High-Performance Connected Digit Recognition Using Maximum Mutual Information Estimation

Yves Normandin, Régis Cardin, and Renato De Mori, *Senior Member, IEEE*

***Abstract*— Hidden Markov Models (HMM's) are one of the most powerful speech recognition tools available today. Even so, the inadequacies of HMM's as a "correct" modeling framework for speech are well known. In this context, it is argued in this paper that the maximum mutual information estimation (MMIE) formulation for training is more appropriate than maximum likelihood estimation (MLE) for reducing the error rate.**

***Corrective MMIE training* is introduced. It is a very efficient new training algorithm which uses a modified version of a discrete reestimation formula recently proposed by Gopalakrishnan *et al.* Reestimation formulas are proposed for the case of diagonal Gaussian densities and their convergence properties are experimentally demonstrated. A description of how these formulas are integrated into our training algorithm is given. Using the MMIE framework for training, it is shown how weighting the contribution of different parameter sets in the computation of output probabilities introduces substantial recognition improvements.**

Using the TIDIGITS connected digit corpus, a large number of experiments are performed with the ideas, techniques, and algorithms presented in this paper. These experiments show that MMIE systematically provides substantial error rate reductions with respect to MLE alone and that, thanks to the new training techniques, these results can be obtained at an acceptable computational cost. The best results obtained in our experiments were 0.29% word error rate and 0.89% string error rate on the adult portion of the corpus.

I. Introduction

In automatic speech recognition (ASR) systems based on Hidden Markov Models (HMM's), the purpose of training is to find the HMM parameter set Θ which will result in the speech decoder with the lowest possible recognition error rate. The set Θ includes all transition probabilities and output distribution parameters in all HMM's used for a given task. Training is done by maximizing some objective function $R(\Theta)$. There are two important and difficult problems to consider. The first one is to determine a meaningful objective function. This function should be such that, whenever $R(\hat{\Theta}) > R(\Theta)$, then $\hat{\Theta}$ results in a better decoder than Θ. Once a function $R(\Theta)$ has been chosen, the second problem (the estimation problem) is to find the parameter set Θ that maximizes it.

By far the most common HMM parameter estimation technique is maximum likelihood estimation (MLE) [1]. Recently, a different type of estimation, called maximum mutual information estimation (MMIE) has been proposed [2]. There have been attempts at empirically justifying the use of MMIE with simple and well-controlled experiments. Some of them [3], [4] demonstrate that, for certain types of estimation problems, MMIE will converge to the optimal decoder even if incorrect modeling assumptions are made, while MLE will not. These experiments thus tend to show that MMIE is more robust than MLE when modeling assumptions are not correct. The fact that most of HMM's modeling assumptions about speech are incorrect could be an argument in favor of MMIE. In some cases, however, it is also possible that neither MMIE nor MLE will converge to the optimal decoder, but another type of estimator will.

It is not clear how these cases relate to speech recognition problems. In general, optimization algorithms will not converge to the global optimum and it is probably not possible to get an HMM-based optimal decoder for speech recognition. Thus the advantage of using MMIE for HMM-based ASR's should be assessed by experimentation. Many results reported in the literature [2], [4]–[6] tend to demonstrate MMIE's usefulness, but not conclusively.

We will show in this paper that, at least for the connected digit task on the TIDIGITS corpus [7], MMIE leads to significant recognition improvements with discrete and semicontinuous HMM's (SCHMM's). In a connected digit recognition experiment using one discrete model per digit, the string error rate was reduced from 1.92% to 1.48% by using MMIE after our standard MLE training. Further improvements (0.89% string error rate with two models per word) were obtained by using a new MMIE algorithm especially conceived for SCHMM's.

II. Relation Between MLE and MMIE

We assume that the result of a speaker pronouncing a word sequence (or message) $\mathbf{w}$ is an acoustic *observation sequence* $\mathbf{y} \equiv \underline{y}_1, \underline{y}_2, \cdots, \underline{y}_{L_{\mathbf{y}}}$. Typically, $\mathbf{y}$ is the result of a frame-based analysis performed on the speech signal produced by the speaker, where $\underline{y}_l$ is the parameter vector extracted from the lth frame. Let us assume that an HMM-type model can be built corresponding to any possible word sequence in the task, and let $\mathbf{m}_{\mathbf{w}}$ be the model corresponding to the word sequence $\mathbf{w}$. This model allows the computation of $P_\Theta(\mathbf{y}|\mathbf{m}_{\mathbf{w}})$, the probability that the model $\mathbf{m}_{\mathbf{w}}$ produced $\mathbf{y}$. Generally, $P_\Theta(\mathbf{y}|\mathbf{m}_{\mathbf{w}})$ is intended as an "estimate" of the probability $P(\mathbf{y}|\mathbf{w})$ that the pronunciation of the word sequence $\mathbf{w}$ resulted in $\mathbf{y}$. The reason for this is that the speech

Manuscript received October 10, 1991; revised April 15, 1993. The associated editor coordinating the review of this paper and approving it for publication was Dr. David Nahamoo.

The authors are with the Centre de recherche informatique de Montréal (CRIM), McGill College, Montréal, Québec, Canada, H3A 2N4.

IEEE Log Number 9215241.

decoder with least probability of error P_e,[1] the *maximum a posteriori* (MAP) decoder, is given by

$$\hat{\mathbf{w}} = \underset{\mathbf{w}'}{\operatorname{argmax}} P(\mathbf{w}'|\mathbf{y}) = \underset{\mathbf{w}'}{\operatorname{argmax}} \frac{P(\mathbf{y}|\mathbf{w}')P(\mathbf{w}')}{P(\mathbf{y})} = \underset{\mathbf{w}'}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{w}')P(\mathbf{w}') \quad (1)$$

where $P(\mathbf{w}')$ is the *a priori* probability of the word sequence $\mathbf{w}'$(this probability is usually provided by a language model). Even though this "estimate" is, by any standard, quite inaccurate, it is nonetheless possible to build very effective speech recognizers based on (1). That is, assuming that a language model $P(\mathbf{w})$ is available, recognition is performed by finding $\hat{\mathbf{w}}$ such that

$$\hat{\mathbf{w}} = \underset{\mathbf{w}'}{\operatorname{argmax}} P_\Theta(\mathbf{y}|\mathbf{m}_{\mathbf{w}'})P(\mathbf{w}'). \quad (2)$$

Clearly, training should aim at finding the HMM parameter set Θ such that, whenever $\mathbf{w}$ is pronounced, we have $P_\Theta(\mathbf{y}|\mathbf{m}_\mathbf{w})P(\mathbf{w}) > P_\Theta(\mathbf{y}|\mathbf{m}_{\mathbf{w}'})P(\mathbf{w}')$, for any $\mathbf{w}' \neq \mathbf{w}$, as often as possible. Let us assume that training is done using a set of N *independent* observation sequences $\mathbf{y}^r, r = 1, 2, \cdots, N$, where $\mathbf{y}^r \equiv \underline{y}_1^r, \underline{y}_2^r, \cdots, \underline{y}_{L_r}^r$ corresponds to the word sequence $\mathbf{w}_r$ and $L_r \equiv L_{\mathbf{w}_r}$ is the length of the observation sequence. The objective function used in MLE is

$$R(\Theta) = \prod_{r=1}^{N} P_\Theta(\mathbf{y}^r|\mathbf{m}_r) \quad (3)$$

where $\mathbf{m}_r \equiv \mathbf{m}_{\mathbf{w}_r}$ is the model corresponding to $\mathbf{y}^r$ and $P_\Theta(\mathbf{y}^r|\mathbf{m}_r)$ is the probability that $\mathbf{m}_r$ produced $\mathbf{y}^r$. Thus, using the models corresponding to the observation sequences, MLE increases the *a posteriori* probability of the training data. It is not intuitive how (3) relates to the objective of reducing the error rate. This is especially true since the models not corresponding to the training data are not used for parameter estimation. A possible argument is that it has been shown that, if certain assumptions are satisfied, then (3) will in fact produce the best decoder [8]. However, these assumptions are almost always violated in practical speech recognition applications.

On the other hand, the objective function used in MMIE is

$$R(\Theta) = \prod_{r=1}^{N} P_\Theta(\mathbf{m}_r|\mathbf{y}^r) = \prod_{r=1}^{N} \frac{P_\Theta(\mathbf{y}^r|\mathbf{m}_r)P(\mathbf{m}_r)}{\sum_{\mathbf{w}'} P_\Theta(\mathbf{y}^r|\mathbf{m}_{\mathbf{w}'})P(\mathbf{m}_{\mathbf{w}'})}. \quad (4)$$

MMIE increases the *a posteriori* probability of the model corresponding to the training data, given the data. Since this is also the criterion used in MAP decoding, the relationship between MMIE and error rate is much more intuitive than it is with MLE.

