Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Speech and Language 20 (2006) 625–643
COMPUTER SPEECH AND LANGUAGE
www.elsevier.com/locate/csl

Algorithms for data-driven ASR parameter quantization

Karim Filali [a,*], Xiao Li [b], Jeff Bilmes [b]

[a] Computer Science and Engineering Department, University of Washington, P.O. Box 352350, Seattle, WA 98195, United States
[b] Electrical Engineering Department, University of Washington, Seattle, WA 98195, United States

Received 10 November 2003; received in revised form 19 October 2005; accepted 26 October 2005
Available online 29 November 2005

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.

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 (Moyers, 2001) – some PDA-based ASR systems are starting to appear commercially such as the IBM personal speech assistant (Comerford et al., 2001) and the Microsoft MiPad (Huang et al., 2000) (others are listed in Moyers (2001)). A number of wireless communication companies also launched their products integrated with ASR systems, like Motorola's voice-XML 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

* Corresponding author. Tel.: +1 206 715 3662; fax: +1 206 543 2969.
E-mail addresses: karim@cs.washington.edu (K. Filali), lixiao@ssli.ee.washington.edu (X. Li), bilmes@ssli.ee.washington.edu (J. Bilmes).

0885-2308/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.csl.2005.10.003

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. (2002) describe a combination of several of these approaches in the implementation of an ASR system for mobile phones. 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 (Roy and Johnson, 1997).

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 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 (Vasilache, 2000; Takahashi and Sagayama, 1995) simply clusters the individual elements of parameter vectors (means and diagonal covariances). Sub-vector clustering, also referred to as product VQ,[1] breaks up vectors into sub-vectors, allowing reduced complexity search and reduced storage requirements at the cost of an increase in distortion (Gray and Gersho, 1991). 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 Bocchieri and Mak (1997), 2-dimensional sub-vectors are formed using a greedy algorithm that chooses pairs that are most strongly correlated. In Bocchieri and Mak (2001), 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, but there is also another trade-off involving computation time and memory that arises from the possibility to pre-compute quantities such as Mahanalabolis distances and state log-likelihoods (see for example Ravishankar et al., 1997; Vasilache, 2000; Bocchieri and Mak, 2001). 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.

It is important to emphasize that *parameter quantization* is different from *feature quantization*. In the latter, it is the input vectors (derived from the speech waveforms) to the speech recognizer that are quantized and not the learned distribution parameters. A common application of feature quantization is client–server speech recognition, where the speech signal is processed on the client side (typically a low-resource device) to extract speech feature vectors (Mel Frequency Cepstral Coefficients for example), which are then quantized to minimize the communication cost of sending them to a server where the bulk of the speech processing is performed. We focus on parameter quantization and refer the reader to Malkin et al. (2004) and Li et al. (2004) for a study of how both types of quantization can be combined. The short summary is that feature quantization and parameter quantization are quite complementary and can be optimized independently.

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. We find that simple scalar quantization using separate codebooks per parameter and joint scalar quantization with normalization perform surprisingly well.

In Section 2, we describe our algorithms for picking sub-vectors 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.

[1] We prefer the term sub-vector quantization as it more precisely refers to the idea of partitioning vectors for quantization, while the term "product VQ" can be more general and refers to schemes that break up vectors in ways other than through partitioning (a shape vector and a gain factor for example, or a vector and its residual).

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.[2] In sub-vector quantization, one decides upon M subsets $\{C_j\}_{j=1}^{M}$ of the index set $\mathscr{S} \triangleq \{1, 2, \ldots, D\}$, where $C_j \subseteq \mathscr{S}$ and where $C_j \cap C_m = \varnothing$ for all $j \neq m$ and $\bigcup_j C_j = \mathscr{S}$. For each of these sub-vectors, there is an associated table (which we also call *codebook*) B_j with K_j *codewords* (table entries). This means that the goal is to find the functions

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

where $v_{C_j}^{(i)}$ is a partition of the vector $v^{(i)}$ corresponding to the elements within C_j, $\bar{v}_{C_j}^{k}$ is the kth code word for that partition, and $k \in \{1, \ldots, K_j\}$. 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 standard full *vector* quantization. We also define *composed* quantization to be the application of scalar quantization to vector-quantized parameters to reduce storage costs. 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 cannot be optimized independently, we report in Section 4 results as 2-dimensional plots showing WER versus 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* versus *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 $\{\mathscr{C}_\ell\}_{\ell=1}^{L}$ defining the set of sets $\mathscr{C}_\ell \subseteq \{C_1, C_2, \ldots, C_M\}$ such that $\mathscr{C}_\ell \cap \mathscr{C}_n = \varnothing$ and $\bigcup_\ell \mathscr{C}_\ell = \{C_1, C_2, \ldots, C_M\}$. In this case, the goal is to find the memory-size and WER minimizing functions

$$f_{\mathscr{C}_\ell}(v_{C_j}^{(i)}) = \bar{v}_{\mathscr{C}_\ell}^{k} \ \forall C_j \in \mathscr{C}_\ell, \ 1 \leqslant j \leqslant M, \ \forall i,$$

such that $|C_i| = |C_j|, \ \forall C_i, C_j \in \mathscr{C}_\ell$ (i.e., clusters of different size cannot be quantized together), where $\bar{v}_{\mathscr{C}_\ell}^{k}$ is the kth code word for cluster group ℓ, and $k \in \{1, \ldots, K_\ell\}$.

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 Makhoul et al. (1985) and Digalakis et al. (1998) for a discussion of these issues in speech coding.)

As a concrete example of how we compute the total memory required to store Gaussian means (or variances) under a given quantization scheme, let q be the number of bits we decide to allocate for indexing each codebook (we assume equal allocation of bits to each codebook). The size, K_j, of each codebook B_j, $j \in 1, \ldots, M$ is thus equal to 2^q. Under the disjoint quantization scheme, each codeword in table B_j requires $|C_j| \cdot 32$ bits of storage, assuming 32 bits per unquantized scalar. Summing over all codebooks the total codebook storage is $2^q \cdot 32 \cdot D$ bits ($D = \sum_{j=1}^{M} |C_j|$ because $\{C_j\}_{j=1}^{M}$ form a partition of the index set $\{1,2, \ldots, D\}$). The storage required for the indices themselves is $q \cdot M \cdot N$ bits.

