Comparative study of the Baum-Welch and Viterbi training algorithms applied to read and spontaneous speech recognition

Luis Javier Rodríguez and Inés Torres *

Departamento de Electricidad y Electrónica (Facultad de Ciencias)
Universidad del País Vasco
Apartado 644, 48080 Bilbao, Spain
luisja@we.lc.ehu.es

Abstract. In this paper we compare the performance of acoustic HMMs obtained through Viterbi training with that of acoustic HMMs obtained through the Baum-Welch algorithm. We present recognition results for discrete and continuous HMMs, for read and spontaneous speech databases, acquired at 8 and 16 kHz. We also present results for a combination of Viterbi and Baum-Welch training, intended as a trade-off solution. Though Viterbi training yields a good performance in most cases, sometimes it leads to suboptimal models, specially when using discrete HMMs to model spontaneous speech. In these cases, Baum-Welch shows more robust than both Viterbi training and the combined approach, compensating for its high computational cost. The proposed combination of Viterbi and Baum-Welch only outperforms Viterbi training in the case of read speech at 8 kHz. Finally, when using continuous HMMs, Viterbi training reveals as good as Baum-Welch at a much lower cost.

1 Introduction

Most speech recognition systems use Hidden Markov Models (HMM) to represent the acoustic content of phone-like units and words. Though other criteria may be applied, the reestimation of HMM parameters is commonly done according to the the *Maximum Likelihood Estimation* (MLE) criterion, i.e. maximizing the probability of the training samples with regard to the model. This is done by applying the *Expectation-Maximization* (EM) algorithm [1], which relies on maximizing the log-likelihood from incomplete data, by iteratively maximizing the expectation of log-likelihood from complete data. As shown in [2], this leads to the *Baum-Welch* reestimation formulas.

The MLE criterion can be approximated by maximizing the probability of the best HMM state sequence for each training sample, given the model, which is known as *segmental k-means* [3] or *Viterbi training*. Viterbi training involves

* This work was partially supported by the Basque Country University, under a generic grant for research groups, and the Spanish MCYT, under projects TIC2001-2812-C05-03 and TIC2002-04103-C03-02.

much less computational effort than Baum-Welch, still providing the same —or slightly worse— performance, so it is a common choice among designers of speech recognition systems.

However, the Baum-Welch algorithm shows very interesting properties: (1) in the case of discrete HMMs it does not need *any* model initialization, but just non-zero random values verifying the stochastic constraints; (2) in the case of continuous HMMs, a suitable initialization can be done with the output distribution parameters of discrete HMMs, on the one hand, and the means and variances of acoustic prototypes obtained by vector quantization, on the other —though other strategies have been successfully applied [4]; and (3) it exhaustively uses all the available data to produce robust and optimal estimates. With regard to Viterbi training, (1) it is shown that even in the case of discrete HMMs, it requires some reasonable initialization, either by using the models obtained for other databases, or by training initial models on a hand-labelled subset of the training database; and (2) it makes a limited use of the training data, since only observations inside the segments corresponding to a given HMM state are used to reestimate the parameters of that state, resulting in sharper but less robust models. It will depend on the the amount of available data whether or not Viterbi training produces robust enough estimates. Though the segmentation implicitly done by Viterbi training will not exactly match the right one —i.e. that produced by an expert— they overlap to a great extent, as shown in a previous work [5]. When the amount of training data is large enough, segmentation errors will cancel each other, and the right observations —i.e. the relevant features that identify an HMM state— will stand out.

The rest of the paper is organized as follows: Sections 2 and 3 briefly review the Baum-Welch and Viterbi training algorithms; Section 4 presents a combination of Viterbi segmentation and Baum-Welch reestimation, intended as a trade-off solution; Section 5 compares the amount of training data used to estimate HMM parameters for the three training algorithms; Section 6 defines a measure of segmentation quality based on hand-labelled segmentations generated by experts; Section 7 describes the experimental framework, and presents and discusses phonetic recognition results for read and spontaneous speech databases; finally, conclusions are given in Section 8.

