Algorithms for Data-Driven ASR Parameter Quantization*

Karim Filali\textsuperscript{a}, Xiao Li\textsuperscript{b}, Jeff Bilmes\textsuperscript{b}

\textsuperscript{a}Computer Science and Engineering Department, University of Washington, Seattle WA, 98195
\textsuperscript{b}Electrical Engineering Department, University of Washington, Seattle WA, 98195

Abstract

There is fast growing research on designing energy-efficient computational devices and applications running on them. As one of the most compelling applications for mobile devices, automatic speech recognition (ASR) requires new methods to allow it to use fewer computational and memory resources while still achieving a high level of accuracy. One way to achieve this is through parameter quantization. In this work, we compare a variety of novel sub-vector clustering procedures for ASR system parameter quantization. Specifically, we look at systematic data-driven sub-vector selection techniques, most of which are based on entropy minimization, and others on recognition accuracy maximization on a development set. We compare performance on two speech databases, PHONEBOOK, an isolated word speech recognition task, and TIMIT, a phonetically diverse connected-word speech corpus. While the optimal entropy-minimizing or accuracy-driven quantization methods are intractable, several simple schemes including scalar quantization with separate codebooks per parameter and joint scalar quantization with normalization perform well in their attempt to approximate the optimal clustering.

Key words: Speech recognition, lowpower, clustering, quantization

* This work was supported by National Science Foundation grant ITR/RC-0086032. Email addresses: karim@cs.washington.edu (Karim Filali), lixiao@ssli.ee.washington.edu (Xiao Li), bilmes@ssli.ee.washington.edu (Jeff Bilmes).
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 will undeniably succeed. Indeed, ASR is increasingly used on hand-held devices [14] — some PDA-based ASR systems are starting to appear commercially such as the IBM personal speech assistant [4] and the Microsoft MiPad [8] (others are listed in [14]). A number of wireless communication companies also launched their products integrated with ASR systems, like Motorala’s voiceXML and Nokia 9000 series.

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 such as voltage modulation, computation reduction, fixed-point arithmetic, optimization for special applications (small vocabulary recognition for example), alternative training and decoding algorithms, and low-memory consumption can achieve this goal. Varga et al. describe a combination of several of these approaches in the implementation of an ASR system for mobile phones [19]. Memory reduction is an important area of research because large vocabulary ASR systems use a significant amount of memory to store parameters, typically means and variances of multivariate Gaussian distributions. Associated with a big memory foot-print are higher processor-memory bus traffic and CPU load, all of which significantly increase power consumption [17].

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 typical 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 [20,18] jointly clusters the individual elements of parameter vectors (means and diagonal covariances) in order to achieve lower memory requirements. Sub-vector clustering quantizes subsets of the parameter vectors [7]. In most cases, however, the choice of the sub-vectors uses knowledge only of the type of features used; for example clustering Mel-frequency Cepstral Coefficients (MFCC) as one sub-vector, the first derivatives as a second sub-vector, or grouping each MFCC with its 1st and 2nd derivatives. In [2], 2-dimensional sub-vectors are formed using a greedy algorithm that chooses pairs that are most strongly correlated. In [3], the approach was expanded to higher dimensional sub-vectors using a multiple correlation coefficient. There is clearly a trade-off between the extent
of parameter quantization and how much recognition performance degrades because of the quantization noise, but there is also another trade-off involving computation time and memory that arises from the possibility to pre-compute quantities such as Mahalanobis distances and state log-likelihoods (see for example [16,20,3]). Although, in this work we are only interested in the memory savings of sub-vector quantization, the computation saving techniques in the above references are applicable here.

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, most of which are based on entropy minimization (equivalently mutual information maximization), and others on recognition accuracy maximization on a development set. We compare their performance on two speech corpora, PHONEBOOK and TIMIT. While optimal sub-vector selection 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 and joint scalar quantization with normalization perform surprisingly well.

In section 2, we describe our clustering algorithms and the sub-vector quantization techniques. Section 3 describes the speech corpus used and the experimental setup. In section 4 we show the memory-performance trade-off results of our experiments. Section 5 discusses the results and conclusions.

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. In sub-vector quantization, one decides upon $M$ subsets $\{C_j\}_{j=1}^M$ of the index set $S \triangleq \{1, 2, \ldots, D\}$, where $C_j \subseteq S$ and where $C_j \cap C_m = \emptyset$ for all $j \neq m$ and $\bigcup_j C_j = S$. For each of these sub-vectors, there are $K_j$ code words. This means that the goal is to find the functions

$$f_{C_j}(v^{(i)}_{C_j}) = \hat{v}^k_{C_j}, \quad 1 \leq j \leq M, \quad 1 \leq k \leq K_j, \quad \forall i$$

where $v^{(i)}_{C_j}$ is a partition of the vector $v^{(i)}$ corresponding to the elements within $C_j$, and where $\hat{v}^k_{C_j}$ 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

