Providing personalized Internet services by means of context-aware spoken dialogue systems

David Griol [a,*], José Manuel Molina [a] and Zoraida Callejas [b]
[a] *Department of Computer Science, Carlos III University of Madrid, Avda. de la Universidad, 30, 28911 Leganés, Spain*
E-mail: {david.griol,josemanuel.molina}@uc3m.es
[b] *Department of Languages and Computer Systems, University of Granada, CITIC-UGR, C/ Pdta. Daniel Saucedo Aranda s/n, 18071 Granada, Spain*
E-mail: zoraida@ugr.es

Abstract. The widespread use of new mobile technology implementing wireless communications enables a new type of advanced applications to access information services on the Internet. In order to provide services which meet the user needs through intelligent information retrieval, the system must sense and interpret the user environment and the communication context. Though context-awareness is vital to provide services adapted to the user preferences, it cannot be useful if such services are difficult to access. The development of spoken dialogue systems for these applications facilitates interaction in natural language with the environment which is also benefited from contextual information. In this paper, we propose a framework to develop context-aware dialogue systems that dynamically incorporate user specific requirements and preferences as well as characteristics about the interaction environment, in order to improve and personalize web information and services. We have identified the major components for context-aware dialogue systems and placed them within a general-purpose architecture. The framework also describes a representation mode based on a dialogue register in order to share information between the elements of the architecture, and incorporates statistical methodologies for dialogue management in order to reduce the effort required for both the implementation of a new system and the adaptation to a new task. We have evaluated our proposal developing a travel-planning system, and provide a detailed discussion of its positive influence in the quality of the interaction and the information and services provided.

Keywords: Ambient Intelligence (AmI), context-aware systems, spoken dialogue systems, user adaptation, systems evaluation, multimodality, web interfaces, speech interaction, user interfaces

1. Introduction

Ambient Intelligence (AmI) and Smart Environments (SmE) emphasize on more efficient services support, user-empowerment, and support for human interactions. As discussed in [19], in this vision accessibility and usability of the interfaces is a key aspect for the environment to respond to the presence of individuals in an unobtrusive way [21]. To achieve these objectives, AmI builds on two key technologies: ubiquitous computing and intelligent user interfaces. For this reason, AmI systems usually consist of a set of interconnected computing and sensing devices which surround the user pervasively in his environment and are invisible to him, providing a service that is dynamically adapted to the interaction context, so that users can interact with the system and thus perceive it as intelligent. 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.

[*]Corresponding author. E-mail: dgriol@inf.uc3m.es.

To ensure accessibility and personalization, it is necessary to provide an effective and dynamically adapted interaction between the user and the system [17]. With this objective, as an attempt to enhance and ease human-computer interaction, in the last years there has been an increasing interest in simulating human-to-human communication, employing the so-called dialogue systems [9,27,29]. A dialogue system is an automatic system which functionality is accessible though natural language, emulating human conversation.

Speech and natural language technologies allow users to communicate in a flexible and efficient manner, making possible to access applications in which traditional input interfaces cannot be used (e.g. in-car applications, access for disabled persons, etc.). Also speech-based interfaces work seamlessly with small devices and allow users to easily invoke local applications or access remote information. For this reason, spoken dialogue systems are becoming a strong alternative to traditional graphical interfaces which might not be appropriate for all users and/or applications.

Several authors [33,53] have highlighted the importance of standardizing and sharing a common base for context sensitivity and web information systems. However, most context-aware systems are closed, composed of highly coupled constituents, and generated in an ad-hoc manner for a specific domain [6,53]. The same problem occurs when designing a dialogue system. There is a high variety of applications in which dialogue systems can be used, one of the most widespread are information retrieval from the web [11,56], database systems [5,28], and recommendation systems [24,52]. 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 way, the adaptation of the hand-crafted designed systems to new tasks is a time-consuming process that implies a considerable effort, with the ever-increasing problem of dialogue complexity [35,36,39]. In addition, although much work emphasize the importance of taking into account context information not only to solve the tasks presented to the dialogue system by the user, but also to enhance the system performance in the communication task, this information is not usually considered when designing a dialogue model [20,49].

For these reasons, there has been a growing interest during the last decade in developing statistical approaches to model the different modules that compose a dialogue system [54]. These approaches are usually based on modelling the different processes probabilistically and learning the parameters of the different statistical models from a dialogue corpus. Statistical models can be trained from real dialogues, accounting for the variability in user behaviours. The final objective is to develop dialogue systems that have a more robust behaviour, better portability, and are easier to adapt to different user profiles or tasks. The most common methodology for machine-learning of dialogue strategies is based on an optimization problem using Markov Decision Process (MDP) and reinforcement methods [23]. The main drawback of this approach is the large state space of practical spoken dialogue systems, whose representation is intractable if represented directly [55]. Partially Observable MDPs (POMDPs) outperform MDP-based dialogue strategies since they provide an explicit representation of uncertainty [40]. However, they are limited to small-scale problems, since the state space would be huge and exact POMDP optimization is again intractable for practical applications [55].

In this paper, we propose a method for the development of spoken dialogue systems which allows not only a more natural and intelligent interaction with users, but also to provide them with context-aware web information adapted to their location, preferences and needs. To facilitate a general-purpose behaviour and reduce the effort required for both the implementation of a new system and the adaptation to a new task, we propose to use a statistical methodology for dialogue management in which the dialogue strategy is automatically learned from a dialogue corpus. The statistical dialogue model is then enriched with context information used to adapt the interaction and provide personalized, context-aware services through a natural language conversation. To this end, our dialogue manager is implemented using a classifier based on neural networks, which takes the previous dialogue history and the context information into account to carry out the selection of the next system action. In addition, the adaptation of the dialogue system only requires the acquisition of a dialogue corpus for the new domain. A user modelling technique is proposed to carefully explore the complete set of dialogue situations that can occur during the interaction, include these situations and the corresponding system answers in the dialogue model, and reduce the effort and time that is required for the acquisition of this corpus.

We also provide a complete implementation of our proposal in a travel planning information system, which provides context-aware information related to approaches to the city, flight schedules, weather forecast, car rental, hotel booking, relevant tourist attractions, theatre listings and film showtimes. We have evaluated the developed system and assessed the influence of context information in the quality of the acquired dialogues and the information provided. The results of this evaluation show that context information not only allows a higher success rate in the provision of web information, but also shorter and more informative interactions with users.

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 dialogue systems. Section 3 describes the main characteristics of our architecture for providing context-aware adaptable services using speech-based interfaces. Section 4 describes our proposal to develop context-aware dialogue systems. Section 5 shows a practical implementation of our architecture to generate a context-aware travel-planning system and Section 6 shows the results of its evaluation using a set of defined measures. Finally, our conclusions and future work are presented in Section 7.

2. Related work

According to [7], "*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 application 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. Some authors identify two types of context: *internal* and *external* [18]. 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 studies in the literature focus only on external context. One of the most popular is location information. For example, the Akogrimo project [34] aims at supporting mobile users to access data, knowledge, and computational services on the Grid. It only focuses on context that is related to situations of mobile users, such as user presence and location, and environmental information. Similarly, SMAUG [32] is a multi-agent context-aware system that allows tutors and students 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 [22] 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 [51]. 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. present an application for supporting virtual elderly assistance communities within the framework of the TeleCARE project [1]. One of the most important contributions of our work is to combine both internal and external context information, given that it is essential to provide a useful personalization of the web information and is of great interest to optimize the speech-based interface.

In addition, context awareness should also be employed to adapt the services provided to the user through 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 lack of 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 speech 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 framework due to the reduced number of context-aware speech interfaces that can be found in the literature and its application to very specific domains [48].

The adaptation capabilities of speech interfaces are frequently restricted to static choices. 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 dialogues. Processing context is not only useful to adapt the systems' behaviour, but also to cope with the ambiguities derived from the use of natural language [25,26,49]. For instance, context information can be used to resolve anaphoric references depending on the context of the dialogue or the user location. The performance of a dialogue system also depends highly on the environmental conditions, such for example whether there are people speaking near the system or the noise generated by other devices.

With respect to context representation and modelling, 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 modelling 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 modelling context information. For example, the Anyserver platform [15] 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 modelled based on OWL, while Prezerakos et al. use UML to model context and web services [37]. MyCampus [42] is a Semantic Web environment for context-aware mobile services at the Carnegie Mellon University (CMU). The developed system combines technology development (e.g. OWL Semantic Web reasoning engine, context-aware agents, location tracking functionality, OWL Rule Extension, etc.) with HCI evaluation methodologies.

In our proposal, XML files are used to store the context information transmitted. We combine aspects from the tagged encoding approach with the *dialogue acts* (DA) formalism [50], 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 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, available sensors utilize these techniques through instrumentation or polling mechanisms, and extend their capability by acquiring context information from existing systems.

