HMMS AND RELATED SPEECH RECOGNITION TECHNOLOGIES

Steve Young

Cambridge University Engineering Department
Trumpington Street, Cambridge, CB2 1PZ, United Kingdom
e-mail: sjy@eng.cam.ac.uk

Almost all present day continuous speech recognition (CSR) systems are based on Hidden Markov Models (HMMs). Although the fundamentals of HMM-based CSR have been understood for several decades, there has been steady progress in refining the technology both in terms of reducing the impact of the inherent assumptions, and in adapting the models for specific applications and environments. The aim of this chapter is to review the core architecture of a HMM-based CSR system and then outline the major areas of refinement incorporated into modern-day systems.

1. INTRODUCTION

Automatic continuous speech recognition (CSR) is sufficiently mature that a variety of real world applications are now possible including command and control, dictation, transcription of recorded speech and interactive spoken dialogues. This chapter describes the statistical models that underlie current-day systems: specifically, the hidden Markov model (HMM) and its related technologies.

The foundations of modern HMM-based continuous speech recognition technology were laid down in the 1970's by groups at Carnegie-Mellon, IBM and Bell Labs [1, 2, 3]. Reflecting the computational power of the time, initial development in the 1980's focussed on whole word small vocabulary applications[4, 5]. In the early 90's, attention switched to continuous speaker-independent recognition. Starting with the artificial 1000 word *Resource Management* task [6], the technology developed rapidly and by the mid-1990's, reasonable accuracy was being achieved for unrestricted dictation. Much of this development was driven by a series of DARPA and NSA programmes[7] which set ever more challenging tasks culminating most recently in systems for multilingual transcription of broadcast news programmes[8] and for spontaneous telephone conversations[9].

Although the basic framework for CSR has not changed significantly in the last ten years, the detailed modelling techniques developed within this framework have evolved to a state of considerable sophistication (e.g. [10, 11]). The result has been steady and significant progress and it is the aim of this chapter to describe the main techniques by which this has been achieved. Many research groups have contributed to this progress, and each will typically have their own architectural perspective. For the sake of logical coherence, the presentation given here is somewhat biassed towards the architecture developed at Cambridge and supported by the HTK Software Toolkit[12][1].

The chapter is organised as follows. In section 2, the core architecture of a typical HMM-based recogniser is described [13]. Subsequent sections then describe the various improvements which have been made to this core over recent years. Section 3 discusses methods of HMM parameter estimation and issues relating to improved covariance modelling. In sections 4 and 5, methods of normalisation and adaptation are described which allow HMM-based acoustic models to more accurately represent specific speakers and environments. Finally in section 6, the multi-pass architecture refinements adopted by modern transcription system is described. The chapter concludes in section 7 with some general observa-

[1]Available for free download at `htk.eng.cam.ac.uk`

tions and conclusions.

2. ARCHITECTURE OF A HMM-BASED RECOGNISER

The principal components of a large vocabulary continuous speech recogniser are illustrated in Fig 1. The input audio waveform from a microphone is converted into a sequence of fixed size acoustic vectors $\boldsymbol{Y} = \boldsymbol{y}_1..\boldsymbol{y}_T$ in a process called feature extraction. The decoder then attempts to find the sequence of words $W = w_1 \dots w_K$ which is most likely to have generated $\boldsymbol{Y}$, i.e. the decoder tries to find

$$\hat{W} = \underset{W}{\operatorname{argmax}}\{p(W|\boldsymbol{Y})\} \tag{1}$$

However, since $p(W|\boldsymbol{Y})$ is difficult to model directly, Bayes' Rule is used to transform (1) into the equivalent problem of finding:

$$\hat{W} = \underset{W}{\operatorname{argmax}}\{p(\boldsymbol{Y}|W)p(W)\}. \tag{2}$$

The likelihood $p(\boldsymbol{Y}|W)$ is determined by an *acoustic model* and the prior $p(W)$ is determined by a *language model*.[2] The basic unit of sound represented by the acoustic model is the *phone*. For example, the word "bat" is composed of three phones /b/ /ae/ /t/. About 40 such phones are required for English.

For any given W, the corresponding acoustic model is synthesised by concatenating phone models to make words as defined by a pronunciation dictionary. The parameters of these phone models are estimated from training data consisting of speech waveforms and their orthographic transcriptions. The language model is typically an N-gram model in which the probability of each word is conditioned only on its $N-1$ predecessors. The N-gram parameters are estimated by counting N-tuples in appropriate text corpora. The decoder operates by searching through all possible word sequences using pruning to remove unlikely hypotheses thereby keeping the search tractable. When the end of the utterance is reached, the most likely word sequence is output. Alternatively, modern decoders can generate lattices containing a compact representation of the most likely hypotheses.

The following sections describe these processes and components in more detail.

2.1. Feature Extraction

The feature extraction stage seeks to provide a compact encoding of the speech waveform. This encoding should minimise the information loss and provide a good match with the distributional assumptions made by the acoustic models. Feature vectors are typically computed every 10ms using an overlapping analysis window of around 25ms. One of the simplest and most widely used encoding schemes uses *Mel-Frequency Cepstral Coefficients (MFCCs)*[14]. These are generated by applying a truncated cosine transformation to a log spectral estimate computed by smoothing an FFT with around 20 frequency bins distributed non-linearly across the speech spectrum. The non-linear frequency scale used is called a *Mel Scale* and it approximates the response of the human ear. The cosine transform is applied in order to smooth the spectral estimate and decorrelate the feature elements.

Further psychoacoustic constraints are incorporated into a related encoding called *Perceptual Linear Prediction (PLP)* [15]. PLP computes linear prediction coefficients from a perceptually weighted non-linearly compressed power spectrum and then transforms the linear prediction coefficients to cepstral coefficients. In practice, PLP can give small improvements over MFCCs, especially in noisy environments and hence it is the preferred encoding for many systems.

In addition to the spectral coefficients, first order (delta) and second-order (delta-delta) regression coefficients are often appended in a heuristic attempt to compensate for the conditional independence assumption made by the HMM-based acoustic models. The final result is a feature vector whose dimensionality is typically around 40 and which has been partially but not fully decorrelated.

[2] In practice, the acoustic model is not normalised and the language model is often scaled by an empirically determined constant and a word insertion penalty is added i.e. in the log domain the total likelihood is calculated as $\log p(\boldsymbol{Y}|W) + \alpha p(W) + \beta|W|$ where α is typically in the range 8 to 20 and β is typically in the range 0 to -20.

2.2. HMM Acoustic Models

As noted above, the spoken words in W are decomposed into a sequence of basic sounds called *base phones*. To allow for possible pronunciation variation, the likelihood $p(\boldsymbol{Y}|W)$ can be computed over multiple pronunciations[3]

$$p(\boldsymbol{Y}|W) = \sum_{Q} p(\boldsymbol{Y}|Q)p(Q|W) \tag{3}$$

Each Q is a sequence of word pronunciations $Q_1 \ldots Q_K$ where each pronunciation is a sequence of base phones $Q_k = q_1^{(k)} q_2^{(k)} \ldots$. Then

$$p(Q|W) = \prod_{k=1}^{K} p(Q_k|w_k) \tag{4}$$

where $p(Q_k|w_k)$ is the probability that word w_k is pronounced by base phone sequence Q_k. In practice, there will only be a very small number of possible Q_k for each w_k making the summation in (3) easily tractable.

Each base phone q is represented by a continuous density hidden Markov model (HMM) of the form illustrated in Fig 2 with transition parameters $\{a_{ij}\}$ and output observation distributions $\{b_j()\}$. The latter are typically mixtures of Gaussians

$$b_j(\boldsymbol{y}) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}_{jm}, \Sigma_{jm}) \tag{5}$$

where $\mathcal{N}$ denotes a normal distribution with mean $\boldsymbol{\mu}_{jm}$ and covariance Σ_{jm}, and the number of components M is typically in the range 10 to 20. Since the dimensionality of the acoustic vectors $\boldsymbol{y}$ is relatively high, the covariances are usually constrained to be diagonal. The entry and exit states are *non-emitting* and they are included to simplify the process of concatenating phone models to make words.

Given the composite HMM Q formed by concatenating all of its constituent base phones then the acoustic likelihood is given by

$$p(\boldsymbol{Y}|Q) = \sum_{X} p(X, \boldsymbol{Y}|Q) \tag{6}$$

[3]Recognizers often approximate this by a *max* operation so that alternative pronunciations can be decoded as though they were alternative word hypotheses.

where $X = x(0)..x(T)$ is a state sequence through the composite model and

$$p(X, \boldsymbol{Y}|Q) = a_{x(0),x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{y}_t) a_{x(t),x(t+1)} \tag{7}$$

The acoustic model parameters $\{a_{ij}\}$ and $\{b_j()\}$ can be efficiently estimated from a corpus of training utterances using Expection-Maximisation (EM)[16]. For each utterance, the sequence of baseforms is found and the corresponding composite HMM constructed. A forward-backward alignment is used to compute state occupation probabilities (the E-step) and the means and covariances are then estimated via simple weighted averages (the M-step)[12, Ch 7]. This iterative process can be initialised by assigning the global mean and covariance of the data to all Gaussian components and setting all transitition probabilities to be equal. This gives a so-called *flat start* model. The number of component Gaussians in any mixture can easily be increased by cloning, perturbing the means and then re-estimating using EM.

This approach to acoustic modelling is often referred to as the *beads-on-a-string* model, so-called because all speech utterances are represented by concatenating a sequence of phone models together. The major problem with this is that decomposing each vocabulary word into a sequence of context-independent base phones fails to capture the very large degree of context-dependent variation that exists in real speech. For example, the base form pronunciations for “mood” and “cool” would use the same vowel for “oo”, yet in practice the realisations of “oo” in the two contexts are very different due to the influence of the preceding and following consonant. Context independent phone models are referred to as *monophones*.

A simple way to mitigate this problem is to use a unique phone model for every possible pair of left and right neighbours. The resulting models are called *triphones* and if there are N base phones, there are logically N^3 potential triphones. To avoid the resulting data sparsity problems, the complete set of *logical* triphones L can be mapped to a reduced set of physical models P by clustering and tying together the parameters in each cluster. This mapping process is illustrated in Fig 3 and the parameter tying is illustrated in Fig 4 where the notation x-q+y denotes the triphone corresponding to phone q spoken in the context of a preceding phone x and a following phone y. The base phone pronunciations Q are derived by simple look-up from the pronunciation dictionary, these are then mapped to logical phones according to the context, finally the logical phones are mapped to physical models. Notice that the context-dependence spreads across word boundaries and this is essential for capturing many important phonological processes. For example, the [p] in "stop that" has its burst suppressed by the following consonant.

