DATA-DRIVEN VECTOR CLUSTERING FOR LOW-MEMORY FOOTPRINT ASR

Karim Filali[†], *Li Xiao*[*], *Jeff A. Bilmes*[*]
{karim,lixiao,bilmes}@ssli.ee.washington.edu
SSLI-Lab, University of Washington, Depts. of EE[*] and CS[†]

ABSTRACT

It is important to produce automatic speech recognition (ASR) systems that use as few computational and memory resources as possible, especially in low-memory/low-power environments such as for personal digital assistants. One way to achieve this is through parameter quantization. In this work, we compare a variety of novel subvector clustering procedures for ASR system parameter quantization. Specifically, we look at systematic data-driven subvector selection techniques based on entropy minimization, and compare performance on a 150-word isolated word speech recognition task. While the optimal entropy-minimizing quantization methods are intractable, we show that although several of our heuristic techniques are elaborate in their attempt to approximate the optimal clustering, a simple scalar quantization scheme using separate codebooks performs remarkably well.

1. INTRODUCTION

For certain applications, automatic speech recognition (ASR) will undoubtedly become the dominant human-computer interface methodology. For example, whenever hands are occupied (e.g., while driving), or where hand-based interfaces are bulky (using personal digital assistances (PDAs) or cell phones), ASR ultimately will succeed. Indeed, ASR is increasingly used on hand-held devices [8] — some PDA-based ASR systems are starting to appear commercially such as the IBM personal speech assistant [3] and the Microsoft MiPad [5] (others are listed in [8]).

Compared to their wired brethren, these portable computing devices invariably have limited computational and memory resources and strict power consumption constraints. Therefore, as more functionality is pushed into and better performance is demanded of portable ASR systems, it becomes crucial to investigate power saving techniques. Several approaches can achieve this goal such as voltage modulation, computation reduction, optimization for special applications (small vocabulary recognition), modified decoding algorithms, and low-memory consumption. The last approach addresses the problem of limited storage (ASR systems use a significant amount of memory to store parameters, typically means and variances of multivariate Gaussian distributions or vector-quantized codebooks), the higher power consumption associated with higher memory usage and processor memory traffic [11], and computation [10].

A simple yet effective way to reduce the required resources with little effect on performance is to use fewer bits per parameter. This is done by further quantizing numerical representations well below the typically 32 or 64 bits per parameter used with the IEEE floating point standard. In the past, several techniques have been used to achieve such quantization. Scalar quantization [12] jointly clusters the individual elements of parameter vectors (means and diagonal covariances) in order to achieve lower memory requirements. Moreover, subvector clustering and quantization has been applied to this problem [10]. In most cases, however, the choice of the subvectors uses knowledge only of the type of features used; for example clustering MFCC's as one subvector, the first derivatives as a second subvector, or grouping each MFCC with its 1st and 2nd derivatives. In [2], 2-dimensional subvectors are formed using a greedy algorithm that chooses pairs that are most strongly correlated.

This work was funded by NSF Award-0086032

In this paper, we evaluate and compare a variety of novel methods for sub-vector quantization of parameters of continuous density hidden Markov model (HMM) ASR systems. Specifically, we look at systematic data-driven vector clustering techniques based on entropy minimization (equivalently mutual information maximization), and compare their performance on a 150-word isolated word speech recognition task [9]. While the optimal entropy-minimizing quantization method is intractable, we show that although several of our heuristic techniques are elaborate in their attempt to approximate the optimal clustering, simple scalar quantization using separate codebooks per parameter performs surprisingly well.

In section 2, we describe our quantization algorithms and the subvector quantization techniques. Section 3 describes the speech corpus used and the experimental setup. In section 4 we show the memory-performance tradeoff results of our experiments. Section 5 discusses the results and concludes.

