The Use of Context in Large Vocabulary Speech Recognition

Julian James Odell

Queens' College

March 1995

Dissertation submitted to the University of Cambridge

for the degree of Doctor of Philosophy

Summary

In recent years, considerable progress has been made in the field of continuous speech recognition where the predominant technology is based on hidden Markov models (HMMs). HMMs represent sequences of time varying speech spectra using probabilistic functions of an underlying Markov chain.

However, because the probability distribution represented by a HMM is very simple, its discriminative ability is limited. As a consequence, a careful choice of the units represented by each model is required in order to accurately model the variation inherent in natural speech. In practice, much of the variation is due to consistent contextual effects and can be accounted for by using context dependent models.

In large vocabulary recognition the use of context dependent models introduces two major problems. Firstly, some method must be devised to determine the set of contexts which require distinct models. Furthermore, this must be done in a way which takes account of the sparsity and unevenness of the training data. Secondly, a strategy must be devised which allows efficient decoding using models incorporating context dependencies both within words and across word boundaries. This thesis addresses both of these key problems.

Firstly, a method of constructing robust and accurate recognisers using decision tree based clustering techniques is described. The strength of this approach lies in its ability to accurately model contexts not appearing in the training data. Linguistic knowledge is used, in conjunction with the data, to decide which contexts are similar and can share parameters. A key feature of this approach is that it allows the construction of models which are dependent upon contextual effects occurring across word boundaries.

The use of cross word context dependent models presents problems for conventional decoders. The second part of the thesis therefore presents a new decoder design which is capable of using these models efficiently. The decoder is suitable for use with very large vocabularies and long span language models. It is also capable of generating a lattice of word hypotheses with little computational overhead. These lattices can be used to constrain further decoding, allowing efficient use of complex acoustic and language models.

The effectiveness of these techniques has been assessed on a variety of large vocabulary continuous speech recognition tasks and results are presented which analyse performance in terms of computational complexity and recognition accuracy. The experiments demonstrate state of the art performance and a recogniser using these techniques was used in the 1994 US ARPA CSR Evaluations where it returned the lowest error rate of any system tested.

Keywords: speech recognition, hidden Markov models, context, decoding.

Acknowledgments

First I must thank my supervisor, Professor Steve Young. His help and advice (together with a little prodding) are ultimately responsible for this thesis. He has been a continuous source of inspiration, and support throughout my PhD. In particular, by providing HTK as a base from which I could start my experiments, he made my task immeasurably simpler. I must also thank Phil Woodland, not just for his work on HTK, but also for his help in running experiments. A great deal of the large vocabulary work was performed as part of a team and Phil's perseverance and hard work are responsible for the excellent recognition results obtained. To the other members of the CU-HTK team, Valtcho Valtchev (language models), Mark Gales (noise) and Chris Leggetter (adaptation), must go, not just thanks for their efforts, but also apologies for having to alpha test my code.

I must also acknowledge the Engineering and Physical Sciences Research Council for funding my research. Together with Queens' College, the Engineering Department and the NSA they also funded my attendance at workshops in America during which I was able to visit Austin, New York, Boston, Washington and Princeton.

Special thanks go to the ECR lab support team who managed to keep the computer system running despite my efforts to overwhelm it. In no particular order these include Richard Prager, Tony Robinson, Andy Gee, Carl Seymour and Patrick Gosling (and no doubt many others and I apologise for not mentioning them). Paul Mossip (help@CAIP) deserves mention as well for his support efforts and for reconfiguring the compute server at Rutgers more times than I care to remember.

My housemates at 71 Maids Causeway (Costi, Roy, Simon, Sis, Darin, Henk and Kevin) are also deserving of my thanks. They managed to keep me fed, busy and relatively sane over the years, not to mention alive despite the by now infamous barbecue incident.

Finally I would like to thank the various members of the Cambridge University Engineering Department Speech Vision and Robotics group for providing a pleasant environment to work, lively social activities and helpful advice. In these ways Mark, Chris, Kate, Dan, Gareth, Matt, Carl, Tina, Steve, Simon, Valtcho and many others have all contributed a great deal to this thesis.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where stated. It has not been submitted in whole or part for a degree at any other university.

The length of this thesis including footnotes and appendices is approximately 46,000 words.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For most people their first glimpse of automatic speech recognition technology will probably be in a science fiction movie where the robots and computers interact with the characters using natural spoken language. However their first use of a speech recogniser will probably be a simple telephone enquiry system only capable of making the most rudimentary distinctions.

These different abilities represent the extremes of speech recognition research. However, the 'intelligent' machines from the movies also embody a separate ability. The ability to understand speech and hold a meaningful dialogue is a separate field which relies on speech recognition technology for its front-end processing. The recognition system is responsible for converting the acoustic signal into a sequence of symbols, often words, which represent the spoken message. Current research is focussed on increasing the range of speech that can be accurately recognised.

1.1 Speech Recognition

The telephone enquiry systems with which people first come into contact tend to rely on simple yes/no decisions or selections from a fixed menu of options. This means that they appear to be slow, clumsy and often confusing. Expanding the capabilities of such systems (thus expanding the range of tasks for which automatic speech recognisers are useful) requires that the number of distinctions such systems can make to be extended. This is accomplished by increasing the number of different words that can be recognised (the *vocabulary*) and allowing those words to be combined in more ways (by using a more complex *grammar*). However, as the number of permitted variations increases, the perplexity and confusability of the task rises. Maintaining high levels of accuracy needs improvements to both the acoustic models, which provide information about which words best match the speech signal, and the language model, which provides prior knowledge of likely word sequences.

Figure 1.1 shows the process of recognition broken into distinct stages. First the person thinks of what they wish to say and articulates it. The resulting acoustic signal is processed into a form suitable for recognition and then a model of language in conjunction with the acoustic models of speech are used to determine what the person said. Even without the need to understand the spoken message, the recognition process involves many different disciplines.

- Physiology.

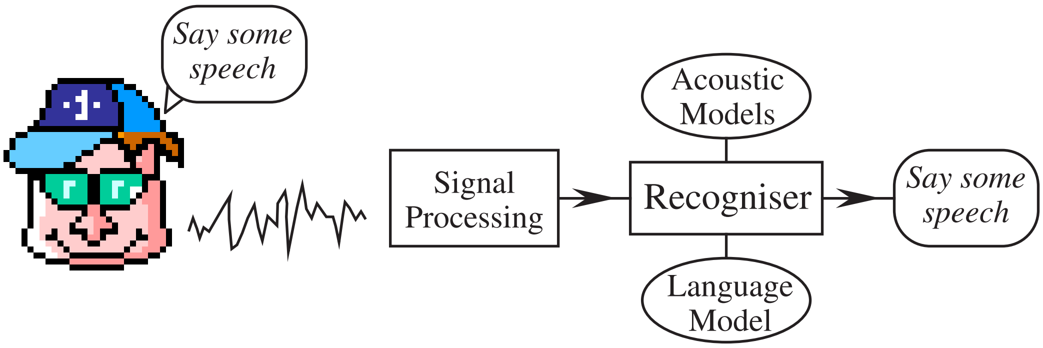

Figure 1.1: The speech production/recognition process.

Each person speaks in a slightly different way. Normally this does not cause other people any significant difficulty in recognising their speech. Substantially different acoustic realisations of the same underlying utterance are usually perceived by others as being the same. A successful recogniser must be able to duplicate this behaviour and treat as similar sounds which are perceived alike.

- Signal Processing.

  In order to exhibit this robust behaviour the acoustic signal must be converted into a representation orientated to recognition. This requires careful signal processing to remove extraneous acoustic information but not the distinctions that are important for recognition.

- Pattern Recognition.

  Once the signal has been transformed into a more perceptually oriented form it must be recognised, or decoded, and transformed into the underlying sequence of symbols. This decoding process requires patterns or models against which unknown utterances can be compared to deduce the most likely sequence of symbols represented.

- Information Theory.

  The models against which unknown utterances are compared must be representative of speech in general. They must be robust enough to capture the variation which naturally occurs in human speech. Often this will require the model parameters to be estimated (even if only by the system designer) in a robust manner. Information theory allows the designer to ensure that the chosen parameters are in some way optimal.

- Computer Science.

  Nowadays the whole process of recognition is performed using digital computers. Before the advent of fast and cheap digital signal processing, discrete electronics would be used to carry out any necessary signal processing in the analogue domain. However the relative cost of these methods has changed dramatically and most current research is based solely in the digital domain. The astronomical rate of increase in the speed and capabilities of computers has allowed similar increases in the complexity of recognition tasks attempted. However as the tasks attempted become more complex, the efficiency of the algorithms used in recognition becomes increasingly important.

The work described in this thesis has focused on increasing the accuracy of the pattern matching stage for large vocabulary recognition. As this is intimately related to the computationally expensive decoding process, the computer science aspects of the problem are also important. Tasks considered 'large vocabulary' only a few years ago (such as the 1000 word Resource Management task) have been supplanted and the goal posts moved. For example, the most complex system developed using the techniques described later uses a vocabulary of over 65,000 words to recognise speech from an unlimited vocabulary.

1.2 Development of Speech Recognisers

Research into automatic speech recognition began in the 1950's and has been an active area of study since then. The initial attempts at automatic speech recognition were based on template matching. Template based approaches compare templates (which are usually actual examples of speech) with unknown words. The symbol associated with the best matching template is hypothesised for the unknown speech. This approach is simple to implement but lacks generality.

Simple pattern matching techniques are not suited to recognition of fluent (or *continuous*) speech because segment boundaries are difficult to detect. In normal speech, word boundaries are not associated with any particular acoustic event. To allow the utterance to be segmented into words a short period of silence was therefore required between words. Each word could then be compared with the templates to produce an *isolated* word recogniser.

Since a fixed set of example templates are used for recognition, this approach tends to be best suited to small vocabulary speaker dependent recognition. The need for an example of each word restricts vocabulary size and as only a single template is matched at a time the system tends not to be robust to changes in speaker.

Attempts to solve these problems by adding supplementary expert knowledge (from linguists, spectrogram readers and other speech specialists) met with only limited success. The expert knowledge tended to be encapsulated in the form of rules which decided when particular features were present. Often these decisions were based on identifying the resonances (or *formants* present in the signal. However formants are only easily detected in vowels and together with the natural variability of speech, meant that it was difficult to make these yes/no distinctions reliably. Once errors were made they were difficult to correct.

1.3 Stochastic Process Models

Word template based approaches are limited by the need for examples of each word and by their inability to allow for the natural variability of speech. Knowledge based approaches are limited by the difficulty in defining rules that can make subtle distinctions and spot trends against a background of significant natural variability.

These limitations can be overcome by using stochastic process models of sub-word units. One type of stochastic process model, the hidden Markov model [76], is the basis of the majority of current speech recognition systems and the work in this thesis. Stochastic approaches use a database of examples to estimate the parameters of a probabilistic model. When the available data contains sufficient examples of each model, it is possible to produce a representation of speech which is robust to the variations present in natural speech. The form of this representation can vary from simple probability distributions estimated directly from the data to complex classifiers such as artificial neural networks [77].

Because the signal is represented as a process, it is possible to concatenate a series of models to produce a composite model of a complete continuous speech utterance. This feature makes it possible to use lower level representations for the underlying speech. Rather than using a distinct model for each word a smaller set of sub-word models together with a dictionary are used to construct a composite model for the word. This decreases the amount of training data required to cover a particular vocabulary by using supplementary knowledge from the dictionary.

For English (and American English) the phoneme seems to be the best choice of sub-word unit representing a compromise between invariance and data availability. The relatively small number of phonemes allows a small set of phone models to cover the whole of the English language and ensures that several examples of each will occur in a database of reasonable size.

1.4 Modelling Context

The pattern matching techniques used for recognition rely on the invariance of the data represented by a particular model. In normal fluent speech every instance of a particular sound can be different. Some of this variation is random but a great deal of it can be accounted for by consistent *contextual* effects.

To improve recognition accuracy it is necessary to take account of these consistent variations whilst still allowing for the random variations always present in speech. One of the most important causes of consistent variation is co-articulation, in which the realisation of a particular phone is effected by its neighbours. Each instance is influenced not just by the position of the vocal articulators during its production but also by their movements before and after. When all phonetic contexts are enumerated the number of distinct models required increases by several orders of magnitude. This re-introduces one of the drawbacks of whole word modelling; the need for sufficient examples of each basic unit. With a small set of sub-word phone models a reasonably sized corpus will provide examples of all phones. However, the uneven distribution of phones and contexts in general speech means that even in very large databases many contexts will occur only a few times, if at all. Increasing the robustness of large recognition systems without giving up the ability to account for consistent contextual variation requires some form of parameter sharing or smoothing to ensure that the available training data is used as efficiently as possible.

This dissertation addresses the problem of modelling contexts accurately and robustly paying special attention to constructing models for contexts in the absence of data. The ability to construct models for these *unseen* contexts is needed to produce accurate and robust recognition systems for unlimited vocabularies. A top down clustering approach has been developed which allows linguistic knowledge to be used to supplement the data. In the absence of examples of a particular context its model will share parameters with contexts which are considered linguistically similar and for which there is data.

1.5 Decoding Issues

Accurate recognition requires accurate acoustic and language modelling techniques used with accurate decoders. Accurate acoustic and language models require that contextual features which lead to consistent variations (such as those described above) are taken into account. During decoding the use of such models can lead to dependencies spanning several words. Conventional decoding techniques are not suited to this type of modelling and do not scale well to very large vocabularies. To take advantage of the enhanced acoustic modelling techniques developed an improved decoder was needed.

1.6 Thesis Structure

The recognition systems developed in this dissertation are based on hidden Markov models. Chapter 2 describes the structure of these models and outlines the algorithms needed to train and test recognisers using them. It also introduces the terminology used in the remainder of the thesis.

This work has tackled the problems raised by modelling context in particular addressing the issues of unseen contexts and model trainability by the sharing of parameters. The choice of models to share parameters is based upon both the data and prior linguistic knowledge. This linguistic knowledge is used to ensure that reasonable models are produced for contexts which do not appear in the data. Appendix B lists the set of phonetic features which are used to encapsulate this linguistic knowledge as well as describing the phone sets and dictionaries used in this work. This decision tree based approach to model building is presented in chapter 3 following a review of the use of context dependent acoustic modelling.

One of the major drawbacks with systems making use of context dependent models (in particular if the context of surrounding words is considered) is the computational expense of recognition using conventional decoding techniques. Chapter 4 reviews these techniques and shows why these are unsuited to recognition using extended context systems.

A novel decoder architecture is described in chapter 5. This architecture allows efficient recognition using context dependent models and very large vocabularies. It also supports the generation of multiple hypotheses (in the form of a lattice) with little or no computational overhead.

Results produced by this decoder using decision tree clustered models are presented in chapter 6 for a variety of large vocabulary speaker independent continuous speech recognition tasks. These tasks are described in appendix A. These results show that the decision tree based acoustic modelling techniques are capable of state of the art performance whilst the decoder architecture provides an efficient means for using them.

Finally a summary of the work is given in chapter 7 which also suggests future areas of research.

Chapter 2

Hidden Markov Models

This chapter outlines the theoretical framework for hidden Markov models (*HMMs*). It begins by explaining what a hidden Markov model is and how it operates. Then it explains how they may be used for speech recognition and outlines the algorithms needed to estimate the model parameters and recognise speech. This chapter also introduces the terminology which will be used throughout this thesis.

2.0.1 HTK: A Hidden Markov Model Toolkit

This work has made extensive use of HTK, a toolkit for the manipulation and use of hidden Markov models. This was written by S. J. Young and P. C. Woodland [94] to enable research into speech recognition using hidden Markov models to progress without needing to re-invent (or at least re-implement) the wheel. I must (again) acknowledge both HTK and its authors for making my life immeasurably easier since the majority of the computer programs written and experiments performed used parts of the toolkit. Full details of the toolkit and the facilities it provides can be found in [97].

The toolkit consists of two parts;

- Libraries.

  The libraries provide functions that enable the manipulation of models and data as well as providing a consistent user interface to the tools.

