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 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. We find that the required bit rate to achieve the recognition performance of high-quality unquantized speech is just 2000 bits per second.

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]-[8]. Applications include Internet telephony, speech-enabled browsers, speech and natural language understanding systems, and speaker verification. Developers have successfully 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 of the client-server speech recognition 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.

Essentially, 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 tradeoff between transmission rates and perceptual reproduction quality. To achieve high perceptual quality at low transmission rates, 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 for recognition.

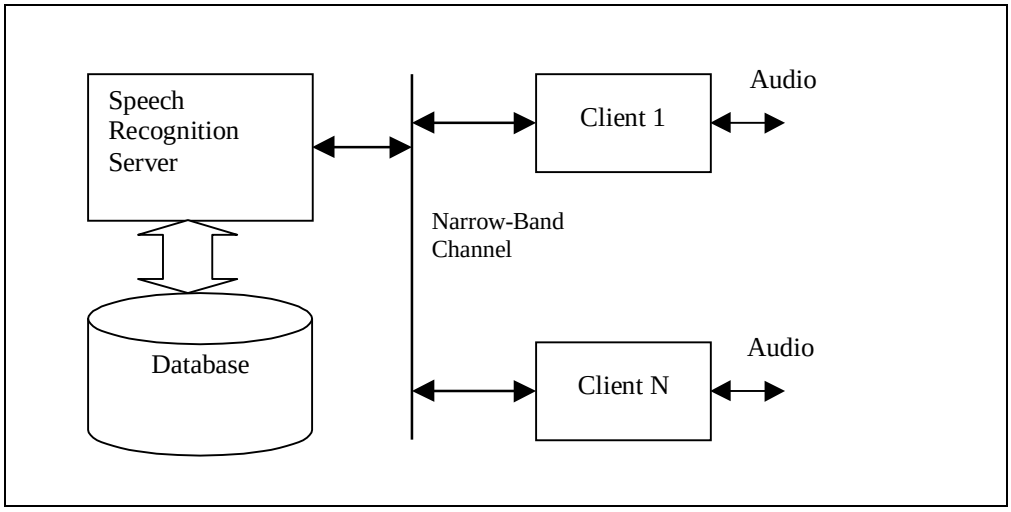

Figure 1: Client-server speech recognition system.

In Section 2 of this paper, we briefly review HMM-based speech recognizers. Section 3 examines alternative architectures for the implementation of speech-enabled applications over the WWW. In Section 4 we discuss techniques for encoding the front-end feature vectors at the client side and in Section 5 we present our experimental results. Section 6 presents our summary and conclusions.

2 HMM-BASED SPEECH RECOGNITION

Today's state-of-the-art speech recognizers are based on statistical techniques, with hidden Markov modeling being the dominant approach [9]. The typical components of a speech recognition and understanding system are the front-end processor, the decoder with its acoustic and language models, and the language understanding component. The latter component extracts the meaning of a decoded word sequence, and is an essential part of a natural language system. The remainder of this section briefly reviews the front end and the decoder.

The *front-end processor* typically performs a short-time Fourier analysis and extracts a sequence of observation vectors (or *acoustic* vectors) $X = \left[x_1, x_2, \cdots, x_T\right]$. Many choices exist for the

acoustic vectors, but the cepstral coefficients have exhibited the best performance to date [10]. The sequence of acoustic vectors can either be modeled directly, or vector-quantized first and then modeled.

The *decoder* is based on a communication theory view of the recognition problem, trying to extract the most likely sequence of words $W = [w_1, w_2, \cdots, w_N]$ given the set of acoustic vectors *X*. This can be done using Bayes' rule:

$$\hat{W} = \arg\max_W P(W|X) = \arg\max_W \frac{P(W)P(X|W)}{P(X)}.$$

The probability $P(W)$ of the word sequence *W* is obtained from the *language model*, whereas the *acoustic model* determines the probability $P(X|W)$.