2 The Baum-Welch algorithm: single vs. embedded model reestimation

The Baum-Welch algorithm is based on the computation of two functions, known as *Forward* and *Backward* probabilities, $\alpha(i,t)$ and $\beta(i,t)$, for each state $i \in [1,N]$ of an HMM and each frame $t \in [1,T]$ of an observation sequence $O = O_1, O_2, \ldots, O_T$. Computing these functions yields a complexity of order $O(N^2T)$. Once computed, *Forward* and *Backward* probabilities are used to weight the contributions of each observation O_t to the HMM parameters. Reestimation formulas can be found in [2]. Note that *each* observation O_t contributes to the reestimation of *all* the HMM parameters.

If L observation sequences $O^{(l)} = O_1^{(l)}, O_2^{(l)}, \ldots, O_{T_l}^{(l)}$, with $l = 1, 2, \ldots, L$, are explicitly available for an HMM, the resulting procedure is known as *single model Baum-Welch* reestimation [6]. This procedure is typically applied for training a word HMM starting from speech samples of that word, or for initializing phone HMMs starting from explicit hand-labelled phone segments. For each observation sequence $O^{(l)}$, the *Forward* and *Backward* probabilities must be computed, and then various contributions and norms accumulated, which yields a computational complexity of order $O(3N^2T_l + NT_l + NT_lRC)$ for discrete HMMs, and $O(3N^2T_l + NT_l + NT_lMD)$ for continuous HMMs, where R is the number of acoustic streams or representations, C the number of symbols in the discrete output distributions, M the number of gaussian components in the mixtures used to represent the output distributions in continuous HMMs, and $D = \sum_{i=1}^{R} d_i$ —with d_i the dimension of the acoustic stream i— the total number of components of acoustic vectors in the continuous case. In the most common configurations, the last term of the summation is dominant. Summing for all the training sequences and all the phone HMMs, we get complexities of order $O(N\mathcal{T}RC)$ and $O(N\mathcal{T}MD)$, respectively, where $\mathcal{T}$ is the length of the training database.

However, usually only a small fraction of the speech databases is hand-labelled, so phone HMMs must be jointly trained starting from phonetic transcriptions of speech utterances. This is known as *embedded model Baum-Welch* reestimation [6]. For each training sequence $O^{(l)}$ a large HMM $\Lambda_{train}(O^{(l)})$ is built by concatenating the phone HMMs corresponding to the transcription of $O^{(l)}$. If no phone skips are allowed in training, the possible state transitions reduce to those ocurring inside a phone HMM or between two consecutive phone HMMs. This results in that both the *Forward* and *Backward* procedures yield complexities of order $O(N^2T_lF_l)$, where F_l is the length of the phonetic transcription of $O^{(l)}$. Not only the *Forward* and *Backward* procedures, but also their contributions to the HMM parameters must be computed, resulting a complexity of order $O(5NT_lF_l + 3N^2T_lF_l + NT_lF_lRC)$ in the discrete case, and $O(5NT_lF_l + 3N^2T_lF_l + NT_lF_lMD)$ in the continuous case. Again, the last term is dominant, so these complexities can be approximated by $O(NT_lF_lRC)$ and $O(NT_lF_lMD)$, respectively. Defining $\mathcal{F} = \frac{1}{\mathcal{T}} \sum_{l=1}^{L} T_lF_l$, and summing for all the training utterances, we get complexities of order $O(N\mathcal{T}\mathcal{F}RC)$ and $O(N\mathcal{T}\mathcal{F}MD)$, respectively. So the *embedded model Baum-Welch* reestimation is approximately $\mathcal{F}$ times more expensive than the *single model Baum-Welch* reestimation.

3 The Viterbi training algorithm

The Viterbi algorithm [7] can be applied to get the most likely state sequence $\hat{S}^{(l)}$ in the training sequence HMM $\Lambda_{train}(O^{(l)})$. This is sometimes called *forced alignment*, and takes a computational complexity of order $O((NF_l)^2T_l)$. Viterbi reestimation is based on maximizing the likelihood of $\hat{S}^{(l)}$, given the observation sequence $O^{(l)}$ and the model $\Lambda_{train}(O^{(l)})$. It uses the most likely state sequence

