QUANTIZATION OF CEPSTRAL PARAMETERS FOR SPEECH RECOGNITION OVER THE WORLD WIDE WEB

V. Digalakis[1,2], L. Neumeyer[2] and M. Perakakis[1]

(1) Dept. of Electronics and Computer Engineering
Technical University of Crete
Hania, 73100, GREECE

(2) SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025, USA

ABSTRACT

We examine alternative architectures for a client-server model of speech-enabled applications over the World Wide Web. We compare a server-only processing model, where the client encodes and transmits the speech signal to the server, to a model where the recognition front end, implemented as a Java applet, runs locally at the client and encodes and transmits the cepstral coefficients to the recognition server over the Internet. We follow a novel encoding paradigm, trying to maximize recognition performance instead of perceptual reproduction, and we find that by transmitting the cepstral coefficients we can achieve significantly higher recognition performance at a fraction of the bit rate required when encoding the speech signal directly.

1. INTRODUCTION

Motivated by the explosive growth of the Internet, speech researchers have been working on the integration of speech technologies into the World Wide Web (WWW) [1-5]. Applications include Internet telephony, speech-enabled browsers, speech and natural language understanding systems, and speaker verification. Developers have succesfully adapted existing systems, or created new ones, that can be deployed over the WWW.

In this paper we consider a client-server speech recognition system. We assume that communication channels between the client and the server may have limited bandwidth. That would be a realistic assumption in applications that communicate over the Internet or through wireless channels. The architecture is shown in Figure 1. A central server provides speech recognition services. The clients are deployed on heterogeneous environments, such as personal computers, smart devices, and mobile devices. Speech is captured by the clients, and after some local processing, the information is sent to the server. The server recognizes the speech according to an application framework and sends the result string or action back to the client.

Esentially, this system uses two major speech technologies: speech recognition and speech coding. In a complex dialog system, coding would be required to present audio prompts to the user. Standard coding techniques can be used to send the speech over low-bandwidth channels and produce perceptually acceptable speech to the user. In this paper, however, we focus on the opposite path, that is, the speech data sent from the client to the server.

Traditional speech coding research focuses on the performance trade-off between transmission rates and perceptual reproduction quality. To achieve this goal, several succesful techniques have been developed, resulting in dramatic technological advances. The data compression problem for state-of-the-art hidden Markov model (HMM) based speech recognition systems differs from the traditional speech coding problem in that the optimization criterion is recognition accuracy instead of perceptual quality of the reproduced data. In addition to the practical goal of developing a client-server architecture, we also have an interest in understanding how much and what information is actually being modeled by the HMMs. Understanding what data is redundant in the representation of the speech signal may open the door to new ideas on how to better model it.

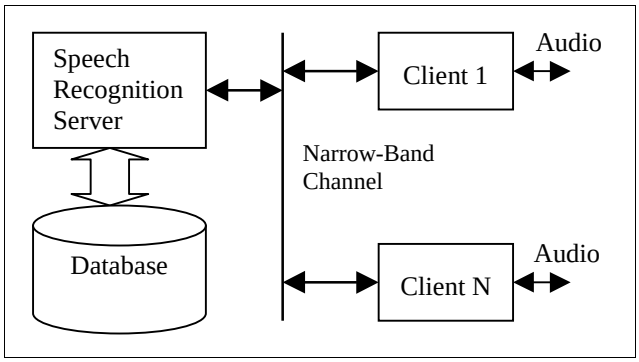

Figure 1 Client-server speech recognition system.

The remainder of this paper is organized as follows. In Section 2 we review alternative architectures for the implementation of speech-enabled applications over the WWW. In Section 3 we discuss the coding of the front-end feature vectors at the client side. In Section 4 we present our experimental results, and we summarize and conclude in Section 5.

2. SYSTEM ARCHITECTURES

There are several alternative architectures for applications incorporating speech recognition technology on the WWW, three of which are examined here. The first strategy is to not do any processing related to the recognition/understanding process at the client side, and simply transmit the user's voice to the server. The second alternative is to do most of the speech recognition processing at the client side, and then transmit the result to the server. Finally, an intermediate solution is to do only the front-end processing at the client and transmit only the information that the recognizer needs through the network.

