Acoustic Modelling for Large Vocabulary Continuous Speech Recognition

Steve Young

Engineering Dept., Cambridge University
Trumpington Street, Cambridge, CB2 1PZ, UK
email: sjy@eng.cam.ac.uk

Summary. This chapter describes acoustic modelling in modern HMM-based LVCSR systems. The presentation emphasises the need to carefully balance model complexity with available training data, and the methods of state-tying and mixture-splitting are described as examples of how this can be done. Iterative parameter re-estimation using the forward-backward algorithm is then reviewed and the importance of the component occupation probabilities is emphasised. Using this as a basis, two powerful methods are presented for dealing with the inevitable mis-match between training and test data. Firstly, MLLR adaptation allows a set of HMM parameter transforms to be robustly estimated using small amounts of adaptation data. Secondly, MMI training based on lattices can be used to increase the inherent discrimination of the HMMs.

1. Introduction

The röle of a Large Vocabulary Continuous Speech Recognition (LVCSR) System is to transcribe input speech into an orthographic transcription. Modern LVCSR systems have vocabularies of 5000 to 100000 distinct words and they were developed initially for transcribing carefully spoken dictated speech. Today, however, they are being applied to much more general problems such as the transcription of broadcast news programmes [18, 20] where a variety of speakers, speaking styles, acoustic channels and background noise conditions must be handled.

This chapter describes current approaches to acoustic modelling for LVCSR. Following a brief overview of LVCSR system architecture, HMM-based phone modelling is described followed by an introduction to acoustic adaptation techniques. Finally, some recent research on MMI-based discriminative training for LVCSR is presented as an illustration of possible future developments.

All of the techniques described have been implemented by the author and his colleagues at Cambridge within the HTK LVCSR system [22, 21]. This is a modern design giving state-of-the-art performance and it is typical of the current generation of recognition systems.

2. Overview of LVCSR Architecture

The basic components of an LVCSR system are shown in Fig. 1. The input speech is assumed to consist of a sequence of words and the probability of any specific word sequence can be determined from a *language model*. This is typically a statistical

N-gram model in which the probability of each individual word is conditional only on the identity of the $N-1$ preceding words.

Each word is assumed to consist of a sequence of basic sounds called *phones*. The sequence of phones constituting each word is determined by a pronouncing dictionary and each phone is represented by a hidden Markov Model (HMM). A HMM is a statistical model which allows the distribution of a sequence of vectors to be represented. Given speech parameterised into a sequence of spectral vectors, each phone model determines the probability that any particular segment was generated by that phone.

Thus, for any spoken input to the recogniser, the overall probability of any hypothesised word sequence can be determined by combining the probability of each word as determined by the HMM phone models and the probability of the word sequence as determined by the language model. It is the job of the *decoder* to efficiently explore all the possible word sequences and find the particular word sequence which has the highest probability. This word sequence then constitutes the recogniser output.

A final step in modern systems is to use the recognised input speech to adapt the acoustic phone models in order to make them better matched to the speaker and environment. This is indicated in Fig. 1 by the broken arrow leading from the decoder back to the phone models.

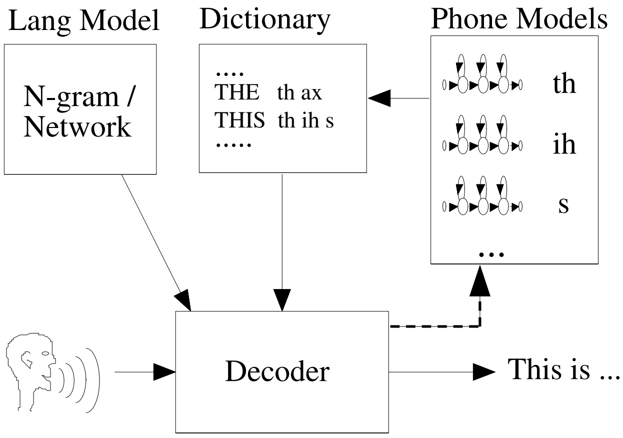

FIGURE 1. The Main Components of an LVCSR System

The mathematical model underlying the above system design was established by Baker, Jelinek and their colleagues from IBM in the 1970's [3, 13]. Figure 2 shows in more detail the way that the probability $P(W|\boldsymbol{Y})$ of a hypothesised word sequence W can be computed given the parameterised acoustic signal $\boldsymbol{Y}$.

The unknown speech waveform is converted by the front-end signal processor into a sequence of acoustic vectors, $\boldsymbol{Y} = \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$. Each of these vectors is

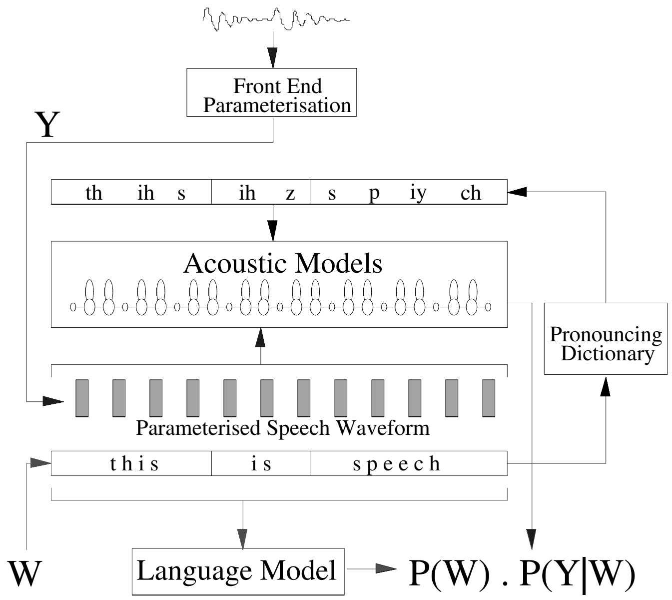

FIGURE 2. The LVCSR Computational Model

a compact representation of the short-time speech spectrum covering a period of typically 10 msecs. If the utterance consists of a sequence of words W, Bayes' rule can be used to decompose the required probability $P(W|\boldsymbol{Y})$ into two components, that is,

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W|\boldsymbol{Y}) = \underset{W}{\operatorname{argmax}} \frac{P(W)P(\boldsymbol{Y}|W)}{P(\boldsymbol{Y})}$$

This equation indicates that to find the most likely word sequence W, the word sequence which maximises the product of $P(W)$ and $P(\boldsymbol{Y}|W)$ must be found.

Figure 2 shows how these relationships might be computed. A word sequence $W =$"This is speech" is postulated and the language model computes its probability $P(W)$. Each word is then converted into a sequence of phones using the pronouncing dictionary. The corresponding HMMs needed to represent the postulated utterance are then concatenated to form a single composite model and the probability of that model generating the observed sequence $\boldsymbol{Y}$ is calculated. This is the required probability $P(\boldsymbol{Y}|W)$. In principle, this process can be repeated for all possible word sequences and the most likely sequence selected as the recogniser output[1].