  Of particular relevance to my work were

  - HModel. Load, save and access the models.
  - HSpIO. Load and save speech data files.
  - HLabel. Load and save label files.
  - HMath. Basic mathematical functions.

- Tools.

  There are a variety of tools that build on the libraries to provide commonly needed facilities for the manipulation of models, data and labels.

– HERest. An embedded Baum-Welch reestimation tool (Section 2.6).
– HVite. A static network Viterbi decoder (Section 4.2).
– HCode. Generates data files by parameterising the acoustic signal. (Section A.1).
– HHEd. A tool for manipulating models.
– HLEd. A tool for manipulating label files.

2.1 Basics

A hidden Markov Model is a stochastic process model in which a discrete time signal is generated from a series of connected *states*. Each time step, or *frame*, the model changes state in accordance with a set of *transition probabilities*. The resultant state then generates a single *observation* in accordance with the *output probability distribution* of that state. Each model has two distinguished states; an entry state, which is the state of the model before the generative process begins, and an exit state, which is the final model state reached once the generative process terminates. Neither of these states generate any observations and so do not have an associated output probability distribution, all the observations are generated by the remaining *emitting* states.

An example of a hidden Markov model generating a series of observations and the corresponding signal is shown in figure 2.1.

The transition probabilities a_{ij} are the conditional probabilities that a model in state i (which may be the entry state, state 1) will change state to state j (which may be the exit state, state N).

So

$$a_{ij} \quad = \quad Prob(x(t+1) = j \mid x(t) = i), \tag{2.1}$$

where $x(t)$ is the state of the model at time t.

The transition probability a_{ij} is assumed to be constant and does not vary over time. For all initial states $i = 1, ..., N-1$ these values should satisfy

$$\sum_{j=2}^{N} a_{ij} \quad = \quad 1.0. \tag{2.2}$$

The output probability distribution function $b_j(\mathbf{o}_t)$ describes the distribution of observations produced by state j. It gives the probability (for discrete output symbols) or likelihood (for continuous output distributions) of state j generating the observation $\mathbf{o}_t$. To avoid unnecessary repetition, the symbol *Pr(event)* and the term *likelihood* will henceforth be used to describe both likelihoods and probabilities depending on whether $b_j(\mathbf{o}_t)$ is a discrete distribution or a continuous density function.

So

$$b_j(\mathbf{o}_t) \quad = \quad Pr(\mathbf{o}_t|x(t) = j), \tag{2.3}$$

is the likelihood of state j generating the observation $\mathbf{o}_t$.

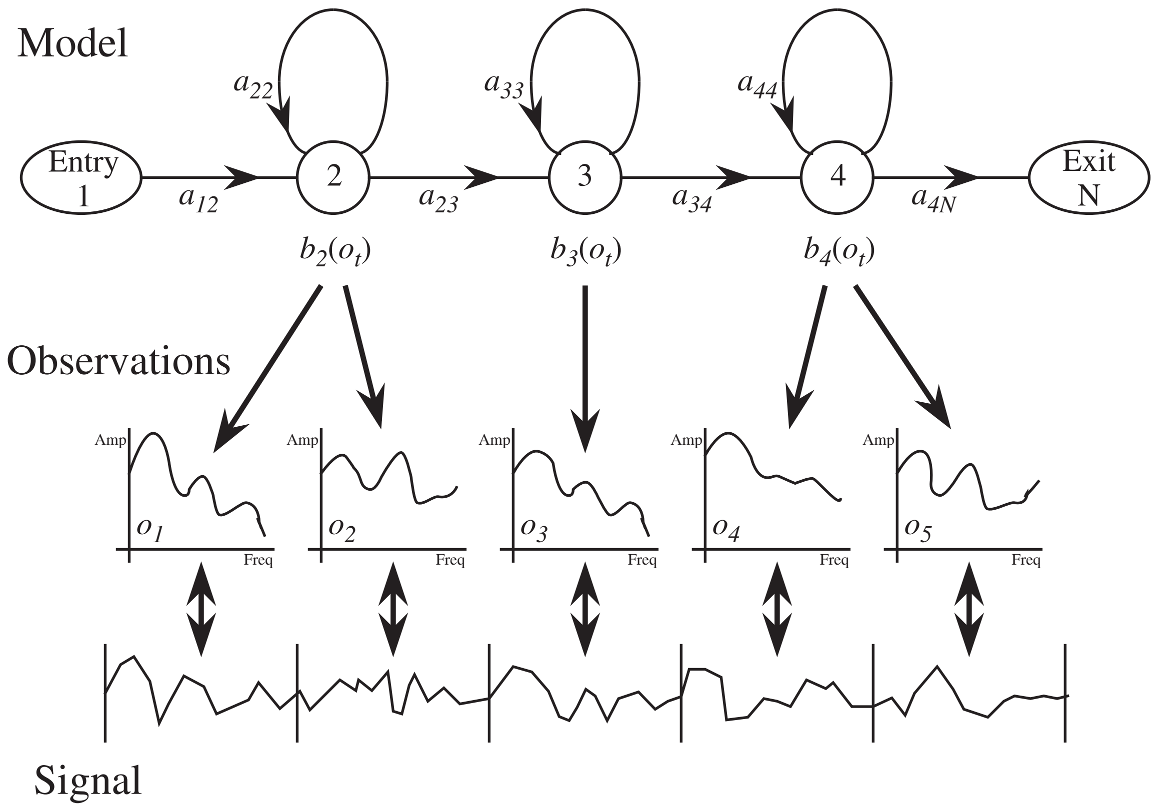

Figure 2.1: An example hidden Markov model.

For discrete distributions

$$\sum_{\mathbf{o}} b_j(\mathbf{o}) \quad = \quad 1.0, \tag{2.4}$$

and for continuous distributions

$$\int_{\mathbf{o}} b_j(\mathbf{o})\ d\mathbf{o} \quad = \quad 1.0. \tag{2.5}$$

for all emitting states $j = 2, ..., N-1$

As well as being used in this generative fashion, the model can be used to calculate the likelihood of a signal, $\mathbf{O}$, consisting of T frames, $\mathbf{o}(1), ..., \mathbf{o}(T)$, being produced by a particular sequence of states, $X = x(1), ..., x(T)$. This is given by the product of the likelihood of each observation $\mathbf{o}_t$ being generated by its associated state, $x(t)$ and the probability of the state sequence $x(t)$ calculated from the transition probabilities.

For the example given in Figure 2.1,

$$Pr(\mathbf{O}, X) = a_{12}\ b_2(\mathbf{o}_1)\ a_{22}\ b_2(\mathbf{o}_2)\ a_{23}\ b_3(\mathbf{o}_3)\ a_{34}\ b_4(\mathbf{o}_4)\ a_{44}\ b_4(\mathbf{o}_5)\ a_{4N}. \tag{2.6}$$

Or more generally,

$$Pr(\mathbf{O}, X) = a_{1x(1)} b_{x(1)}(\mathbf{o}_1) \left(\prod_{t=2}^{T} a_{x(t-1)x(t)} b_{x(t)}(\mathbf{o}_t) \right) a_{x(T)N}. \tag{2.7}$$

Normally only the signal and model parameters will be known and the state sequence will be *hidden*, hence the name hidden Markov models. In this case the likelihood of the model generating the observation must be calculated by summing over all possible state sequences,

$$Pr(\mathbf{O}) = \sum_{X} a_{1x(1)} b_{x(1)}(\mathbf{o}_1) \left(\prod_{t=2}^{T} a_{x(t-1)x(t)} b_{x(t)}(\mathbf{o}_t) \right) a_{x(T)N}. \tag{2.8}$$

Alternatively, it can be approximated as the likelihood of the most probable state sequence, that is

$$\hat{P}(\mathbf{O}) = \max_{X} \left\{ a_{1x(1)} b_{x(1)}(\mathbf{o}_1) \left(\prod_{t=2}^{T} a_{x(t-1)x(t)} b_{x(t)}(\mathbf{o}_t) \right) a_{x(T)N} \right\}. \tag{2.9}$$

Neither of these quantities is computable directly because of the large number of possible state sequences but efficient iterative procedures for evaluating both exist and are described in section 2.4.

2.2 Using Hidden Markov models for Speech Recognition

The strength of hidden Markov models lies not with the accuracy with which they model the process of speech production but with the existence of computationally efficient algorithms for estimating the model parameters given example observation sequences of known class, called *training*, and for choosing which model best matches an observation sequence of unknown class, called *recognition* or *decoding*.

Ideally this training process should ensure that the model parameters are chosen to maximise the accuracy of the resulting recogniser. However the computational cost of such *discriminative* training schemes [12] is often prohibitive. Consequently a computationally efficient maximum likelihood training procedure is more commonly used. This is an iterative estimation-maximisation procedure in which an initial set of models is used to estimate the hidden state sequence and then new better estimates of the model parameters are chosen to maximise the likelihood of this sequence. These values are then used to estimate the state sequence again and obtain still better parameters in an iterative process of parameter refinement.

This *re-estimation* process requires example observation sequences of known class and an initial set of models. This *training data* should adequately represent the variation expected from unseen *test data* to ensure that the model parameters can be calculated accurately. The initial models should be accurate enough to estimate the state sequence with reasonable accuracy. In practice, the re-estimation procedure is robust enough to ensure that the initial model parameters are not crucial and that no special precautions, beyond ensuring that there are enough examples of each class from a selection of speakers, need to be taken to ensure that the training data is representative.

The requirement for sufficient training data governs the choice of the basic modelling unit for a system. Since the recogniser should eventually produce words, the choice of words as the basic unit would give the simplest system. However, this would require that the training data contain many examples of every word in the vocabulary and this is only practicable for very small vocabularies (a few tens of words). For larger vocabularies, sub-word units must be used to ensure that there will be enough examples to accurately estimate each model. For English this tends to mean that *phonemes* (or phonemes in context) are used as the basic modelling unit. However, to avoid any confusion arising from the use of the term phoneme in linguistics, the neutral word *phone* will be used for this modelling unit. These represent a reasonable compromise between consistency (since much of the variability is due to consistent contextual effects) and trainability (since English requires only 40-50 phones, a modest amount of training data is needed to ensure complete coverage).

2.2.1 Phone Modelling

The training procedure uses a set of training examples which correspond to known models. The frames of these examples are aligned against the states of the corresponding model and estimates formed for the model parameters. The training continues by using the new parameters values to re-align the examples and produce better estimates for the model parameters in an iterative re-estimation process. This re-estimation process requires data of known class with known boundaries. Since the training data will normally consist of complete sentences, there are two ways in which the phone level models can be matched to the sentence level utterances. Either the utterance can be segmented into individual phones or the phone models concatenated to produce a model for the complete utterance. It is difficult to accurately segment an utterance at the phone level and so the second method is preferable. The construction of *composite* models which represent a complete utterance is facilitated by the addition of non-emitting entry and exit states to each phone model. This removes the need to construct a separate transition matrix for each composite model and allows the phone models to be used within the composite unchanged.

Constructing the composite model requires a phone level transcription for each utterance. Normally only a word level orthography will be available and consequently a dictionary is needed to convert this to phone or model level. Decisions must be taken as to where inter-word silences occur and, if multiple pronunciations are present in the dictionary, which pronunciation best matches each word. The resulting sequence is then recorded in a *label file* which can be used to construct the composite model for the file during re-estimation.

A choice must also be made of the topology for each phone model. This is not critical but two features are important.

- Left to right topology.

  Speech is an ordered sequence of sounds and individual phones have temporal structure which should be enforced by the model topology.

- Minimum Duration.

Different phones and even different occurrences of a single phone can have widely varying durations. However each phone (when it is actually present) tends to have a minimum duration and, for this and a variety of other reasons, best performance is obtained by ensuring that each speech model has to emit a certain minimum number of observations between its entry and exit states.

Taking these factors into account, the topology of the example model shown in figure 2.1 is used for the majority of phone models. However, silence (or other pauses between words) has little temporal structure and a more complicated topology is needed for these models. (In fact two separate models are used and silence modelling is explained in detail in chapter 6).

2.2.2 Hidden Markov Model Assumptions

Implicit in the structure of hidden Markov models are a set of assumptions about the structure of the process that they represent. However these are not necessarily true for speech signals.

The assumptions are that;

- The observations accurately represent the signal.

  Normally the observations take the form of some type of short term (10-50ms) spectra (either smoothed spectra, linear prediction coefficients or a derivative). These will not exactly represent the underlying speech. However, the speech will be nearly stationary over such short periods and the observations a reasonably accurate representation of the speech.

- The observations are independent of each other.

  The likelihood of generating each observation is dependent only upon the state and is independent of all other observations. This is not typically true of speech since the spectrum tends to change only slowly compared to the frame rate to ensure that the parameterised observations are an accurate representation of the original signal. However the effect of the resulting high degree of correlation between subsequent observations can be reduced by augmenting the observations with derivative (rate of change with respect to time) parameters. When these are used the correlation does not seem to have an adverse effect on the recognition and attempts to model the correlation explicitly, such as segmental models [61], have met with only limited success.

- The between state transition probabilities are constant.

  This implies that, if unconstrained, the number of consecutive frames generated from a single state would have an exponential distribution. This is not true for speech (for which a gamma distribution may be more appropriate [48]). However the transition probabilities and the duration distributions that they represent only make a small contribution to the total likelihood of an utterance, which is dominated by the likelihoods of the observations. This means that the observed duration distributions do not take the form of exponential distributions. Figure 2.2 shows the actual and parametric distributions for the duration of a typical state. The dominance of the observation likelihoods together with the high

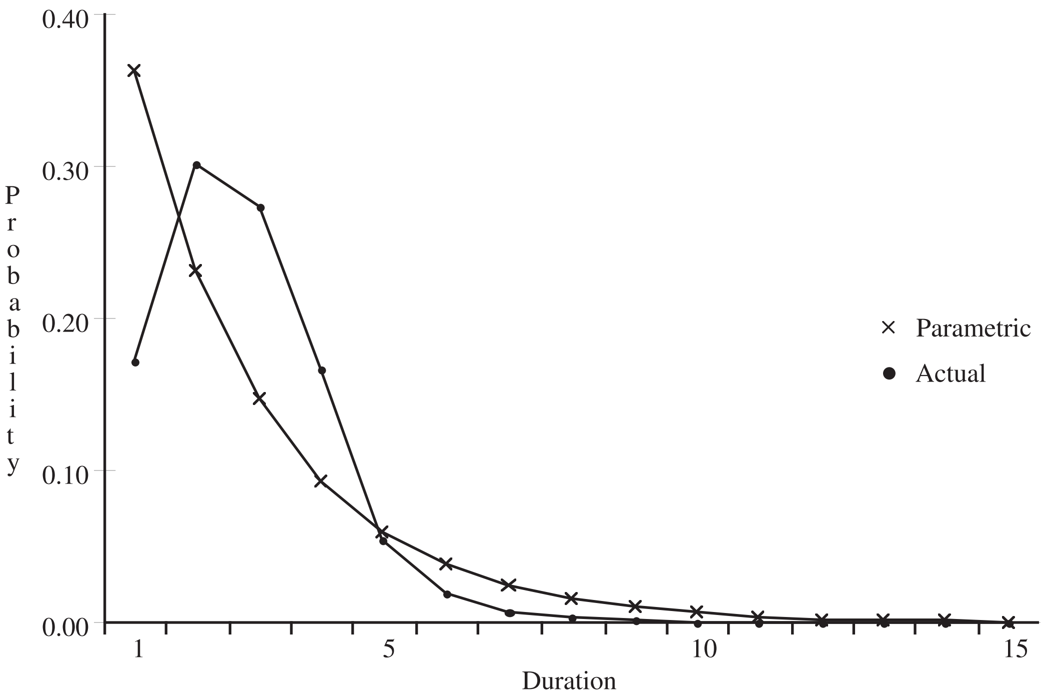

Figure 2.2: The duration distribution of a typical state.

degree of duration variability in normal speech has meant that the addition of duration models to HMM based recognisers [26] has met with only limited success.