In HMM-based recognizers, the probability of an observation sequence for a given word is obtained by building a finite-state model, possibly by concatenating models of the elementary speech sounds or phones. The state sequence $S = [s_1, s_2, \cdots, s_T]$ is modeled as a Markov chain, and is not observed. At each state s_t and time *t*, an acoustic vector is observed based on the distribution $b_{s_t} = p(x_t|s_t)$, which is called *output distribution.* Because HMMs assume, for simplicity, that observations are independent of their neighbors, first- and second-order derivatives of the cepstral coefficients are included in the acoustic vector x_t. If the front-end vector quantizes the acoustic vectors, the output distributions take the form of discrete probability distributions. If the acoustic vector generated by the front end is passed directly to the acoustic model, then continuous-density output distributions are used, with the multivariate-mixture Gaussians the most common choice:

$$b_s(x_t) = \sum_{i=1}^{K} p(\omega_i|s) N(x_t; \mu_{si}, \Sigma_{si}),$$

where $p(\omega_i|s)$ is the weight of the *i*-th mixture component in state *s*, and $N(x; \mu, \Sigma)$ is the multivariate Gaussian with mean μ and covariance Σ. In this work, we use continuous-density HMMs with mixture components that are shared across HMM states [11]. Continuous density

HMMs exhibit superior recognition performance over their discrete-density counterparts [9],[11].

3 SPEECH RECOGNITION OVER THE WWW

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 perform no processing related to the recognition/understanding process at the client side, but to simply transmit the user's voice to the server. The second alternative is to perform 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.

3.1 Server-Only Processing

When all the recognition processing takes place at the server side, we have the smallest computational and memory requirements on the client, allowing a wide range of client machines to 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 5, 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. The server-only approach was, however, followed by early applications [2],[3], and was attractive in the beginning because it overcame problems associated with audio capture and transmission standards.

3.2 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. However, this approach, 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, such as speaker verification [5]. An example of this approach is the Speech Aware Multimedia (SAM) system developed by Texas Instruments [6], in which the recognition grammar is downloaded from the server to client, and recognition is done locally, even though influenced by the server. Applications based on dynamic and complex grammars that require rapid database access are also not good candidates for a client-only architecture.

3.3 Client-Server Processing

The client-server approach is based on two key observations:

- 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 representation of the speech signal used for recognition concentrates on the part of the signal that is related to the vocal-tract shape.

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.

Of course, encoding and transmitting only the front-end processed tokens can become a disadvantage, since, without any representation of the speech associated with these tokens, the input speech cannot be labeled. As a result, it may not be possible to monitor in-service performance, or to collect labeled speech data for development and performance improvement. To overcome this limitation, and collect labeled data during the initial deployment of the application, it is possible to transmit the original speech encoded using a very-low-bit-rate coder as side information. This side information can be transmitted on top of the encoded front-end tokens during the development phase only.

The client-server model can be applied to the Internet, as well as to Wireless channels. The Aurora Project is a joint initiative, where a number of companies, including Alcatel, Ericsson, IBM, Matra, Nokia, Nortel and Siemens, are working to establish a global standard for distributed speech recognition over wireless channels [7]. The speech representations adopted by different speech recognition servers are usually very similar. However, the definition of the standard is not an easy task, since the speech-recognition developers put a significant amount of effort into fine-tuning the recognizer parameters to the specific representation, making it difficult to switch to a new, albeit similar representation. This task could be facilitated by the adoption of

an intermediate representation, such as the mel-filterbank coefficients, which exists in many front-ends. On the Internet, the client-server model has been adopted by the BBN SPIN (Speech on the Internet) system that was presented in [8]. Although the details of this system are not known, it was reported that it encodes speech at 3.8 kbps in a form suitable for discrete-density HMMs.

