

# SPEECH RECOGNITION OVER THE INTERNET USING JAVA

Zhemín Tu and Philipos C. Loizou

Department of Applied Science  
University of Arkansas at Little Rock  
Little Rock, AR 72204

zxtu@ualr.edu, loizou@ualr.edu -- <http://giles.ualr.edu/asd/speech>

## ABSTRACT

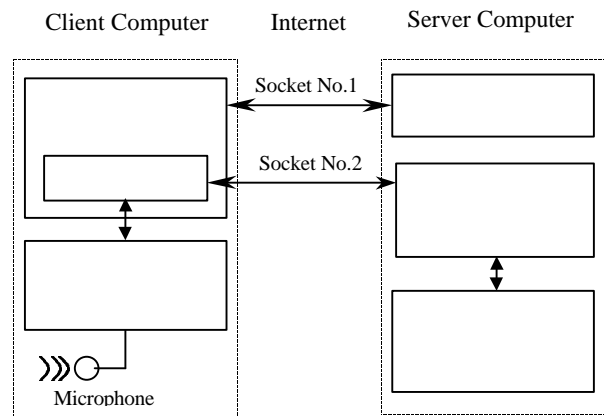
A speech recognition system based on an Internet client-server model is presented in this paper. A Java applet records the voice at the client computer, sends the recorded speech file over the Internet, and the server computer recognizes the speech and displays the recognized text back to the user. Using this structure, an isolated digit recognition application was realized.

## 1. INTRODUCTION

With the explosive growth of the Internet technology, both speech researchers and computer software engineers have been putting a great deal of effort to integrate speech functions into Internet applications [1-3]. For simple applications, voice playback and voice recording functions may be sufficient. For complex applications, however, speech recognition and synthesis functions are needed.

This paper presents a client-server based, speech recognition system. In this system, a user can use a World Wide Web (WWW) browser such as the *Netscape Communicator* to do speech recognition by visiting a particular web site. The architecture of the system is shown in Figure 1. In the client computer, a microphone is needed to record speech. Since Sun Microsystems' Java Development Kit (JDK) currently does not support voice recording, we had to use a third party product for recording. A local process is responsible for recording the speech voice and transmitting the speech data to the applet, which is downloaded and run by the web browser of the client computer. The server computer contains a web server where a WWW page is held, a speech recognition server, which is responsible for speech recognition management, and a speech recognizer which recognizes speech.

No coding method [1] was used to compress the speech data before transmission over the Internet, since we felt that a single speech utterance would not be a heavy burden on the network.



**Figure 1.** Block diagram of the client-server based speech recognition system. The left block represents the structure of the client computer, while the right block represents the structure of the server computer. The Internet connects the server and the client computer. Two sockets are used for web browser and speech recognition, respectively.

The remainder of this paper is organized as follows. In Section 2 we review alternative system structures for the implementation of speech-enabled applications over the WWW. In Section 3, we describe the implementation of our JAVA-based speech recognition system. In Section 4 we present some of our experimental results, and we conclude in Section 5.

## 2. System Structures

There are several architectures for incorporating speech applications on the WWW [1], two of which are reviewed here.

### 2.1 Stand-alone Structure

In the stand-alone architecture, the client computer needs a copy of the stand-alone program, which is used to record voice, send voice data and display the recognized text. This stand-alone program is also responsible for

showing the recognition results to users through its own user-machine interface.

The benefit of using a stand-alone model is that both hardware and software resources of the client computer are free to be used. This is convenient for the client program to control a soundcard. Given that currently there is no JDK supporting voice recording, the above benefit will be attractive until *Sun Microsystems, Inc.* puts forward its new JDK with voice recording capability.

The stand-alone model has also disadvantages.

First, there is the upgrading problem. The users will need to keep downloading the latest version of the software, or need to change the settings frequently. Researchers are developing speech recognition algorithms all the time. When it is time to upgrade the speech recognition server, it is very likely that the client software will also need upgrading.

Second, there is the portability problem. People who are using different computer platforms have to download different kind of client software, and have to set up the software by themselves. The portability problem also costs the system developer much more money because he has to provide several sets of software for different platform users.

## 2.2 Browser-embedded Structure

Alternatively, we can realize the client-server based speech recognition system using a browser-embedded model. In this system, the user can use a WWW browser to access the speech recognition server, and then perform the speech recognition task. The Java applet is responsible for recording voice, transferring speech data, and displaying the recognized text.