2. CLUSTERING ALGORITHMS

In the general problem of sub-vector quantization, we are given N vectors $v^{(i)}, i = 1, \ldots, N$ each of dimension D, which are to be quantized in some way. In this work, the N $v^{(i)}$s consist of the N means or N diagonal covariance matrices in a Gaussian-mixture HMM-based ASR system.[1] In sub-vector quantization, one decides upon M subsets $\{C_j\}_{j=1}^{M}$ of the index set $\mathcal{S} \triangleq \{1, 2, \ldots, D\}$, where $C_j \subseteq \mathcal{S}$ and where $C_j \cap C_m = \emptyset$ for all $j \neq m$ and $\bigcup_j C_j = \mathcal{S}$. For each of these sub-vectors, there are K code words. This means that the goal is to find the functions

$$f_{C_j}(v_{C_j}^{(i)}) = \bar{v}_{C_j}^{k} \quad 0 \leq j \leq M, \;\; 1 \leq k \leq K, \;\; \forall i$$

where $v_{C_j}^{(i)}$ is a partition of the vector $v^{(i)}$ corresponding to the elements within C_j, and where $\bar{v}_{C_j}^{k}$ is the k^{th} code word for that partition. Note that if $|C_j| = 1 \; \forall j$, then this corresponds to element-wise *scalar* quantization, and if $|C_j| = D$ (implying that $M = 1$), then this corresponds to full *vector* quantization. Anything in between, we will refer to as *subvector* quantization. In this general scheme, any vector element may be clustered with any set of other vector elements. The overall goal is to find the number of clusters M, the clusters themselves $\{C_j\}_{j=1}^{M}$ satisfying the above, the code-book size K (assumed to be the same for each cluster), and the quantization function $\{f_{C_j}(\cdot)\}_{j=1}^{M}$. The above quantities need to be found such that both the total memory and computation required is minimized, and also such that the word error rate (WER)

[1] In this work, we always quantize means and variances separately.

increase (relative to a baseline without quantization) is at a minimum. Because these two minimization criteria are independently optimizable, we report results as two-dimensional plots showing WER vs. total space required (equivalently number of bits per parameter). Plots which are both lower and to the left are preferable.

We further distinguish between two quantization styles, *disjoint* vs. *joint* quantization. Disjoint quantization is described above. With *joint* quantization, different clusters are quantized together using the same code book, meaning that we form the L sets $\{\mathcal{C}_\ell\}_{\ell=1}^{L}$ defining the set of sets $\mathcal{C}_\ell \subseteq \{C_1, C_2, \ldots, C_M\}$ such that $\mathcal{C}_\ell \cap \mathcal{C}_n = \emptyset$ and $\bigcup_\ell \mathcal{C}_\ell = \{C_1, C_2, \ldots, C_M\}$. In this case, the goal is to find the memory-size and WER minimizing functions

$$f_{\mathcal{C}_\ell}(v_{C_j}^{(i)}) = \bar{v}_{\mathcal{C}_\ell}^{k} \;\; \forall C_j \in \mathcal{C}_\ell, \;\; 1 \le k \le K, \;\; \forall i$$

such that $|C_i| = |C_j|, \forall C_i, C_j \in \mathcal{C}_\ell$ (i.e., clusters of different size cannot be quantized together), where $\bar{v}_{\mathcal{C}_\ell}^{k}$ is the k^{th} code word for cluster group ℓ.

From the above, we see that there are broadly two separate issues to solve. The first is how to select the number M and set of subvectors $\{C_j\}_{j=1}^{M}$, what we call the *clustering* problem. The second issue is how to perform the quantization once the clustering has been chosen. In this paper, we only address the first issue. This is because in a preliminary study, we investigated several quantization procedures including LBG [7], LVQ [6] and the K-means algorithm. We found that the choice of the quantization algorithm has little effect on the final WER. Therefore, all of our sub-vector clustering experiments use a simple hierarchical quantization algorithm similar to LBG meaning that code words are iteratively split and adjusted to minimize distortion (Euclidian squared distance).

2.1. Vector, scalar, and composed quantization

Vector and scalar quantization are defined in the previous section. We also define a method we refer to as *composed* quantization, where we first quantize the data vectors using vector quantization, then quantize the scalars of the quantized vectors using joint scalar quantization.

