Multi-Modal voice application design in a Multi-Client environment

Jozef Ivanecký, Markus Klehr,
Volker Fischer, Siegfried Kunzmann

IBM Pervasive Computing, European Voice Technology Development
Gottlieb-Daimler-Str. 12, 68165 Mannheim, Germany
{ivanecky, klehr, vfischer, kunzmann}@de.ibm.com

Abstract. The seamless access to information and services is a key requirement for any pervasive or ubiquitous computing environment, and the access via any client is becoming a more and more feature within this scenario. The paper describes our efforts towards a multi–client voice application with focus on an embedded client, like e.g. a commercially available PDA. We give an overview on speech recognition techniques suited for the special requirements of the expected acoustic environments and explore the ability to design applications that are able to run on different voice and GUI capable devices.

1 Introduction

The VoiceXML standard is currently becoming a widely accepted way to design voice controlled applications. Whereas with HTML and additional tools such as ECMA script or Perl it is possible to design graphical user interfaces and simple but already dynamically interacting applications, the combination of VoiceXML and HTML allows us to create a multi-modal interface that simultaneously supports also human voice. Since no standardized combination of both markup languages is available, we developed a multi-client approach that allows to write applications in VoiceXML, HTML, or both type of dialog descriptions, depending on the kind of modality provided by the client.

Figure 1 shows the general architecture of such a scenario. The central part is an application server that generates appropriate VoiceXML and HTML pages. We assume 3 different kinds of clients:

– clients with a graphical user interface only
– clients with a voice user interface only
– clients with a multi-modal user interface combining graphic and voice

As a graphical user interface we assume the standard HTML browser. For "Voice only" we assume a voice server capable to handle input via a telephone line. In the third case the client is a multi-modal VoiceXML browser that is able to process VoiceXML pages as well as HTML [9]. A client-independent application server uses XSLT style-sheets that depend on the client's capabilities for the generation of a proper output.

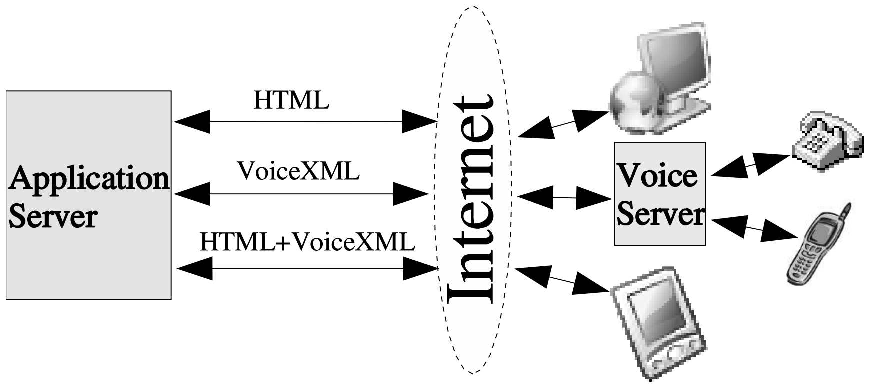

Figure 1. The multi-client scenario.

To demonstrate the capabilities of this scenario we present a "traffic jam" application that allows to obtain online information about the current traffic situation on any highway in Germany. The server side of the application is entirely written in PHP. The server generates VoiceXML and HTML pages which may also contain ECMA script. The application is completely platform independent, and the support for a new client can simply be added by the definition of new a XSL style sheet.

The remainder of the paper is organized as follows: In Section 2 we give a brief overview over speech recognition techniques for client and server sided speech recognition. In Section 3 we focus on the application design with an emphasis on XSLT client dependent output formatting. Section 4 describes the example application and a brief summary is given in Section 5.

2 Speech Recognition

Voice driven interfaces for consumer devices such as PDAs, mobile phones, smart–phones, or car navigation systems are becoming increasingly popular. While today there is no doubt that the overwhelming majority of such appliances will make use of the well established Hidden Markov Model (HMM) based approach to statistical speech recognition, it is also obvious that special needs of the embedded and mobile domain must be considered. In the remainder of this section we will review the basic requirements imposed by the scenario under consideration and will describe how these are taken into account in the training of acoustic models as well as in the recognition.