The Context Management Service (CMS) is an open infrastructure for managing context information developed in the Amigo project [38], which focuses on ambient intelligence for the networked home. The role of the CMS is to acquire information coming from various sources (physical sensors, user activities, and applications in process or Internet applications), combine these pieces of information and provide them to context aware services. The main objective of the Context Casting Project (C-CAST) [4] is to use context awareness and multicasting technologies to evolve mobile multimedia multicasting and exploit the integration of mobile devices with the environment. The project addresses the development of mechanisms for autonomous context driven content creation, adaptation and media delivery. As will be described in Section 3, we use a commercial platform that captures external context, then this information, together with the user profile, is used in our architecture to provide web information and services that are adapted to the user location, geographical context, communication context preferences and needs.

Reasoning techniques can be also 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. In our case, there is a module fully integrated into the dialogue system architecture that processes the semantic definition of the application, that is, not only the web services area managed, but also 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 modelled in our architecture using frames [30]. A frame is a structure for representing a concept or situation which has several associated attributes (slots) and values [8]. 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 analysed, it must be conveniently stored. Relational databases are widely used to store context information in context-aware systems out of the web services domain [16,31], even in the case of XML or ontology-based systems.

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)[1]. OASIS proposed a web services context specification[2] 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 XML files to transfer the context information modelled following the OASIS specifications, and the management of the information is carried out in the dialogue manager of the proposed dialogue system, as will be described in the Section 4.

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 the content 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 content in a form suitable for the context of the requester, optimize the communication, and select the most suitable system action in each dialogue situation.

3. Our framework to develop context-aware systems

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 dialogue system based on a previous work [14]. Figure 1 shows the complete architecture proposed for the provision of context-aware web information and services. As it can be observed, different systems and modules cooperate to provide adapted information and services. Users access this information and services by means of mobile devices or PDAs. The *Facilitator System* supplies the different services in the system and is bound to a *Dialogue System*, used to interact with users and provide them with these services. Context information is acquired and managed by means of two main systems. The *Positioning System* communicates with the ARUBA positioning system to extract and transmit positioning information of the user. Finally, a *Log Analyser System* generates and updates the user profile, which are used by the Dialogue System to adapt their behaviour taking into account the preferences detected in the users' previous dialogues and the information related to the environment for the current interaction.

We have implemented the Facilitator System using the Appear IQ Platform (AIQ)[3]. The platform features a distributed modular architecture that supports multiple network configurations and can be deployed as a distributed system. It 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 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 information and 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

[1]http://www.w3.org/TR/soap
[2]http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.pdf
[3]http://www.appearnetworks.com

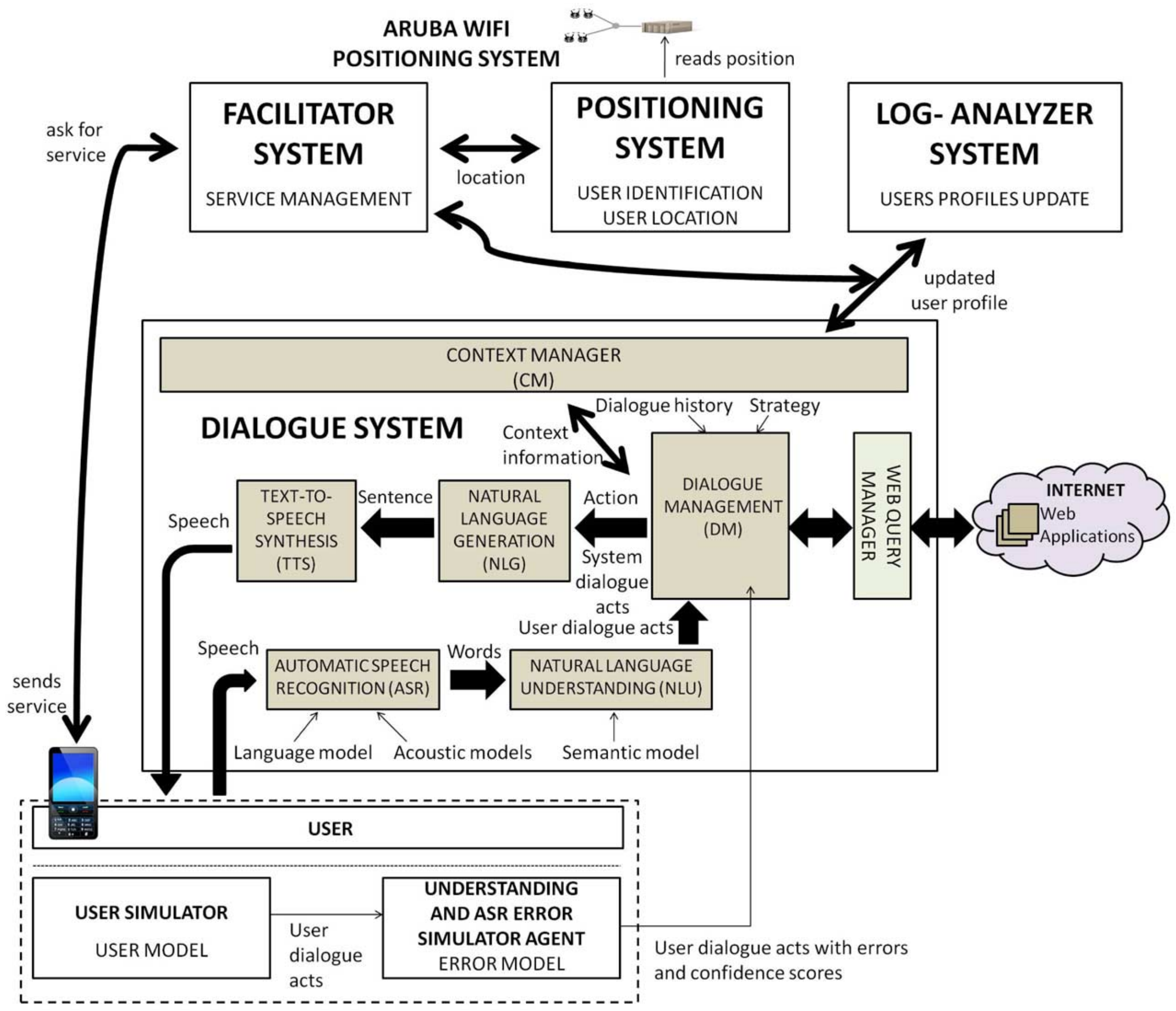


Fig. 1. Graphical scheme of the proposed context-aware architecture.

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 handheld 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 in the client side are:

– 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;
– 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;

– 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:

– The Synchronization Module compiles contextual data to gain an understanding of the user's informational needs;
– Available data is filtered against the user's context to determine what information should be the most relevant;
– 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.

The complete process can be summarized as follows (Fig. 2):

1. *Device Detection*: Once the user is detected in the network by the ACP, it 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.
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 dialogue 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 dialogue manager. Each module uses this knowledge to load its specific information and models.
5. *Spoken communication*: The user starts the interaction with the dialogue system. Throughout the interaction, each module can update the active user profile. Depending on the information that is modified, a Context Manager sends the value of the new features only to the modules in the dialogue system that require such information. Each module in the dialogue 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 dialogue session.
7. *Discard Service*: when a user leaves the network or if a context condition has changed for a service.

4. Management of context-awareness within our framework

As stated in the introduction, a dialogue system can be understood as an automatic system able of emulating human conversation, with the aim that the system meets a specific functionality (usually providing information or performing a specific 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 dialogue turns. To do this, it is necessary to endow the system with the abilities to reference information that has appeared previously during the dialogue, take the initiative to recover the dialogue after a failure, request information that is necessary to fulfil 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 utter-

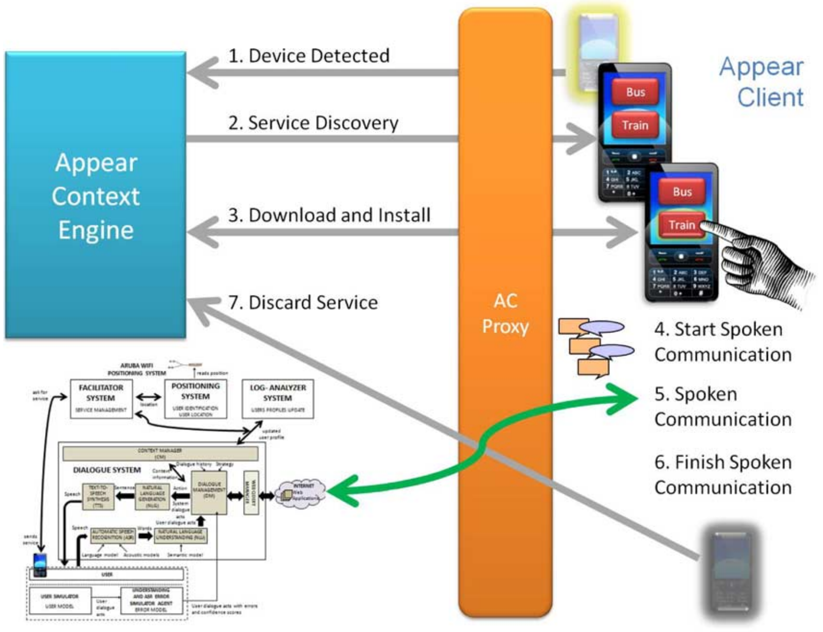


Fig. 2. Process followed in the proposed architecture to provide context-aware web information and services.