$\hat{S}^{(l)}$ to estimate the HMM parameters, so each observation $O_t^{(l)}$ *only* contributes to reestimate the parameters of the most likely state at time t, $s_t^{(l)}$. Reestimation formulas can be found in [3]. Without phone skips, the possible state transitions in $\Lambda_{train}(O^{(l)})$ reduce to those ocurring inside a phone HMM or between two consecutive phone HMMs. So the computational complexity of the algorithm reduces to $O(N^2 F_l T_l)$. On the other hand, for each observation sequence, contributions to the estimation of HMM parameters must be computed, which yields a complexity of order $O(T_l RC)$ in the discrete case, and $O(T_l MD)$ in the continuous case. Summing for all the training utterances, we get complexities of order $O(N^2 \mathcal{TF} + \mathcal{T}RC)$ and $O(N^2 \mathcal{TF} + \mathcal{T}MD)$, respectively. In practice, these complexities are between one and two orders of magnitude lower than those of the *embedded model Baum-Welch* algorithm.

The sharpness of Viterbi estimates becomes a problem in the case of discrete HMMs, since some symbols not seen in training for a particular output distribution, may appear in an independent test corpus, thus leading to zero probability and breaking the search. So a kind of smoothing must be applied to the output distribution parameters. The simplest technique —which we apply in our implementation— consists of changing only values under a certain threshold τ, by assigning them the value τ and renormalizing the distribution to verify the stochastic constraints. Threshold smoothing only guarantees that the search will not crash, and more sophisticated techniques can be found in the literature. However, such techniques make assumptions —which might not be true— about the underlying distributions. On the other hand, the *embedded model* Baum-Welch reestimation provide smooth and robust parameters in a more natural way —at the expense of higher computational costs.

4 Combining Viterbi segmentation and single model Baum-Welch reestimation

We propose the following methodology:

1. For each training sequence $O^{(l)}$, the Viterbi algorithm is applied to find the most likely state sequence $\hat{S}^{(l)}$ in the training sequence HMM $\Lambda_{train}(O^{(l)})$.
2. The single model Baum-Welch reestimation formulas are used to update the parameters of each phone HMM, starting from the phone segments obtained after step (1) over $\Omega = \{O^{(l)} | l = 1, 2, \ldots, L\}$.
3. Steps (1) and (2) are repeated until convergence.

This is a sort of tradeoff between the Baum-Welch and Viterbi training algorithms, the most likely phone segmentation —corresponding to the most likely state sequence— playing the role of a hand-labelled segmentation in *single model Baum-Welch* reestimation. On the one hand, this algorithm requires less computational effort than the *embedded model Baum-Welch* algorithm: summing for all the training utterances, complexity is of order $O(N^2 \mathcal{TF} + N\mathcal{T}RC)$ in the discrete case, and $O(N^2 \mathcal{TF} + N\mathcal{T}MD)$ in the continuous case, which is slightly

higher than that of Viterbi reestimation. On the other hand, the resulting estimates are expected to be more robust than Viterbi estimates, since all the state sequences inside phone segments —instead of just the most likely state sequence— are considered.

5 Counting effective training data

The Baum-Welch and Viterbi training algorithms differ in the use of the data. Whereas Viterbi uses each observation vector O_t to reestimate the parameters of just *one specific* HMM state, Baum-Welch uses it to reestimate the parameters of *all* the HMM states. In other words, Baum-Welch estimates are obtained from much more data than Viterbi estimates. To evaluate these differences, we next count the number of observation vectors *effectively* used to estimate HMM parameters for the three training algorithms.

In the case of *embedded model* Baum-Welch, assuming no phone skips in the training model $\Lambda_{train}(O^{(l)})$ of an observation sequence $O^{(l)}$, the function $\alpha^{(l)}(i,t)$ is zero for the first i/N frames, and in the same way the function $\beta^{(l)}(i,t)$ is zero for the last $F_l - i/N - 1$ frames. So for each given state i of $\Lambda_{train}(O^{(l)})$, there are $F_l - 1$ frames for which the corresponding contributions to the HMM parameters are zero. Summing for all the states of phone HMMs in $\Lambda_{train}(O^{(l)})$ and for all the training utterances, we find that the number of *effective* training observations is $\sum_{l=1}^{L}\sum_{f=1}^{F_l}\sum_{e=1}^{N}(T_l - F_l + 1) = N\sum_{l=1}^{L}[T_l F_l - F_l^2 + F_l] = N\mathcal{T}\mathcal{F} - NL(\bar{F^2} - \bar{F})$, where $\bar{F^2} = \frac{1}{L}\sum_{l=1}^{L} F_l^2$ and $\bar{F} = \frac{1}{L}\sum_{l=1}^{L} F_l$.