2.3 Output Probability Distributions

Each emitting state of an HMM has an associated output probability distribution which determines the likelihood of the observations which are generated by that state. This distribution must be specific enough to allow discrimination between different sounds as well as being robust enough to allow for the expected variability inherent in natural speech.

The most commonly used distributions are,

- *Continuous*:

  A Gaussian or a mixture of Gaussian probability density functions,

$$\begin{aligned} b_{jm}(\mathbf{o}_t) &= N(\mathbf{o}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \\ &= \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_{jm}|}} e^{-\frac{1}{2}(\mathbf{o}_t-\boldsymbol{\mu}_{jm})'\boldsymbol{\Sigma}_{jm}^{-1}(\mathbf{o}_t-\boldsymbol{\mu}_{jm})}, \qquad (2.10) \\ b_j(\mathbf{o}_t) &= \sum_{m=1}^{M} c_{jm} b_{jm}(\mathbf{o}_t). \qquad (2.11) \end{aligned}$$

Where n is the dimensionality of the data and c_{jm} , $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$ are the *weights*, *means* and *covariances* of m^{th} component of the mixture Gaussian distribution from state j .

- *Discrete*:

  The observations are quantised into a number of symbols and each state has a discrete distribution that gives the probability of each symbol being generated by that state. The symbols are normally generated by a *vector quantiser* which assigns a symbol to each observation vector by choosing the closest example from a codebook. In order to adequately capture the variation in the observations it is often necessary to break the observation vector into several *streams*, each of which has its own codebook which generates a symbol for each observation.

$$b_j(\mathbf{o}_t) = \prod_{b=1}^{B} p_{jb}[v_b(\mathbf{o}_t)]. \tag{2.12}$$

  Here $v_b(\mathbf{o}_t)$ is the output of the vector quantiser using codebook b for observation $\mathbf{o}_t$ and $p_{jb}[v]$ is the probability of state j generating symbol v from the stream associated with b.

- *Tied-mixture*:

  All states share a common set of Gaussian probability density functions but each state has its own set of weights [10].

$$\begin{aligned} b_m(\mathbf{o}_t) &= N(\mathbf{o}_t; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \\ &= \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_j|}} e^{-\frac{1}{2}(\mathbf{o}_t-\boldsymbol{\mu}_m)'\boldsymbol{\Sigma}_m^{-1}(\mathbf{o}_t-\boldsymbol{\mu}_m)}, \end{aligned} \tag{2.13}$$

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{M} c_{jm} b_m(\mathbf{o}_t). \tag{2.14}$$

  Where c_{jm} is the state specific weight of the m^{th} shared Gaussian distribution with mean and covariance $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$.

During recognition and training, the calculation of many mixture Gaussian probability functions is computationally expensive [11]. This is especially true if distributions with full covariance matrices are used. As a result, many continuous density systems use diagonal covariance matrices, since this reduces both computational and storage requirements by a large factor. This will be a poor approximation unless the individual components of the observation vector are statistically independent. Performing a discrete cosine transformation on the log power spectrum ensures that the individual components are nearly independent and mixture Gaussian distributions can capture the most important inter-dependencies as well as multiple modes in the data.

The use of discrete or tied mixture distributions greatly reduces this computation and so much of the early work in speech recognition was based on these. However comparative results have shown that systems based on continuous density models typically yield higher accuracy [65]. The work in this thesis is based on systems using continuous density models (using output probability distributions consisting of mixtures of diagonal covariance Gaussians).

2.4 State Sequence Estimation

Both training and recognition procedures require estimation of the 'best' state sequence. For training this is needed to form new estimates of the model parameters and for recognition the likelihood of the path is used to decide between alternative recognition hypotheses.

The state sequence and its likelihood can be found in one of two ways;

- Probabilistically, using *total likelihood.*

  This calculates the likelihood of an utterance summed over all possible state sequences and finds the posterior probability of each observation being generated by each state. The posterior probability that a state was occupied at a particular time (and emitted the associated observation) is called the *occupancy.*

- Deterministically, using *maximum likelihood.*

  This calculates the likelihood of the most likely state sequence and for this sequence finds which state generated each observation.

2.4.1 Probabilistic state sequence estimation

The probability $\gamma_j(t)$ that a particular observation, $\mathbf{o}_t$, was generated from state j can be found using the *Forward-Backward* algorithm. This is an iterative procedure requiring two passes to calculate:

- $\alpha_j(t) = Pr(\mathbf{o}_1, ..., \mathbf{o}_t, x(t) = j)$.

  The *forward* likelihood of generating the observations from time 1 to t and the model ending in state j at t.

- $\beta_j(t) = Pr(\mathbf{o}_{t+1}, ..., \mathbf{o}_T | x(t) = j)$.

  The *backward* likelihood of generating the observations from time $t+1$ to T if the model was in state j at time t.

These can be efficiently calculated with the following iterations

$$\begin{aligned} \alpha_j(1) &= a_{1j} b_j(\mathbf{o}_1), \\ \alpha_j(1 < t \leq T) &= \sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} b_j(\mathbf{o}_t), \\ \alpha_N(T^+) &= \sum_{i=2}^{N-1} \alpha_i(T) a_{iN}, \end{aligned} \tag{2.15}$$

and

$$\begin{aligned} \beta_i(T) &= a_{iN}, \\ \beta_i(1 \leq t < T) &= \sum_{j=2}^{N-1} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_j(t+1), \\ \beta_1(1^-) &= \sum_{j=2}^{N-1} a_{1j} b_j(\mathbf{o}_1) \beta_j(1). \end{aligned} \tag{2.16}$$

The start and end conditions of these iterations enforce the condition that the model starts from state 1 before the first observation (1^-) and ends in state N after the final one (T+). Direct calculation of these quantities leads to numerical underflow and so either the logarithms of the likelihoods are used in the calculations or some form of normalisation must be performed after each frame.

The addition of non-emitting *confluent* entry and exit states in composite models complicates these calculations although their basic form is unchanged. Full equations for this case can be found in [97].

We wish to calculate

$$\begin{aligned} \gamma_j(t) &= Prob(x(t) = j|\mathbf{o}_1, ..., \mathbf{o}_T) \\ &= \frac{Pr(\mathbf{o}_1, ..., \mathbf{o}_T, x(t) = j)}{Pr(\mathbf{o}_1, ..., \mathbf{o}_T)}. \end{aligned} \tag{2.17}$$

From the definitions above

$$L = \alpha_N(T^+) = \beta_1(1^-) \quad = \quad Pr(\mathbf{o}_1, ..., \mathbf{o}_T), \tag{2.18}$$

and

$$\alpha_j(t)\beta_j(t) \quad = \quad Pr(\mathbf{o}_1, ..., \mathbf{o}_T, x(t) = j). \tag{2.19}$$

Hence,

$$\gamma_j(t) \quad = \quad \frac{\alpha_j(t)\beta_j(t)}{L}. \tag{2.20}$$

2.4.2 Deterministic state sequence estimation

The single most likely path through the model may also be required. This can be found using a similar iterative procedure to that above, except rather than summing the likelihoods over all paths, a decision is made and the most likely path chosen.

$$\begin{aligned} \phi_j(1) &= a_{1j} b_j(\mathbf{o}_1), \\ \phi_j(1 < t \leq T) &= \max_{i=2}^{N-1} \{\phi_i(t-1)a_{ij}\} \, b_j(\mathbf{o}_t), \\ \phi_N(T^+) &= \max_{i=2}^{N-1} \{\phi_i(T)a_{iN}\}. \end{aligned} \tag{2.21}$$

$\phi_j(t)$ is the likelihood of the most likely state sequence ending in state j at time t and its computation is often referred to as the *Viterbi* algorithm [87].

The state sequence is determined by a second *traceback* pass which examines the choices made at each frame to recover the most likely path. This requires that the choice made at every frame is recorded,

$$\begin{aligned} \chi_j(1 < t \leq T) &= \operatorname*{argmax}_{i=2}^{N-1} \{\phi_i(t-1)a_{ij}\}, \\ \chi_N(T^+) &= \operatorname*{argmax}_{i=2}^{N-1} \{\phi_i(T)a_{ij}\}. \end{aligned} \tag{2.22}$$

Pass	Beam width	Active States	Relative Time
Forwards	None	175.7	1.000
Forwards	500	20.8	0.143
Forwards	250	14.4	0.104
Backwards	20	3.1	0.025
Backwards	10	2.8	0.023

Table 2.1: Variation in computational complexity with pruning.

This allows the most likely state sequence $X = x(1), ...x(T)$ to be recovered by stepping backwards through these records (remembering that $x(1^-) = 1$ and $x(T^+) = N$).

$$\begin{aligned} x(T) &= \chi_N(T^+), \\ x(1 \leq t < T) &= \chi_{x(t+1)}(t+1). \end{aligned} \tag{2.23}$$

2.5 Pruning

The computation required to find the most likely state sequence, which is needed for both training and recognition, is significant, especially when the state space is large. As a consequence, a variety of techniques have been developed to reduce the total computation required.

The majority of these techniques focus on reducing (or *pruning*) the search space [50].

It can be empirically observed that the values of γ tend to be very small ($\ll 10^{-5}$) except for a few of the most likely states. Consequently only these few states have significant effects on the calculations and by assuming that the effects of the other states are insignificant, the computation required for the unlikely states avoided. This saving is implemented by *beam pruning* whereby only states whose likelihood is within a fixed ratio, called the *beam width*, of the most likely one are considered important and the likelihood of all others is assumed to be zero. This type of pruning is particularly important during recognition when the search space is larger.

When the most likely state is known (for instance when both $\alpha_j(t)$ and $\beta_j(t)$ have been found) it is possible to prune the search space very aggressively. However, in the above state sequence estimation algorithms, it is only possible to determine the most likely state sequence during the second pass. Once the the forward pass has been completed (to find $\alpha_j(t)$) the search space for the backward pass can be heavily pruned since the exact probability of occupying each state can be found from the product $\alpha_j(t)\beta_j(t)$. The beam width chosen needs to be of the order of the overall accuracy required. Say conservatively around 10^{-10} or as a natural logarithm around 20.

In order to apply pruning to the first pass it is necessary to 'guess' which are the most likely states. It turns out that it is reasonable to assume that for each frame the most likely states are those with the highest value of $\alpha(t)$ or $\phi(t)$ (independent of the remaining observations). This is an approximation and may lead to an inaccurate choice of most likely state. Consequently the beam width used in the first pass must be larger than that used in the second pass, when both α and β are known, to ensure that the most likely state is not pruned from the search during the initial pass. Typically its value, expressed as a natural logarithm, will need to be a factor of 10 to 100 times larger.

Although this means that the first pass takes an order of magnitude more computation than the second pass, it can still be an order of magnitude faster than a pass that does not use any pruning. Table 2.1 gives some example figures for the effect of pruning on both forward (first) and backward (second) passes. The numbers given are averages computed during training (when the size of the search space is limited to the number of states present in the sequence of phone models which represent each sentence). For recognition (when the sequence itself must be determined and so the search space much larger) the effect of pruning is more dramatic. Beam widths are expressed as natural logarithms and are chosen to minimise search errors due to pruning. The relative time gives an indication of the effect of pruning on the total computational required to estimate the state sequence.

2.6 Parameter Estimation

Once the state sequence has been found better estimates are needed for the model parameters. These values can then be used to find a more accurate state sequence and iteratively refine the models. Two sets of parameters need to be estimated;

- The output probability distributions $b_j(o)$ for each state j. For continuous mixture Gaussian probability distributions, estimates are needed for c_{jm}, $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$ from equations 2.10 and 2.11.
- The transition probabilities a_{ij} between states.

Maximum likelihood estimates for these parameters take the form of averages of, either the proportion of times a particular event took place (for probabilities) or the data (for parameters) [9],[35].

Maximum likelihood estimates for the output probability distributions are given by

$$\hat{c}_{jm} = \frac{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{jm}^{e}(t)}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{j}^{e}(t)}, \tag{2.24}$$

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{jm}^{e}(t)\mathbf{o}_t^e}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{jm}^{e}(t)}, \tag{2.25}$$

$$\hat{\boldsymbol{\Sigma}}_{jm} = \frac{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{jm}^{e}(t)(\mathbf{o}_t^e - \hat{\boldsymbol{\mu}}_{jm})(\mathbf{o}_t^e - \hat{\boldsymbol{\mu}}_{jm})'}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{jm}^{e}(t)}, \tag{2.26}$$

where $\hat{\boldsymbol{\mu}}_{jm}$ and $\hat{\boldsymbol{\Sigma}}_{jm}$ are the mean and covariance of mixture m of state j , $\mathbf{o}_t^e$ is the t^{th} frame of example e of the training data and $\gamma_{jm}^e(t)$ is the probability that observation $\mathbf{o}_t$ of example e was produced by mixture m of state j given by

$$\gamma_{jm}(t) = \gamma_j(t) \frac{c_{jm} b_{jm}(\mathbf{o}_t)}{b_j(\mathbf{o}_t)}. \tag{2.27}$$

Here $\gamma_j(t)$ is the probability that $\mathbf{o}_t$ was generated by state j. This would normally be calculated by the forward-backward algorithm outlined above and the whole parameter re-estimation process is referred to as *Baum-Welch* re-estimation.

The sum $\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_j^e(t)$ gives an estimate of the number of observations that are averaged to estimate the parameters for state j and is referred to as the *state occupancy*.

New estimates for the transition probabilities are given by

$$\hat{a}_{ij} = \frac{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_i^e(t) \, (a_{ij} b_j(\mathbf{o}_{t+1}) \beta_j(t+1) / \beta_i(t))}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_i^e(t)}. \tag{2.28}$$

Calculation of these values is complicated by the use of composite models with non-emitting entry and exit states but the basic form is unchanged. Full equations can be found in [97].

2.7 Recognition

Recognition is accomplished by selecting the model (or composite model) $\hat{X}$ that best fits an unknown observation sequence $\mathbf{O}$.

$$\hat{X} = \operatorname*{argmax}_{X} \{Pr(X|\mathbf{O})\}. \tag{2.29}$$

Applying Bayes' rule gives

$$Pr(X|\mathbf{O}) = \frac{Pr(\mathbf{O}|X) Pr(X)}{Pr(\mathbf{O})}, \tag{2.30}$$

and since the denominator is constant over X combining these equations gives

$$\hat{X} = \operatorname*{argmax}_{X} \{Pr(\mathbf{O}|X) Pr(X)\}. \tag{2.31}$$

The evaluation of $Pr(\mathbf{O}|X)$ for a known (composite) model was covered in the section 2.4. But one of the main reasons for the success of hidden Markov models in continuous speech recognition is the existence of computationally tractable methods for evaluating $\hat{X}$ over arbitrary model sequences. This procedure, *decoding*, is vital for viable speech recognition and some of the techniques used are described in chapter 4.

The set X over which the product is maximised controls what can be recognised and will be defined by some form of syntax. This can vary from choosing between a few simple sequences of models to choosing a path through a network first defining words as sequences of phone models and then a sentence as an arbitrary sequence of words. The value of $Pr(X)$ allows us to assign prior probabilities to the various hypotheses and is usually referred to as the *language model*. This can vary from a simple null model in which all strings are equally likely, to huge n-gram models with millions of parameters.

2.8 Summary

This chapter has outlined the way in which hidden Markov models are used for speech recognition. Further chapters will develop these ideas and the calculations described will form the basis of the algorithms for both decision tree construction (chapter 3) and recognition (chapters 4 and 5).

Chapter 3

Context Dependency in Speech

This chapter begins by describing the types of context dependency that arise in speech and how these may be exploited to improve the accuracy of speech recognition systems. It outlines several implementations of context dependency including the use of decision trees. It then describes a computationally efficient decision tree based method for producing context dependent systems using continuous density models.

The work has built on research carried out for an MPhil in Computer Speech and Language Processing (1992) into the use of decision trees to cluster phonetic contexts [57]. This presented work on small scale decision tree experiments using the TIMIT database. These techniques have been developed to give improved accuracy and increased computational efficiency to allow their application to larger databases.

The final method developed has enabled the construction of continuous speech recognition systems for unlimited vocabulary with, as results presented in chapter 6 will show, state of the art performance.

