SPEECH RECOGNITION SYSTEM

SURABHI BANSAL RUCHI BAHETY

ABSTRACT

Speech recognition applications are becoming more and more useful nowadays. Various interactive speech aware applications are available in the market. But they are usually meant for and executed on the traditional general-purpose computers. With growth in the needs for embedded computing and the demand for emerging embedded platforms, it is required that the speech recognition systems (SRS) are available on them too. PDAs and other handheld devices are becoming more and more powerful and affordable as well. It has become possible to run multimedia on these devices. Speech recognition systems emerge as efficient alternatives for such devices where typing becomes difficult attributed to their small screen limitations. This paper characterizes an SR process on PXA27x XScale processor, a widely used platform for handheld devices, and implement it for performing tasks on media files through a Linux media player, Mplayer.

INTRODUCTION

Speech recognition basically means talking to a computer, having it recognize what we are saying, and lastly, doing this in real time. This process fundamentally functions as a pipeline that converts PCM (Pulse Code Modulation) digital audio from a sound card into recognized speech. The elements of the pipeline are:

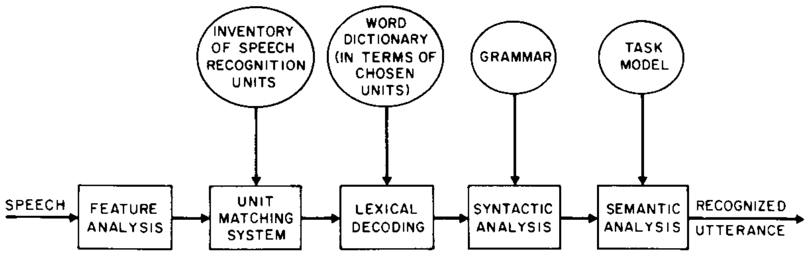

Figure 1: Block diagram of a speech recognizer [16]

- **Transform the PCM digital audio into a better acoustic representation** – The input to speech recognizer is in the form of a stream of amplitudes, sampled at about 16,000 times per second. But audio in this form is not useful for the recognizer. Hence, Fast-Fourier transformations are used to produce graphs of frequency components describing the sound heard for 1/100th of a second. Any sound is then identified by matching it to its closest entry in the database of such graphs, producing a number, called the “feature number” that describes the sound.

- **Unit matching system** provides likelihoods of a match of all sequences of speech recognition units to the input speech. These units may be phones, diphones, syllables or derivative units such as fenones and acoustic units. They may also be whole word units or units corresponding to group of 2 or more words. Each such unit is characterized by some HMM whose parameters are estimated through a training set of speech data.
- **Lexical Decoding** constraints the unit matching system to follow only those search paths sequences whose speech units are present in a word dictionary.
- **Apply a "grammar"** so the speech recognizer knows what phonemes to expect. This further places constraints on the search sequence of unit matching system. A grammar could be anything from a context-free grammar to full-blown English.
- **Figure out which phonemes are spoken –** This is a quite dicey task as different words sound differently as spoken by different persons. Also, background noises from microphone make the recognizer hear a different vector. Thus a probability analysis is done during recognition. A hypothesis is formed based on this analysis. A speech recognizer works by hypothesizing a number of different "states" at once. Each state contains a phoneme with a history of previous phonemes. The hypothesized state with the highest score is used as the final recognition result.

RELATED WORK

A lot of speech aware applications are already there in the market. Various dictation softwares have been developed by Dragon [12], IBM and Philips [13]. Genie [11] is an interactive speech recognition software developed by Microsoft. Various voice navigation applications, one developed by AT&T, allow users to control their computer by voice, like browsing the Internet by voice. Many more applications of this kind are appearing every day. The SPHINX speech recognizer of CMU [1] provides the acoustic as well as the language models used for recognition. It is based on the Hidden Markov Models (HMM). The SONIC recognizer [2] is also one of them, developed by the University of Colorado. There are other recognizers such as XVoice [4] for Linux that take input from IBM's ViaVoice which, now, exists just for Windows.