In all our experiments, we compute memory usage as follows: we denote the vector quantization resolution level by $q_{\texttt{vec}}$ and scalar quantization level by $q_{\texttt{sca}}$ then the storage needed for vector quantization is $2 \times (q_{\texttt{vec}} \times N + 2^{q_{\texttt{vec}}} \times D \times 32)$ bits, where the factor 2 is due to the quantization of both the means and variances. The first term in the sum corresponds to the storage required for the indices to the quantized data. The second term corresponds to the size of the code book. We assume 32 bits are used for unquantized scalars. For joint scalar quantization, the memory usage is $2 \times (q_{\texttt{sca}} \times N \times D + 2^{q_{\texttt{sca}}} \times 32)$ bits. For disjoint scalar quantization, the memory usage is $2 \times (q_{\texttt{sca}} \times N \times D + 2^{q_{\texttt{sca}}} \times D \times 32)$ bits. For composed quantization, $2 \times (q_{\texttt{vec}} \times N + 2^{q_{\texttt{vec}}} \times D \times q_{\texttt{scal}} + 2^{q_{\texttt{scal}}} \times 32)$ bits are required. Note that the table size term grows exponentially with the number of quantization bits.

2.2. Subvector quantization

Supposing that $v^{(i)}$ is a sample from a random variable V drawn from some distribution $p(v)$, the best quantization (in terms of number of bits per parameter) we can hope to achieve without any penalty is given by $H(V)/D = H(V_1, V_2, \ldots, V_D)/D$ where $H(\cdot)$ is the entropy function. Moreover, assuming sufficient samples $v^{(i)}$ (i.e., that N is large), it can be shown by the law of large numbers that vector quantization (i.e., $M = 1$) is optimal in that it will minimize the overall distortion between the original and the quantized data. There are two problems, however, with this scheme in practice. First, there is rarely enough data given the high dimensionality D of the parameter vectors. Second, the cost of storing the code book tables becomes prohibitive as the number of bits per quantized vector $q_{\texttt{vec}}$ increases. Subvector quantization, therefore, is an attempt to achieve better than scalar quantization while avoiding the problems mentioned above.

Fixing a particular clustering $\{C_j\}_{j=1}^{M}$, the fewest number of bits per parameter possible under the ideal sub-vector quantization scheme is given by $\frac{1}{M}\sum_{j=1}^{M} H(V_{C_j})$. We expect that below this amount WER would begin to increase dramatically. For example, with scalar quantization we would not hope to quantize without error at anything less than $\sum_j H(V_j)/D$ bits per parameter. Moreover, using entropic inequalities [4], it can be shown that:

$$H(V)/D \le \frac{1}{D}\sum_{j=1}^{M} H(V_{C_j}) \le \frac{1}{D}\sum_{j=1}^{D} H(V_j)$$

Therefore, an inherent tradeoff exists: we prefer large clusters up to the point where the limited amount of data available to perform the multi-dimensional sub-vector quantization and the size of the tables becomes an inhibiting factor.

An additional problem is that designing the best clustering $\{C_j\}_{j=1}^{M}$ is a hopelessly intractable problem. Even in the case where $|C_j| = 2$, finding the optimal clustering has exponential cost. One existing approach therefore is to manually divide the parameters into subsets based on prior knowledge of the vector elements[10]: for example, it might be argued intuitively that the joint entropies $H(\text{MFCCs})$, $H(\text{deltas})$, $H(\text{double deltas})$, $H(\text{log energy})$ will be small. In [2], a greedy algorithm is used to find $M = 13$ clusters that have low entropy.[2]

In the case where $C_j = 2 \;\; \forall j$, minimizing entropy is equivalent to maximizing pair-wise mutual information, as seen using [4] the formula $H(V_m, V_n) = H(V_m) + H(V_n) - I(V_m; V_n)$, where $I(V_m; V_n)$ is the mutual information between V_m and V_n. Moreover, standard linear correlation is an approximation to mutual information [4]. Therefore, the more jointly correlated the components of a subvector, the smaller the entropy will be, meaning the distortion between the quantized and unquantized subvector will be minimized.

We can view the D-dimensional parameters as a D-node fully connected weighted undirected graph, where the weight of each edge denotes the mutual information (or correlation) between the corresponding nodes. Clustering therefore can be seen as finding a graph M-partition, where nodes within each partition are as correlated as possible, and nodes between different partitions are relatively independent.

Based on the above, in this paper we explore various novel data-driven clustering techniques. The basic clustering algorithms are described below:

2.2.1. *Greedy-n Pair*

In this first algorithm, which we call *Greedy-n Pair* (where n is a parameter), we perform a tree search with branching factor n. The nodes of the tree are pairs of vector elements (so that $|C_j| = 2 \;\; \forall j$, and $M = D/2$) with the restriction that no two nodes on the path from the root of the tree to a leaf may contain the same element. The n children of a node are the top n ranked pairs in terms of mutual information between the two corresponding vector elements. Given the discussion in the previous section, the goal is to find the path from root to leaf that has the maximum sum of all the mutual information values of the pairs along the path. This algorithm is summarized as follows:

[2]Actually, they [2] find cluster pairs that are highly correlated, an approximation to low entropy as shown in the next paragraph.

