A CONFIGURABLE DISTRIBUTED SPEECH RECOGNITION SYSTEM

Haitian Xu[1], Zheng-Hua Tan[1], Paul Dalsgaard[1], Ralf Mattethat[2], Børge Lindberg[1]
[1]Speech and Multimedia Communication (SMC), Center for TeleInFrastruktur (CTIF),
Aalborg University, Denmark
[2]Technology Institute, Århus, Denmark
{hx,zt,pd,bli}@kom.aau.dk ralf.mattethat@teknologisk.dk

Abstract

The growth in wireless communication and mobile devices has supported the development of distributed speech recognition (DSR) systems. During the last decade this has led to the establishment of DSR standards and an increased interest in research aimed at systems exploiting DSR. So far, however, DSR-based systems executing on mobile devices are only in their infancy. One reason probably is the missing availability of corresponding easy-to-use software development packages. This paper presents a prototype version of a configurable DSR system for the development of speech enabled applications on mobile devices.

The system is implemented on the basis of the ETSI-DSR advanced front-end and the SPHINX IV recogniser. A dedicated protocol is defined between the DSR client and the recognition server supporting simultaneous access from a number of clients. This makes it possible for different clients to create and configure recognition tasks on the basis of a set of predefined recognition modes.

The paper gives a detailed introduction to this system including its architecture, design considerations and evaluation results.

1. Introduction

It is expected that the growth in wireless communication and mobile devices will enable ubiquitous access to a large pool of different information resources and services. To make such development successful there is a demand to include ASR as a key component into the user interface. Present mobile devices only have limited memory and CPU capacities which pose several challenges to ASR. As a result most ASR systems today executing on mobile devices only support low-complexity recognition tasks such as simple name dialling.

The resource-limitation can be partly alleviated by adopting a client-server based DSR network architecture where only the front-end feature extraction lies in the client while the time-consuming recognition decoding is conducted in a powerful server [1]. With such an architecture the resource limited client becomes independent of the recognition tasks and therefore enables the implementation of complex recognition tasks - e.g. large vocabulary speech recognition.

Another challenge in deploying speech recognition in mobile devices is the continually changing acoustic environment. This additionally requires the deployment of noise-robust signal processing techniques generally of high computational complexity. The introduction of noise robustness techniques in the DSR architecture may be achieved in the client e.g. by establishing feature enhancement. At the server side, this can be accomplished e.g. by compensating acoustic models or modified decoding strategies.

For more than a decade research in the DSR area has led to the establishment of a number of ETSI-DSR standards. The first standard [2] for the cepstral features was published in 2000 with the aim of handling the degradation of ASR over mobile channels caused by both lossy speech coding and transmission errors and enabling interoperability over mobile networks. Currently, one of the most well known standards – the ETSI-DSR advanced front-end (AFE) [3] – further includes client-side technqiues providing the DSR system with excellent noise-robustness. However, even given these DSR standards, it is infrequent to find real-life DSR implementations executing in standard mobile devices, thus manifesting a barrier for applying the DSR technology in speech driven applications.

This paper presents a configurable DSR system that has recently been developed at Aalborg University for being deployed into real-life speech driven applications. The AFE is integrated as part of the client and the SPHINX IV [4] is employed as the back-end recogniser. The AFE as used in the system is modified by optimising some time consuming parts of the FFT algorithm. The system is able to support flexible communication to a number of independent user devices each with different requirements to the complexity of the recognition task (e.g. different vocabularies, grammars, etc).

The remainder of this paper is organised as follows. Section 2 presents the system architecture; section 3 describes a number of factors taken into consideration

during system design and implementation and the system evaluation results are provided in section 4. The conclusions are given in section 5.