---

1 Means and variances are quantized separately.
\(|C_j| = D\) (implying that \(M = 1\)), then this corresponds to full vector quantization. We also define composed quantization to be the application of scalar quantization to vector-quantized parameters. Anything in between scalar and vector quantization, we will refer to as sub-vector 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 sizes \(K_j\), 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) increase (relative to a baseline without quantization) is at a minimum. Because these two minimization criteria are not independently optimizable, we report in section 4 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 of the same size are quantized together using the same codebook, meaning that we form the \(L\) sets \(\{C_\ell\}_{\ell=1}^L\) 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

\[
f_{\mathcal{C}_\ell}(v_{C_j}^{(i)}) = \bar{v}_{\mathcal{C}_\ell}^k \quad \forall C_j \in \mathcal{C}_\ell, \quad 1 \leq j \leq M, \quad 1 \leq k \leq K_\ell, \quad \forall i
\]

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 \(\ell\).

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 sub-vectors \(\{C_j\}_{j=1}^M\), what we call the clustering problem. The second issue is how to perform the quantization once the clustering has been chosen. This entails choosing a quantization algorithm, a distortion measure, and a bit allocation algorithm (refer to [13,6] for a discussion of these issues in speech coding.)

In all our experiments, we compute memory usage as follows: we denote the vector quantization resolution level by \(q_{\text{vec}}\) and scalar quantization level by \(q_{\text{scal}}\), then the storage needed for vector quantization is \(2 \times (q_{\text{vec}} \times N + 2^{q_{\text{vec}}} \times D \times 32)\) 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 codebook. We assume 32 bits are used for unquantized scalars. For joint scalar quantization, the memory usage is \(2 \times (q_{\text{scal}} \times N \times D + 2^{q_{\text{scal}}} \times 32)\) bits. For disjoint scalar quantization, the memory usage is \(2 \times (q_{\text{scal}} \times N \times D + 2^{q_{\text{scal}}} \times D \times 32)\) bits. For composed quantization, \(2 \times (q_{\text{vec}} \times N + 2^{q_{\text{vec}}} \times D \times q_{\text{scal}} + 2^{q_{\text{scal}}} \times 32)\) bits are required.
2.1 Sub-vector quantization

In loss-less compression we want to come up with a minimal set of codewords to represent exactly a much larger set of vectors. Compression works because “natural” and human-generated data has a structure far from random i.e. it has redundancy. Entropy, denoted as $H(\cdot)$, is a measure of this randomness and of how predictable a sample of the data is. The lower the entropy the less random the data is and the easier it is to compress. Even more importantly, Shannon proved that the best compression (the minimal number of bits per sample) that can be achieved is bounded below by the entropy. The difference between compression and quantization is that in the latter we allow some amount of distortion between the quantized codewords and the original vectors, which makes it possible to achieve quantization rates below the entropy.

Hereafter, we consider the vector $v^{(i)}$ a sample from a random vector $V$ drawn from some distribution $p(v)$. This is a valid model of the parameters we want to quantize as long as the probability distribution is estimated accurately. Assuming sufficient samples $v^{(i)}$ (i.e., that $N$ is large) and ignoring the codebook size, 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 using a fixed number of bits per element. This can also be shown by the entropy inequality [5],

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

The entropy rate of an arbitrary quantization scheme, $\frac{1}{D} \sum_{j=1}^{M} H(V_{C_j})$ is upper bounded by that of scalar quantization and lower bounded by that of vector quantization.

Vector quantization is optimal even when vector elements are independent. When dependencies do exist, vector quantization becomes even more advantageous since fewer bits are required to jointly encode correlated elements. In our experiments (section 4) we found that it is indeed the case that dependencies exist between different vector elements and that we should thus expect to see benefits in clustering those elements together.

This analysis gives us a potential scheme for optimally quantizing the parameters. We compute $H(V)$, the smallest number of bits per vector we can use

\[^2\] See [13] for a discussion of the relative merits of vector and scalar quantization applied to speech coding. Also see [9] for a comparison between vector and scalar quantization of LPC feature vectors.
without penalty if we were to compress the vectors, and run the best possible quantization algorithm to determine the codebook. However, this is in the ideal case where we assume sufficient data and we do not take codebook size into consideration. In practice, however, these two problems stand out to be crucial issues.

First, given the high dimensionality of the parameter vectors, there is rarely enough data to accurately compute $H(V)$. Second, the cost of storing the codebook tables becomes prohibitive as the number of bits per quantized vector $q_{vec}$ increases. Therefore, an inherent trade-off 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 become inhibiting factors. Sub-vector quantization, therefore, is an attempt to achieve better than scalar quantization while avoiding the problems mentioned above.

Theoretically speaking, fixing a particular clustering $\{C_j\}_{j=1}^M$, the number of bits sufficient to describe the variables without loss of information is given by $\frac{1}{D} \sum_{j=1}^M H(V_{C_j})$. We expect that below this amount the 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. If the variables in differing clusters are independent, the entropy rate $\frac{1}{D} \sum_{j=1}^M H(V_{C_j})$ then reaches its lower bound in equation 1. Therefore we can achieve the fewest number of bits per parameter under this sub-vector quantization scheme, which we call an optimal clustering.

