VOICE: a Framework for Speech-Based Mobile Systems

Adam Sharp, Stan Kurkovsky
Department of Computer Science
Central Connecticut State University
New Britain, CT, USA
{sharp_adp, kurkovskysta}@ccsu.edu

Abstract

The advent of mobile phones and the Internet opened the doors for an emerging class of applications that connect mobile users to online resources and information services available anytime and anywhere. VoiceXML is an enabling technology for creating a streamlined speech-based interface for web-based information services. In this paper, we describe VOICE, a framework used to develop mobile context-aware speech-enabled systems powered by VoiceXML. Restaurant Search Guide, a simple context-aware recommender system has been built using VOICE to test the feasibility of the proposed framework.

1. Introduction

Two technological innovations introduced within the last two decades, have proved to provide a very significant impact on our everyday lives and habits. The Internet provides new means and ways to access the vast amounts of information and an exploding number of online services available today. It has revolutionized the way people communicate with each other, how they receive news, shop, and conduct other day-to-day activities. Mobile telephones provide users with a simple anytime and anywhere communication tool. Originally designed for voice communication, today mobile phones are capable of connecting their users to a wide variety of Internet-enabled services and applications, which can vary from a simplified web browser to a GPS-enabled navigation system. Experts agree that a combination of these two innovations (online Internet-enabled services accessible via a mobile phone) should have a revolutionary impact on mobile commerce [1, 2, 4, 8, 17]. Current research focuses mostly on applications designed for smart phones, in which the application logic is usually placed within the mobile device. Speech, however, remains the most basic and most efficient form of communication, and providing a form of speech-based communication remains the prevalent function of any mobile phone. Current trends in customer acceptance of these technologies indicate that the voice-based applications connecting mobile users to Interned-enabled information services are positioned for a continued growth [13].

This paper presents VOICE, the Voice Operated Interchangeable Control Environment. The objective of VOICE is to create a highly customizable platform for developing mobile context-aware applications with a voice interface. VOICE-based applications can be accessed through any telephone, as no portion of their logic resides within the mobile device. Context-aware features of VOICE are achieved by analyzing the patterns of requests made by the same user and by matching the similarities among these patterns across a group of users. To test the viability of VOICE, we designed and implemented a voice-operated Restaurant Search Guide, a simple recommender system allowing its users to call in, specify any number of search parameters, and receive suggestions based on the search criteria specified by the user, as well as on the previously gathered user and location context.

The paper is organized as follows. Section 2 briefly introduces VoiceXML, describes related work in the area of recommender systems, and highlights some existing research and industry projects. Section 3 describes a hypothetical scenario of using the Restaurant Search Guide, while Section 4 details the architecture of VOICE. Section 5 concludes the paper with a summary and directions of the future work.

2. Related work

This section briefly describes some of the research topics, technologies, existing projects and applications closely related to VOICE.

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

2.1. VoiceXML

VoiceXML was introduced by the World Wide Web Consortium as a domain-specific application of XML describing a voice-based interface between the human users and computer systems. VoiceXML applications interact with users by playing audio fragments and converting text to speech; they can accept user input by recognizing or recording their voice input and accepting touch-tone input. VoiceXML services appeal to the developers because they isolate the application from many complexities, which include provisioning of resources, concurrent threads of control, and platform-specific APIs [4].

A very large number of commercial VoiceXML applications have been deployed; these include customer relationship management, refilling of medical prescriptions, providing driving directions, flight tracking, and many others. VoiceXML services play a central role in enabling the speech-based interface and many other functionalities of VOICE.

2.2. Recommender systems

Recommender systems suggest services or products to their users; thus, they play a very important role in many e-commerce services. Typically, items can be recommended based on their overall popularity, on the demographics or other context information about the user, or as a prediction of the future behavior of the user based on his/her past actions.