Benefits of the browser-embedded model include:

- The system is easily upgradable. No extra work is needed from the users. If the system designer wants to improve the system performance, he is free to upgrade the speech recognition program. If the client program needs to be changed, then the system designer revises the applet, and puts the revised applet to the web server. Next time the users visit the web site through their web browsers, the browsers will download the new applet, and execute it. From the users' point of view nothing has changed in the system except for the interface.
- The system is portable. Java language is portable across different computer platforms. Therefore the software provider does not need to provide different software to different platform users. All the provider

needs is to put the Java applet, which acts as the client program, into the web server directory. Compared with the stand-alone model, the browser embedded model is more economical in practice.

The main disadvantage of the browser embedded model is that currently the Java applets do not support voice recording. This is mostly because of security reasons. For security reasons, the applet can not run a program located in a local machine, or control a hardware device such as the soundcard. Imagine that when a user browses a web site, a hidden applet can turn on the user's microphone and record the user's conversation with other people in the same office. There is a market demand, however, for the voice recording function in applets. More and more network voice systems are likely to be moved into web browsers. Those voice systems will include network voice communication packages, remote production applications, and the system presented in this paper.

The browser-embedded model is also easily scaled in the sense that the system hardware or software can be extended without any restrictions. This is necessary for the server-client based speech recognition system because the database which contains the HMM models of all the words in the vocabulary may need to be upgraded.

Given the above advantages of the browser-embedded structure, we decided to adopt it for the server-client based speech recognition system.

## 3. Implementation of JAVA-based speech recognition system

The block diagram for the whole system is shown in Figure 1. In the client computer, a microphone is needed to record speech. A local process is responsible for recording speech and transmitting the speech data to the applet, which is downloaded and run by the web browser of the client computer. The server computer contains a web server where a WWW page is held, a speech recognition server which is responsible for speech recognition management, and a speech recognizer which recognizes speech. The speech recognizer is invoked by the speech recognition server.

Two TCP/IP sockets are used in this system. One of them is responsible for web browsing. The socket is a connection between the web browser, which is located in a client computer, and the web server. This socket is only responsible for transmitting data related to web browsing. The other socket is a connection between the applet and the speech recognition server. This socket is responsible for transmitting speech data from the applet to the speech

recognition server, and displaying the recognized text back from the speech recognition server to the applet.

For security reasons, Java requires that the applet can only open a connection back to the same IP address where the web server stays. This means that the speech recognition server has to be at the same IP address as the web server. More realistically, they have to be on the same computer.

As shown in Figure 1, the server-client based speech recognition system can be divided into several components. On the client side, there is the web browser, the applet and the voice-recording process. On the server side, there is the web server, the speech recognition server and the speech recognizer.

Among these components, the web browser, the voice-recording processes and the web server can be found from commercial products. The components that had to be implemented by the system designer were the speech recognition server, the speech recognizer and the Java applet.

### 3.1 The Java Applet

The Java applet provides the interface for speech recognition users, records the users' voice, transmits speech data to the speech recognition server, and displays the recognized text to the users (see Figure 2).

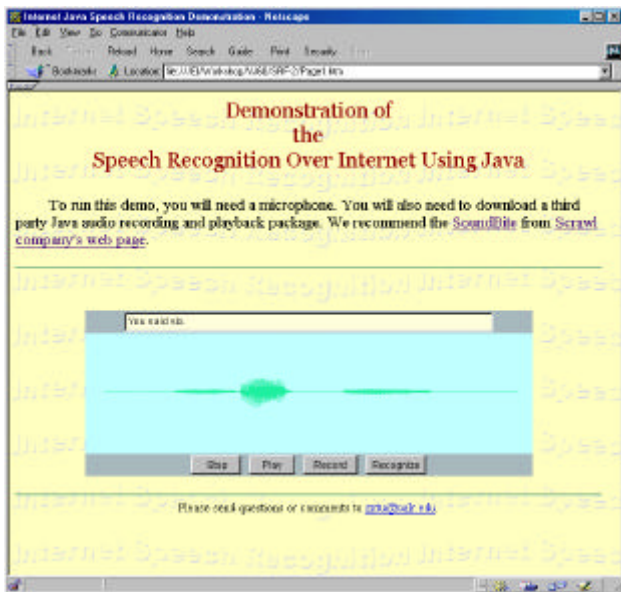


Figure 2. The user interface of the Java-based speech recognition system.

Since there is currently no JDK version supporting applet voice recording, a third party program from

*Scrawl, Inc.*, which can be downloaded from <http://www.scrawl.com>, had to be used to perform sound recording. The third party program is invoked by the applet when sound recording is needed. The applet was designed to be simple so that it can be run on most client computers. Once the applet begins to run, it sends its request to a certain port located in the server computer in order to establish a socket. The recognition server is monitoring the port, and after that, the server and the client establish a connection with each other.