The recognition accuracy of an LVCSR system depends on a wide variety of factors. However, the most crucial system components are the HMM phone models.

[1]In practice, of course, a more sophisticated search strategy is required. For example, LVCSR decoders typically explore word sequences in parallel, discarding hypotheses as soon as they become improbable.

These must be designed to accurately represent the distributions of each sound in each of the many contexts in which it may occur. The parameters of these models must be estimated from data and since it will never be possible to obtain sufficient data to cover all possible contexts, techniques must be developed which can balance model complexity with available data. Also, the HMM parameters must often track changing speakers and environmental conditions. This requires the ability to robustly adapt the HMM parameters from small amounts of acoustic data and potentially errorful transcriptions. These are the topics at the heart of acoustic modelling for LVCSR systems and they provide the focus for the rest of this chapter.

3. Front End Processing

As explained in the previous section, the input speech waveform must be parameterised into a discrete sequence of vectors in order to represent its characteristics using a HMM. The main features of this parameterisation process are shown in Fig. 3.

The basic premise is that the speech signal can be regarded as stationary (i.e. the spectral characteristics are relatively constant) over an interval of a few milliseconds. Hence, the input speech is divided into blocks and from each block a smoothed spectral estimate is derived. The spacing between blocks is typically 10 msecs and blocks are normally overlapped to give a longer analysis window, typically 25 msecs. As with all processing of this type, it is usual to apply a tapered window function (e.g. Hamming) to each block. Also the speech signal is often pre-emphasised by applying high frequency amplification to compensate for the attenuation caused by the radiation from the lips.

Compared to using a simple linear spectral estimate, performance is improved by using a non-linear Mel-filterbank followed by a Discrete Cosine Transform (DCT) to form so-called Mel-Frequency Cepstral Coefficients (MFCCs) [6]. The Mel-scale is designed to approximate the frequency resolution of the human ear being linear upto 1000Hz and logarithmic thereafter. The DCT is computed using

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j-0.5)\right)$$

where m_j is the log energy in each Mel-filter band and c_i is the required cepstral coefficient. The DCT compresses the spectral information into the lower order coefficients and it also has the effect of decorrelating the signal thereby improving assumptions of statistical independence. The MFCC coefficients are often normalised by subtracting the mean. This has the effect of removing any long term spectral bias on the input signal.

The static MFCC coefficients are usually augmented by appending time derivatives

$$\Delta_t = \frac{\sum_{\tau=1}^{D} \tau(c_{t+\tau} - c_{t-\tau})}{2\sum_{\tau=1}^{D} \tau^2}$$

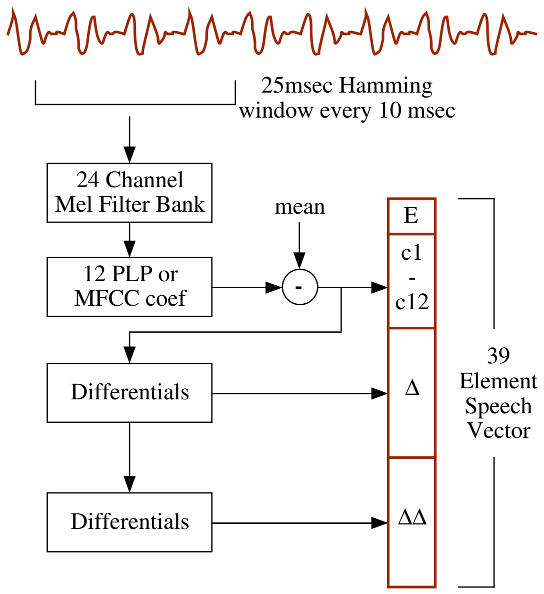

FIGURE 3. Front End Signal Processing

The same regression formula can then be applied to the Δ coefficients to give $\Delta\Delta$ (or acceleration) coefficients. These differentials compensate for the rather poor assumption made by the HMMs that successive speech vectors are independent.

MFCC coefficients are widely used in LVCSR systems and give good results. Similar performance can also be achieved by using LP coefficients to derive a smoothed spectrum which is then perceptually weighted to give Perceptually weighted Linear Prediction (PLP) coefficients[10].

An important point to emphasise is the degree to which the design of the front-end has evolved to optimise the subsequent pattern-matching. For example, in the above, the log compression, DCT transform and delta coefficients are all introduced primarily to satisfy the assumptions made by the acoustic modelling component.

4. Basic Phone Modelling

Each basic sound in an LVCSR system is represented by a HMM which can be regarded as a random generator of acoustic vectors (see Fig. 4). It consists of a sequence of states connected by probabilistic transitions. It changes to a new (possibly the same) state each time period generating a new acoustic vector according to the output distribution of that state. The transition probabilities therefore model the durational variability in real speech and the output probabilities model the spectral variability.

4..1 HMM Phone Models

HMM phone models typically have three *emitting* states and a simple left-right topology as illustrated by Fig 4. The entry and exit states are provided to make it easy to join models together. The exit state of one phone model can be merged with the entry state of another to form a composite HMM. This allows phone models to be joined together to form words and words to be joined together to cover complete utterances.

More formally, a HMM phone model consists of

1. Non-emitting entry and exit states
2. A set of internal states x_j, each with output probability $b_j(\boldsymbol{y}_t)$
3. A transition matrix $\{a_{ij}\}$ defining the probability of moving from state x_i to x_j[2]

For high accuracy, modern systems uses continuous density mixture Gaussians to model the output probability distributions, i.e.

$$b_j(\boldsymbol{y}_t) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\boldsymbol{y}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$$

where $\mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the normal distribution with mean $\boldsymbol{\mu}$ and (diagonal) covariance $\boldsymbol{\Sigma}$.

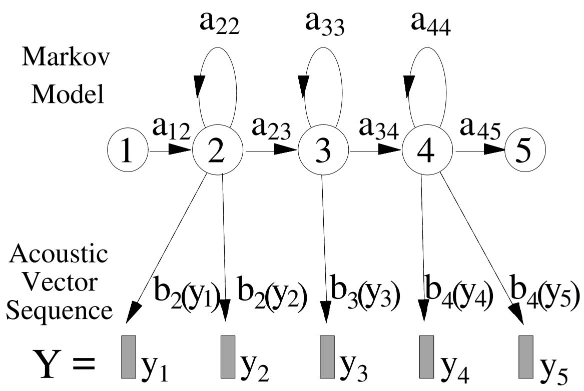

FIGURE 4. A HMM Phone Model

The joint probability of a vector sequence $\boldsymbol{Y}$ and state sequence X given some model M is calculated simply as the product of the transition probabilities and the output probabilities. So for the state sequence X in Figure 4

[2] In practice, the transition matrix parameters have little effect on recognition performance compared to the output distributions. Hence, their estimation is not considered in this chapter.

$$P(\boldsymbol{Y}, X|M) = a_{12}b_2(\boldsymbol{y}_1)a_{22}b_2(\boldsymbol{y}_2)a_{23}b_3(\boldsymbol{y}_3)\ldots$$

More formally, the joint probability of an acoustic vector sequence $\boldsymbol{Y}$ and some state sequence $X = x(1), x(2), x(3), \ldots, x(T)$ is

$$P(\boldsymbol{Y}, X|M) = a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{y}_t)a_{x(t)x(t+1)} \tag{1}$$

where $x(0)$ is constrained to be the model entry state and $x(T+1)$ is constrained to be the model exit state.