Techniques employed by recommender systems can be broadly divided into two categories: information filtering (IF) and collaborative filtering (CF). IF methods usually require having a user profile indicating the user's preferences or needs; these profiles can be either entered by the user or created by the system itself by observing the user's actions. Such profiles or user context information can be effective when the system needs to predict user's interest to a previously unseen or unrated item. IF mechanisms are relatively easy to implement and they are not dependent on the number of users in the systems; on the other hand, the quality of recommendations greatly depends on the quality of information about the content items.

CF techniques, another mechanism used by recommender systems, require a database of user opinions about each of the content items. The system can gather these opinions by directly asking the users to rate the content items or by observing the user's actions, such as viewing a content item description, placing a product in the shopping cart or in the wish list, or purchasing a product. CF mechanisms make recommendations by identifying groups of users with similar opinions on a particular content item and then making a prediction by combining opinions within that group of users. CF does not require a special high-quality source of information about each of the items (it is replaced by the database of the user opinions); on the other hand, the quality of recommendations made by CF techniques dramatically increases with the number of user opinions about each of the content items. CF systems naturally suffer from the scarcity and cold-start problems stemming from a small number of users or a low number of opinions about a specific content item, which may result in the inability to make recommendations [14].

Recommender systems are a good fit in the domain of convenient and ubiquitous mobile devices. Mobile recommender systems [10, 21] differ from their full-scale counterparts primarily due to the restrictions imposed on them by the mobile devices on which they run. Smart phones and PDAs have screens that are often too small to display the information needed to convey all details of a specific recommendation. These features of mobile devices necessitate a compromise between the accuracy of recommendation and simplicity of the interface of a mobile recommender system. Furthermore, current infrastructure of the mobile communications industry makes it very difficult for anyone other than the network carrier (such as Verizon or Sprint) or a large information service provider (such as AOL or MSN) to supply and distribute the application software needed to run and use a particular mobile software system on a large scale. Mobile systems with speech-based interface, however, have no special distribution requirements and can be accessed by a large range of users of different classes of mobile devices.

VOICE is well suited for developing mobile recommender systems with speech interfaces. The Restaurant Search Guide developed in the VOICE framework includes a simple recommender component.

2.3. Speech-enabled and context-aware applications

Dey [5] defines context as "any information that can be used to characterize the situation of an entity." Here, an entity can represent a user, location or any object which is relevant to the system/user interaction. In practice, context-aware applications use context information to deliver services whose relevancy depends on the current task of the user. Application context may refer to current location, time, user identity, as well as other users and objects [6, 7].

Current research confirms an intuitive suggestion that a speech-based interface improves the end-user

acceptance of mobile applications, both in the area of mobile commerce and in the area of mobile information access [2, 8]. In recent years, a large number of applications have been developed to take advantage of VoiceXML-enabled speech interfaces that are readily available through any telephone connection. Below is a brief summary of an exemplary selection of some of these applications.

Careflow's Dictation application system [3] provides physicians with a speech-based interface to a service for transcription of patient records. The system's middleware generates VoiceXML dynamically through an interface with the enterprise-level CORBA services. Together with other medical transcription and healthcare documentation management services, this system has been deployed to a number of healthcare providers.

Deployed at several hospitals, Vocera [19] provides a single communications solution for mobile workers, such as nurses. It operates via a specially designed small WiFi-capable device that has a hands-free speech interface and can be worn as a badge. System servers provide voice recognition and enterprise functionality, such as access to directories, paging, and voice mail.

VoiceLETS (Voice Law Enforcement Tactical System) [9] allows police officers in Alabama to make routine queries and returns search results without a human dispatcher. Queries handled by VoiceLETS deal primarily with driver and vehicle information. VoiceLETS is driven by VoiceXML; all of its data comes from the databases run by the motor vehicle and corrections departments in Alabama. The viability of such a system is undeniable, taking into account the fact that police officers are free to use their hands and do not have to look at a communication device while dealing with a potentially dangerous situation.

Several multi-modal web browsers [11, 20], along with a number of systems for transcoding HTML content into audio via VoiceXML [12, 16] have been developed in recent year. Such systems hold a great promise for people with visual impairments; however, while such systems are technologically possible, their real world usability is currently very limited.

