

SERVER-ASSISTED SPEECH RECOGNITION OVER THE INTERNET

*Aldebaro Klautau** *Nikola Jevtić* *Alon Orlitsky†*

ECE Department, UCSD
9500 Gilman Drive
La Jolla, CA 92093, USA
<http://speech.ucsd.edu>

ABSTRACT

We propose a new architecture for deploying speech recognition over the Internet. The client performs the recognition, but is assisted by the server who computes the speech parameters. To demonstrate the architecture, we developed a Java-based Web-navigation system where the precomputed HMM models of the hyperlinked words are stored on the Web page and downloaded by the client. We tested the system on a digit-recognition example. The results show that with quantization and compression of the speech parameters, good recognition can be achieved in acceptable download and calculation time even on clients with modest connection speeds and computational powers.

1. INTRODUCTION

Automatic speech recognition (ASR) has many Internet applications: it can be used for Web navigation, data entry, database access, browser and applet control, and for general enhancement of the Internet experience.

Consequently, a considerable amount of work has been devoted to Web-related ASR (Web-ASR) research and development. Some Web-ASR applications already exist while others are being developed (e.g., [1-9]). Recently, the WWW Consortium has started a work group dedicated to network speech applications [10], and the first specification of VoiceXML [11], an extensible markup language for incorporation of voice services into applications, was released.

Web-ASR architectures fall in two groups, based on who performs the recognition. In server-based systems, the client sends the speech signal to the server for recognition, whereas in client-based systems, the client performs the recognition task.

Server-based architectures are useful when the client does not have the capability to interpret the speech. For example when an ASR system does not exist on the client machine, or the system exists but cannot handle the relevant language or vocabulary set. Client-based applications are preferable when there may be many recognition requests that could overload a central server or when the round-trip time is significant.

In this paper, we propose a variant of client-based architectures. The client still performs the recognition, but the server assists the client by computing speech parameters, e.g., word or phoneme models, which he then sends to the client. This architecture is mainly useful for applications that do not require a large vocabulary and are speaker-independent.

To reduce computation, the server precalculates the models associated with the active vocabulary of each Web page just once. He stores them on the page along with the corresponding text and this combined information is downloaded by the client. The architecture is illustrated in Figure 1.

This *server-assisted* Web-ASR architecture combines advantages of the two pure approaches. As with client-based systems, it avoids overloading the server. And like server-based architectures, it lets the client support multiple languages and vocabulary sets without the specialized software required for each.

The architecture has potential drawbacks as well. It is mainly applicable to speaker-independent systems. The server must be able to compute the speech parameters. And, if the active vocabulary is large, downloading the models may take too long. This last concern is further addressed in Section 3 where we show that if the parameters are properly quantized and compressed, downloading the models for moderate vocabulary should not be a problem.

Precursors of this architecture appeared in [4, 9] where the client assists the server by computing some of the speech features and in [2] where grammar informa-

* Supported by CAPES, Brazil.

† Supported by NSF grant 9815018.

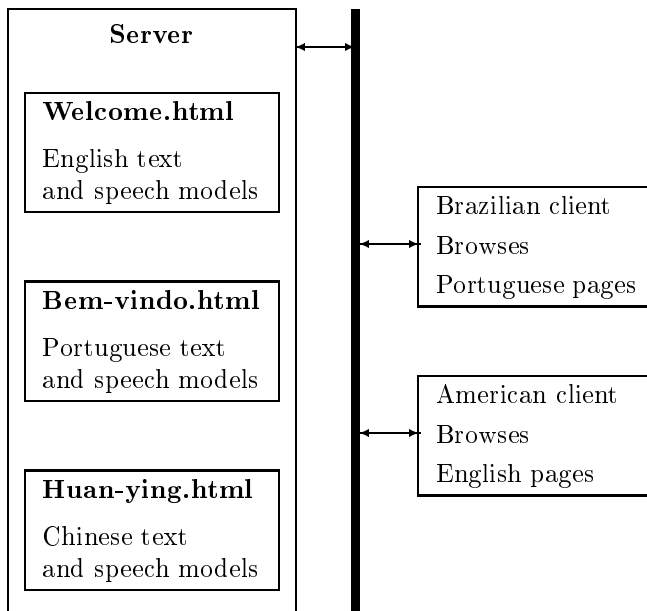


Figure 1: Example of a server-assisted system.

tion is sent to the client. In our proposed architecture, however, the server computes actual speech parameters that could also be computed by the client.