The clustering of logical to physical models typically operates at the state-level rather than the model level since it is simpler and it allows a larger set of physical models to be robustly estimated. The choice of which states to tie is made using decision trees[17]. Each state position[4] of each phone q has a binary tree associated with it. Each node of the tree carries a question regarding the context. To cluster state i of phone q, all states i of all of the logical models derived from q are collected into a single pool at the root node of the tree. Depending on the answer to the question at each node, the pool of states is successively split until all states have trickled down to leaf nodes. All states in each leaf node are then tied to form a physical model. The questions at each node are selected from a predetermined set to maximize the likelihood of the training data given the final set of state-tyings. If the state output distributions are single component Gaussians and the state occupation counts are known, then the increase in likelihood achieved by splitting the Gaussians at any node can be calculated simply from the counts and model parameters without reference to the training data. Thus, the decision trees can be grown very efficiently using a greedy iterative node splitting algorithm. Fig 5 illustrates this tree-based clustering. In the figure, the logical phones s-aw+n and t-aw+n will both be assigned to leaf node 3 and hence they will share the same central state of the representative physical model.[5]

The partitioning of states using phonetically-driven decision trees has several advantages. In particular, logical models which are required but were not seen at all in the training data can be easily synthesised. One disadvantage is that the partitioning can be rather coarse. This problem can be reduced using so-called *soft-tying*[18]. In this scheme, a post-processing stage groups each state with its one or two nearest neighbours and pools all of their Gaussians. Thus, the single Gaussian models are converted to mixture Gaussian models whilst holding the total number of Gaussians in the system constant.

To summarise, the core acoustic models of a modern speech recogniser is typically comprised of a set of tied-state mixture Gaussian HMM-based acoustic models. This core is commonly built in the following steps[12, Ch 3]:

1. A flat-start monophone set is created in which each base phone is a monophone single-Gaussian HMM with means and covariances equal to the mean and covariance of the training data.

2. The parameters of the single-Gaussian monophones are iteratively re-estimated using 3 or 4 iterations of EM.

3. Each single Gaussian monophone q is cloned once for each distinct triphone x-q+y that appears in the training data.

4. The set of training data single-Gaussian triphones is iteratively re-estimated using EM and the state occupation counts of the last iteration are saved.

5. A decision tree is created for each state in each base phone, the single-Gaussian triphones are mapped into a smaller set of tied-state triphones and iteratively re-estimated using EM.

6. Mixture components are iteratively split and re-estimated until performance peaks on a held-out development set.

The final result is the required tied-state context-dependent mixture Gaussian acoustic model set.

[4]invariably each phone model has three states

[5]The total number of tied-states in a large vocabulary speaker independent system typical ranges between 1000 and 5000 states

2.3. N-gram Language Models

The prior probability of a word sequence $W = w_1..w_K$ required in (2) is given by

$$p(W) = \prod_{k=1}^{K} p(w_k|w_{k-1},\ldots,w_1) \qquad (8)$$

For large vocabulary recognition, the conditioning word history in (8) is usually truncated to $N-1$ words to form an N-*gram* language model

$$p(W) = \prod_{k=1}^{K} p(w_k|w_{k-1},w_{k-2},\ldots,w_{k-N+1}) \quad (9)$$

where N is typically in the range 2 to 4. The N-gram probabilities are estimated from training texts by counting N-gram occurrences to form maximum likelihood (ML) parameter estimates. For example, let $C(w_{k-2}w_{k-1}w_k)$ represent the number of occurrences of the three words $w_{k-2}w_{k-1}w_k$ and similarly for $C(w_{k-2}w_{k-1})$, then

$$p(w_k|w_{k-1},w_{k-2}) \approx \frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})} \qquad (10)$$

The major problem with this simple ML estimation scheme is data sparsity. This can be mitigated by a combination of discounting and backing-off, known as *Katz smoothing*[19]

$$\begin{aligned} p(w_k&|w_{k-1},w_{k-2}) \\ &= \frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})} && \text{if } C > C' \\ &= d\frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})} && \text{if } 0 < C \le C' \\ &= \alpha(w_{k-1},w_{k-2})\, p(w_k|w_{k-1}) && \text{otherwise} \quad (11) \end{aligned}$$

where C' is a count threshold, d is a discount coefficient and α is a normalisation constant. Thus, when the N-gram count exceeds some threshold, the ML estimate is used. When the count is small the same ML estimate is used but discounted slightly. The discounted probability mass is then distributed to the unseen N-grams which are approximated by a weighted version of the corresponding bigram. This idea can be applied recursively to estimate any sparse N-gram in terms of a set of back-off weights and $N-1$-grams. The discounting coefficient is based on the Turing-Good estimate $d = (r+1)n_{r+1}/rn_r$ where n_r is the number of N-grams that occur exactly r times in the training data. Although Katz smoothing is effective, there are now known to be variations which work better. In particular, Kneser-Ney smoothing consistently outperforms Katz on most tasks [20, 21].

An alternative approach to robust language model estimation is to use class-based models in which for every word w_k there is a corresponding class c_k [22, 23]. Then,

$$p(W) = \prod_{k=1}^{K} p(w_k|c_k)p(c_k|c_{k-1},\ldots,c_{k-N+1}) \qquad (12)$$

As for word based models, the class N-gram probabilities are estimated using ML but since there are far fewer classes (typically a few hundred) data sparsity is much less of an issue. The classes themselves are chosen to optimise the likelihood of the training set assuming a bigram class model. It can be shown that when a word is moved from one class to another, the change in perplexity depends only on the counts of a relatively small number of bigrams. Hence, an iterative algorithm can be implemented which repeatedly scans through the vocabulary, testing each word to see if moving it to some other class would increase the likelihood [24].

In practice it is found that for reasonably sized training sets, an effective language model for large vocabulary applications consists of a word-based trigram or 4-gram interpolated with a class-based trigram.

2.4. Decoding and Lattice Generation

As noted in the introduction to this section, the most likely word sequence $\hat{W}$ given a sequence of feature vectors $\boldsymbol{Y} = \boldsymbol{y}_1..\boldsymbol{y}_T$ is found by searching all possible state sequences arising from all possible word sequences for the sequence which was most likely to have generated the observed data $\boldsymbol{Y}$. An efficient way to solve this problem is to use dynamic programming. Let $\phi_j(t) = \max_X \{p(\mathbf{y}_1,\ldots,\mathbf{y}_t, x(t)=j|\mathcal{M})\}$ i.e. the maximum probability of observing the partial sequence $\boldsymbol{y}_1..\boldsymbol{y}_t$ and then being in state j at time t given the model $\mathcal{M}$. This probability can be efficiently computed using the Viterbi algorithm

$$\phi_j(t) = \max_i \{\phi_i(t-1)a_{ij}\} \, b_j(\mathbf{y}_t) \qquad (13)$$

It is initialised by setting $\phi_j(t)$ to 1 for the initial state and 0 for all other states. The probability of the most likely word sequence is then given by $\max_j\{\phi_j(T)\}$ and if every maximisation decision is recorded, a traceback will yield the required best matching state/word sequence.

In practice, a direct implementation of the above algorithm becomes unmanageably complex for continuous speech where the topology of the models, the language model constraints and the need to bound the computation must all be taken account of. Fortunately, much of this complexity can be abstracted away by a simple change in viewpoint.

First, the HMM topology can be made explicit by constructing a recognition network. For task oriented applications, this network can represent the utterances that the user is allowed to say i.e. it can represent a recognition grammar. For large vocabulary applications, it will typically consist of all vocabulary words in parallel in a loop. In both cases, words are represented by a sequence of phone models as defined by the pronunciation dictionary (see Fig 6), and each phone model consists of a sequence of states as indicated by the inset. If a word has several pronunciations as in the general case described by (3), they are simply placed in parallel.

Given this network, at any time t in the search, a single hypothesis consists of a path through the network representing an alignment of states with feature vectors $\boldsymbol{y}_1..\boldsymbol{y}_t$, starting in the initial state and ending at state j, and having a log likelihood $\log \phi_j(t)$. This path can be made explicit via the notion of a *token* consisting of a pair of values $< logP, link >$ where $logP$ is the log likelihood[6] and $link$ is a pointer to a record of history information [25]. Each network node corresponding to each HMM state can store a single token and recognition proceeds by propagating these tokens around the network.

The basic Viterbi algorithm given above can now be recast for continuous speech recognition as the *Token Passing* algorithm shown in outline in Fig 7. The term *node* refers to a network node corresponding to a single HMM state. These nodes correspond to either entry states, emitting states or exit states. Essentially, tokens are passed from node to node and at each transition the token score is updated.

[6]Often referred to as the *score*

When a token transitions from the exit of a word to the start of the next word, its score is updated by the language model probability plus any word insertion penalty. At the same time the transition is recorded in a record R containing a copy of the token, the current time and identity of the preceding word. The $link$ field of the token is then updated to point to the record R. As each token proceeds through the network it accumulates a chain of these records. The best token at time T in a valid network exit node can then be examined and traced back to recover the most likely word sequence and the boundary times.

The above Token Passing algorithm and associated recognition network is an exact implementation of the dynamic programming principle embodied in (13). To convert this to a practical decoder for speech recognition, the following steps are required:

1. For computational efficiency, only tokens which have some likelihood of being on the best path should be propagated. Thus, every propagation cycle, the log probability of the most likely token is recorded. All tokens whose probabilities fall more than a constant below this are deleted. This results in a so-called *beam search* and the constant is called the *beam width*.

2. As a consequence of beam search, 90% of the computation is actually spent on the first two phones of every word, after that most of the tokens fall outside of the beam and are pruned. To exploit this, the recognition network should be *tree-structured* so that word initial phones are shared (see Fig 8).

3. However, sharing initial phones makes it impossible to apply an exact language model probability during word-external token propagation since the identity of the following word is not known. Simply delaying the application of the language model until the end of the following word is not an option since that would make pruning ineffective. Instead, an incremental approach must be adopted in which the language model probability is taken to be the maximum possible probability given the set of possible following words. As tokens move through the tree-structured word

graph, the set of possible next words reduces at each phone transition and the language model probability can be updated with a more accurate estimate.