ance: 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), access web services and databases to extract the information required by the user, adapt the interaction to the context features described above, decide what action or actions should be performed after each user request 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 dialogue system can be described in terms of the following modules. 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 *Dialogue Manager* determines the next action to be taken by the system following a dialogue strategy. The *Web Query Manager* receives requests to the system web services, processes the information and returns the result to the dialogue manager. The *Natural Language Generator module* (NLG) receives a formal representation of the system action and generates a user response that can include multimodal information (video, data tables, images, gestures, etc.). Finally, a *Text to Speech Synthesizer* (TTS) generates the audio signal transmitted to the user.

To deal with context information and personalize web services, we have incorporated a new module in the architecture of a dialogue system, as shown in Fig. 1. This module, that we have called *Context Manager*, loads, updates and manages context information associated to each of the users. Once the Context Manager receives the user's identification, it reads the context information from the user profile at the beginning of the interaction in order to adapt the behaviour of the different modules in the dialogue system taking this information into account. 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.
- Skill level: This level is estimated by taking into account variables like the number of previous sessions, dialogues and dialogue 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: They store the count of each action over the system that a user performs. Users' preferences are automatically evaluated considering the user's most required services during the previous dialogues, date and hour of the previous interactions, most frequent objectives and restrictions, and preferred output modality.

This information is used by the push/provisioning, synchronization and device management context-aware modules in the ACE, which 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 information and enhancement of web services at boot or runtime. Then, the device accesses the information on the Internet taking into account the data stored in the ACE and including the context information.

The dialogue system receives user utterances, which are transmitted to the automatic speech recognition system and semantically analysed by the natural language understanding module to obtain a frame-based representation including confidence scores. This process can lead the system to ask the user for additional information, to require a confirmation for information already provided by the user, or to generate a response once the user has provided all the required information. In the last case, the Web Query Manager receives this information and provides the dialogue manager with the result of the query (e.g., the list of recommended hotels that fulfils the user requirements, weather forecast for the specific city and dates, etc.).

The system answer is then transmitted to the language generator module. This module takes into account the information received from both the dialogue manager and the web query manager 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 dialogue.

The transmission of the context between modules is carried out by sending XML packages based on the OASIS Web Services Context Specification. Figure 3 shows an example of a XML package including the context information that is available after the user's identification and location.

```
<?xml version=``1.0'' encoding=``UTF-8''?>
<soap:Envelope
xmlns:soap=``www.w3.org/2002/06/soap-envelope''>
<soap:Header>
<xsd:context
xmlns=``docs.oasis-open.org/ws-caf/2005/10/wsctx''
expiresAt=``2011-10-26T23:59:00+01:00''
xmlns:wsdl=``schemas.xmlsoap.org/wsdl/''
xmlns:soapbind=``schemas.xmlsoap.org/wsdl/soap/''
xmlns:example=``example.com/context_travel_planning''
soap:mustUnderstand=``1''>
<context-identifier>
docs.oasis-open.org/ws-caf/2005/10/wsctx/abcdef:012345
</context-identifier>
<context-service>
<xsd:address> http://www.uc3m.es/inf/travel_planning
</xsd:address>
<parent-context expiresAt=``2011-11-27T23:59+01:00''>
<xsd:complexType name=``Identification''>
<xsd:user>
<xsd:name=``Antonio L\'{o}pez''/>
<xsd:gender=``Male''/>
<xsd:age=``53''/>
<xsd:role=``Passenger''/>
<xsd:speech_pathologies=``None''/>
<xsd:channel=``Mobile phone''/>
<xsd:location=``Barajas Airport''/>
<xsd:date=``2011-10-27''/>
<xsd:local_time=``09:00am''/>
<xsd:MAC_address=``00-18-41-32-0B-59''/>
<xsd:preferred_voice=``Female''/>
<xsd:preferred_query=``Hotel\_Booking''/>
<xsd:alternative_query=``Weather\_Forecast''/>
<xsd:preferred_city=``Granada''/>
<xsd:preferred_hotel\_category=``Four stars''/>
<xsd:preferred_hotel\_name=``El Cabanyal''/>
<xsd:preferred_time=``09:00am-11:00am''/>
<xsd:preferred_modality=``Speech''/>
<xsd:previous_interactions=``12''/>
<xsd:average_duration=``02:17''/>
<xsd:average_turns=``9''/>
<xsd:success=``0.80''/>
<xsd:last_dialogue=``2011-11-20 09:37''/>
<xsd:language=``Spanish''/>
</xsd:user>
</xsd:complexType>
</soap:Header>
<soap:Body>
</soap:Body>
</soap:Envelope>
```

Fig. 3. XML Package defined following the OASIS Web Services Context Specification.

4.1. Context-aware dialogue management

As described previously, the dialogue manager is in charge of processing the contextual information to decide the system response to every user request. The traditional approach to do this is to handcraft a series of rules which determine such behaviour. However, this design method is very time consuming and has the ever-increasing problem of dialogue complexity. As an alternative, statistical models can be trained from real dialogues, modelling the variability in user behaviours. Although the construction and parametrization of the model depend on expert knowledge of the task, the final objective is to develop dialogue systems that have a more robust behaviour, better portability, and are easier to adapt to different user profiles and interaction domains.

Our dialogue manager follows this paradigm and is mainly based on modelling the sequences of the system and user dialogue acts and the introduction of a partition in the space of all the possible sequences of dialogue acts. This partition, which is defined taking into account the data supplied by the user throughout the dialogue, makes the estimation of a statistical model from the training data manageable.

One of the main problems which must be considered by the dialogue manager is the propagation of errors through the different modules in the system. The recognition module must deal with the effects of spontaneous speech and noisy environments; consequently, the sentence provided by this module could incorporate some errors. The understanding module could also add its own errors (which are mainly due to the lack of coverage of the semantic domain). Therefore, it is desirable to provide the dialogue manager with information about what parts of the user utterance have been clearly recognized and understood and what parts have not. In our system, the understanding module provides the dialogue manager with the semantic representation (generally one frame or several frames) associated to the user input together with its confidence scores [10]. These confidence measures are used by our statistical dialogue manager to automatically adapt its strategy to the reliability of the information provided by the understanding module.

The new system utterance is selected by means of a classification, which we have implemented using neural networks.

In order to control the interactions integrating context information, our dialogue manager represents the dialogues as sequences of pairs (A_i, U_i), where A_i is the output of the dialogue system (the system answer) at time i, expressed in terms of dialogue acts; and U_i is the semantic representation of the user turn (the result of the understanding process of the user input) at time i, expressed in terms of frames. This way, each dialogue is represented by:

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

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

In this framework, we consider that, at time i, the objective of the dialogue 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 dialogue, that is to say, the sequence of states of the dialogue preceding time i:

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

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 dialogue preceding time i). This data structure, that we call *Dialogue Register* (DR), contains the information provided by the user throughout the dialogue and the context information that is provided by the Context Manager.

All the information captured by the DR_i at a given time i is a summary of the information provided by the sequence $S_1, \ldots, S_{i-1}$. Note that different state sequences can lead to the same DR. For a sequence of states of a dialogue, there is a corresponding sequence of DR:

$$\begin{tikzcd} S_1, & \ldots, & S_i, & \ldots, & S_n \\ DR_0 \arrow[u] & DR_1 \arrow[u] & DR_{i-1} \arrow[u] & & DR_{n-1} \arrow[u] \end{tikzcd}$$

where DR_0 captures the default information of the dialogue manager, and the values of the following DR are updated taking into account the information supplied by the evolution of the dialogue.

For the dialogue 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 specific 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.

System Dialogue Act: (Provide-Film_Showtimes)
Here <*User-Name*>, I inform you about the film <*Film*> on <*Date*>.
Here <*User-Name*>, I inform you about films at the cinema <*Cinema*> on <*Date*>.
Here <*User-Name*>, I inform you about films at the cinema <*Cinema*> on <*Date*> at <*Time*>.
I inform you about the films at the cinema <*Cinema*>.