Designing $\{C_j\}_{j=1}^M$ which approximates the optimal clustering, however, 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 [16]: for example, it might be argued intuitively that the joint entropies $H(MFCCs)$, $H(deltas)$, $H(double\ deltaside)$, $H(log\ energy)$ will be small. In [3], a greedy algorithm is used to find clusters that have low entropy based on a multiple correlation coefficient. In subsection 2.1.1, we describe an more exhaustive algorithm based on mutual information.

In the case where $C_j = 2 \ \forall j$, minimizing entropy is equivalent to maximizing pair-wise mutual information, as seen using 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 $V_n$ [5]. Moreover, standard linear correlation is an approximation to mutual information [5]. Therefore, the more jointly correlated the components of a sub-vector, the smaller the entropy will be, meaning the distortion between the quantized and unquantized sub-vector 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 mu-
tual 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 as independent as possible.

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

2.1.1 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 (i.e. each vector element belongs to a unique cluster). The $n$ children of a node are the top $n$ ranked pairs in terms of mutual information between the two corresponding vector elements. The larger $n$ is the more exhaustive the search is, but also the longer the running time.

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:

**Input:** MI values of all possible pairs of vector elements.  
**Output:** Assignment of vector elements to sub-vectors.

1. Form all possible nodes (pairs of vector elements) and sort them in decreasing order of weight (MI between the two elements of the node).

2. Construct the search tree by placing the top $n$ nodes under the root. For subsequent levels, each node is assigned as children the $n$ nodes that come after it in the ordered list from step 1.

3. Find the path from the root to the leaves that maximizes the sum of the nodes’ weights along the path. This can be done recursively in a top-down manner using depth-first search or in a bottom-up fashion as follows:
   Starting from the leaf level of the tree, consider consecutive subsets of size $n$ (which correspond to children of different patents) and mark the highest scoring nodes among each subset. Add the weight of those marked nodes to the weight of their parents. Then repeat the same procedure at this higher level. The final cluster choice will be the unique path that contains only marked nodes.
**Greedy-1 Pair**  Greedy-1 is a special case \( (n = 1) \) of the algorithm described above. Here we greedily select the node with maximum MI, assign it to a cluster, then select the next best node consistent with previous choices until \( D/2 \) nodes are selected. As mentioned above, a similar algorithm has been used in [2], but it uses correlation instead of mutual information to cluster pairs of vector elements.

**Greedy-1 Triplet**  The Greedy-\( n \) Pair algorithm can be generalized to the case where the tree-nodes can have more than two elements \( (|C_j| > 2) \). In this work, for computational reasons, we consider the less general case when \( n = 1 \) and \( |C_j| = 3 \) and we call the technique *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 pairs of scalar elements. Another more accurate way to evaluate dependency between elements of clusters larger than two is to compare the entropies of the clusters. We discuss this alternative approach in 2.1.2. Other than the different size of the clusters, the selection algorithm is the same as *Greedy-1 pair*.

This algorithm can be extended to the more general case, greedy-\( n \) \( m \)-let where \( n \) is the branching factor as before and \( m \) is the size of the clusters formed. The measure of dependency here is again the average pairwise mutual information.

### 2.1.2 Linear Entropy Minimization

The previous schemes require a uniform sub-vector size (i.e., \( |C_i| = |C_j| \quad \forall i \neq j \)) even though smaller or larger sub-vectors might exhibit a higher degree of correlation (and thereby better overall quantization). Another problem with the *Greedy-1 Triplet* algorithm is that the mutual information of the clusters is approximated by the sum of pair-wise mutual information values. The advantage of forming clusters of size larger than two is thus lost since we are not able to capture complex dependencies. In the following scheme, we allow clusters with different sizes and calculate the entropy of the cluster exactly by making a linear dependence assumption i.e we assume the probability distribution over vector elements is Gaussian. Then we can calculate the entropy of a sub-vector \( V_S \) of dimension \( |S| \) as \( H(V_S) = \frac{1}{2} \log((2\pi e)^{|S|} \times \text{det}(K)) \), where \( K \) is the covariance matrix of the Gaussian.

The algorithm proceeds as follows:

Input: Parameter vectors to be quantized and \( p \), the maximum cluster size.
Output: Assignment of vector elements to sub-vectors.
1. Calculate the entropies of all sub-vectors of size up to \( p \) (there are \( \binom{D}{1} + \binom{D}{2} + \cdots + \binom{D}{p} \) possible sub-vectors) and normalize each entropy by the size of the corresponding sub-vector to avoid penalizing large sub-vectors.

2. Rank entropies in increasing order.

3. Iteratively choose the cluster with the lowest entropy and remove all elements in the cluster from further consideration. The cluster assignment is over when there are no elements left.

We will refer to this sub-vector quantization scheme as *Entropy-min-\( p \)* where \( p \) is the maximum cluster size allowed.

Below we describe a different algorithm that allows varied sized clusters.