At the time of this writing, the authors of this paper could not identify any existing projects aimed at developing VoiceXML-enabled context-aware systems, which would be similar to VOICE in their functionality. However, many existing systems [3, 9, 19] employ architectural solutions that overall are similar to VOICE: they integrate back-end speech recognition services (VoiceXML or proprietary) with enterprise databases and provide anytime access over a tether-less infrastructure (mobile phone or WiFi).

3. A scenario of using the VOICE system

John is a newcomer to the town of Enfield, CT, and therefore knows very little about the area. It is coming up on dinner time; fortunately John knows of a restaurant guide available through his mobile phone. As John connects to the Restaurant Search Guide driven by VOICE, the system automatically checks the history of prior calls and determines that John has not previously used the system. The VOICE system has no user context information for John and it cannot offer any personalized features; thus, VOICE automatically extends all of the spoken prompts to make John's first call an easy experience. After checking its database of previous sessions that contains the user context information, and determining that John is a new user, VOICE responds to John with a greeting as shown in the first system prompt in Figure 1.

System:	*Welcome to the Restaurant Search Guide […]. Say 'Zip Code' to select the zip code in which to search. Say 'Type' to select the restaurant type. Say 'Cuisine' to select the cuisine that you desire. If you need further assistance at any time, you may say 'Help,' and if you ever want to start over, say 'Start Over'.*
User:	*Zip Code*
System:	*Please say or type your zip code.*
User:	*06082*
System:	*You have entered 06082. Please select what to complete next: Say 'Type' to select the restaurant type, say 'Cuisine' to select the cuisine that you desire, or say 'Start Over' to start from the beginning.*
User:	*Type*
System:	*Would you like an eat-in or take-out restaurant?*
User:	*Take out*
System:	*You have selected take out. Please select what to complete next: Say 'Cuisine' to select the cuisine that you desire, or say 'Start Over' to start from the beginning.*
User:	*Cuisine*
System:	*What type of cuisine would you like?*
User:	*Chinese*
System:	*You have selected Chinese. Please select from the following restaurants in or around 06082:* *1) Cheng's Palace* *...*
User:	*One*
System:	*Cheng's Palace. 123 Main Street, Enfield, Connecticut, 06082. Phone number: 860-555-1234. Please say 'repeat' to have this information repeated to you. Would you like to get directions to this restaurant?*
User:	*No*
System:	*Thank you for using the Restaurant Search Guide.*

Figure 1. A typical dialog of a user with the VOICE-based Restaurant Search Guide

A few days later, John decides that he wants to go to dinner at some Italian restaurant in town. After John calls the Restaurant Search Guide once more, VOICE determines that he has called in before, retrieves his context information, and automatically sets his zip code to 06082, responding to John with "Welcome back to the Restaurant Search Guide. Are you calling for restaurants in 06082?" to which John responds in the affirmative. After selecting an eat-in Italian restaurant, John opts to have the system give him the

directions from his current location to the restaurant (and possibly repeat a few times, just so he can make sure that he copies down the directions exactly).

Over time, John makes multiple calls to the Restaurant Search Guide, where the VOICE system keeps track of the user context information, such as his preferences for zip code, cuisine, and restaurant type. After some time, the system may note that John eats a lot of Chinese take-out and begin making suggestions for this type of restaurants in his area. If John stops asking for Italian restaurants in the area, the system will 'forget' that John likes this type of food, and will not automatically suggest Italian restaurants.

4. Architecture of the VOICE system

The VOICE system has a multi-tiered architecture that is distributed across multiple servers, as shown in Figure 2. The first tier, VoiceXML service provider, contains a VoiceXML processor and any applicable audio files (in our example, we used the service provided by BeVocal.com). The VOICE System tier and the VOICE Application tier contain the bulk of the VOICE system and application logic. The VOICE System tier also hosts the Context Information database. The Data Provider tier contains one or more servers providing the Data Source(s) that the Data Source Adapter(s) will query. The Restaurant Search Guide described in the previous section is an application developed within the VOICE framework and will be used throughout this section to illustrate many of the concepts presented herein.