In the case of Viterbi training, each observation sequence $O^{(l)}$ is divided into F_l segments, and each observation vector in those segments is assigned to a specific state in $\Lambda_{train}(O^{(l)})$. Let $n^{(l)}(f,i)$ be the number of observation vectors assigned to state i of the phone HMM f. Then, summing for all the states of phone HMMs in $\Lambda_{train}(O^{(l)})$ and for all the training utterances, the number of *effective* training vectors is $\sum_{l=1}^{L}\sum_{f=1}^{F_l}\sum_{i=1}^{N} n^{(l)}(f,i) = \sum_{l=1}^{L} T_l = \mathcal{T}$.

In the case of combined Viterbi segmentation + *single model* Baum-Welch reestimation, segmentation is done only at the phone level, and the parameters of each HMM state are trained with *all* the observation vectors assigned to the corresponding phone. Let $n^{(l)}(f)$ be the number of observation vectors assigned to the phone HMM f in $\Lambda_{train}(O^{(l)})$. Then, the number of *effective* training vectors for the entire training database is $\sum_{l=1}^{L}\sum_{f=1}^{F_l}\sum_{e=1}^{N} n^{(l)}(f) = N\sum_{l=1}^{L}\sum_{f=1}^{F_l} n^{(l)}(f) = N\sum_{l=1}^{L} T_l = N\mathcal{T}$.

This means that the *embedded model* Baum-Welch algorithm uses $\mathcal{F} - \frac{L}{\mathcal{T}}(\bar{F^2} - \bar{F})$ times more data than the combined approach, which, on the other hand, uses N times more data than Viterbi training.

6 Measuring segmentation quality

Viterbi training depends on the quality of the segmentation implicitly associated to the most likely state sequence, so we set out to evaluate automatic segmenta-

tions. A subset of the training utterances may be hand labelled and segmented. Then, hand-labelled segments may be compared with those automatically produced by forced alignment —based on a given set of HMMs. Let n_r be the number of frames assigned by forced alignment to the same phone than experts, and n_w the number of frames assigned to the *wrong* phones. Then $S = \frac{n_r}{n_r+n_w} * 100$ is the percentage of frames correctly classified. If the HMMs used in forced alignment were obtained through Viterbi training, S can be interpreted as the amount of acoustic information *correctly* used to estimate the parameters of such models.

7 Phonetic recognition experiments

Experiments were carried out over four different databases, whose main features are shown in Table 1.

Table 1. Speech databases used in phonetic recognition experiments.

	SENGLAR16	**SENGLAR08**	**INFOTREN**	**CORLEC-EHU-1**
Sampling rate (kHz)	16	8	8	16
Speech modality	read	read	spontaneous	spontaneous
Recording conditions	microphone laboratory	telephone simulated laboratory	telephone office	analog tape all environments
Other design issues	phonetically balanced	phonetically balanced	task specific	generic, noisy
Speakers (training/test)	109/37	109/29	63/12	79/37 73/43 80/36
Utterances (training/test)	1529/700	1529/493	1349/308	1414/723 1433/704 1427/710
Frames (training/test)	469620/244026	469626/179762	703719/182722	1357783/681601 1355586/683698 1365399/673985
Phones (training/test)	60399/32034	60399/23607	62729/13683	189108/90596 187331/92373 182969/96735
$\mathcal{F}$	47.93	47.93	82.88	323.25 338.25 335.06
$\mathcal{F} - \frac{L}{T}(\bar{F}^2 - \bar{F})$	41.74	41.74	75.18	278.98 292.23 290.75