### 2.1.3 Maximum clique quantization

In our *maximum-clique* scheme, we adopt a structural approach in which the dependency graph described in Section 2.1 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. Essentially, it is attempting to minimize \( \frac{1}{D} \sum_{j=1}^{M} H(V_{C_j}) \) in equation 1, without the constraint that \( C_j \)'s must be of equal size. When there are two overlapping cliques, the one with the maximum average mutual information is chosen and its elements are removed from the graph.

The algorithm is described below:

Input: MI values of all pairs of vector elements.
Output: Assignment of vector elements to sub-vectors.

1. Starting from a complete graph (all nodes are connected to each other) where the weight of each edge corresponds to the pair-wise mutual information, eliminate all edges with weights below a chosen threshold.

2. Find the maximum sized clique of the graph. If there is more than one such clique, choose the one with the maximum average mutual information between its nodes.

---

\(^3\) A clique is often defined as a maximal complete set, but in some literature it refers to any complete subset. To avoid any ambiguity, we use the expression “maximum clique.”
3. Assign this clique to a sub-vector and eliminate it from the graph.

4. Repeat 2 and 3 until all nodes are assigned to a sub-vector.

The search space for the set of thresholds is of course very large but in practice we can choose a threshold by doing a significance test, i.e. we compute MI from random data and set the threshold around the highest value observed. Herein we have tried different threshold values that result in clique sizes that tend to perform better.

2.2 Joint quantization

The discussion above assumes disjoint quantization, where each sub-vector is clustered using a separate codebook. This makes the codebook size a great factor in memory consumption. An alternate scheme is to quantize sub-vectors of the same size jointly, the motivation being that the codebooks for different sub-vectors could have much overlap in their value ranges and that more data would be available to perform the quantization on. In the extreme case, where all equal-sized sub-vectors have the same probability distribution, the sub-vectors will have identical codebooks. In this case, joint quantization can achieve a distortion as low as that of disjoint quantization while the memory for codebooks is dramatically reduced.

Different sub-vectors, however, are not necessarily identically distributed. We, therefore, normalize all vector elements to have the same mean and covariance (under the Gaussian assumption, a probability distribution is fully determined by its mean and covariance), apply the joint quantization algorithm, and then convert the quantized vectors back to vectors with the original means and variances. When all the sub-vectors are scalars, this normalization procedure results in all elements $V_j, j = 1, \ldots, D$ in the vector $V$ being identically distributed. In general, however, component-wise normalization is not sufficient to ensure sub-vectors $V_{C_j}, j = 1, \ldots, M$ have the same distribution (for that, the off-diagonal elements of the covariance matrices need to be made the same). Nevertheless, the overlap between the ranges of the sub-vectors increases and in the two cases (scalar sub-vectors and pairs) in which we used normalization, this scheme proved to work better than quantization without normalization.

The normalization procedure is described below:

1. Calculate the sample mean and variance for each element $V_j$ in vector $V$. 

10
2. Normalize each element according to its corresponding mean and variance.

3. Quantize the normalized parameters.

4. At decoding time, the values stored in the codebook need to be rescaled according to the mean and variance associated with the vector element index.

When converting quantized normalized variances back to their original value ranges, these variances can sometimes take on negative values. We solve this problem by clamping such negative variances to zero (or a very small value), which works well since, for a variance to be assigned a negative codeword, it must have been close to zero in the first place.

2.3 Accuracy-based sub-vector selection

While entropy is a theoretically sound criterion for choosing sub-vectors that minimize quantization distortion, what we are really after is a partition of the parameter vectors such that recognition accuracy remains close to the baseline. The computational limitations of doing an optimal partition search are made worse by the need to perform a full recognition experiment for each candidate partition. We, therefore, use a smaller development set (the “core set” in the TIMIT distribution) which consists of about 15% of the utterances in the complete test set and limit the cluster sizes to two. We experiment with two algorithms: a greedy accuracy-based algorithm, which is the direct analogue of the Greedy-1 Pair algorithm, except that we replace mutual information with recognition accuracy as a criterion for ranking candidate vector component pairs. The greedy accuracy-based algorithm is outlined below:

Input: $D$-dimensional mean and variance parameter vectors.
Output: Assignment of vector elements to sub-vectors.

1. Form a list of all $\binom{D}{2}$ possible vector element pairs.

2. Form a partition such that a new pair is one subvector and the remaining elements are one-component subvectors.

3. Run a recognition experiment using the above partition and store the corresponding recognition accuracy.

4. Go to 2 and repeat until all pairs are considered.
5. Rank the list of pairs in descending order of recognition accuracy and select the top pairs such that they cover all vectors elements and no component is redundant. If \( D \) is odd, the last remaining component is assigned a sub-vector.

The second algorithm is a hierarchical scheme which starts with a partition where each component is assigned to a sub-vector and iteratively pairs components that result in the highest accuracy when they form a subvector. Again for computational reasons, we limit the hierarchy to three levels. The hierarchical accuracy-based algorithm proceeds as follows:

Input: \( D \)-dimensional mean and variance parameter vectors.
Output: Assignment of vector elements to sub-vectors.

1. Form a list of all \( \binom{D}{2} \) possible vector element pairs. Let \( P \) be the set of selected pairs so far. Initially \( P = \emptyset \).

