A Multi-Modal Architecture for Cellular Phones

Luca Nardelli
ITC-irst
Via Sommarive 19, 38050
Povo, Trento, Italy
lunarde@itc.it

Marco Orlandi
Waycom s.r.l.
Via Boraxe 54, 17011 Albisola,
Savona, Italy
morlandi@waycom.it

Daniele Falavigna
ITC-irst
Via Sommarive 19, 38050
Povo, Trento, Italy
falavi@itc.it

Categories and Subject Descriptors: D.2.8 Hardware: input/output and data communications. General Terms: design, performance. Keywords: multimodality, automatic speech recognition, VoiceXML, mobile devices.

1. INTRODUCTION

Today, both Automatic Speech Recognition (ASR) and Text To Speech (TTS) systems are enough reliable to cope with a large set of telephone applications [1, 2]. At the same time, personal smart devices are becoming more and more powerful, in terms of computation and memory capabilities, while we are noticing a convergence between telephone and data networks, including WiFi networks. All of this is causing the increase in the demand of systems that can combine different interacting modalities. For example, ASR is necessary in absence of standard keyboards or in situations where user hands are busy; TTS is also needed when the devices have not sufficient display capabilities. In general, the combination of voice and graphic interfaces allows users to have a more comfortable interaction with a service, provided that the underlying system is transparent with respect to the chosen modality, i.e. the user has nothing to notify to the system itself.

In a previous paper [3] we have presented a system architecture that can synchronize events coming from different devices (e.g. keyboard, mouse, microphone, etc), allowing to handle flexible multi-modal interactions using Personal Digital Assistants (PDAs) as client devices. In this paper we will describe a possible extension of the proposed architecture allowing to handle interactions with new generation cellular phones.

Although the various modules of the system could be freely distributed inside a communication network, we will describe a “server side” solution, where both ASR and TTS resources are not embedded on the client smartphone, but are instead located on a remote server. This has the disadvantage that the speech signal needs be transmitted to the server, increasing the bandwidth requirements, but allows having solely a graphic interface running on the client (to be more precise, in the proposed system also a “start-end point detection” algorithm runs on the client, as will be seen below). Furthermore, speech recognition grammars do not need to be transmitted to the client, allowing to develop sophisticated and flexible applications, especially when the required size of speech recognition grammars is large or when they need to be dynamically changed during the interaction.

We will focus our attention on a smart cellular phone (Sony Ericsson P900) which allows to handle the communication with a Web server through a GPRS channel. At present, we are developing some multi-modal prototype services (mainly navigation inside a geographic map and form filling of Web pages) using such device.

2. DESCRIPTION OF THE SYSTEM

The system architecture is depicted in figure 1. As previously mentioned, we are adopting a server-side solution where all of the resources are located on the server. This last one hosts:

- the Web server, which handles requests for both HTML or VoiceXML (see http://www.voicexml.org) documents. It also manages the resources needed for both ASR and TTS functions (e.g. speech recognition grammars, speech files, etc);
- the Speech Server, which manages both ASR and TTS engines;
- the multi-modal server, which integrates the VoiceXML Interpreter.

We point out that both the multi-modal server and the VoiceXML Interpreter have been developed in our Labs, using the Java programming language for maintaining the portability among different platforms. Furthermore, both the ASR engine and the Speech Server, Spinet (Speech Into Enriched Text), have been developed by us; instead, the TTS is one of those commercially available. A Java Application Programming Interface (API), whose specification is similar to the Java Speech API (see [4]), allows the VoiceXML Interpreter Context (represented by the multi-modal server) to use the speech resources provided by Spinet.

The multi-modal server is formed by 3 main components:

- a Voice Manager, that manages the smartphone’s microphone (which can be resumed and paused) and sends the voice signal to Spinet;
• a Visual Manager, that controls a data connection (synchronization channel) with the smartphone;
• a Dialog Manager (DM), corresponding to the VoiceXML Interpreter Context, that integrates the VoiceXML Interpreter. The DM controls the whole interaction flow as will be seen below.

Figure 1: Architecture of the multi-modal browsing system.

The two managers (voice and visual) interact with the VoiceXML Interpreter through the DM. Two different gates listen, on two different sockets ports, to user requests and create new instances of the appropriate managers.

Our solution for synchronizing the client Graphic User Interface (GUI), in general an HTML browser, with the Voice User Interface (VUI, specified by a set of VoiceXML documents), consists in opening and maintaining a socket connection, over GPRS, between the smartphone and the multi-modal server. This connection is indicated in figure 1 as “synchronization channel”. Information is exchanged along this connection in order to:

• modify the specific GUI (i.e. changing the value of text fields) displayed by the smartphone, according to the input provided by the user’s utterances.
• send commands to the multi-modal server to update the VoiceXML context according to “non-voice” input provided by the user (e.g. by keyboard, mouse, pen, etc.).

For example, when the user utters one or more values for filling fields in a displayed form, the system modifies the loaded Document Object Model (DOM) pages in order to show the uttered data. On the other hand, if the user types in the value of a field, the Web browser communicates the event to the multi-modal browsing system on the server side through the Visual Manager, which in turn updates the corresponding VoiceXML field item variable. This process is controlled by both some JavaScript functions and a Java applet loaded in an hidden frame (see [3] for more details).

Note that the system allows to freely interact using separately the voice or the visual modality, or a free combination of both of them. In particular the architecture also permits to access the information on the Web server only through the voice channel, allowing the development of a pure voice portal.

The communication channels (voice, synchronization and control) depicted in figure 1 can make use of different transmission networks (e.g. the telephone for transmitting the voice, the Internet, a LAN, etc.); in particular, all the necessary information could be transmitted over the Internet. In this last case the voice signal has, in general, to be compressed before transmission, e.g. by means of some Voice Over IP (VOIP) algorithms, due to limitations in the available bandwidth.

However, the server-side solution has the advantage that the speech resources, including speech recognition grammars, are located on the server. In this way, unbearable delays can be avoided when an application requires to dynamically load large grammars.

2.1 Client side on the Sony-Ericsson P900

The smartphone chosen to develop the prototype demo is the Sony-Ericsson P900, a pen-based smartphone that integrates a powerful processor (based on the ARM1 architecture) and a user friendly display (208x320 pixels); furthermore, the P900 uses a real-multitasking Operating System, the Symbian 7.0 OS (www.symbian.com). A rich C++ API allows to use low level features, like audio streaming/coding based on an Adaptive Multi-Rate (AMR) codec, full screen access, GPRS/bluetooth connectivity and a customizable pen-based user interface (see www.uiq.com).

As shown in figure 1, the client application running on the P900 consists of 3 main modules:

• a voice module acquires voice from the microphone.
• the module also implements a simple start-end point detection algorithm, which cuts out the signal to be sent to the ASR using a procedure based on a dynamic threshold of the speech signal energy.
• a module that manages two socket connections opened over GPRS: the voice and the synchronization channel;
• an GUI handling input from both pen or buttons and displaying the data provided by the user’s utterance.

All modules on the smartphone have been developed in the C++ programming language.

3. REFERENCES