Emerging standards like VoiceXML offer an up to now unknown degree of freedom in the dialog design for the voice driven interfaces. Therefore, sub–word HMM based recognizers that offer a large flexibility in the vocabulary and dialog design, are becoming indispensable, whereas whole–word based recognizers, which are still predominant in many small vocabulary embedded speech recognition applications, will more and more disappear from the scene. Recent industrial joint efforts in the creation of high quality speech data bases for consumer devices (see [12]) already take this into consideration by collecting a substantial amount of rich context data. We have recently started to explore the offered flexibility of such speech data bases by the design of a special phone set that uses both general and application specific phones (e.g. for digits). By a careful analysis of relationships between recognition errors and model inventory, we were able to reduce the digit error rate in an automotive environment by up to 20 percent relative, without affecting the recognition accuracy for other applications.

It is well known that recognition accuracy improves significantly, if the acoustic model is trained with data that matches the target domain, e.g. in the type and amount of environment noise . Since voice interfaces for embedded devices are expected to work under a wide variety of conditions — consider, for example, the use of your PDA in your office, your car, or in a train station — the acoustic model must therefore incorporate training speech that properly reflects the characteristics of different environments.

Finally, but probably most important, the design of an embedded speech recognizer has to deal with only limited computational resources, both in terms of CPU power and memory capacity, that today's embedded devices can offer. While some applications may run entirely on the local device and therefore require a relatively compact acoustic model, others may defer parts of the recognition process to a recognition server, which requires compatibility of at least the client's and server's acoustic front–end.

The latter is ensured by the use of a standard acoustic front–end, that computes 13 Mel Frequency Cepstrum Coefficients (MFCC) every 15 milliseconds. Utterance based cepstral mean subtraction and C0 normalization are applied to compensate for the acoustic channel and the first and second order delta coefficients are computed to capture the temporal dynamics of the speech signal. While more recently other feature extraction techniques such as MVDR [8] have been demonstrated to provide superior accuracy in noisy environments, cepstral coefficients are well suited for low bit–rate compression and transmission [11]. Moreover, speech can be reconstructed from cepstral coefficients and a simple pitch tracker [6], which we consider as a prerequisite for the text–to–speech component of future multi–modal interfaces for embedded devices.

The use of additive noise from the real environment is a well known method to increase the robustness of a speech recognizer under adverse conditions (e.g. [13, 3]). While we concentrated on the use of engine noise for in–car speech recognition in the past [10]), we have more recently started to incorporate a wider variety of non–stationary noise types that were collected with commercially available PDAs.

Recognizer training comprises the definition of a suitable HMM inventory and the determination of the HMM parameters. For that purpose, the training data is viterbi–aligned against its transcription in order to obtain an allophonic label for each feature vector. Context dependent non cross–word triphone HMMs are obtained from the leaves of a decision network [1] that is constructed by asking binary questions about the phonetic context P_i for each feature vector,

$i = -1, \ldots, 1$. These questions are of the form: "Is the phone in position i in the subset S_j?", and the subsets are derived from meaningful phone classifications commonly used in speech analysis. We found small gains from using only clean data for the construction of the HMM inventory. Finally, the data at each leaf of the network is used in a k–means procedure to obtain initial output probabilities whose parameters are then refined by running a few iterations of the forward–backward algorithm.

The k–means procedure follows a simple rule of thumb and equally distributes a fixed number Gaussian mixture components across the HMM states. Usually, in a highly dynamic and heterogeneous environment, an increased total number of Gaussians can significantly improve the recognition accuracy. However, this is infeasible for applications that have to deal with a limited amount of memory, and therefore the determination of an appropriate acoustic model size is of particular importance.

The Bayesian Information Criterion

$$BIC(M) = \log L(X, M) - 1/2(\#(M) \cdot \log(n)) \tag{1}$$

is a model selection criterion that penalizes the likelihood L(X,M) of a data set X of size n by the number of parameters #(M) in the model [7]. We used BIC based clustering as an alternative to the k–means procedure and found that the method can produce both smaller models and more accurate results; cf. [10].