Under joint quantization, if we assume here for simplicity equal-sized sub-vectors, i.e., $|C_j| = c, j = 1 \ldots M$ for some constant c ($1 \leqslant c \leqslant D$), then the codebook storage is $2^q \cdot 32 \cdot c$ bits. The index storage is the same as above.

[2] Means and variances are quantized separately.

2.1. Sub-vector quantization

In lossless compression we want to come up with a minimal set of codewords to represent exactly a much larger set of vectors. Lossless compression works because "natural" and human-generated data have a structure far from random, i.e. they have 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 lossless 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 $v^{(i)}$ to be 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.[3] This can also be shown by the entropy inequality (Cover and Thomas, 1991),

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

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 because of the space filling advantage (Lookabaugh and Gray, 1989). 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 would compute $H(V)$, the smallest number of bits per vector we can use without penalty if we were to compress the vectors, and run the best possible quantization algorithm to determine the codebook. However, this is the ideal case where we assume sufficient data and do not take codebook size into consideration. In practice, these two problems stand out to be crucial.

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 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 is an attempt to perform better than scalar quantization while avoiding the problems mentioned above.

Designing the sub-vector partitions $\{C_j\}_{j=1}^{M}$, however, is a hopelessly intractable problem. Even in the case where $|C_j| = 2$, finding the optimal clustering is NP-complete.[4] One existing approach therefore is to manually divide the parameters into subsets based on prior knowledge of the vector elements (Ravishankar et al., 1997): for example, it might be argued intuitively that the joint entropies H(MFCCs), H(deltas), H(double deltas), H(log energy) will be small. In Bocchieri and Mak (2001), a greedy algorithm is used to find clusters that have low entropy based on a multiple correlation coefficient. In Section 2.1.1, we describe a more exhaustive algorithm based on mutual information.

[3] See (Makhoul et al., 1985) for a discussion of the relative merits of vector and scalar quantization applied to speech coding. Also see (Juang et al., 1982) for a comparison between vector and scalar quantization of LPC feature vectors.

[4] We can reduce to this problem from the Traveling Salesman Problem: Given a graph $G = (V,E)$ and a distance $d(u,v)$ associated with each edge (u,v), we create a $V \times V$ matrix. For each pair of vertices (u,v) with no edge between u and v put – inf in the corresponding matrix entry; otherwise put $-d(u,v)$. The $|C_j| = 2$ case clustering problem is to find matrix entries such that their sum is maximal and satisfy the constraint that no two selected entries are in the same column or same row (a component cannot be in two clusters at the same time). The solution is also a solution to TSP because each vertex is visited only once and the sum of distances traveled is minimized.

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 (Cover and Thomas, 1991). Moreover, standard linear correlation is monotonically related to mutual information (Cover and Thomas, 1991). 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 less.

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 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. Note that there are several other clustering approaches that are applicable in our setting; for example, a recent algorithm in Narasimhan et al. (2005) guarantees a factor 2 approximation to the optimal solution to our clustering problem for any fixed M.[5] Spectral clustering techniques (e.g. Ng et al., 2001), and graph cut techniques (e.g. Chung, 1997) have also been popular clustering techniques.

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 (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, and 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 (empty) 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 leafs that maximizes the sum of the nodes' weights along the path. This can be done recursively in a top-down manner using branch-and-bound search.

The run-time of the *Greedy-n Pair* algorithm is dominated by the $O(n^D)$ search in step 3. The input to the algorithm requires $O(D^2)$ computations of mutual information.

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 Bocchieri and Mak (1997), but it uses linear 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

[5] In the case $M = 2$, a polynomial-time optimal algorithm exists.

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 can be either average pairwise mutual information, or the joint entropy over *m* variables.

2.1.2. Linear entropy minimization

The previous schemes require a uniform sub-vector size (i.e., $|C_i| = |C_j|\ \forall i,j$) even though smaller or larger sub-vectors might exhibit a higher degree of correlation (and thereby better overall quantization). In the following scheme, we allow clusters with different sizes and calculate the entropy of the cluster. We make a linear dependence assumption by assuming the probability distribution over vector elements is Gaussian. This assumption simplifies the calculation of the entropy as a closed formula exits: $H(V_S) = \frac{1}{2}\log((2\pi e)^{|S|} \times \det(K))$, where V_S is a sub-vector of dimension $|S|$ and K is the covariance matrix of the Gaussian (Cover and Thomas, 1991).

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 $M(p) = \binom{D}{1} + \binom{D}{2} + \cdots + \binom{D}{p}$ 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.

The complexity of *Entropy-min-p* is $M(p)log(M(p))$ (where $M(p)$ is defined in step 1) because of the sorting in step 2.

2.1.3. Maximum clique quantization

In our *max-clique*[6] scheme, we adopt a structural approach in which the dependency graph described in Section 2.1 is pruned so that only those edges with weights above some threshold remain. A maximum clique finding algorithm is then applied to the sparse graph.[7] Essentially, it is attempting to minimize $\frac{1}{D}\sum_{j=1}^{M} H(V_{C_j})$ in Eq. 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 pairwise 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.

[6] A clique is often defined as a maximal complete subset (all edges are present in the subset), but in some literature it refers to any complete subset. To avoid any ambiguity, we use the expression "maximum clique".

[7] The max-clique problem is NP-complete; however by limiting the maximum clique size (and thus the depth of the search), and operating on sparse enough graphs, our problem instances are computable in a reasonable amount of time.