2. System architecture

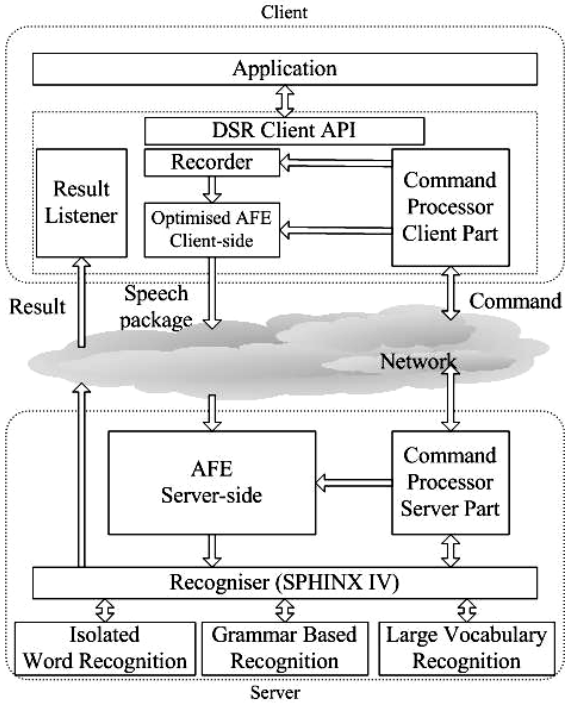

Fig.1 The system architecture

2.1. System architecture

As illustrated in Fig. 1 an embedded *Recorder* in the client simply collects speech signals using a pre-defined sampling rate. The optimised *AFE client-side* module [3] enhances input speech data and generates voice activity detection (VAD) information which together with the set of cepstral features are encoded sequentially and packed into speech packages for network transmission.

At the server side the received speech packages are processed by the *AFE server-side* module. Firstly - on the detection of transmission errors - error concealment is conducted for feature reconstruction. Secondly, the error-corrected speech packages are decoded into a set of cepstral features and VAD information. Subsequently, the cepstral features are processed by the SPHINX speech recogniser. The recogniser presents its result (either the best or N-best results) at the utterance end – detected by the VAD information - and transmits back to the *Result Listener* of the client. To increase system usability and flexibility, three typical recognition modes are represented, namely: *Isolated word recognition, Grammar based recognition* and *Large vocabulary recognition*. Each is defined by a set of prototype files at the server side. The choice is done at system initialisation, and specific settings can be changed at any time. The setting may be different across a group of end-users.

A *Command Processor* is implemented at both the client and server side to support the interchange of configuration commands. Potential commands include control commands to start or stop recognition, choice of recognition mode, commands providing feedback information from the server to a client (e.g. success or failure of any user request), etc.

2.2. Data streams and network load

In the current implementation two network connections are established for each client accessing the system. The first is the data channel for transmitting speech packages and recognition results, and the second is the control channel for the transmission of control commands. Both connections are socket based.

During the recognition process the majority of network load is on the data channel. In full consistency with the DSR standard [3] using an 8 kHz sampling rate, the load on the data channel is about 5.6kbps. The control channel load is small and negligible as compared to the data channel and varies only with the user control settings. With this limited overall bandwidth requirements, the system can be successfully operated over almost all kinds of networks.

2.3. Client interactions with potential applications

The client is implemented in C/C++ for efficiency and portability across different devices and operating systems.

The client is encapsulated into a single dynamical link library (DLL) which supports a series of simple-to-use Application Programming Interfaces (API). For some of them (e.g. functions for acquiring results) different manners are offered. The client has so far been used together with applications written in C, C++, Java or C# demonstrating its compatibility.

Currently available API functions are listed in Fig.2 where they are separated into six clusters covering the following overall functionalities:

- *Initialisation and release functions*: to allocate or de-allocate resources in the client
- *Network functions*: to connect or disconnect a client and the server. The *Connect* function takes three parameters specifying the server IP address, the port number and the recognition mode
- *Grammar control functions*: to configure the active vocabularies or rule grammars in a Java Speech

Grammar Format [9] for the isolated word and grammar-based modes, respectively

- *Recognition control functions*: to start or stop the recogniser at the server
- *Results related functions*: to acquire results. Both synchronous and asynchronous manners are provided for the application. With the synchronous manner, the application inserts a call back function through *SetCallBackFunction* by which it receives a "callback" notification when the result is ready. With the asynchronous manner, the *GetResult* function is called by the application and only returns when the result is ready.

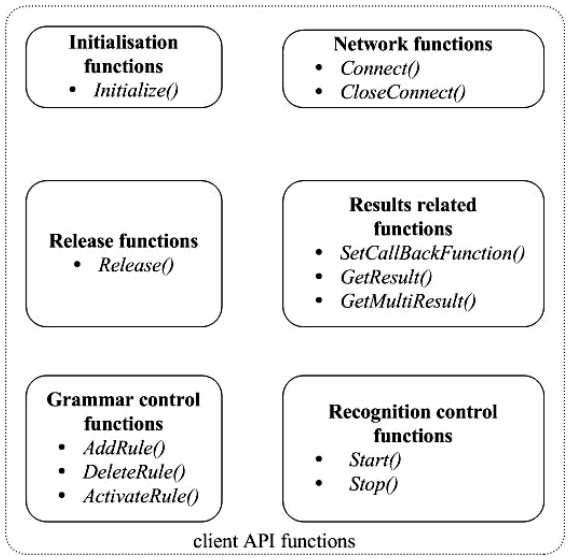

Fig.2 Client API functions

3. System implementation criteria

The design and implementation of the system requires a number of special overall considerations to be accounted for at both the client and the server side.