Fig. 4. Examples of the set of templates defined to take into account context information in the NLG module.

– 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 dialogues, the selection of the best A_i is computed as:

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

The last state (S_{i-1}) is considered for the selection of the system answer due to the fact that a user turn can provide information that is not contained in the DR, but is important to decide the next system answer. This is the case of task-independent information (e.g., *Affirmation*, *Negation* and *Not-Understood* dialogue acts). This maximization can be solved by means of a classification process, for which we use a multilayer perceptron (MLP) [41]. The input layer of the MLP receives the current situation of the dialogue, 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 dialogue acts (i.e., system prompts) defined for a specific task.

Once the dialogue manager has selected which is the next system action, this action has to be transformed 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 dialogue register and the user profile to generate an answer for the user. Figure 4 shows an example including different templates defined for the case of a travel-planning system providing film information and showtimes.

4.2. Modelling the user interaction

The success of statistical approaches for dialogue management depends on the quality of the data used to develop the dialogue model. Considerable effort is necessary to acquire and label a corpus with the data necessary to train a good model. A technique that has attracted increasing interest is based on the automatic generation of dialogues between the dialogue manager and an additional module, called the user simulator, which represents user interactions with the dialogue system. The user simulator makes it possible to generate a large number of dialogues in a very simple way, reducing the time and effort required for acquiring the corpus and learning the dialogue model. In addition, the construction of probabilistic user models make possible to model the specific characteristics of users interacting with the developed system and automatically learn optimal system responses for the set of possible dialogue states.

We have developed a methodology to automatically generate dialogues, which can be used for learning and evaluating statistical dialogue models [13]. Our methodology is based on the interaction of a user simulator and a dialogue manager. Both modules use a random selection of one of the possible answers defined for the semantics of the task (user and system dialogue acts). At the beginning of the simulation, all the system answers are defined with the same probability. When a successful dialogue is simulated, the probabilities of the answers selected by the dialogue manager during that dialogue are incremented before beginning a new simulation.

The user simulation resembles 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 shown in Fig. 1. The semantic selected for the dialogue 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 included 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 different ASR and NLU modules.

The model employed for introducing errors and confidence scores is inspired in the one presented in [46]. 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 dialogue 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 labelled 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 dialogue 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.

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

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

– The dialogue exceeds a maximum number of system turns defined taking into account the requirements of the task.
– The answer selected by the dialogue 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 dialogue is selected once the system has provided the information defined in the objective(s) of the dialogue. The dialogues that fulfil this condition before the maximum number of turns are considered successful.

5. Case application: A travel-planning context-aware system

We have applied our context aware methodology to develop and evaluate an adaptive system for a travel-planning domain. The system provides context-aware information in natural language in Spanish about approaches to a city, flight schedules, weather forecast, car rental, hotel booking, tourist attractions, theatre listings, and film showtimes. The information offered to the user is extracted from a web page that users can visually complete to incorporate additional information about a city already present in the system, update this information or add new cities. Different Postgress databases are used to store this information and automatically update the data that is included in the application. In addition, several functionalities are related to dynamic information (e.g., weather forecast, flight schedules) directly obtained from webpages and web services. Thus, our system provides speech access to facilitate travel-planning information that is adapted to each user taking context into account.

Semantic knowledge is modelled in our architecture using the classical frame representation of the meaning of the utterance. We defined eight concepts to represent the different queries that the user can perform (*City-Approaches*, *Flight-Schedules*, *Weather-Forecast*, *Car-Rental*, and *Hotel-Booking*, *Tourist-Attractions*, *Heater-Listings*, and *Film-Show times*). Three task-independent concepts have also been defined for the task (*Affirmation*, *Negation*, and *Not-Understood*). The attributes required by the system to generate a response to the different user queries and the different system responses are shown in Table 1. A total of 101 system actions (DAs) were defined taking into account the information that the system provides, requests or confirms.

Using the *City_Approaches* functionality, it is possible to know how to get to a specific city using different means of transport. If specific means are not provided by the user, then the system provides the complete in-

Table 1

Semantic representation defined for the travel-planning domain

Query	Attributes	System responses
City_Approaches	*City*, *Means_Transport*, *Origin_City*	Ask and Confirm each of the attributes, *Provide-City_Approaches*
Flight_Schedules	*Origin_City*, *Destination_City*, *Departure_Date*, *Departure_ Hour*, *Arrival_Date*, *Arrival_Hour*, *Ticket_ Class*	Ask and Confirm each of the attributes, *Provide-Flight_Schedules*
Weather_Forecast	*City*, *Country*, *Date*	Ask and Confirm each of the attributes, *Provide-Weather_Forecast*
Car_Rental	*City*, *Country*, *Pick_ Up_Date*, *Drop_Off_Date*, *Car_Type*, *Company*, *Driver_Age*, *Office*	Ask and Confirm each of the attributes, *Provide-Car_Rental*
Hotel_Booking	*City*, *Country*, *Hotel_Name*, *Hotel_Category*, *Check_in_Date*, *Check_out_Date*, *Number_Rooms*, *Number_ People*	Ask and Confirm each of the attributes, *Provide-Hotel_Booking*
Tourist-Attractions	*Country*, *City*	Ask and Confirm each of the attributes, *Provide-Tourist-Attractions*
Theatre_Listings	*Country*, *Category*, *Show*, *Theatre*, *Date*, *Hour*	Ask and Confirm each of the attributes, *Provide-Theatre_Listings*
Film_Showtimes	*Country*, *Category*, *Film*, *Cinema*, *Date*, *Hour*	Ask and Confirm each of the attributes, *Provide-Film_Showtimes*

formation available for the required city. Users can optionally provide an origin city to try to obtain detailed information taking into account this origin. Context information taken into account to adapt this information includes user's current position, and preferred means of transport and city.

The *Flight_Schedules* functionality provides flight information considering the user's requirements. Users can provide the origin and destination cities, ticket class, departure and/or arrival dates, and departure and/or arrival hours. Using *Weather_ Forecast* it is possible to obtain the forecast for the required city and dates (for a maximum of 5 days from the current date). For both functionalities, this information is dynamically extracted from external webpages. Context information taken into account includes user's current location, preferred dates and/or hours, and preferred ticket class.

The *Car_Rental* functionality provides this information taking into account users' requisites including the city, pick-up and drop-off date, car type, name of the company, driver's age, and office. The provided information is dynamically extracted from different webpages. The *Hotel_Booking* functionality provides hotels which fulfil the user's requirements (city, name, category, check-in and check-out dates, number of rooms, and number of people).

The *Tourist-Attractions* functionality provides information about places of interest for a specific city, which is directly extracted from the webpage designed for the application. This information is mainly based on users recommendations that have been incorporated in this webpage. The *Theatre_Listings* and *Film_Showtimes* respectively provide information about theatre performances and film showtimes that takes into account the users requirements. These requirements can include the city, name of the theatre/cinema, name of the show/film, category, date, and hour. This information is also considered to adapt both functionalities and then provide context-aware information.

An example of the semantic interpretation of a user utterance using the list of dialogue acts described in Table 1 is shown in Fig. 5.

The DR defined for the task is a sequence of 57 fields, corresponding to:

- The eight concepts defined for the dialogue act representation (*City-Approaches*, *Flight-Schedules*, *Weather-Forecast*, *Car-Rental*, and *Hotel-Booking*, *Tourist-Attractions*, *Theatre-Listings*, and *Film-Showtimes*).
- A total of 45 possible attributes for the concepts (as defined in Table 1).
- The three task-independent concepts that users can provide (*Acceptance*, *Rejection* and *Not-Understood*).
- A reference to the user profile.

A set of 150 scenarios were manually defined to cover the different queries to the system including different user requirements and profiles. Basic scenarios defined only one objective for the dialogue; i.e. the user aims at obtaining information about only one type of the possible queries to the system (e.g., to obtain

Input sentence:
[SPANISH] *Sí, me gustaría conocer los accesos en coche y los hoteles de cuatro estrellas disponibles en Valencia para mañana.*
[ENGLISH] *Yes, I would like to know how to get to Valencia by car and which four stars hotels are available for tomorrow.*

Semantic interpretation:
(*Affirmation*)
(*City_Approaches*)
  City: Valencia
  Means_Transport: Car
(*Hotel_Booking*)
  City: Valencia
  Hotel_Booking: Car
  Category: Four Stars
  Check_in_Date: Tomorrow

Fig. 5. An example of the labeling of a user turn in the travel-planning system.

flight schedules from an origin city to a destination for a specific date). More complex scenarios included more than one objective for the dialogue (e.g., to obtain information about how to get to a specific city, as well as car rental and hotel booking information). An example of the latter scenarios is as follows:

```
User name: Antonio L\'{o}pez
Location: Barajas Airport
Date and Time: 2011-10-27, 9:00am
Device: PDAQ 00-18-41-32-0B-59
Objective: Hotel booking
and Weather Forecast for Granada
```

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 dialogue system to personalize the provided service. In the previous example, the information defined for the user profile is the following:

```
Name: Antonio L\'{o}pez
-------------------
Gender: Male | Age: 53 |
Language: Spanish | Skill level:
High | Pathologies: None...
-------------------
Preferences: Flight schedules, Granada,
business, weekend, ...
-------------------
Current_Location: Barajas Airport |
Check-in zone
```

Figure 6 shows a dialogue for the travel-planning domain acquired without incorporating context information, and a dialogue of the same scenario acquired using our approach. Turns 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 dialogue starts all the values in the dialogue register are initialized to “0”. The information provided by the Context Manager at the beginning of the dialogue and the information provided by the users in each dialogue turn is employed to update the previous DR and obtain the current one, as can be observed in Fig. 7. The example shows the semantic interpretation of a user’s utterance provided by the NLU module.

