Spoken Language Interface for Mobile Devices

João Freitas[1], António Calado[1], Maria João Barros[2], Miguel Sales Dias[1]

[1] MLDC, Microsoft Language Development Center, Edifício Qualidade C1-C2 Av. Prof. Doutor Aníbal Cavaco Silva Tagus Park 2744-010 Porto Salvo, Portugal
[2] Eurescom GmbH, Wieblinger Weg 19/4, 69123 Heidelberg, Germany
i-joaof@microsoft.com, i-antonc@microsoft.com, barros@eurescom.eu, Miguel.Dias@microsoft.com

Abstract. In this paper, we present a set of optimizations for a spoken language interface for mobile devices that can improve the recognition accuracy and user interaction experience. A comparison between a speech and a graphical interface, when used to accomplish the same task, is provided. The implications of developing a spoken language interface and integrating speech recognition and text-to-speech modules for European Portuguese in a mobile device, are also discussed. The paper focuses in the speech recognition module and in an algorithm for name matching optimization that provides the user with a more comfortable interaction with the device. Usability evaluation trials have shown that spoken language interfaces can provide an easier and more efficient use of the device, especially within a community of users less experienced in handling mobile devices.

Keywords: language interface; speech interface; human-computer interface; mobility; user experience; usability evaluation.

1 Introduction

Spoken language interfaces impose a well know challenge to application developers: speech recognition is not perfect. We this in mind, software engineers should fully understand the strengths and weaknesses of the underlying speech technologies and identify the appropriate methodology and context, to use speech technology effectively. The process of integrating spoken language technologies in mobile applications as a Human-Computer Interface (or HCI) modality is highly dependent on the nature of the service provided by such technologies. In any case, any new HCI design should make the interaction between the user and the device easier. A well-designed HCI requires the consideration of the application user group, while making sure that the interface matches the way users expect it to behave [1]. The mobile environment adds challenges to the development of speech applications, due to the hardware limitations, e.g. in memory and in performance, and the conditions imposed by the different usage scenarios, e.g. background noise so common on mobility and the problem of device positioning towards the user. Nevertheless, the integration of spoken language technologies in mobile devices is gaining momentum. Common scenarios "busy hands, busy eyes" of the mobile world, e.g. driving, are classic examples of the need of integrating speech applications in the mobility world. The

trend leads to a presence of automatic systems in our daily lives and in many situations the space for a large keyboard or a mouse is inexistent, e.g. wearable computers. Speech is a solution and can be adopted as a command and control modality in parallel with others, or as a primary interface modality for multimodal interfaces. The combination of speech with other interaction modalities is generally more effective than a unimodal speech interface, due the inexistence of a 100% accurate method to perform speech recognition [1], [2], which resides in the probabilistic nature of current technologies.

Speech applications face yet another challenge: users want to interact with the application in their native language and want to use their own pronunciation. The localization of speech, that is, the provision of speech recognition and synthesis (spoken language interface) for a new language, includes a complex set of engineering and computational linguistics procedures and modules, which vary if we are considering recognition or synthesis. Some are of linguistic nature (pronunciation lexicons, phone sets, annotated and orthographically transcribed speech corpus, etc.) and some of a more mathematical/stochastically/algorithmic nature (acoustic models, language models, prosody models, grapheme-phoneme mappings, etc.). The example for mobile application presented in this paper, has a spoken language interface for European Portuguese (EP), designed for Pocket PC, which allows the user to place phone calls to any contact in the contact list. The purpose of this work is to demonstrate that a well designed speech HCI, in comparison with a graphical HCI, makes the usage of mobile devices easier, for a diversity of users including those with no experience in mobility. The application accommodates the user through a name matching algorithm and a well designed grammar, with the intention of simplifying the recognition problem. It is shown that by modelling the grammar, the application developer can improve the recognition accuracy and still implement a comfortable spoken language interface. This paper is organized as follows. Section 2 describes the basic components used in the development of speech applications. Section 3 details the development of a speech application, along with a name matching algorithm and grammar design considerations. In section 4, tests results are presented and discussed. Section 5 presents a usability evaluation trial and its results. Finally, in Section 6, the work conclusions are summarized.