4.1. User Session State / Context Manager

The User Session State / Context Manager (USSCM) controls the states of the current session given its XML Application Schema, determining what states are available to transition to next, as well as maintaining the values of each variable in the current state. The USSCM also queries the Context Information database to retrieve the user context information; it determines if the current caller has used the system before, automatically filling out global variables which indicate whether they have called and what their most likely inputs will be. The USSCM also creates and passes the list of variables and the internal representation of the menus and prompts to the Flow Control Module, and receives updates to these variables from the Flow Control Module.

As the user interacts with the system, the USSCM may also execute queries on the Context Information database to keep track of the user context, which will be used to determine information in further sessions with the VOICE system. In the Restaurant Search Guide example, the user context information includes John's zip code and restaurant preferences that are stored in the Previous Sessions database each time John selects a restaurant or changes his zip code, even if he decides to change his preferences later.

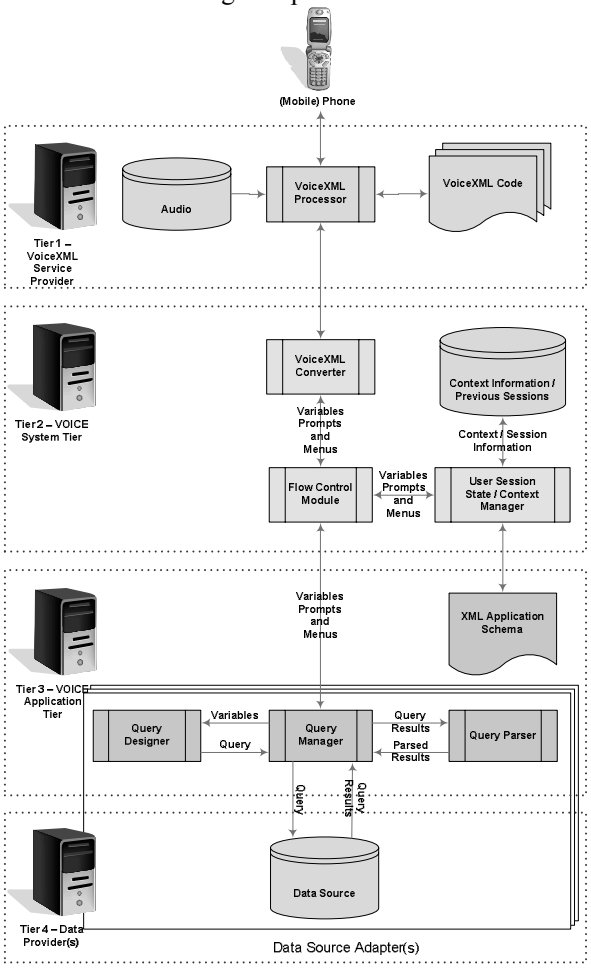

Figure 2. Architecture of the VOICE framework

4.2. Flow Control Module

The Flow Control Module (FCM) is the central part of the VOICE system; it parses the user input and interacts with the USSCM to determine the state of the current session. The FCM also interacts with the Data Source Adapters which have been defined in the XML Application Schema for the current state (shown in Figure 4), as well as sending all internal menu and prompt representations to the VoiceXML Converter. When a user enters information that needs to be stored in a variable, the FCM makes sure that the information is valid (i.e. fits the given type of the variable) and, if necessary, will automatically issue error messages to the VoiceXML converter to be spoken to the user.

The FCM issues all queries for information to the appropriate Data Source Adapter, and then can return the results of those queries to the VoiceXML Converter (if used in a prompt) or fills in some internal variables with the results from the query.