Background noise is the worst part of a speech recognition process. It confuses the recognizer and makes it unable to hear what it is supposed to. One such recognizer has been devised for robots [17] that, despite of the inevitable motor noises, makes it communicate with the people efficiently. This is made possible by using a noise-type-dependent acoustic model corresponding to a performing motion of robot. Optimizations for speech recognition on a HP SmartBadge IV embedded system [19] has been proposed to reduce the energy consumption while still maintaining the quality of the application. Another such scalable system has been proposed in [18] for DSR (Distributed Speech recognition) by combining it with scalable compression and hence reducing the computational load as well as the bandwidth requirement on the server. Various capabilities of current speech recognizers in the field of telecommunications are described in [20] like Voice Banking and Directory Assistance.

PROJECT DESCRIPTION

Our main goal is to integrate the Mplayer with the SR system to control the functioning of a media file. The block diagram shown below depicts its functioning:

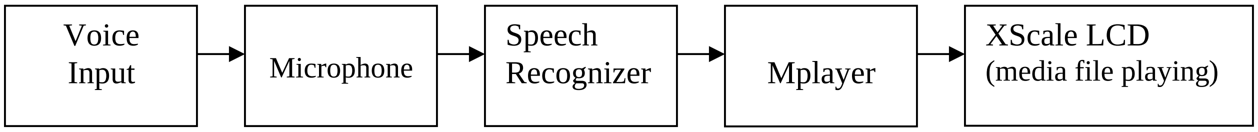

Figure 2: Block diagram for the SRS functioning integrated with Mplayer

The target hardware platform for this work is PXA27X mainstone board. It serves as a prototype for handheld devices. The mainstone board is 208 MHz Intel PXA27X processor, with 64MB of SDRAM, 32MB of flash memory and a quarter-VGA color LCD screen. We chose this particular device because it runs the GNU/Linux R_ operating system, simplifying the initial port of our system. To build our system, a GCC 3.4.3 cross-compiler is used as it is built with the crosstool script. Let us describe each component of our system.

Voice Input: The input is human voice which, as explained before, is sampled at rate of 16,000 per second. It should be given in live mode. But because of some conflicts in the channel settings of the sound card and that used by the software, we are not able to do it in live mode. We are running the recognizer in batch mode, instead, i.e. taking input in the form of a pre-recorded audio file (in RAW format).

Microphone: The microphone that we are using for recognition is built onto the PXA27x platform itself. It has got its own advantages and disadvantages:
Advantages:

- Nothing to plug in.
- User’s hands are free.

Disadvantages:

- Low accuracy unless the user is close to the monitor.
- Not good in a noisy environment.

Speech Recognizer: Platform speed directly affected our choice of a speech recognition system for our work. Though all the members of the SPHINX recognizer family have well-developed programming interfaces, and are actively used by researchers in fields such as spoken dialog systems and computer-assisted learning, we chose the PocketSphinx [9, 10] as our speech decoder which is particularly meant for embedded platforms. It is a version of open-source Sphinx2 speech recognizer which is faster than any other SR system.

We cross-compiled PocketSphinx on XScale and executed the various test scripts present in it. Both the digits and word recognition scripts are up and running on PXA27x. The voice input is supposed to be given from its microphone. PXA27x microphone is set to accept STEREO input only. Whereas, the PocketSphinx speech decoder takes voice input in MONO format only. Due to limitations in the code, we were not able to solve this problem, and hence we are using pre-recorded audio files for this purpose. The SR process decodes these input files, identifies the command and generates output accordingly.

In our application, two input files have been used - ready.raw and stop.raw. The first one is having the utterance - PLAY - that when recognized by SRS, fires a command to the bash asking it to play a media file. It sends the Mplayer command -

```
mplayer inc_160_68.avi
```

The Mplayer starts playing this particular file on the pxa27x LCD display. This is done by creating a child process so that the speech recognition system keeps running to take further inputs. The second command has the utterance - STOP - that kills the playing process.