The difference between MMIE and MLE is quite clear. On the one hand, MLE training attempts to find the parameter set Θ such that $P_\Theta(\mathbf{y}|\mathbf{m}_\mathbf{w})$ will approximate $P(\mathbf{y}|\mathbf{w})$ as closely as possible. This "estimate" will then be used to approximate the MAP decoder based on (1). Arguably, this is quite an indirect approach. MMIE training, on the other hand, directly attempts to approximate $P(\mathbf{w}|\mathbf{y})$, the probability used in MAP decoding. In this context, the parametric family of probability distributions of interest is $P_\Theta(\mathbf{m}_\mathbf{w}|\mathbf{y})$, which is expressed as

$$P_\Theta(\mathbf{m}_\mathbf{w}|\mathbf{y}) = \frac{P_\Theta(\mathbf{y}|\mathbf{m}_\mathbf{w})P(\mathbf{w})}{\sum_{\mathbf{w}'} P_\Theta(\mathbf{y}|\mathbf{m}_{\mathbf{w}'})P(\mathbf{w}')}. \quad (5)$$

Our hypothesis is that $P_\Theta(\mathbf{m}_\mathbf{w}|\mathbf{y})$ in (5) can be a much better approximation to $P(\mathbf{w}|\mathbf{m})$ than $P_\Theta(\mathbf{y}|\mathbf{m}_\mathbf{w})$ can be to $P(\mathbf{y}|\mathbf{w})$. The functions $P_\Theta(\mathbf{y}|\mathbf{m}_\mathbf{w})$ and $P(\mathbf{w})$ are of interest only to the extent that they are used in (5). In fact, they don't even have to be distributions. This is important because modifications will be introduced to the expression used to compute $P_\Theta(\mathbf{y}|\mathbf{m}_\mathbf{w})$ which will no longer retain the properties of adistribution.

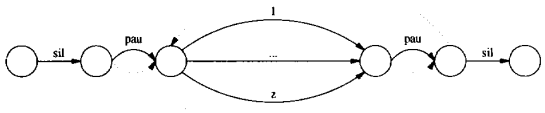


Fig. 1. Looped model used for connected digit recognition performed using a Viterbi search.

III. Use of MMIE for ASR

For several reasons, training HMM's with MMIE involves a higher computational complexity than using MLE. Let $R(\Theta)$ be the MMIE objective function, as expressed in (4). One thing that makes MMIE computationally more complex is the sum over $\mathbf{w}'$ in the denominator which can have a very large number of terms.

Let us define a model $\mathbf{m}_{\text{gen}}$ such that $P_\Theta(\mathbf{y}|\mathbf{m}_{\text{gen}}) = \sum_{\mathbf{w}'} P_\Theta(\mathbf{y}|\mathbf{m}_{\mathbf{w}'})P(\mathbf{m}_{\mathbf{w}'})$. In some applications (such as connected digit recognition), for which the language model is very simple (e.g., $P(\mathbf{m}_\mathbf{w})$ independent from $\mathbf{m}_\mathbf{w}$), it may be possible to create a compact model $\mathbf{m}_{\text{gen}}$ in which there is a path corresponding to every path in every possible model $\mathbf{m}$ in the application. One example is the looped model used for connected digit recognition shown in Fig. 1. In general, if recognition is performed with a Viterbi search on some model $\mathbf{m}_{\text{gen}}$, then the same model could be used for the denominator of (4); however, such a model could still be much too big for a practical implementation of MMIE training.

If it is not possible to build a satisfactory $\mathbf{m}_{\text{gen}}$ of a reasonable size, then the denominator of (4) will usually be approximated by using a much smaller number of component models [2], [4], [5]. In this case, the sum should be taken over the most probable models, which can be determined using, for example, a so-called "N-best" search [9], [10].

Another reason why the practical use of MMIE is difficult is that there are no reestimation formulas of the type used with MLE. This may impose the use of general optimization techniques such as gradient descent, which, because of slow convergence, are computationally expensive. Indeed, while MLE training can usually be done with less than four iterations, MMIE could easily require dozens. For example, the stopping criterion used by Brown [4] in his MMIE training experiments is when each of 10 consecutive iterations results in less than a 2% improvement in the objective function. Since

[1] $P_e = P(\hat{\mathbf{w}} \neq \mathbf{w})$, where it is assumed that $\mathbf{w}$ is the word sequence spoken and $\hat{\mathbf{w}}$ is the word sequence selected by the recognizer.

each MMIE iteration typically requires three times the time of an MLE iteration, the result is that Brown's MMIE training takes around 35 times more time than MLE training [4]. This might be unaffordable in many cases.

A. A Reestimation Formula for Discrete HMM's

Let Θ be the parameter set of the HMMs of a given task in which discrete HMM's are used. Let θ be one element in the parameter set Θ. For example, θ may be the probability of a particular codeword in one of the HMM's output distributions, or it may be the probability of a transition. We are interesting in finding a way to estimate a new parameter set Θ' such that $R(\Theta') \geq R(\Theta)$, with $R(\Theta)$ given in (4). Such a way was recently proposed by Gopalakrishnan *et al.* [10] (from IBM), in the form of the following reestimation-like formula

$$\hat{\theta} = \frac{\theta\left(\frac{\partial \log R(\Theta)}{\partial \theta} + D\right)}{\sum_{\theta'} \theta'\left(\frac{\partial \log R(\Theta)}{\partial \theta'} + D\right)} \tag{6}$$

where D is a constant to be determined, and the sum is taken over all parameters belonging to the same distribution as θ. For example, for a codeword probability, the sum is taken over all codewords belonging to the same codebook of the same output distribution. Gopalakrishnan *et al.* have shown, using a development based on Baum and Eagon [11], that there is a value $D_{\min}$ such that, if $D \geq D_{\min}$, (6) is guaranteed to give $R(\Theta') \geq R(\Theta)$. Using $R(\Theta)$ in (4), the gradient becomes

$$\frac{\partial \log R(\Theta)}{\partial \theta} = \sum_{r=1}^{N} \left\{ \frac{1}{P_\Theta(\mathbf{y}^r|\mathbf{m}_r)} \frac{\partial P_\Theta(\mathbf{y}^r|\mathbf{m}_r)}{\partial \theta} - \frac{\sum_{\mathbf{m}'} P(\mathbf{m}') \frac{\partial P_\Theta(\mathbf{y}^r|\mathbf{m}')}{\partial \theta}}{\sum'_{\mathbf{m}} P(\mathbf{m}') P_\Theta(\mathbf{y}^r|\mathbf{m}')} \right\} \tag{7}$$

where Θ is one of the parameters in θ. Using $\mathbf{m}_{\text{gen}}$, (7) reduces to

$$\frac{\partial \log R(\Theta)}{\partial \theta} = \sum_{r=1}^{N} \left\{ \frac{1}{P_\Theta(\mathbf{y}^r|\mathbf{m}_r)} \frac{\partial P_\Theta(\mathbf{y}^r|\mathbf{m}_r)}{\partial \theta} - \frac{1}{P_\Theta(\mathbf{y}^r|\mathbf{m}_{\text{gen}})} \frac{\partial P_\Theta(\mathbf{y}^r|\mathbf{m}_{\text{gen}})}{\partial \theta} \right\}. \tag{8}$$

From (8), we have, for discrete output probabilities

$$\frac{\partial \log R(\Theta)}{\partial \theta} = \frac{1}{\theta}(c_\theta - c_\theta^{\text{gen}}) \tag{9}$$

where c_θ represents the standard MLE count for parameter θ and c_θ^{gen} is the corresponding count obtained using the general model. That is, if θ is the parameter corresponding to the probability $p(k|b)$ of codeword k in distribution b, then

$$c_\theta = \sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_{(l|y_l^r = k)} \frac{P_{\Theta,\, l}(t, \mathbf{y}^r|\mathbf{m}_r)}{P_\Theta(\mathbf{y}^r|\mathbf{m}_r)}$$

$$c_\theta^{\text{gen}} = \sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_{(l|y_l^r = k)} \frac{P_{\Theta,l}(t, \mathbf{y}^r|\mathbf{m}_{\text{gen}})}{P_\Theta(\mathbf{y}^r|\mathbf{m}_{\text{gen}})} \tag{10}$$

where b_t is the output distribution associated to transition t, $P_{\Theta,\, l}(t, \mathbf{y}^r|\mathbf{m}_r)$ is the probability that $\mathbf{y}^r$ was generated by $\mathbf{m}_r$ using a transition sequence in which t was taken at time l, and $\sum_{(t|b_t \equiv b)}$ means a sum over all transitions sharing the output distribution b. Let us define a quantity $\Psi_{\Theta,\, l}(t, \mathbf{y}^r)$ as

$$\Psi_{\Theta,\, l}(t, \mathbf{y}^r) = \frac{P_{\Theta,\, l}(t, \mathbf{y}^r|\mathbf{m}_r)}{P_\Theta(\mathbf{y}^r|\mathbf{m}_r)} - \frac{P_{\Theta,\, l}(t, \mathbf{y}r|\mathbf{m}_{\text{gen}})}{P_\Theta(\mathbf{y}^r|\mathbf{m}_{\text{gen}})}. \tag{11}$$