2.1 Server-only Processing

When all the recognition processing takes place at the server side, we have the smallest computational and memory requirements on the type of client machines that can access the speech-enabled application. Speech can be transmitted to the server either through the Internet, by using some traditional speech coding techniques, or via a second channel, such as the telephone. An example of the Internet-based transmission is the approach followed by DEC, using a voice plug-in [1]. The disadvantage of this approach is that the user cannot access these applications through low-bandwidth connections, since, as we shall see in Section 3, recognition performance degrades for rates below 32 Kbps (kilobits per second). In low-bandwidth connections, voice can be transmitted to the server by a telephone line. This approach also degrades performance, since, in general, recognition performance is lower in toll-quality than in high-quality data. It is also inconvenient (the user is typically prompted by the application to dial a telephone number, which in the case of modem-based connections may not exist), and adds the cost of the telephone connection to the user. It was, however, followed by early applications [2, 3]. This approach was attractive in the beginning because it overcame problems associated with audio capture and transmission standards.

2.2 Client-only Processing

A different strategy is to run the recognition and understanding engines at the client machine. The obvious advantages are that a high-bandwidth connection is not required, and that recognition can be based on high-quality speech, since the sampling and feature extraction takes place at the client side. The system is also less dependent on the transmission channel and therefore more reliable. This approach, however, limits significantly the types of clients that the speech-enabled application can support, since they must be powerful enough to perform the heavy computation that takes place in the recognition process. In addition, local processing may not be desirable for certain types of applications, like speaker verification [5]. Applications based on dynamic and complex grammars that require rapid database access are also not good candidates for a client-only architecture.

2.3 Client-Server Processing

This approach is based on two key observations:

- The feature extraction is only a small part of the computation that takes place in a speech recognition and understanding application.
- Speech recognition needs only a small part of the information that the speech signal carries.

The first observation implies that we can run the front-end processing (the feature extraction) at the client side on a much wider range of machines than the ones that will support the whole recognition process. There are additional advantages of client-server processing over the client-only model. The recognizer may need information that exists on the server side in order to guide the decoding process; this information would have to be transmitted to the client in the client-only model, something unnecessary in the client-server model since the decoding takes place at the server side. To make speech recognition servers available from a variety of systems, front-end processing and compression can be standardized. Standard front-end modules can be installed on the client machines as a system resource, a Java applet, or a browser plug-in.

Our second observation clearly shows the advantage of client-server processing over the server-only model. Traditional speech coding focuses on the perceptual reproduction quality of the coded speech. As a result, the speech coder may transmit redundant information, and at the same time introduce noise to the features that are important in the recognition process because of bandwidth limitations. When the objective is to transmit the speech to a recognition server, there is a clear shift in the speech coding paradigm, and the objective of the coding process should be recognition accuracy. If the information used by the recognition process is contained in a set of features, then only this set of features needs to be compressed and transmitted to the server. For example, typical state-of-the-art speech recognizers represent the vocal tract information using a set of the first few cepstral coefficients. In view of our objective, we should expect a significant reduction in bit rate if we encode this set of cepstral features, as opposed to encoding the speech signal itself.

3. CODING OF CEPSTRAL FEATURES

In the server-only model, toll-quality speech can be coded and transmitted to the server by using standard speech coding techniques, like ADPCM at 32 Kbps, or newer schemes that are used today in mobile telephony, like GSM or CELP at bit rates of 13 Kbps or below. In Section 4, however, we show that in addition to the recognition performance degradation that one encounters when using toll quality instead of high-quality speech, we have an additional drop in performance when hybrid coding schemes like GSM or CELP are used at low bit rates.

In contrast, for the client-server approach, we only need to transmit the set of coefficients that will be used in recognition. Mel frequency-warped cepstral coefficients (MFCCs), are a common set of features used by many state-of-the-art HMM-based speech recognizers. Typical choices for the dimension of the feature vector and the rate at which it is computed are 13 and 100 times per second, respectively [6]. Secondary features, like the first- and second-order derivatives of this feature vector that are also used in recognition, do not have to be coded and transmitted, since this information can be obtained at the server side. Hence, one needs only to quantize a total of 1300 parameters per second of speech.

Discrete-density HMMs also quantize the front-end features and then model directly the quantized features, using discrete densities. A common choice is to use six features - namely, the energy, the vector of cepstral coefficients, as well as their first- and second-order derivatives - and quantize them by using separate vector-quantization (VQ) codebooks. In a typical discrete-density HMM [6], 256-dimensional codebooks are used for the cepstral coefficients and their derivatives, and 32-dimensional codebooks are used for the three energy features. If a discrete HMM approach is adopted for our client-server model, the required bit rate would be (3x8+3x5)x100 bps = 3.9 Kbps. Although this rate is significantly lower than the rate required to code the speech signal directly, it comes at a significant price in recognition accuracy: a one-and-a-half- to two-fold increase in

word-error rate has been reported for discrete-density HMMs when compared to their continuous-density counterparts.