In our work, we follow the client-server model using the encoding scheme that is described in Section 4. We implemented a highly modular signal processing front-end in Java to compute the cepstral coefficients and encode the parameters. We verified that the system is fast enough to handle the feature extraction in real-time using a Pentium 166Mhz computer and a Java virtual machine (JVM) with a just-in-time (JIT) compiler. We also ran benchmarks to compare performance on the computation of the fast Fourier transform. We found that the optimized C code is twice as fast as the Java implementation. We believe that as the JVMs become more efficient the gap between C and Java performance will become even smaller.

The Java applet is downloaded from the server. By default, the Java security model prevents an applet from accessing native resources. There are various possible approaches to grant permission to access native resources. The various approaches for handling security policies in the Java model are beyond the scope of this paper.

4 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 5, 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 need only transmit the set of coefficients that will be used in recognition. Mel frequency-warped cepstral coefficients (MFCCs), constitute the set of features used by most state-of-the-art HMM-based speech recognizers today [10]. Because of their superior recognition performance, we have chosen to encode and transmit the set of cepstral coefficients, rather than working with representations that are typically used in the speech-coding

applications. 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 [11]. 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, and their first- and second-order derivatives–and quantize them by using separate vector-quantization (VQ) codebooks. In a typical discrete-density HMM [11], 256-dimensional codebooks are used for the 12-dimensional cepstral-coefficient vector and the corresponding vectors of cepstral derivatives, and 32-dimensional codebooks are used for the three scalar energy features. If a discrete HMM approach is adopted for our client-server model, the required bit rate would be (3x8+3x5)x100 bits per second (bps), or 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 compared with 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 codewords 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 codewords, 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 codewords 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 [12]. To improve the resolution with which the acoustic space is represented, without the significant costs incurred by 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. The total number of codewords that represent the acoustic space is the product of the number of codewords used for the representation of each subvector. The same technique is also used for coding several types of speech analysis parameters, including log-area-ratios (LARs) in traditional speech coding applications [13].

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.

The simpler approach is the extreme case of scalar quantization, where the subvectors consist of single cepstral coefficients. One can use either uniform or non-uniform quantization levels. In the latter case, the quantizer levels are matched to the statistics of the coefficient that is being quantized. In the experiments that we present in Section 5, we have used both approaches. In the nonuniform quantization scheme, we used the empirical distribution function as an optimal companding function [12], since the random variable $Y=F_X(X)$ obeys a uniform distribution. The empirical distribution can be estimated by using a large number of utterances from different speakers.

In the more general case, the dimensions of the subspaces used in the product code are larger than one. Although more complex variations of product codes exist [12], we are interested here

in partitioned VQ, where we simply partition the cepstral vector into two or more nonoverlapping subvectors. Product codes provide significant savings in memory storage of the codewords and reduce the computational cost for separable distortion measures [13]. Both these types of savings are very important in our application, because of the large number of codewords that must be used for good recognition performance. Since the coding of the cepstral vectors takes place at the client, heavy memory and computational requirements can significantly limit the types of machines that can access a speech-enabled WWW site.

An important issue in the design of a product code is the method used to partition the feature vector into a number of subvectors. A product code is optimal if the component vectors are independent and the distortion measure is separable [13]. Hence, one can partition the cepstral vector into subvectors by trying to satisfy the independence criterion. One approach is to partition the cepstral vector using the matrix of the estimated pairwise correlation coefficients of its elements. Each cepstral coefficient can be assigned to the subvector with the elements that are more correlated on average. An alternative, knowledge-based approach is to partition the vector of MFCCs into subvectors that contain consecutive coefficients, so that the most important low-order coefficients are grouped together.

Once the subvectors of the product code are formed, the next important design question is how to allocate the bits among the respective codebooks. Since we are interested in coding speech features for recognition, we have designed a bit-allocation algorithm that uses the word-error rate as a metric. Specifically, we start with an initial bit allocation to subvectors, and then increase the bit rate by adding bits to the subvectors that yield the maximal incremental increase in recognition performance as follows:

Initialization: Allocate the initial number of bits to subvectors and evaluate speech recognition performance. Set this as the current configuration.