4. The HMMs linked into the recognition network should be context dependent and for best performance, this dependency should span word boundaries. At first sight this requires a massive expansion of the network, but in fact a compact static network representation of cross-word triphones is possible [26].

5. The dynamic programming principle relies on the principle that the optimal path at any node can be extended knowing only the state information given at that node. The use of N-gram language models causes a problem here since unique network nodes would be needed to distinguish all possible $N-1$ word histories and for large vocabulary decoders this is not tractable. Thus, the algorithm given in Fig 7 will only work for bigram language models. A simple way to solve this problem is to store multiple tokens in each state thereby allowing paths with differing histories to "stay alive" in parallel. Token propagation now requires a merge and sort operation which although computationally expensive can be made tractable.

The above description of a large vocabulary decoder covers all of the essential elements needed to recognise continuous speech in real time using just a single pass over the data. For off-line batch transcription of speech, significant improvements in accuracy can be achieved by performing multiple passes over the data. To make this possible, the decoder must be capable of generating and saving multiple recognition hypotheses. A compact and efficient structure for doing this is the *word lattice*[27, 28, 29].

A word lattice consists of a set of nodes representing points in time and a set of spanning arcs representing word hypotheses. An example is shown in Fig 9 part (a). In addition to the word ids shown in the figure, each arc can also carry score information such as the acoustic and language model scores. Lattices are generated via the mechanism for recording word boundary information outlined in Fig 7, except that instead of recording just the best token which is actually propagated to following word entry nodes, all word-end tokens are recorded. For the simple single-token Viterbi scheme, the quality of lattices generated in this way will be poor because many of the close matching second best paths will have been pruned by exercising the dynamic programming principle. The multiple token decoder does not suffer from this problem, especially if it is used with a relatively short-span bigram language model.

Lattices are extremely flexible. For example, they can be rescored by using them as an input recognition network and they can be expanded to allow rescoring by a higher order language model. They can also be compacted into a very efficient representation called a *confusion network*[30, 31]. This is illustrated in Fig 9 part (b) where the "-" arc labels indicate null transitions. In a confusion network, the nodes no longer correspond to discrete points in time, instead they simply enforce word sequence constraints. Thus, parallel arcs in the confusion network do not necessarily correspond to the same acoustic segment. However, it is assumed that most of the time the overlap is sufficient to enable parallel arcs to be regarded as competing hypotheses. A confusion network has the property that for every path through the original lattice, there exists a corresponding path through the confusion network. Each arc in the confusion network carries the posterior probability of the corresponding word w. This is computed by finding the *link probability* of w in the lattice using a forward-backward procedure, summing over all occurrences of w and then normalising so that all competing word arcs in the confusion network sum to one. Confusion networks can be used for minimum word-error decoding, to provide confidence scores and for merging the outputs of different decoders [32, 33, 34, 35] (see section 6).

Finally, it should be noted that all of the above relates to one specific approach to decoding. If simple Viterbi decoding was the only requirement, then there would be little variation amongst decoder implementations. However, the requirement to support cross-word context-dependent acoustic models and long span language models has led to a variety of design strategies. For example, rather than have multiple tokens, the network state can be dynamically expanded to explicitly represent the currently hypothesised cross-word acoustic and long-span language model contexts [36, 37]. These dynamic network decoders are more flexible than static network decoders, but they are harder to implement efficiently. Recent advances in weighted finite-state transducer technology offer the possibility of integrating all of the required information (acoustic models, pronunciation, language model probabilities, etc) into a single very large but highly optimised network) [38]. This approach offers both flexibility and efficiency and is therefore extremely useful for both research and practical applications.

A completely different approach to decoding is to avoid the breadth-first strategy altogether and use a depth-first strategy. This gives rise to a class of recognisers called *stack decoders*. Stack decoders were popular in the very early developments of ASR since they can be very efficient. However, they require dynamic networks and their run-time search characteristcs can be difficult to control [39, 40].

3. HMM-BASED ACOUSTIC MODELLING

The key feature of the HMM-based speech recognition architecture described in the preceding section is the use of diagonal covariance multiple component mixture Gaussians for modelling the spectral distributions of the speech feature vectors. If speech really did have the statistics assumed by these HMMs and if there was sufficient training data, then the models estimated using maximum likelihood would be optimal in the sense of minimum variance and zero bias [41]. However, since this is not the case, there is scope for improving performance both by using alternative parameter estimation schemes and by improving the models. In this section, both of these aspects of HMM design will be discussed. Firstly, discriminative training is described and then methods of improved covariance modelling will be explored.

3.1. Discriminative Training

Standard maximum likelihood training attempts to find a parameter set λ which maximises the log likelihood of the training data, i.e. for training sentences $\boldsymbol{Y}_1 \ldots \boldsymbol{Y}_R$, the objective function is

$$\mathcal{F}_{\text{ML}}(\lambda) = \sum_{r=1}^{R} \log p_\lambda(\boldsymbol{Y}_r|\mathcal{M}_{W_r}) \qquad (14)$$

where W_r is the word sequence given by the transcription of the r'th training sentence and $\mathcal{M}_{W_r}$ is the corresponding composite HMM synthesised by concatenating phone models (denoted by Q in section 2.2). This objective function can be maximised straightforwardly using a version of EM known as the Baum-Welch algorithm [42]. This involves iteratively finding the probability of state-component occupation for each frame of training data using a forward-backward algorithm, and then computing weighted averages. For example, defining the following "counts"

$$\Theta_{jm}^r(\mathcal{M}) = \sum_{t=1}^{T_r} \gamma_{jm}^r(t)\boldsymbol{y}_t^r \bigg|_{\mathcal{M}} \qquad (15)$$

and

$$\Gamma_{jm}^r(\mathcal{M}) = \sum_{t=1}^{T_r} \gamma_{jm}^r(t) \bigg|_{\mathcal{M}} \qquad (16)$$

where $\gamma_{jm}^r(t)$ is the probability of the model occupying mixture component m in state j at time t given training sentence $\boldsymbol{Y}_r$ and model $\mathcal{M}$, then the updated mean estimate is given by ML as

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{r=1}^R \Theta_{jm}^r(\mathcal{M}_{W_r})}{\sum_{r=1}^R \Gamma_{jm}^r(\mathcal{M}_{W_r})} \qquad (17)$$

i.e. the average of the sum of the data weighted by the model component occupancy.

The key problem with the ML objective function is that it simply fits the model to the training data and it takes no account of the model's ability to discriminate. An alternative objective function is to maximise the conditional likelihood using the Maximum Mutual Information (MMI) criterion [41, 43]

$$\mathcal{F}_{\text{MMI}}(\lambda) = \sum_{r=1}^{R} \log \frac{p_\lambda(\boldsymbol{Y}_r|\mathcal{M}_{W_r})p(W_r))}{\sum_W p_\lambda(\boldsymbol{Y}_r|\mathcal{M}_W)p(W))} \qquad (18)$$

Here the numerator is the likelihood of the data given the correct word sequence W_r whilst the denominator is the total likelihood of the data given all possible word sequences W. Thus, the objective function is maximised by making the correct model sequence likely and all other model sequences unlikely. It is therefore discriminative. Note also that it incorporates the effect of the language model and hence more closely represents recognition conditions.

There is no simple EM-based re-estimation scheme for (18), however, there is an approximate scheme known as the Extended Baum-Welch algorithm in which stability is achieved for the parameter updates by adding a constant D to both the numerator and denominator. For example, (17) in the extended scheme becomes

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{r=1}^{R}\{\Theta_{jm}^{r}(\mathcal{M}_{\text{num}}) - \Theta_{jm}^{r}(\mathcal{M}_{\text{den}})\} + D\boldsymbol{\mu}_{jm}}{\sum_{r=1}^{R}\{\Gamma_{jm}^{r}(\mathcal{M}_{\text{num}}) - \Gamma_{jm}^{r}(\mathcal{M}_{\text{den}})\} + D} \tag{19}$$

where $\mathcal{M}_{\text{num}}$ is the combined acoustic and language model used to compute the numerator of (18), and $\mathcal{M}_{\text{den}}$ is the combined acoustic and language model used to compute the denominator. In the latter case, it is understood that the counts in $\Theta_{jm}^{r}(\mathcal{M}_{\text{den}})$ are summed over all word sequences. For large vocabulary continuous speech, this is approximated by computing lattices and summing over all lattice arcs. Although the numerator counts can be computed as in ML, in practice, the numerator counts are also computed using lattices since this provides a convenient way to take account of multiple pronunciations[44]. As can be seen, counts in the numerator are reduced if there are similar confusing counts in the denominator. The constant D acts like an interpolation weight between the new estimate and the existing estimate.

In a little more detail, the MMI training process is as follows. Numerator and denominator lattices are generated for every training utterance using $\mathcal{M}_{\text{num}}$ and $\mathcal{M}_{\text{den}}$, respectively. $\mathcal{M}_{\text{num}}$ comprises the current phone models integrated into a graph of alternative word pronunciations, and $\mathcal{M}_{\text{den}}$ comprises the normal recogniser set-up with two exceptions. Firstly, a weaker language model is used. Typically the lattices are generated using a bigram language model and then rescored using a unigram [45]. Secondly, the likelihoods of the acoustic models are scaled by the inverse of the normal LM scale factor [46]. Both of these modifications have been found to increase the number of confusions in the denominator lattice, thereby improving subsequent generalisation. Once the word level lattices have been generated, a Viterbi decode is performed on each lattice arc to obtain a phone-level segmentation. Forward-backward is then applied to obtain the component level counts and the model parameters re-estimated using (19) (and similar formulae for the variances and mixture weights). This process is iterated and the state-component alignments are recomputed every 3 or 4 iterations. The word-level lattices are typically held fixed throughout the training process.

The constant D must be chosen to be large enough to ensure convergence but small enough to ensure acceptably fast training. In practice, D is chosen to ensure that variance updates are positive and it is normally set specific to each phone or Gaussian [46].