The degradation in accuracy of the discrete-density HMMs can be attributed to the low resolution with which the space of observation features (the acoustic space) is represented. If we look at the subspace of cepstral coefficients, a typical discrete-density HMM uses a VQ codebook with 256 bins to represent a 12-dimensional space. Increasing the codebook size is not a feasible solution, since it complicates significantly both the client and server processes. The computation and memory requirements of the vector quantizer, which in our case will run at the client, will be proportional either to the number of bins, if a linear vector quantizer is used, or to their logarithm (i.e., the number of bits), when a tree-structured vector quantizer is employed. Most significant, however, is the cost at the server side. The number of parameters for a discrete-density HMM is proportional to the number of bins in the quantizer. For medium to large vocabulary applications, there are millions of parameters in discrete-density HMMs, and hence increasing the codebook size is not a feasible solution.

A standard technique for managing a large compression task is to decompose it into smaller sub-tasks [7]. To improve the resolution with which the acoustic space is represented, without the significant costs incurred when increasing the vector codebook size in discrete HMMs, we can employ scalar, or subspace quantization of the cepstral coefficients. Hence, we partition the cepstral vector into subvectors, and then encode the subvectors by using separate codebooks. In the extreme case, the subvectors consist of single cepstral coefficients. The total number of bins that represent the acoustic space is the product of the number of bins used for the representation of each subvector. To avoid the increase in the number of discrete-HMM parameters, we have chosen to model speech using continuous-density HMMs at the server. The subvectors are encoded at the client side, transmitted through the network, and then mapped to their centroids at the server. These centroids are then the input to the recognition process. To summarize, employing scalar, or subspace quantization of the cepstral coefficients has the following benefits:

- The acoustic space may be represented with a high-resolution, keeping the computational and memory requirements of the quantizer at the client side at a low cost.
- The centroids of the product-code can be used as input to a continuous-density HMM maintaining high recognition accuracy.
- There is no need to transmit secondary features, like the first- and second-order derivatives, maintaining the required bit rate at low levels.

4. EXPERIMENTS

To experiment with the quantization of cepstral parameters for speech recognition over the WWW, we have selected the ATIS domain. This is the domain of the first speech-enabled application over the WWW developed at SRI International [2]. In addition, there are available for the ATIS domain both high-quality and toll-quality data, which allows us to compare the server-only architecture, which uses toll-quality speech, with the client-server model which can use high-quality data.

4.1 Baseline and Server-only Performances

The recognizer used throughout our experiments is SRI's DECIPHER phonetically tied-mixture speech-recognition system [6]. The signal processing consists of a filterbank-based front end that generated six feature streams: the cepstrum, the cepstral energy, and their first- and second-order derivatives. Eight cepstral coefficients were used for telephone-quality speech, whereas for high-quality data we increased this number to twelve. A bigram language model was used throughout our experiments. The performance of the baseline recognizer high-quality speech was evaluated at 6.55% word-error rate (WER). Although not directly comparable, since it was evaluated on a different set of speakers than the high-quality baseline, the performance on telephone quality speech is significantly lower, measured at 12.7% WER. Compared to the telephone-quality baseline, the recognition performance did not degrade when the data was encoded using the G721 32-Kbps ADPCM coding standard. However, when speech was encoded with the full-rate RPE-LTP GSM 13-Kbps speech encoder used in cellular telephony, the WER increased to 14.5%. These results, summarized in Table 1, indicate the recognition performance of the server-only model for bit rates ranging between 13 and 64 Kbps.

4.2 Client-server Performance

We first quantized the cepstral coefficients of telephone-quality speech by using scalar quantization, and evaluated the recognition performance for various numbers of bits per coefficient. We investigated both uniform and nonuniform quantization. In the nonuniform quantization scheme, we used

Condition	Bit Rate (Kbps)	Word-error Rate (%)
M-law	64	12.7
GSM encoding	13	14.5

Table 1: Bit rates and word-error rates for different speech encoding schemes in the server-only processing model.

the empirical distribution function as an optimal companding function [7], since the random variable $Y=F_X(X)$ obeys a uniform distribution. The empirical distribution was estimated by using 800 utterances from different speakers. These results are summarized in Table 2.

We can see that the recognition performance is essentially flat for 4 to 8 bits per cepstral coefficient, and starts to degrade for lower numbers of quantization levels. Although we use a very simple quantization scheme, the WER of 13.2% at 3.6 Kbps is significantly better than the GSM performance, although the latter used a four-times higher bit rate. In addition, we see that the nonuniform quantization outperforms the uniform quantization significantly, especially at low numbers of bits per cepstral coefficient.

		Word-error Rate (%)	
Bits/Coef.	Bit Rate (Kbps)	Uniform	Nonuniform
8	7.2	12.55	12.82
7	6.3	12.65	12.87
6	5.4	13.08	12.65
5	4.5	13.14	13.62
4	3.6	17.43	13.19
3	2.7	45.47	14.64
2	1.8	108.9	21.07

Table 2: Bit rates and word-error rates for scalar quantization of cepstral coefficients in telephone-quality speech.