Step 1: For each subvector, increase its allocated number of bits by one and evaluate speech recognition performance, keeping the number of bits assigned to each of the remaining subvectors as in the current configuration. Assign the additional bit to the subvector that resulted to the maximal increase in recognition performance, and set the new assignment as the current configuration.

Step 2: If the desired recognition performance has been achieved, or the maximum available bit rate has been reached, stop. Otherwise, go to step 1.

Any available metric can be used to evaluate speech recognition performance. In this work we have used the word-error rate (WER), which is the percentage of words that were 'erroneously' recognized (i.e., the recognizer has added, deleted or replaced some of the words that have been spoken in the initial sentence). Thus:

$$WER = \frac{INS + DEL + SUB}{TOTAL} \times 100\% .$$

Although the above procedure is computationally expensive, due to the multiple recognition experiments that must be run at each step, it is only executed once during the initial design of the quantizer. If, however, a faster allocation scheme is desired, the total assigned bits in the second step can be incremented in steps of multiple bits.

5 EXPERIMENTS

To experiment with the quantization of cepstral parameters for speech recognition over the WWW, we have selected the air-travel information (ATIS) domain [14]. In the ATIS domain, a user can get flight information and book flights across the United States using natural language. It consists of a vocabulary of approximately 1,500 words, with a moderate perplexity (a measure of difficulty). This is the domain of the first speech-enabled application over the WWW

developed at SRI International [2]. In addition, both high-quality and toll-quality data are available for the ATIS domain, 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.

5.1 Baseline and Speech-Encoding Performances

The recognizer used throughout our experiments is SRI's DECIPHER^{TM} speech-recognition system [11]. It uses continuous-mixture density HMMs, with Gaussians that are shared across acoustically similar states. 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. The coefficients were computed at a rate of 100 times per second. A bigram language model was used throughout our experiments. The performance of the baseline recognizer high-quality speech was evaluated at 6.55% WER using a test set of 34 male and female speakers with 400 utterances. 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 with 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.

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.

5.2 Scalar Quantization 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, the empirical distribution was estimated by using 800 utterances from a different set of speakers than those included in the test set. 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 bit rate that was four times higher. 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. In Table 3 we present the recognition results for scalar quantization of the cepstral coefficients of a high-quality front end. 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 at 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. 13 kbps).

		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.

Figure 2 plots 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.

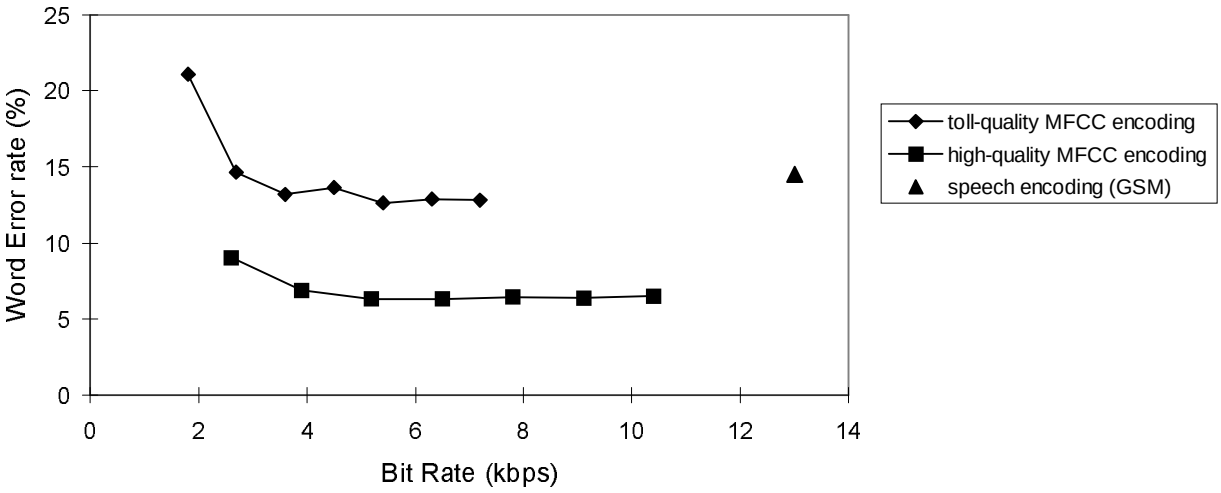

Figure 2: Recognition performance as a function of the bit rate for speech coding (GSM) and for MFCC-coefficient encoding using non-uniform scalar quantization. The coefficient encoding performance is shown for both a high-quality and a toll-quality front end.