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. 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 an important 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. 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 (if they are indeed Gaussian). In general, however, component-wise normalization is not sufficient to ensure sub-vectors $V_{C_j}, j = 1, \ldots, M$ have the same Gaussian 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.
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 setting such negative variances to a very small positive 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 as close as possible to the baseline. The goal of our accuracy-based sub-vector selection scheme is to incrementally build a partition that results in the smallest decrease in accuracy. This selection scheme is similar to discriminative training approaches, where the goal is to learn a model (partitions in our case) that directly optimizes a given discriminative criterion (e.g., accuracy) as opposed to optimizing a related quantity (e.g., likelihood, entropy). While such approaches often produce good results, the trade-off is that the discriminative criterion is usually difficult to optimize, in particular our accuracy-based approach requires a full recognition test to be performed for each candidate partition. The scheme is therefore only feasible if the recognition time

is short. To ensure this is the case, we use a smaller development set (the ''core set'' in the TIMIT distribution) which consists of about 15% of the utterances from the complete test set. We also restrict the cluster sizes to two to limit the number of partitions to consider. We experiment with two algorithms.

First, we define a greedy accuracy-based algorithm, *Max-acc-greedy-pairwise*, 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 sub-vector and the remaining elements are one-component sub-vectors.
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 two pairs overlap. If D is odd, the last remaining component is assigned a sub-vector.

Max-acc-greedy-pairwise requires $O(D^2)$ recognition runs.

The second algorithm, *Max-acc-agglo* is a greedy agglomerative 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 sub-vector. The algorithm stops when accuracy decreases by more than a threshold. The agglomerative accuracy-based algorithm proceeds as follows:

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

1. Let P be the set of sub-vectors selected so far. Initially all elements of P are one-component sub-vectors.
2. Find the pair of elements of P that maximizes recognition accuracy when merged together into a new sub-vector. Replace the pair of elements with the new sub-vector.
3. Go to 2 and repeat until accuracy drops (in practice, a small decrease can be tolerated). The final partition consists of elements of P.

Step 2 of *Max-acc-agglo* requires $O(D^2)$ recognition runs to find the best pair of sub-vectors to merge. Because, in the worst case, we need to perform $O(D)$ such merges, the overall complexity of the algorithm is $O(D^3)$.

Table 1 summarizes all the algorithms described in this section.

Table 1
Summary of algorithms

Algorithm	Description	Parameters
Composed	Vector quantization followed by scalar quantization	None
Greedy-n Pair	Tree search with branching factor n for sub-vectors of size 2	n: Tree branching factor
Greedy-1 Triplet	Greedy search for sub-vectors of size 3	None
Entropy-min-p	Selection of sub-vectors that minimize linear entropy	p: Max sub-vector size
Max-clique-t	Subvecs as cliques in sparse graph encoding pairwise MI	t: Threshold for removing low MI edges
Joint-quant	Vector elements clustered together across dimensions	None
Max-acc-pairwise	Accuracy-based greedy pairwise sub-vector quantization	None
Max-acc-agglo	Accuracy-based agglomerative clustering	None

3. Experimental setup

We use two speech corpora to test our clustering algorithms. NYNEX PHONEBOOK is a phonetically-rich, isolated-word, telephone-speech database (Pitrelli et al., 1995). It contains 93,667 isolated-word utterances and totals 23 h of speech. PHONEBOOK is a good database to use for this study because it 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 h 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 25 ms window and a 10 ms 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. No pruning threshold was used in order to discount the possibility of it affecting performance at different degrees at different quantization rates.

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 described by Bilmes (1999). 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) consists of 1344 utterances. A core test set of 194 utterances was used as a development set for a couple of our experiments. Forty two 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. They account for a very small percentage of the total number of parameters and quantizing them does not affect the WER in a significant way.

We use a hierarchical clustering scheme similar to LBG (Linde et al., 1980)[8] to perform quantization. The distortion measure used is Euclidean distance. Other work has used measures such as the Batacharaya measure (Mak, 1998). We have conducted pilot scalar quantization experiments with two other clustering schemes: kmeans and LVQ (Kohonen et al., 1995). Our scheme performed best and also had the advantage of being simple and not requiring any parameter tuning compared to LVQ.

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%. To our knowledge these baselines are at or near the current state of the art performance for these datasets and dictionaries (see for example (Zhao et al., 1991; Kingsbury et al., 1997; Zweig et al., 1998; Richardson et al., 2000; Livescu and Glass, 2001)).

4. Results

In this section, we evaluate the various clustering methods that were described in Section 2. Except for the last Section (4.1), which summarizes the results on the TIMIT corpus, all the following experiments were conducted using the PHONEBOOK corpus.

Vector, composed, joint scalar, and disjoint scalar quantization. Fig. 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 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 joint and disjoint

[8] The difference lies in that we split one cluster at a time instead of doubling the number of clusters at each epoch.

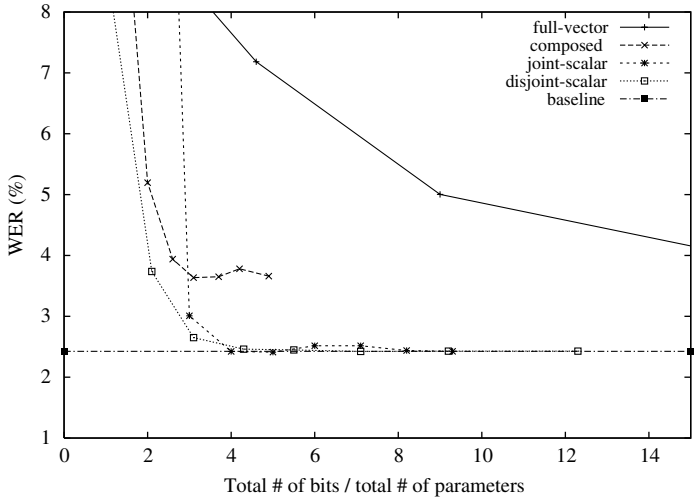

Fig. 1. Comparison between vector, composed, joint scalar and disjoint scalar quantization. Composed quantization is the application of scalar quantization to vector-quantized parameters (in this particular case a 15-bit/parameter vector quantization was used).

schemes achieve baseline word-error rate. Composed quantization shown in Fig. 1 is the result of running standard vector quantization at 15 bits per parameter followed by scalar quantization (at varying quantization levels shown in the graph). It achieves the same error rate vector quantization achieves using 15 bits but with only 3 bits per parameter. However, the WER does not decrease further even with additional bits. This is clearly a limitation of the composed scheme that the lowest WER it can achieve is bounded by the WER of the vector quantization it builds upon.