3.1. Command Control Protocol (CCP) and configurability

Proper communication between the client and the server is supported by the *Command processor* together with a simple communication protocol CCP as illustrated in Fig.3. The CCP consists of a number of control commands that submit the requirements from the client and acquire the feedback from the server. Specifically the client may choose the character set and recognition mode, change recogniser configuration such as grammars, control of recognition start and stop etc. The client may be notified by the server about the success or failure of server actions for each client control command. Each control command is acknowledged to ensure error-free consistent command- transmissions over the command channel.

By means of the CCP the client can flexibly configure the system functionality.

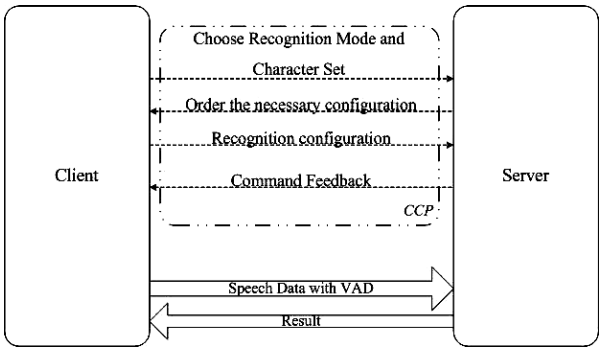

Fig.3 CCP in client-server communication

3.2. Client efficiency

Today the CPU in most mobile devices is only able to conduct floating point arithmetic by replacing it by fixed point calculations. This causes a dramatic drop in computational efficiency. The system presented in this paper integrates a fixed-point AFE implementation. This executes several times faster than the floating-point AFE. It is further optimised for one of the most popular CPUs used in PDAs, the Intel® XScale [5] by substituting high-level language instructions with CPU-dependent ASM instructions within the most time-critical part of the FFT module.

Finally, multi-thread programming [6] is utilised with the goal of utilising the CPU time optimally.

3.3. Choice and optimisation of speech recogniser

The speech recogniser is the primary module in the server controlling not only the recognition performance but also the flexibility and usability of the system. There are several advantages given by the choice of the SPHINX IV recogniser.

- SPHINX IV is based on an object-oriented design which makes its integration with other DSR modules simple
- SPHINX IV supports a wide range of recognition tasks rendering it dynamically configurable
- SPHINX IV is computationally efficient and has shown good recognition performance [4].

However, during system development it is observed that the loading time for acoustic and language models for a large vocabulary speech recognition task is rather long. This has resulted in the optimisation of the Java

source code in parts of the recogniser by using more efficient C code.

3.4. Support of character sets

Given the fact that the ANSI character set is not always supported in mobile devices, both the ANSI and the UNICODE character sets are supported in communicating commands and results between the client and the server.

3.5. Choice of recognition mode

Each of the modes includes a set of pre-trained acoustic models, a SPHINX recogniser configuration file (in XML format) and a vocabulary. Additionally a language model should be contained for the *Large vocabulary recognition* mode. Wordlists (active vocabularies) and grammars are dynamically configured by the application through the APIs.

4. Evaluations

The DSR system described in this paper is evaluated with respect to its recognition accuracy and time efficiency.

4.1. Recognition performance

Four tests have been conducted each on a recogniser with the cepstral features calculated either by a floating point or by a fixed point AFE - with or without VQ. The first two tests use the HTK recogniser [8]. The word models used with these tests are trained using the HTK training software. The latter two tests deploy the SPHINX recogniser. The word models are trained by the SPHINX training tools.

The speech data used for all tests are from the English connected digits recognition task in the Aurora 2 database [7]. Each digit is modelled by 16 HMM states each with three Gaussian mixtures. The acoustic models are trained using the "Multi-condition training" settings in [7] where clean speech and noise data of test Set A are added. The training data thus includes four types of noise ("Subway", "Babble", "Car" and "Exhibition"). The speech features are the normally used 39-dimensional MFCC vector.

The average word accuracies for the four test sessions and across the four set of test data are shown in Table 1.

It is observed that the vector quantised AFE features, only cause the recognition accuracy to drop slightly and rather uniformly across the four types of noise data. With the current setting of the two recognisers, the SPHINX IV shows lower averaged word accuracy as compared to the floating point HTK recogniser. It is noted that the fixed-point SPHINX recogniser gives results that are very close to those resulting from the floating point AFE.