To demonstrate the architecture's feasibility and to evaluate the download time, we used it to implement a speech-based Web navigation system. The user views the Web page and follows the hyperlinks by simply saying the underlined word associated with the link. The server precomputes the word models of all the underlined words and stores them on the page along with all other link information. When the client downloads the page, he extracts the word models and uses them to associate speech with the next site location.

The next section describes the general concepts involved in the implementation. Section 3 describes a specific example for digit recognition and test results showing that good recognition can be achieved with very moderate download time.

2. IMPLEMENTATION

This section describes the concepts involved with implementing the architecture. For simplicity, we consider a system for isolated word recognition. Specifically, we show how to construct a speech-based Web navigation system where the user views the Web page and follows the hyperlinks by simply saying the underlined word associated with the link. The server precomputes the word models of all the underlined words,

and stores them on the page along with all other link information. When the client downloads the page, he extracts the word models and uses them to associate speech with the next site location. A demonstration is available on our Web site [12].

This application is suitable for our architecture because it requires only isolated word recognition, simplifying the demonstration task, and at any given time the number of links, or active words, is kept small. However, the system can be extended to other applications, including connected speech, if the grammar perplexity [13] is not large, namely, few word models are introduced for each new context (e.g., a new Web document).

The code consists of two parts. The resident server part which trains and stores the word models, and a mobile client part which resides on the client and lets him acquire the speech and perform the recognition.

We begin by describing the server code that is responsible for training the models. For every page, the active words are identified either automatically, or manually by the designer. These are the words that the system will recognize for this page. For each active word, the program creates a model, e.g., by consulting a speech database. The models are then stored on the page along with the associated words.

The client code contains procedures for speech acquisition, endpoint detection, feature extraction, and speech recognition.

The client needs to download his part of the code. This can be done using two standard methods: stand-alone and browser embedded [5]. In stand-alone, the software is obtained separately, either downloaded through the net or through a storage device. In browser-embedded implementations the software is downloaded by the browser when needed.

The proposed architecture can be combined with either methodology. For concreteness we assume here that the software is browser embedded. Hence it is downloaded automatically by the browser when he visits a speech-enabled page for the first time.

Once the client software is in place, the client can recognize speech. Every downloaded page contains the embedded word models. (Alternatively, only previously unencountered models are downloaded.) The client uses his software and the downloaded models to perform the recognition.

Since many models may need to be downloaded, one of the main research challenges is to compress the pre-computed parameters so that they can be downloaded quickly and efficiently. Some preliminary work on the subject is described below.

The next section describes a specific implementation

of such a system for digit recognition.

3. EXAMPLE: DIGIT RECOGNITION

The previous section outlined a general server-assisted browser-embedded isolated word recognition system. We now describe in detail a digit recognition example which we implemented to demonstrate the architecture's feasibility.

The system recognizes the English digits 0 (pronounced "zero") to 9. As indicated below, the initial software download takes about 8 seconds and page download takes roughly 3 seconds assuming a 28,8 *Kbps* connection. The system was tested on real-time and achieved good results. When tested off-line with ten speakers, it performed without errors.

To enable its deployment over the Internet, the system was written exclusively in Java.

The models are continuous left-right hidden Markov models (HMM's) with $N = 5$ states and no skips. Each state is represented by a mixture of $M = 5$ Gaussians with a diagonal covariance matrix. They are created using conventional training techniques [13]. The initial models are estimated using a variant of the K -means algorithm combined with Viterbi alignment, and are then iteratively improved using the Baum-Welch algorithm.

The models were built from the TIDIGITS database [14]. 46 speakers were used for training, each contributing one sample of every digit.

The signal was filtered and downsampled to 8 *KHz*. It was then segmented into frames, 30 *ms* long and overlapped by 20 *ms*.

Each frame is converted to $Q = 39$ features: 12 mel-frequency cepstral coefficients (MFCCs) [15], energy, and their first and second derivatives.

The recognition uses a Viterbi algorithm to find the best model. Ten new people were used for testing. The system achieved 0 error as would be expected from the simplicity of the task and testing procedure. This indicates that a simpler HMM configuration could be adopted, but our objective was not to completely optimize the system.

For real-time recognition, the endpoints are identified by a voice activity detection (VAD) algorithm which is based on energy and zero-crossing rate.

We tested the system operation on various PC computers. It performed well even when tested on a 166 *MHz* Pentium processor with 32 *MB* of RAM. The average recognition time was 200 *ms* in this case.