Pairwise dependencies. The extent to which sub-vector quantization improves upon scalar quantization depends on the strength of correlation between vector components. From Fig. 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 were generated randomly according to a normal distribution with diagonal covariance. The 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.

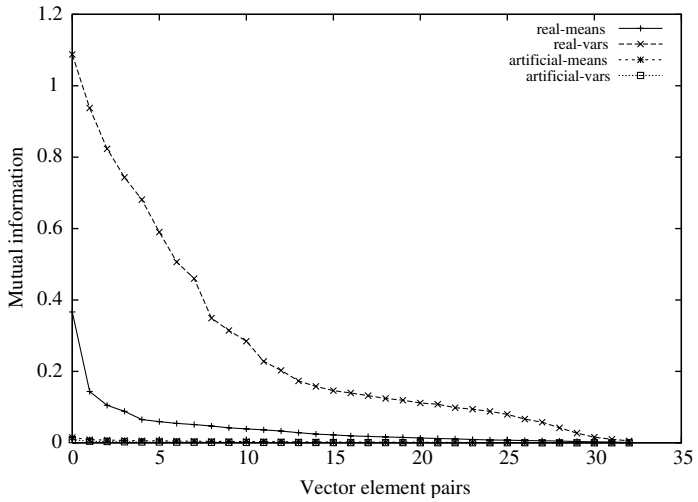

Fig. 2. Mutual information values of ordered pairs of vector elements for real and artificial data (for both means and variances). Artificial data are randomly generated according to a diagonal covariance Gaussian distribution. For clarity, only every tenth of the $\binom{26}{2}$ pairs are shown.

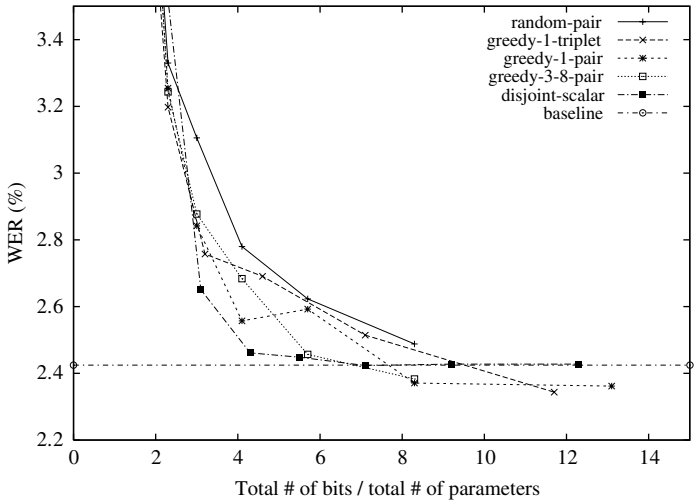

Fig. 3. Comparison of greedy-n schemes with disjoint scalar quantization and a procedure that randomly selects vector element pairs.

Greedy-n schemes. We compare several greedy-*n* schemes with disjoint scalar quantization and a procedure that selects pairs randomly (Fig. 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 Section 4.1, we look at some hand-crafted vector partitions that make intuitive sense and we see again that unless there are underlying correlations according to which the partitions are built, performance is poor.

Comparing the greedy-*n* algorithms, where *n*, the search branching factor, ranges from 1 to 8 (for $n \in \{3 \ldots 8\}$ the clusters turned out to be exactly the same), we see that there is not a significant difference[9] 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 a ''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.

It is interesting to note that several quantization schemes perform better than the baseline (for example *Greedy-3-pair* and *Greedy-1-pair* at around 8 bits per parameter). This might indicate a type of regularization taking place. It would be interesting to retrain our parameters under this quantization constraint. Vasilache (2000), however, uses a similar retraining procedure (*dynamic quantization*) and does not find it to improve over simply quantizing the parameters at the end.

Greedy-1 triplet does the worse among the greedy schemes and although not shown in Fig. 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, at a rate of 8 bits per parameter or higher, the greedy schemes slightly outperform the disjoint scalar algorithm, 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.

It might be possible that a better heuristic or an exhaustive sub-vector partition search does better in comparison with the disjoint scalar scheme; however the worsening performance as the dimension of the sub-vectors increases suggests that the higher-dimensional schemes are penalized by their greater codebook storage

[9] By this we do not mean that there is no statistical difference, but that from a practical speech recognition application point of view the difference in error rate is so small that there is no reason to prefer one approach over the other.

requirements. Recall, there are two components to the total memory storage required by a clustering scheme, index memory and codebook storage. For a fixed table size, smaller sub-vector schemes can encode more codewords than larger sub-vector schemes and conversely, given the same bit allocation per scalar, higher dimensional schemes will require larger codebooks. For example, using two bits per scalar, both disjoint scalar and greedy-1 pair clusterings yield $(2^2)^{26} = (2^4)^{13}$ possible vector codewords (assuming an input parameter vector dimension $d = 26$), but the first scheme requires codebooks with total number of entries $2^2 \times 26$ while the later scheme requires $2^4 \times 26$ entries.

On the other hand, because of the vector quantizer advantage discussed in Section 2.1, higher dimensional sub-vector clustering schemes achieve a smaller distortion at the same bit rate, or equivalently, they can use lower bit rates for the same overall distortion. There are, therefore, trade-offs along two axes: storage-wise, larger sub-vectors reduce index memory requirements but increase table size, and performance-wise, smaller sub-vectors increase distortion but a reduced storage cost. We can see these trade-off at play in Fig. 3: as the quantization rate gets higher, the cost of storing the indices starts to dominate and that is when the larger sub-vector schemes start to perform better.