Then, from (6) and (11), we obtain

$$\hat{p}(k|b) = \frac{\sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_{(l|y_l^r = k)} \Psi_{\Theta,\, l}(t, \mathbf{y}^r) + D p(k|b)}{\sum_{k'} \sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_{(l|y_l^r = k')} \Psi_{\Theta,l}(t, \mathbf{y}^r) + D}. \tag{12}$$

What needs to be determined is the value of D. It is clear from (12) that the greater D, the less $\hat{p}(k|b)$ will differ from $p(k|b)$; thus, for fast convergence, D needs to be as small as possible. Gopalakrishnan *et al.* have shown that there is a value $D(\Theta)$ such that, for any $D > D(\Theta)$, (12) is guaranteed to converge. However, as we shall see, $D(\Theta)$ is usually so large that using $D > D(\Theta)$ renders (12) practically useless [12]. For smaller D, there is no theoretically proven convergence; however, Gopalakrishnan *et al.* report that using

$$D = \max_{\theta}\left\{ -\frac{\partial \log R(\Theta)}{\partial \theta},\ 0 \right\} + \varepsilon \tag{13}$$

where ε is a small positive constant, results in fast convergence. Even though our experiments using (12) with (13) also consistently demonstrated convergence, we generally found that convergence was too slow to be useful. Following an argument of Merialdo [6], we conjectured that by removing emphasis from the low-valued parameters in the gradient vector, convergence could be improved. Merialdo had found that when a parameter θ is very small, the division by θ in (9) often causes the corresponding gradient coordinate to have a large magnitude. The consequence is that the search is often concentrated on coordinates corresponding to very low-valued parameters; however since these values are small, they are also unreliably estimated. Merialdo argues that the search should put more emphasis on better estimated, high-valued parameters.

In his gradient descent based MMIE training experiments, Merialdo improved convergence by replacing (9) by

$$\frac{\partial \log R(\Theta)}{\partial \theta} \approx \frac{c_\theta}{\sum_{\theta' \in b(\theta)} c_{\theta'}} - \frac{c_\theta^{\text{gen}}}{\sum_{\theta' \in b(\theta)} c_{\theta'}^{\text{gen}}}, \tag{14}$$

where the notation $\sum_{\theta' \varepsilon b(\theta)}$ means a summation over all parameters θ' belonging to the same distribution as θ. We naturally thought that (14) could also improve convergence when (12) is used instead of gradient descent. This proved to be indeed the case. All our experiments demonstrate that convergence, though not guaranteed at each iteration, is substantially improved. We also experimented with different variants of (14) based on the same idea. One example is

$$\frac{\partial \log R(\Theta)}{\partial \theta} \approx \frac{1}{\theta}(c_\theta - c_\theta^{\text{gen}}) \frac{(c_\theta + c_\theta^{\text{gen}})}{\sum_{\theta' \varepsilon b}(c_{\theta'} + c_{\theta'}^{\text{gen}})}. \tag{15}$$

Convergence results using (14) and (15) instead of (9) are illustrated in Fig. 2. The application is connected digits recognition and the training set includes all utterances from 10 male and

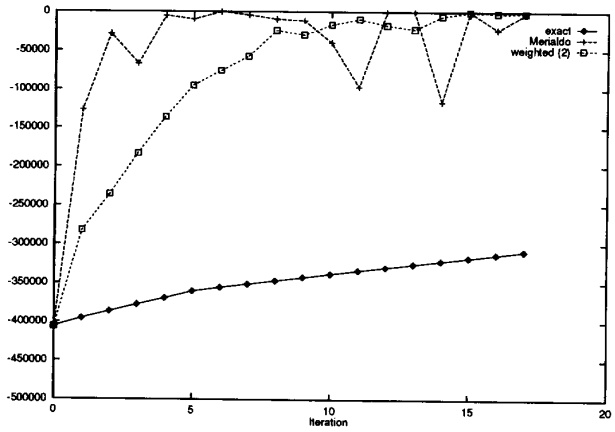


Fig. 2. Value of $\log_{1.001} R(\Theta)$ as a function of the iteration number.

10 female adult speakers. As the graph illustrates, (14) quickly gives a very good estimate. While additional iterations often produce degradations, the resulting estimates are always much better than those obtained with (9). Use of (15) results in a much smoother convergence than (14) which, even though slightly slower than (14), is still much faster than (9).

B. Corrective MMIE Training

From the above results, we have made the following observations:

1) As proposed by the IBM researchers, their reestimation formula does seem to converge. Unfortunately, considering the computation time required to perform each MMIE iteration, the convergence is much too slow to be really useful.
2) By changing the gradient expression in the reestimation formula, dramatically faster convergence is obtained. However, the resulting formula appears quite unstable near the maximum value of the objective function. In other words, the modified formula appears to be most effective when the objective function is far from the optimum value or, rather, when the need for improvement is greatest.

Based on these observations, we considered two possible avenues. The first one would use the reestimation formula with the Merialdo gradient for the first few iterations and then, when the value of the objective function appears "good enough," we would switch to a smoother reestimation formula. Although this appeared a natural thing to do, we preferred a different avenue, based on the following mathematical observation.

Examining expression (7) for the gradient, it can be observed that, when $P(\mathbf{m}_r) \gg P(\mathbf{m}')$ for $\mathbf{m}' \neq \mathbf{m}_r$, then the second term is dominated by the $\mathbf{m}_r$ contribution and the difference is very close to zero. In other words, the training strings which would be correctly recognized ($P(\mathbf{m}_r) > P(\mathbf{m}')$ for $\mathbf{m}' \neq \mathbf{m}_r$), should contribute far less to the MMIE counts than the other strings and could, as a first approximation, be ignored from the training set (for the iteration considered). This is interesting for two reasons. First, since the proportion of incorrectly recognized strings in the training set is relatively small, a MMIE training iteration could be very fast if it used only those strings. Second, because of its convergence properties, the reestimation formula modified with the Merialdo gradient should be very effective on training sets composed only of incorrectly recognized strings.

The result is the *Corrective MMIE training* algorithm, which, as will be shown in the experiments, is a fast and powerful training algorithm. It is initialized with the HMM's obtained after a pre-determined number of MLE iterations. Subsequently, each iteration is a two-step process. First, recognition is performed on the training set and then reestimation is done using only those sentences that were incorrectly recognized. The set of incorrectly recognized strings is called the *reestimation set*. The aim is to correct as many errors as possible from the training set, hoping that this will improve results on the test set as well. Reestimation is done using (12)–(14), and the HMM parameters obtained are smoothed with the ones from the previous iteration using a weight that is dependent on the number of errors in the training set.

C. Codebook Exponents

In both the discrete and semicontinuous cases, the output distributions used in our HMM's are based on multiple codebooks. This means that the parameters extracted from the speech frames are in fact made of different *parameter sets*, each corresponding to a different codebook. These parameter sets are assumed independent and, using separate distributions for each parameter set, output probabilities are computed by simply multiplying the probabilities obtained with each of these distributions.

Since the parameter sets are assumed independent, they should contain complementary information. It seems very unlikely, however, that all parameter sets contain exactly the same amount of information about the spoken message. Even if all parameter sets are useful (and important) for recognition, some may be more useful than others. Yet, all contribute equally to the total output probability. The question is what can be done about this?

As suggested in [5], one answer might be to weigh the contribution of each parameter set using *codebook exponents*,[2] as follows

$$b(\underline{y}) = \prod_{c=1}^{NC} [P_c(\underline{y}_c|b)]^{\lambda_c}, \tag{16}$$

where b is an output distribution, NC is the number of codebooks, $P_c(\underline{y}_c|b)$ is the distribution associated with the cth parameter set, and λ_c is the corresponding codebook exponent. Note that, if $\lambda_c \neq 1$, then it is no longer true that $\int_{\underline{y}} b(\underline{y}) d\underline{y} = 1$, which, from a theoretical point of view, may seem to be a problem. However, remember that the probabilistic model we are interested in is really $P_\Theta(\mathbf{m}_\mathbf{w}|\mathbf{y})$ as expressed in (5), which, regardless of λ_c, will still sum

[2]The term "codebook" is sometimes used to designate the parameter set quantized by a given codebook. In that context, "codebook exponent" means that the exponent is applied to the probability of the parameter set corresponding to a given codebook.

to 1. Also, it may be argued that in practice, since HMM's are anything but an accurate acoustical model of speech, it can be justified to depart somewhat from a strict probabilistic framework.