The total size of the ASR code, compressed in JAR format, is 26 *KB*. Hence should take about 8 seconds to

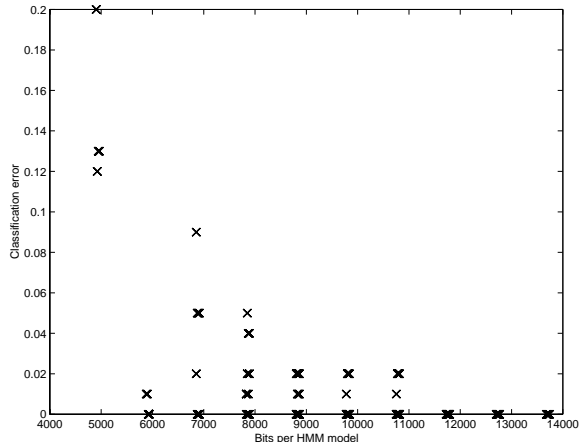


Figure 2: Classification error vs. quantization rate.

download using a 28.8 *Kbps* connection. This download should occur just once.

In all subsequent interactions, only the HMM models must be transmitted. This requires

$$(N - 1) + N(M - 1) + 2NQM$$

parameters per model. In our example, this translates into 1974 parameters per model.

A simple 4-byte representation of each parameter would require about 8 *KB* per model, hence roughly 80 *KB*, or 640 *Kb* for all ten digits. This would take roughly 22 seconds using a 28.8 *Kbps* connection.

To reduce download time, we quantize the parameters. The transition probability and weight parameters are quantized uniformly and the Gaussians parameters are quantized uniformly in the log domain. Different number of bits are allocated to different parameter types.

We evaluated the recognizer's error rate with different choices of bit allocations and quantization levels. Figure 2 shows a scatter plot of the error rate vs. the number of bits allocated to each HMM model using various quantization schemes.

As the figure shows, even with 6 *Kb* per HMM, there are quantization schemes with no classification errors. Using these schemes we can compress all 10 digit models to just 60 *Kb*, which take roughly 2.5 seconds to download on a 28.8 *Kbps* connection.

4. SUMMARY

We presented a new architecture for deploying speech recognition over the Internet. In this architecture, the client performs the recognition, but the server assists the client by precomputing the speech parameters.

The architecture combines advantages of server- and client-based architectures. In particular, it avoids server overload and allows for clients to download the software quickly and recognize multiple languages and vocabulary sets.

We showed how the architecture can be applied to build an isolated word recognition system, and implemented and tested a digit recognizer. The results show that good recognition can be obtained with very short download and recognition times even with moderate connection speeds and computational capabilities.

Possible extensions include noise modeling for a more robust recognition, and better compression of the HMM parameters for further download speedup. In particular, one can improve our quantization by exploiting correlations between the features, and by varying the quantization accuracy within each parameter type (e.g., [4, 9]).

REFERENCES

- [1] <http://www.speech.sri.com/demos/atis.html>.
- [2] K. Kondo and C. Hemphill. A WWW browser using speech recognition and its evaluation. *Systems and Computers in Japan*, pages 57–67, Sep. 1998.
- [3] S. Bayer. Embedding speech in web interfaces. *International Conference on Spoken Language Processing*, pages 1684–7, Oct. 1996.
- [4] V. Digalakis, L. Neumeyer, and M. Perakakis. Quantization of cepstral parameters for speech recognition over the World Wide Web. *IEEE Journal on Sel. Areas in Communications*, pages 82–90, Jan. 1999.
- [5] Z. Tu and P. Loizou. Speech recognition over the Internet using Java. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, AZ:2367–70, Mar. 1999.
- [6] D. Goddeau, W. Goldenthal, and C. Weikart. Deploying speech applications over the Web. *European Conference on Speech Communication and Technology*, pages 685–688, Sep. 1997.
- [7] C. T. Hemphill and Y. K. Muthusamy. Developing Web-based speech applications. *European Conference on Speech Communication and Technology*, pages 895–898, Sep. 1997.
- [8] D. Vaufreydaz, J. Rouillard, and M. Akbar. A network architecture for building applications that use speech recognition and/or synthesis. *European Conference on Speech Communication and Technology*, pages 2159–62, Sep. 1999.
- [9] G. Ramaswamy and P. Gopalakrishnan. Compression of acoustic features for speech recognition in network environments. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 977–980, June 1998.
- [10] <http://www.w3.org/voice>.
- [11] <http://www.voicexml.org>.
- [12] <http://speech.ucsd.edu/demos>.
- [13] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.
- [14] <http://www ldc.upenn.edu>.
- [15] S. Davis and P. Mermelsteing. Comparison of parametric representations of monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoustic., Speech, Signal Processing*, pages 357–66, Aug. 1980.