2. Form a partition where pairs in \( P \) are assigned to two-component sub-vectors while the remaining elements are assigned to one-component sub-vectors.

3. Run a recognition experiment using the above partition, pick the pair that gives the highest accuracy and add it to \( P \).

4. Go to 2 and repeat until \( |P| = \lfloor D/2 \rfloor \) or some other criterion is satisfied (e.g. the accuracy drops below a certain threshold). The final partition consists of pairs in \( P \) and any remaining elements not covered by the element pairs.

3 EXPERIMENTAL SETUP

We use two speech corpora to test our clustering algorithms. NYNEX PHONE-BOOK is a phonetically-rich, isolated-word, telephone-speech database[15]. It contains 93,667 isolated-word utterances and totals 23 hours of speech. PHONE-BOOK is a good database to use for this study because its contains a variety of isolated words, which, we think, will be the most likely form of speech input using handhelds at least in the near term. The second corpus, TIMIT consists of read continuous speech. It contains 6300 sentences, totaling 2.5 hours of speech.

The feature extraction parameters and the training configuration were chosen such that the baseline results are close to the state of the art for each database.
in terms of performance and the number of parameters required. In the front-end, a 25ms window and a 10ms frame shift were used in addition to mean subtraction. For the back-end, strictly left-to-right HMM-based phone models were used except for an optional beginning and ending silence model. A very high pruning threshold was used for decoding.

**PHONEBOOK** speech data are represented using 12 MFCCs plus \( c_0 \) and their deltas (first-order time derivatives) resulting in \( d = 26 \) dimensional feature vectors. The training and test sets are as defined in [1]. We use a dictionary (part of the PHONEBOOK distribution) of 42 phones, four states per phone, 12 Gaussians per state, yielding a total of 1900 mean and variance vectors. The results presented in this paper were obtained on a 150-word test set, but we have also obtained similar results using 300 and 600-word test sets.

**TIMIT** feature vectors consist of 12 MFCCs, log energy, their deltas and double deltas (second-order time derivatives). The test set TIMIT distribution’s standard complete test set which consists of 1344 utterances. A core test set of 194 utterances was used as a development set for a couple of our experiments. 42 phones, three states per phone, 24 Gaussians per state were used to yield 3007 39-dimensional vectors for each of the mean and variance parameters.

We only quantize 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 a very small percentage of the number of means or variances nor does it affect WER in a significant way.

We use a hierarchical clustering scheme similar to LBG [11] \(^4\) to perform quantization. We have also experimented with LVQ [10] and k-means and while there are some differences in the WER versus memory curves for the different algorithms, we choose to focus on the problem of sub-vector selection. The distortion measure used is Euclidean distance. Other work has used measures such as the Batacharaya measure [12].

The baseline word error rate on the 150-word Phonebook test set is 2.42%, while the WER on the TIMIT full test set is 13.3%.

4 RESULTS

In this section, we evaluate the various clustering methods that were described in section 2. Except for the last subsection (4.1), which summarizes the results

\(^4\) The difference lies in that we split one cluster at a time instead of doubling the number of clusters at each epoch
on the TIMIT corpus, all the following experiments were conducted using the PHONEBOOK corpus.

**Vector, composed, joint scalar, and disjoint scalar quantization** Figure 1 shows a comparison between vector, composed, scalar, and disjoint scalar clustering methods. The horizontal line corresponds to the 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. Using only four bits per parameter (this includes the storage required for the codebooks and corresponds to about 15% of the original memory requirements), both the disjoint and disjoint schemes achieve baseline word-error rate. Composed quantization alleviates the table size problem which penalizes vector quantization, but still does not outperform scalar quantization.

![Graph](image.png)

Fig. 1. Comparison between vector, composed, joint scalar and disjoint scalar quantization. Composed quantization is the application of scalar quantization to vector-quantized parameters (in this particular case a 10-bit vector quantization was used).

**Pairwise dependencies** The extent to which sub-vector quantization improves upon scalar quantization depends on the strength of correlation between vector components. From figure 2 it is clear that there are indeed dependencies between vector element pairs, as evidenced by positive mutual information values that are larger than those derived from artificial data that was generated randomly according to a normal distribution with the same parameters as the real data. Moreover, MI values from the randomly generated data are very close to zero as expected, which increases our confidence—but does not guarantee—that real data MI values are also accurately estimated.
Fig. 2. Mutual information values of ordered pairs of vector elements for real and artificial data (for both means and variances). Artificial data is randomly generated according to a Gaussian distribution with the same parameters as real data. For clarity, only every tenth of the $\binom{26}{2}$ pairs are shown.

**Greedy-n schemes** We compare several greedy-$n$ schemes with disjoint scalar quantization and a procedure that selects pairs randomly (figure 3). All mutual information-based greedy algorithms outperform the random selection procedure, indicating that inter-component dependencies are indeed a good criterion for deciding which vector elements to cluster together as sub-vectors. In subsection 4.1, we look at some hand-crafted vector partitions that make intuitive sense and we see again that unless there are underlying correlations to back those partitions up, performance is poor.