In practice, of course, only the observation sequence $\boldsymbol{Y}$ is known and the underlying state sequence X is hidden. This is why it is called a *Hidden Markov Model*. For recognition, $P(\boldsymbol{Y}|M)$ can be approximated by finding the state sequence which maximises equation 1. A simple algorithm exists for computing this efficiently called the *Viterbi* algorithm and it is the basis of many decoder designs where determination of the most likely state sequence is the key to recognising the unknown word sequence[17].

4..2 HMM Parameter Estimation

In this chapter, the main interest is in designing accurate HMM phone models and estimating their parameters. For the moment, assume that there is a single HMM for each distinct phone and that there is a single spoken example available to estimate its parameters. Consider first the case where each HMM has a single state and each state has only a single Gaussian component. In this case, the state mean and covariance would be given by simple averages

$$\boldsymbol{\mu}_i = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}_t$$

$$\boldsymbol{\Sigma}_i = \frac{1}{T}\sum_{t=1}^{T} (\boldsymbol{y}_t - \boldsymbol{\mu_i})(\boldsymbol{y}_t - \boldsymbol{\mu_i})'$$

This can be extended to the case of a real HMM with multiple states and multiple Gaussian components per state, by using weighted averages as follows

$$\boldsymbol{\mu}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t)\boldsymbol{y}_t}{\sum_{t=1}^{T} \gamma_{jm}(t)} \tag{2}$$

$$\boldsymbol{\Sigma}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t)(\boldsymbol{y}_t - \boldsymbol{\mu_i})(\boldsymbol{y}_t - \boldsymbol{\mu_i})'}{\sum_{t=1}^{T} \gamma_{jm}(t)} \tag{3}$$

where $\gamma_{jm}(t)$ is the so-called *component occupation probability*. The key idea here is that each training vector is distributed amongst the HMM Gaussian components according to the probability that it was generated by that component. Since $\gamma_{jm}(t)$ depends on the existing HMM parameters, an iterative procedure is suggested

1. choose initial values for the HMM parameters
2. compute the component occupation probabilities in terms of the existing HMM parameters
3. update the HMM parameters using equations 2 and 3

The component occupation probabilities can be computed efficently using a recursive procedure known as the *Forward-Backward* algorithm. Firstly, define the *forward probability* $\alpha_j(t) = P(\boldsymbol{y}_1 \ldots \boldsymbol{y}_t, x_t = j)$. As illustrated by Fig. 5, this can be computed recursively by

$$\alpha_j(t) = \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} b_j(\boldsymbol{y}_t)$$

Similarly, the *backward probability* is defined as $\beta_j(t) = P(\boldsymbol{y}_{t+1} \ldots \boldsymbol{y}_T | x_t = j)$, this can also be computed recursively by

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(\boldsymbol{y}_{t+1}) \beta_j(t+1)$$

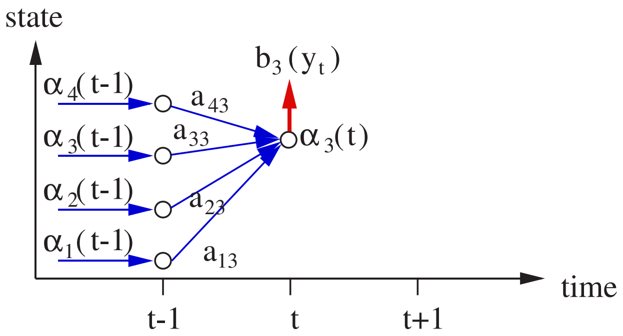

FIGURE 5. The Forward Probability Calculation

Given the forward and backward probabilities, the state occupation probability is simply

$$\gamma_j(t) = \frac{1}{P} \alpha_j(t) \beta_j(t)$$

where $P = P(\boldsymbol{Y}|M) = \alpha_N(T)$, and the component occupation probability is

$$\gamma_{jm}(t) = \frac{1}{P} \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} c_{jm} \mathcal{N}(\boldsymbol{y}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \beta_j(t)$$

The estimation of HMM parameters using the above procedure is an example of the Expectation-Maximisation (EM) algorithm and it converges such that the likelihood of the training data given the HMM i.e. $P(\boldsymbol{Y}|M)$ achieves a local maximum [4, 7].

Although the above is now established text-book material, it is not usually presented in terms of simple weighted averages. This is a pity since even though it lacks mathematical rigour, it offers considerable insight into the reestimation process. For example, it is easy to see that when multiple training instances are provided, the same basic equations 2 and 3 still apply. The sums required to compute the numerators and denominators of these equations are first accumulated over all of the data, and then the parameters are updated.

To complete the presentation of basic HMM phone model estimation, one final unrealistic assumption must be removed. In practice, there is no access to individual speech segments corresponding to a single phone model. Instead, the training data consists of naturally spoken utterances annotated at the word level. Rather than attempting to segment this data, it can be used directly for parameter estimation by adopting an *embedded training* paradigm as illustrated in Fig. 6. The phone sequence corresponding to each training utterance is determined from a dictionary. Then a composite HMM is constructed by concatenating all of the phone models and the numerator and denominator statistics needed for equations 2 and 3 are accumulated for all of the phones in the sequence. This is repeated for all of the training data and finally, all of the phone model parameters are re-estimated in parallel.

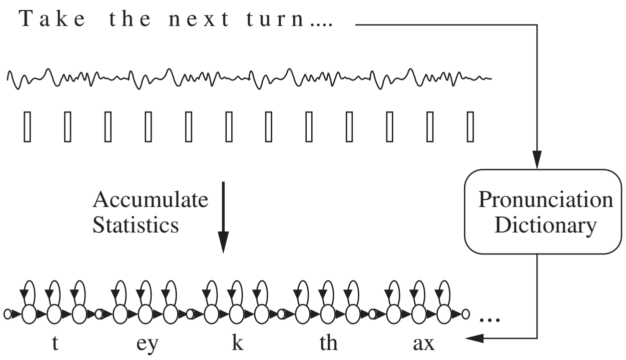

FIGURE 6. Embedded HMM Training

4..3 Context-Dependent Phone Models

So far there has been an implicit assumption that only one HMM is required per phone, and since approximately 45 phones are needed for English, it may be thought that only 45 phone HMMs need be trained. In practice, however, contextual effects cause large variations in the way that different sounds are produced. Hence, to achieve good phonetic discrimination, different HMMs have to be trained for each different context. The simplest and most common approach is to use *triphones* whereby every phone has a distinct HMM model for every unique pair of left and right neighbours. For example, suppose that the notation `x-y+z` represents the phone `y` occurring after phone `x` and before phone `z`. The phrase, "Beat it!" would be represented by the phone sequence `sil b iy t ih t sil`, and if triphone HMMs were used the sequence would be modelled as

```
sil sil-b+iy b-iy+t iy-t+ih t-ih+t ih-t+sil sil
```

Notice that the triphone contexts span word boundaries and the two instances of the phone `t` are represented by different HMMs because their contexts are different. This use of so-called *cross-word triphones* gives the best modelling accuracy but leads to complications in the decoder. Simpler systems result from the use of *word-internal triphones* where the above example would become

```
sil b+iy b-iy+t iy-t ih+t ih-t sil
```

Here far fewer distinct models are needed simplifying both the parameter estimation problem and decoder design. However, the cost is an inability to model contextual effects at word boundaries and in fluent speech these are considerable.