Confidence scores also provided by this module to determine the reliability are shown between brackets. In the example, the confidence score assigned to the attribute *Destination_City* is very low. Thus, a “2” value is assigned to the corresponding position of the DR_1. The concept *Flight_Schedules* and the attribute *Departure_Date* are recognized with a high confidence score, assigning a “1” value in the corresponding positions of the DR_1. The attributes *Origin_City* and *Ticket_Class* (marked in yellow) are provided by the Context Manager.

As the input to the MLP is generated using DR_1, along with the codification of the labeling of the last system turn (S_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, and a 0.00 probability to the rest of 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. Evaluation of our proposal for the travel-planning system

It is very difficult to define new procedures and measures unanimously accepted by the scientific community for the evaluation of voice-based systems. In [47] and [44], the authors propose a set of statistical mea-

S1: Welcome to the travel-planning system. How can I help you?	*No prompts personalization.*
U1: I want to know flight schedules to Granada.	
S2: From which airport do you want to leave?	*There is not information about the current location of the user.*
U2: Barajas in Madrid.	
S3: Please, tell me the date.	*No date stored.*
U3: Today.	
S4: There are two flights leaving from Madrid to Granada at 10:00am and 19:00pm. 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 235.5 euros. Anything else?	
U6: I want to know car rental information for Granada.	
S7: Please, tell me the office.	*No information about the user preferences: preferred office.*
U7: Europcar at Granada Renfe Train Station.	
S8: Please, tell me the category of the car.	*No information about the user preferences: preferred car type.*
U8: Small.	
S9: Please, tell me your age.	*No information about the user age.*
U9: 53.	
S10: There is a nissan micra for 34 euros a day. Anything else?	
U10: No, thank you.	
S11: Thank you for using the travel-planning system. Have a good trip!	

S1: Good morning Antonio, I check flight schedules leaving from Barajas Airport. Do you want timetables information?	*Prompt personalization including the user's name. The user location is known and his preference about consulting flight schedules.*
U1: Yes, I want to know timetables to Granada.	
S2: There are two flights from Madrid to Granada today, at 10:00am and 19:00pm. 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 dialogues.*
U2: Please, tell me the price of the first one.	
S3: The price in business class is 235.5 euros. Anything else?	*The system uses the ticket-class preferences included in the user profile.*
U3: I want to know car rental information for Granada.	
S4: There is a nissan micra for 34 euros a day at Europcar Granada Renfe Train Station. Anything else?	*The system uses the information about the user preferences (preferred office and car type) and age.*
U4: No, thank you.	
S5: Thank you Antonio. Have a good trip!	*Prompt personalization including the user's name.*

Fig. 6. An example of a dialogue for the travel-planning domain using the context-unaware system (above) or the context-aware system (below).

sures to evaluate the quality of a simulated corpus. Three dimensions are defined: high-level features (dialogue and turn lengths), dialogue style (speech-act frequency, proportion of goal-directed actions, grounding, formalities, and unrecognized actions, proportion of information provided, reprovided, requested and rerequested), and dialogue efficiency (goal completion rates and times). Additionally, the simulation presented in [2,43,45] is evaluated by testing the similarity between real and simulated data by means of statistical measures (dialogue length, task completion rate and dialogue performance).

We have adapted the previously described measures considering the information that is available in the definition of the dialogue system and the context information included in the user profile. Our proposed measures can be classified into three groups: task success/efficiency measures, high-level dialogue features, and dialogue style/cooperativeness measures. These measures evaluate the overall quality of the acquired dialogues and provided services as a whole. In addition, we have carried out a specific evaluation of the operation of our user modelling and dialogue management methodologies.

Table 2

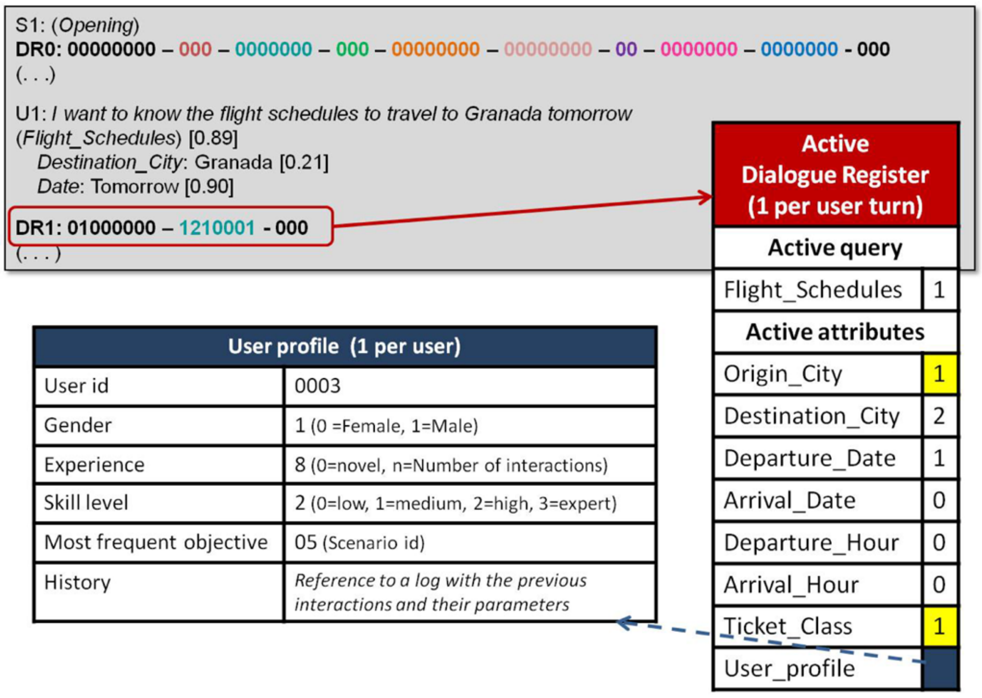
Goal completion rates for the simulation process (percentage of successful dialogues with respect to the total amount of automatically generated dialogues) and completion times (in dialogue turns)

	Percentage completed	Completion time
Context-unaware	19.35%	11.16
Context-aware	34.46%	6.63

6.1. Evaluation results of the user simulator

Firstly, we evaluated the user simulator developed. We were interested in assessing its efficiency and the dialogue success rate. Table 2 shows the goal achievement rates and goal completion times for the context-aware and context-unaware systems. The first important advantage of our proposal is that the percentage of dialogues in which the system successfully provided the complete list of services predefined for the corresponding scenario was higher.

While with the context-unaware system only 19.35% of the dialogues generated were successful, this percentage increases to 34.46% for the context-aware system. This is mainly due to the valuable context information that is directly provided to the system with-


S1: *(Opening)*
DR0: 00000000 – 000 – 0000000 – 000 – 00000000 – 00000000 – 00 – 0000000 – 0000000 - 000
(. . .)

U1: *I want to know the flight schedules to travel to Granada tomorrow*
(*Flight_Schedules*) [0.89]
Destination_City: Granada [0.21]
Date: Tomorrow [0.90]

DR1: 01000000 – 1210001 - 000
(. . .)

User profile (1 per user)	
User id	0003
Gender	1 (0 =Female, 1=Male)
Experience	8 (0=novel, n=Number of interactions)
Skill level	2 (0=low, 1=medium, 2=high, 3=expert)
Most frequent objective	05 (Scenario id)
History	*Reference to a log with the previous interactions and their parameters*

Active Dialogue Register (1 per user turn)	
Active query	
Flight_Schedules	1
Active attributes	
Origin_City	1
Destination_City	2
Departure_Date	1
Arrival_Date	0
Departure_Hour	0
Arrival_Hour	0
Ticket_Class	1
User_profile	

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

out 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 dialogues acquired with the context-aware system, recognition errors are less probable. Therefore, the context-aware user simulator provides more successful dialogues, which is a very important implication for the area of automatic generation of dialogue corpora.