Fig. 3. Comparison of greedy-$n$ schemes with disjoint scalar quantization and a procedure that randomly selects vector element pairs.

Comparing the greedy-$n$ algorithms, where $n$, the search branching factor, ranges from 1 to 8 (for $n \in \{3...8\}$ the clusters turned out to be exactly the same), we see that there is not a significant difference between the corresponding WER-versus-memory curves. Since a larger $n$ means a more thorough search and given that the quantity we are trying to optimize, total MI over all
selected pairs, does not increase by much as \( n \) increases, we hypothesize that there might be dominance effect whereby a few of the most correlated pairs account for the largest gains in terms of minimizing quantization distortion and word error rate. While it is conceivable that greedy-\( n \) pair for \( n > 8 \) would achieve a better clustering, the computational cost of the clustering algorithm quickly becomes prohibitive.

Greedy-1 triplet does the worse among the greedy schemes and although not shown in figure 3, performance keeps dropping as the dimensionality of the sub-vectors increases and we move to greedy-1 quadruplet and quintuplet. The same trend is found in the comparison between the entropy-minimization schemes, which we discuss below.

Even though the greedy schemes slightly outperform the disjoint scalar algorithm—and even do better than the baseline—at rate of 8 bits per parameter and higher, disjoint quantization achieves near baseline performance and does slightly better than the best of the greedy schemes at a rate of only 4 bits per parameter. Clearly, the large decrease in storage achieved by the disjoint scheme outweighs the small decrease in recognition accuracy. Why no greedy subvector clustering algorithm outperforms the scalar scheme given that significant component dependencies exist? There are several possible explanations. First of all, these algorithms are greedy heuristics and are not guaranteed to find the optimal clustering. However, even if we suppose the optimal clustering could be found, the poor performance of the greedy-1 triplet (and higher dimensional schemes) suggests there might be a more intrinsic reason which is that, the strength of the vector element correlation is not enough to offset the cost incurred by the sub-vector schemes in terms of table size: there are two components to the total memory storage required by a clustering scheme, index memory and codebook storage. Because smaller sub-vectors translate into more codewords that can be formed (for example, using two bits per scalar value the disjoint scheme yields \((2^2)^{26}\) possible codewords while the greedy-1 pair only \((2^2)^{13}\), fewer bits per sub-vector can be used and still have a good coverage of the original space of parameter vectors—and thereby a low quantization distortion. Fewer bits per sub-vector translate into smaller codebooks. On the flip side, because larger sub-vector quantization is more efficient than smaller sub-vector quantization, it can use fewer bits per component to store indices; however, since there a fewer large sub-vector more bits per sub-vector are needed to cover the parameter space and that means larger codebooks are needed. As the quantization rate gets higher, the cost of storing the indices starts to dominate and that is when we can see the larger sub-vector schemes start to perform better (at around 8 bits per parameter on our task as mentioned above).

Finally, another factor that enters into play is the estimation of the mutual information, which relies on having enough data to reliably model a complex
probability distribution (a mixture of Gaussians in our case). One way to rule out the data sparsity factor is to compare MI-based schemes to correlation-based schemes. Correlation requires less data for it to be estimated accurately.

**Mutual information-based versus correlation-based sub-vector selection** Figure 4 shows the results of greedy-1 pair computed using two different forms of mutual information approximation. The first way uses a mixture of Gaussians to estimate the joint density of the pair, and then uses that joint density to compute the mutual information. The second method assumes that the two random variables are jointly Gaussian, and computes the resulting MI analytically [5]. Note that the second way is equivalent to computing simple correlation. Correlation captures only first order (linear) dependencies and because MFCC features are processed using a Discrete Cosine Transform (DCT) to reduce the linear correlation between vector components, we would expect mutual information to better reflect the remaining correlations since it captures linear and non-linear dependencies, the latter having the potential of being stronger than the former because of the aforementioned DCT. What we find, nonetheless, is that the most correlated vector component pairs, albeit not in the exact same order, are the same whether MI or correlation is used. Linear dependencies still dominate as evidenced by the MI values, which, for the top pairs, are around .7 using linear MI and around 1.0 for the Gaussian mixture-based MI. I.e. the non-linear portion of the dependencies is about .3. Also, the fact that the order of the most dependent pairs is mostly preserved is an indicator that the non-linear dependencies “parallel” the linear ones. The similar performance of correlation-based and MI-based greedy-1 in figure 4 reflects the similar dependency patterns.

![Fig. 4. Comparison between pairwise quantization using MI and correlation.](image)

**Entropy minimization clustering** The linear entropy minimization scheme selects sub-vectors of variable size up to \( p \). In figure 5, results are shown for
As $p$ increases the WER versus memory curve is shifted up and to the left. As discussed above regarding the greedy heuristics, the likely reason for this is that, in general, with larger clusters, more bits per cluster need to be used to encode a given number of different codewords compared to smaller clusters. And even though the average number of bits per vector element is actually smaller for larger clusters, the table size—an exponential term—quickly swamps clusterings that use large clusters.