The mel-scale cepstral coefficients (MFCC) and energy (E) —computed in frames of 25 milliseconds, taken each 10 milliseconds— were used as acoustic features. The first and second derivatives of the MFCCs and the first derivatives of E were also computed. Four acoustic streams were defined ($R = 4$): MFCC, ΔMFCC, Δ^2MFCC and (E,ΔE). In the discrete case, vector quantization was applied to get four codebooks, each one consisting of 256 centroids ($C = 256$) minimizing the distortion in coding the training data. The set of sublexical units consisted of 23 context-independent phones and two auxiliary units: *silence* and *filler*, this latter used only with spontaneous speech. Each sublexical unit was

represented with a left-right HMM consisting of three states ($N = 3$) with self-loops but no skips, the first state being initial and the last one final. The output probability at each time t was computed as the non-weighted product of the probabilities obtained for the four acoustic streams. No phonological restrictions were applied. Instead, transitions between sublexical HMMs in the recognition model were given a fixed weight w —sometimes called *insertion penalty*. Only the most common values were applied, namely $w = 1.0$ and $w = 1/H$, H being the number of phone HMMs.

When using discrete HMMs to model read speech at 16 kHz, all the algorithms yielded almost the same performance. This reveals that applying threshold smoothing to Viterbi estimates is enough to handle independent data in recognition. However, with read speech at 8 kHz significant differences were found, as shown in Table 2. The *embedded model* Baum-Welch algorithm gave the best result (63.12%, with $w = 1.0$), more than 5 points better than that obtained through Viterbi (57.71% with $w = 1.0$) and more than 3 points better than that obtained through the combined approach (59.55% with $w = 1.0$). Note also that including insertion penalties ($w = 1/H$) did not improve the performance.

Table 2. Recognition rates using discrete HMMs. Numbers in parentheses indicate the iteration for which the maximum rate was found.

	$w = 1.0$			$w = 1/H$		
	emBW	**Vit**	**Vit+smBW**	**emBW**	**Vit**	**Vit+smBW**
SENGLAR16	65.67 (8)	65.66 (10)	65.64 (9)	65.28 (8)	65.36 (3)	65.35 (7)
SENGLAR08	63.12 (8)	57.71 (12)	59.55 (11)	62.70 (7)	57.70 (10)	59.15 (8)
INFOTREN	50.70 (19)	49.63 (19)	49.48 (8)	52.53 (20)	51.50 (11)	50.97 (19)
CORLEC-EHU-1 (1)	42.27 (40)	29.98 (16)	29.74 (8)	42.06 (40)	30.07 (17)	30.44 (16)
CORLEC-EHU-1 (2)	44.45 (20)	43.71 (20)	43.98 (20)	44.85 (20)	44.33 (20)	44.68 (20)
CORLEC-EHU-1 (3)	46.10 (17)	45.27 (17)	45.45 (18)	46.29 (17)	45.70 (16)	45.95 (14)
CORLEC-EHU-1 (average)	44.27	39.65	39.72	44.40	40.03	40.36

Do these differences come from a poor segmentation at 8 kHz? Table 3 shows the value of the parameter S defined in Section 6, computed for a hand-labelled part of the training database —the same for SENGLAR16 and SENGLAR08–, consisting of 162 utterances. It reveals that the *quality* of automatic segmentations produced with Viterbi HMMs reduces from almost 90% at 16 kHz to less than 60% at 8 kHz. On the other hand, all the training approaches yielded similar S values at 16 kHz, whereas remarkable differences were observed at 8 kHz, specially between Viterbi HMMs and *embedded model* Baum-Welch HMMs. However, since these latter were not trained based on *that* segmentation but instead on *all* the possible state sequences, the resulting parameters were smoother and more robust than those obtained through Viterbi. This demonstrates that the best HMMs —in terms of recognition rates— might not provide the best segmentations, and vice versa.

Things were quite different when using discrete HMMs to model spontaneous speech. First, as shown in Table 2, recognition rates were much lower: more

Table 3. *Quality* of automatic segmentations obtained with the best discrete HMMs for SENGLAR16 and SENGLAR08.