3.1 Contextual Variation

In order to maximise the accuracy of hidden Markov model based speech recognition systems it is necessary to carefully tailor their architecture to ensure that they exploit the strengths of hidden Markov models whilst minimising the effects of their weaknesses.

In practice, this means that the signal parameterisation and model structure are chosen to allow accurate prediction of the observations using mixture Gaussian state probability distributions. In order for these to accurately represent and recognise speech, it is necessary to ensure that their between class variance is higher than the within class variance.

This is accomplished in two ways.

- The signal parameterisation and state probability distributions are chosen to ensure that sounds that are perceived (or that are intended to be perceived) as similar result in similar observations (according to the probability distributions used).

- The classes are chosen to ensure that they can be accurately represented by the hidden Markov models.

The use of context is concerned with the second of these methods. The parameterisation chosen (*mel-frequency cepstral coefficients* see section A.1) has been used extensively [16] and has proved to be one of the best choices for large vocabulary continuous speech recognition. Much of the variability inherent in speech is due to contextual effects and, by taking these contextual effects into account, the variability can be reduced and the accuracy of the models increased.

These contextual factors can be, broadly, split into two levels;

- Session effects.

  These factors are constant over a single session with the recogniser and can be split into speaker and environmental effects. Often the environment can be controlled by minimising the background noise and ensuring that the same microphone is used. This leaves the differences between speakers as the major source of variation at this level.

- Local effects.

  These concern variations within an utterance. The most pronounced effects are due to co-articulation but other prosodical factors such as stress and emphasis can be important.

3.2 Session Effects

Results have shown that a well trained speaker dependent system is significantly more accurate than a similar speaker independent system [30]. This is due to the fact that a single person's speech is relatively consistent and that much of the acoustic variation in speech occurs across speakers. Tailoring a recognition system to a particular speaker enables it to capture effects due to;

- Gender and age.

  There are marked differences between the speech of men, women and children due to consistent differences in the size and development of the vocal organs.

- Dialect

  Consistent differences exist between the speech of speakers of the same language who are from different regions, both within a country and often more markedly between different countries that (supposedly) share a language.

- Style

  Each person tends to have their own speaking style and a speaker dependent system may be able to capture some of the peculiarities of a particular person.

When a recogniser must operate in a speaker independent fashion and be ready to accept anybody's speech the system will not initially be suited to a new speaker. However some of the benefits of speaker dependent systems can be gained by either

- Operating recognisers in parallel.

  Rather than using a single recogniser optimised to give the best performance for any speaker it is possible to use, in parallel, a set of recognisers which have been tuned to particular types of speaker.

  If

  - this tuning can improve the performance for the particular speaker type, and
  - new speakers can be assigned to use the correct recogniser,

  then this procedure will give an overall improvement in accuracy.

  These conditions can only be met for the more pronounced differences between speakers such as gender or dialect.

  Each recogniser is then trained on the data specific to its type of speaker or, to improve robustness, trained on all the available data and then tuned to the more specific data. The resulting recognisers are initially run in parallel and the speaker type chosen by determining which system best matches the new data (by finding the hypothesis with the highest likelihood). This gives both the answer and the speaker type. Once the speaker type has been confidently (and correctly) estimated only a single system needs to be used.

  This may seem impractical as initially the computational load appears to rises linearly with the number of systems, however, in practice one system tends to dominate quickly and the computational load is high for only the first few seconds of speech.

  Figure 3.1 shows how this would be implemented for a typical gender dependent system with two recognisers, one optimised for male speech and the other for female speech.

- Adapting the recogniser to match the new speaker.

  The initial model set is altered, or *adapted*, to better match the current speaker. Once tentative guesses for the first few sentences have been made (and possibly corrected by the user) these sentences, the *adaptation data*, can be used to estimate the characteristics of the speaker and the system's models adjusted accordingly. If the sentences are corrected before being used for adaptation this procedure is described as *supervised*, otherwise it is *unsupervised*. Over time, the performance of the system improves as the models become better matched to the speaker and ultimately a speaker independent system is transformed into a speaker dependent one. Figure 3.2 shows how this operates.

  Adaptation can be accomplished by estimating new parameters for the models directly from the data [22] but this will only update the parameters of the models for which there are examples in the adaptation data. These approaches are therefore only applicable when there is sufficient adaptation data from the new speaker to allow a substantial proportion of the system to be updated. This may be the case if a speaker enrolment procedure is used and new speakers provide several minutes of speech from a text specifically chosen to ensure coverage for most of the system.

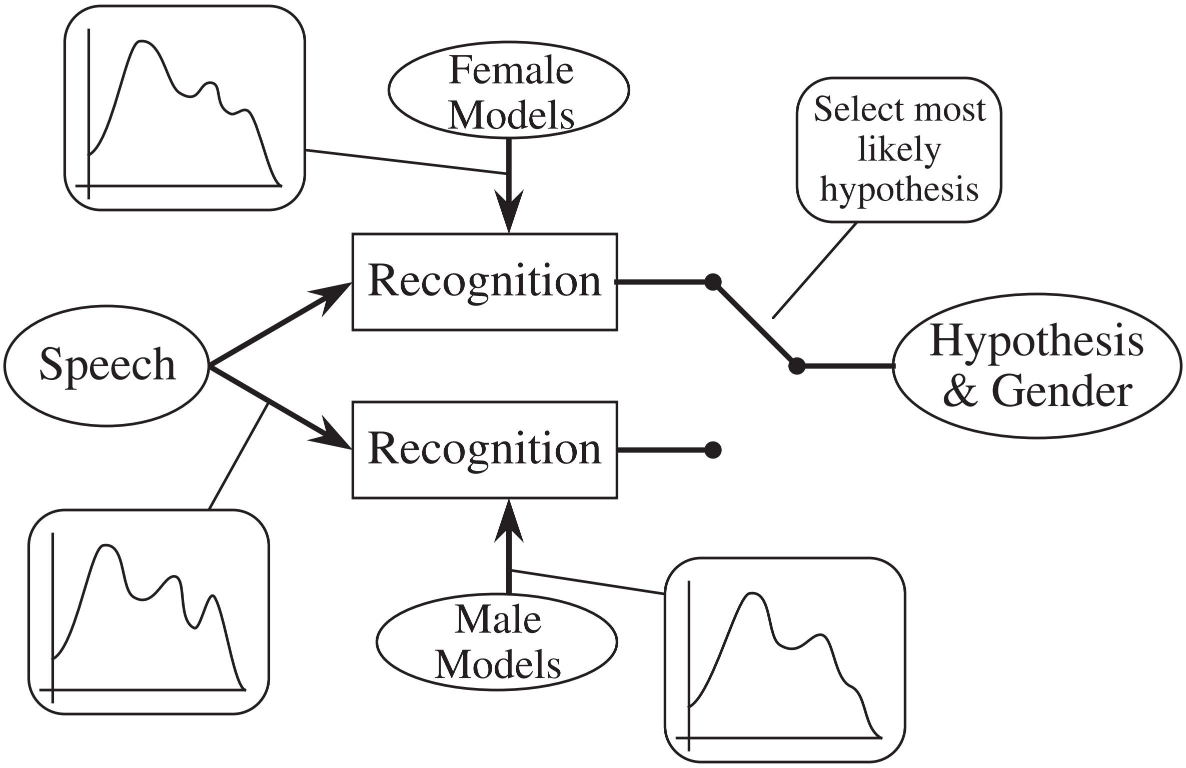

Figure 3.1: A gender dependent system using parallel recognisers.

To adapt large systems with millions of parameters from smaller amounts of data, it is insufficient to update just the models for which there are explicit examples in the adaptation data. To obtain the maximum performance increase from the small amount of data, it is necessary to generalise from the examples that occur and adapt all the parameters in the system.

Maximum likelihood linear regression [47] estimates a set of linear transformations that are used to update the means of continuous density output probability distributions. These transformations are chosen to maximise the likelihood of the adaptation data. The technique has proved suitable for adapting large systems using small amounts of data and has been incorporated into some of the recognition systems described in chapter 6.

The way in which a recogniser is used determines which of these techniques will perform best. Adaptation cannot be used if each new speaker only uses the recogniser for a few sentences as there is no chance to reliably learn their characteristics. Also care must be taken if the speaker can change without warning because the system will become adapted to the previous speaker and may perform poorly for a different one. A single sentence may well be enough to determine which of a set of parallel systems best matches the new speaker and so the parallel approach can almost always be used.

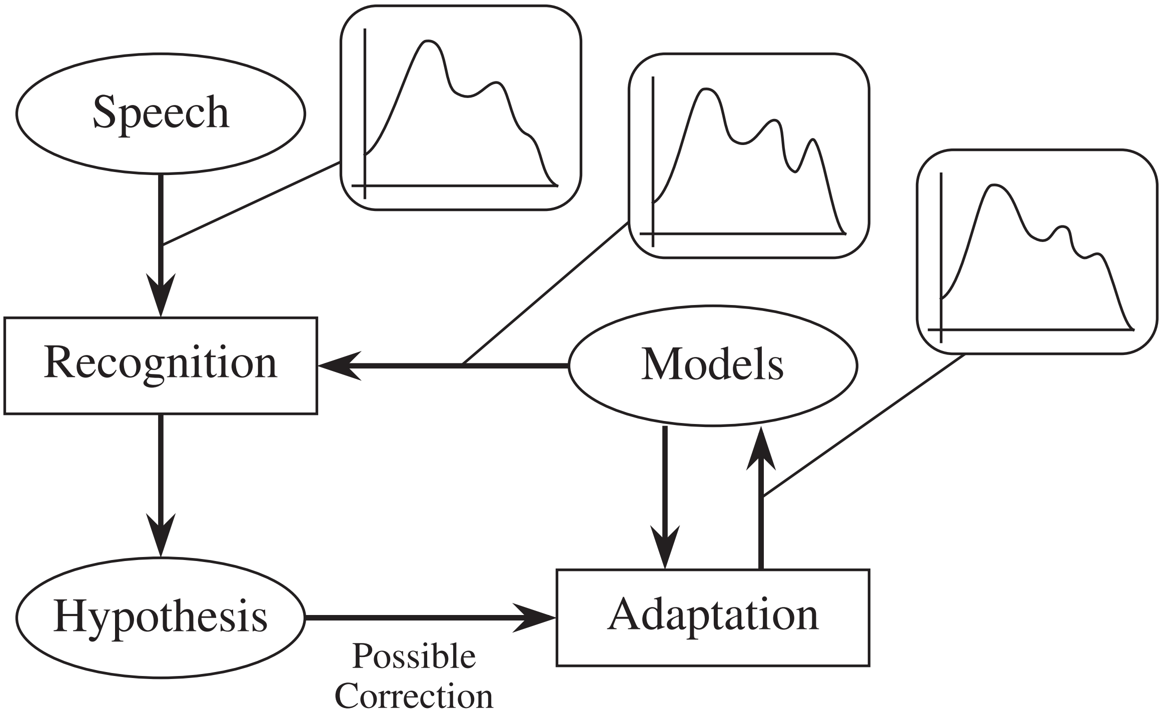

Figure 3.2: A speaker adaptation scheme.

It is also possible to make use of both techniques and initially use parallel systems to choose the speaker characteristics, then, once enough data is available, adapt the chosen system to better match the speaker.

3.3 Local Effects

Since speech is produced by physical articulators, which are not suited to drastic and sudden movement, the acoustic realisation of a particular phone is heavily influenced by the preceding and following positions of the articulators. This effect is called *co-articulation*.

The vocal articulators can only move at relatively slow speeds and so do not remain in constant positions throughout the duration of a phone. They are always in motion, moving from the position required to articulate the previous phone towards the position required for the next phone, via the position needed for the current phone. Often, especially during fluent speech, the articulators may not even reach their target positions and phones will only be partially articulated.

For human listeners this does not present a problem but for stochastic speech recognisers relying on the invariance of each phone this is a cause of poor performance.

Co-articulation means that the acoustic realisation of a phone in a particular phonetic context is more consistent than the same phone occurring in a variety of contexts. Figure

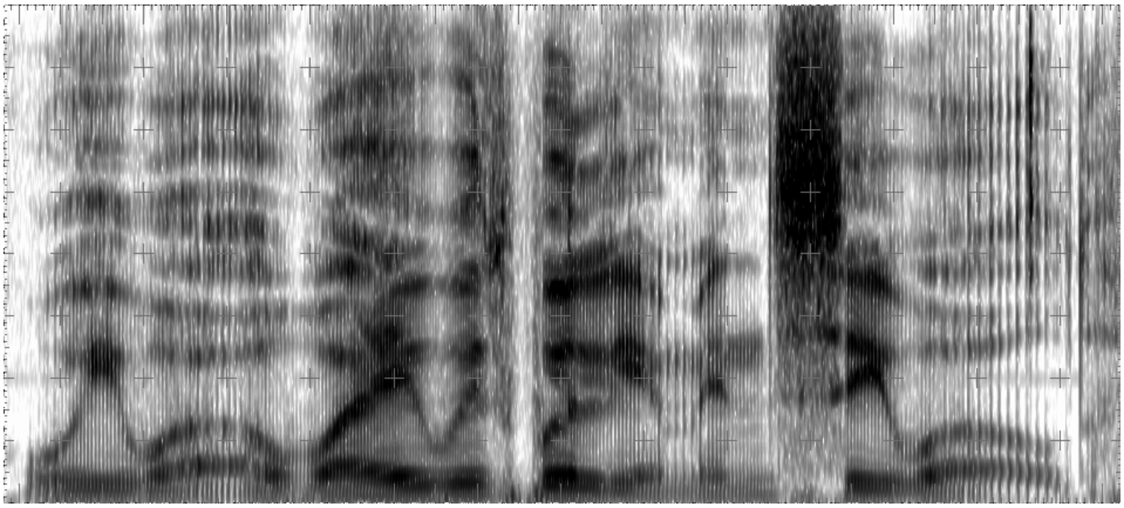

Figure 3.3: Context variability of the phone `w`.

3.3 shows a spectrogram for the phrase "We were away with William in Sea World". Each realisation of the `w` phone varies considerably but the most similar are the two occurrences in the same triphone context (underlined).

Recognition systems that capture this systematic variability will perform better than those that do not simply because the models will provide a more consistent and accurate representation of speech [43].

3.3.1 Context Dependent Phonetic Models

One of the ways in which this systematic variability can be captured is through the use of context dependent phonetic models. If the position of the articulators can be inferred from the identity of the phone then the co-articulatory effects could be predicted from the identities of the surrounding phones.

Hence models specific to both a phone and the phonetic context in which it occurs capture these effects and give more accurate performance. These context dependent phonetic models can be labelled using the standard HTK convention so `a-b+c` is an occurrence of phone `b` with `a` as its immediate predecessor and `c` immediately following it. Both `a-` and `+c` are optional so `b`, `a-b`, `b+c` and `a-b+c` are all occurrences of the phone `b` but with differing amounts of context specified.

The amount of context can vary. The most commonly considered values are;

- Monophone context.

  A single model represents a phone in all contexts. For the LIMSI phone set from appendix B, this means that a total of 45 models (plus silence) are required and the word *STEAK* will be represented by the model string,

$STEAK =$ `sil s t ey k sil` .

- Biphone context.

  Each model represents a phone with a particular left (or right) context. For the 45 phone set (plus silence), there are a total of 2071 possible models. For example,

  $STEAK =$ `sil sil-s s-t t-ey ey-k sil` (left biphones) or

  $STEAK =$ `sil s+t t+ey ey+k k+sil sil` (right biphones).

- Triphone context.

  Each model represents a phone with specific left and right contexts. There are 95221 possible triphones for the 45 phone set.

  STEAK = `sil sil-s+t s-t+ey t-ey+k ey-k+sil sil`

- Arbitrary context.

  There is no limit to how far this process can be taken. In general the amount of context can be specified by the number of phones considered. If the preceding and following N phones are specified this will be referred to as $+/-N$ context. For example if the preceding and following two phones are specified (*quinphone* context) this is labelled as $+/-2$.