5.3 Product-Code Quantization Performance

In the previous section we encoded the cepstral coefficients using scalar quantization with a constant number of bits per coefficient. In this section, we present our experiments using product code VQ with a variable number of bits per subvector. In all our experiments, the codebooks for each subvector were estimated by running the generalized Lloyd algorithm on the same 800 utterances that were used to estimate the empirical distribution in the nonuniform scalar quantization experiments. The codebooks were initialized using binary splitting [13].

We first compared the two alternative approaches for partitioning the cepstral coefficients into subvectors. In Table 4 we present, for the case of five subvectors, the WERs of the correlation- and knowledge-based approaches at various bit rates, as we measured them at various stages of the bit-allocation algorithm. The five subvectors consisted of the cepstral coefficients {(1,5), (3,9,12,13), (4,6), (2,7,11), (8,10)} and {(1,2), (3,4), (5,6,7), (8,9,10), (11,12,13)} for the

	Word-Error Rate (%)	
Bit Rate (bps)	Correlation-based partitioning	Knowledge-based partitioning
1400	18.77	11.71
1600	13.36	9.30
1800	10.24	8.10
1900	8.92	6.99
2000	8.38	6.63
2100	7.72	
2200	7.01	

Table 4: Bit rates and word-error rates for product-code VQ using 5 subvectors created by either a correlation-based or a knowledge-based approach.

correlation-based and the knowledge-based partition schemes, respectively. We see that the knowledge-based partitioning exhibits significantly better performance at all bit rates, and converges to the unquantized WER of 6.55% at a lower bit rate than the correlation-based scheme. We found experimentally that the problem with the correlation-based partitioning was the very low correlation between the various cepstral coefficients, which resulted in somewhat arbitrary partitions. This problem can be resolved by measuring phone-specific correlation coefficients, rather than pooling all the speech data together. Given the exceptional performance of the knowledge-based partitioning, which achieved the WER of the unquantized speech at just 2000 bps, we adopted the knowledge-based scheme for the rest of our experiments.

We then examined the behavior of the bit-allocation algorithm for various numbers of subvectors in the product-code VQ. In Table 5 we present the case of five subvectors. The initial bit rate was 1200 bps, and the algorithm was initiated by distributing twelve bits to the five subvectors, as shown in the first row of Table 5. To speed up the process, the number of allocated bits was increased by a step of two bits in the first iterations of the algorithm (until 1800 bps), and by a single bit in the latter stages of the algorithm. We can see that the initial WER of 16.79%

	Composition of subvectors by MFCC coefficients						
	1,2	3,4	5,6,7	8,9,10	11,12,13		
Total bits	Number of bits assigned to each subvector at each iteration					Bit Rate (bps)	Word-Error Rate (%)
12	3	3	2	2	2	1200	16.79
14	5	3	2	2	2	1400	11.71
16	5	3	4	2	2	1600	9.30
18	5	3	4	4	2	1800	8.10
19	5	4	4	4	2	1900	6.99
20	5	5	4	4	2	2000	6.63

Table 5: Progression of the bit-allocation algorithm for the case of five subvectors. The bits assigned to each subvector, the total bit rate, and the corresponding word-error rate are shown at intermediate steps of the algorithm.

decreases very rapidly and approaches the unquantized-speech performance at 2000 bps. The significance of the low-order coefficients is also obvious: The additional bits are allocated to the low-order subvectors first, and the final bit allocation uses more bits for the first two subvectors, although they are composed of only two coefficients each.