![Fig. 5. sub-vector quantization using linear entropy minimization. Entropy-min-4 and entropy-min-5 are not shown for clarity but they follow the same trend as entropy-min2,3,6](image)

**Maximum-clique clustering** We compare two maximum-clique clustering schemes with disjoint scalar quantization (figure 6). The thresholds we choose in this experiment are 4% and 8%, meaning we keep 4% and 8% respectively of the highest weight edges in the graph. Once again, the performance is no better than disjoint scalar quantization. In other 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. Higher thresholds yield denser graphs (more edges are retained) and consequently larger cliques, a fact which, per the discussion above, means a worse performance.

**Joint quantization** Lastly, in figure 7, we compare joint vs. disjoint quantization for two different clustering methods (scalar and greedy-1 pair).

The joint schemes, joint scalar and joint greedy-1 pair, do slightly better than disjoint scalar and disjoint greedy-1 pair respectively. For example the joint scalar scheme reaches the baseline WER (2.42%) with only 12.5% of the baseline memory, while the disjoint scheme achieves an WER of 2.49% with 13.3% of the baseline memory. However, that normalized joint schemes involve two
extra steps, normalization, which is done offline and the conversion to the original range of values, which has to be done online.\footnote{It is possible to perform the conversion offline at the cost of more memory usage, since we would need to store as many codebooks as there sub-vectors instead of just one codebook.}

4.1 TIMIT results

Because the experiments on PHONEBOOK database did not show the advantage of using sub-vector quantization one would expect given the dependencies that exist among the parameter vector components, we applied a subset of our quantization algorithms to the TIMIT corpus.\footnote{We did not perform joint quantization experiments on TIMIT; however, given the PHONEBOOK results and the results presented in this section, we do not expect much difference between disjoint and joint schemes.} TIMIT being a continuous
speech database with higher baseline error that PHONEBOOK, it is interesting to learn whether the same conclusions about the relative performance of the different quantization schemes hold.

**Entropy-based quantization** We first compare the various disjoint schemes used on PHONEBOOK above. As figure 8 shows, scalar quantization again turns out to be a very good compromise between performance and storage requirements. And as before, while algorithms such as greedy-1, linear-entropy-min-3, and the max-clique, which did relatively well on PHONEBOOK, performed well on TIMIT, there is no indication that any of the entropy-based schemes could be a better choice than plain scalar quantization.

![Figure 8. TIMIT entropy-based disjoint schemes comparison.](image)

**Dependency patterns** It is interesting to look at the most dependent vector component pairs. Table 1 shows mean vector component pairs chosen by the greedy-1 algorithm and their corresponding MI values. For mean vectors, except for a few cases, MFCCs and their double deltas exhibit the strongest correlations. For variance vectors, the trend seems to be that the strongest correlations occur between consecutive deltas or double deltas and between deltas and double deltas (table 2). We found the same variance vector correlation patterns on PHONEBOOK. We did not use double deltas on PHONEBOOK and obviously the mean vector correlations patterns were found to be different. In [3] and [12], very similar results to the mean correlations are reported on a different speech corpus, ATIS (Air Travel Information System). The authors compute the correlations on the feature vectors themselves instead of the parameter vectors but report that they found little difference in the derived partitions.

20
<table>
<thead>
<tr>
<th>Pair</th>
<th>MI</th>
</tr>
</thead>
<tbody>
<tr>
<td>c10,Δc10</td>
<td>.92</td>
</tr>
<tr>
<td>c12,Δc12</td>
<td>.90</td>
</tr>
<tr>
<td>c11,Δc11</td>
<td>.88</td>
</tr>
<tr>
<td>c8,Δc8</td>
<td>.82</td>
</tr>
<tr>
<td>c7,Δc7</td>
<td>.73</td>
</tr>
<tr>
<td>c9,Δc9</td>
<td>.72</td>
</tr>
<tr>
<td>c5,Δc5</td>
<td>.70</td>
</tr>
<tr>
<td>c6,Δc6</td>
<td>.68</td>
</tr>
<tr>
<td>c2,Δc2</td>
<td>.67</td>
</tr>
<tr>
<td>c1,E</td>
<td>.67</td>
</tr>
<tr>
<td>c4,Δc4</td>
<td>.66</td>
</tr>
<tr>
<td>Δc4,ΔE</td>
<td>.54</td>
</tr>
<tr>
<td>c3,Δc3</td>
<td>.45</td>
</tr>
<tr>
<td>Δc1,Δc7</td>
<td>.33</td>
</tr>
<tr>
<td>ΔΔc1,ΔΔE</td>
<td>.33</td>
</tr>
<tr>
<td>Δc2,Δc8</td>
<td>.33</td>
</tr>
<tr>
<td>Δc3,Δc5</td>
<td>.23</td>
</tr>
<tr>
<td>Δc6,Δc11</td>
<td>.10</td>
</tr>
<tr>
<td>Δc10,Δc12</td>
<td>0.094</td>
</tr>
<tr>
<td>Δc9</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 1: TIMIT’s most dependent mean vector components as measured using mutual information.