The use of Gaussian mixture output distributions allows each state distribution to be modelled very accurately. However, when triphones are used they result in a system which has too many parameters to train. For example, a large vocabulary cross-word triphone system will typically need around 60,000 triphones[3]. In practice, around 10 mixture components per state are needed for reasonable performance. Assuming that the covariances are all diagonal, then a recogniser with 39 element acoustic vectors would require around 790 parameters per state. Hence, 60,000 3-state triphones would have a total of 142 million parameters!

The problem of too many parameters and too little training data is absolutely crucial in the design of a statistical speech recogniser. Early systems dealt with the problem by tying all Gaussian components together to form a pool which was then shared amongst all HMM states. In these so-called tied-mixture systems, only the mixture component weights were state-specific and these could be smoothed by interpolating with context independent models[11, 5]. Modern systems, however, commonly use a technique called *state-tying* [12, 24]. in which states which are acoustically indistinguishable are tied together. This allows all the data associated with each individual state to be pooled and thereby gives more robust estimates for the parameters of the tied-state.

State-tying is illustrated in Fig 7. At the top of the figure, each triphone has its own private output distribution. After clustering similar states together and tying, several states share distributions. This figure also illustrates an important practical advantage of using Gaussian mixture distributions in that it is very simple to increase the number of mixture components in a system by so-called *mixture splitting*. In

[3]With 45 phones, there are $45^3 = 91125$ possible triphones but not all can occur due to the phonotactic constraints of the language

mixture-splitting, the more dominant Gaussian components in each state are cloned and then the means are perturbed by a small fraction of the standard deviation. The resulting HMMs are then re-estimated using the forward-backward algorithm. This process can be repeated so that a single Gaussian system can be converted to the required multiple mixture component system in just a few iterations.

Mixture-splitting allows a tied-state system to be built using single Gaussians and then converted to a multiple component system after the states have been tied. This avoids the problem of having too little data to train untied mixture Gaussians and it simplifies the clustering process since it is much easier to compute the similarity between single Gaussian distributions.

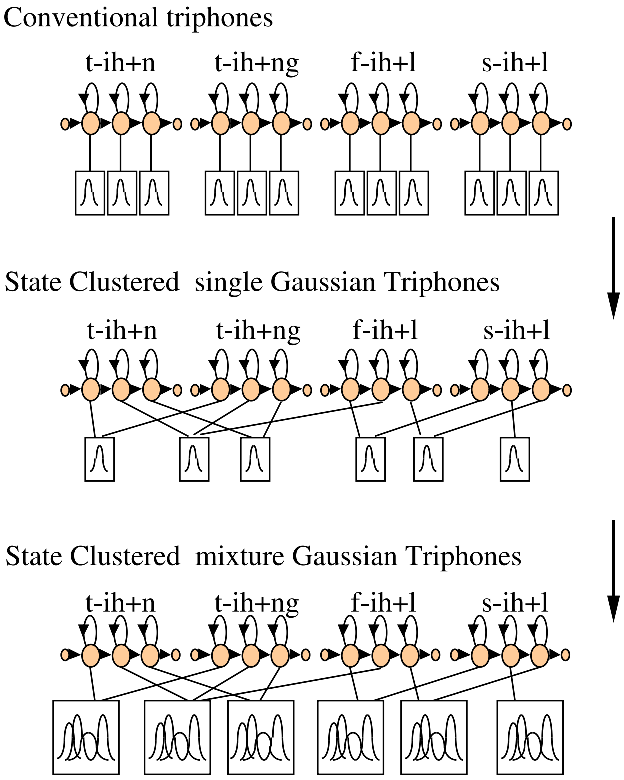

FIGURE 7. Tied-State Triphone Construction

Although almost any clustering technique could be used to decide which states to tie, in practice, the use of *phonetic decision trees*[2, 14, 23] is preferred. In decision tree-based clustering, a binary tree is built for each phone and state position. Each tree has a yes/no phonetic question such as “Is the left context a nasal?” at each node. Initially all states for a given phone state position are placed at the root node of a tree. Depending on each answer, the pool of states is successively split and this continues until the states have trickled down to leaf-nodes. All states in the same leaf node are then tied. For example, Fig 8 illustrates the case of tying the centre states of all triphones of the phone /aw/ (as in “out”). All of the states trickle down the tree and depending on the answer to the questions, they end up at one of the shaded terminal nodes. For example, in the illustrated case, the centre state of `s-aw+n` would join the second leaf node from the right since its right context is a central consonant, and its right context is a nasal but its left context is not a central stop.

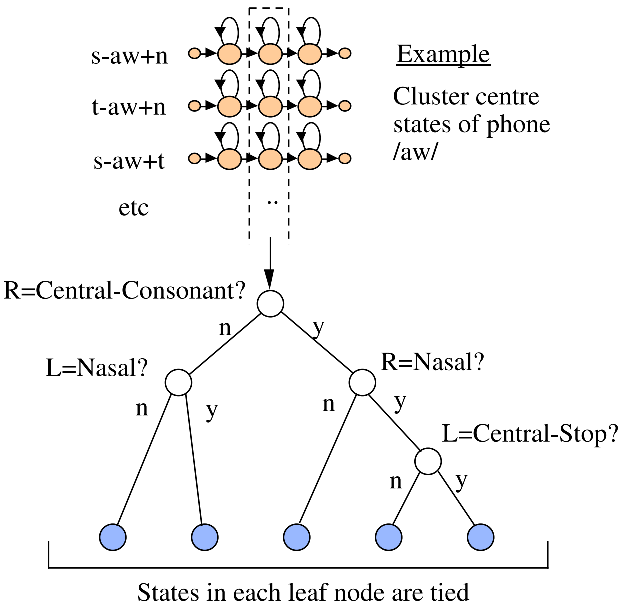

FIGURE 8. Phonetic Decision Tree-based Clustering

The questions at each node are chosen from a large predefined set of possible contextual effects in order to maximise the likelihood of the training data given the final set of state tyings. The tree is grown starting at the root node which represents all states as a single cluster. Each state s_i has an associated set of observations $\boldsymbol{Y} = \{\boldsymbol{y}_{i,1}, \ldots, \boldsymbol{y}_{i,N_i}\}$. If $\boldsymbol{S} = \{s_1, s_2, \ldots, s_k\}$ defines a pool of states, then the log likelihood of the data associated with this pool is defined as

$$L(\boldsymbol{S}) = \sum_{i=1}^{K} logP(\boldsymbol{Y}_i|\boldsymbol{\mu}_S, \boldsymbol{\Sigma}_S)$$

This is the likelihood of the data if all of the associated states are merged to form a single Gaussian with mean $\boldsymbol{\mu}_S$ and variance $\boldsymbol{\Sigma}_S$.