2 Speech Applications

There are four classes of applications requiring different user interfaces: Desktop, Telephony, Home and Mobility. Desktop speech applications include widely used computing environments, such as Microsoft Windows and Microsoft Office. Telephony applications require server-side speech applications, such as Microsoft Speech Server and Microsoft Exchange. Home user interfaces are usually localized in the TV, living room or kitchen and speech introduces a great benefit since home appliances don't have a keyboard or a mouse and the traditional graphical user interface application can't be directly extended for this category. In the mobile case, Smartphone, PDA and automotive are the most important mobile scenarios due to the

physical size and the hands-busy and eyes-busy constraints [1]. All classes of speech applications have several development stages in common, as depicted in figure 1.

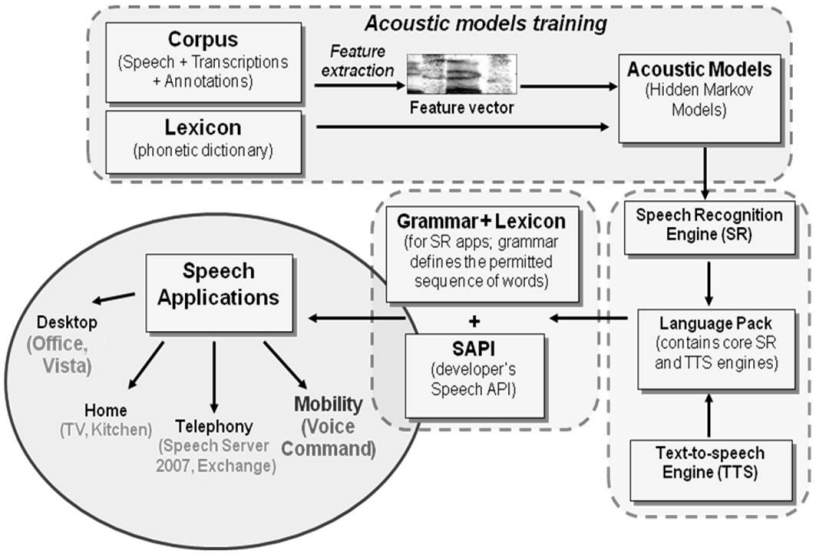

Fig. 1. Components of speech applications

The used speech recognition technology is based on statistical data-driven models of time-series known as Hidden Markov Models (HMM's) [3]. Each HMM in a speech recognition system models the time-based acoustic information of a specific speech segment within a language, which needs to be trained with real speech recorded for the language and acoustic environment in question, for example, EP telephony speech. These speech segments become representations of the speech units and can be of different sizes, e.g. whole sentences, whole words or even sub-word phonetic units like phones, diphones, triphones, syllables, etc.

2.2 Speech Recognition Engine

The application is responsible for loading the SR engine and for requesting actions/information from it. It communicates with the engine via a Speech API (SAPI) interface [4]. The SAPI abstracts the developer from the low level details of the SR engine. Nonetheless, it's essential that the developer is aware of the potential, functionality and technique performed by the SR engine, in order to model and optimize the target application. Usually, speech recognition engines for mobile devices present lower performance when compared to desktop engines. Optimizations, like fixed point calculations, lead to less recognition accuracy. Due to this, applications should perform a compensating effort in terms of recognition, e.g. minimize the number of grammar items, take in consideration the speech corpus used to train the acoustic models and model the application flow to avoid errors. In section 3 several optimization examples are presented.

2.3 Grammars

Speech applications are often built with a command and control application in mind. Section 3 shows an example that uses SAPI command and control features to implement its functionalities. Command and control features are implemented through context-free grammars (CFGs). This kind of grammar defines a set of imbricated production rules. These rules are able to generate a set of words and combinations of these words that can be used to build all types of allowed sentences. The result of a grammar production can be seen as a list of valid words/sentences that can be pronounced by the user, when interacting with the application via speech, which is passed on to the SR engine. Speech applications use grammars to improve recognition accuracy by restricting and indicating to the engine which words/sentences should be expected. The valid sentences need to be carefully chosen, considering users profile and the application nature. However, other approaches to SR without using CFG do exist [5], [6].