Mutual information-based versus correlation-based sub-vector selection. Fig. 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 (Cover and Thomas, 1991). Note that the second way is equivalent to computing simple correlation, which captures only first order (linear) dependencies. MFCC features are processed using a discrete cosine transform (DCT) to reduce the linear correlation between vector components and we would expect the orthogonalization effect to carry over from the feature space to the parameter space. We would therefore 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 0.7 using linear MI and around 1.0 for the Gaussian mixture-based MI, i.e., the non-linear portion of the dependencies is about 0.3. 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 Fig. 4 reflects the similar dependency patterns.

Entropy minimization clustering. The linear entropy minimization scheme selects sub-vectors with maximum length p. In Fig. 5, results are shown for $p = 2$, 3, and 6. 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,

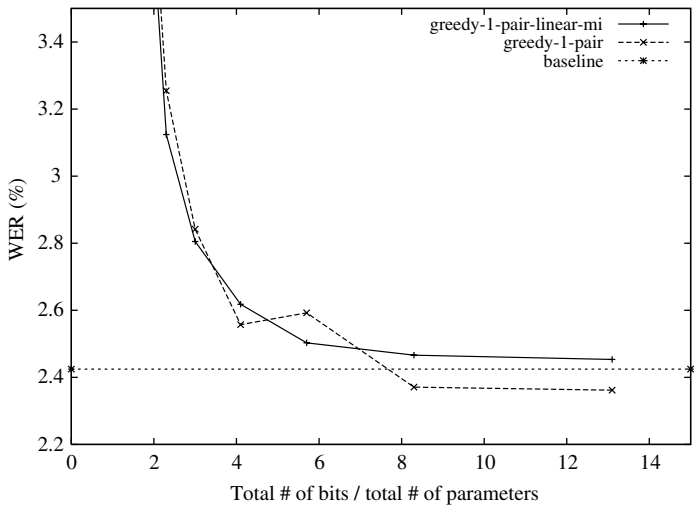

Fig. 4. Comparison between pairwise quantization using MI and correlation.

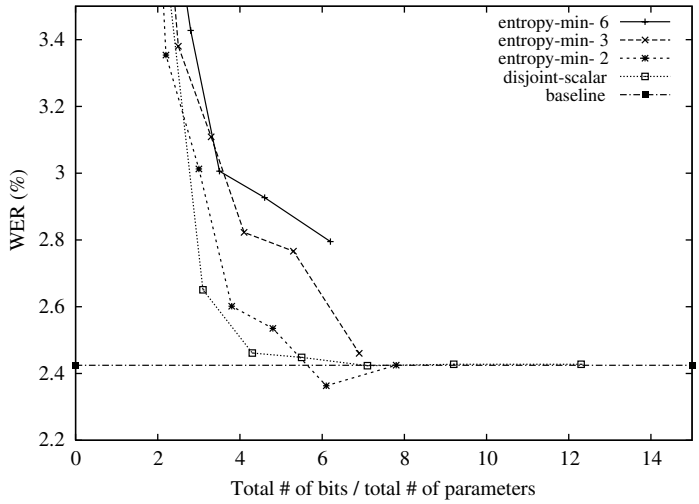

Fig. 5. Sub-vector quantization using linear entropy minimization. Entropy-min-4 and entropy-min-5 are not shown for clarity but they follow the same trend as entropy-min2,3,6.

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.

Maximum-clique clustering. We compare two max-clique clustering schemes with disjoint scalar quantization (Fig. 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 worse performance.

Joint quantization. Lastly, in Fig. 7, we compare joint versus 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

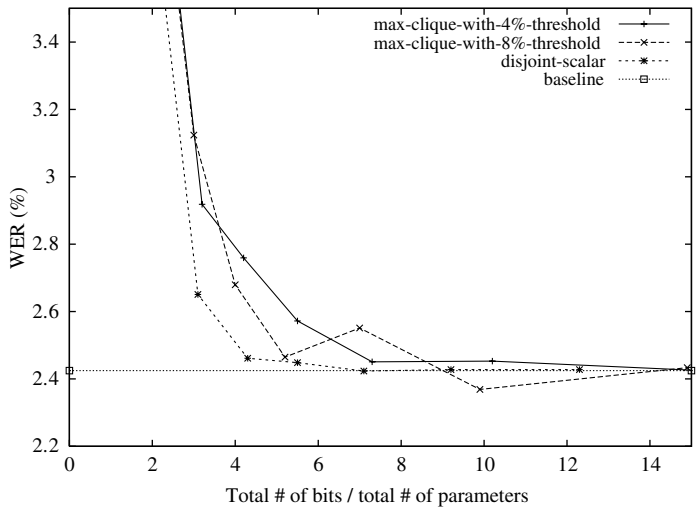

Fig. 6. Threshold-based maximum clique quantization.

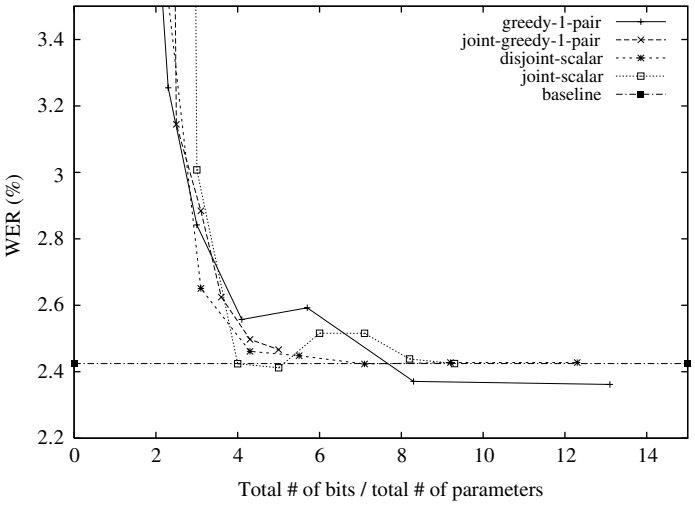

Fig. 7. Comparison between joint and disjoint quantization schemes.