Our analysis shows that not only the dialogues of the context-aware system achieve their goals more frequently, but also their average completion time is shorter. As Table 2 shows, the average duration expressed in number of turns for the dialogues acquired using the context-unaware system is 11.16 turns. This duration decreases to 6.63 turns using the context-aware system.

We have also evaluated the percentage of dialogue objectives that are successfully provided regardless of the success of the dialogue as a whole (the dialogue is only considered successful when it provides the complete information that was required). This percentage is 50.61% for the context-unaware system and 76.85% for the context-aware dialogues. The reason for this difference is that context 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 amount of information that the user must convey, but also the possible confirmations of these values.

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

6.2. *Evaluation results for the dialogue management methodology*

From our previous work on statistical dialogue management [12], we propose four measures to evaluate the performance of the dialogue manager. The first measure, which we call *Unseen Situations*, makes reference to the percentage of dialogue situations that are present in the test partition but are not present in the corpus used to train the dialogue manager. The other three measures are calculated by comparing the answer automatically generated by the dialogue 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 mea-

Table 3

Results of the evaluation of the dialogue management methodology

	Unseen situations	Exact responses	Correct responses	Erroneous responses
Context-unaware	26.43%	84.34%	93.89%	6.11%
Context-aware	15.23%	93.56%	97.68%	2.32%

sures are: i) *Exact Responses*: the percentage of answers provided by the dialogue manager that are equal to the reference answer in the corresponding turn of the training corpus; ii) *Correct Responses*: the percentage of answers provided by the dialogue manager that are coherent with the current state of the dialogue although they are not identical to the reference answer; iii) *Erroneous Responses*: the percentage of answers provided by the dialogue manager that would cause the failure of the dialogue.

Two dialogue managers were developed using respectively each one of the two corpora of 1000 dialogues 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 3 shows the results of the evaluation. As can be observed, the number of unseen situations (not present in the training corpus) is reduced in the context-aware system and the variability of the different dialogues is reduced, as the number of turns is also lower, as shown in the dialogue examples presented in Fig. 6). This is the main cause of obtaining better results for the rest of measures in the evaluation of the dialogue manager developed for the context-aware system.

The results of the *%exact* and *%correct* measures show the satisfactory operation of the developed dialogue manager for both systems. The answer generated by the manager matches the reference answer for the corresponding turn by a percentage of 84.34% and 93.56% for the context-unaware and context-aware systems respectively.

Finally, the percentage of answers generated by the MLP that can cause the failure of the system is reduced from 6.11% for the context-unaware system to only 2.32% for the context-aware system. An answer that is coherent with the current state of the dialogue is generated in 97.68% of cases for this system. These last two results also demonstrate the correct operation of the classification methodology developed to deal with the complete set of possible situations during a dialogue.

6.3. Evaluation results of the high-level dialogue features

By means of high-level dialogue features, we evaluate the duration of the dialogues, how much information is transmitted in individual turns, and how active the dialogue participants are. These dialogue features cover the following statistical properties: i) Dialogue length, measured as the mean and shape of the distribution of the number of turns per task, the number of turns of the shortest dialogue, the number of turns of the longest dialogue, and the number of turns of the most frequent dialogue; ii) Percentage of different dialogues in each corpus and the number of repetitions of the most frequent dialogue; iii) Turn length, measured by the number of actions per turn; and iv) Participant activity as a ratio of system and user actions per dialogue. Table 4 shows the comparison of the different high-level measures for the context-aware and context-unaware systems.

As it can be observed, there is also a reduction in the number of turns of the longest, shortest and most frequent dialogues for the context-aware system. The number of different dialogues is also lower using the context-aware system, due to the reduction in the number of turns, as it can be observed in the number of repetitions of the most frequent dialogue. 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 of the dialogues acquired with or without context information (Fig. 8) shows that the context-unaware dialogues have the largest standard deviation given that the task length of these dialogues is more disperse. Context-aware dialogues have minimum deviation since the successful dialogues are usually those which require the minimum number of turns to achieve the objective(s) predefined for the different scenarios.

Table 4

Results of the high-level dialogue features defined for the comparison of the two kinds of dialogues for the travel-planning information system

	Context-unaware	Context-aware
Percentage of different dialogues	85.54%	71.69%
Number of repetitions of the most frequent dialogue	4	9
Number of turns of the most frequent dialogue	8	6
Number of turns of the shortest dialogue	6	6
Number of turns of the longest dialogue	26	18

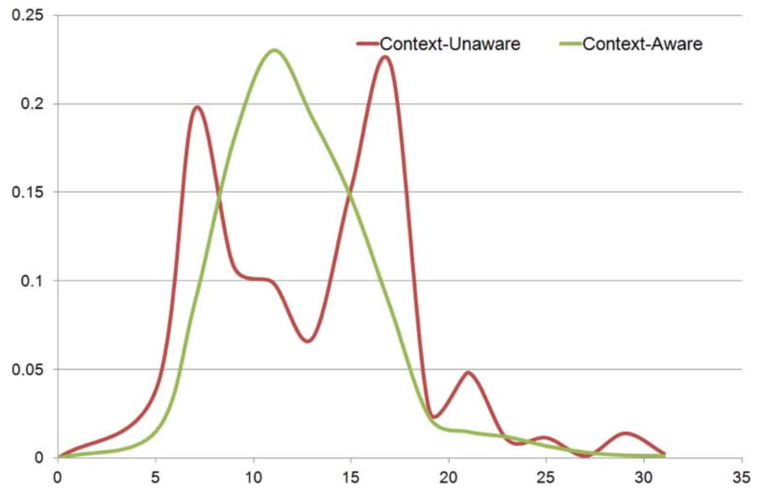


Fig. 8. Task length distribution.

Regarding the dialogue participant activity, context-aware dialogues have a higher proportion of system actions, as less information requires confirmation in the context-aware system. There is also a slight reduction in the mean values of the turn length for the context-aware dialogues. These dialogues are statistically shorter, as they provide 1.25 actions per user turn instead of the 1.49 actions provided by the context-unaware dialogues. This is again because the users have to explicitly provide more information in the context-unaware system.

6.4. *Evaluation results for the dialogue style and cooperativeness*

Dialogue style and cooperativeness measures analyse the frequency of different speech acts and reflect the proportion of actions that are goal-directed (i.e. not indexed in dialogue formalities). For dialogue style features, we defined and counted a set of system/user dialogue acts. On the system side, we measured the confirmation of concepts and attributes, questions to require information, and system answers generated after a database query. On the user side, we 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 experiments described in this subsection cover the following statistical properties: frequency of different user and system actions (dialogue acts), and proportion of goal-directed actions (request and provide information) versus grounding actions (confirmations) and rest of actions.

The histograms in Figs 9 and 10 respectively show the frequency of the most dominant user and system dialogue acts in the context-aware and context-unaware dialogues. 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* dia-

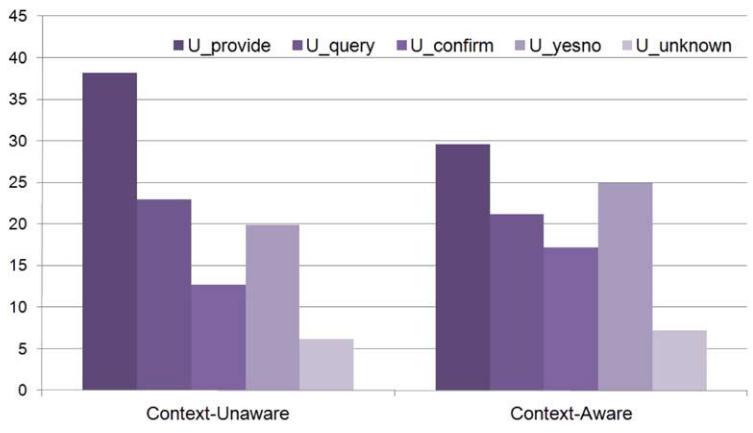


Fig. 9. Histogram of user dialogue acts.