3 Example Application Development

In this section, the stages of development of an application that allows a user to place a phone call to a contact, in a Pocket PC device, are described. The intention is to present its architecture and the relevant design decisions, considering the language, mobile environment and speech recognition procedure. The example application was referenced as "Pocket Reco" and it is a C++ application developed for Pocket PC devices, interacting with the user through a spoken language interface in EP and influenced by applications such as Microsoft Voice Command [7], Cyberon Voice Commander [8] and Voice Signal [9]. The application allows the user to place phone calls to any contact in the MS Windows contact list and to consult his/her agenda. It was developed using Microsoft Visual Studio 2005 and MS Windows Mobile 5.0 Pocket PC SDK [10]. The application is localized for EP, in a context where the amount of applications with spoken language interfaces in EP, that exist in the market, is reduced. If we consider only the mobility world, that number decreases even more. It's important that Portuguese users have the possibility to use speech recognition and speech synthesis applications in European Portuguese, especially those with special needs, such as with vision impairments, which depend on this type of applications to use a mobile device.

3.1 System Description

When the user initiates the application Pocket Reco, by pressing a button or by running the executable file, the speech recognition and text-to-speech modules are initiated. To notify the user, an icon is loaded to the notifications tray and a spoken message is synthesized. At this point the application expects the user to input a speech command. If no expected command is recognized there is a predefined timeout. Once the speech command is recognized, a set of actions associated with the command are executed. When these actions are finished, the application ends, unloading all

resources. The user can input commands to make phone calls or check the calendar. When the user chooses to make a phone call by saying the respective command followed by the contact name, it may be the case that the contact has more than one phone number (mobile, home, work, etc). In this situation the user is notified to choose between the available numbers by saying the respective command.

3.2 Architecture

Pocket Reco contains a SR and a TTS module, which communicate with the respective engines through SAPI. SAPI communicates with the engines through a device driver interface [4]. Both SR and TTS engines contain a SAPI interface layer, allowing the use of SAPI runtime. Figure 2 illustrates the interaction of the application with the engines, through SAPI.

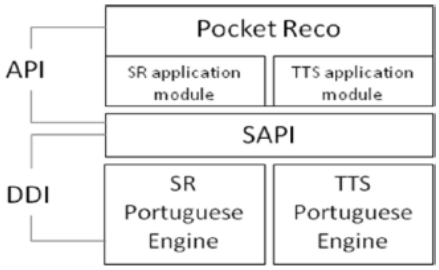

Fig. 2. Diagram of the speech architecture

3.3 CFG Grammar

The application uses a CFG grammar to parse the recognizer output. The grammar is composed by dynamic and static rules. The dynamic rules are empty when the application starts and their contents are updated in runtime. The separation between static and dynamic rule contents allows the application to start and load the static content, without loading the SAPI grammar compiler, thus preventing additional delays in the start-up sequence. When the application loads the dynamic content it forces SAPI to initialize the backend grammar compiler [4].

In Pocket Reco there is a distinction between static rule content and dynamic rule content, in order to develop a well-designed grammar and to improve initial SAPI grammar compiler performance. The dynamic content is, in this case, the list of contacts names, as these can only be accessed at runtime. The static rules that use the contact names will reference the respective dynamic rule.

3.4 Name Matching Algorithm

A spoken language interface with command and control modality, like Pocket Reco, should supply natural and simple voice commands. This way, the user will not need to