This pool of states $\boldsymbol{S}$ is now split into two partitions by asking a question based on the phonetic context. Since the likelihood of each partition is computed using the overall mean and variance for that partition, the total likelihood of the partitioned data will increase by an amount

$$\Delta = L(\boldsymbol{S}_y) + L(\boldsymbol{S}_n) - L(\boldsymbol{S})$$

Δ is therefore computed for all possible questions and the question q* which maximises it is selected. The process then repeats by splitting each of the two newly formed nodes. It is terminated when either Δ falls below a predefined threshold or when the amount of data associated with one of the split nodes would fall below a threshold.

Note that provided the state occupancy counts γ_j are retained from the reestimation of the original untied single Gaussian system, all of the likelihoods needed for the above tree growing procedure can be computed directly from the model parameters and no reference is needed to the original data.

In practice, phonetic decision trees give compact good-quality state clusters which have sufficient associated data to robustly estimate mixture Gaussian output probablity functions. Furthermore, they can be used to synthesise a HMM for any possible context whether it appears in the training data or not, simply by descending the trees and using the state distributions associated with the terminating leaf nodes. Finally, phonetic decision trees can be used to include more than simple triphone contexts. For example, questions spanning ± 2 phones can be included and they can also take account of the presence of word boundaries.

5. Adaptation for LVCSR

Large vocabulary speech recognisers require very large databases of acoustic data to train them. These databases usually contain many speakers recorded under controlled conditions, typically noise-free and wide-band. The resulting HMMs are therefore speaker independent (SI) and optimised for a specific microphone and environment.

For practical applications, an LVCSR system trained in this way results in a number of limitations

- SI performance is inferior to speaker dependent (SD) performance
- many speakers are outliers with respect to the original training population and will therefore be poorly recognised
- channel conditions will vary with different microphones and recording conditions
- background noise is common

Hence, there is often a *mis-match* between the training and testing conditions and it is important to reduce this mis-match as much as possible by using the test data itself to adapt the HMM parameters to be more suited to the current speaker, channel and environmental conditions.

There are a number of distinct modes of adaptation

- *Supervised* – an exact transcription of all the adaptation data is available
- *Unsupervised* – the recogniser output is used to transcribe the adaptation data
- *Enrolment Mode* – the adaptation data is applied off-line prior to recognition
- *Incremental Mode* – each new recogniser output is used to augment the adaptation data.
- *Transcription Mode* – non-causal, all recognised speech is saved, used for adaptation, then all speech is re-recognised

Clearly the choice and combination of modes depends on the application and ergonomic considerations. For example, a personal desk-top dictation system will typically use supervised enrolment, whereas an off-line broadcast news transcription service will use unsupervised transcription mode.

5..1 Maximum Likelihood Linear Regression

There are many different approaches to adaptation, but one of the most versatile is Maximum Likelihood Linear Regression (MLLR) [15, 9]. MLLR seeks to find an affine transform of the Gaussian means which maximises the likelihood of the adaptation data, i.e.

$$\hat{\boldsymbol{\mu}}_r = \boldsymbol{A}_m \boldsymbol{\mu}_r + \boldsymbol{b}_m = \boldsymbol{W}_m \boldsymbol{\eta}_r$$

where $\boldsymbol{W}_m = [\boldsymbol{b}_m \ \boldsymbol{A}_m]$ and $\boldsymbol{\eta}_r = \left[1 \ \boldsymbol{\mu}_r^T\right]^T$.

The key to the power of this adaptation approach is that a single transformation $\boldsymbol{W}_m$ can be shared across a set of Gaussian mixture components. When the amount of adaptation data is limited, a single transform can be shared across all Gaussians in the system. As the amount of data increases, the HMM state components can be grouped into classes with each class having its own transform. As the amount of data increases further, the number of classes and therefore transforms increases correspondingly leading to better and better adaptation.

The number of transforms is usually determined automatically using a *regression class tree* as illustrated in Fig. 9. Each node represents a regression class i.e. a set of Gaussian components which will share a single transform. For a given adaptation set, the tree is descended and the most specific set of nodes is selected for which there is sufficient data (for example, the filled-in nodes in the figure). The regression class tree itself can be built using similar techniques to those described in the previous section for state-clustering [8].

5..2 Estimating the MLLR Transforms

As its name suggests, the parameters of the transforms $\boldsymbol{W}_m$ are estimated so as to maximise the likelihood of the adaptation data with respect to the transformed HMM parameters. This log likelihood $\mathcal{L}$ is given by

$$\mathcal{L} = \sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_r(t) log \left\{ K_r exp(-\frac{1}{2}(\boldsymbol{y}(t) - \boldsymbol{W}_m \boldsymbol{\eta}_r)' \Sigma_r^{-1} (\boldsymbol{y}(t) - \boldsymbol{W}_m \boldsymbol{\eta}_r) \right\}$$

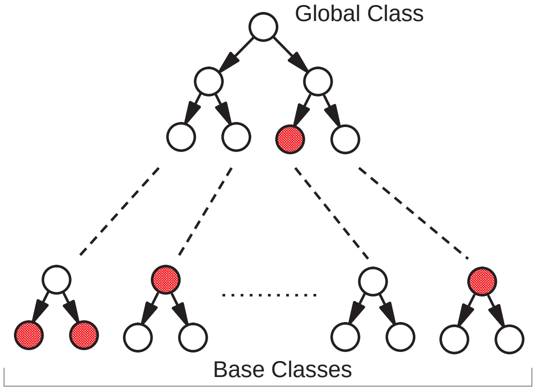

FIGURE 9. An MLLR Regression Tree

where r ranges over the R Gaussian components belonging to the regression class associated with transform $\boldsymbol{W}_m$ and K_r are normalising constants. Differentiating wrt to $\boldsymbol{W}_m$ and setting the result equal to zero gives

$$\sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_r(t)\Sigma_r^{-1}\boldsymbol{y}(t)\boldsymbol{\eta}_r' = \sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_r(t)\Sigma_r^{-1}\boldsymbol{W}_m\boldsymbol{\eta}_r\boldsymbol{\eta}_r'$$

which can be written in matrix form as

$$\boldsymbol{Z} = \sum_{r=1}^{R}\boldsymbol{V}^r\boldsymbol{W}_m\boldsymbol{D}^r$$

There is no computationally efficient solution for this in the full covariance case. However, for diagonal covariance, the i'th row of $\boldsymbol{W}_m$ is given by

$$\boldsymbol{z}_i' = \boldsymbol{w}_i'\sum_{r=1}^{R}v_{ii}^r\boldsymbol{D}^r$$

which can be solved by inverting the matrix $\boldsymbol{D}^r$.