4.3. VoiceXML Converter

The VoiceXML Converter takes all user input interpreted by the VoiceXML processor and converts it into usable internal representations (to be checked for validity by the FCM). It also transforms the internal menus and prompts into valid VoiceXML resulting in audio prompts which will be spoken aloud to the user. After the VoiceXML Converter translates the user input into usable representations, it passes the data to the FCM for further validation and updating, to ensure that the text entered is, indeed, valid.

4.4. Context Information Database

The Context Information / Previous Sessions Database holds all of context information derived from the analysis of data gathered during the previous VOICE sessions, including the calling number, what options were selected, and what information was provided to the VOICE system.

4.5. XML Application Schema

The XML Application Schema is integral to the functionality of VOICE, as it describes each prompt from which the audio spoken to the user is created, each of the variables and variable types that can (or need to) be filled out.

User Session States. At any given moment during the user's interaction with the system, VOICE is said to be in one of the user session states defined in the application schema. A user session state describes the different prompts and available variables for that state along with a list of other states to which the system can transition from this state. In each state, the audio prompt is described, either in a form usable by the VoiceXML text-to-speech translator, or as an audio file which will be played to the user. Each state also contains a list of preconditions that must be met before this state can be entered into. The user session state also describes any applicable Data Source Adapters, which variables to pass through to those adapters, and what to do with the results of the queries that are passed back from the adapters.

Application Variables. The type of an application variable is determined by the common VoiceXML variable types, and can also be given specific "grammars" which allow VOICE to accept a wide range of specialized input. These variables can be either state-specific (those that are not retained and passed through the different VOICE modules), or global variables (every module and state in the system has access to their values). Once a type is set for a given variable, the VOICE system automatically ensures that each user input matches the type of the given application variable, or will prompt the user to enter information that correctly fills the application variable in question.

Available User Session States. A list of one or more user session states that are available to move to once this state has been passed are also included in the description of each user session states, and are referenced by state name. The available states may have preconditions described in VoiceXML notation.

4.6. Data Source Adapter

The Data Source adapter consists of 4 parts: the Query Manager, the Query Designer, the Query Parser, and the Data Source. The data source adapter allows VOICE to connect and query a specific source of information, which could be a local database, an XML Web Service, or an application API. In our Restaurant Search Guide example, there are two Data Source Adapters: one to receive a listing of all available restaurants in a given zip code (using the Google API), and the other to get directions from a caller's location to the restaurant.

4.7. Query Manager

The Query Manager is essentially a middleman; it merely passes information between the FCM and the Query Designer/Parser. The Query Manager receives the list of available variables from FCM, which it then sends to the Query Designer. Once the Query Designer has created a query for use with the Data Source, the Query Manager executes this query and returns the results to the Query Parser. Once these results have been analyzed by the Query Parser, it sends updated variables and information back to the FCM for further use.

4.8. Query Designer

The Query Designer uses the list of available variables given to it by the Query Manager to create a query which will be executed against the Data Source in the appropriate format. For example, the Query Designer could form an SQL query to be used with a local database, or form a URL to be passed to a web-based application. Once the Query Designer has created the query, it is passed back to the Query Manager.

4.9. Query Parser

The Query Parser takes the results received from the Data Source in its native format and given to it by the Query Manager, analyzes the results, and parses them into usable data. In some cases, this data is in the form of internal variables, while other data can be made into user prompts. In Restaurant Search Guide example, the results of one query are parsed into a prompt from which the user selects a restaurant about which to get more information; another query on a separate Data Source Adapter returns the directions from John's current location to the restaurant.

5. Conclusion and future work

In this paper, we presented VOICE, a framework for developing mobile context-aware systems with speech interface accessible over a telephone connection. To prove the feasibility of the architecture of VOICE, we implemented a Restaurant Search Guide, a simple context-aware recommender system that helps its users with finding a restaurant according to multiple combinations of criteria; the system connects to multiple data source providers to obtain restaurant data and to generate driving directions to the chosen restaurant. We plan to further validate this solution by conducting a usability study based in the analysis of the system usage patterns.