memorize the commands and the required words will be spoken naturally. The speech command responsible for placing a phone call to a contact in the contact list is composed by a static part, such as “Chama” (“Call”), that resides in the grammar file, and a dynamic part composed by the contact name. If we consider a user with a large contact list he/she might not remember exactly the contact’s name. In a different scenario, one can consider the case of persons with many names (six and seven names occur frequently within European Portuguese full names), which might be called by their first, their last, their middle or any combinations of these names and more. For example, if we download a contact list from an Exchange server with a contact named João Paulo de Oliveira Esperança Gonçalves, he might be known as “João Paulo” by user “A” and “João Esperança” by user “B”. Considering the functionality provided by Pocket Reco to make a phone call to a contact in the contact list, the correspondent command is dependent on the contact name. The developed name matching algorithm allows the user to say the contact name in any form he/she is used to. Instead of residing in the grammar only the first and last name of each contact, all different names from the contact list are placed into the CFG grammar in separate phrases, without repetitions. This will allow the recognizer to accept any combination of names, e.g. “Chama João Paulo Gonçalves”, “Chama João Gonçalves”, etc. After the recognition stage, the spoken contact name must be analysed. The application will access the contact list, which is, in this particular case, residing in memory (loaded when the application starts), and search, through this list, the contact that has the highest number of matches with the spoken name. With the proposed algorithm there is the chance of name ambiguities. When this happens, the algorithm either chooses the first name occurrence or gives the user the possibility of solving the ambiguity, which is the most correct approach. This can be achieved by giving the user the contacts possibilities and respective full names (through TTS), asking him to choose the desired contact. The disadvantage of providing flexibility to the user’s interface with this algorithm is the increased number of items in the grammar, leading to an increase of the word error rate in the recognition process. There must be a compromise between application flexibility and recognition accuracy, depending of the application user’s group and the recognition engine.

3.5 Locale Culture

The application localization is an important factor in the application design. It is necessary to understand the locale specific issues relative to the application. In the specific case of EP, it was necessary to identify the naming procedures executed by EP speakers, or in other words, checking the common way of placing a phone call. For example, in EP it is common to say “Chama” or “Liga” followed by the person name and less common to say “Efectuar chamada”. The inclusion of the most common words in the grammar to describe an action leads to a more natural user interaction.

4 Experimental Results

In this section, a subjective usability evaluation experiment that compares the accomplishment of a task through the use of a graphical interface and of a speech interface is presented. This set of subjective usability tests compares the usage of a graphical interface with the usage of a speech interface while performing a common task in mobile devices.

The tests were applied to a universe of 30 unpaid adult persons, 16 male and 14 female, from the Portuguese academia. Two groups, each with 15 subjects, were considered: one group had prior experience with speech interfaces in mobility; the other group presented no previous experience with speech interfaces and the experience with mobile devices is resumed to cell phones. Both groups covered all the considered ranges of ages as shown in the table 1. Each person performed two tasks, calling to a predefined contact number that resided in a contact list with 20 contacts, through the graphical HCI and the same task through the speech HCI, e.g. "Call to Pedro Silva mobile phone". The usability experiment collected the time duration of the task as a metric that relates to its difficulty. Each subject received a script with clear instructions on how to perform both tasks. The tasks were executed by each subject in a random order, under the same conditions, with Pocket Reco running on a Pocket PC (HTC P3600 - 400MHz, 128MB ROM, 64MB SDRAM), in the following environments: Office, Home, Party (Crowded room) and Car.

Both groups presented similar average value of recognition accuracy - around 84% - with an advantage of 0,5% for the subjects with previous experience on speech applications in mobile devices. Table 2 shows the average time (in seconds) taken to accomplish the proposed tasks.

Table 1. Subjects distribution.

Age	Subjects	
	With experience	*Without experience*
< 24	4	4
24-29	6	2
30-35	3	3
> 36	2	6
Total	15	15

Table 2. Time taken to perform the tasks.

	GUI (seconds)	Speech (seconds)
With mobility experience	15,0	12,2
Without mobility experience	26,5	14,9
Both groups	20,8	13,6

In the presented results there is a clear advantage in the time metric (34,6% lower), for actions executed with the help of the speech interface. Nonetheless, some of the subjects (especially those with no experience) had to execute more than one attempt, due to recognition errors, to accomplish the task using the speech interface. The

recognition errors happened due to background noise and the way subjects interacted with the speech interface, such as bad positioning of the device, covering the microphone and uttering words unclearly, often using abbreviations.