As the figures above show, the number of models increases rapidly with the amount of context specified. However these numbers are upper limits and many sequences of phones will occur rarely if ever in real speech. Beyond triphones it is not feasible to consider all distinct contexts explicitly, and often triphones are used as a convenient way of labelling contexts rather than having a distinct model for each triphone context.

3.3.2 Word Boundaries

The presence of word boundaries in the phone sequences complicates the use of context dependent phonetic models and this can be dealt with in one of two ways.

- Word Internal Context Dependency.

  Word boundaries represent a distinct context and further expansion of the context across word boundaries is blocked.

  $STEAK\ AND\ CHIPS =$ `sil s+t s-t+ey t-ey+k ey-k ae+n ae-n+d n-d`
  `ch+ih ch-ih+p ih-p+s p-s sil`

- Cross Word Context Dependency.

  Expansion of context can occur into surrounding words. The presence of word boundaries can be either ignored or used as additional contextual information.

STEAK AND CHIPS = `sil sil-s+t s-t+ey t-ey+k ey-k+ae k-ae+n ae-n+d n-d+ch d-ch+ih ch-ih+p ih-p+s p-s+sil sil`

In continuous speech (as opposed to isolated word speech with each word delimited by silence) co-articulatory effects occur across word boundaries since these often have little, if any, acoustic significance.

However there are several reasons why it is easier to implement a system that only uses contextual information available from within the word.

- Size.

  The total number of contexts is much smaller than in the cross word case because many contexts will never appear in a word. This means that a greater proportion of contexts will be seen in the training data and the problem of unseen contexts is of less importance. The total number of contexts depends on the dictionary but, for a twenty-six thousand word Wall Street Journal dictionary (the Wall Street Journal database is described in appendix A), a word internal triphone system needs models for 14,300 distinct contexts whereas a cross word triphone system requires over 54,400 models. Note, however, that only 22,804 of these appear in the *SI284* training data.

- Complexity.

  With a word internal system every realisation of a word is the same and so can be taken straight from a dictionary. With cross word context dependency this is not true and, in the case of a triphone based system, the choice of the first and last models of each word depends upon the preceding and following words. As chapter 4 will explain this greatly complicates the decoding process.

The effect of these computational problems can be minimised by using suitable techniques and the use of cross word context dependency can lead to increased system accuracy [29]. These increases in modelling accuracy become more important as task constraints are relaxed and the recognition system is required to distinguish between more words. This is especially true when the speaker's style is more fluent as the assumption that each word can be modelled separately becomes increasingly poor. For best performance, large vocabulary continuous speech recognition must accurately account for consistent contextual effects independent of the position of word boundaries.

3.4 Trainability

To maximise the performance of a hidden Markov model based recogniser it is necessary to strike a balance between the level of detail of the models (controlled by the number of parameters in the system) and the ability to accurately estimate those parameters from the data.

Best accuracy would be expected from the most detailed models, but only if they accurately represent the data. Consequently it is necessary to ensure that all the parameters of the model are representative of speech in general. Thus it is not feasible to estimate a parameter from

Phone	Examples	Distinct Contexts		Phone	Examples	Distinct Contexts	
		$+/-1$	$+/-2^*$			$+/-1$	$+/-2^*$
Vowels				Plosives			
aa	43825	341	8463	b	46105	588	9469
ae	71905	454	16590	d	101054	1034	19980
ah	45146	273	7720	g	19165	478	4087
ao	35290	295	8488	k	97017	861	16453
aw	11648	201	2812	p	68054	750	11647
ax	176455	790	32639	t	191656	1135	36363
axr	64592	953	15976	Fricatives			
ay	36338	520	8627	dh	56735	284	10537
eh	78607	414	12530	th	13218	375	2777
er	16436	370	3830	f	49646	732	11515
ey	52824	716	13279	v	46569	557	8710
ih	145302	465	25020	s	145234	1021	26206
ix	19989	43	3707	sh	24290	406	3054
iy	100048	850	23104	z	83458	865	19161
ow	33337	558	8150	zh	1801	34	179
oy	4949	99	915	Affricates			
uh	9566	85	2442	ch	11187	422	2722
uw	39125	358	9376	jh	15496	461	3757
Glides				Nasals			
l	79982	797	15709	m	75612	789	12628
el	22509	564	6315	em	458	37	134
r	118750	597	20504	n	194493	892	32570
w	39678	429	8080	en	6280	151	1454
y	22485	177	3303	ng	25299	169	6485
hh	28946	412	6485	Total	2763559	22804	502981

Table 3.1: Variation in phone and context occurrences

only a few examples and this is especially true of variance parameters. In practice, somewhere between ten and a thousand examples are required to robustly estimate a single mean and variance.

Table 3.1 shows the variation in phoneme occurrences for the LIMSI phone set (described in appendix B) in the *SI284* section of the Wall Street Journal Database (appendix A). It also shows the number of different contexts in which each phone occurs. The $+/-1$ column is the number of different triphone contexts and the $+/-2^*$ column is the number of different contexts when the preceding and following two phones, the position of word boundaries and the speaker's gender are considered. This table shows that the distribution of examples in the training data (and speech in general) is not uniform and the number of examples of different phonemes varies by over two orders of magnitude.

Despite the relatively large size of this database, the average number of examples of each $+/-2^{*}$ context is only 5.5 and for $+/-1$ it is 121.2. It is impractical to train a separate model from only a few occurrences especially if mixture Gaussian distributions are to be used. Since there are many triphone contexts which occur only a few times in the training data and many more that do not occur at all, special efforts must be made to ensure that a triphone system is *trainable* and that its parameters can be estimated reliably. This problem becomes even more acute if larger amounts of context are to be taken into account.

There are several ways in which the trainability of a system can be increased;

- Backing-Off

  When there is insufficient data to train a given model it is possible to *back-off* and use a less specific model for which there is enough data. For example, a biphone model could substitute for a triphone which had only a few examples in the training data. If there were few occurrences of that biphone, a monophone model could be used. This guarantees that every model used is well trained but it can mean that relatively few models will have full triphone context especially if the training data is relatively sparse (which is normally the case when cross word triphones are used).

- Smoothing

  In order to maintain a greater degree of context dependency it is possible to smooth the parameters of the more specific model with those of the less specific model. One way in which this can be accomplished [45] is to use interpolation between the less and more specific models (with the interpolation weights chosen using deleted interpolation [34]). This preserves the context dependency of the unsmoothed models but increases their robustness by effectively sharing training data from other contexts to produce more accurate parameters.

- Sharing

  Another method for increasing the robustness of the system is to explicitly share models or parts of models between different contexts [93]. This is sensible since the acoustic realisations of a phone occurring in different contexts are often very similar. This method also ensures that all the system parameters are well trained whilst maintaining the model's context dependency.

All these techniques require that a choice is made about which parameters are backed-off to, smoothed with or shared with others. For a simple backing-off strategy a simple and obvious hierarchy exists; triphones are more specific and less trainable than biphones which are more specific and less trainable than monophones. However this is also the weakness of the scheme because these are big jumps in specificity. Similarly, smoothing of parameters must occur through some form of hierarchy. However more flexibility is possible since different parts of a model can be smoothed in variable proportions with different models. For instance, the initial state can be smoothed with a left biphone to preserve as much left context dependency as possible, whilst the final state can be smoothed with the corresponding right biphone.

Finally sharing presents even more possibilities, parameter sharing between models of the same complexity is possible as well as sharing with models further up the hierarchy. Sharing schemes can be divided into two approaches.

- Bottom-up Approaches

  These initially assume that all contexts are different. Then a merging process is used to produce more trainable but less specific models.

- Top-down Approaches

  These initially assume that all contexts are the same and are grouped. Then a splitting procedure is used to produce more specific and more accurate models.

3.5 Bottom-up Approaches

Bottom-up approaches to the data insufficiency problem start by assuming that all contexts are distinct but to ensure that the parameters of each model can be reliably estimated some form of sharing or smoothing is required. This is accomplished by examining the original models and determining sets that can share parameters. These sets are chosen to ensure that the resulting models can be estimated robustly and that members of the sets are sufficiently similar to ensure that the models will provide accurate representations.

In practice, the first condition is met by ensuring that the number of examples in each set exceeds a threshold and the second, by ensuring that parameters of the original models were sufficiently similar.

3.5.1 Generalised Triphones

One way of implementing parameter sharing is to compare models from different triphone contexts and merge those which are most similar. The merged models will be estimated from more data and, if the realisations of the phone in the different contexts are similar, this will give more accurate models (as well as a reduction in the total size of the system). [43] shows, for discrete distribution models, that the resulting *generalised triphones* show better performance than triphones smoothed with less specific models using deleted interpolation.

However sharing at the model level may not be the most appropriate method for models composed of distinct states.

3.5.2 State Clustering

Sharing distributions at the state level allows finer distinctions to be made between models by allowing left and right contexts to be modelled separately. Extending the above method for constructing generalised triphones to cluster discrete distributions at the state rather than model level resulted in further increases in recognition accuracy [32].

Previous work using continuous distribution models [91] has used a method of model building that shares the output probability distributions amongst states. This sharing is constrained so that distributions are specific to a particular state position in a particular phone and they are only shared amongst the same state occurring in different contexts. Hence, for the architecture with three emitting states, the distribution clustering procedure is performed three times for each phone in the phone set.

The clustering is performed on single Gaussian diagonal covariance models in two stages;

- First is an iterative merging procedure which merges the most similar pair of distributions, chosen because they have the minimum distance, $d(i,j)$ between them. This stage terminates when this minimum distance exceeds a predetermined threshold.
- A second merging procedure ensures the trainability of the models by ensuring that the occupancy γ of each tied distribution exceeds some threshold. Each distribution with an occupancy below this threshold is merged with the nearest distribution (with the minimum value of $d(i,j)$).

The distance between distributions (which will initially be for a single context and later for a cluster of contexts) i and j is calculated using

$$d(i,j) \quad = \quad \left[\frac{1}{n} \sum_{k=1}^{n} \frac{(\boldsymbol{\mu}_{ik} - \boldsymbol{\mu}_{jk})^2}{\boldsymbol{\sigma}_{ik}\boldsymbol{\sigma}_{jk}} \right]^{\frac{1}{2}} \tag{3.1}$$

Where n is the dimensionality of the data and $\boldsymbol{\mu}_{sk}$ and $\boldsymbol{\sigma}_{sk}$ are the mean and the variance of the k^{th} dimension of the Gaussian distribution of state s (either i or j). The values of γ_s for the untied distributions are calculated during training (see 2.6) and then summed to give the occupancy after merging.

This procedure results in a set of models with Gaussian probability distributions for clusters of contexts with similar acoustic features and ensures that each distribution has enough training examples to accurately estimate its parameters. This reduces the total number of distributions (and thus system size) significantly but results in a much smaller reduction in the number of distinct models because different models may share two state distributions and only differ in the final one. This preserves a higher degree of context dependency by allowing for contextual factors that only effect parts of a phone. For instance, models with the same right context but different left contexts may have different initial state distributions whilst sharing those for the final and centre states.

The complexity of the system (and the accuracy of the models) is then increased in a step by step fashion by increasing, in lock-step, the number of components in each of the tied distributions. This is done by a splitting procedure in which the mixture component with the largest weight c_{sm} in each shared distribution s is duplicated. The resulting identical components have their means perturbed from the original values by 0.2 standard deviations, their weights halved and their variances left unchanged. The resulting system is then retrained using Baum-Welch reestimation. This mixture splitting is then repeated as necessary to smoothly increase the complexity of the system. This procedure is referred to as *up-mixing* and figure 3.4 shows how this is implemented.

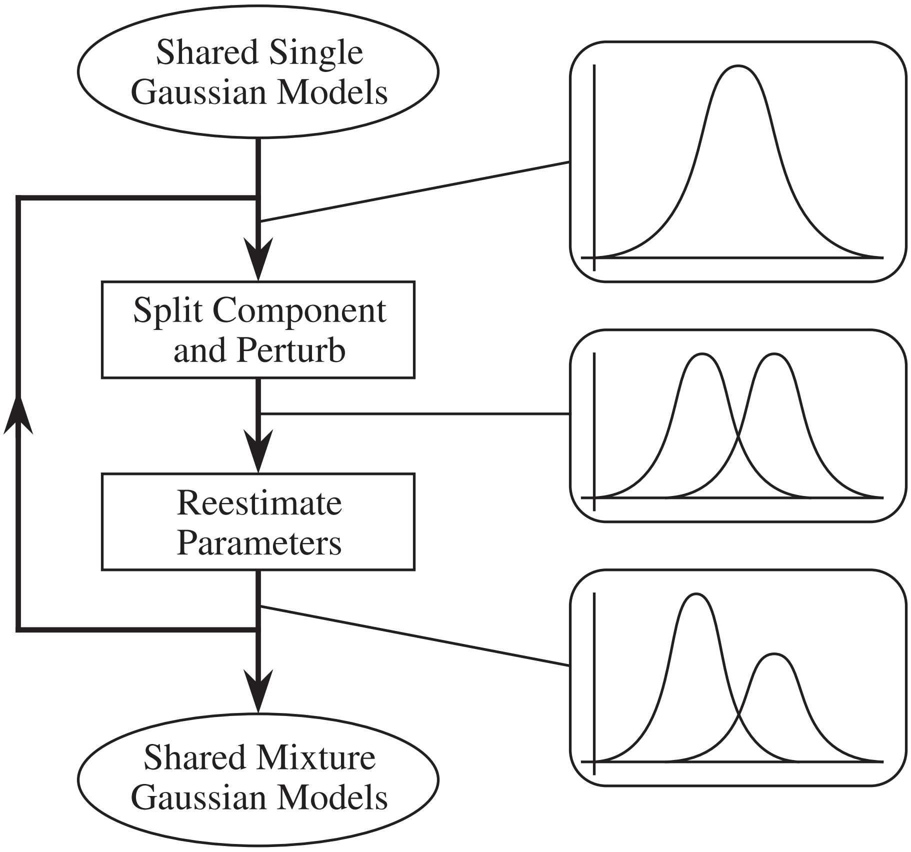

Figure 3.4: Up-mixing: Increasing system complexity.

This procedure is a robust way of increasing the system complexity until performance on development test data peaks or some predetermined level of system complexity is reached.

3.5.3 Top Down Approaches

The bottom up approaches are limited as they require examples of each context to produce initial estimates of the model parameters used in the clustering procedure. This makes it impossible to use such methods to construct models for contexts that may occur during recognition but do not appear in the training data. It is also unreliable for contexts that only occur a few times, since the examples may be unrepresentative and so the parameter estimates used during clustering will be inaccurate. Instead a simple backing-off procedure must be used and less specific models substituted for the *unseen* models.

This problem can be minimised by ensuring that the training data gives adequate coverage of the models needed for recognition. But this is possible only for small vocabularies and systems using word internal context dependency. For larger vocabularies and for cross word context dependent systems, it is virtually impossible to ensure that the training data will include examples of every possible context.

For example, despite the relatively large size of the Wall Street Journal Database, only 22,804 of a possible 95,221 triphones appear in the *SI284* training data and only 14,545 appear at least ten times.

The accuracy of such a recognition system will be severely compromised unless better estimates can be found for the parameters of the unseen models. When this is not possible, there may be no benefit in using a system with cross word context dependency because so many of the models will have been backed-off to use biphone or monophone models.

Using a top down clustering procedure based on decision trees avoids the problem of unseen models by using linguistic knowledge together with the training data to decide which contexts (including the unseen ones) are acoustically similar.

A decision tree for each phoneme selects which of a set of models is used in each context. The model is chosen by traversing the tree, starting from the root node then selecting the next node depending upon the answer to a simple question about the current context. For binary decision trees these questions will normally be yes/no questions concerning membership of particular sets of phones.