Table 1 Test set A word accuracy (%)

	Subway	Babble	Car	Exhibition	Average
HTK (Floating point)	91.64	90.30	93.77	91.46	**91.79**
HTK (Floating point, VQ)	91.37	90.19	93.48	91.54	**91.65**
SPHINX (Floating point, VQ)	89.71	90.19	91.45	90.19	**90.38**
SPHINX (Fixed point, VQ)	89.73	90.18	91.44	90.17	**90.38**

4.2. Client resource consumptions

Tests have been conducted on PC or PDA devices running either Windows or Windows CE (Pocket PC 2002 or 2003) using wired or WiFi network connections.

The size of the client DLL library file is limited to only about 74Kbytes, and the maximal memory consumption at run-time is less than 29Kbytes.

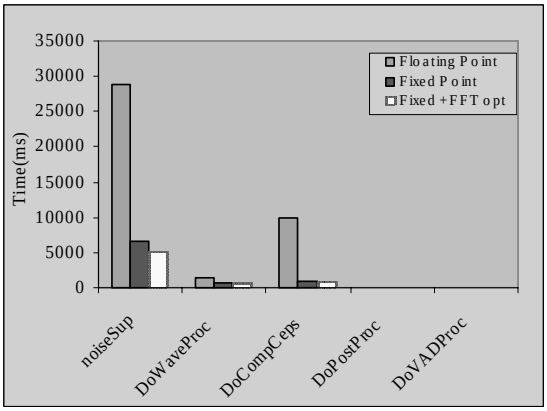

Fig.4 Time efficiency comparisons for the major components in AFE

The optimisation described in section 3.1 is evaluated on a H5550 IPAQ with a 400MHz XScale CPU and 128 MB memory. The data used is an 11-second test sound clip sampled at 8 KHz. The detailed test results are shown in Fig.4 with the comparison of the major components in the AFE, namely noise suppression, waveform processing, cepstral calculation, cepstral post processing and VAD processing [3]. The overall results are provided in Table 2 indicating a large performance reduction in execution time when replacing the floating point AFE with the fixed point algorithm in combination with FFT optimisation.

Table 2 Real-time efficiency in using different realisations of the AFE

Algorithm	Floating Point AFE	Fixed Point AFE	Fixed Point AFE + FFT optimisation
X Real-time	3.98	0.82	0.69

5. Conclusion

This paper introduces a DSR-based system that is developed on the basis of the AFE and the SPHINX IV speech recogniser. The system is designed for being configurable and facilitating simultaneous access from a number of clients each with its own requirements to the recognition task. A system communication protocol is designed to control the interaction between the client and the server. The recogniser supports multiple recognition modes covering isolated word recognition tasks, grammar based recognition tasks, and large vocabulary continuous speech recognition tasks.

The system has been tested and shows good performances both in respect to real-time efficiency and recognition accuracy.

6. Acknowledgement

This project is supported by the CNTK (Centre for Network and Service Convergence) project which is funded partly by the Danish Ministry of Research, Technology and Development and partly by the industrial partners. It includes participations from Danish telecommunication companies and two Danish technical universities. The Authors wish to thank Dr. David Pearce, Motorola Corporation for providing the help for the fixed-point AFE client implementation.

7. References

[1] Z.-H.Tan, P.Dalsgaard and B.Lindberg, "Automatic speech recognition over error-prone wireless networks," to appear in *Speech Communication*, 2005.

[2] ETSI draft standard doc. Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms, ETSI ES 202 108 V1.1.2 (2000-04), April 2000

[3] 3GPP TS 26.243: "ANSI-C code for the Fixed-Point Distributed Speech Recognition Extended. Advanced Front-end", December, 2004

[4] W.Walker, P.Lamere, and P.Kwok et.al. "Sphinx-4: A Flexible Open Source Framework for Speech Recognition", Technical report TR-2004-139, Sun corporation, USA, 2004

[5] Intel® XScale technology overview: http://www.intel.com/design/intelxscale/index.htm

[6] J.Richter. "Programming applications for Microsoft Windows", 4th Edition, Microsoft Press 1999, USA

[7] H.G.Hirsch and D.Pearce, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions", ISCA ITRW ASR2000 (Automatic Speech Recognition: Challenges for the Next Millennium), Paris, France, September 18 - 20, 2000

[8] S.Young. "HTK: Hidden Markov Model Toolkit V1.5". Cambridge Univ. Eng. Dept. Speech Group and Entropic Research Lab. Inc., Washington DC, Dec. 1993

[9] "Grammar Format Specification", Technical documentation, Sun Microsystems, Inc, October, 1998