MMI training can provide consistent performance improvements compared to similar systems trained with ML [46]. However, it can be argued that it places too much emphasis on training utterances which have a low posterior probability and not enough weight on training utterances near the decision boundary as in for example Minimum Classification Error (MCE) training [47]. MCE, however, focuses on overall sentence-level accuracy, and it is not appropriate for lattice-based training of large systems. Minimum Phone Error (MPE) training addresses this issue by attempting to maximise the posterior utterance probability scaled by the *Raw Phone Accuracy* (RPA) [48]

$$\mathcal{F}_{\text{MPE}}(\lambda) = \sum_{r=1}^{R} \frac{\sum_{W} p_{\lambda}(\boldsymbol{Y}_r|\mathcal{M}_W)p(W)\text{RPA}(W,W_r)}{\sum_{W} p_{\lambda}(\boldsymbol{Y}_r|\mathcal{M}_W)p(W)} \tag{20}$$

where as in lattice-based MMI training, the sums over W are taken over all word sequences appearing in the lattice generated by $\mathcal{M}_{\text{den}}$. The RPA is a measure of the number of phones accurately transcribed in each word string hypothesis W. Given the times of the phone boundaries, each phone in W is matched against the corresponding time segment in the transcription W_r. If the phones are the same, the RPA is incremented by the percentage overlap, otherwise it is decremented by the percentage overlap. Parameter optimisation is similar to the MMI process described above except that the counts are computed only on the denominator lattice. The numerator lattice provides only the transcriptions needed for determining the RPA. Essentially, the counts are composed from the occupation probabilities scaled by the RPA. If they are positive, they contribute to the numerator terms in the update equations, and if they are negative, they contribute to the denominator terms (see [48] for details).

The generalisation capabilities of discriminatively trained systems can be improved by interpolating with ML. For example, the H-criterion interpolates objective functions: $\alpha\mathcal{F}_{\text{MMI}} + (1-\alpha)\mathcal{F}_{\text{ML}}$ [49]. However, choosing an optimal value for α is difficult and the effectiveness of the technique decreases with increasing training data [50]. A more effective technique is *I-smoothing* which increases the weight of the numerator counts depending on the amount of data available for each Gaussian component [48]. This is done by scaling the numerator counts $\Gamma^r_{jm}(\mathcal{M}_{\text{num}})$ and $\Theta^r_{jm}(\mathcal{M}_{\text{num}})$ by

$$1 + \frac{\tau}{\sum_r \Gamma^r_{jm}(\mathcal{M}_{\text{num}})} \tag{21}$$

where τ is a constant (typically about 50). As τ increases from zero, more weight is given to ML.

3.2. Covariance Modelling

An alternative way of improving the acoustic models is to allow them to more closely match the true distribution of the data. The baseline acoustic models outlined in section 2.2 use mixtures of diagonal covariance Gaussians chosen as a compromise between complexity and modelling accuracy. Nevertheless, the data is clearly not diagonal and hence finding some way of improving the covariance modelling is desirable. In general the use of full covariance Gaussians in large vocabulary systems would be impractical due to the sheer size of the model set [7]. However, the use of shared or *structured* covariance representations allow covariance modelling to be improved with very little overhead in terms of memory and computational load.

One the simplest and most effective structured covariance representations is the Semi-Tied Covariance (STC) matrix [51]. STC models the covariance of the m'th Gaussian component as

$$\hat{\Sigma}_m = A^{-1}\Sigma_m(A^{-1})^T \tag{22}$$

where Σ_m is the component specific diagonal covariance and A is the STC matrix shared by all components. If the component mean $\boldsymbol{\mu}_m = A\hat{\boldsymbol{\mu}}_m$ then component likelihoods can be computed by

$$\mathcal{N}(\boldsymbol{y};\hat{\boldsymbol{\mu}}_m,\hat{\Sigma}_m) = |A|\mathcal{N}(A\boldsymbol{y};\boldsymbol{\mu}_m,\Sigma_m) \tag{23}$$

Hence, a single STC matrix can be viewed as a linear transform applied in feature space. The component parameters $\boldsymbol{\mu}_m$ and Σ_m represent the means and variances in the transformed space and can be estimated in the normal way by simply transforming the training data. The STC matrix itself can be efficiently estimated using a row by row iterative scheme. Furthermore it is not necessary during training to store the full covariance statistics at the component level. Instead, an interleaved scheme can be used in which the A matrix statistics are updated on one pass through the training data, and then the component parameters are estimated on the next pass [51][8]. This can be integrated into the *mixing-up* operation described in section 2.2. For example, a typical training scheme might start with a single Gaussian system and an identity A matrix. The system is then iteratively refined by reestimating the component parameters, updating the A matrix, and mixing-up until the required number of Gaussians per state is achieved. As well as having a single global A matrix, the Gaussian components can be clustered and assigned one A matrix per cluster. For example, there could be one A matrix per phone or per state depending on the amount of training data available and the acceptable number of parameters. Overall, the use of STC can be expected to reduce word error rates by around 5% to 10% compared to the baseline system. In addition to STC, other types of structured covariance modelling include factor-analysed HMMs [52], sub-space constrained precision and means (SPAM) [53], and EMLLT [54].

It can be shown [51] that simultaneous optimisation of the full set of STC parameters (i.e. $\{A, \boldsymbol{\mu}_m, \Sigma_m\}$) is equivalent to maximising the auxiliary equation

$$\mathcal{Q}_{\text{STC}} = \sum_{t,m} \gamma_m(t) \log\left(\frac{|A|^2}{|\text{diag}(A\,W^{(m)}A^T)|}\right) \tag{24}$$

where

$$W^{(m)} = \frac{\sum_t \gamma_m(t)\bar{\boldsymbol{y}}_m(t)\bar{\boldsymbol{y}}_m(t)^T}{\sum_t \gamma_m(t)} \tag{25}$$

and where $\bar{\boldsymbol{y}}_m(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{\mu}}_m$. If each Gaussian component is regarded as a class, then $W^{(m)}$ is

[7]Although given enough data it can be done [11]

[8]The means can in fact be updated on both passes.

the within class covariance and it can be shown that (24) is the maximum likelihood solution to a generalised form of linear discriminant analysis called Heteroscedastic LDA (HLDA) in which class covariances are not constrained to be equal [55]. The matrix A can therefore be regarded as a feature space transform which discriminates Gaussian components and this suggests a simple extension by which A can also perform a subspace projection, i.e.

$$\hat{\boldsymbol{y}} = A\boldsymbol{y} = \left[\begin{array}{c} A_{[p]}\boldsymbol{y} \\ A_{[d-p]}\boldsymbol{y} \end{array} \right] = \left[\begin{array}{c} \hat{\boldsymbol{y}}_{[p]} \\ \hat{\boldsymbol{y}}_{[d-p]} \end{array} \right] \quad (26)$$

where the d-dimensional feature space is divided into p *useful* dimensions and $d-p$ *nuisance* dimensions. The matrix $A_{[p]}$ projects $\boldsymbol{y}$ into a p-dimensional subspace which is modelled by the diagonal Gaussian mixture components of the acoustic models. The $d-p$ nuisance dimensions are modelled by a global non-discriminating Gaussian. Equation 24 can therefore be factored as

$$\mathcal{Q}_{\text{HLDA}} = \sum_{t,m} \gamma_m(t) \log\left(\frac{|A|^2}{|\text{diag}(A_{[p]}\ W^{(m)} A_{[p]}^T)||\text{diag}(A_{[d-p]}\ T A_{[d-p]}^T)|} \right) \quad (27)$$

where T is the global covariance of the training data. The forms of equations (24) and (27) are similar and the optimal value for $A_{[p]}$ can be estimated by the same row-by-row iteration used in the STC case.

For application to LVCSR, $\boldsymbol{y}$ can be constructed either by concatenating successive feature vectors, or as is common in HTK-based systems, the standard 39-element feature vector comprised of static PLP coefficients plus their 1st and 2nd derivatives are augmented by the 3rd derivatives and then projected back into 39 dimensions using a 39×52 HLDA transform.

Finally note that, as with semi-tied covariances, multiple HLDA transforms can be used to allow the full acoustic space to be covered by a set of piecewise linear projections [56].

4. NORMALISATION

Normalisation attempts to condition the incoming speech signal in order to minimise the effects of variation caused by the environment and the physical characteristics of the speaker.

4.1. Mean and Variance Normalisation

Mean normalisation removes the average feature value and since most front-end feature sets are derived from log spectra, this has the effect of reducing sensitivity to channel variation. Cepstral variance normalisation scales each individual feature coefficient to have a unit variance and empirically this has been found to reduce sensitivity to additive noise[57].

For transcription applications where multiple passes over the data are possible, the necessary mean and variance statistics should be computed over the longest possible segment of speech for which the speaker and environment conditions are constant. For example, in broadcast news transcription this will be a speaker segment and in telephone transcription it will be a whole side of a conversation. Note that for real time systems which operate in a single continuous pass over the data, the mean and variance statistics must be computed as running averages.

4.2. Gaussianization

Given that normalising the first and second order statistics yields improved performance, an obvious extension is to normalise the higher order statistics so that the features are Gaussian distributed. This so-called *Gaussianization* is performed by finding a transform $z = \phi(y)$, on a per element basis, such that $p(z)$ is Gaussian. One simple way to achieve this is to estimate a cumulative distribution function (cdf) for each feature element y

$$F_0(y) = \frac{1}{N}\sum_{i=1}^{N} h(y - y_i) = rank(y_i)/N \quad (28)$$

where $y_1..y_N$ are the data to be normalised, $h(.)$ is the step function and $rank(y_i)$ is the rank of y_i when the data are sorted. The required transformation is then

$$z_i = \Phi^{-1}\left(\frac{rank(y_i)}{N}\right) \quad (29)$$

where $\Phi(\cdot)$ is the cdf of a Gaussian [58].

One difficulty with this approach is that when the normalisation data set is small the cdf estimate can be noisy. An alternative approach is to estimate an M component Gaussian mixture model (GMM) on the data and then use this to approximate the cdf[59], that is

$$z_i = \Phi^{-1}\left(\int_{-\infty}^{y_i} \sum_{m=1}^{M} c_m \mathcal{N}(y; \mu_m, \sigma_m^2) dy\right) \quad (30)$$

where μ_m and σ_m^2 are the mean and variance of the m'th GMM component. This results in a smoother and more compact representation of the Gaussianization transformation[60].