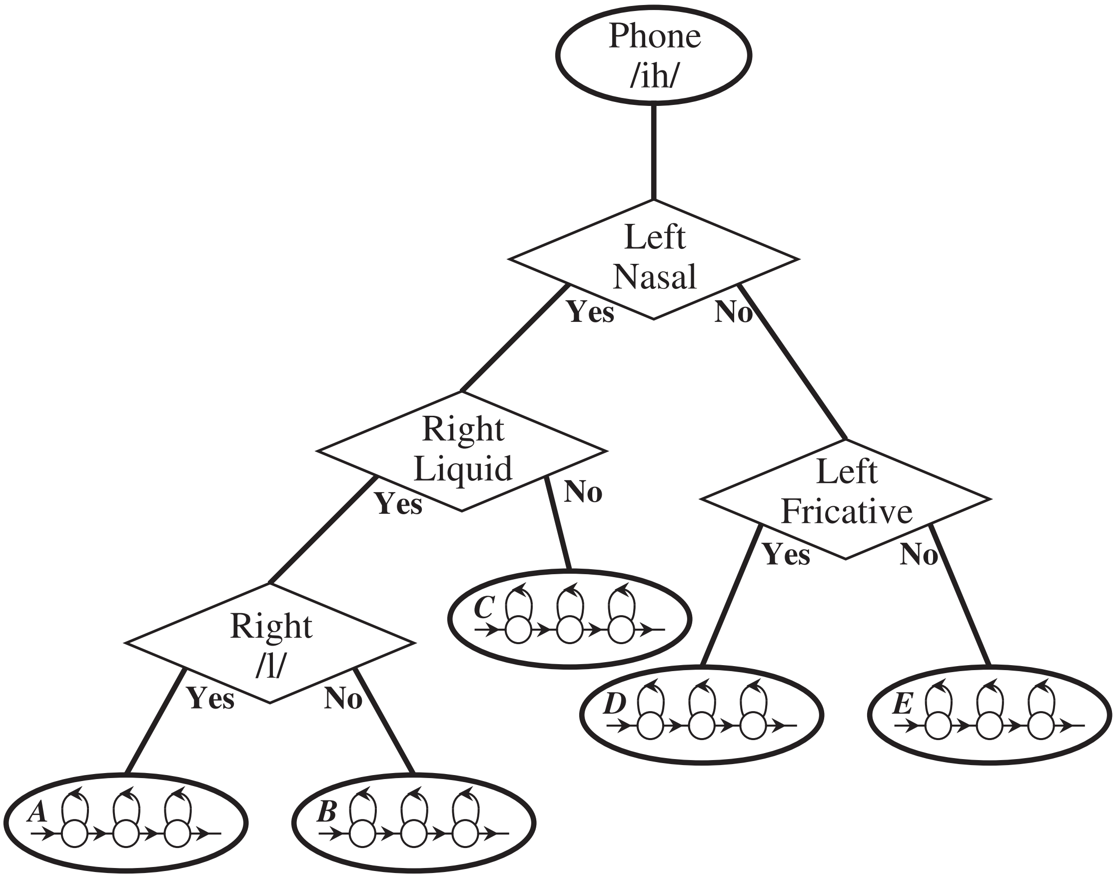

Figure 3.5: A decision tree for `ih`.

For example, in the decision tree shown in figure 3.5, the root question is answered by checking to see if the immediately preceding phone (the left context) is a nasal (`n`, `en`, `m`, `n`, `em` or `ng`). If the actual context was `m-ih+t` the next question to be asked would concern whether the following phone was a liquid (`l`, `el`, `r`, `w`, `y` or `hh`). Since `t` is not a member of this set and the answer no results in a terminal node, the model labelled `C` would be used in this context.

This procedure has several advantages over bottom up clustering

- The hierarchical structure and the form of the questions means that the tree will find an equally context dependent model for every context. This removes the need to back-off to less specific models for contexts that have not occurred in the training data.
- Expert knowledge can be incorporated in the form of the set of questions that are used to split each node of the tree and this will be used to determine which contexts are similar to any unseen ones.
- The construction procedure can be constrained to ensure that leaf nodes are only generated for sets of contexts that have sufficient examples in the training data to reliably train an accurate model. This removes the need to apply this constraint separately and the clustering does not suffer from the use of badly under-trained parameters.
- A greater degree of context dependency than triphones can be implemented by extending the type of questions. For example, they could refer to wider contexts or to different features. Careful construction of the tree will ensure that only contextual effects which lead to consistent variation will be considered.

3.6 Decision Trees

To exploit the advantages of the top-down decision tree based approach, it is necessary to be able to automatically construct the trees. The construction procedure should aim to ensure that the resultant set of models provide an accurate and robust estimate of the underlying speech. [7] developed methods for constructing decision trees for discrete distribution models.

The trees were constructed in locally optimal fashion starting from a single root node representing all contexts. As each node is created, an optimal question chosen from a finite set is selected to maximise the increase in probability of a Poisson model at the resultant terminal nodes generating the training examples. Then the current set of terminal nodes is searched to find the one which can be split using its optimum question to provide the largest increase in the total probability of the training data. Provided that this increase exceeds a threshold and that the number of training examples associated with the node exceeds a threshold, the node is divided using the optimal question and two new terminal nodes created. When none of the terminal nodes can be split the procedure terminates and the tree is finished.

This algorithm would be suitable for constructing decision trees for continuous density models if suitable criteria for choosing questions could be found for continuous parameters rather than VQ symbols.

Initial work [57] attempted to maximise the self similarity of the groups of examples produced by each question using a *dynamic time warping* based distance measure [82]. A similar method was used in [19]. This was computationally very expensive to evaluate since it required the calculation and storage of the distance between every pair of examples in each group (calculated using dynamic time warping). These values were averaged over all pairs of members in the group to give a measure of the group's self similarity.

This technique scaled very badly to larger tasks as the calculation and storage of the distances grew with the square of the number of examples of each phone (since the initial group contained all examples), as did the evaluation of the self similarity of each group. This technique was therefore only suited to very small scale experiments.

To improve scalability with increasing amounts of data, a template based approach is needed to ensure that the computation and storage requirements only grows in a linear fashion with the number of examples of each phone.

In informal experiments, it was found that the distances calculated using dynamic time warping could be quite accurately approximated using just a linear time warping. This is probably due to the short duration of the phone level segments (averaging only 7 frames). Using linear time warping it is computationally simple to estimate both a fixed length template and the distance between each example and the template. More importantly, the computational and storage requirements grow linearly with the size of the database and so larger databases can be used.

However, both of these techniques required phone aligned data and were not very well matched to the hidden Markov models used in the recognition system constructed from the resultant decision trees. This led to relatively poor performance because phone examples that appeared similar using this procedure could not necessarily be represented accurately by a single hidden Markov model. Other results [31] had also suggested that sharing at the state distribution rather than at the model level led to improved performance. This approach also has the benefit of simplicity since the underlying model topology is used and there is no need to perform additional alignment (such as linear or dynamic time warping) whilst constructing the trees.

The next section describes the decision tree clustering procedure used for generating the recognition systems described in chapter 6. The clustering procedure is closely tied to the structure of the models and so is simpler, quicker and more accurate than the previous techniques based on clustering direct from the data.

3.7 Decision Tree Construction

There are a number of aims for the decision tree construction procedure;

- Each leaf must have a minimum number of examples (a minimum occupancy) to ensure that the parameters of the final models can be estimated accurately.

- A finite set of questions can be used to divide each node. This constrains the way each node may be divided but allows the incorporation of expert knowledge needed to predict contextual similarity when little or no data is available to determine which contexts are acoustically similar.

- Hidden Markov models should be able to accurately capture the variability of the terminal nodes. For this work, mixture Gaussian probability distributions should be able to accurately represent the training examples at each terminal node.

These criteria mean that careful choices must be made for the set of questions that can be asked at each node of the tree and for the way in which the effects of using each questions is evaluated.

The questions are chosen on the basis of linguistic knowledge that suggests certain phones may produce certain types of articulatory effect. Appendix B contains the set of questions used in this work. They represent a maximal set since it was found that the addition of extra linguistically motivated questions did not degrade performance whilst restricting the set to just a few of the major features did. The questions are symmetric because there were no clear reasons for supposing that preceding contextual factors would be different from following ones.

The first of the aims listed above can be satisfied by restricting the choice of question to those that ensure that any nodes created have a sufficient number of associated examples in the training data. This restricted set of questions is searched in an effort to maximise the accuracy of the resultant hidden Markov models. Ideally this means attempting to minimise the within class variance whilst maximising the between class variance. As previously mentioned such discriminative schemes tend to be computationally expensive because the choice of parameters for each class depends on the choices made for every other class.

A simpler scheme which attempts to maximise the accuracy of the models with respect to their own class is thus more suitable. This can be accomplished using a maximum likelihood approach which is attractive since it is well matched to the way the parameters of models are subsequently estimated. A variance clustering technique using a similar likelihood based clustering approach is described in [36].

In fact, decision tree based system building can be viewed as the constrained maximum likelihood optimisation of the architecture of the system followed by a maximum likelihood optimisation of the parameters in the resulting system.

3.7.1 Likelihood Based Decision Criteria

Performing a full maximum likelihood training pass is computationally very expensive and this cannot be used to calculate the likelihoods of the numerous possible architectures. However, an estimate of the log likelihood of the training data for a particular set of state distributions can be found in a computationally efficient manner if the following assumptions are made;

- The assignments of observations to states (that is to say the values of $\gamma_j(t)$) are not altered during the clustering procedure. In practice, careful choice of the initial state assignments ensures that any changes are not significant.

- The contribution of the transition probabilities to the total likelihood can be ignored. This is linked to the previous point, although the transition probabilities will have a significant effect on the total likelihood, their contribution would only change if changes occurred in the state assignments. These are assumed fixed throughout the clustering procedure and so the contribution of the transition probabilities is constant and unaffected by the clustering.

- The total likelihood of the data can be approximated by a simple average of the log likelihoods weighted by the probability of state occupancy. This is an approximation unless the values of γ are zero or one, as is the case for deterministic state assignments but is often nearly true for probabilistic assignments

Given these assumptions

$$\begin{aligned}\mathcal{L} &= \sum_{e=1}^{E}\sum_{t=1}^{T_e}\sum_{s\in S}\ln(Pr(\mathbf{o}_t^e;\boldsymbol{\mu}_s,\boldsymbol{\Sigma}_s))\gamma_s^e(t), &(3.2)\\ &\approx \ln(Pr(\mathbf{O};S)) &(3.3)\end{aligned}$$

is the approximate log likelihood of a set models comprising the set of distributions S generating the training data O consisting of E examples.

For simple Gaussian distributions

$$\begin{aligned}\ln(Pr(\mathbf{o}_t^e;\boldsymbol{\mu}_s,\boldsymbol{\Sigma}_s)) &= \ln\left(\frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_s|}}e^{-\frac{1}{2}(\mathbf{o}_t^e-\boldsymbol{\mu}_s)'\boldsymbol{\Sigma}_s^{-1}(\mathbf{o}_t^e-\boldsymbol{\mu}_s)}\right)\\ &= \ln\left(\frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_s|}}\right)-\frac{1}{2}(\mathbf{o}_t^e-\boldsymbol{\mu}_s)'\boldsymbol{\Sigma}_s^{-1}(\mathbf{o}_t^e-\boldsymbol{\mu}_s)\\ &= -\frac{1}{2}\left(n\ln(2\pi)+\ln(|\boldsymbol{\Sigma}_s|)+(\mathbf{o}_t^e-\boldsymbol{\mu}_s)'\boldsymbol{\Sigma}_s^{-1}(\mathbf{o}_t^e-\boldsymbol{\mu}_s)\right) \qquad (3.4)\end{aligned}$$

So

$$\mathcal{L} = \sum_{e=1}^{E}\sum_{t=1}^{T_e}\sum_{s\in S}-\frac{1}{2}\left(n\ln(2\pi)+\ln(|\boldsymbol{\Sigma}_s|)+(\mathbf{o}_t^e-\boldsymbol{\mu}_s)'\boldsymbol{\Sigma}_s^{-1}(\mathbf{o}_t^e-\boldsymbol{\mu}_s)\right)\gamma_s^e(t) \qquad (3.5)$$

However the parameter reestimation formula 2.26 gives

$$\boldsymbol{\Sigma}_s = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_s^e(t)(\mathbf{o}_t^e-\boldsymbol{\mu}_s)(\mathbf{o}_t^e-\boldsymbol{\mu}_s)'}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_s^e(t)}$$

So

$$\sum_{e=1}^{E}\sum_{t=1}^{T_e}(\mathbf{o}_t^e-\boldsymbol{\mu}_s)'\boldsymbol{\Sigma}_s^{-1}(\mathbf{o}_t^e-\boldsymbol{\mu}_s)\gamma_s^e(t) = n\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_s^e(t) \qquad (3.6)$$

This gives

$$\mathcal{L} = \sum_{s\in S}-\frac{1}{2}\left(n(1+\ln(2\pi))+\ln(|\boldsymbol{\Sigma}_s|)\right)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_s^e(t) \qquad (3.7)$$

Often the value of $\boldsymbol{\Sigma}_s$ will not be calculated directly from the data but from the statistics from each unique context c. This reduces both the storage and computational requirements since the number of different contexts is (often substantially) smaller than the number of examples.

In this case

$$\begin{aligned}
\boldsymbol{\Sigma}_s &= E[\mathbf{o}^2] - E[\mathbf{o}]^2 \\
&= \frac{\sum\limits_{c\in C(s)} \gamma_c(\boldsymbol{\Sigma}_c + \boldsymbol{\mu}_c.\boldsymbol{\mu}_c{}')}{\sum\limits_{c\in C(s)} \gamma_c} - \left(\frac{\sum\limits_{c\in C(s)} \gamma_c\boldsymbol{\mu}_c}{\sum\limits_{c\in C(s)} \gamma_c}\right) . \left(\frac{\sum\limits_{c\in C(s)} \gamma_c\boldsymbol{\mu}_c}{\sum\limits_{c\in C(s)} \gamma_c}\right)' \qquad (3.8)
\end{aligned}$$

Where $C(s)$ is the set of contexts that are to be represented by the distribution of tied state s. The parameters for each context are calculated from the data using

$$\gamma_c = \sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_c^e(t) \qquad (3.9)$$

$$\boldsymbol{\mu}_c = \frac{\sum\limits_{e=1}^{E}\sum\limits_{t=1}^{T_e} \gamma_c^e(t)\mathbf{o}_t^e}{\gamma_c} \qquad (3.10)$$

$$\boldsymbol{\Sigma}_c = \frac{\sum\limits_{e=1}^{E}\sum\limits_{t=1}^{T_e} \gamma_c^e(t)(\mathbf{o}_t^e - \boldsymbol{\mu}_c)(\mathbf{o}_t^e - \boldsymbol{\mu}_c)'}{\gamma_c} \qquad (3.11)$$

Here $\gamma_c^e(t)$ is the probability, for the context c, of state occupation at time t of example e.

Building the decision tree consists of changing the set of distributions S to maximise the total likelihood of the training data whilst ensuring sufficient data is available to train each resulting distribution. Because splitting a given distribution is assumed to have no effect on the remaining distributions only the local improvement in the total likelihood need be calculated. Splitting a node changes the set of distributions S by replacing the parent p distribution with a set of descendants D.

The total likelihood in this case is given by

$$\begin{aligned}
\mathcal{L} &= -\sum_{s\in S, s\neq p} \frac{1}{2}\left(n(1+\ln(2\pi)) + \ln(|\boldsymbol{\Sigma}_s|)\right) \sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_s^e(t) \\
&\quad - \sum_{d\in D} \frac{1}{2}\left(n(1+\ln(2\pi)) + \ln(|\boldsymbol{\Sigma}_d|)\right) \sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_d^e(t) \qquad (3.12)
\end{aligned}$$

So the change in overall log likelihood, which is the quantity that needs to be maximised, is just the difference between the likelihood of the parent and its descendants (all other parts of the system are unaffected).

$$\begin{aligned}
\delta\mathcal{L} &= -\sum_{d\in D} \frac{1}{2}\ln(|\boldsymbol{\Sigma}_d|) \sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_d^e(t) \\
&\quad + \frac{1}{2}\ln(|\boldsymbol{\Sigma}_p|) \sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_p^e(t) \qquad (3.13)
\end{aligned}$$

This only requires the calculation of $\boldsymbol{\Sigma}_d$ and $\gamma_d = \sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_d^e(t)$ for each of the descendants (as presumably the parent's values will have already been calculated). The constraint on trainability is implemented by assuming $\delta\mathcal{L} = 0$ when γ_d falls below a threshold. The complexity of these operations depends on the number of examples or classes present in the parent node and should scale well when large amounts of data are used. This is particularly true if the examples have already been assigned to a smaller number of contextual classes and equations 3.8 to 3.11 are used to calculate the required values from the class means, occupancies and covariances.