5. Usability evaluation

In this section, we present the results of a wider usability evaluation test. The goal of this second set of experiments was to assess the usefulness of a speech interface in comparison with the mobile device Graphical User Interface (GUI) provided by Windows Mobile, using a localized EP version of Voice Command [7] and, this time, cover the majority of the basic functionalities provided by a Pocket PC. To determine the usability of a speech interface, an evaluation methodology was adopted and a usability test was developed. The device used was the same HTC P3600 that was used for the previous tests, however, the amount of stock data increased from 20 to 300 contacts and 50 new music files where inserted in the device storage card. The tests were conducted at home and office environments. In the home environment the tests took place under a common living room scenario (television on, though relatively quiet). In the office environment the tests took place in a room with an average 4 to 5 people talking/working.

5.1 Methodology

The usability testing experiment was designed to assess the usefulness of Voice User Interface (VUI) interaction compared to traditional GUI interaction. For this, the time taken and efficiency of the subjects when accomplishing two groups of tasks commonly available in a mobile device was evaluated. The tasks are described in table 3. Nowadays, there are many actions and services provided by a Pocket PC or a Smartphone; so we selected a first group of tasks that are the primary reason or essential when using a device and, consequently, more common amongst users. A second group of tasks was also selected and represents actions that are not so popular, but depending on the user, they can be more or less used. The tasks were designed with the intent of being the most global and representative of a group of services.

Table 3. Tasks performed in the usability evaluation test.

Common tasks	**Other tasks**
1. Placing a phone call	5. Lookup contact information
2. Checking calendar	6. Open an application
3. Verifying device status	7. Play media
4. Change profile or sound options	8. Manipulate media

These sets of tasks are performed using both interfaces alternately (e.g. the subject performs the tasks with the VUI and then performs the same tasks using the GUI; the next subject first performs the tasks with GUI and then performs the same tasks with

the VUI). By alternating the interface order independence is gained, breaking any kind of relation between the interfaces that may affect test results. The tests, regarding each interface, are preceded by a four minutes session for interface adaptation. The adaptation is divided in a two minutes explanation and followed by a two minutes training session, so that the subject is able to adapt himself/herself to the device. Each subject also receives a two pages user guide and a half page quick user guide, about each interface, explaining how to use the device. The tutorial about the user interfaces include the tasks mentioned in table 3. The tutorials are as clear and equivalent as possible. Each testing session took about 15-25 minutes, depending on the subject feedback.

5.2 Subject profiles

The usability experiment was run on 35 unpaid subjects with ages between 23 and 65. There were 26 male and 9 female subjects with a variety of business occupations (IT engineers, Senior Consultants, Retired, Students, etc). In order to obtain a reference profile the subjects were asked to fill a questionnaire about their previous experience on speech interfaces, use of stylus (on mobile devices) and mobile devices in general. The results reveal that the majority of the subjects had low experience with speech interfaces and in a scale of 0 to 3 (0 – None, 1 – Low, 2 – Medium, 3 - High) presented an 1, 2 average value; a medium experience in the use of stylus, with an 1,8 average on the same scale and high experience with mobile devices, presenting an 2,7 average (again same scale of 0 to 3). These results are presented in the graphic of fig. 3 below and were gathered through a questionnaire filled before executing the tasks.

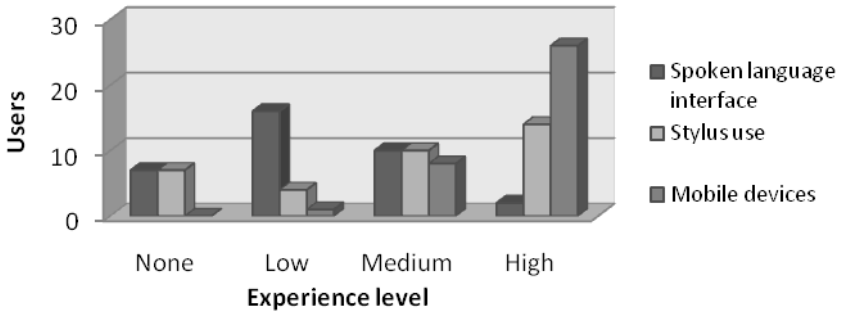

Fig. 3. Subjects experience profile

5.3 Evaluation results and analysis

All subjects were successful in completing the experiments with both interfaces, showing different degrees of efficiency and taking different durations to fulfil the tasks. During the experiments we became aware of some problems that rose while using the system and received valuable feedback which is described below. It was also performed an analysis on: the time that the subjects spent to perform the

experiment; the number of attempts when performing a task; why subjects failed to use the VUI on a first attempt; the questionnaires filled after the usability test.