The same algorithm can be used to assign a variable number of bits to each coefficient in the nonuniform scalar quantization, since it is a special case of product-code VQ with single-element subvectors. The progression of the algorithm in this case is shown in Table 6. The initial bit rate was 1700 bps by assigning 17 bits to the 13 coefficients, as shown in the first row of Table 6. The algorithm was sped up by increasing the number of bits at each step by two, and by assigning them to the two coefficients that decreased the WER the most. We can see that in this case rates of at least 2600 to 2800 bps are required before the unquantized-speech performance is reached. The final bit allocation uses three bits for the first four cepstral coefficients, and two bits for the remaining coefficients.

	MFCC coefficient index														
	1	2	3	4	5	6	7	8	9	10	11	12	13		
Total bits	Number of bits assigned to each coefficient at each iteration													Bit Rate (bps)	Word-Error Rate (%)
17	2	2	2	2	1	1	1	1	1	1	1	1	1	1700	12.78
18	3	2	2	2	1	1	1	1	1	1	1	1	1	1800	10.66
20	3	3	2	3	1	1	1	1	1	1	1	1	1	2000	8.69
22	3	3	3	3	1	2	1	1	1	1	1	1	1	2200	7.67
24	3	3	3	3	2	2	1	1	1	1	1	2	1	2400	6.99
26	3	3	3	3	2	2	1	1	2	1	2	2	1	2600	6.81
28	3	3	3	3	2	2	1	2	2	2	2	2	1	2800	6.71
30	3	3	3	3	2	2	2	2	2	2	2	2	2	3000	6.55

Table 6: Progression of the bit-allocation algorithm for the case of scalar quantization (13 subvectors). The bits assigned to each coefficient, the total bit rate, and the corresponding word-error rate are shown at intermediate steps of the algorithm.

In Figure 3, we have plotted speech recognition performance as a function of the bit rate for different numbers of subvectors in the product-code VQ (three and five), and for the non-uniform scalar quantization with a variable number of bits per coefficient. In the same figure, we also show the WER for nonuniform scalar quantization using two bits per coefficient. The partitioning of cepstral coefficients into subvectors for the case of five subvectors was given above, whereas for the case of three subvectors, the partitioning was {(1,2,3), (4,5,6,7,8), (9,10,11,12,13)}. Scalar quantization with a variable number of bits demonstrates significantly better performance than the scalar quantization scheme with a fixed number of bits per coefficient that we examined in Section 5.2, reducing the WER to 6.81% from 9.04% at 2600 bps. Product code VQ, however, performs significantly better than scalar quantization at any bit rate. When comparing the three- and five-subvector cases, we see that they behave similarly for

low bit rates (below 1800 bps), but then the five-subvector scheme converges faster to the unquantized speech performance.