1. Sort the nodes in decreasing weight
2. Recursively, find the node that maximizes the sum of its weight and the weight of the best path below it.
3. Assign each node in the path with the maximum weight to a 2-d subvector.

2.2.2. *Greedy*-1 *Triplet*

The above algorithm can be generalized to the case where the tree-nodes can have more than two elements ($|C_j| = 3$), a technique we call *Greedy*-1 *Triplet*. In the procedure we implemented, the measure of mutual dependency within elements of a cluster is the average pair-wise mutual information between all pair of scalar elements. Other than the different size of the clusters, the selection algorithm is the same as *Greedy*-1 *pair*.

2.2.3. *Maximum clique quantization*

The previous schemes require a uniform subvector size (i.e., $|C_i| = |C_j| \quad \forall i \neq j$) even though smaller or larger subvectors might exhibit a higher degree of correlation (and thereby better overall quantization). In our *maximum-clique* scheme, we adopt a structural approach in which the dependency graph described in Section 2.2 is pruned so that only a percentage of the edges with weights above some threshold remain. A maximum clique finding algorithm is then applied to the sparse graph. When there are two overlapping cliques, the one with the maximum average mutual information is chosen and its elements are removed from the graph.

2.2.4. *Joint quantization*

The discussion above assumes disjoint quantization, where each subvector is clustered using a separate code book. The alternate scheme is to quantize subvectors of the same size jointly, the motivation being that a better clustering might be achieved given that more data is available per subvector and that different subvectors could have overlapping value ranges. To ensure this is the case, we normalize all vector elements to have the same mean and variance, apply the joint quantization algorithm, and then convert the quantized vectors back to vectors with the original means and variances. This proved to work better than joint quantization without normalization

3. DATABASE

In all experiments reported in this work, we use NYNEX PHONEBOOK, a phonetically-rich, isolated-word, telephone-speech database[9]. Speech data is represented using 12 MFCCs plus c_0 and their deltas resulting in a $d = 26$ element feature vector every 10ms. The training and test sets are as defined in [1]. Test words do not occur in the training vocabulary, so test word models are constructed using phone models learned during training. Strictly left-to-right transition matrices were used except for an optional beginning and ending silence model. Four states per phone were used leading to a total of 165 hidden states, using the dictionary contained with the PHONEBOOK distribution.

In our results, we quantize only the means and variances of Gaussian distributions which are used to model the state output probabilities in a continuous density HMM. Mixture coefficients are left unquantized. Quantizing them neither achieves significant memory savings since they account for less than 5% of the number of means or variances nor does such an operation affect WER in a significant way. There are a total of 1900 mean and variance vectors, leading to $1900 \times 26 \times 2 = 98800$ 32-bit scalars for both the means and variances.

4. RESULTS

In this section, we evaluate the various clustering methods that were described in previous sections.

4.1. Vector, composed, scalar, and disjoint scalar

Figure 1 shows a comparison between vector, composed, scalar, and disjoint scalar clustering methods. The single cross at the right of the plot shows baseline performance, with no quantization (meaning 32-bits per parameter, and a memory cost that does not require a table). As can be seen, the scalar quantization schemes (both joint and disjoint) perform significantly better than either the vector or the composed schemes. Scalar quantization uses only 15.7% of the memory of the baseline, but has a 2.55% WER (only a 5.0% relative increase over the baseline). Composed quantization alleviates the table size problem which penalizes vector quantization, but still does not beat scalar quantization.

It is also worth noting that although disjoint scalar quantization might seem to require more memory than joint quantization, disjoint achieve a WER of about 2.65% with just 3 bits per parameter (9.8% of the baseline memory), and 2.46% with just 4 bits (13.3% of the baseline memory).