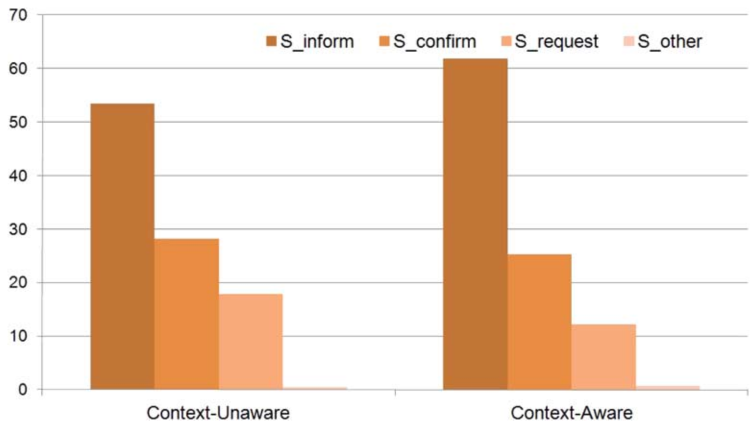


Fig. 10. Histogram of system dialogue acts.

logue 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, dialogue formalities or out of task information).

In both cases, it can be observed that there are significant differences in the distribution of dialogue acts. On the one hand, 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. It can also be observed a higher proportion of yes/no actions for the context-aware dialogues. 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 a 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 11 shows a comparison between these categories. As can be observed, the dialogues 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 behaviour of our system with real users using the same set of scenarios designed for the user simulation. A total of 150 dialogues were recorded from interactions of six users employing the context-aware and context-unaware systems. The evaluation was carried out by students and lectur-

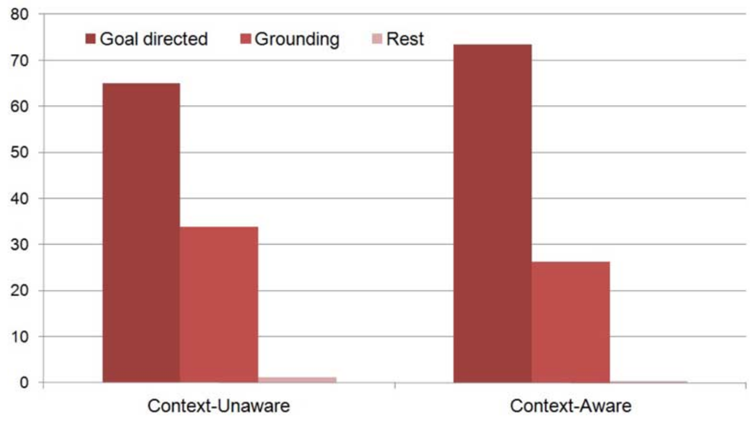


Fig. 11. Proportions of dialogue spent on-goal directed actions, ground actions and the rest of possible actions.

Table 5
Results of the objective evaluation of the context-aware and context-unaware systems with real users

	Success rate	nT	Confirmation rate	%ECR	nCE	nNCE
Context-unaware	82%	15.6	29%	78%	0.82	0.23
Context-aware	94%	8.4	26%	87%	0.91	0.14

ers 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. Dialogue success rate. This is the percentage of successfully completed tasks. In each scenario, the user has to obtain one or several items of information, and the dialogue success depends on whether the system provides correct data (according to the aims of the scenario) or incorrect data to the user.
2. Average number of turns per dialogue (nT).
3. Confirmation rate. It was computed as the ratio between the number of explicit confirmations turns (nCT) and the number of turns in the dialogue (nCT/nT).
4. Average number of corrected errors per dialogue (nCE). The average of errors detected and corrected by the dialogue manager. We have considered only those which modify the values of the attributes and thus could cause the failure of the dialogue. The errors are detected using the confidence scores provided by the ASR and NLU modules. Implicit and explicit confirmations are employed to confirm or require again values detected with low reliability.
5. Average number of uncorrected errors per dialogue (nNCE). This is the average of errors not corrected by the dialogue 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 nCE/ (nCE + nNCE).

The results presented in Table 5 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 12% absolute. Using the context-aware system, the average number of required turns is also reduced from 15.6 to 8.4. These values are slightly higher for both systems as in some dialogues 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, given that less information is required to the user, reducing the probability of introducing ASR errors. The main problem detected was related to user inputs misrecognized with a very high ASR confidence, and this erroneous information was forwarded to the dialogue manager.

Table 6
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	4.1	4.8	3.9	3.6	3.2
Context-aware	4.3	4.7	4.6	4.5	3.5

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 6 shows the average results of the subjective evaluation.

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 fulfil the complete set of objectives of the scenario and the suitability of the interaction rate during the dialogue.

7. Conclusions

Context-aware systems along with mobile devices offer new opportunities in the areas of knowledge representation, natural language processing and intelligent information retrieval from the web. In this paper, we have combined different aspects from these important research areas to provide context aware adaptable web information and services by means of speech interaction, which facilitates a personalized and more natural access to information. In order to do this, we have contributed a framework which can be used to develop context-aware spoken dialogue systems that can be easily integrated in hand-held mobile devices. The framework is comprised of an architecture in which different systems and modules cooperate to provide adapted services, and a representation mode for knowledge sharing between the components of the architecture.

Within our framework, the interaction is managed dynamically using a classification process to select the best system answer by taking into account the previous history of the dialogue. We have adapted this methodology to develop a context-aware dialogue manager that uses 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 a context manager included in the proposed architecture. To store and share context information we have defined a data structure that manages user profiles. These profiles include not only information external to the users such as their location, but also information about their preferences and needs, some of them automatically extracted from previous interactions.

We have provided a complete implementation of our architecture in a system that provides personalized information in a travel-planning information system. To develop this system we have defined the complete requirements for the task and developed the different modules, and the necessary information to be incorporated in the user profile. For its evaluation we have employed a statistical user simulation technique that makes possible to acquire a great number of dialogues and then carry out a detailed evaluation of the quality of both the dialogues and the services. A total of 2000 successful dialogues have been acquired, along with 150 additional dialogues with six real users. From these dialogues we have studied the influence of context information on the quality of the services that are provided by the system.

To compare the dialogues 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 have evaluated the success of the dialogues and services provided as well as their efficiency and variability with regard to the different objectives specified in the set of dialogue scenarios.

The results 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 up to a 50% in several cases. The quality of the interaction between the user and the system is increased, as context-aware dialogues 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.

As future work, we will carry out a detailed study of the user rejections of system-hypothesized actions using the values extracted from the user profile, and study the benefits that could be derived from including additional interaction modalities. We are also interested in applying our proposal to multi-domain tasks in order to measure the capability of our methodology to adapt efficiently to AmI contexts that vary dynamically.

Acknowledgements

Research funded by projects CICYT TIN2011-28620-C02-01, CICYT TEC2011-28626-C02-02, CAM CONTEXTS (S2009/TIC-1485), and DPS2008-07029-C02-02.

References

[1] H. Afsarmanesh, V. Guevara Masís, and L.O. Hertzberger, Virtual comunity support in telecare, in: *Proc. of the 4th IFIP Working Conference on Virtual Enterprises (PRO-VE'03)*, 2003, pp. 211–220.

[2] H. Ai, A. Raux, D. Bohus, M. Eskenazi, and D. Litman, Comparing spoken dialog corpora collected with recruited subjects versus real users, in: *Proc. of the 8th SIGdial Workshop on Discourse and Dialogue*, 2007, pp. 124–131.

[3] D.R. De Almeida, C. De Souza Baptista, E.R. Da Silva, C.E. Campelo, H.F. De Figueiredo, and Y.A Lacerda, A context-aware system based on service-oriented architecture, in: *Proc. of the 20th Int. Conference on Advanced information Networking and Applications (AINA'06)*, 2006, pp. 205–210.

[4] J. Antoniou, C. Christophorou, J. Simoes, and A. Pitsillides, Adaptive network-aided session support in context-aware converged mobile networks, *International Journal of Autonomous and Adaptive Communication Systems* **5**(3) (2012), 1–23, Inderscience.

[5] D. Bohus, S. Grau, D. Huggins-Daines, V. Keri, G. Krishna, R. Kumar, A. Raux, and S. Tomko, Conquest – an open-source dialog system for conferences, in: *Proc. of 7th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, 2007, pp. 9–12.

[6] B. Brown and R. Randell, Building a context sensitive telephone: Some hopes and pitfalls for context sensitive computing, *Computer Supported Cooperative Work* **13** (2004), 329–345, Springer.

[7] A.K. Dey and G.D. Abowd, Towards a better understanding of context and context-awareness, in: *Proc. of the 2000 Conference on Human Factors in Computer Systems (CHI'00)*, 2000, pp. 304–307.

[8] R.E. Fikes and T. Kehler, The role of frame-based representation in knowledge representation and reasoning, in: *Communications of the ACM*, Vol. 28, 1985, pp. 904–920.

[9] P. Filipe and N. Mamede, Ambient intelligence interaction via dialogue systems, in: *Ambient Intelligence*, In-teh, 2010, pp. 109–124.

[10] F. García, L.F. Hurtado, E.Sanchis, and E. Segarra, The incorporation of confidence measures to language understanding, in: *Proc. of TSD'03*, 2003, pp. 165–172.