A significant advantage of running the front end at the client side, however, is that we can use the high-quality front end that uses a higher sampling rate and a larger number of bits per waveform sample. The baseline performance for the high-quality front end is 6.55% WER. Although the bit rates are slightly increased when compared to the telephone-quality front end, because of the larger number of cepstral coefficients used, we can see that the recognition performance is significantly better in comparable bit rates. For example, transmission of the high-quality cepstral coefficients at 3.9 Kbps yields a WER of 6.88%, whereas transmission of the toll-quality coefficients at 3.6 Kbps resulted in a 13.19% WER. When compared to the server-only processing model using GSM encoding, the performance improvement is even bigger: we get less than half the error rate (6.88% vs. 14.5%) at less than a third bits per second (3.9 Kbps vs. 13Kbps).

		Word-error Rate (%)	
Bits/Coef.	Bit Rate (Kbps)	Uniform	Nonuniform
8	10.4	6.65	6.53
7	9.1	6.76	6.40
6	7.8	6.65	6.43
5	6.5	6.96	6.32
4	5.2	6.96	6.32
3	3.9	12.45	6.88
2	2.6	95.43	9.04

Table 3: Bit rates and word-error rates for scalar quantization of cepstral coefficients in high-quality speech.

We have plotted, in Figure 2, speech recognition performance as a function of the bit rate for the three cases we examined: direct encoding of the speech signal, transmission of the cepstral coefficients of a telephone-quality front end, and transmission of the cepstral coefficients of a high-quality front end. We can see that, at any bit rate, the best strategy is to encode the high-quality cepstral coefficients. Further coding efficiency was obtained by using variable number of bits per coefficients. For example, by using 3 bits for cepstral coefficients 0-3 and 2 bits for coefficients 4-12 the error rate was 6.55% at 3 kbps.

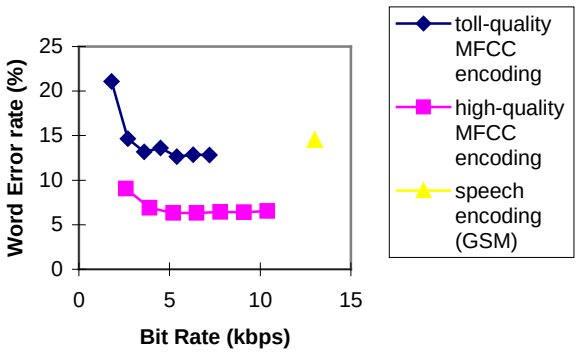

Figure 2: Recognition performance as a function of the bit rate for coefficient and waveform coding.

5. CONCLUSIONS

We investigated different strategies for encoding and transmitting speech in speech-enabled applications on the WWW. We found that nonuniform scalar quantization of MFCCs obtained using a high-quality front end had a much lower error rate than encoding the speech signal directly, at a fraction of the bit rate required for GSM encoding. The required bit rate to achieve the performance of a recognizer running locally with a high-quality front end was found to be less than 4 Kbps.

6. ACKNOWLEDGMENTS

This work was accomplished under a contract to Telia Research of Sweden and by internal SRI research and development funds.

7. REFERENCES

[1] D. Goddeau, W. Goldenthal and C. Weikart, "Deploying Speech Applications over the Web," *Proceedings Eurospeech*, pp. 685-688, Rhodes, Greece, September 1997.

[2] L. Julia, A. Cheyer, L. Neumeyer, J. Dowding and M. Charafeddine, "http://www.speech.sri.com/demos/atis.html," *Proceedings AAAI'97*, Stanford, CA, March 1997.

[3] E. Hurley, J. Polifroni and J. Glass, "Telephone Data Collection Using the World Wide Web," *Proceedings ICSLP*, pp. 1898-1901, Philadelphia, PA, October 1996.

[4] S. Bayer, "Embedding Speech in Web Interfaces," *Proceedings ICSLP*, pp. 1684-1687, Philadelphia, PA, October 1996.

[5] M. Sokolov, "Speaker Verification on the World Wide Web," *Proceedings Eurospeech*, pp. 847-850, Rhodes, Greece, September 1997.

[6] V. Digalakis and H. Murveit, "Genones: Optimizing the Degree of Mixture Tying in a Large Vocabulary Hidden Markov Model Based Speech Recognizer," *IEEE Trans. Speech Audio Processing*, pp. , July 1996.

[7] A. Gersho and R. M. Gray, "*Vector Quantization and Signal Compression,*" Kluwer Academic Publishers, 1991.