4.3. Vocal Tract Length Normalisation

Variations in vocal tract length cause formant frequencies to shift in frequency in an approximately linear fashion. Thus, one simple form of normalisation is to linearly scale the filter bank centre frequencies within the front-end feature extractor to approximate a canonical formant frequency scaling[61]. This is called *Vocal Tract Length Normalisation (VTLN)*.

To implement VTLN two issues need to be addressed: definition of the scaling function and estimation of the appropriate scaling function parameters for each speaker. Early attempts at VTLN used a simple linear mapping but as shown in Fig. 10(a) this results in a problem at high frequencies where female voices have no information in the upper frequency band and male voices have the upper frequency band truncated. This can be mitigated by using a piece-wise linear function of the form shown in Fig. 10(b)[57]. Alternatively, a bilinear transform can be used[62]. Parameter estimation is performed using a grid search plotting log likelihoods against parameter values. Once the optimal values for all training speakers have been computed, the training data is normalised and the acoustic models re-estimated. This is repeated until the VTLN parameters have stabilised. Note here that when comparing log likelihoods resulting from differing VTLN transformations, the Jacobean of the transform should strictly be included. This is however very complex to estimate and since the application of mean and variance normalisation will reduce the affect of this approximation, it is usually ignored.

For very large systems, the overhead incurred from iteratively computing the optimal VTLN parameters can be considerable. An alternative is to approximate the effect of VTLN by a linear transform. The advantage of this approach is that the optimal transformation parameters can be determined from the auxiliary function in a single pass over the data[63].

VTLN is particularly effective for telephone speech where speakers can be clearly identified. It is less effective for other applications such as broadcast news transcription where speaker changes must be inferred from the data.

5. ADAPTATION

A fundamental idea in statistical pattern classification is that the training data should adequately represent the test data, otherwise a mis-match will occur and recognition accuracy will be degraded. In the case of speech recognition, there will always be new speakers who are poorly represented by the training data, and new hitherto unseen environments. The solution to these problems is *adaptation*. Adaptation allows a small amount of data from a target speaker to be used to transform an acoustic model set to make it more closely match that speaker. It can be used both in training to make more specific and/or more compact recognition sets and it can be used in recognition to reduce mismatch and the consequent recognition errors.

There are varying styles of adapation which affect both the possible applications and the method of implementation. Firstly, adaptation can be *supervised* in which case accurate transcriptions are available for all of the adaptation data, or it can be *unsupervised* in which case the required transcriptions must be hypothesised. Secondly, adaptation can be *incremental* or *batch-mode*. In the former case, adaptation data becomes available in stages, for example, as is the case for a spoken dialogue system when a new caller comes on the line. In batch-mode, all of the adaptation data is available from the start as is the case in off-line transcription.

This section describes the main approaches to adaptation and its application in both recognition and training.

5.1. Maximum A Posteriori (MAP) Adaptation

Given some adaptation data $\boldsymbol{Y}_1..\boldsymbol{Y}_R$ and a model $\mathcal{M}$ with parameters λ, MAP-based parameter estimation seeks to maximise the following objective function,

$$\mathcal{F}_{\text{MAP}}(\lambda) = \sum_{r=1}^{R} \log p(\boldsymbol{Y}_r|\mathcal{M}_{W_r})p(\lambda) \tag{31}$$

Comparing this with the ML objection function given in (14), it can be seen that the likelihood is weighted by the prior. The choice of distribution for this prior is problematic since there is no conjugate prior density for a continuous Gaussian mixture HMM. However, if the mixture weights and Gaussian component parameters are assumed to be independent, then a finite mixture density of the form $p_D(\boldsymbol{c}_j)\prod_m p_W(\boldsymbol{\mu}_{jm}, \Sigma_{jm})$ can be used where $p_D(.)$ is a Dirichlet distribution over the vector of mixture weights $\boldsymbol{c}_j$ and $p_W(.)$ is a normal-Wishart density. It can then be shown that this leads to parameter estimation formulae of the form

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\tau\boldsymbol{\mu}_{jm} + \sum_{r=1}^{R}\Theta_{jm}^{r}(\mathcal{M}_{W_r})}{\tau + \sum_{r=1}^{R}\Gamma_{jm}^{r}(\mathcal{M}_{W_r})} \tag{32}$$

where $\boldsymbol{\mu}_{jm}$ is the prior mean and τ is a parameter of $p_W(.)$ which is normally determined empirically. Similar, though rather more complex, formulae can be derived for the variances and mixture weights [64]

Comparing (32) with (17), it can be seen that MAP adapation effectively interpolates the original prior parameter values with those that would be obtained from the adaptation data alone. As the amount of adaptation data increases, the parameters tend asymptotically to the adaptation domain. This is a desirable property and it makes MAP especially useful for porting a well-trained model set to a new domain for which there is only a limited amount of data.

A major drawback of MAP adaptation is that every Gaussian component is updated individually. If the adaptation data is sparse, then many of the model parameters will not be updated. Various attempts have been made to overcome this (e.g. [65, 66]) but MAP nevertheless remains ill-suited for rapid incremental adaptation.

5.2. ML-based Linear Transforms

An alternative approach to adaptation is to build a set of linear transforms to map an existing model set into a new adapted model set such that the likelihood of the adaptation data is maximised. This is called *Maximum Likelihood Linear Regression(MLLR)* and unlike MAP, it is particularly suited to unsupervised incremental adaptation.

There are two main variants of MLLR: unconstrained and constrained[67]. In unconstrained MLLR, separate transforms are trained for the means and variances,

$$\hat{\boldsymbol{\mu}}_{jm} = G\boldsymbol{\mu}_{jm} + \boldsymbol{b}; \qquad \hat{\Sigma}_{jm} = H\Sigma_{jm}H^T \tag{33}$$

The formulae for constrained MLLR (CMLLR) are identical except that $G = H$. The maximum likelihood estimation formulae are given in [68]. Whereas there are closed-form solutions for unconstrained MLLR, the constrained case is similar to the semi-tied covariance transform discussed in section 3.2 and requires an iterative solution. However, CMLLR has other advantages as discussed in section 5.3 below. Linear tranforms can also be estimated using discriminative criteria [69, 70, 71].

The key to the power of the MLLR adaptation approach is that a single transform matrix G can be shared across a set of Gaussian mixture components. When the amount of adaptation data is limited, a global 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 to use for any specific adaptation set can be determined automatically using a *regression class tree* as illustrated in fig 11. Each node represents a regression class i.e. a set of Gaussian components which will share a single transform. The total occupation count associated with any node in the tree can easily be computed since the counts are known at the leaf nodes. Then, for a given set of adaptation data, the tree is descended and the most specific set of nodes is selected for which there is sufficient data (for example, the shaded nodes in the figure).

When used in unsupervised mode, MLLR is normally applied iteratively[72]. First, the unknown test speech is recognised, then the hypothesised transcription is used to estimate MLLR transforms. The test speech is then re-recognised using the adapted models. This is repeated until convergence is achieved. A refinement of this is to use recognition lattices in place of the 1-best hypothesis to accumulate the adaptation statistics. This approach is more robust to recognition errors and avoids the need to re-recognise the data since the lattice can simply be rescored[73].

5.3. Adaptive Training

Ideally, an acoustic model set should encode just those dimensions which allow the different classes to be discriminated. However, in the case of speaker independent (SI) speech recognition, the training data necessarily includes a large number of speakers and hence acoustic models trained directly on this set will have to "waste" a large number of parameters encoding the variablity between speakers rather than the variability between spoken words which is the true aim.

One way to overcome this is to replace the single SI model set with a cluster of more specific models where each model can be trained on more homogenous data. This is called *Cluster Adaptive Training (CAT)*. At recognition time, a linear combination of models is selected where the set of interpolation weights, in effect, forms a speaker specific transform [74, 75, 76]. More recently discriminative techniques have been applied to CAT with some success [77].

An alternative approach to CAT is to use adaptation to transform each training set speaker to form a canonical model. This is called *Speaker Adaptive Training (SAT)* and the conceptual schema for this is shown in Fig. 12 [78]. When only mean transformations are used, SAT is straightforward. A transform is estimated for each speaker, and then the parameters of the canonical model set are estimated by modifying the statistics to account for the tranform. For example, to estimate the canonical model means, the counts in (15) and (16) are modified as follows:

$$\Theta^r_{jm}(\mathcal{M}) = \sum_{t=1}^{T_r} \gamma^r_{jm}(t) G^{(r)\,T} \Sigma^{-1}_{jm} (\boldsymbol{y}^r_t - \boldsymbol{b}^{(r)}) \Bigg|_{\mathcal{M}} \tag{34}$$

and

$$\Gamma^r_{jm}(\mathcal{M}) = \sum_{t=1}^{T_r} \gamma^r_{jm}(t) G^{(r)\,T} \Sigma^{-1}_{jm} G^{(r)} \Bigg|_{\mathcal{M}} \tag{35}$$

where $G^{(r)}, \boldsymbol{b}^{(r)}$ is the transform for the speaker uttering training sentence r. The mean is then estimated using (17) as normal.

Rather than modifying the statistics, the use of CMLLR allows adaptive training to be simplified further and allows combined mean and variance adaptation. Similar to the case for semi-tied covariances, the CMLLR transformed likelihood can be computed simply by regarding it as a feature space transformation, i.e. for any mixture componet m of state j

$$\mathcal{N}(\boldsymbol{y}; \hat{\boldsymbol{\mu}}_{jm}, \hat{\Sigma}_{jm}) = \frac{1}{|G|} \mathcal{N}(G^{-1}(\boldsymbol{y} - \boldsymbol{b}); \boldsymbol{\mu}_{jm}, \Sigma_{jm}) \tag{36}$$

where $\hat{\boldsymbol{\mu}}_{jm}$ and $\hat{\Sigma}_{jm}$ are the transformed means and variances as in (33) (with $G = H$)). Thus a SAT system can be built using CMMLR simply by iterating between the estimation of the canonical model using estimation of the transformed training data and the transforms using the canonical model.

Finally, note that SAT trained systems incur the problem that they can only be used once transforms have been estimated for the test data. Thus, an SI model set is typically retained to generate the initial hypothesised transcription or lattice needed to compute the first set of transforms.