The so created acoustic model can run with either IBM's large vocabulary telephony speech recognition engine, which employs a fast pre–selection of candidate words and an asynchronous stack search algorithm, or with a time–synchronous viterbi–decoder. The latter is the core of IBM's Embedded Speech Engine (ESE), which is designed for the use with a moderate vocabulary size and finite state grammars. The highly portable and scalable ESE can run on any suitable 32 bit general–purpose CPU; see [4] for an overview on design issues and performance.

3 The Multi-client Design

This section gives a brief overview on the multi-client capabilities of the application. The techniques described here constitute one of several possible approaches, and were selected because they allow to easily extend the application for a new client.

Originally, we defined several requirements for the application behavior:

1. The application has to cover different clients.
2. The incorporation of a new client into the running application has to be maximally simplified.
3. The executive code has to be separated from the data as well as dialog control description.

To achieve the above mentioned requirements we decided for the use of XML, XSL and XSLT in a PHP environment. The dialog flow is described in XSL files.

For the different clients the different XSL files are defined wherever it is required. The dynamical data – like e.g. client side application data, grammars, etc. – are stored in the XML files. Beside this two type of files there are also static data like audio files, graphical images, static grammar or other static data which can be used for improving the design for a certain client.

The PHP server side application is listening to the clients' requests. Based on the incoming HTTP request and the type of the client, the PHP script selects the appropriate XSL file (holding the client dependent static data), an XML file (holding the client independent dynamical data), and some additional static data files to generate the requested page (pages). The internal structure of the server side is depicted in Figure 2.

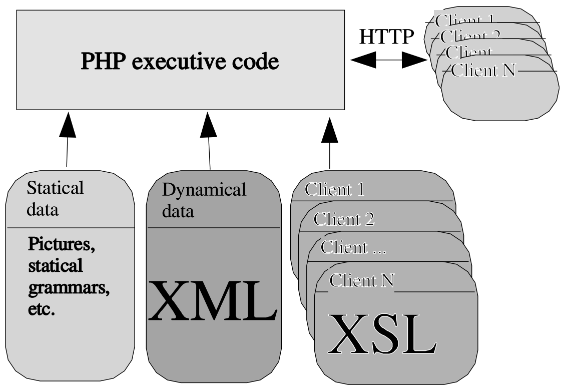

Figure 2. Internal architecture of the server side.

We refined the above mentioned techniques during the development of several simple multi-client case study applications. Experience gathered during the development procedure shows that this multi-client approach is sufficient for the efficient covering of several different clients.

4 Application

As a prototype application we implemented a simple traffic jam information system, where we can easily demonstrate the multi-client as well as the multi-modal access. With any of the available clients, the user is able to obtain information about the current traffic situation on any highway in Germany. After obtaining the requested information, the user can either finish the dialog, or initiate an appropriate action, like e.g. inform a partner that he is in traffic jam and will be late.

Differences between the clients lead to the following scenarios:

– In case of a graphical user interface, the user can choose the required information by clicking on a map. The list of messages is displayed. After picking

up the concrete message, the map with highlighted problematic area is displayed. Optionally an SMS can be sent to a phone number stored in the clients address book or a new number.
- For telephony clients the entire communication is done by voice. Voice input is recognized by use of IBM's standard ViaVoice telephony engine, and the system response is produced by use of IBM's trainable concatenative Text-to-Speech system [5]. The user can either listen to all the messages or can interrupt the output at any time and continue with the dialog. At the end the user can again either send the message to a number from the address book or dictate a new telephone number.
- In case of a multi-modal client both of the previous scenarios are available simultaneously, and it is possible to use any modality at any time. However, different from the standard telephony recognition system, in case of voice communication barge-in is not available, because of some hardware limitations in today's handhelds (half duplex audio chips). The client used in this scenario is a commercially available PDA that runs a multi-modal browser and is connected to the Internet via GPRS. Because all application data needs to be transferred from the server, the images were designed with respect to the connection speed. Both speech recognition and synthesis entirely run on the client side and use IBM's embedded ViaVoice speech recognition engine and a high-quality concatenative Text-to-Speech system with low footprints [2].