A similar expression can be used to find the change in likelihood when a set of distributions D are merged to produce a single distribution m.

$$\begin{aligned} \delta\mathcal{L} &= -\frac{1}{2}\ln(|\boldsymbol{\Sigma}_m|)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_m^e(t) \\ &\quad + \sum_{d\in D}\frac{1}{2}\ln(|\boldsymbol{\Sigma}_d|)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_d^e(t) \end{aligned} \tag{3.14}$$

Similarly if models (consisting of a set of state distributions) are to be clustered (to produce generalised triphones) the likelihoods are summed over all three states to give $\delta\mathcal{L}$ and a single decision tree is generated for each base phone.

3.8 Implementation

3.8.1 State Assignment

An initial assignment of observations to states is needed to calculate the above likelihood from the values of γ_c, $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ for each distinct context c. For the bottom up state clustering procedure described in [91] a set of single gaussian triphone models are trained using Baum-Welch re-estimation to provide statistics about each context.

This is simple to implement and provides the required statistics. However, it does have two drawbacks.

- The state assignments may not be representative.

  Untied triphones with Gaussian distributions are used to state align the data in order to gather statistics about different contexts. These models can be both severely undertrained, because few examples occur of a particular context occur in the training data and inaccurate, because a Gaussian distribution may not be adequate to capture the variability of the data.

- The number of models required may be very large.

  When a greater degree of context dependency than simple triphones is desired the need for a model for each distinct context may be unrealisable. For instance, in the WSJ *SI284* database there are over half a million distinct contexts when the previous and following two phonemes, the position of word boundaries and the gender of the speaker are taken into consideration.

Consequently a better procedure for obtaining the necessary statistics was designed.

Firstly, a well trained system is used to perform a deterministic state alignment of the data and this alignment is stored in the form of label files. The system chosen should produce representative alignments and this can be checked by ensuring that its recognition accuracy is relatively high. In the early stages of system development, a well trained context independent system is used and later, when a tied state context dependent system is available this can be used. The assignment does not have to be deterministic but the storage requirements for probabilistic assignments can become prohibitive and it is not clear that they would be significantly more accurate.

These state alignments are used to collect from the training data the statistics which are necessary for the decision tree construction. Since each tree can be constructed independently this leads to a dramatic reduction in the number of different contexts for which data is required at any one time.

State assignment was performed using well trained tied state models and untied single Gaussian ones. In both cases, systems with similar performance were produced and so it seems that the clustering procedure is robust with respect to the initial state assignments.

3.8.2 Tree Construction

Each tree is built top-down in an iterative fashion by sequentially splitting the distribution that results in the largest increase in likelihood. This is found for each node by evaluating the increase in likelihood for each allowable question. This procedure is only locally optimal but constructing a globally optimal decision tree is a computationally intractable problem and may not be the best solution. The constraints that are placed on the tree construction in the form of the questions used to split each node contain extra information which is important for deciding what happens to unseen contexts. If attempts are made to produce globally optimal trees by using more complex questions at each node this could compromise the ability to construct accurate models for unseen contexts.

In preliminary experiments, attempts to produce more optimal trees by allowing both merging and splitting to occur during tree building had little effect on the objective function for trees with more than a few terminal nodes. However, this did greatly complicate the structure of the trees and so was not pursued any further.

The only exception is the addition of a final iterative merging pass in which the decrease in likelihood caused by merging pairs of terminal distributions is calculated and whilst the minimum value falls below a threshold the distributions are merged so that those terminal nodes share the same distribution.

In informal experiments the addition of this final pass reduced the total number of system parameters without effecting the initial recognition accuracy and the parameter reduction should lead to increased system robustness.

This algorithm is summarised diagrammatically in figure 3.6.

Part of this procedure is determining which of the set of questions results in the largest increase in log likelihood. Since each node can only be split using a small number of questions, a blind search in which the change in likelihood is calculated for every possible question is

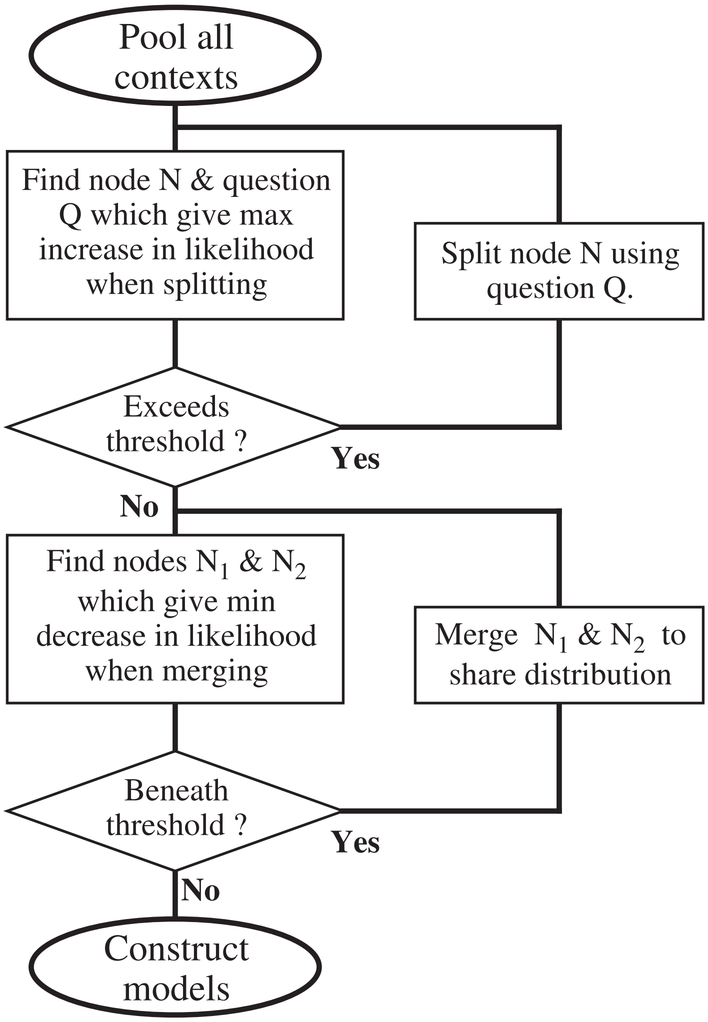

Figure 3.6: Algorithm for constructing decision trees.

computationally tractable. This can be made more efficient however by observing that many of the questions divide the data on the basis of the same contextual unit. For instance there will be many questions concerning the identity of the immediately preceding phone. Since there are only a fixed number of equivalence classes for each contextual unit (in this case the phone set), the data can be initially split into these classes and intermediate values computed for γ, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. This simplifies the calculations because the evaluation for each question about left context requires summing values over the classes rather than the greater number of distinct contexts. This reduces the total computational required from $O(N_c N_q)$ to $O(N_c + N_p N_q)$ where N_c is the number of distinct contexts, N_q is the number of questions about left context and N_p the number of phones in the phone set. This can be a substantial saving when N_c is much greater than N_p.

3.8.3 Gender Differences

Preliminary experiments showed that there are marked differences between the acoustic parameters of speakers of different gender. When questions concerning the gender of the speaker are added to those available for tree construction these almost always appear near the root of the tree. This shows that there are large and consistent differences due to the speaker's gender.

Experiments were performed using systems that were gender independent, totally gender dependent (a gender determining question appeared at the root of each tree) and optionally gender dependent (where the gender determining question was available for use at any point in the tree). For the systems which were optionally gender dependent the gender determining question appeared often and the majority of states tended to be gender dependent (the few which were not tended to have relatively small amounts of training data available). This indicates that there are large differences between the speech of different genders. Despite these acoustic differences between genders, gender dependent systems did not, as might be expected, produce consistently more accurate results than gender independent systems [92],[90]. This may well be due to the acoustic differences being so large that imposing gender consistency across a sentence does not increase the accuracy enough to offset the reduced amount of data available to train the parameters of each system.

Because of this behaviour, the way in which gender dependent systems were produced was altered.

Gender dependent systems were produced by first cloning a gender independent system to produce identical male and female models. Then the means and weights of the mixture components were re-estimated separately from the gender specific data, whilst holding the component variances and transition probabilities at the gender independent values. The variances were not updated because of the reduced quantity of data available. The two systems were then run in parallel (occasionally together with the original gender independent system) for recognition.

This technique gave more consistent performance improvements but as results in chapter 6 will show gender dependent systems still failed to perform substantially better than comparable gender independent systems.

It was felt that, for gender independent trees, the large between gender differences in the acoustic parameters could result in biased choices of the best questions. If a particular set of contexts had more examples from speakers of a particular gender, the criteria used for choosing which question to split each node with could be heavily biased towards questions that would actually split the data on the basis of gender rather than actual phonetic contextual variations.

To prevent this use of 'virtual' gender questions, a procedure in which the male and female data were clustered separately was adopted for gender independent systems. Each shared distribution used separate gender dependent means and variances and the trees were built effectively using two component mixture Gaussian distributions at each node. This also meant that the data sufficiency requirements could be applied to each gender independently to ensure that the parameters of a gender dependent system cloned from the gender independent one could be estimated accurately.

3.8.4 Feature selection

Extending the types of questions that are available for use during the construction of the trees provides a well motivated way of assessing the usefulness of a particular feature.

For instance, results in [6] showed a substantial increase in system accuracy when, rather than taking into account only immediately preceding and following phones, the phonetic context was defined by the preceding and following five phones. Results in [44] show that the accuracy of a recognition system on the Resource Management task was increased by the addition of function word models. It is also possible that system accuracy may be increased by explicitly modelling the presence of word boundaries [25] or by specifically modelling dialect differences.

All of these approaches can be evaluated by extending the set of questions beyond those just examining the adjacent phones. The use of such questions during the construction of decision trees would give a reasonable indication if consistent differences are due to wider context, word identity, speaker identity or position of word boundaries.

If these questions are used extensively then a system incorporating these dependencies can be built and tested but if they are not used the computation this entails can be avoided. This gives a relatively quick method of deciding which contextual features may be important to a system.

3.9 Summary

This chapter has outlined a method for constructing decision trees for sharing output probability distributions amongst states of continuous density models. The decision trees enable the construction of systems using models incorporating more contextual information (such as long distance cross word phonetic contextual effects) for improved accuracy.

Experimental results presented in chapter 6 will show that this method provides state of the art performance for a variety of large vocabulary tasks.

Much of the increase in accuracy results from the ability to construct cross word context dependent models which without the decision tree clustering would otherwise be severely undertrained. However efficient use of these models during recognition requires improved decoding techniques and these will be discussed in the next two chapters.

Chapter 4

Decoding

This chapter describes several decoding techniques suitable for recognition of continuous speech using hidden Markov models. In particular it is concerned with the use of cross word context dependent acoustic and long span language models which are necessary to achieve maximum modelling accuracy.

It begins by specifying the requirements of an ideal decoder.

Standard time-synchronous network based decoders are described along with an implementation strategy based on the concept of token passing. The use of beam pruning and different architectures to increase the speed of decoding together with extensions for N-best decoding are outlined. The problems which arise when this architecture is extended to use large vocabularies, cross word context dependent acoustic and long span language models are also highlighted.

Several best first search techniques are then described, in particular A* searches and the stack decoder architecture implementation.

Finally a recogniser capable of using cross word context dependent acoustic and long span language models is described. This decoder is based on the stack decoder architecture but functions in a predominantly time-synchronous fashion.

4.1 Requirements

The requirements of an 'ideal' decoder are very simple, it should find the most likely grammatical hypothesis for an unknown utterance. The likelihood will be composed of several elements.

- Acoustic Model Likelihood.

  The likelihood of the utterance given by the hidden Markov acoustic models that represent the hypothesis.

- Language Model Likelihood.

  A likelihood based upon the probability of a hypothesis given the prior knowledge of likely sentences.

The grammar used may be complex, and specify a syntax for the whole sentence, or simple, and allow any word to follow any other word.

The total number of hypotheses for each utterance can be virtually infinite and so it is not possible to enumerate every hypothesis and calculate its likelihood to find the most likely. To ensure that recognition is computationally tractable it is necessary to share computation between the common portions of different hypotheses.

Sometimes the above knowledge sources may be augmented with other information such as duration models, pronunciation probabilities or deterministic language models. Sometimes these additional knowledge sources can be computationally expensive to apply without significantly increasing search locality. Consequently they are not incorporated directly into the search for the most likely hypotheses but are applied later to a subset of the search space. For instance they can be used to augment the likelihoods of the N most likely hypotheses and when these have similar acoustic and language model likelihoods the additional knowledge sources will be used to choose between them. To do this N-best rescoring, the decoder needs to generate a list of likely hypotheses for each utterance. This list can be generated explicitly or the decoder can produce a lattice of hypotheses with different word hypotheses spanning various portions of the utterance.

An ideal decoder would have the following characteristics;

- Efficiency.

  The computation required during decoding must be reasonable. What is considered reasonable may vary according to the task that the recogniser performs. For dictation or similar on-line tasks the decoding process must take place in better than real-time to ensure that the system does not lag behind the speaker. For system development, real-time operation is not required but the computation involved in testing a recognition system should not be out of proportion with that required for its construction.

- Accuracy.

  The ideal decoder would always find the most likely grammatical sequence of words for each utterance. If this is not the case, the decoder generates *search errors* in addition to modelling errors and overall system accuracy is reduced. To ensure that decoding is computationally tractable it may not be possible to guarantee that the most likely sequence of words is always found. However, a good decoder will ensure that the proportion of errors due to search is always relatively small.

- Scalability.

  Ideally a decoder will scale well as the task it performs and the models it uses become more complex. The increase in computation required to use more complex systems should be of the same order as the increase in accuracy that they afford. For instance, increasing the recognition vocabulary can reduce the error rate if the extra words are recognised correctly. However the accuracy will not rise linearly with the size of the vocabulary because the extra words are relatively uncommon. Hopefully the computation required by the decoder would also increase less than linearly with the size of the vocabulary.

- Versatility.

An ideal decoder would allow a variety of constraints and knowledge sources to be incorporated directly into the search without compromising its efficiency. For accurate continuous speech recognition this currently means using n-gram language and (cross-word) context dependent models.

The remainder of this chapter will describe some of the techniques used for decoding and how well they meet these requirements. Time-synchronous decoders, which parse the utterance one observation at a time, will be discussed first followed by best first decoders which initially pursue the most likely hypothesis and only later consider less likely alternatives.

4.2 Time-Synchronous Decoding

The method for finding the most likely state sequence through a composite model for a particular utterance, described in section 2.4.2, can be extended to choosing between words.

A simple isolated word recogniser can be produced by creating a composite model in which a sequence of models representing each word in the vocabulary are placed in parallel between utterance initial and final silence models. This structure is shown in black in figure 4.1 for a recogniser with a four word vocabulary (*AND BILL BIT BEN*) composed of monophone models.

The most likely state sequence is used to decide which word most closely matches the utterance by finding through which of the words the most likely path passed.