6. MULTI-PASS RECOGNITION ARCHITECTURES

The previous sections have reviewed some of the basic techniques available for both training and adapting a HMM-based recognition system. In general, any particular combination of model set and adaptation technique will have slightly different characteristics and make different errors. Furthermore, if the outputs of these systems are converted to confusion networks as explained in section 2.4, then it is straightforward to combine the confusion networks and then select the word sequence with the overall maximum posterior likelihood. Thus, modern transcription systems typically utilise multiple model sets and make multiple passes over the data.

A typical architecture is shown in Fig. 13. A first pass is made over the data using a relatively simple SI model set. The 1-best output from this pass is used to perform a first round of adaptation. The adapted models are then used to generate lattices using a basic bigram or trigram word-based language model. Once the lattices have been generated, a range of more complex models and adaptation techniques can be applied in parallel to provide n candidate output confusion networks from which the best word sequence is extracted. These 3rd pass models may include ML and MPE trained systems, SI and SAT trained models, triphone and quinphone models, lattice-based MLLR, CMLLR, 4-gram language models interpolated with class-ngrams and many other variants. For examples of recent large-scale transcription systems see [11, 60, 79].

The gains obtained from this type of system combination can vary but overall performance is more robust across a range of task domains. Finally, note that adaptation can work more effectively if the required hypothesised transcriptions are generated by a different system. Thus, cross adaptation is also an increasingly popular architectural option.

7. CONCLUSIONS

This chapter has reviewed the core architecture of a HMM-based CSR system and outlined the major areas of refinement incorporated into modern-day systems. Diagonal covariance continuous density HMMs are based on the premise that each input frame is independent, its components are decorrelated and have Gaussian distributions. Since none of these assumptions are true, the various refinements described above can all be viewed as attempts to reduce the impact of these false assumptions. Whilst many of the techniques are quite complex, they are nevertheless effective and overall substantial progress has been made. For example, error rates on the transcription of conversational telephone speech were around 45% in 1995. Today, with the benefit of more data, and the refinements described above, error rates are now well below 20%. Similarly, broadcast news transcription has improved from around 30% WER in 1997 to below 15% today.

Despite their dominance and the continued rate of improvement, many argue that the HMM architecture is fundamentally flawed and performance must asymptote. Of course, this is undeniably true since we have in our own heads an existence proof. However, no good alternative to the HMM has been found yet. In the meantime, the performance asymptote seems to be still some way away.

8. REFERENCES

[1] JK Baker. The Dragon System - an Overview. *IEEE Trans Acoustics Speech and Signal Processing*, 23(1):24–29, 1975.

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

[3] BT Lowerre. *The Harpy Speech Recognition System*. PhD thesis, Carnegie Mellon, 1976.

[4] LR Rabiner, B-H Juang, SE Levinson, and MM Sondhi. Recognition of Isolated Digits Using HMMs with Continuous Mixture Densities. *ATT Technical J*, 64(6):1211–1233, 1985.

[5] LR Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc IEEE*, 77(2):257–286, 1989.

[6] PJ Price, W Fisher, J Bernstein, and DS Pallet. The DARPA 1000-word Resource Management database for continuous speech recognition. In *Proc ICASSP*, volume 1, pages 651–654, New York, 1988.

[7] SJ Young and LL Chase. Speech Recognition Evaluation: A Review of the US CSR and LVCSR Programmes. *Computer Speech and Language*, 12(4):263–279, 1998.

[8] DS Pallet, JG Fiscus, J Garofolo, A Martin, and M Przybocki. 1998 Broadcast News Benchmark Test Results: English and Non-English Word Error Rate Performance Measures. Technical report, National Institute of Standards and Technology (NIST), 1998.

[9] JJ Godfrey, EC Holliman, and J McDaniel. SWITCHBOARD. In *Proc ICASSP*, volume 1, pages 517–520, San Francisco, 1992.

[10] G Evermann, HY Chan, MJF Gales, T Hain, X Liu, D Mrva, L Wang, and P Woodland. Development of the 2003 CU-HTK Conversational Telephone Speech Transcription System. In *Proc. ICASSP*, Montreal, Canada, 2004.

[11] H Soltau, B Kingsbury, L Mangu, D Povey, G Saon, and G Zweig. The IBM 2004 Conversational Telephony System for Rich Transcription. In *Proc ICASSP*, Philadelphia, PA, 2005.

[12] SJ Young *et al*. *The HTK Book Version 3.4*. Cambridge University, `http://htk.eng.cam.ac.uk`, 2006.

[13] SJ Young. Large Vocabulary Continuous Speech Recognition. *IEEE Signal Processing Magazine*, 13(5):45–57, 1996.

[14] SB Davis and P Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans Acoustics Speech and Signal Processing*, 28(4):357–366, 1980.

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

[16] AP Dempster, NM Laird, and DB Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J Royal Statistical Society Series B*, 39:1–38, 1977.

[17] SJ Young, JJ Odell, and PC 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, 1994.

[18] X Luo and F Jelinek. Probabilistic Classification of HMM States for Large Vocabulary. In *Proc Int Conf Acoustics, Speech and Signal Processing*, pages 2044–2047, Phoenix, USA, 1999.

[19] SM Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recogniser. *IEEE Trans ASSP*, 35(3):400–401, 1987.

[20] H Ney, U Essen, and R Kneser. On Structuring Probabilistic Dependences in Stochastic Language Modelling. *Computer Speech and Language*, 8(1):1–38, 1994.

[21] SF Chen and J Goodman. An Empirical Study of Smoothing Techniques for Language Modelling. *Computer Speech and Language*, 13:359–394, 1999.

[22] PF Brown, VJ Della Pietra, PV de Souza, JC Lai, and RL Mercer. Class-based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479, 1992.

[23] R Kneser and H Ney. Improved Clustering Techniques for Class-based Statistical Language Modelling. In *Proc Eurospeech*, pages 973–976, Berlin, 1993.

[24] S Martin, J Liermann, and H Ney. Algorithms for Bigram and Trigram Word Clustering. In *Proc Eurospeech'95*, volume 2, pages 1253–1256, Madrid, 1995.

[25] SJ Young, NH Russell, and JHS Thornton. Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems. Technical Report CUED/F-INFENG/TR38, Cambridge University Engineering Department, 1989.

[26] K Demuynck, J Duchateau, and D van Compernolle. A Static Lexicon Network Representation for Cross-Word Context Dependent Phones. In *Proc Eurospeech*, pages 143–146, Rhodes, Greece, 1997.

[27] SJ Young. Generating Multiple Solutions from Connected Word DP Recognition Algorithms. In *Proc IOA Autumn Conf*, volume 6, pages 351–354, 1984.

[28] H Thompson. Best-first enumeration of paths through a lattice - an active chart parsing solution. *Computer Speech and Language*, 4(3):263–274, 1990.

[29] F Richardson, M Ostendorf, and JR Rohlicek. Lattice-Based Search Strategies for Large Vocabulary Recognition. In *Proc ICASSP*, volume 1, pages 576–579, Detroit, 1995.

[30] L Mangu, E Brill, and A Stolcke. Finding Consensus Among Words: Lattice-based Word Error Minimisation. *Computer Speech and Language*, 14(4):373–400, 2000.

[31] G Evermann and PC Woodland. Posterior Probability Decoding, Confidence Estimation and System Combination. In *Proc. Speech Transcription Workshop*, Baltimore, 2000.

[32] G Evermann and PC Woodland. Large vocabulary decoding and confidence estimation

using word posterior probabilities. In *Proc. ICASSP 2000*, pages 1655–1658, Istanbul, Turkey, 2000.

[33] V Goel, S Kumar, and B Byrne. Segmental Minimum Bayes-Risk ASR Voting Strategies. In *ICSLP*, 2000.

[34] J Fiscus. A Post-Processing System to Yield Reduced Word Error Rates: Recogniser Output Voting Error Reduction (ROVER). In *Proc IEEE ASRU Workshop*, pages 347–352, Santa Barbara, 1997.

[35] D Hakkani-Tur, F Bechet, G Riccardi, and G Tur. Beyond ASR 1-best: Using word confusion networks in spoken language understanding . *Computer Speech and Language*, In press, 2006.

[36] JJ Odell, V Valtchev, PC Woodland, and SJ 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, 1994.

[37] X Aubert and H Ney. Large Vocabulary Continuous Speech Recognition Using Word Graphs. In *Proc ICASSP*, volume 1, pages 49–52, Detroit, 1995.

[38] M Mohri, F Pereira, and M Riley. Weighted Finite State Transducers in Speech Recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[39] F Jelinek. A Fast Sequential Decoding Algorithm Using a Stack. *IBM J Research and Development*, 13, 1969.

[40] DB Paul. Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder. In *Proc ICASSP*, pages 693–996, Toronto, 1991.

[41] A Nadas. A Decision Theoretic Formulation of a Training Problem in Speech Recognition and a Comparison of Training by Unconditional Versus Conditional Maximum Likelihood. *IEEE Trans Acoustics Speech and Signal Processing*, 31(4):814–817, 1983.

[42] B-H Juang, SE Levinson, and MM Sondhi. Maximum Likelihood Estimation for Multivariate Mixture Observations of Markov Chains. *IEEE Trans Information Theory*, 32(2):307–309, 1986.

[43] LR Bahl, PF Brown, PV de Souza, and RL Mercer. Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. In *Proc ICASSP*, pages 49–52, Tokyo, 1986.

[44] V Valtchev, JJ Odell, PC Woodland, and SJ Young. MMIE Training of Large Vocabulary Recognition Systems. *Speech Communication*, 22:303–314, 1997.

[45] R Schluter, B Muller, F Wessel, and H Ney. Interdependence of Language Models and Discriminative Training. In *Proc IEEE ASRU Workshop*, pages 119–122, Keystone, Colorado, 1999.

[46] P Woodland and D Povey. Large Scale Discriminative Training of Hidden Markov Models for Speech Recognition. *Computer Speech and Language*, 16:25–47, 2002.

[47] W Chou, C H Lee, and BH Juang. Minimum Error Rate Training Based on N-best String Models. In *Proc ICASSP 93*, pages p652–655, Minneapolis, 1993.

[48] D Povey and P Woodland. Minimum Phone Error and I-Smoothing for Improved Discriminative Training. In *Proc ICASSP*, Orlando, Florida, 2002.

[49] PS Gopalakrishnan, D Kanevsky, A Nadas, D Nahamoo, and MA Picheny. Decoder Selection based on Cross-Entropies. In *Proc ICASSP*, volume 1, pages 20–23, New York, 1988.