All clients are providing the same information constrained just by the lack of the modality. The full benefits from using voice simultaneously with the GUI is available only for the multi-modal client.

5 Summary

In this paper we described various aspects of the development of a multi-client, multi-modal voice application. We started with the description of techniques for the training of highly noise robust acoustic models that are well suited for both client and server sided speech recognition. We also described the design of a real-life applications that demonstrates the feasibility of the chosen approach. While client independence and the arbitrary use of an input modality are important features of the application design, we consider the XSLT approach as distinct characteristics of the application.

Acknowledgement. The authors wishes to thank the Voice Technologies and System Group, IBM Research Prague, for the support with multi-modal browser. Special thanks to Luboš Ureš and Bořivoj Tydlitát.

References

1. L. Bahl, P. de Souza, P. Gopalakrishnan, D. Nahamoo, M. Picheny. Context-dependent Vectoy Quantization for Continous Speech Recognition. In *Proc. of the*

IEEE Int. Conference on Acoustics, Speech, and Signal Processing, Minneapolis, 1993.

2. D. Chazan, R. Hoory, Z. Kons, D. Silberstein, A. Sorin. Reducing the Footprint of the IBM Trainable Speech Synthesis System. In *Proc. of the 7th Int. Conf. on Spoken Language Processing*, Denver, 2002.
3. R. Bippus, A. Fischer, V. Stahl. Domain Adaptation for Robust Automatic Speech Recognition in Car Environments. In *Proc. of the 6th Europ. Conf. on Speech Communication and Technology*, volume 5, pages 1943–1946, Budapest, 1999.
4. L. Comerford, D. Frank, P. Gopalakrishnan, R. Gopinath, J. Sedivy. The IBM Personal Speech Assistant. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, 2001.
5. R. Donovan, E. Eide. The IBM Trainable Speech Synthesis System. In *Proc. of the 5th Int. Conf. on Spoken Language Processing*, Sydney, 1998.
6. D. Chazan, R. Hoory, G. Cohen, M. Zibulski. Speech Reconstruction from Mel Frequency Cepstrum Coefficients. In *Proc. of the IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pages 1299–1302, Istanbul, Turkey, 2000.
7. S. Chen, P. Gopalakrishnan. Clustering via the Bayesian Information Criterion with Applications to Speech Recognition. In *Proc. of the IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pages 645–648, Seattle, 1998.
8. S. Dharanipragada, B. Rao. MVDR Based Feature Extraction for Robust Speech Recognition. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, 2001.
9. V. Fischer, C. Günther, J. Ivanecký, J. Šedivý, L. Ureš. Towards Multi-Modal Interfaces for Embedded Devices, In *Proc. of the 13. Konferenz Elektronische Sprachsignalverarbeitung*, Dresden, 2002.
10. V. Fischer, S. Kunzmann. Bayesian Information Criterion based Multi-style Training and Likelihood Combination for Robust Hands Free Speech Recognition in the Car. In *Proc. of the IEEE Workshop on Handsfree Speech communication*, Kyoto, 2001.
11. G. Ramaswamy, P. Gopalakrishnan. Compression of Acoustic Features for Speech Recognition in Mobile Environments, In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 977–980, Seattle, 1998.
12. R. Siemund, H. Höge, S. Kunzmann, K. Marasek. SPEECON – Speech Data for Consumer Devices. In *Proc. of the 2nd Int. Conf. on Language Resources & Evaluation*, pages 883–886 Athens, 2000.
13. A. Varga, H. Steeneken, M. Tomlinson, J. Jones. The NOISEX-92 Study on the Effect of Additive Noise on Automatic Speech Recognition. In *Booklet of the NOISEX-92*, CD-Rom, 1992.
14. VoiceXML 2.0, W3C, *http://www.w3.org/TR/2001/WD-voicexml20-20011023*, Working Draft, Oct 2001.