There is an additional advantage to (16). It has often been found that transition probabilities have negligible effect on the overall recognition process. Because of that, some systems simply don't use them at all. One important reason, however, why their effect is negligible is that the dynamic range of transition probabilities is very small compared to that of output probabilities.[3] Moreover, as output distributions become more complex (in number of codebooks, number of parameters, etc), this difference in dynamic ranges increases. The λ_cs in (16) can compensate for the dynamic ranges difference in order to improve the model.

We treat exponents as a set of parameters separate from all the other parameters. For every iteration, each of these two parameter sets is estimated independently, assuming the other set fixed. Note that even though each estimate separately optimizes $R(\Theta)$, this may not be true of the combined estimate. In practice, however, this does not appear to be a problem. Exponents are estimated using a simple line search in the gradient direction. At each iteration, the initial step size is chosen so that no exponent changes by more than 40% of its original value. If this doesn't increase $R(\Theta)$, the step size is slowly reduced until a value for the exponents is found such that $R(\Theta)$ is greater than its original value.

D. Continuous Densities

The reestimation formula (6) only applies to discrete distributions. However, it is known [4] that MMIE can result in substantially improved recognition results when continuous HMMs are used. It would thus be useful to have a relation like the one in (6) for continuous densities. In this subsection the problem for the case of diagonal covariance Gaussian densities is considered for the sake of simplicity. Details of mathematical derivations are presented in [12]. The conclusions derived here are also applicable to SCHMM's.

Without loss of generality, 1-D densities will be assumed. Let $b = N(y, \mu_b, \sigma_b)$ be such a density. b can be approximated with arbitrary precision by a discrete distribution. Partition the real axis (domain of the density) into three non-overlapping intervals $I_1 = (-\infty, \mu_b - \nu\sigma_b)$, $I_2 = [\mu_b - \nu\sigma_b, \mu_b + \nu\sigma_b]$ and $I_3 = (\mu_b + \nu\sigma_b, +\infty)$. Choose ν such that all points y_l^r in the training data fall in the second interval I_2. This is always possible since the training material is available and the range of y_l^r is finite. Now, partition I_2 into M nonoverlapping subintervals $I_2k, k = 1, \cdots, M$ of width $\Delta = 2\nu\sigma_b/M$. This construction is illustrated in Fig. 3.

Given a continuous random variable Y_b, we can define a discrete distribution by the M probabilities $a_b(k) = P(Y_b \in I_{2k})$ and we can set those probabilities to

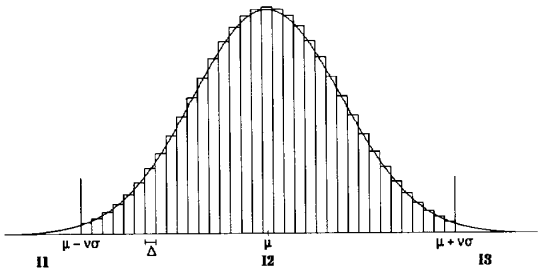


Fig. 3. Partitioning of a Gaussian density into nonoverlapping intervals.

$$a_b(k) = \frac{N(\tilde{y}_k, \mu_b, \sigma_b)\Delta}{\sum_k N(\tilde{y}_k, \mu_b, \sigma_b)\Delta} \tag{17}$$

where $\tilde{y}_k$ is the mid-point of the interval I_{2k}. Let the partitions be the same for all densities and let $k(y)$ be a scalar quantizer mapping y to its partition, that is, if $y \in I_{2i}$, then $k(y) = i$. We can find Δ and ν such that $a_b(k)/\Delta$ approximates $b(y) = N(y, \mu_b, \sigma_b)$ with any desired precision. If, in $P_\Theta(\mathbf{y}^r|\mathbf{m})$ and $P_{\Theta,l}(t, \mathbf{y}^r|\mathbf{m})$, we replace $N(y, \mu_b, \sigma_b)$ by $a_b(k(y))$, we get $P_{\Theta_d}(\mathbf{y}^r|\mathbf{m})$ and $P_{\Theta_d,l}(t, \mathbf{y}^r|\mathbf{m})$, where Θ_d is a discrete parameter vector. Observe that

$$\lim_{\substack{\Delta \to 0 \\ \nu \to \infty}} \frac{P_{\Theta_d,l}(t, \mathbf{y}^r|\mathbf{m})}{P_{\Theta_d}(\mathbf{y}^r|\mathbf{m})} = \frac{P_{\Theta,l}(t, \mathbf{y}^r|\mathbf{m})}{P_\Theta(\mathbf{y}^r|\mathbf{m})} \tag{18}$$

$$\lim_{\substack{\Delta \to 0 \\ \nu \to \infty}} \frac{P_{\Theta_d}(\mathbf{y}^r|\mathbf{m})P(\mathbf{m})}{P_{\Theta_d}(\mathbf{y}^r|\mathbf{m}_{\text{gen}})} = \frac{P_\Theta(\mathbf{y}^r|\mathbf{m})P(\mathbf{m})}{P_\Theta(\mathbf{y}^r|\mathbf{m}_{\text{gen}})}. \tag{19}$$

This means that, in the limit, the MLE counts for the discrete HMM's are the same as the ones from the continuous ones and that $R(\Theta_d) = R(\Theta)$. Now, let us concentrate on output distributions. It is possible to show that, if D is large enough, the problem of maximizing the quantity introduced in relation (4) can be converted into the problem of maximizing

$$S(\Theta) = \sum_b \sum_k \left(\sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_{(l|y_l^r = k)} \psi_{\Theta,l}(t, \mathbf{y}^r) + D a_b(k) \right) \log \frac{\hat{a}_b(k)}{a_b(k)}. \tag{20}$$

The proof is quite lengthy and is given elsewhere [12]. It is based on an inequality for logarithms which, in order to be applied, requires all terms to be positive. The constant D in (20) is used for that purpose. Note that (20) provides a derivation of the reestimation formula (12) different from the one used by Gopalakrishnan *et al.*. Indeed, optimizing (20) subject to the constraint that $\sum_k \hat{a}_b(k) = 1$ gives (12). The derivation used here, however, allows an extension to be carried out to the continuous case, as we will see presently.

Suppose as before that $a_b(k)/\Delta = N(\tilde{y}_k, \mu_b, \sigma_b)$, that is, $a_b(k)/\Delta$ is an approximation to a Gaussian density. Then we know that as $\Delta \to 0$ and $\nu \to \infty$, the discrete counts in (20) become equal to the continuous ones. Suppose further that we also want $\hat{a}_b(k)/\Delta = N(\tilde{y}_k, \hat{\mu}_b, \hat{\sigma}_b)$. Then, taking the derivatives of (20) with respect to $\hat{\mu}_b$ and $\hat{\sigma}_b$ and making them

[3] This is similar to the mismatch between acoustic probabilities and language model probabilities, which requires that the language model contribution to the log likelihood be multiplied by a certain factor. This factor is usually determined empirically. MMIE is again a good framework for determining this factor automatically.

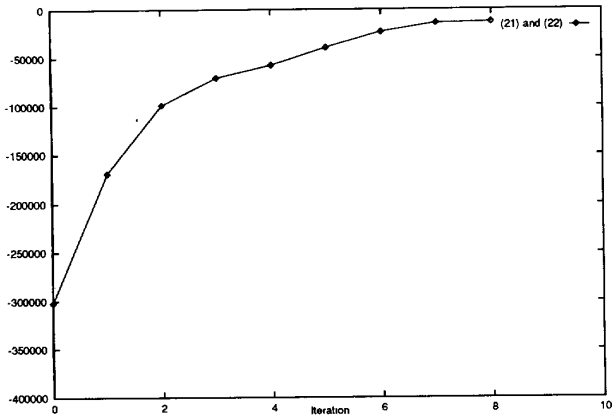


Fig. 4. Value of $\log_{1.001} R(\Theta)$ as a function of the iteration number.

equal to zero, we obtain

$$\hat{\mu}_b = \frac{\sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_l \psi_{\Theta,l}(t, \mathbf{y}^r) y_l^r + D\mu_b}{\sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_l \psi_{\Theta,l}(t, \mathbf{y}^r) + D} \tag{21}$$

$$\hat{\sigma}_b^2 = \frac{\sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_l \psi_{\Theta,l}(t, \mathbf{y}^r) y_l^{r\,2} + D(\sigma_b{}^2 + \mu_b{}^2)}{\sum_{r=1}^{N} \sum_{(t|b_t \equiv b)} \sum_l \psi_{\Theta,l}(t, \mathbf{y}^r) + D} - \hat{\mu}_b^2 \tag{22}$$

It can be shown that $D \to \infty$ as $\Delta \to 0$. This means that (21) and (22) cannot in practice be used within their proven convergence region. However, they can be interpreted as gradients, in the sense that they indicate a direction in which an infinitesimal step is guaranteed to increase the objective function.