[50] PC Woodland and D Povey. Large Scale Discriminative Training for Speech Recognition. In *ISCA ITRW Automatic Speech Recognition: Challenges for the Millenium*, pages 7–16, Paris, 2000.

[51] MJF Gales. Semi-tied Covariance Matrices For Hidden Markov Models. *IEEE Trans Speech and Audio Processing*, 7(3):272–281, 1999.

[52] AV Rosti and M Gales. Factor Analysed Hidden Markov Models for Speech Recognition. *Computer Speech and Language*, 18(2):181–200, 2004.

[53] S Axelrod, R Gopinath, and P Olsen. Modeling with a Subspace Constraint on Inverse Covariance Matrices. In *ICSLP 2002*, Denver, CO, 2002.

[54] P Olsen and R Gopinath. Modeling Inverse Covariance Matrices by Basis Expansion. In *Proc ICSLP 2002*, Denver, CO, 2002.

[55] N Kumar and AG Andreou. Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition. *Speech Communication*, 26:283–297, 1998.

[56] MJF Gales. Maximum Likelihood Multiple Subspace Projections for Hidden Markov Models. *IEEE Trans Speech and Audio Processing*, 10(2):37–47, 2002.

[57] T Hain, PC Woodland, TR Niesler, and EWD Whittaker. The 1998 HTK System for Transcription of Conversational Telephone Speech. In *Proc IEEE Int Conf Acoustics Speech and Signal Processing*, pages 57–60, Phoenix, 1999.

[58] G Saon, A Dharanipragada, and D Povey. Feature Space Gaussianization. In *Proc ICASSP*, Montreal, Canada, 2004.

[59] SS Chen and R Gopinath. Gaussianization. In *NIPS 2000*, Denver, Colorado, 2000.

[60] MJF Gales, B Jia, X Liu, KC Sim, P Woodland, and K Yu. Development of the CUHTK 2004 RT04 Mandarin Conversational Telephone Speech Transcription System. In *Proc. ICASSP*, Philadelphia, PA, 2005.

[61] L Lee and RC Rose. Speaker Normalisation Using Efficient Frequency Warping Procedures. In *ICASSP'96*, Atlanta, 1996.

[62] J McDonough, W Byrne, and X Luo. Speaker Normalisation with All Pass Transforms. In *Proc ISCLP98*, Sydney, 1998.

[63] DY Kim, S Umesh, MJF Gales, T Hain, and P Woodland. Using VTLN for Broadcast News Transcription. In *ICSLP04*, Jeju, Korea, 2004.

[64] J-L Gauvain and C-H Lee. Maximum a Posteriori Estimation of Multivariate Gaussian Mixture Observations of Markov Chains. *IEEE Trans Speech and Audio Processing*, 2(2):291–298, 1994.

[65] SM Ahadi and PC Woodland. Combined Bayesian and Predictive Techniques for Rapid Speaker Adaptation of Continuous Density Hidden Markov Models. *Computer Speech and Language*, 11(3):187–206, 1997.

[66] K Shinoda and C H Lee. Structural MAP Speaker Adaptation using Hierarchical Priors. In *Proc ASRU'97*, Santa Barbara, 1997.

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

[68] MJF Gales. Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition. *Computer Speech and Language*, 12:75–98, 1998.

[69] F Wallhof, D Willett, and G Rigoll. Frame-discriminative and Confidence-driven Adaptation for LVCSR. In *Proc ICASSP'00*, pages 1835–1838, Istanbul, 2000.

[70] L Wang and P Woodland. Discriminative Adaptive Training using the MPE Criterion. In *Proc ASRU*, St Thomas, US Virgin Islands, 2003.

[71] S Tsakalidis, V Doumpiotis, and WJ Byrne. Discriminative Linear Transforms for Feature Normalisation and Speaker Adaptation in HMM Estimation. *IEEE Trans Speech and Audio Processing*, 13(3):367–376, 2005.

[72] P Woodland, D Pye, and MJF Gales. Iterative Unsupervised Adaptation using Maximum Likelihood Linear Regression. In *ICSLP'96*, pages 1133–1136, Philadelphia, 1996.

[73] M Padmanabhan, G Saon, and G Zweig. Lattice-based Unsupervised MLLR for Speaker Adaptation. In *Proc ITRW ASR2000: ASR Challenges for the New Millenium*, pages 128–132, Paris, 2000.

[74] TJ Hazen and J Glass. A Comparison of Novel Techniques for Instantaneous Speaker Adaptation. In *Proc Eurospeech'97*, pages 2047–2050, 1997.

[75] R Kuhn, L Nguyen, J-C Junqua, L Goldwasser, N Niedzielski, S Finke, K Field, and M Contolini. Eigenvoices for Speaker Adaptation. In *ICSLP'98*, Sydney, 1998.

[76] MJF Gales. Cluster Adaptive Training of Hidden Markov Models. *IEEE Trans Speech and Audio Processing*, 8:417–428, 2000.

[77] K Yu and MJF Gales. Discriminative Cluster Adaptive Training. *IEEE Trans Speech and Audio Processing*, 2006.

[78] T Anastasakos, J McDonough, R Schwartz, and J Makhoul. A Compact Model for Speaker Adaptive Training. In *ICSLP'96*, Philadelphia, 1996.

[79] R Sinha, MJF Gales, DY Kim, X Liu, KC Sim, and PC Woodland. The CU-HTK Mandarin Broadcast News Transcription System. In *ICASSP'06*, Toulouse, 2006.

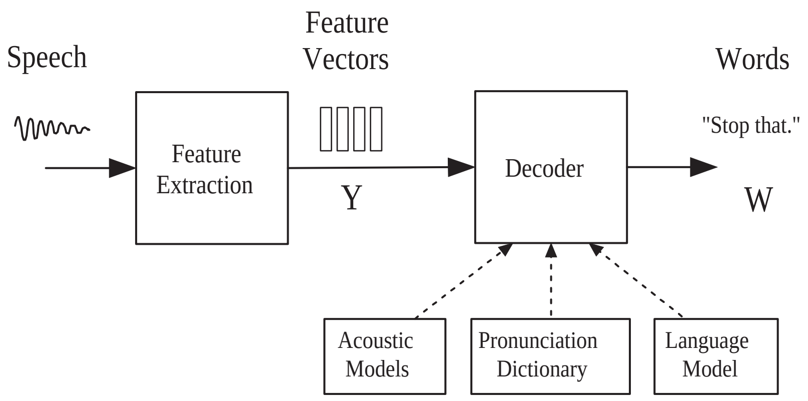

Figure 1: Architecture of a HMM-based Recogniser

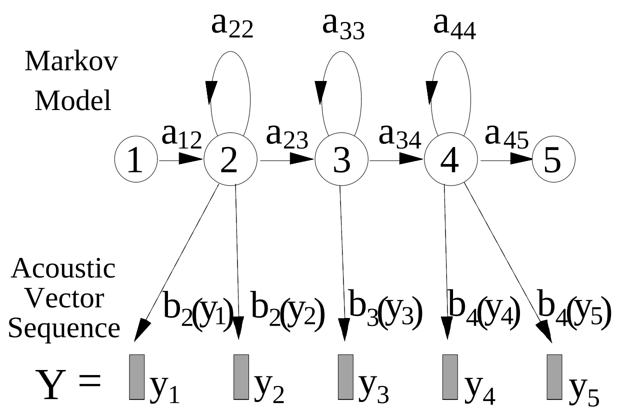

Figure 2: HMM-based Phone Model

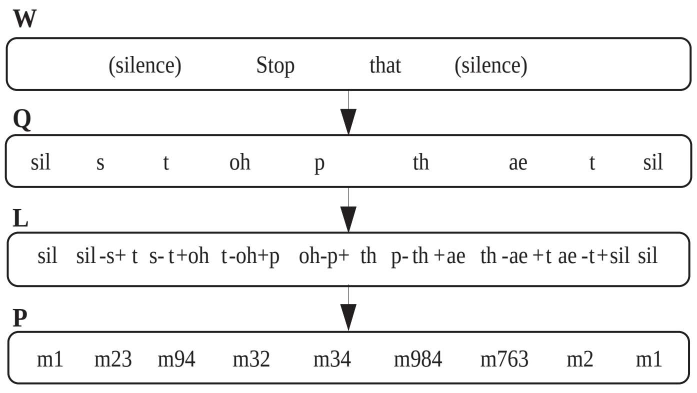
W
(silence) Stop that (silence)
Q
sil s t oh p th ae t sil
L
sil sil-s+t s-t+oh t-oh+p oh-p+th p-th+ae th-ae+t ae-t+sil sil
P
m1 m23 m94 m32 m34 m984 m763 m2 m1