When analysing the times taken in the several tasks, there is a clear advantage of the VUI placing a call to a determined contact, checking a contact card and playing a group of songs (fig. 4). VUI's advantage in such tasks is explained by the task complexity. All of these tasks have in common the fact that they require a set of actions, such as menu navigation, scrolling, etc, to be executed with success. The VUI gives a way of accessing all of these services through a single action, as simple as issuing a voice command. The inverse situation can be noticed when checking tomorrow's appointments or passing to the next song. To execute these tasks both interfaces only need a single action, so it is natural to have a balance in terms of time when executing them. Despite the time needed to accomplish a task, it will be shown that the subject not always prefer the fastest way to perform the task.

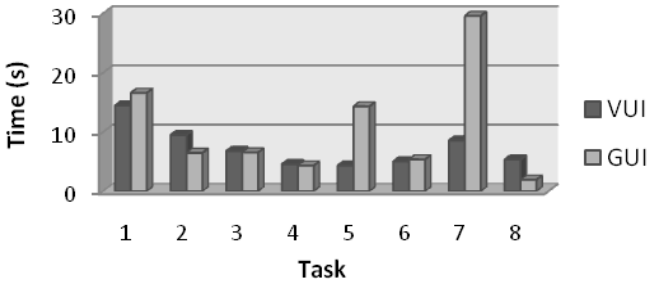

Fig. 4. Time taken in accomplishing the tasks (VUI and GUI)

When using the VUI, recognition errors are expected to occur, especially considering that the majority of the subjects reported a low experience level with speech interfaces. The main reasons for recognition errors in these tests were: The spoken command is not included in the grammar; the command sentence was spoken in an incorrect order (e.g. in the grammar is available the command sentence "Que horas são?" and the user says "São que horas?"); covering the microphone with the hand that holds the device when uttering the voice command; bad interaction with the push-to-talk (PTT) button, such as saying the command without pushing the button; taking more time than usual to input the command (after pressing the PTT button); push the PTT button and not wait for the noise or visual signal from the application to input the voice command; and talking before introducing the speech command (when the application is waiting for a command).

Regarding the number of attempts taken to accomplish the tasks, it was registered that the number of attempts decreases during the test execution, except for task number five. The decrease in the number of attempts can be justified by the quick adaptation of the subjects to the VUI, since the tasks where independent. This was also visible during the tests. The exception verified, for task number five (lookup for contact information), is due to the lack of synonyms in the application grammar to perform this particular action.

5.4 Subject feedback

The subjects provided valuable feedback during and after the experiment, mostly regarding the VUI interface. Right after the testing session, the subjects were asked to fill a short questionnaire that allowed extracting the opinions and preferences of the subjects, when asked to compare both interfaces and about the VUI.
From section 5.2 can be concluded that the subjects are more familiar with the GUI, as opposed to VUI. Ideally, the subjects had equal familiarity with both modalities so that we could make a more precise comparison based on it. Despite that, there are still some interesting conclusions that one can extract. When analysing the provided feedback, a clear preference for the VUI in accomplishing the tasks is observed, despite the time taken to accomplish the task. The tasks one and seven reveal that interaction via VUI is overwhelming, when compared to the GUI. This is justified by the fact that, when using the GUI, inherent actions such as finding a contact to place a call, or having to go to the music repository to play a group of songs requires additional attention, time, visual interaction and manual interaction with the device. On the other hand, the VUI only requires a single voice command to index a contact number or an album artist.