The value for D used in our experiments with (21) and (22) is determined as follows. We compute the minimum value that will ensure a positive variance for all distributions and we set D to twice that value. Although somewhat arbitrary, our experiments showed that this performs quite well. Fig. 4 shows an example of convergence when (21) and (22) are used to reestimate the means and variances of a multiple-codebook, semicontinuous HMM (codebook probabilities are kept fixed). The application is the same as the one used for deriving the results in Fig. 2.

IV. Experimental Setting and Summary of Previous Results

The connected digit recognition experiments were performed using the adult speaker portion of the TIDIGITS connected digit corpus [7]. This large corpus contains utterances from a total of 326 speakers (111 men, 114 women and 101 children), coming from 21 geographical regions of the continental United States (approximately 5 men and 5 women and 5 children per region). Only the adult portion of the corpus is dialectically balanced. The corpus vocabulary is made of the digits '1' to '9', plus 'oh' and 'zero', for a total of 11 words. Each speaker contributes to the corpus with two repetitions of each digit in isolation and 55 digit strings, evenly distributed into lengths 2, 3, 4, 5, and 7. This makes a total of 77 digit strings, or 253 digits per speaker. Each string is stored in a separate signal file, with some silence (or background noise) preceding and following the speech signal. Approximately half the speakers have been assigned to the training set, the remaining half make up the testing set.

This corpus contains high quality sound and high signal to noise ratio (SNR). It was originally sampled at 20 kHz using a 16 bit A/D and a 10 kHz antialiasing filter. It has subsequently been downsampled by NIST to 10 kHz. The 10 kHz version was used in the first set of experiments and the 20 kHz version (from the CD-ROM release) in the second.

There are many dimensions along which recognition performances can be improved. The most important of them are model selection, model parameter estimation and learning strategies, and acoustic parameter choice. The effectiveness of improved learning algorithms and strategies is assessed by the experiments described in Section V.

As the vocabulary of the corpus is small and the performance of the baseline system used for comparison is already good, a very low error rate is achieved not only by improved learning methods but also by the use of more and better acoustic information and by the introduction of more detailed models. These aspects are experimentally investigated in Section VI.

Following other researchers [13]–[16], experiments were performed only on the adult portion of the corpus containing 225 speakers (111 men and 114 women), 112 of which (55 men, 57 women) are used for training and 113 (56 men, 57 women) for testing. This is the standard set used by most researchers, which makes result comparisons relatively meaningful. In our 10 kHz version of the corpus, out of a total of 17 325 signal files, 20 contained errors and could not be used. Of these, 8 were in the training set and 12 in the testing set. As a result, our training set contained 8616 digit strings (28 302 digits) and our test set contained 8689 digit strings (28 543 digits).

There are usually two types of recognition experiments performed with the TIDIGITS connected digit corpus. In the first type, *known-string-length* recognition, the number of digits in each digit string is assumed to be known a priori. In this case, the number of errors is computed by comparing in sequence the digits in the true and the recognized strings and counting the number of mismatches. Thus, all recognition errors are assumed to be substitution errors. In the second, more difficult type, *unknown-string-length* recognition, the string length is assumed to be unknown. This means that the true and the recognized strings do not necessarily contain the same number of digits. The number of errors is computed by first doing an optimal dynamic programming based alignment between the true and the recognized strings [17]. This alignment produces three types of errors: insertions, deletions and substitutions. In both cases, results are usually reported in terms of word and string error rates (or recognition rates).[4]

Only unknown length recognition experiments were performed. The word error rate is computed with the following

[4]Note that our scoring algorithm was compared to the NIST scoring algorithm, and they were found to give identical results.

TABLE I
PREVIOUSLY REPORTED RESULTS ON THE TIDIGITS CONNECTED DIGIT CORPUS

Year	Comments	Word	String
1985	Kopec & Bush (isolated)	2.0	-
1986	Bush & Kopec (unknown length)	-	3.5
1986	Bush & Kopec (known length)	-	2.2
1987	Bush & Kopec (complete corpus)	1.5	4.0
1988	Rabiner *et al.*	-	4.96
1989	Rabiner *et al.*	-	4.84
1989	Doddington	0.5	1.5
1991	Wilpon *et al.*	-	1.4

expression

$$\text{word error rate} = \frac{\text{insertions} + \text{deletions} + \text{substitutions}}{\text{total number of words}} \times 100\%$$

and the string error rate is computed with the following relation

$$\text{string error rate} = \frac{\text{number of strings with one or more errors}}{\text{total number of strings}} \times 100\%.$$

Experiments with the TIDIGITS corpus have been reported in the literature since the corpus was made available. In 1985, Kopec and Bush [18] reported a 2% error rate using only the isolated digits in the adult portion of the corpus. In 1986, Bush and Kopec [14] reported results on connected digit recognition experiments using about half of the adult portion of the corpus. Separating male and female talkers, they achieved 3.5% and 2.2% string error rate for unknown length and known length recognition, respectively.

In the following year, the same authors reported results obtained using the entire adult portion of the corpus. Using separate models for male and female talkers, they achieved a 4% string error rate (around 1.5% word error rate).

In 1988, Rabiner *et al.* [19] reported a 2.94% string error rate on the same task. This result was obtained using four models per digit and Gaussian mixture distributions. In the following year performance increased to 2.84% string error rate after experimenting with different clustering procedures.

Remarkable improvements were obtained by Doddington [15] and Wilpon *et al.* [20]. Doddington achieved a 1.5% string (0.5% word) error rate using *phonetically sensitive discriminants*. His system used an 18-element feature vector obtained from a 32-element feature vector via principal component analysis. He used separate male and female models in which each state corresponds, on average, to one frame of speech. For each state, a linear discriminant transformation matrix was computed using "in-class data" and "confusion data." The resulting complexity is about equivalent to a single full covariance density per state. Wilpon *et al.* obtained 1.4% string error rate using higher order spectral and energy features, and models using Gaussian mixture distributions with a large number of mixture components.

These previously reported results are summarized in Table I.

TABLE II
BASELINE SYSTEM PARAMETER SETS

Name	Description	Codebook Size
MCC	6 mel scaled FFT based cepstral coefficients	128
ΔMCC	The time derivatives of MCC	128
E and ΔE	The signal energy and its time derivative	32

TABLE III
COMPOSITION OF WORD MODELS

1	w-ax + n-tail	7	s + eh + v-7 + ax + n-tail
2	t + uw	8	ey + pau + t-8
3	th + r-iy	9	n-head + ay + n-tail
4	f + ow-r	oh	oh
5	f + ay + v-5	zero	z + iy-r-ow
6	s + ih-k + pau + k-s		

V. EXPERIMENTAL COMPARISON OF DIFFERENT LEARNING PROCEDURES

A baseline system was developed in order to quantify the advantages obtained by the use of the new training methods. The baseline system is a standard HMM-based system with discrete output distributions and one model per digit. It uses three codebooks, corresponding to the sets of acoustic parameters [12], [21], [22], shown in Table II.

In the experiments described in this Section, the 10 KHz version of the TIDIGITS corpus was used and analysis was performed without prior endpoint detection, using a frame rate of 10 ms, a preemphasis coefficient of 0.95 and a Hamming window of 256 samples. The codebooks were created using the entire training set of the corpus.

Experiments were performed only on strings of unknown length. This was done by applying the Viterbi algorithm on the loop of all digits model in parallel as shown in Fig. 1. The word models are built from a set of unit models, using the lexical representation described in Table III.

The units used in Table III were empirically chosen because they correspond to a phonemic representation with a reasonable selection of allophones taking into account significant context dependencies. They may be viewed as word-dependent units. Another design decision concerns the topology of the models and the tying of probability distributions. Model duration information can be taken into account by suitably crafting model topology.

Unit models are assembled with three basic components called: nucleus, head/tail, and silence. The topology of the three basic components is shown in Fig. 5. Within a basic component, all probability distributions are tied. The construction of a word model is performed by concatenation of the basic components. Table IV shows the number of basic components per model.

We used hand-segmented labeled speech utterances from 78 speakers of the training set and their corresponding labels to bootstrap the training procedure. Bootstrap consists of individually training each unit with four iterations of MLE training. It was found that bootstraping, by properly initializing

TABLE IV
MODEL COMPOSITION

Model	Head	Nucleus	Tail	Silence	Model	Head	Nucleus	Tail	Silence
w-ax	1	6	1		n-tail	1	3	1	
t	1	3	1		uw	1	8	1	
th	1	2	1		r-iy	1	5	1	
f	1	2	1		ow-r	1	11	1	
ay	1	8	1		v-5	1			
s	1	3	1		ih-k	1	3	1	
k-s	1	3	2		eh	1	4	1	
ax	1	1	1		v-7	1			
ey	1	7	1		t-8	1			
n-head	1	2	1		ow	1	8	1	
z	1	2	1		iy-r-ow	1	10	1	
pau				1	sil		1		1

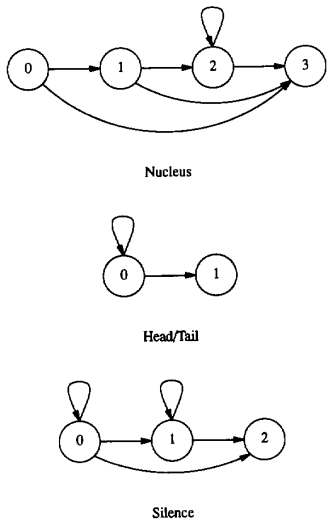


Fig. 5. Basic components of unit models.

the unit models, consistently produces improved recognition rates. However, it does not seem that a large amount of data is necessary for bootstraping since no significant performance variation was observed when only 14 (male-only) speakers were used instead of 78.