In addition to mean adaptation, variance adaptation is also possible. A particularly simple form of transform to use for this is $\boldsymbol{H}_m$ where

$$\hat{\boldsymbol{\Sigma}}_r^{-1} = \boldsymbol{C}_r\boldsymbol{H}_m^{-1}\boldsymbol{C}_r'$$

and where $\boldsymbol{C}_r$ is the Choleski factor of $\boldsymbol{\Sigma}_r^{-1}$. $\boldsymbol{H}_m$ is easy to estimate, because rewriting the quadratic in the exponent of the Gaussian as

$$\frac{1}{2}\left((\boldsymbol{C}_r'\boldsymbol{y}(t) - \boldsymbol{C}_r'\boldsymbol{\mu}_r)'\boldsymbol{H}_m^{-1}(\boldsymbol{C}_r'\boldsymbol{y}(t) - \boldsymbol{C}_r'\boldsymbol{\mu}_r)\right)$$

it can be seen that the form is the same as for the re-estimation of the HMM variances using equation 3, i.e.

$$\hat{\boldsymbol{H}}_m = \frac{\boldsymbol{C}'_m \left[\sum_{t=1}^{T} \gamma_m(t)(\boldsymbol{y}(t) - \boldsymbol{\mu}_m)(\boldsymbol{y}(t) - \boldsymbol{\mu}_m)'\right] \boldsymbol{C}_m}{\sum_{t=1}^{T} \gamma_m(t)}$$

Instead of having a separate transform for the means and variances, a single *constrained* transform can be applied to both, i.e.

$$\begin{aligned} \hat{\boldsymbol{\mu}}_r &= \boldsymbol{A}_m \boldsymbol{\mu}_r + \boldsymbol{b}_m \\ \hat{\boldsymbol{\Sigma}}_r &= \boldsymbol{A}_m \boldsymbol{\Sigma}_r \boldsymbol{A}'_m \end{aligned}$$

This has no closed-form solution but an iterative solution is possible [9]. A key advantage of this form of adaptation is that the likelihoods can be calculated as

$$\mathcal{L}(\boldsymbol{y}(t); \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{A}, \boldsymbol{b}) = \mathcal{N}(\boldsymbol{A}\boldsymbol{y}(t) + \boldsymbol{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log(|\boldsymbol{A}|)$$

This means that the transform can be applied to the data rather than the HMM parameters which may be more convenient for some applications. When using incremental adaptation, this transform can also be more efficient to compute since although it is iterative, only one iteration is needed for each new increment of adaptation data and, unlike the unconstrained case, it does not require any expensive matrix inversions.

Finally, it should be noted that for unsupervised adaptation, the quality of the transforms depends on the accuracy of the recogniser output. One obvious way to improve this is to iterate the recognition and adaptation cycle.

6. Progress in LVCSR

Progress in LVCSR over the last decade has been tracked by the US National Institute of Standards and Technology (NIST) in the form of annual speech recognition evaluations. These have evolved over the years but the basic style is that participating organisations are provided with the necessary training data and some development test data at the start of the year. Towards the end of the year, NIST then distribute *unseen* evaluation test data and each organisation then recognises this data and sends the output back to NIST for scoring. Initially, the participating organisations were all US funded research groups, but since 1992, the evaluations have been open to non-US groups.

Table 6. lists the different evaluation tasks along with their main charactistics. In this table, the test mode indicates whether or not the evaluation data has a closed or open vocabulary. If the vocabulary is open, then the test data will contain so-called Out-of-Vocabulary (OOV) words which contribute to the error rate. PP denotes perplexity which is similar to the *average branching factor* and indicates the degree of

uncertainty as each new word is encountered. The % word error (WER) rates indicate the approximate performance of the best systems at the time they were tested.

RM denotes the *Naval Resource Management Task* which is an artificial task based on spoken access to a database of naval information. WSJ (Wall Street Journal) and NAB (North American Business news) are large vocabulary dictation tasks in which the source material is taken from either the WSJ or more generally, a range of US newspapers (NAB). Finally, the current BN (Broadcast News) task involves the transcription of arbitrary broadcast news material. This challenging task introduces many new problems including the need to segment and classify a continuous audio stream, handle a range of speakers and channels, and cope with a wide variety of interfering signals including noise, music and other speakers. Note that all of these tasks involve speaker independent recognition of continuous speech.

As can be seen from the table, the state of the art on clean speech dictation within a limited domain such as business news is around 7%WER. The LVCSR systems which can achieve this are typically of the sort described in this chapter i.e. tied-state mixture Gaussian HMM based with cross-word triphones, N-gram language models and incremental unsupervised MLLR. The error rates for broadcast news transcription are much higher reflecting the many additional problems that it poses. However, this is an active area of research and the error rates will fall quickly.

When	Task	Train Data	Vocab Size	Test Mode	PP	WER %
87-92	RM	4 Hrs	1k	Closed	60	4
92-94	WSJ	12 Hrs	5k	Closed	50	5
92-94	WSJ	66 Hrs	20k	Open	150	10
94-95	NAB	66 Hrs	65k	Open	150	7
95-96	BN	50 Hrs	65k	Open	200	30

7. Discriminative Training for LVCSR

All of the methods described in the preceding sections are so-called *Maximum Likelihood* (ML) methods. They are based on the simple premise that the parameters of an LVCSR system should be designed to give the closest possible fit to the training data, and where appropriate the adaptation data. Unfortunately, as noted already, there is often a mis-match between the training and test data so that maximising the fit to the training data does not necessarily mean that the ultimate recognition performance will be optimised.

All this has been well-known for many years and several alternative parameter estimation schemes have been developed. In particular, a maximum mutual information (MMI) criterion can be used [1] which seeks to increase the *a posteriori* probability of the model sequence corresponding to the training data given the training data.

More formally, for R training observations $\{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_r, \ldots \boldsymbol{Y}_R\}$ with corresponding transcriptions $\{w_r\}$, the MMI objective function is given by

$$\mathcal{F}(\lambda) = \sum_{r=1}^{R} \log \frac{P_\lambda(\boldsymbol{Y}_r | \mathcal{M}_{w_r}) P(w_r)}{\sum_{\hat{w}} P_\lambda(\boldsymbol{Y}_r | \mathcal{M}_{\hat{w}}) P(\hat{w})}$$

where $\mathcal{M}_w$ is the composite model corresponding to the word sequence w and $P(w)$ is the probability of this sequence as determined by the language model.