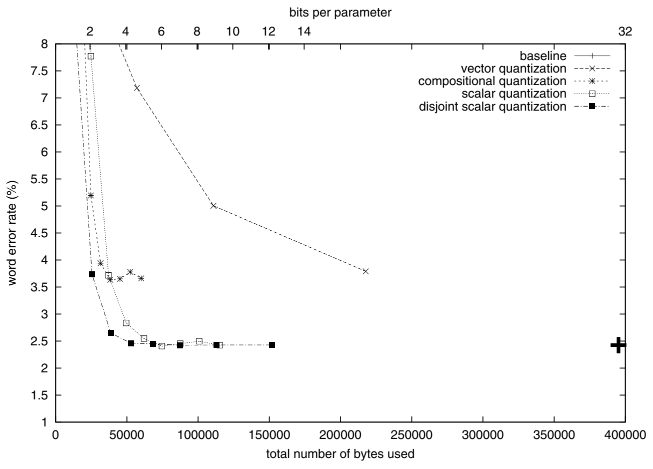

Fig. 1. Comparison between vector, composed, joint scalar and disjoint scalar quantization.

The remaining results we report are all presented together in Figure 2.

The top-left of the figure shows a comparison of Greedy-1 pair, disjoint scalar, and a random procedure (cluster pairs are chosen randomly). From the results, we see that the 2-dimensional subvector selected by Greedy-1 algorithm does do better than those selected randomly, although it does not outperform disjoint scalar quantization probably because Greedy-1 pair is only a heuristic. The plot does show that there appears to be an advantage in clustering correlated elements together, as is expected.

The top-middle of Figure 2 compares the Greedy-N pair procedure, with $N = 1$ and $N = 6$. For $N \in \{3, 4, 5, 6\}$ the clustering results were exactly the same, which is why we report only these two cases. As can be seen, there does not seem to be a clear advantage of Greedy-6 pair over Greedy-1 pair. In fact the resulting entropy sums were almost identical. While it might be that Greedy-N pair for $N > 6$ would achieve a better clustering, the computational cost quickly becomes prohibitive.

The top-right of the figure shows a comparison between Greedy-1 pair computed using simple linear correlation and an ap-