After bootstraping, three iterations of MLE training are performed on the entire training set. For each digit string, the corresponding model is built with optional silence at the beginning and end, and with an optional pause between each digit. The Baum-Welch algorithm [1] is applied to the generated model, using the observation sequence from the corresponding speech utterance.

Then, the *corrective MMIE training* algorithm introduced in Section III is applied. In each iteration, recognition is performed on the training set and MMIE training is applied *only* to the incorrectly recognized strings. The obtained parameters (Θ_{MMIE}) are smoothed with those from the previous iteration(Θ_{old}), as follows

$$\Theta_{\text{new}} = \alpha\Theta_{\text{old}} + (1 - \alpha)\Theta_{\text{MMIE}} \tag{23}$$

where α, the smoothing constant, should be a function of the number of errors in the training set. The idea of smoothing is that, since each training iteration is done using only a small fraction of the entire training set, it appears prudent to keep a memory of the models from the previous iteration. This is done because the assumption that correctly recognized strings in the training set will not affect reestimation is only approximately true. In practice, we start with $\alpha = 0.0$ for the first iteration, and increase it by 0.1 at each iteration either until it reaches 0.9 or until the number of errors in the training set becomes too small.

Finally, in order to see whether the initial codebook exponent values are important, the whole experiment is done once again, but this time using the last exponents obtained as initial values. This is what we call the second training phase. The initial models are the same as the ones used in the first phase. The only difference is that, during bootstraping and MLE training, the exponents used are different from 1.0. Note that, even though MLE training does not modify codebook exponents, it nonetheless takes advantage of "better" exponent values.

Training of semicontinuous HMM's is similar. To compute the initial tied mixtures components, all frames from the training set are assigned to the closest codeword in the codebook. Then, for each codeword, means and variances are computed using all frames assigned to it. During MLE training, the mixture component parameters are updated as proposed in [12]. During MMIE training, the mixture parameters are updated as described in Section III.

A. Results with Discrete HMM's

Table V reports the error rates and the detail on the number of word errors due to insertions, deletions and substitutions for the entire test set. The row labeled MLE refers to experiments with parameters estimated after three iterations of MLE training with exponents all equal to 1.0.

The row labeled MMIE refers to experiments with HMM's trained with the standard MLE procedure followed by nine iterations of corrective MMIE training (with the codebook exponents fixed at 1.0). The row labeled *MMIE + exponents* is similar except that the codebook exponents are now trained.

TABLE V
FIRST PHASE RESULTS: PERCENT RECOGNITION ERRORS (IN THE FIRST COLUMN FOR WORDS AND THE SECOND COLUMN FOR STRINGS) AND ABSOLUTE ERRORS ON THE TEST SET WITHOUT EXPONENT (FIRST TWO LINES) AND AFTER EXPONENT TRAINING DURING MMIE TRAINING

Training Method	Word	String	Ins	Del	Subs
MLE	1.36	3.90	50	112	225
MMIE	0.92	2.79	40	56	166
MMIE + exponents	0.85	2.58	38	42	163

TABLE VI
SECOND PHASE RESULTS: PERCENT RECOGNITION ERRORS ON THE TEST SET WITH TRAINING DONE USING, AS INITIAL EXPONENTS, THOSE OBTAINED AT THE END OF THE FIRST PASS

Training Method	Word	String	ins	del	subs
MLE	1.21	3.48	50	89	207
MMIE + exponents	0.75	2.23	36	38	140

TABLE VII
CODEBOOK EXPONENTS OBTAINED AFTER THE FIRST AND THE SECOND TRAINING PHASE

	Exponents		
Phase	$C_1 - C_6$	$\Delta C_1 - \Delta C_6$	$E\& \Delta E$
1	0.7072	1.0282	0.7577
2	0.4858	0.9719	0.6535

The exponents obtained at the end of nine training iterations are shown in Table VII. We can see that they are quite different from their initial value of 1.0. The exponents weigh the contribution of each codebook in the output probability. Their values may be considered good indicators of the relative usefulness of each parameter set. They may also compensate for differences in the dynamic range of probabilities from one codebook to another. Since they are not normalized, they also weigh the contribution of output probabilities with respect to that of transition probabilities.

Table VI reports the results obtained during the second pass of training. The difference with the first pass is that the initial exponents used (for MLE and MMIE training) are the ones obtained at the end of the first training pass. The exponents obtained at the end of the second training pass are shown in Table VII. It appears from these values that, at least for the digit task, discrimination is improved by emphasizing the dynamic acoustic parameters.

The advantages of corrective MMIE training are evident from Tables V and VI supporting the value of the theoretical considerations made in Section III. The errors on the training set were reduced to a few units after corrective MMIE training, showing that such a procedure has also performed a sort of *ad hoc* adaptation of the models to a number of special cases. Nevertheless, the results on the test set show that *corrective MMIE training*, while providing adaptation to some specific cases, has also produced a generalization better than the one achievable with MLE training (at least for the size of the vocabulary and the composition of the available training set).

TABLE VIII
FRAME-DEPENDENT CODEBOOK EXPONENTS OBTAINED AT THE END OF THE FIRST TRAINING PHASE

	Exponents (Phase 1)		
Category	$C_1 - C_6$	$\Delta C_1 - \Delta C_6$	$E\,\Delta E$
Sonorant/nasal	0.6679	1.0585	0.7903
Silence/noise	0.9822	0.9794	1.0432
Fricative/plosive	0.8559	1.0814	0.8277
Context-independent	0.7072	1.0282	0.7577

TABLE IX
PERCENT OF RECOGNITION ERRORS ON THE TEST SET OBTAINED USING MLE TRAINING WITH DIFFERENT INITIAL CODEBOOK EXPONENTS

Type of Experiment	Word	String
MLE	1.36	3.90
MLE + FI exponents	1.21	3.48
MLE + FD exponents	1.14	3.22

TABLE X
PERCENT OF RECOGNITION ERRORS ON THE TEST SET OBTAINED USING MMIE TRAINING WITH DIFFERENT INITIAL CODEBOOK EXPONENTS. EXCEPT IN THE FIRST ROW, EXPONENTS ARE MODIFIED DURING MMIE TRAINING

	Phase 1		Phase 2	
Type of Experiment	Word	String	Word	String
MMIE	0.92	2.79	-	-
MMIE + FI exponents	0.85	2.58	0.75	2.23
MMIE + FD exponents	0.78	2.36	0.73	2.16

In order to verify whether different codebook exponents should be used with different types of sounds, a recurrent neural network (RNN) of the type described in [21] has been used to label the speech frames as one of three categories: sonorant/nasal, silence/noise, and fricative/plosive. Then, for each frame, the exponents used are made dependent on the category in which the frame is classified by the RNN (see [12]).

Table VIII shows the frame-dependent (FD) codebook exponents obtained at the end of the first phase, for each of the three categories. We can see that there are noticeable differences between the categories. In particular, the silence/noise category puts almost equal weights on all parameter sets, which is quite different from the frame-independent (FI) codebook exponents obtained in the previous experiment.

Table IX compares the recognition results obtained on the test set when MLE training is used with different exponent values. Table X compares the recognition results obtained after MMIE training, using the different types of codebook exponents.

As a final experiment with discrete HMM's, we used an RNN with four outputs describing the following phonetic classes:

- silence and noise
- fricatives and plosives
- nasals
- vowels, liquids, and glides.

The four outputs of this RNN and a set of energy contour descriptors [21] were used as additional observations for the HMM's. With such an addition, a word error rate of 0.65% and a string error rate of 1.98% was obtained with discrete HMM's. The use of RNN's eliminated certain errors and introduced new ones in minor quantities. The net benefit introduced by this RNN tends to vanish if additional acoustic features are introduced as parameters or better HMM's (like SCHMM's) are used. For these new models, better RNN's should be designed in order to expect a benefit from their use. As these RNN's are not available yet, their use was not extended to SCHMM's.

