lech T, Wienhofen L (2005) AmbieAgents: a scalable infrastructure for mobile and context-aware information services. In: Proc of the 4th int joint conference on autonomous agents and multiagent systems (AAMAS'05), pp 625–631
24. Liu J, Seneff S, Zue V (2010) Dialogue-Oriented review summary generation for spoken dialogue recommendation systems. In: Proc of human language technologies: the 2010 annual conference, pp 64–72. of the North American Chapter of the ACL
25. López-Cózar R, Araki M (2005) Spoken, multilingual and multimodal dialogue systems: development and assessment. Wiley, New York
26. López-Cózar R, la Torre AD, Segura J, Rubio A, Sánchez V (2002) Testing dialogue systems by means of automatic generation of conversations. Interact Comput 14(5):521–546
27. Matsumura K, Ishida T, Murakami Y, Fujishiro Y (2006) Situated web service: context-aware approach to high-speed web service communication. In: Proc of IEEE int conference on web services (ICWS'06), pp 673–680
28. Melin H, Sandell A, Ihse M (2001) CTT-bank: a speech controlled telephone banking system—an initial evaluation. In: TMH quarterly progress and status report (TMH-QPSR), vol 1, pp 1–27
29. Minsky M (1975) The psychology of computer vision. McGraw-Hill, New York, pp 211–277. Chapter: A framework for representing knowledge
30. Moubaiddin A, Obeid N (2009) Partial information basis for agent-based collaborative dialogue. Int J Appl Intell 30:142–167
31. Mäntyjärvia J, Seppänen T (2003) Adapting applications in handheld devices using fuzzy context information. Interact Comput 15(4):521–538
32. Naguib H, Coulouris G, Mitchell S (2001) Middleware support for context-aware multimedia applications. In: Proc of the 3rd int working conference on new developments in distributed applications and interoperable systems, pp 9–22
33. Newcomer E, Lomow G (2005) Service-oriented architecture with web services. Addison-Wesley, Reading
34. Nieto-Carvajal I, Botía J, Ruiz P, Gómez-Skarmeta A (2004) Implementation and evaluation of a location-aware wireless multi-agent system. In: Proc of the int conference on embedded and ubiquitous computing (EUC'04), pp 528–537
35. Nihei K (2004) Context sharing platform. NEC J Adv Technol 1(3):200–204
36. O'Shea K (2012) An approach to conversational agent design using semantic sentence similarity. Int J Appl Intell. doi:10.1007/s10489-012-0349-9
37. Osland P, Viken B, Solsvik F, Nygreen G, Wedvik J, Myklbust S (2006) Enabling context-aware applications. In: Proc of the int conference on convergence in services, media and networks (ICIN'06), pp 1–6
38. Paek T, Pieraccini R (2008) Automating spoken dialogue management design using machine learning: an industry perspective. Speech Commun 50:716–729
39. Partala T, Kallinen A (2012) Understanding the most satisfying and unsatisfying user experiences: emotions, psychological needs, and context. Interact Comput 24(1):25–34
40. Pieraccini R, Rabiner L (2012) The voice in the machine: building computers that understand speech. MIT Press, Cambridge
41. Prezerakos G, Tselikas N, Cortese G (2007) Model-driven composition of context-aware web services using contextUML and aspects. In: Proc of IEEE int conference on web services (ICWS'07), pp 320–329
42. Rouillard J (2007) Web services and speech-based applications around voicexml. J Netw 2(1):27–35
43. Rumelhart DE, Hinton GE, Williams RJ (1986) PDP: computational models of cognition and perception, I. MIT Press, Cambridge, pp 319–362. Chapter: Learning internal representations by error propagation
44. Schatzmann J, Georgila K, Young S (2005) Quantitative evaluation of user simulation techniques for spoken dialogue systems. In: Proc of the 6th SIGdial workshop, pp 45–54
45. Schatzmann J, Weilhammer K, Stuttle M, Young S (2006) A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. Knowl Eng Rev 21(2):97–126
46. Schatzmann J, Thomson B, Weilhammer K, Ye H, Young S (2007) Agenda-based user simulation for bootstrapping a POMDP dialogue system. In: Proc of human language technologies HLT/NAACL'07 conference, pp 149–152
47. Schatzmann J, Thomson B, Young S (2007) Error simulation for training statistical dialogue systems. In: Proc of IEEE automatic speech recognition and understanding workshop (ASRU'07), pp 526–531
48. Scheffler K, Young S (2001) Corpus-based dialogue simulation for automatic strategy learning and evaluation. In: Proc of the 2nd

meeting of the North American chapter of the association for computational linguistics (NAACL-2001). Workshop on adaptation in dialogue systems, pp 64–70
49. Seneff S, Adler M, Glass J, Sherry B, Hazen T, Wang C, Wu T (2007) Exploiting context information in spoken dialogue interaction with mobile devices. In: Proc of int workshop on improved mobile user experience (IMUx'07), pp 1–11
50. Sheng Q, Benatallah B (2005) ContextUML: a UML-based modeling language for model-driven development of context-aware web services development. In: Proc of IEEE int conference on mobile business (ICMB'05), pp 206–212
51. Stolcke A, Coccaro N, Bates R, Taylor P, Ess-Dykema CV, Ries K, Shriberg E, Jurafsky D, Martin R, Meteer M (2000) Dialogue act modeling for automatic tagging and recognition of conversational speech. Comput Linguist 26(3):339–373
52. Strauss P, Minker W (2010) Proactive spoken dialogue interaction in multi-party environments. Springer, Berlin
53. Truong H, Dustdar S, Baggio D, Corlosquet S, Dorn C, Giuliani G, Gombotz R (2008) inContext: a pervasive and collaborative working environment for emerging team forms. In: Proc of the int symposium on applications and the Internet (SAINT'08), pp 1–8
54. Truong HL, Dustdar S (2009) A survey on context-aware web service systems. Int J Web Inf Syst 5(1):5–31
55. Truong HL, Juszczyk L, Manzoor A, Dustdar S (2007) ESCAPE - an adaptive framework for managing and providing context information in emergency situations. In: Proc of the 2nd European conference on smart sensing and context (EuroSSC 2007), pp 207–222
56. Wang C, Li T, Feng L (2008) Context-aware environment-role-based access control model for web services. In: Proc of the int conference on multimedia and ubiquitous engineering (MUE'08), pp 288–293
57. Zuidweg M, Filho JG, van Sinderen M (2003) Using P3P in a web services-based context-aware application platform. In: Proc of the 9th open European summer school and IFIP (EUNICE'03), pp 238–243