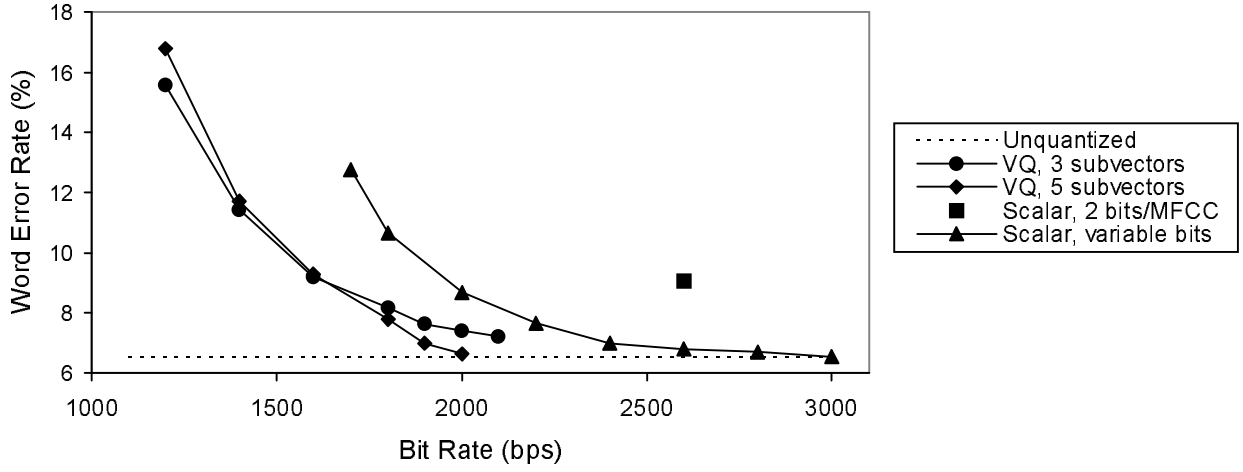

Figure 3: Recognition performance as a function of the bit rate for various types of MFCC encoding: nonuniform scalar quantization with constant and variable number of bits per coefficient; product-code vector quantization with different numbers of subvectors.

6 CONCLUSIONS

We investigated different strategies for encoding and transmitting speech in speech-enabled applications on the WWW. Using the server-only model with GSM encoding of speech, a performance of 14.5% WER was achieved at a bit-rate of 13 kbps. However, using the client-server model, for encoding MFCCs resulted in a much lower error rate–6.5% WER–since a high-quality front end can be used at the client side. This improvement in performance also comes at a fraction of the bit rate required for GSM encoding. A bit rate of 3900 bps is required when nonuniform scalar quantization with a constant number of bits per coefficient is used. This rate is reduced to 2800 bps with non-uniform scalar quantization with variable number of bits per coefficient, and to just 2000 bps when product-code vector quantization is used.

Other techniques, like predictive VQ, can be used to reduce the bit rate by taking advantage of the high correlation across time that cepstral vectors exhibit. We must, however, also consider other aspects of the problem, like the computational complexity of the encoder, which in our

case runs at the client side. The free nature of the Internet may limit the amount of encoding one can do. A wireless personal digital assistant (PDA) may be more likely to benefit from more encoding time and less transmission time, in which case the product-code VQ at 2 kbps may be required. Other types of clients may benefit more from the simplicity of the scalar quantization and transmit at 2800 bps.

Our work also has significant implications from the speech recognition perspective. The rate of 2 kbps, which is all that is required to achieve unquantized speech performance, is rather surprising. It is intriguing to see how low the bit rate can get, and to discover how much information is redundant in the cepstral features, since this may help us learn how to better model speech for recognition.

7 ACKNOWLEDGMENTS

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

8 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] C. Hemphill and Y. Muthusamy, "Developing Web-Based Speech Applications," *Proceedings Eurospeech*, Rhodes, Greece, September 1997.

[7] The Aurora Project, announced at Telecom 95, "http://gold.ity.int/TELECOM/wt95", Geneva, October 1995. See also "http://fipa.comtec.co.jp/fipa/yorktown/nyws029.htm".

[8] D. Stallard, "The BBN SPIN System", presented at the Voice on the Net Conference, Boston, Mass., September 1997.

[9] S. J. Young, "A Review of Large-Vocabulary Continuous-Speech Recognition," *IEEE Signal Processing Magazine*, pp. 45-57, September 1996.

[10] S. B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Trans. Acoustics Speech and Signal Processing*, Vol. ASSP-28(4), pp. 357-366, August 1980.

[11] 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. 281-289, July 1996.

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

[13] J. Makhoul, S. Roucos and H. Gish, "Vector Quantization in Speech Coding," *Proceedings of the IEEE*, Vol. 73, No. 11, pp. 1551-1588, November 1985.

[14] P. Price. "Evaluation of spoken language systems: The ATIS domain*," Proceedings of the Third DARPA Speech and Natural Language Workshop*, Hidden Valley, Pennsylvania, June 1990, Morgan Kaufmann.