There are several directions for adding functionality to VOICE architecture. Mixed-initiative dialogs could be used to simplify the number of prompts presented to the user in order to obtain a minimally necessary amount of data needed for a successful execution of queries. To streamline the process of customization of VOICE during the development of specific applications, special templates could be developed for different types of Data Source Adapters corresponding to a local database, an XML Web Service, or an application-specific or platform-specific API.

6. References

[1] N. Anerousis and E. Panagos. "Making Voice Knowledge Pervasive", *Pervasive Computing*, IEEE CS, Apr-Jun 2002, pp. 42-48.

[2] S. Chang and M. Heng. "An Empirical Study on Voice-Enabled Web Applications", *Pervasive Computing*, IEEE CS, Jul-Sep 2006, pp. 76-81.

[3] J. Chugh, V. Jagannathan. Voice-Enabling Enterprise Applications. In Proceedings of the 11th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE CS, pp. 188-189.

[4] P. Danielsen. "The Promise of a Voice-Enabled Web", *IEEE Computer*, IEEE CS, Aug 2000, pp. 104-106.

[5] A. Dey. "Understanding and Using Context", *Personal and Ubiquitous Computing*, Springer, Jan 2001, pp. 4-7.

[6] A. Dix, et al. "Exploiting Space and Location as a Design Framework for Interactive Mobile Systems", *ACM Transactions on Computer-Human Interaction*, ACM, Sep 2000, pp. 185-321.

[7] P. Dourish. "What We Talk About When We Talk About Context," *Personal and Ubiquitous Computing*, Springer, Feb 2004, pp. 19-30.

[8] Y. Fan, et al. "Speech Interface: An Enhancer to the Acceptance of M-commerce Applications", Proceedings of the 2005 International Conference on Mobile Business (ICMB'05), IEEE, July 2005, pp. 445-451.

[9] J. Gilbert et al. "VoiceLETS Backs Up First Responders", *Pervasive Computing*, IEEE CS, Jul-Sep 2005, pp. 92-96.

[10] H. van der Heijden, et al. "Mobile Recommendation Systems for Decision Making 'on the go'", Proceedings of the 2005 International Conference on Mobile Business, IEEE, Jul 2005, pp. 137-143.

[11] J. Kleindienst, et al. "CATCH-2004 Multi-Modal Browser: Overview Description with Usability Analysis", Proceedings of the 4th IEEE International Conference on Multimodal Interfaces, IEEE CS, pp. 442-447.

[12] J. Kong. "Browsing Web Through Audio", Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing, IEEE CS, pp. 279-280.

[13] M. Lucente, "Conversational Interfaces for E-commerce Applications", *Communications of the ACM*, ACM, Sep 2000, p. 59.

[14] S. Middleton, et al. "Ontological User Profiling in Recommender Systems", *ACM Transactions on Information Systems*, ACM, Jan 2004, pp. 54-88.

[15] M. Satyanarayanan. "Challenges in Implementing a Context-Aware System", *Pervasive Computing*, IEEE CS, Jul-Sep 2002, p. 2.

[16] Z. Shao et al. "Transcoding HTML to VoiceXML Using Annotation." Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, IEEE CS, pp. 249-258.

[17] A. Srinivasan and E. Brown. "Is Speech Recognition Becoming Mainstream?" *Computer*, IEEE CS, Apr 2002, pp. 38-41.

[18] W. Srisa-an, et al. "Putting Voice into Wireless Communications", *IT Professional*, IEEE CS, Jan 2002, pp. 62-64.

[19] V. Stanford. "Beam Me Up, Doctor McCoy", *Pervasive Computing*, IEEE CS, Jul 2003, pp. 13-18.

[20] A. Tiwari et al. "Conversational Multi-modal Browser: an Integrated Multi-modal Browser and Dialog Manager". Proceedings of the 2003 Symposium on Applications and the Internet, IEEE CS, Jan 2003, pp. 348-351.

[21] B. Zhou et al. "Enhancing Mobile Web Access Using Intelligent Recommendations". *Intelligent Systems*, IEEE CS, Jan 2006, pp. 28-34.