[11] J. Glass, G. Flammia, D. Goodine, M. Phillips, J. Polifroni, S. Sakai, S. Seneff, and V. Zue, Multilingual spoken-language understanding in the MIT Voyager system, *Speech Communication* **17** (1995), 1–18, Elsevier.

[12] D. Griol, L.F. Hurtado, E. Segarra, and E. Sanchis, A statistical approach to spoken dialog systems design and evaluation, *Speech Communication* **50**(8–9) (2008), 666–682, Elsevier.

[13] D. Griol, Z. Callejas, and R. López-Cózar, Acquiring and evaluating a dialog corpus through a dialog simulation technique, in: *Proc. of the 9th SIGdial Workshop on Discourse and Dialogue*, 2009, pp. 326–332.

[14] D. Griol, N. Sánchez-Pi, J. Carbó, and J.M. Molina, An architecture to provide context-aware services by means of conversational agents, *Advances in Intelligent and Soft Computing*, **79** (2010), 275–282, Springer.

[15] B. Han, W. Jia, J. Shen, and M.C. Yuen, Context-awareness in mobile web services, in: *Proc. of the 2nd Int. Symposium on Parallel and Distributed Processing and Applications (ISPA 2004)*, 2008, pp. 519–528.

[16] K. Henricksen, J. Indulska, and A. Rakotonirainy, Modeling context information in pervasive computing systems, in: *Proc. of the 1st Int. Conference on Pervasive Computing*, 2002, pp. 167–180.

[17] R. Hervás and J. Bravo, Towards the ubiquitous visualization: Adaptive user-interfaces based on the Semantic Web, *Interacting with Computers* **23**(1) (2011), 40–56, Elsevier.

[18] H. Kang, E. Suh, and K. Yoo, Packet-based context aware system to determine information system user's context, *Expert Systems with Applications* **35** (2008), 286–300, Elsevier.

[19] S. Kartakis, A design-and-play approach to accesible user interface development in Ambient Intelligence Environments, *Journal Computers in Industry* **61**(4) (2010), 318–328, Elsevier.

[20] J. Ko, F. Murase, T. Mitamura, E. Nyberg, M. Tateishi, and I. Akahori, Context-aware dialog strategies for multimodal mobile dialog systems, in: *Proc. of AAAI International Workshop on Modeling and Retrieval of Context*, 2006, pp. 7–12.

[21] G.L. Kovács and S. Kopácsi, Some aspects of Ambient Intelligence, *Acta Polytechnica Hungarica* **3**(1) (2006), 35–60, Óbuda University.

[22] T.C. Lech and L.W.M. Wienhofen, 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)*, 2005, pp. 625–631.

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

[24] J. Liu, S. Seneff, and V. Zue, Dialogue-oriented review summary generation for spoken dialogue recommendation systems, in: *Proc. of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, 2010, pp. 64–72.

[25] J. McCarthy, Generality in artificial intelligence, *Communications of the ACM* **30**(12) (1987), 1030–1035, ACM.

[26] J. McCarthy, *Formalization of Common Sense, papers by John McCarthy edited by V. Lifschitz*, Ablex, 1990.

[27] M.F. McTear, *Spoken Dialogue Technology: Towards the Conversational User Interface*, Springer, 2004.

[28] H. Melin, A. Sandell, and M. Ihse, CTT-bank: A speech controlled telephone banking system – an initial evaluation, in: *TMH Quarterly Progress and Status Report (TMH-QPSR)*, Vol. 1, 2001, pp. 1–27.

[29] W. Minker, R. López-Cózar, and M.F. McTear, The role of spoken language dialogue interaction in Intelligent Environments, *Journal of Ambient Intelligence and Smart Environments* **1**(1) (2009), 31–36, IOS Press.

[30] M. Minsky, A framework for representing knowledge, in: *The Psychology of Computer Vision*, McGraw-Hill, 1975, pp. 211–277.

[31] H. Naguib, G. Coulouris, and S. Mitchell, Middleware support for context-aware multimedia applications, in: *Proc. of the 3rd Int. Working Conference on New Developments in Distributed Applications and Interoperable Systems*, 2001, pp. 9–22.

[32] I. Nieto-Carvajal, J.A. Botía, P.M. Ruiz, and A.F. Gómez-Skarmeta, Implementation and evaluation of a location-aware wireless multi-agent system, in: *Proc. of the Int. Conference on Embedded and Ubiquitous Computing (EUC'04)*, 2004, pp. 528–537.

[33] K. Nihei, Context sharing platform, *NEC Journal of Advanced Technology* **1**(3) (2004), 200–204, National Library of Australia.

[34] P.O. Osland, B. Viken, F. Solsvik, G. Nygreen, J. Wedvik, and S.E. Myklbust, Enabling context-aware applications, in: *Proc. of the Int. Conference on Convergence in Services, Media and Networks (ICIN'06)*, 2006, pp. 1–6.

[35] T. Paek and R. Pieraccini, Automating spoken dialogue management design using machine learning: An industry perspective, *Speech Communication* **50** (2008), 716–729, Elsevier.

[36] R. Pieraccini and D. Lubensky, Spoken language communication with machines: The long and winding road from research to business, *LNAI* **3533** (2005), 6–15, Springer.

[37] G.N. Prezerakos, N.D. Tselikas, and G. Cortese, Model-driven composition of context-aware web services using ContextUML and aspects, in: *Proc. of IEEE Int. Conference on Web Services (ICWS'07)*, 2007, pp. 320–329.

[38] F. Ramparany, R. Poortinga, M. Stikic, J. Schmalenstroer, and T. Prante, An open context information management infrastructure – the IST-Amigo project, in: *Proc. of 3rd IET International Conference on Intelligent Environments (IE'07)*, 2007, pp. 398–403.

[39] J. Rouillard, Web services and speech-based applications around VoiceXML, *Journal of Networks* **2**(1) (2007), 27–35, Academy Publisher.

[40] N. Roy, J. Pineau, and S. Thrun, Spoken dialogue management using probabilistic reasoning, in: *Proc. of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, 2000, pp. 93–100.

[41] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning internal representations by error propagation, in: *PDP: Computational Models of Cognition and Perception, I*, MIT Press, 1986, pp. 319–362.

[42] N. Sadeh, F. Gandon, and O.B. Kwon, Ambient Intelligence: The MyCampus Experience, Technical report, School of Computer Science, Carnegie Mellon University, 2005.

[43] J. Schatzmann, K. Georgila, and S. Young, Quantitative evaluation of user simulation techniques for spoken dialogue systems, in: *Proc. of the 6th SIGdial Workshop on Discourse and Dialogue*, 2005, pp. 45–54.

[44] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies, *Knowledge Engineering Review* **21**(2) (2006), 97–126.

[45] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, Agenda-based user simulation for bootstrapping a POMDP dialogue system, in: *Proc. of Human Language Technologies HLT/NAACL'07 Conference*, 2007, pp. 149–152.

[46] J. Schatzmann, B. Thomson, and S. Young, Error simulation for training statistical dialogue systems, in: *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU'07)*, 2007, pp. 526–531.

[47] K. Scheffler and S. Young, 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*, 2001, pp. 64–70.

[48] B.N. Schilit, N.I. Adams, and R. Want, Context-aware computing applications, in: *Proc. of the Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85–90.

[49] S. Seneff, M. Adler, J.R. Glass, B. Sherry, T.J. Hazen, C. Wang, and T. Wu, Exploiting context information in spoken dialogue interaction with mobile devices, in: *Proc. of Int. Workshop on Improved Mobile User Experience (IMUx'07)*, 2007, pp. 1–11.

[50] A. Stolcke, N. Coccaro, R. Bates, Paul P. Taylor, C. Van Ess-Dykema, K.K. Ries, E. Shriberg, D. Jurafsky, R. Martin, and M. Meteer, Dialogue act modeling for automatic tagging and recognition of conversational speech, *Computational Linguistics* **26**(3) (2000), 339–373, MIT Press.

[51] P. Strauss and W. Minker, *Proactive Spoken Dialogue Interaction in Multi-Party Environments*, Springer, 2010.

[52] C.A. Thompson, M.H. Göker, and P. Langley, A personalized system for conversational recommendations, *Journal of Artificial Intelligence Research* **21** (2004), 393–428, AAAI Press.

[53] H.L. Truong and S. Dustdar, A survey on context-aware web service systems, *Int. Journal of Web Information Systems* **5**(1) (2009), 5–31, Emerald.

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

[55] S. Young, J. Schatzmann, K. Weilhammer, and H. Ye, The hidden information state approach to dialogue management, in: *Proc. of 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007, pp. 149–152.

[56] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington, JUPITER: A telephone-based conversational interface for weather information, *IEEE Transactions on Speech and Audio Processing* **8**(1) (2000), 85–96.