12.5% of the baseline memory, while the disjoint scheme achieves an WER of 2.49% with 13.3% of the baseline memory. However, 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.[10]

4.1. TIMIT results

The experiments on the PHONEBOOK database show scalar quantization to perform best even though one would expect sub-vector quantization do better because it can take advantage of the dependencies in the data. In this section, we apply a subset of our quantization algorithms to the TIMIT corpus.[11] 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 Fig. 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.

For clarity, standard vector quantization is not shown in Fig. 8 because its performance is far inferior to that of the other schemes. At around 5 total bits per parameter, the word error rate is as high as 67% and it only decreases to 50% at 12 bits per parameter. This is not very surprising since the parameter vectors are 39-dimensional the table size penalty accrues with increasing dimension because of the exponential increase in the number of possible codewords.

One technique used to reduce the impact of the table storage and which does not involve breaking vectors up into sub-vectors is *residual* or *multiple stage* quantization (Juang and Gray, 1982). Residual quantization operates over two or more stages. In the first one a crude standard vector quantization is performed and in the subsequent stages the residual vectors obtained by taking the difference between the original vectors and their best approximations[12] so far are quantized. By using lower quantization rates at each stage, residual quantization can reduce storage requirements and still produce good approximations. We have experimented with

[10] 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 are sub-vectors instead of just one codebook.

[11] 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.

[12] The sum of the original quantized vectors and the quantized residuals in the previous stages.

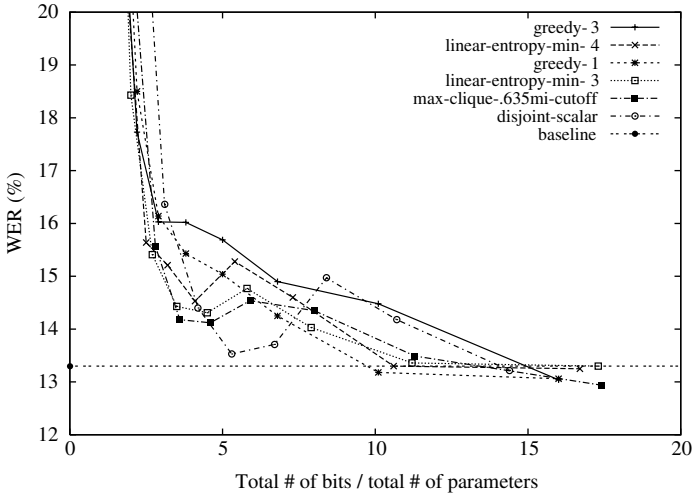

Fig. 8. TIMIT entropy-based disjoint schemes comparison.

multiple stage quantization to reduce the memory required by the standard vector quantizer. The staged approach does reduce storage: it achieves a lower WER (60% versus 67%) using fewer total bits per parameter (3 versus 5); however, compared to scalar quantization this scheme performs far worse (at 5 bits per parameter, it achieves 66% WER compared to 13.53%).

Dependency patterns. It is interesting to look at the most dependent vector component pairs. Table 2 shows mean vector component pairs chosen by the greedy-1 algorithm and their corresponding MI values. For mean vectors, except for a few cases, parameters for 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 3). 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 Bocchieri and Mak (2001) and Mak (1998), very similar results to the mean correlations are reported on a different speech corpus, Air Travel Information System

Table 2
TIMIT's most dependent mean vector components as measured using mutual information

Pair	MI
c10,ΔΔc10	0.92
c12,ΔΔc12	0.90
c11,ΔΔc11	0.88
c8,ΔΔc8	0.82
c7,ΔΔc7	0.73
c9,ΔΔc9	0.72
c5,ΔΔc5	0.70
c6,ΔΔc6	0.68
c2,ΔΔc2	0.67
c1,E	0.67
c4,ΔΔc4	0.66
Δc4,ΔE	0.54
c3,ΔΔc3	0.45
Δc1,Δc7	0.33
ΔΔc1,ΔΔE	0.33
Δc2,Δc8	0.33
Δc3,Δc5	0.23
Δc6,Δc11	0.10
Δc10,Δc12	0.094
Δc9	N/A

Table 3
TIMIT's most dependent variance vector components as measured using mutual information

Pair	MI
Δc8,Δc9	1.1
ΔE,ΔΔE	1.1
Δc10,Δc11	1.1
Δc5,Δc6	1.1
ΔΔc8,ΔΔc9	1.1
ΔΔc5,ΔΔc6	1.1
ΔΔc10,ΔΔc11	1.1
ΔΔc2,ΔΔc3	1.1
Δc1,ΔΔc1	1.0
Δc2,Δc4	0.98
ΔΔc4,ΔΔc7	0.86
Δc7,Δc12	0.85
c11,c12	0.60
c8,c9	0.57
c5,c7	0.45
Δc3,ΔΔc12	0.43
c6,c10	0.40
c1,E	0.34
c2,c4	0.25
c3	N/A

(ATIS). Unlike us, however, they do not compute correlations directly on the mean and variance parameter vectors but on the feature vectors themselves. It might seem surprising that the correlations between feature vector components are similar to those between mean parameter vector components and not variance vector components, but as class-conditional averages of the feature vectors, means vectors are likely to preserve the same correlations whereas there is no reason for variance components to correlate in a similar fashion to the data.

Heuristic partitions. Before we look at the accuracy-driven schemes, we compare three "natural" partition schemes to their data-driven counterparts:

(1) Clustering consecutive pairs together.
(2) Clustering MFCCs, deltas, and double deltas as three separate sub-vectors.[13]
(3) Clustering the triples (MFCC,DELTA,DDELTA), i.e., each component and its first and second derivatives form a sub-vector.
(4) Clustering deltas as one sub-vector, double deltas as a second one, and keeping MFCCs disjoint.

The first partition scheme is motivated by the existence of strong correlation between consecutive variance parameter vector components. The second and third schemes are widely used for both feature and parameter sub-vector quantization because of the assumptions that the strongest correlations exist between each vector component and its time derivatives, and within each of the three types of components: base components, first derivatives, and second derivatives. The last scheme is an attempt to first of all exploit the fact that, for variance parameters, the strongest correlations are among the delta or the double delta group. Moreover, we want to test the hypothesis that the base components of the parameter vectors are the most important to quantize with minimal distortion and hence we quantize them using scalar quantization, the best scheme according to our experiments. For the time derivatives, we are willing to incur a higher distortion by using vector quantization but save on storage.