<table>
<thead>
<tr>
<th>Pair</th>
<th>MI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δc8,Δc9</td>
<td>1.1</td>
</tr>
<tr>
<td>ΔE,ΔΔE</td>
<td>1.1</td>
</tr>
<tr>
<td>Δc10,Δc11</td>
<td>1.1</td>
</tr>
<tr>
<td>Δc5,Δc6</td>
<td>1.1</td>
</tr>
<tr>
<td>ΔΔc8,ΔΔc9</td>
<td>1.1</td>
</tr>
<tr>
<td>ΔΔc5,ΔΔc6</td>
<td>1.1</td>
</tr>
<tr>
<td>ΔΔc10,ΔΔc11</td>
<td>1.1</td>
</tr>
<tr>
<td>ΔΔc2,ΔΔc3</td>
<td>1.1</td>
</tr>
<tr>
<td>Δc1,Δc1</td>
<td>1.0</td>
</tr>
<tr>
<td>Δc2,Δc4</td>
<td>.98</td>
</tr>
<tr>
<td>ΔΔc4,ΔΔc7</td>
<td>.86</td>
</tr>
<tr>
<td>Δc7,Δc12</td>
<td>.85</td>
</tr>
<tr>
<td>c11,c12</td>
<td>.60</td>
</tr>
<tr>
<td>c8,c9</td>
<td>.57</td>
</tr>
<tr>
<td>c5,c7</td>
<td>.45</td>
</tr>
<tr>
<td>Δc3,Δc12</td>
<td>.43</td>
</tr>
<tr>
<td>c6,c10</td>
<td>.40</td>
</tr>
<tr>
<td>c1,E</td>
<td>.34</td>
</tr>
<tr>
<td>c2,c4</td>
<td>.25</td>
</tr>
<tr>
<td>c3</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 2: TIMIT’s most dependent variance vector components as measured using mutual information.

**Hand-concocted partitions** The fact that there is a pattern to the correlations among parameter vector components suggest some hand-concocted partition schemes might do well. Figure 9 compares the following schemes:

1. Clustering consecutive pairs together.
2. Clustering MFCCs, deltas, and double deltas as three separate sub-vectors.
3. Clustering the triples (MFCC,DELTA,DDELTA).
4. Clustering deltas as one sub-vector, double deltas as a second one, and keeping MFCCs disjoint.
The consecutive pair clustering curve closely follows the curve of the greedy-1 scheme (which, in the case of variances, predominantly clusters consecutive pairs). The other partition schemes, on the other hand, perform poorly.

![Accuracy-driven quantization](image)

**Fig. 9. TIMIT hand-crafted sub-vector partition disjoint schemes comparison.**

**Accuracy-driven quantization** We have tried two recognition accuracy based algorithms for finding optimal vector partitions, *max-accuracy-greedy-pairwise* and *max-accuracy-hierarchical-pairwise*, described in section 2. The first is the equivalent of the greedy-1 scheme but uses accuracy instead of entropy to rank component pairs; the second, hierarchically builds a partition by successively pairing vector components which result in the least decrease in accuracy. Because of computational constraints, however, we have limited the second scheme to three levels of clustering, meaning that at most 3 pairs are formed and scalar quantization is used for the remaining components. We allocate 5 bits per sub-vector during the partition search as, from the previous experiments, it achieves a performance close to baseline but still leaves some room for improvement.

Max-accuracy-greedy performs consistently worse than greedy-1. There are two possible explanations for that: the performance development core test set might not be a good indicator of the performance on the complete test set, even though we have tried to rule out this possibility by comparing a few quantization curves using both test sets. The more likely explanation is that the effect of pairing components based on the maximum accuracy criterion is not cumulative. This hypothesis is reinforced by the fact that max-accuracy-hierarchical achieves a lower error rate at most quantization levels (Figure 10).
5 CONCLUSION

We have evaluated a number of novel methods for producing sub-vector-based parameter quantization in Gaussian-mixture HMM-based ASR systems. We find that three methodologies are the overall best in their attempt to reduce memory: disjoint scalar, joint scalar and joint greedy-1-pair quantization. They do better than more elaborate heuristics. With less than a seventh of the parameters storage, a near-baseline WER is achieved.

Even though the disjoint scalar scheme requires more storage for its separate codebooks, its memory/WER curve is close to the best one achieved. Moreover it does not require the normalization/re-conversion step in the normalized schemes. It is surprising that a disjoint scheme would do well at all because of the table storage, but it becomes less surprising when we look at the added flexibility to form codewords when separate codebooks are used. Joint schemes are constrained to share the same codebook among sub-vectors and therefore for a fixed number of bits per codeword, they encode fewer combinations of codewords than a disjoint scheme does.

An issue that we have not investigated in this work is the interaction between the clustering schemes and the quantization algorithms. It is possible that different combined clustering-quantization schemes would perform better, although we do not expect the improvements to be very important given the difficulty to improve upon the simple schemes presented in this paper.

Acknowledgments

We would like to thank NSF for funding this project under Award-0086032.
References