Challenges

1) The main challenge for us was to identify an efficient SRS, that is able to run on Linux and can be cross-compiled.
2) At first we used SPHINX2 as our decoder. But some of the scripts present in it were not working on pxa27x because of their dependence on PERL which is very difficult to cross-compile for XScale. Patches are available to cross-compile it for Strong-arm processor. Even the authors of those patches are not able to answer how to do it on XScale.
3) Pocketsphinx supports mono channel audio as the input. The audio codec AC97 has some lines of code which hardcodes the channel in stereo without considering the mono channel. The biggest reason for not working with livemode is this as the number of samples produced in stereo are twice than it is required, causing the recognition to fail.
4) There were some memory problems which we faced while we were running the software on the mainstone board. For this, we had to set parameters like **–mmap** option which prevented memory overflows.

Limitations

1. The dictionary of PocketSphinx is huge and needs to be optimized.
2. Our dictionary was supposed to contain four words – PLAY, PAUSE, STOP and REPEAT that need to be pronounced in a specific way to enable the SRS to recognize them. Among these, we are only able to implement PLAY and STOP commands for a **single** media file.
3. The conflict between the software and sound card channel settings is a big problem for this system.

RESULTS

INPUT	***OUTPUT***
ready.RAW	Mplayer starts playing the media file inc_160_68.avi
stop.RAW	Mplayer stops

CONCLUSION AND FUTURE ENHANCEMENTS

The speech recognizer that we chose for pxa27x, PocketSphinx, is the first open-source embedded SR system that is capable of real-time, medium-vocabulary continuous speech recognition. We are able to recognize both digits and words on the board. We tried our best to run this system in live mode, but because of the software limitations (i.e. not able to accept stereo input), a conflict between the software and the sound card input instead, we are only able to do that in batch mode.

This work can be taken way too far from the state in which it is now. The current software doesn't support stereo channel. This can be made more adaptable for any kind of channel changes.

The author of this software is going to address the channel conflict problem in the next version, where every alternate sample will be dropped so that we have enough number of samples for mono mode which is the main requirement for PocketSphinx to work in live mode. Also, in the present work, we have hardcoded the filename which we want to play. This can be further programmed to take input from the user himself as to what file he wants to play. Besides all these, this application can be deployed for use on other handheld and mobile devices.

REFERENCES

[1] Kai-Fu Lee, Hsiao-Wuen Hon, and Raj Reddy, *An Overview of the SPHINX Speech Recognition System*. IEEE Transactions on Acoustics, Speech and Signal Processing,
[2] Pellom, B., *Sonic: The University of Colorado Continuous Speech Recognition System*.
[3] http://www.tldp.org/HOWTO/Speech-Recognition-HOWTO/index.html
[4] http://www.zachary.com/s/xvoice
[5] http://xvoice.sourceforge.net/
[6] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, Joe Woelfel, *Sphinx-4: A Flexible Open Source Framework for Speech Recognition*.
[7] A. Hagen, D. A. Connors, B. L. Pellom, *The Analysis and Design of Architecture Systems for Speech Recognition on Modern Handheld-Computing Devices.*
[8] http://project.uet.itgo.com/speech.htm
[9] David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alex I. Rudnicky, *PocketSphinx: A Free, Real-Time Continuous Speech Recognition System for handheld devices*.
[10] http://www.pocketsphinx.org
[11] http://www.microsoft.com/intdev/agent/
[12] http://www.dragonsys.com
[13] http://www.speech.be.philips.com/index.htm
[14] Ben Shneiderman, *The Limits of Speech Recognition.*
[15] Stefan Eickeler, K. Biatov, Martha Larson, J. Kohler, *Two Novel Applications of Speech Recognition Methods for Robust Spoken Document Retrieval.*
[16] Lawrence R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.*
[17] Yoshitaka Nishimura, Mikio Nakano, Kazuhiro Nakadai, *Speech Recognition for a Robot under its Motor Noises by Selective Application of Missing Feature Theory and MLLR.*
[18] Naveen Srinivasamurthy, Antonio Ortega, Shrikanth Narayanan, *Efficient Scalable Speech Compression for Scalable Speech Recognition.*
[19] Brian Delaney, Tajana Simunic, Nikil Jayant, *Energy Aware Distributed Speech Recognition for Wireless Mobile Devices.*
[20] Lawrence R. Rabiner, *Applications of Speech Recognition in the Area of Telecommunications.*