	Vit	**Vit+smBW**	**emBW**
SENGLAR16	89.72	89.64	87.98
SENGLAR08	58.86	55.03	41.67

than 10 points lower than those obtained for SENGLAR08 in the case of INFOTREN, and more than 20 points lower than those obtained for SENGLAR16 in the case of CORLEC-EHU-1. Second, the *embedded model* Baum-Welch algorithm yielded the best results in all the cases. Third, the combined approach slightly outperformed Viterbi training in the case of CORLEC-EHU-1, but it led to slightly worse results in the case of INFOTREN. Fourth, both Viterbi and the combined approach might lead to very suboptimal models, as in the case of CORLEC-EHU-1(1), where the recognition rate for the *embedded model* Baum-Welch algorithm was 12 points higher than those obtained for the other approaches. This may be due (1) to very inaccurate segmentations in the training corpus, (2) to a great mismatch between the training corpus and the test corpus, either because of speaker features or because noise conditions, and (3) to the increased acoustic variability of spontaneous speech, which may require a more sophisticated smoothing technique for Viterbi estimates. Finally, insertion penalties did almost always improve performance, more clearly in the case of INFOTREN. This may reveal the presence of silences and spontaneous speech events like filled pauses or lengthened vowels, lasting more than the average and implying the insertion of short phones in recognition.

Attending to the asymptotic complexities given in Sections 2, 3 and 4, the *embedded model* Baum-Welch algorithm should be 100 times more expensive than Viterbi training for SENGLAR16 and SENGLAR08, 144 times more expensive for INFOTREN, and 254 times more expensive for CORLEC-EHU-1. However, attending to the experiments, the *embedded model* Baum-Welch algorithm took between 13 and 23 times the time of Viterbi training. This may be due to the fact that the *Forward* and *Backward* functions were zero much more times than expected and the corresponding contributions to the HMM parameters were not really computed. This is an important result, since the *embedded model* Baum-Welch algorithm, in practice, seems to be *only* one order of magnitude more expensive than Viterbi training. Regarding the combined approach, it was in practice (with $N = 3$) around 2 times more expensive than Viterbi training, which is coherent with their asymptotic complexities.

As shown in Table 4, when using continuous HMMs to model read speech, all the training algorithms yielded similar rates. Again, insertion penalties did not improve performance for read speech. Among the mixture sizes tested, $M = 32$ revealed as the best choice, yielding a good balance between recognition rate and computational cost. However, the best rates were found for $M = 64$, outperforming discrete HMMs in more than 7 points for SENGLAR16, and in more than 5 points for SENGLAR08.

Does this mean that Viterbi segmentation improves when using continuous HMMs? The answer is *no*. In fact, as shown in Table 5, except for the *embedded*

Table 4. Recognition rates using continuous HMMs with various mixture sizes (M = 8, 16, 32 and 64), over the read speech databases SENGLAR16 and SENGLAR08.

		$w = 1.0$			$w = 1/H$		
	M	**emBW**	**Vit**	**Vit+smBW**	**emBW**	**Vit**	**Vit+smBW**
SENGLAR16	**8**	70.24 (12)	70.26 (12)	70.43 (12)	69.79 (12)	70.01 (12)	69.82 (10)
	16	71.76 (12)	71.91 (12)	71.83 (12)	71.36 (10)	71.56 (12)	71.45 (12)
	32	72.72 (12)	72.72 (10)	72.72 (11)	72.47 (11)	72.37 (12)	72.42 (11)
	64	73.18 (6)	73.16 (6)	73.15 (6)	72.88 (7)	72.85 (11)	72.90 (11)
SENGLAR08	**8**	65.82 (12)	66.01 (12)	65.91 (12)	65.71 (12)	65.86 (12)	65.77 (12)
	16	67.26 (12)	67.35 (11)	67.28 (12)	67.12 (12)	67.10 (11)	67.13 (12)
	32	68.37 (12)	68.32 (12)	68.30 (12)	68.30 (12)	68.54 (11)	68.21 (12)
	64	68.65 (12)	68.73 (9)	68.76 (11)	68.67 (12)	68.60 (9)	68.60 (7)

model Baum-Welch HMMs for SENGLAR08, the quality of segmentations obtained with continuous HMMs is worse than that obtained with discrete HMMs. The improvement must be claimed for the continuous representation of the output distributions, which in the case of Viterbi training compensates for the sharpness of the estimates. So Viterbi training reveals as the best choice for continuous HMMs, since it yields the same rates as *embedded model* Baum-Welch at a much lower cost.