B. Results with Semicontinuous HMM's

All SCHMM experiments reported here assume diagonal covariance Gaussian densities. Since the cepstral coefficients are relatively uncorrelated [23], this assumption seems reasonable and reduces both the number of parameters to estimate and the computation time. SCHMM distributions are computed using the following relation

$$b(\underline{y}) = \prod_{c=1}^{NC} \sum_{k=0}^{K_c-1} P_c(\underline{y}_c|k)p_c(k|b) \qquad (24)$$

where NC is the number of codebooks and K_c is the number of mixture components in the cth codebook. Unless specified otherwise, however, all semicontinuous experiments (training and recognition) were performed by considering only the three most probable components (densities) in the mixture. Thus, referring to equation (24), we assume $P_c(\underline{y}_c|k) = 0$ if k is not one of the three most probable mixture components. This may affect performance but it substantially reduces the execution time.

Semicontinuous HMM's have been trained following the steps used in the discrete case. That is, use initial models with uniform distributions (mixture weights) and initial mixtures components as described above, and perform bootstraping followed by MLE training (using the same number of iterations). This has the advantage of jointly optimizing (with MLE) the codebook probabilities and the mixture parameters.

In addition to experiments performed using, as before, one model per unit, MMIE training with multiple models per unit was also considered. We chose to use male and female models, thus doubling the total number of models. Note, however, that the same tied mixture components are used for both male and female models. Since the information about the speaker sex is available in the corpus, bootstraping and MLE training used this information.

Table XI shows how the recognition rate on the test set changes as the number of MLE training iterations increases. This experiment seemed relevant since our other experiences with semicontinuous HMM's (in particular for wordspotting applications) tend to show that SCHMM's require more MLE training iterations than do discrete HMM's. This may be explained by the fact that the continuous mixture components are shared by all distributions. In this case, however, the performance flattens out rather quickly and very little improvement is observed after the fifth iteration.

TABLE XI
ERROR RATE ON THE TEST SET WITH SCHMM'S AFTER ADDITIONAL MLE TRAINING ITERATIONS

Iteration	Word	String	Ins	Del	Sub
3	0.75	2.36	39	77	99
4	0.74	2.27	39	75	96
5	0.70	2.18	38	72	91
6	0.72	2.21	39	72	94
7	0.73	2.23	40	69	99
8	0.71	2.16	38	69	95
9	0.72	2.19	38	67	100

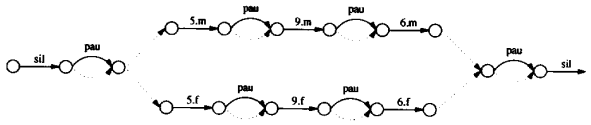


Fig. 6. Model $\mathbf{m_w}$ for $\mathbf{w}$ = 5-9-6.

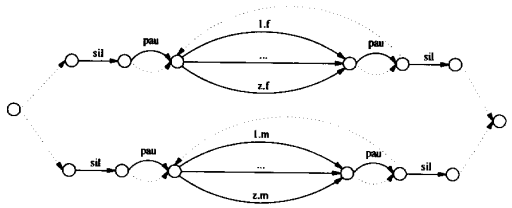


Fig. 7. Model $\mathbf{w_{gen}}$ used for training with separate male and female models.

Earlier, we presented the results of a MMIE convergence experiment with the reestimation formulas for continuous densities proposed in this paper. We now look at whether their use can translate into better recognizers. This is also an opportunity to further compare the performance of semicontinuous and discrete HMM's.

It is not immediately clear, however, how MMIE training should be done when several models per unit are used, especially when, as is our case, the clusters (male and female) are determined *a priori* and the information about each speaker's cluster is available from the corpus. The question is whether or not we should enforce the sex of speakers in the training process. If we did, it would, in effect, add gender recognition to the problem of digit recognition. Since this is not useful for our purpose, we decided not to use the information about sex during MMIE training. Suppose one of the digit training sequences contains w=5-9-6. Then the "good" model used for training is the one illustrated in Fig. 6. In all cases, the model $\mathbf{m_{gen}}$ will be the one illustrated in Fig. 7.

Table XII summarizes the experimental results with SCHMM's. Note that, in order to get the MMIE results, only the second training phase was used in this case. The initial exponents used for this second phase were those obtained with one model per unit. Note also that the MMIE results are given after the standard nine iterations and the number of MLE iterations is given, both for the MLE results and for the MMIE results (in which case it indicates the number of MLE iterations performed before MMIE training).

TABLE XII
ERROR RATE ON THE TEST FOR DIFFERENT EXPERIMENTS WITH SCHMM'S

Experiment	# models	MLE Iterations	Word	String	Ins	Del	Sub
MLE	1	3	1.22	3.51	47	99	202
MMIE + exponents	1	3	0.66	2.01	47	30	112
MLE	2	3	0.75	2.36	39	77	99
MLE	2	9	0.72	2.19	38	67	100
MMIE + exponents	2	3	0.49	1.51	45	16	78

TABLE XIII
SUMMARY OF THE ERROR RATES OBTAINED IN THE FIRST SET OF EXPERIMENTS

Experiment	Word	String
Discrete MLE	1.36	3.90
Discrete MMIE	0.92	2.79
Discrete MMIE + FI exponents	0.75	2.23
Discrete MMIE + FD exponents	0.73	2.16
SCHMM MLE	1.22	3.51
SCHMM MMIE + exponents	0.66	2.01
SCHMM 2 models MLE	0.72	2.19
SCHMM 2 models MMIE + exponents	0.49	1.51

C. Summary of Results

Table XIII summarizes the results obtained in the different experiments described in this section. These results exhibit a clear superiority of semicontinuous HMM's over discrete HMM's and, in both cases, the usefulness of MMIE for training HMM's.

VI. EXPERIMENTS WITH IMPROVED MODELS AND NEW ACOUSTIC PARAMETERS

A second set of experiments was performed with improved models and new and more informative acoustic features. For this purpose, the 20 kHz version of the TIDIGITS corpus was used with higher order spectral and energy features whose introduction is motivated by recent results demonstrating their usefulness for speech recognition [20]. The 20 kHz version of the corpus, now available on CD-ROM, did not contain any corrupted signal files, as did the previous versions. The higher sampling frequency allowed the addition of filters to the mel-scaled filter bank.

Word HMM's are described by a sequence of unit symbols. Each unit symbol corresponds to a component HMM. The choice of basic units is somehow arbitrary and, for this particular task, a set of word-dependent units was chosen. The structure of these units is a standard 4-state, left-right HMM in which output probability distributions, associated with the transitions, are tied to the departure state of the transition. The topology is shown in Fig. 8. The number on each arc indicates the tying of the output distribution.

The system once again uses both a silence model, for background noise, and a pause model, for short between-word pauses (which are often contaminated by breath noise). The topology of the "silence" and "pause" model is shown in Fig. 9.

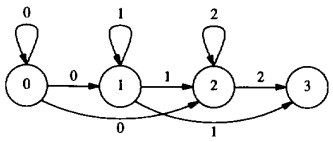


Fig. 8. Unit model.

TABLE XIV
NUMBER OF UNIT MODELS USED FOR EACH WORD IN THE VOCABULARY

Digit	#	Digit	#
1	6	2	6
3	6	4	7
5	7	6	8
7	9	8	5
9	6	oh	5
zero	9		

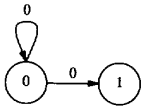


Fig. 9. Silence and pause models.

TABLE XV
AUGMENTED PARAMETER SET

Name	Description	Codebook Size
MCC:	12 mel scaled FFT based cepstral coefficients	256
ΔMCC:	The time derivatives of MCC	256
ΔΔMCC:	The time derivatives of ΔMCC	256
E:	The signal energy	32
ΔE:	The time derivative of the signal energy E	32
ΔΔE:	The time derivative of the ΔE	32

Six codebooks were now used with the new baseline system, corresponding to the parameter sets in Table XV.

Analysis was performed without prior endpoint detection, using a frame rate of 10 ms, preemphasis of 0.95 and a Hamming window of 512 sample points. The codebooks were created using the entire training set of the corpus.

The addition of new features and the use of better topologies resulted in an enhancement of the performances in all the experiments. As the error rate gets small, the advantages of using MMIE, while still remaining evident, decreases.

A. Results

Training was done as before by bootstrapping, followed by standard MLE training, followed by corrective MMIE training with codebook exponents training. Results appear in Tables XV and XVI. *BOOTSTRAP* refers to the error rates obtained on the test set after bootstrapping, *MLE* refers to those after six iterations of MLE training, while *MMIE* refers to those after four iterations of *corrective MMIE training*.