The numerator of $\mathcal{F}(\lambda)$ corresponds to the likelihood of the training data given the correct model sequence, whereas the denominator corresponds to its likelihood given all the other possible sequences. Maximising the numerator whilst simultaneously minimising the denominator gives HMMs trained using the MMI criterion improved discrimination compared to ML.

The problem with using MMI in practice is that the denominator is impossible to compute for anything other than simple isolated word systems which have a finite number of possible model sequences to consider. Modern LVCSR systems, however, are capable of generating lattices of alternative recognition hypotheses. This last section on acoustic modelling explains how these lattices can be used to discriminatively train the HMMs of an LVCSR system using the MMI criterion [19].

To make the evaluation of $\mathcal{F}(\lambda)$ tractable, the denominator can be approximated by

$$\sum_{\hat{w}} P_\lambda(\boldsymbol{Y}_r | \mathcal{M}_{\hat{w}}) P(\hat{w}) \Rightarrow P_\lambda(\boldsymbol{Y}_r | \mathcal{M}_{rec})$$

where $\mathcal{M}_{rec}$ is a model constructed such that for all paths in every $\mathcal{M}_{\hat{w}}$ there is a corresponding path of equal probability in $\mathcal{M}_{rec}$ i.e. $\mathcal{M}_{rec}$ is the model used for recognition. Thus, the MMI objective function now becomes

$$\mathcal{F}(\lambda) = \sum_{r=1}^{R} \log \frac{P_\lambda(\boldsymbol{Y}_r | \mathcal{M}_{cor})}{P_\lambda(\boldsymbol{Y}_r | \mathcal{M}_{rec})}$$

Unlike the ML case, it is not possible to derive provably convergent re-estimation formula. However, Normandin has derived the following formulae which work well in practice [16]

$$\hat{\mu}_{j,m} = \frac{\left\{\theta^{cor}_{j,m}(\boldsymbol{Y}) - \theta^{rec}_{j,m}(\boldsymbol{Y})\right\} + D\mu_{j,m}}{\left\{\gamma^{cor}_{j,m} - \gamma^{rec}_{j,m}\right\} + D} \tag{4}$$

$$\hat{\sigma}^2_{j,m} = \frac{\left\{\theta^{cor}_{j,m}(\boldsymbol{Y}^2) - \theta^{rec}_{j,m}(\boldsymbol{Y}^2)\right\} + D(\sigma^2_{j,m} + \mu^2_{j,m})}{\left\{\gamma^{cor}_{j,m} - \gamma^{rec}_{j,m}\right\} + D} - \hat{\mu}^2_{j,m} \tag{5}$$

where

$$\theta_{j,m}(x) = \sum_{r=1}^{R} \sum_{t=1}^{T_r} x^r(t)\ \gamma^r_{j,m}(t)$$

and

$$\gamma_{j,m} = \sum_{r=1}^{R} \sum_{t=1}^{T_r} \gamma_{j,m}^{r}(t)$$

In these equations, D is a constant which determines the rate of convergence of the re-estimation formula. If D is too big then convergence is too slow, if it is too small then instability can occur. In practice, D should be set to ensure that all variances remain positive. It is also beneficial to compute separate values of D for each phone model.

As with ML-based parameter estimation, the crucial quantities to compute are the component occupation probabilities $\gamma_{j,m}^{cor}$ and $\gamma_{j,m}^{rec}$. The former is straightforward but the latter requires all possible word sequences to be considered. As noted earlier, however, lattices provide a tractable way of approximating this. A lattice is a directed graph in which each arc represents a hypothesised word. Within any given lattice, it is simple to compute the probability of being at any node using the forward-backward algorithm. For node l in the lattice and preceding words $w_{k,l}$ spanning nodes k to l, the forward probability is given by

$$\bar{\alpha}_l = \sum_k \bar{\alpha}_k P_{acoust}(w_{k,l}) P_{lang}(w_{k,l})$$

where P_{acoust} is the likelihood of word $w_{k,l}$ hypothesised between the time instances corresponding to nodes k and l, and P_{lang} is the language model probability of $w_{k,l}$. The backward probabilities $\bar{\beta}_k$ are computed in a similar fashion starting from the end of the lattice. For each pair of nodes k and l, the corresponding $\bar{\alpha}_k$ and $\bar{\beta}_l$ can be used to compute the required occupation probabilities within the word hence the quantities needed to compute the reestimation equations 4 and 5 can be calculated.

The overall framework of MMI training using lattices is illustrated in Fig. 10. First a pair of lattices is generated for each sentence in the training database: one for the numerator using the recogniser constrained by the correct word sequence, and the other using the unconstrained recogniser. The re-estimation process then consists of rescoring the lattices with the current model set, computing the occupation probabilities and finally, updating the parameters. Note that strictly the lattices should be recomputed at every reestimation cycle but this would be computationally very expensive and probably unnecessary since the set of confusable word sequences will change very little.

The effectiveness of the MMI training procedure is illustrated in Fig. 11 which shows the training of a simple single Gaussian WSJ system using 60 hours of training data. The diagram on the left shows the way the MMI objective function increases at each iteration. The diagram on the right plots the % WER on both the training data and an evaluation test set. As can be seen, the errors on the training set are substantially reduced whereas much more modest improvements on the test set are obtained. More formal testing of the lattice-based MMI training procedure on a full WSJ system has shown that between 5% and 15% relative reductions in error rate can be achieved [19]. More importantly, perhaps, it appears that MMI is most effective with smaller less complex systems (i.e. systems with relatively few

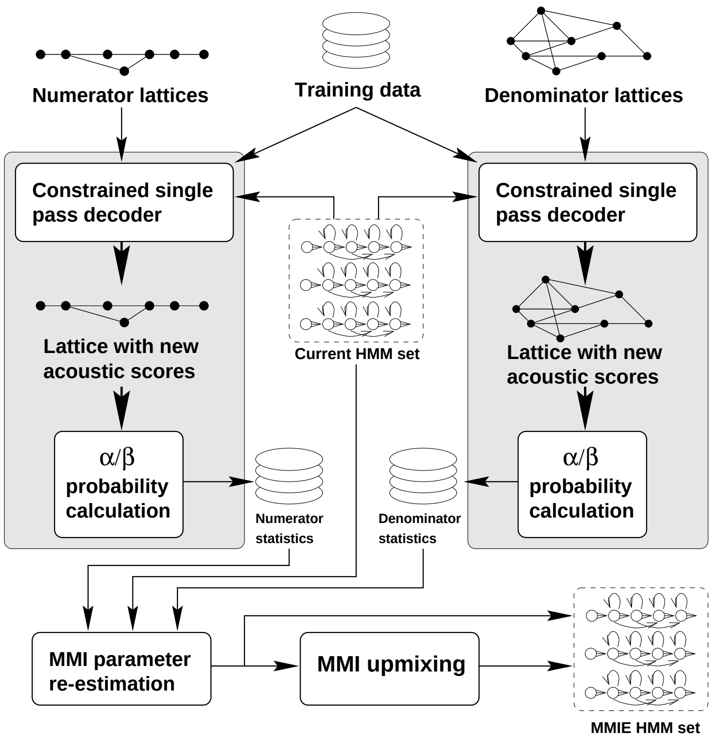

FIGURE 10. Lattice-based Framework for MMI Training of an LVCSR System

mixture components per state). Thus, MMI training may be particularly useful for making small compact LVCSR systems without sacrificing accuracy.