### 3.2 The Speech Recognition Server

The speech recognition server is running on the host computer all the time, just like the web server. As soon as the recognition server is started, a socket is created. The recognition server keeps listening to the socket until a client requests to connect to the server.

Once a user visits the speech recognition web page through a web browser, the web browser downloads the applet contained in the web server and runs it. The applet then requests to create a socket between the speech recognition server and itself. After the recognition server receives the request, it generates a new thread, in which a new socket is created. Then, the socket takes care of data communication between the recognition server and the client.

There are a few protocols between the recognition server and the applet that need to be maintained. For example, before an applet transmits the speech data to the server, it has to tell the server how much data is going to be transmitted. This is important because otherwise the server would not know where the end of the speech data is. The server keeps reading the socket, and blocks it.

The speech recognition server is also responsible for invoking the speech recognizer after it receives all the speech data and stores it to the server's hard disk. This is realized by a system call. The speech recognition server and the speech recognition processes communicate with each other by reading and writing files in the server's local hard disks.

### 3.3 The Speech Recognizer

To demonstrate the Java-based speech recognition system, we used a small-vocabulary, isolated word recognition task, and in particular the digit recognition task. The core of the recognizer was based on continuous density HMMs.

The Texas Instrument's speaker-independent digits (TIDIGITS) database was used to train whole-word HMMs. A total of 50 speakers was used for training. Each speaker said the digits 0-9, and "oh" twice. Unlike the

TIMIT database, the TIDIGITS database did not provide phonetic transcriptions for each digit. Therefore, the digits had to be manually segmented in our lab. A semi-automatic program was used to segment the TIDIGITS speech database. An endpoint detector first segmented the speech data automatically, and then displayed the segmentation. If the segmentation was not correct, the user indicated the correct endpoints using a mouse.

Mel-frequency cepstrum coefficients (MFCC) were used for features. Speech was first segmented into 25.6-msec frames and then pre-emphasized with a first order FIR filter of the form  $(1-0.97z^{-1})$ . A Hamming window was applied every 10 msec, with a 15.6-msec overlap between consecutive frames. In each frame, 14 mel-frequency cepstrum coefficients were computed by applying 24 triangular filters on the FFT magnitude spectrum, and then computing the discrete cosine transform of the log-filterbank energies. A 30<sup>th</sup> dimension feature vector consisting of 14 MFCC coefficients, 14 delta-MFCC coefficients, normalized energy and delta-energy was used to train the HMMs.

The training procedure consisted of 10 iterations of the segmental k-means algorithm [4] over all training data in order to get initial estimates for the word models, followed by five iterations of the Baum-Welch algorithm. A six-state left-to-right continuous HMM with 4 mixtures was used for each of the 11 digits. Diagonal covariance matrices were used for all models. Recognition was carried out using the Viterbi algorithm.

The training was done first on a Unix workstation, and the HMM models were then transferred to the server computer. The only programs running on the server computer were the feature extraction program and the Viterbi recognizer.

#### 4. System Performance

The speech recognizer, running on the server computer was first tested (off-line) on the TIDIGITS database. Digits produced by twenty additional speakers from the TIDIGITS database, not included in training, were used for testing. The word error rate was 2%. The speech recognizer was also tested on 10 different speakers (from our lab) who used the Java-based speech recognition system. The resulting word error rate was 4%. The fact that the error rate of the Java-based system was higher than the error rate obtained using digits from the TIDIGITS was not surprising given the differences in recording conditions. More research needs to be done to make the Java-based system more robust to different microphones and different background environments.

#### 5. SUMMARY

A Java-based speech recognition system based on a client-server model was presented. The Java applet records the voice at the client computer, sends the recorded speech file over the Internet, and the server computer recognizes the speech and displays the recognized text back to the user. The Java applet can be accessed from our web site at: <http://giles.ualr.edu/asd/speech>.

#### 6. ACKNOWLEDGMENTS

This research was partly supported by a grant from the Arkansas Science and Technology Authority.

#### 7. REFERENCES

- [1] V. Digalakis, L. Neumeyer and M. Perakakis "Quantization of Cepstral Parameters for Speech Recognition Over the World Wide Web," *Proceedings ICASSP 98*, pp. 989-992, 1998.
- [2] D. Goddeau, W. Goldenthal and C. Weikart, "Deploying speech applications over the Web," *Proc. Eurospeech*, pp. 685-688, September 1997.
- [3] S. Bayer, "Embedding speech in Web interfaces," *Proc. ICSLP*, pp. 1684-1687, October 1996.
- [4] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.