As can be seen in Fig. 9, 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 second and third partition schemes perform poorly, which does not come as a surprise as we have already seen that in our tasks

[13] To be accurate we should write "parameters for MFCCs, deltas...", but for convenience we allow this abuse of language.

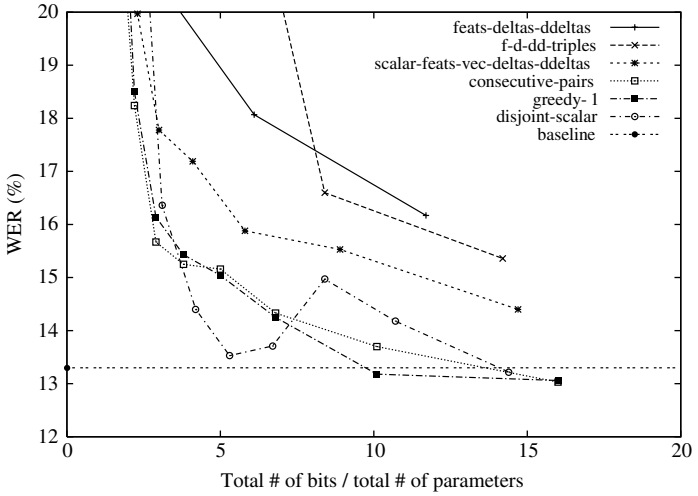

Fig. 9. TIMIT heuristic sub-vector partition disjoint schemes comparison.

performance quickly deteriorates with increasing sub-vector dimensions. The last quantization scheme does not do as well as one would expect if our hypothesis that the parameters for time derivatives are not very susceptible to quantization noise were true.

Accuracy-driven quantization. We have tried two recognition accuracy-based algorithms for finding optimal vector partitions, *Max-accuracy-greedy-pairwise* and *Max-accuracy-agglomerative*, 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, iteratively builds a partition by pairing vector components which result in the least decrease in accuracy compared to the partition in the previous step. 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-pairwise 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

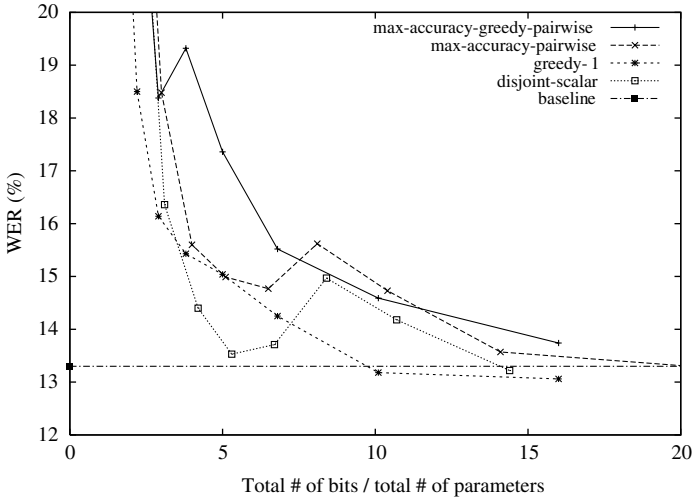

Fig. 10. TIMIT accuracy-based disjoint schemes comparison.

maximum accuracy criterion is not cumulative and a greedy approach is inappropriate. This hypothesis is reinforced by the fact that the less greedy Max-accuracy-agglomerative achieves a lower error rate than Max-accuracy-greedy-pairwise at most quantization levels (Fig. 10). However, it is telling that even for Max-accuracy-agglomerative, the maximum sub-vector size obtained in the final partition was 2, and there were only two such sub-vectors. Higher levels of clustering caused the error rate to reach the level of the Max-accuracy-greedy-pairwise algorithm.

5. Conclusion

We have defined and evaluated a number of novel methods for producing sub-vector-based parameter quantization in Gaussian-mixture HMM-based ASR systems. We find that, whether we use entropy-based or accuracy-driven sub-vector search methods, three schemes are the overall best for reducing memory: disjoint scalar, joint scalar and joint greedy-1-pair quantization. They do better than more elaborate heuristics and achieve near-baseline error rate with less than a seventh of the storage requirements.

Even though, compared to the joint scheme, the disjoint scalar scheme requires more table 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 seems surprising that a disjoint scheme would do so well given the constraint of not sharing codewords and the resulting higher table storage requirements. That same constraint, however, is key to the disjoint scheme's good performance overall because it allows fewer bits per parameter compared to a joint scheme where enough bits need to be allocated to index the single codebook (which, although smaller than all the disjoint codebooks taken together, is larger than any single one of them).

Similar constraints explain the better storage/performance trade-off of scalar and low-dimensional schemes compared to standard VQ and high-dimensional sub-vector quantizers. By partitioning vectors, the optimality of vector quantization is lost but a dramatic reduction in storage ensues as the VQ's exponentially growing codebook is replaced by a Cartesian product of smaller codebooks.

We have mentioned another type of product VQ, namely residual quantization, which quantizes the full vectors over multiple stages to reduce table storage (as well as codeword search complexity). While, in our experiments, it too has suffered from the curse of dimensionality, it did perform better than standard VQ. It is an interesting future research direction to explore the interaction between residual quantization and sub-vector quantization as a way to further mitigate the impact of table storage and at the same time use higher dimensional sub-vectors that better capture dependencies.

Finally, while we have focused on a low-power application of ASR parameter quantization, large vocabulary speech recognition (LVSR) typically uses a very large number of parameters and could also benefit from the simple quantization schemes that were found to be effective in our setting. It is already standard practice in LVSR decoding to vector quantize Gaussian parameters and limit the number of full Gaussian evaluations required by pruning or approximating low-likelihood clusters or HMM states. Two such approaches are described in Knill et al. (1996) and Saon et al. (2005). It would interesting to combine these computation-saving approaches with memory-saving scalar quantization. Our *composed clustering* scheme is one step in that direction.