TABLE XVI
RECOGNITION ERRORS WITH DISCRETE HMM's, SIX PARAMETER SETS, ONE MODEL PER WORD AND 20 kHz SAMPLING

	One Model		Two Models	
	Word	String	Word	String
BOOTSTRAP	0.66	1.92	0.45	1.37
MLE 6	0.61	1.77	0.43	1.29
MMIE 4	0.50	1.48	0.41	1.21

TABLE XVII
RECOGNITION ERRORS WITH SEMICONTINUOUS HMM's, SIX PARAMETER SETS, TWO MODELS PER WORD AND 20 kHz SAMPLING

	One Model		Two Models	
	Word	String	Word	String
BOOTSTRAP	0.48	1.49	0.40	1.22
MLE 6	0.40	1.20	0.34	1.01
MMIE 4	0.35	1.00	0.29	0.89

B. Discussion of Errors

An analysis of the errors make evident some typical problems. Most deletions and insertions involve the digit 'oh' which is also often confused with '4' and '2'. This indicates a difficulty of the system in distinguishing the weak fricative 'f' and the 't' of '2' from the background noise.

In some cases, '5', 'oh', and '8' become '9'. These errors, as well as less frequent errors involving '9', show a weakness in the system in recognizing nasality.

Other frequent errors involve the word '8', frequently deleted or inserted. This fact, as well as confusions between '4' and '5', '2', and '3' show that the system is weak in characterizing certain types of coarticulations involving complex dynamics of the vocal tract.

Perhaps some of the above mentioned problems could be solved by increasing the size and the variety of the training set. Some benefits could also be achieved by speaker adaptation, when possible, because in many cases a type of error appears only for one speaker of the test set. Other possibilities worth investigating in the future are the use of new input parameters, extracted, for example, by connectionist models, or the use of specialized post processors in a system that computes new probabilities and updates the order of the N-best candidates proposed by the actual system.

VII. CONCLUSION

A number of conclusions can be derived from the experiments described in this paper. First, at least for small vocabularies, using MMIE following MLE can result in significantly improved recognition rates, compared to MLE alone.

Second, suitably chosen multiple parameter sets that can be used under the assumption of statistical independence give better performances than a single set especially if probabilities of each set are weighted.

Third, SCHMM's trained with the algorithm proposed in this paper give systematically better results than discrete HMM's. Nevertheless, the results of discrete HMM's are good enough to justify the use of these models in a practical system because the time of recognition with these models is much lower than the complexity of the other models considered in this paper.

We have introduced an efficient new training algorithm, "corrective MMIE training," which has allowed us to obtain these improvements with a small number of iterations, each of which is usually faster than a standard MLE training iteration. This algorithm is the result of a modification that we introduced into a reestimation formula for discrete distributions proposed by Gopalakrishnan *et al.*, and of the idea of only using errors in the training set for reestimating the HMM parameters. Taken separately, none of these ideas would have performed very well; however, taken together they led to systematically fast convergence in practice.

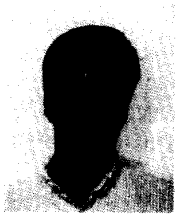
ACKNOWLEDGMENT

The authors wish to thank L. Vroomen (CRIM) for his support in software development and G. Doddington for useful discussions and suggestions. They also wish to thank the anonymous reviewers for their careful review of the paper.

REFERENCES

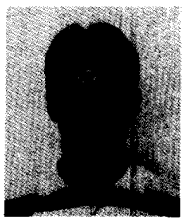
[1] L. E. Baum, "An inequality with applications to statistical estimation for probabilisitic functions of Markov processes," *Inequalities*, pp. 1–8, 1972.
[2] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proc. ICASSP-86* (Tokyo, Japan), 1986, pp. 49–52.
[3] A. Nadas, D. Nahamoo, and M. A. Picheny, "On a model-robust training method for speech recognition," *IEEE ASSP Mag.*, pp. 1432–1436, Sept. 1988.
[4] P. F. Brown, "The acoustic-modeling problem in automatic speech recognition," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 1987.
[5] Y. L. Chow, "Maximum mutual information estimation of HMM parameters for continuous speech recognition using the N-best algorithm," in *Proc. ICASSP-90*, 1990, pp. 701–704.
[6] B. Merialdo, "Phonetic recognition using hidden Markov models and maximum mutual information training," in *Proc. ICASSP-88*, 1988.
[7] R. G. Leonard, "A database for speaker-independent digit recognition," in *Proc. ICASSP-84*, 1984.
[8] A. Nádas, "A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood," *IEEE Trans. Acoust., Speech, Signal Processing*, pp. 814–817, Aug. 1983.
[9] R. Schwartz and Y. L. Chow, "The N-best algorithm: an efficient and exact procedure for finding the N most likely sentence hypotheses," in *Proc. ICASSP-90* (Albuquerque, NM), Apr. 1990.
[10] P. S. Gopalakrishnan, D. Kanevsky, A. Nádas, and D. Nahamoo, "An inequality for rational functions with applications to some statistical estimation problems," in *IEEE Trans. Inform. Theory*, Jan. 1991.
[11] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology," *Bulletin of AMS*, pp. 360–363, 1967.
[12] Y. Normandin, "Hidden Markov models, maximum mutual information estimation, and the speech recognition problem," Ph.D. dissertation, McGill Univ., Montreal, Quebec, 1991.
[13] E. L. Bocchieri and G. R. Doddington, "Frame-specific statistical features for speaker-independent speech recognition," in *IEEE Trans. Acoust., Speech, Signal Processing*, Aug. 1986.
[14] M. A. Bush and G. E. Kopec, "Network-based connected digit recognition using explicit acoustic-phonetic modeling," in *Proc. ICASSP-86* (Tokyo, Japan), 1986, pp. 1097–1100.
[15] G. R. Doddington, "Phonetically sensitive discriminants for improved speech recognition," in *Proc. ICASSP-89* (Glasgow, Scotland), 1989.

[16] L. R. Rabiner, J. G. Wilpon, and F.K. Soong, "High performance connected digit recognition using hidden Markov models," *IEEE ASSP Mag.*, Aug. 1989.
[17] J. Picone, K. M. Goudie-Marshall, G. R. Doddington, and W. Fisher, "Automatic text alignment for speech system evaluation," *IEEE Trans. Acoust., Speech, Signal Processing*, Aug. 1986.
[18] G. E. Kopec and M. A. Bush, "Network-based isolated digit recognition using vector quantization," *IEEE ASSP Mag.*, Aug. 1985.
[19] L. R. Rabiner, J. G. Wilpon, and F. K. Soong, "High performance connected digit recognition using hidden Markov models," in *Proc. ICASSP-88* (New York), Apr. 1988.
[20] J. G. Wilpon, C. H. Lee, and L. R. Rabiner, "Improvements in connected digit recognition using higher order spectral and energy features," in *Proc. ICASSP-91* (Toronto, Ontario), 1991, pp. 349–352.
[21] R. Cardin, Y. Normandin, and R. De Mori, "High performance connected digit recognition using maximum mutual information estimation," in *Proc. ICASSP-91* (Toronto, Ontario), 1991, pp. 533–536.
[22] Y. Normandin and S. D. Morgera, "An improved MMIE training algorithm for speaker-independent small vocabulary, continuous speech recognition," in *Proc. ICASSP-91* (Toronto, Ontario), 1991.
[23] M. J. Hunt and C. Lefebvre, "Distance measures for speech recognition," *Aeronaut. Note NAE-AN-57*, National Research Council Canada, Mar. 1989.



Yves Normandin graduated in 1985 from Laval University, Quebec, Canada, in engineering physics. He received the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1986, and the Ph.D. degree in electrical engineering from McGill University, Montréal, Quebec, in 1991.

Since 1986, he has been with CRIM where he has helped create the Speech Understanding Group, which he heads. He is also Adjunct Professor in McGill University's Electrical Engineering Department.



Regis Cardin graduated in 1984 from Concordia University, Montréal, Canada, in computer science. In 1986 he received the M.S. in computer science, also from Concordia University. He has received the Ph.D. degree in computer science at McGill University, Montréal, Quebec.

Since 1990, he has been with CRIM in the Speech Understanding Group.

Renato De Mori (M'83–SM'89) was born in Milan, Italy in 1941 and received the doctorate degree in electronic engineering from Politecnico di Torino, Torino, Italy, in 1967.

He became Full Professor in Italy in 1975. Since 1986, he has been Professor and the Director of the School of Computer Science at McGill University, Montreal, Quebec, Canada. Since 1987, he has been Vice-President of the Centre de Recherche en Informatique de Montré al, a research center involving seven universities and more than 40 industries. In 1991, he became Associate of the Canadian Institute for Advanced Research and Project Leader of the Institute for Robotics and Intelligent Systems, a Canadian Center of Excellence. He is the author of many publications in the areas of computer systems, pattern recognition, artificial intelligence, and connectionist models. He has been Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE and is now on the board of the following international journals: *Signal Processing, Speech Communication, Pattern Recognition Letters, Computer Speech and Language and Computational Intelligence*. His research interests are in stochastic parsing techniques, connectionist models, and reverse engineering.

Dr. De Mori has been member of various committees in Canada, Europe, and the United States.