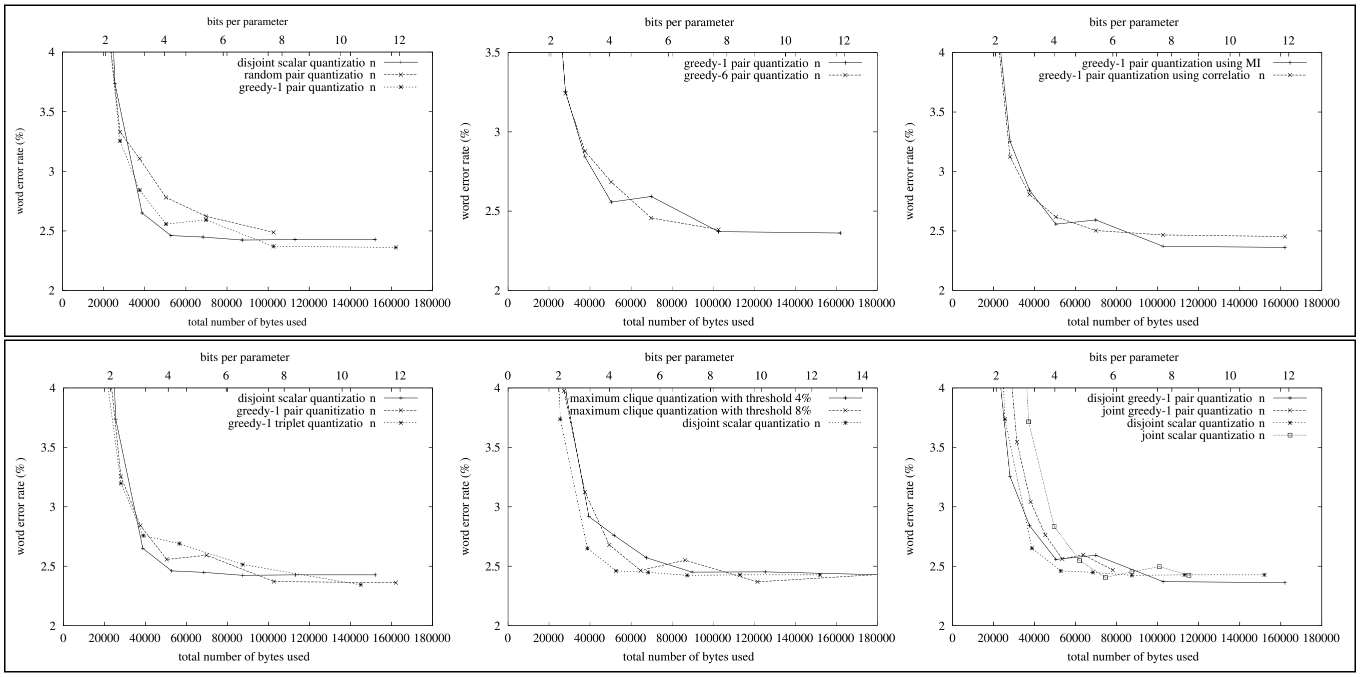

Fig. 2. Various comparison between clustering schemes.

proximation to true mutual-information. It is safe to say correlation is sufficient, probably because the amount of data ($N = 1900$) is not large enough to produce a reliable estimate of more accurate mutual-information approximations.

The bottom-left of Figure 2 compares disjoint scalar, greedy-1 pair, and greedy-1 triplet. The results show that using this heuristic, clusters of size three $|C_j| = 3$ do not perform better than simple disjoint scalar quantization, again presumably either because of data sparsity issues (small N) or clustering approximations.

The bottom-middle of the figure compares different thresholds for the maximum clique strategy. The thresholds we choose in this experiment is 4% and 8%, meaning we keep either 4% or 8% of the highest weight edges in the graph. Once again, the performance is no better than disjoint scalar quantization. In further experiments (not shown in the plot), we find that differences are negligible with a threshold ranging between 4% and 15%, and that quantization gets worse when the threshold is increased further.

Lastly, in the bottom-right of the figure, we compare joint vs. disjoint quantization for two different clustering methods (scalar as in Figure 1, and greedy-1 pair). As can be seen, both disjoint scalar quantization does better than joint scalar and disjoint greedy-1 pair quantization does better than joint greedy-1 pair. This leads us to the conclusion that the disjoint quantization methods (where each cluster has its own code book) are advantageous, even given potential data-sparsity issues.

5. DISCUSSION

The results above evaluate a number of novel methods for producing subvector-based parameter quantizations in Gaussian-mixture HMM-based ASR systems. We find that the disjoint scalar method is the best overall methodology, beating the much more elaborate heuristics that use, for example, mutual information, correlation and maximum clique discovery. In future work, we plan to investigate additional clustering and quantization schemes in order to improve results further.

6. REFERENCES

[1] J.A. Bilmes. Buried Markov models for speech recognition. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Phoenix, AZ, March 1999.

[2] Enrico Bocchieri and Brian Mak. Subspace distribution clustering for continuous observation density hidden markov models. In *Proc. Eurospeech '97*, pages 107–110, Rhodes, Greece, 1997.

[3] L. Comerford, D. Frank, P. Gopalakrishnan, R. Gopinath, and J. Sedivy. The IBM personal speech assistant. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, 2001.

[4] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[5] X. Huang, A. Acero, C. Chelba, L. Deng, D. Duchene, H. Hon, and et al. Mipad: A next generation pda prototype. In *Proc. Int. Conf. on Spoken Language Processing*, Beijing, China, November 2000.

[6] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. Lvq pak: The learning vector quantization program package, 1995.

[7] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer. *IEEE transactions on communications*, COM-28(1):84–95, 1980.

[8] G. Moyers. A handy way to interact. *Speech Technology*, pages 14–17, July/August 2001.

[9] J. Pitrelli, C. Fong, S.H. Wong, J.R. Spitz, and H.C. Lueng. PhoneBook: A phonetically-rich isolated-word telephone-speech database. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 1995.

[10] M. Ravishankar, R. Bisiani, and E. Thayer. Sub-vector clustering to improve memory and speed performance of acoustic likelihood computation. In *Proceedings of Eurospeech*, pages 151–154, Rhodes, Greece, 1997.

[11] Kaushik Roy and Mark C. Johnson. Software design for low power. In *Low Power Design in Deep Submicron Electronics. Proceedings of the NATO Advanced Study Institute*, pages 433–460, Dordrecht, Netherlands, 1997. Kluwer Academic Publishers.

[12] M. Vasilache. Speech recognition using hmms with quantized parameters. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 1999.