Acknowledgement

This material was supported by National Science Foundation grant ITR/RC-0086032.

References

Bilmes, J.A., 1999. Buried Markov models for speech recognition. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Phoenix, AZ, March.

Bocchieri, E., Mak, B., 1997. Subspace distribution clustering for continuous observation density hidden Markov models. In: Proceedings of the Eurospeech '97, Rhodes, Greece, pp. 107–110.

Bocchieri, E., Mak, B., 2001. Subspace distribution clustering hidden Markov model. IEEE Transactions on Speech and Audio Processing 9 (3), 264–275.

Chung, F.R., 1997. Spectral Graph TheoryCBMS Regional Conference Series in Mathematics, vol. 92. American Mathematical Society.
Comerford, L., Frank, D., Gopalakrishnan, P., Gopinath, R., Sedivy, J., 2001. The IBM personal speech assistant. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, Utah.
Cover, T.M., Thomas, J.A., 1991. Elements of Information Theory. Wiley, New York.
Digalakis, V., Neumeyer, L., Perakakis, M., 1998. Product-code vector quantization of cepstral parameters for speech recognition over the web. In: Proceedings of the International Conference on Spoken Language Processing, December.
Gray, R.M., Gersho, A., 1991. Vector Quantization and Signal Compression. Kluwer, Dordrecht, Netherlands.
Huang, X., Acero, A., Chelba, C., Deng, L., Duchene, D., Hon, H., et al. 2000. Mipad: A next generation PDA prototype. In: Proceedings of the International Conference on Spoken Language Processing, Beijing, China, November.
Juang, B.H., Gray Jr., A.H., 1982. Multiple stage vector quantization for speech coding. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 1 (April), 597–600.
Juang, B.H., Wong, D.Y., Gray Jr., A.H., 1982. Distortion performance of vector quantization for LPC voice coding. IEEE Transactions on Acoustics, Speech, and Signal Processing 30 (2), 307–309.
Kingsbury, Brian E.D., Morgan, N., Greenberg, S., 1997. Improving ASR performance for reverberant speech. In: ESCA workshop of Robust Speech Recognition, Pont-a-Mousson, pp. 87–90.
Knill, K., Gales, M., Young, S., 1996. Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs. In: Proceedings of the International Conference on Spoken Language Processing, Philadelphia, PA, October, pp. 470–473.
Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., Torkkola, K., 1995. LVQ PAK: The learning vector quantization program package.
Li, X., Malkin, J., Bilmes, J., 2004. Codebook design for ASR systems using custom arithmetic units. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal, Canada, May.
Linde, Y., Buzo, A., Gray, R.M., 1980. An algorithm for vector quantizer design. IEEE Transactions on Communications COM-28 (1), 84–95.
Livescu, K., Glass, J., 2001. Segment-based recognition on the phonebook task: Initial results and observations on duration modeling. In: European Conference on Speech Communication and Technology (Eurospeech), Aalborg, Denmark.
Lookabaugh, T.D., Gray, R., 1989. High-resolution quantization theory and the vector quantizer advantage. IEEE Transactions on Information Theory 35 (5), 1020–1033.
Mak, B., 1998. Towards A Compact Speech Recognizer: Subspace Distribution Clustering Hidden Markov Model. Ph.D. thesis, Massachusetts Institute of Technology, April.
Makhoul, J., Roucos, S., Gish, H., 1985. Vector quantization in speech coding. Proceedings of the IEEE 73 (11), 1551–1588.
Malkin, J., Li, X., Bilmes, J., 2004. Custom arithmetic for high-speed, low-resource ASR systems. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal, Canada, May.
Moyers, G., 2001. A handy way to interact. Speech Technology (July/August), 14–17.
Narasimhan, M., Jojic, N., Bilmes, J., 2005. Q-clustering. In: Neural Information Processing Systems (NIPS). Vancouver, Canada, December.
Ng, A.Y., Jordan, M.I., Weiss, Y., 2001. On spectral clustering: Analysis and an algorithm. In: Advances in Neural Information Processing Systems, vol. 14.
Pitrelli, J., Fong, C., Wong, S.H., Spitz, J.R., Leung, H.C., 1995. PhoneBook: A phonetically-rich isolated-word telephone-speech database. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing.
Ravishankar, M., Bisiani, R., Thayer, E., 1997. Sub-vector clustering to improve memory and speed performance of acoustic likelihood computation. In: Proceedings of the Eurospeech, Rhodes, Greece, pp. 151–154.
Richardson, M., Bilmes, J., Diorio, C., 2000. Hidden-articulator Markov models: Performance improvements and robustness to noise. In: Proceedings of the International Conference on Spoken Language Processing, Beijing, China.
Roy, Kaushik, Johnson, Mark C., 1997. Software design for low power. In: Low Power Design in Deep Submicron Electronics. Proceedings of the NATO Advanced Study Institute. Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 433–460.
Saon, G., Povey, D., Zweig, G., 2005. Anatomy of an extremely fast LVCSR decoder. In: European Conference on Speech Communication and Technology (Eurospeech), Lisbon, Portugal, September, pp. 549–552.
Takahashi, S., Sagayama, S., 1995. Four-level tied-structure for efficient representation of acoustic modeling. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. 520–523.
Varga, I., Aalburg, S., Andrassy, B., Astrov, S., Bauer, J.G., Beaugeant, C., Geissler, C., Hoge, H., 2002. ASR in mobile phones – an industrial approach. IEEE Transactions on Speech and Audio Processing 10 (November), 562–569.
Vasilache, M., 2000. Speech recognition using HMMs with quantized parameters. In: Proceedings of the International Conference on Spoken Language Processing.
Zhao, Y., Wakita, H., Zhuang, X., 1991. An HMM based speaker-independent continuous speech recognition system with experiments on the TIMIT DATABASE. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. 333–336.
Zweig, G., Russell, S.J., 1998. Speech recognition with Dynamic Bayesian Networks, in: AAAI/IAAI, pp. 173–180.