Table 5. *Quality* of automatic segmentations obtained with the best continuous HMMs for SENGLAR16 and SENGLAR08.

	Vit	**Vit+smBW**	**emBW**
SENGLAR16	88.05	88.00	87.95
SENGLAR08	53.57	53.52	53.41

According to the asymptotic complexities obtained above, using the *embedded model* Baum-Welch algorithm to train continuous HMMs with 8, 16, 32 and 64 gaussians per mixture, should be approximately 60, 85, 105 and 120 times more expensive than using Viterbi training, for SENGLAR16 and SENGLAR08. In practice, the *embedded model* Baum-Welch algorithm was only between 15 and 30 times more expensive than Viterbi training. On the other hand, according to the experiments —and as expected from the asymptotic complexities—, using the combined approach to train continuous HMMs for SENGLAR16 and SENGLAR08, was again around 2 times more expensive than using Viterbi.

8 Conclusions

In this paper we showed the robustness of the *embedded model* Baum-Welch reestimation of discrete HMMs, compared to Viterbi training, for both read and spontaneous speech. In the case of spontaneous speech, the presence of noises, filled pauses, silences, lengthenings and other *long-lasting* events —revealed by the need for insertion penalties in recognition—, could seriously degrade the performance of Viterbi training, whereas Baum-Welch estimates kept robust to those phenomena. Experiments showed that Baum-Welch reestimation was *only*

an order of magnitude more expensive than Viterbi training. The combination of Viterbi segmentation and *single model* Baum-Welch reestimation only outperformed Viterbi training in the case of read speech at 8 kHz, but still yielding lower rates than the *embedded model* Baum-Welch reestimation.

On the other hand, it was shown that the best models in terms of recognition accuracy may not provide the best segmentations. The best models in terms of segmentation quality were always obtained through Viterbi training, which is coherent, since Viterbi training aims to maximize the probability of the most likely state sequence, given the model and the sample. When using discrete HMMs, Viterbi training led to worse estimates than Baum-Welch. This may be explained either by a strong dependence on segmentation quality, by a great mismatch between training and test data, or by the increased acoustic variability of spontaneous speech, which may require a more sophisticated smoothing technique for Viterbi estimates.

With regard to continuous HMMs, all the training approaches yielded the same performance, so Viterbi training —the cheapest approach— revealed as the best choice. It was shown that continuous HMMs did not yield better segmentations than discrete HMMs, so —as expected— the performance of Viterbi training did not only depend on the quality of segmentations but also on the *acoustic resolution* of HMMs, which is drastically increased with the continuous representation of output distributions. Finally, phonetic recognition experiments with continuous HMMs are currently in progress for the spontaneous speech databases INFOTREN and CORLEC-EHU-1, applying only Viterbi training.

References

1. A.P. Dempster, N.M. Laird and D.B. Rubin. *"Maximum likelihood from incomplete data via the EM algorithm"* Journal of the Royal Statistical Society, Series B, Vol. 39, no. 1, pp. 1-38, 1977.
2. X.D. Huang, Y. Ariki and M.A. Jack. *"Hidden Markov Models for Speech Recognition"* Edinburgh University Press, 1990.
3. B.H. Juang and L.R. Rabiner. *"The segmental k-means algorithm for estimating parameters of Hidden Markov Models"* IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 38, no. 9, pp. 1639-1641, September 1990.
4. L.R. Rabiner, J.G. Wilpon and B.H. Juang. *"A segmental k-means training procedure for connected word recognition"*. AT&T Technical Journal, Vol. 65, no. 3, pp. 21-31, May-June 1986.
5. I. Torres, A. Varona and F. Casacuberta. *"Automatic segmentation and phone model initialization in continuous speech recognition"*. Proceedings of the CRIM/FORWISS Workshop on Progress and Prospects of Speech Research and Technology, pp. 286-289, Munich, Germany, 5-7 September, 1994.
6. S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev and P. Woodland. *"The HTK Book (version 2.2)"* Entropic Speech Technology, January 1999.
7. G.D. Forney. *"The Viterbi algorithm"* Proceedings of the IEEE, Vol. 61, pp. 268-278, March 1973.