During the testing period the subjects gave their opinion on the application, presenting a set of suggestions that include having a more stronger component of visual feedback when stimulated with a voice command, e.g. when asked for available albums the application gives a list of the available albums in the device (on which we can navigate with speech). This could be implemented as an option. To conclude the analysis about the subject's feedback, 88,6% of the subjects prefer the VUI interface when asked to choose between the two interfaces, and only 11,4% prefer the GUI interface. Nonetheless, 21% of the subjects mentioned that their option on which interface to use depends on the situation and location. These subjects commonly choose the GUI interface for public places such as shopping malls, and the VUI for "busy hands, busy eyes" scenarios such as, automotive and sports.

6 Summary

The known disadvantage of speech HCI, including its applicability in mobility, is that speech recognition is not 100% accurate, due to its probabilistic nature. This fact may lead to recognition errors which the application must handle and try to resolve, for example, by asking for confirmation on a given voice command to avoid false-positive results. In some use cases, with noisy environments or less experienced users, several attempts may be needed to successfully accomplish the task. In speech recognition for mobility, the optimizations performed over the models and SR engine often decrease the components performance. If that happens, the application implementation can compensate this decrease in quality. The grammar is an application component that the developer can design carefully to optimize the recognition.

In this paper we have described a set of optimizations to improve recognition accuracy and user interaction experience, in the framework of a spoken language

interface in a mobile device. In order to complement the approach it was developed an example application that places a phone call to a contact, through a spoken language interface. The application interacts with the SR engine through SAPI and implements a name matching algorithm to improve the user experience. It was also presented the results of a usability evaluation trial, to demonstrate the advantage of using a speech HCI versus a graphical HCI. In this work, it was shown that the set of words chosen to be recognized will influence the application performance. This choice of words is influenced by the acoustic models, the quality of the speech recognition engine and language specifications. The developer has the responsibility to balance the application according to its usage scenario and needs to adapt the application to the user group. In this paper we presented an application that takes into account all these considerations, discussing the advantages and drawbacks of the various design choices. The name matching algorithm here presented is an example which trades recognition accuracy for a more intuitive and natural interface. The executed tests have shown that we can manipulate the grammar to improve the performance of the algorithm. The conducted subjective usability tests demonstrated that users with no experience in speech applications and mobile devices can easily learn how to execute a simple call action in the device, through a spoken language interface. The presented results showed that the time to accomplish a simple task, like placing a phone call, is 34,6% lower when using a speech interface as a replacement for a graphical interface. The time difference between the two types of interfaces decreases drastically according to the level of the user expertise with speech applications in mobile devices. The conclusions also showed that complex tasks, such as playing a group of songs, are easier to accomplish with a VUI (Voice User Interface). A quick adaptation by the users to the spoken language interface, during the conducted usability tests, was verified, resulting in lesser attempts to accomplish a task.

Acknowledgements

The authors would like to thank to the Microsoft Language Development Center members, in Microsoft Portugal, and to all the persons that participated in the usability experiments.

References

1. Huang, X., Acero, A., Hon, H.-W.: Spoken Language Processing - A Guide to Theory, Algorithms, and System Development. Prentice Hall, New York (2001)
2. Acero, A.: Building Voice User Interfaces. In: MSDN Magazine. February edition (2006)
3. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. In: IEEE 77(2):257–286 (1989)
4. Microsoft Speech SDK, http://msdn.microsoft.com/library/aa914072.aspx
5. Acero, A.: Acoustical and Environmental Robustness in Automatic Speech Recognition. The Springer International Series in Engineering and Computer Science, Vol. 201 (1993)

6. Stern, R., Acero, A., Liu, F., Ohshima, Y.: Signal Processing for Robust Speech Recognition. In: Automatic Speech and Speaker Recognition, Advanced Topics, Kluwer Academic Publishers (1996)
7. Microsoft Voice Command, http://www.microsoft.com/windowsmobile/en-us/downloads/microsoft/about-voice-command.mspx
8. Cyberon Voice Commander, http://www.cyberon.com.tw
9. Voice Signal, http://www.voicesignal.com
10. MSDN Windows Mobile 5.0 SDK Documentation, http://msdn.microsoft.com/en-us/library/ms376766.aspx