8. Conclusions

This chapter has described acoustic modelling in modern HMM-based LVCSR systems. The presentation has emphasised the need to carefully balance model complexity with available training data. The methods of state-tying and mixture-splitting allow this to be achieved in a simple and straightforward way. Iterative parameter re-estimation using the forward-backward algorithm has been described and the importance of the component occupation probabilities has been emphasised. Using this as a basis, two powerful methods have been presented for dealing with the inevitable mis-match between training and test data. Firstly, MLLR adaptation allows a set of HMM parameter transforms to be robustly estimated using small amounts of adaptation data. Secondly, MMI training based on lattices can be used to increase the inherent discrimination of the HMMs.

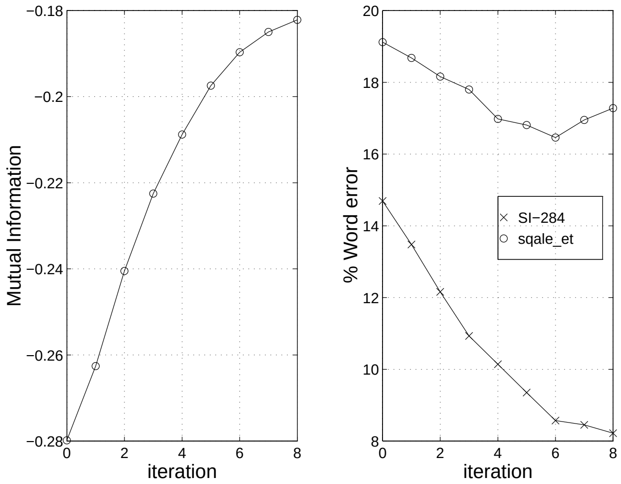

FIGURE 11. MMI Training Performance

Taken together, the methods described allow speaker independent LVCSR systems to be built with average error rates well below 10%. Future developments will aim to reduce this figure further. They will also focus on more general transcription tasks such as the transcription of broadcast news material making the deployment of LVCSR technology feasible across a wide range of IT applications.

9. REFERENCES

[1] L. Bahl, P. Brown, P. de Souza, and R. Mercer. Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. In *Proc ICASSP*, pages 49–52, Tokyo, 1986.

[2] L. Bahl, P. de Souza, P. Gopalakrishnan, D. Nahamoo, and M. Picheny. Context Dependent Modeling of Phones in Continuous Speech Using Decision Trees. In *Proc DARPA Speech and Natural Language Processing Workshop*, pages 264–270, Pacific Grove, Calif, Feb. 1991.

[3] J. Baker. The Dragon System - an Overview. *IEEE Trans ASSP*, 23(1):24–29, 1975.

[4] L. Baum. An Inequality and Associated Maximisation Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. *Inequalities*, 3:1–8, 1972.

[5] J. Bellegarda and D. Nahamoo. Tied Mixture Continuous Parameter Modeling for Speech Recognition. *IEEE Trans ASSP*, 38(12):2033–2045, 1990.

[6] S. Davis and P. Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans ASSP*, 28(4):357–366, 1980.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J Royal Statistical Society Series B*, 39:1–38, 1977.

[8] M. Gales. The Generation and Use of Regression Class Trees for MLLR adaptation. Technical Report CUED/F-INFENG/TR.263, Cambridge University Engineering Department, 1996.

[9] M. Gales. Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition. Technical Report CUED/F-INFENG/TR.291, Cambridge University Engineering Department, 1997.

[10] H. Hermansky. Perceptual Linear Predictive (PLP) Analysis of Speech. *J Acoustical Soc America*, 87(4):1738–1752, 1990.

[11] X. Huang and M. Jack. Semi-continuous hidden Markov models for Speech Signals. *Computer Speech and Language*, 3(3):239–252, 1989.

[12] M.-Y. Hwang and X. Huang. Shared Distribution Hidden Markov Models for Speech Recognition. *IEEE Trans Speech and Audio Processing*, 1(4):414–420, 1993.

[13] F. Jelinek. Continuous Speech Recognition by Statistical Methods. *Proc IEEE*, 64(4):532–556, 1976.

[14] A. Kannan, M. Ostendorf, and J. Rohlicek. Maximum Likelihood Clustering of Gaussians for Speech Recognition. *IEEE Trans on Speech and Audio Processing*, 2(3):453–455, 1994.

[15] C. Leggetter and P. Woodland. Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models. *Computer Speech and Language*, 9(2):171–185, 1995.

[16] Y. Normandin. *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem*. PhD thesis, Dept of Elect Eng McGill University, Mar. 1991.

[17] J. Odell, V. Valtchev, P. Woodland, and S. Young. A One-Pass Decoder Design for Large Vocabulary Recognition. In *Proc Human Language Technology Workshop*, pages 405–410, Plainsboro NJ, Morgan Kaufman Publishers Inc, Mar. 1994.

[18] D. Pallett, J. Fiscus, and Przybocki. 1996 Preliminary Broadcast News Benchmark Tests. In *Proc DARPA Speech Recognition Workshop*, pages 22–46, Chantilly, Virginia, Feb. 1997. Morgan Kaufmann.

[19] V. Valtchev, P. Woodland, and S. Young. Lattice-based Discriminative Training for Large Vocabulary Speech Recognition. In *Proc ICASSP*, volume 2, pages 605–608, Atlanta, May 1996.

[20] P. Woodland, M. Gales, D. Pye, and S. Young. Broadcast News Transcription using HTK. In *Proc ICASSP*, volume 2, pages 719–722, Munich, Germany, 1997.

[21] P. Woodland, M. Gales, D. Pye, and S. Young. The Development of the 1996 HTK Broadcast News Transcription System. In *Proc DARPA Speech Recognition Workshop*, pages 73–78, Chantilly, Virginia, Feb. 1997. Morgan Kaufmann.

[22] P. Woodland, C. Leggetter, J. Odell, V. Valtchev, and S. Young. The 1994 HTK Large Vocabulary Speech Recognition System. In *Proc ICASSP*, volume 1, pages 73–76, Detroit, 1995.

[23] S. Young, J. Odell, and P. Woodland. Tree-Based State Tying for High Accuracy Acoustic Modelling. In *Proc Human Language Technology Workshop*, pages 307–312, Plainsboro NJ, Morgan Kaufman Publishers Inc, Mar. 1994.

[24] S. Young and P. Woodland. State Clustering in HMM-based Continuous Speech Recognition. *Computer Speech and Language*, 8(4):369–384, 1994.