Figure 3: Context Dependent Phone Modelling

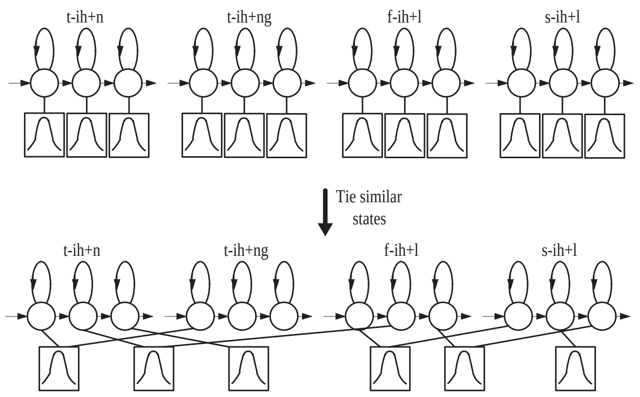

Figure 4: Formation of Tied-State Phone Models

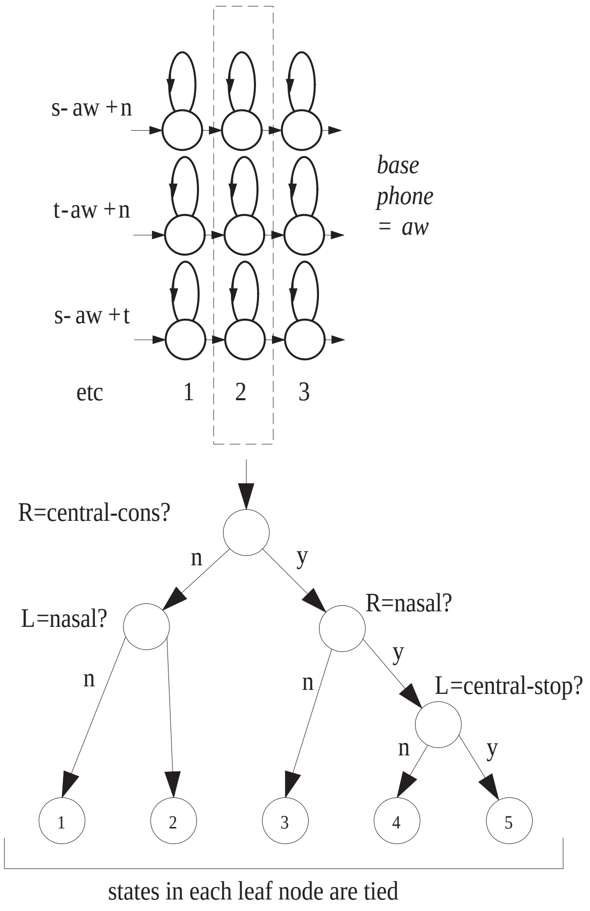

Figure 5: Decision Tree Clustering

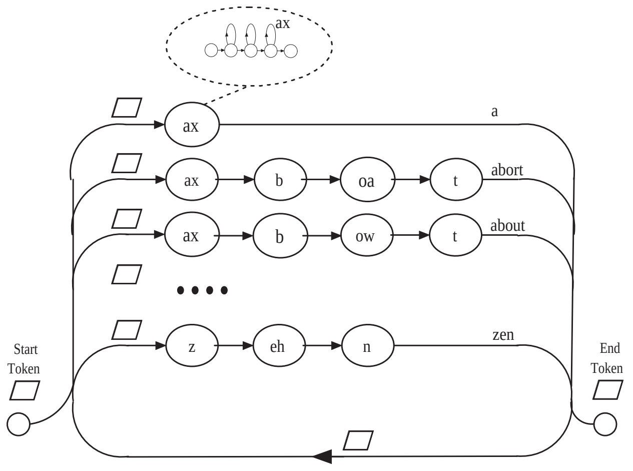

Figure 6: Basic Recognition Network

```
Put a start token $< \log(1), \emptyset >$ in network entry node;
Put null tokens $< \log(0), \emptyset >$ in all other nodes;
for each time $t = 1$ to $T$ do
    – *word internal token propagation*
    for each non-entry node $j$ do
        $MaxP = \log(0)$;
        for each predecessor node $i$ do
            Temp token $Q = Q_i$;
            $Q.LogP$ += $\log(a_{ij})$ [+ $\log(b_j(\mathbf{y}_t))$ if $j$ emitting ];
            If $Q.LogP > MaxP$ then
                $Q_j = Q$; $MaxP = Q.LogP$;
        end;
    end;
    Copy tokens from word internal exits to following entries;
    – *word external token propagation*
    for each word $w$ with entry node $j$ do
        $MaxP = \log(0)$;
        for each predecessor word $u$ with exit node $i$ do
            Temp token $Q = Q_i$;
            $Q.LogP$ += $\alpha \log p(w|u) + \beta$;
            If $Q.LogP > MaxP$ then
                $Q_j = Q$; $MaxP = Q.LogP$; $u' = u$
        end;
        – *Record word boundary decision*
        Create a record $R$;
        $R.Q = Q_j$; $R.t = t$; $R.word = u'$;
        $Q.link = \uparrow R$;
    end;
    Put null token in network entry node;
end;
Token in network exit state at time $T$ represents the best path;
```

Figure 7: Basic Token Passing Algorithm

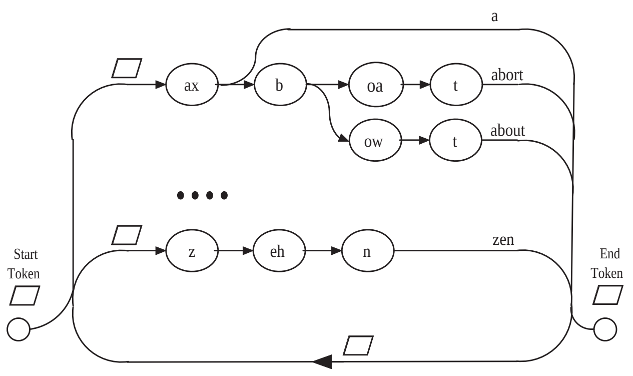

Figure 8: Tree-Structured Recognition Network

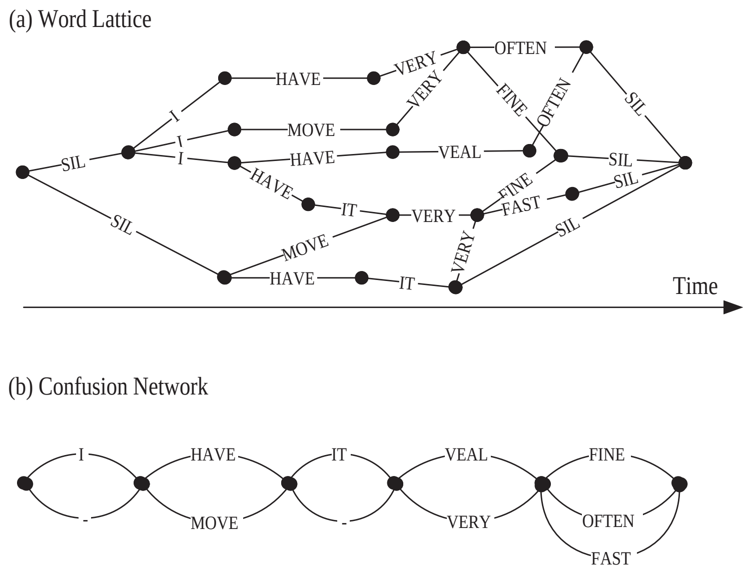

Figure 9: Example Lattice and Confusion Network

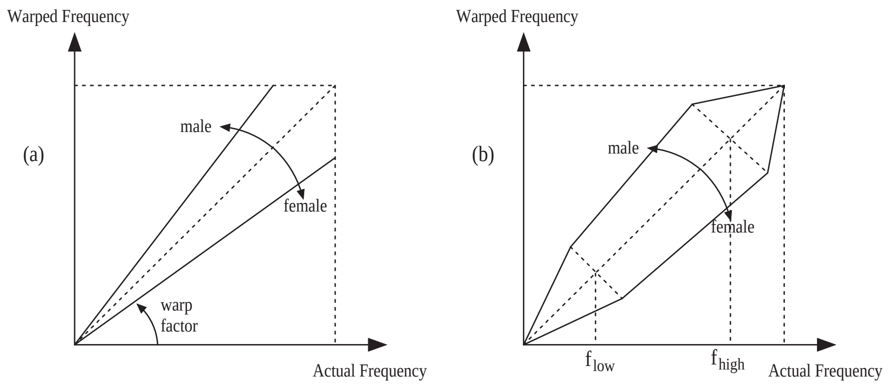

Figure 10: Vocal Tract Length Normalisation

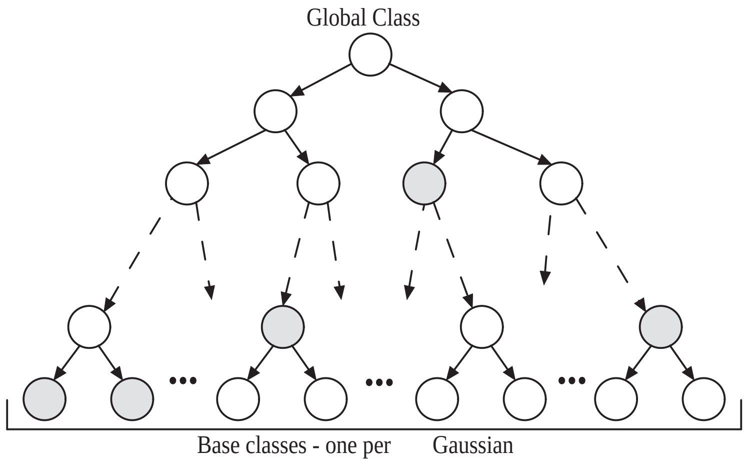

Figure 11: A Regression Class Tree

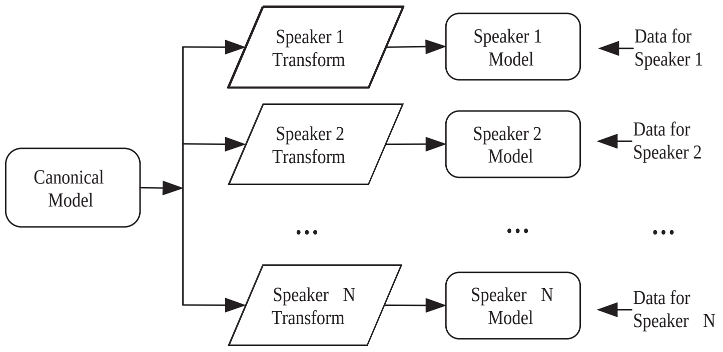

Figure 12: Adaptive Training

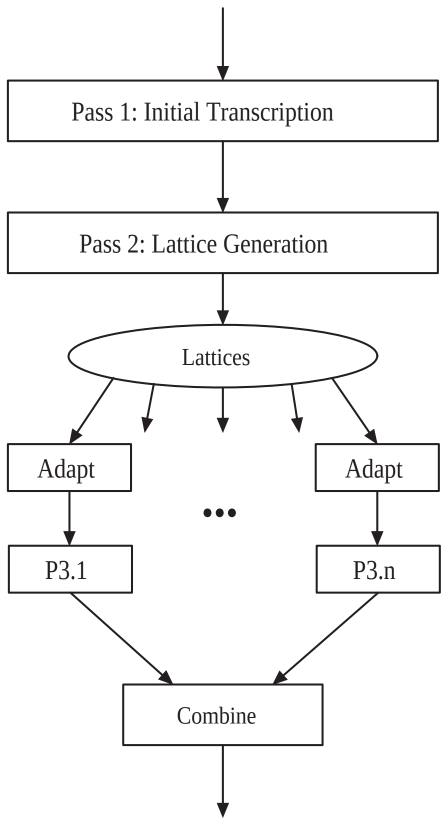

Figure 13: Multi-pass/System Combination Architecture

Index