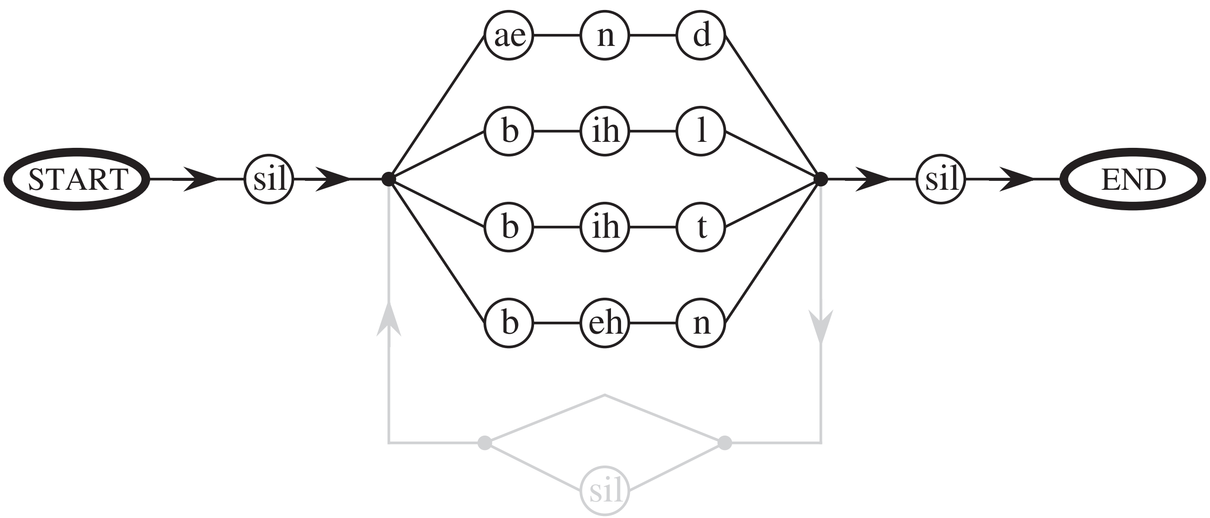

Figure 4.1: A composite model for word recognition.

This architecture can be extended to perform continuous speech recognition by adding a transition back through the composite model from the end to the start of the word models. This enables sequences of words to be recognised as the most likely path can pass through several words in turn. Figure 4.1 shows this extra extension in grey. It also includes an optional between word silence model (`sil`) to allow for any pauses that occur between words.

4.2.1 Token Passing

The most likely state sequence through such a network can be found using the *token passing* implementation of the Viterbi algorithm [96].

A single movable structure called a *token* holds the likelihood of each partial path, ϕ, together with the traceback information, χ, needed to recover its path through the network.

Equations 2.21 and 2.22 are implemented by moving these tokens between states of the *model instances* in the network. Each model instance holds a token in each of its states (including the non-emitting entry and exit states) which represents the most likely partial path ending in that state of the network at the current time.

For each observation in the utterance the token in each state of the network is updated using equations 2.21 and 2.22. The tokens in all states with transitions into the current state are examined to find the most likely one (given by $\mathrm{argmax}_{i=1}^{N-1}\{\phi_i(t-1)a_{ij}\}$). The token in the selected state i is updated with the transition probability and the output likelihood, the choice i recorded (to allow traceback of the token's path through the network) and the resulting token is then moved to the destination state before the next frame is processed.

The traceback path does not normally need to be recorded at the state level. When the most likely sequence of words is required from the recogniser it is sufficient to record only the decisions about transitions between words. Within words the decisions only effect state alignment and there is no need to record this information when only the word sequence is required. To make it easier to determine which transitions occur between words the network is expanded to explicitly identify the beginning and end of each word. Only tokens propagating between the end of one word and the beginning of the next have their traceback information updated.

At the end of the utterance the most likely sequence of words is recovered by traceback through the decisions made about transitions between words.

By expanding the network to include an explicit arc from the end of each word to the start of the next, a bigram language model can be incorporated to improve recognition accuracy. The language model provides a likelihood for each of the between word transitions based on the conditional probability of one word following another. The language model likelihood is given by

$$P_{HW} \quad = \quad (Prob(W|H))^{\omega} + \rho, \tag{4.1}$$

where $Prob(W|H)$ is the conditional probability of a word W following a partial hypothesis H, ω is a grammar scale factor used to optimise the relative weight of the acoustic and language models and ρ is a word insertion penalty used to control the ratio of insertion and deletion errors.

Figure 4.2 shows how the structure of the network is modified to allow the use of a bigram language model and to allow decisions to be recorded at the word rather than the state level by explicitly marking the beginning and end of words. Note that only the loop is shown, utterance

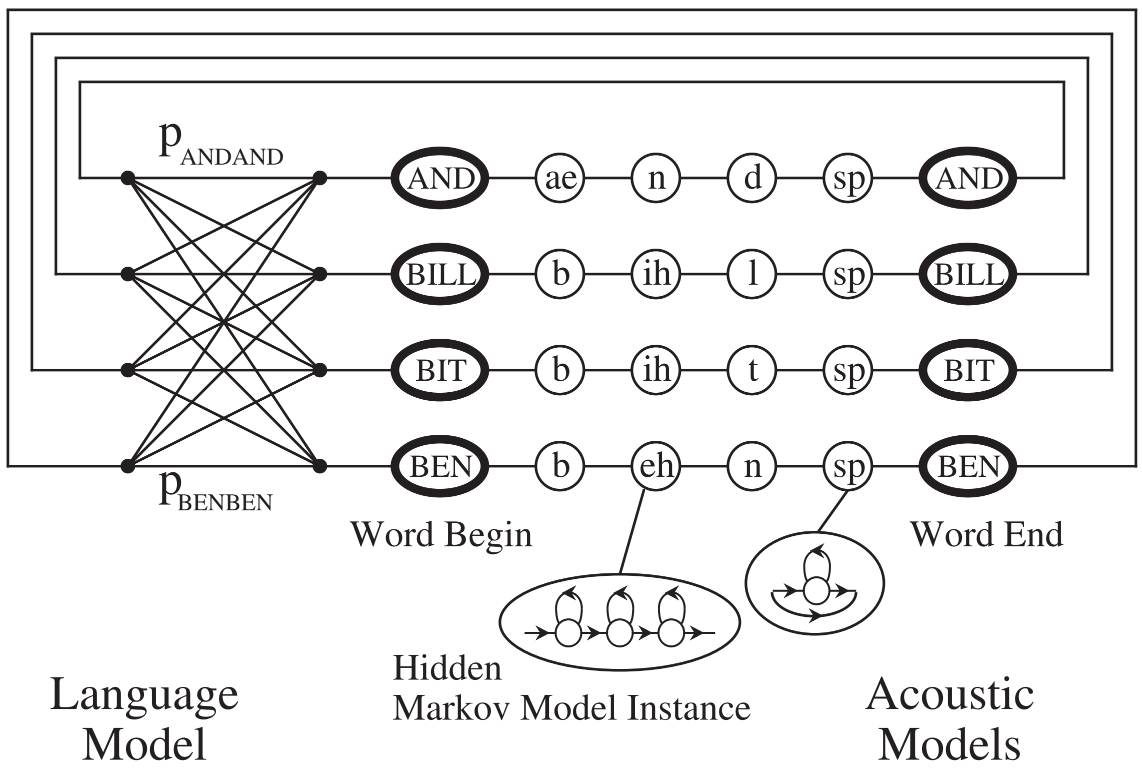

Figure 4.2: A bigram network.

initial and final silence models have been omitted. These would be included as shown in figure 4.1 with transitions from the utterance initial silence into the start of each word and from the end of each word into the utterance final silence. The language model can be used to supply a likelihood for each of the transitions.

Each word instance ends in an explicit optional short pause `sp` silence model to allow an optional period of silence to occur between words. It is no longer possible to share a single between word optional silence model amongst all word instances since the bigram language model requires an explicit transition from the end of each word to the start of the next. The figure shows how the model is made optional by adding a transition between its non-emitting entry and exit states. This makes it possible to traverse the model without consuming any observations. Such models will be referred to as *tee* models.

This architecture can be extended to use word internal context dependent models without expanding the network since a distinct copy of each word already exists in the network. Each word instance uses the context dependent models rather than the simple monophone models. For example, the word BEN represented by the sequence of monophone models `b eh n` in figure 4.2 would use instead the sequence `b+eh b-eh+n eh-n` of word internal context dependent triphone models.

Model position in word	First	Second	Last but one	Last	Word End
Number active	3539	866	265	91	43
Proportion active	65.4%	16.0%	4.9%	1.7%	0.8%
Relative computation	76.0%	18.6%	5.7%	1.9%	

Table 4.1: Variation of model activity over the network in beam pruned search.

4.2.2 Pruning

A search though a complete network is said to be *admissible* as it is guaranteed to find the most likely permissible state (and word) sequence. In such a case, no search errors will be made and the accuracy of the recogniser will be dependent only on the accuracy of the acoustic and language models.

However this search will waste a significant proportion of the total computation performing state alignment and calculating the likelihood of paths that are relatively unlikely. The beam pruning technique described in section 2.5 can significantly reduce the total computational requirements by reducing the size of the search space being actively considered [99].

Every frame, the most likely partial path, or token, in the network is found and its likelihood sets the top of a beam of fixed width (measured in log likelihood). Every active token in the network is examined and if its likelihood falls outside the beam the token is discarded and its likelihood is effectively set to zero. With conservative values for the beam width, few search errors will be made despite the search being *inadmissible* and no longer guaranteed to find the most likely sequence of words.

Search errors occur when a relatively unlikely partial path has a possible continuation significantly more likely than any of the more likely partial paths. If the beam width is too narrow and this partial path is pruned out of the search space, a search error is made and the most likely sequence of words will not be found.

Often paths will merge at the end of words. From this point, all partial paths will have the same continuation and only the relative likelihood difference up to the end of the current word will lead to search errors. When simple fully-connected word loops are used the beam width can be relatively narrow. However, when more complex grammars such as finite state ones spanning the whole utterance are used, the beam width needs to be higher. In this case, different partial paths can have extensions which are different for more of the utterance. This will lead to much larger relative differences in likelihood because the paths do not merge until later in the utterance.

A beam pruned search leads to a highly asymmetrical pattern of activity in the network [53]. Because of the high branching factor at word ends (which are often connected to many if not all of the word beginnings), there is significantly more uncertainty at the start of words. If a single word end is relatively likely, many word initial phone models will be active because of the high number of transitions from the word end to the start of words.

Table 4.1 shows the pattern of model activity during decoding for a five thousand word Wall Street Journal task. The recognition was performed using a back-off bigram language model and a set of word internal triphones. The number of active models in the first two and last two phones of each word is given (note that in the case of short words these overlap) as is the proportion of the models in that position within each word. Finally the proportion of the total computation required by the models in this position is shown. Pruning thresholds were chosen to ensure that only a small fraction of the overall word error rate was due to search errors. There is a steady decline in the proportion (and number) of active models throughout each word and evaluating the first two phones in each word is responsible for almost 95% of the computation.

This asymmetry means that the simple network structure (which is optimal for an unpruned search) may not be optimal for a beam pruned search. The different network architectures described in section 4.2.5 and chapter 5 are tailored to exploit the asymmetry to reduce the average number of active models in a beam pruned search by increasing the total size of the network and allowing sharing.

4.2.3 N-Best decoding

The token passing implementation of the Viterbi algorithm can be extended to perform N-best recognition by storing more than one token in each state.

Using N tokens in each state, with each token representing a different hypothesis, allows the decoder to find the N most likely hypotheses. Unfortunately the computation required scales almost linearly with the total number of hypotheses required. However, in practice it is not necessary to consider each of the N hypotheses individually since many will differ by only a couple words over the whole sentence. The computation for the similar portion of each hypothesis can be shared with only the differences between hypotheses needing multiple calculations.

By assuming that the end time of each word is independent of all previous ones, M tokens in each state can be used to store M hypotheses for the previous word. This assumption is called the *word pair* approximation [84]. Partial paths with the same final word are linked and share a single token with a single likelihood given by the most likely of the partial paths. However, the token holds traceback information for each of the merged partial paths. When the whole utterance has been processed traceback produces a network of word hypotheses with the branching factor controlled by the number of tokens stored in each state. The computation required again scales almost linearly with M but is much less than the exact sentence dependent method because smaller values of M can be used to produce a reasonable number of alternative hypotheses (N). Also the number of sentence hypotheses scales with the length of the sentence and this is a desirable characteristic since longer sentences will, on average, contain more errors.

4.2.4 Limitations

The network architecture described above is only suited to medium vocabulary systems using bigram language models and word internal context dependent models.

The size of the network and the computation involved in the search grows linearly with the size of the vocabulary. Consequently this type of search becomes impractical if the vocabulary exceeds a few thousand words.

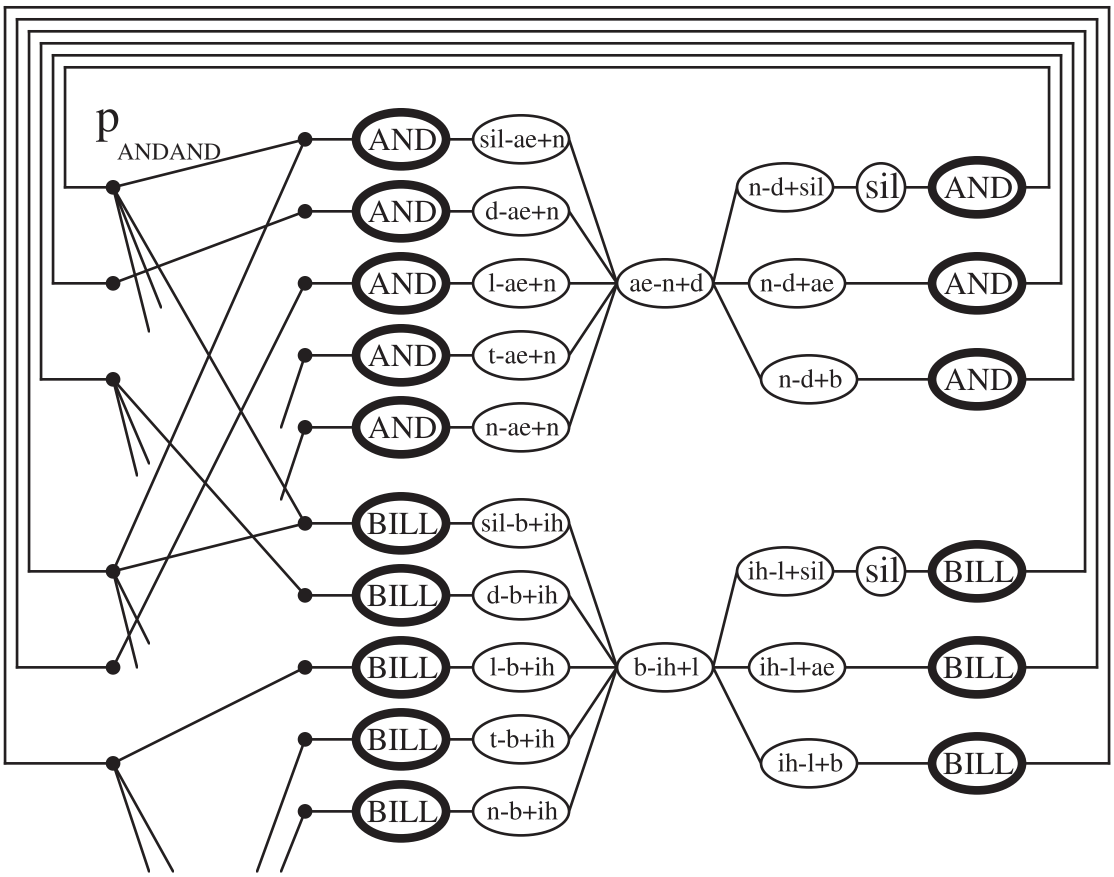

Figure 4.3: Part of a network using cross word triphone context dependent models.

Extending this architecture to use cross word triphones results in a huge increase in the total number of model instances in the network [72]. The initial and final phone of each word must be expanded to use different triphone models dependent on phones in adjacent words. Figure 4.3 shows part of the network from figure 4.2 expanded with cross word triphone models. It shows how the number of model instances required to represent each word grows with the number of word initial and final phones appearing in the vocabulary. For even small vocabularies (of a few hundred words) this can increase the size of the network by an order of magnitude. Even when many contexts share models, the network can easily become much too large for practical decoding. This is especially true because the initial phone of each word is expanded and, as section 4.2.2 described, a high proportion of the total computation is already involved in the word initial models.

Extending the network to allow the use of a trigram or longer span language model requires that the network includes multiple copies of each word to ensure that every transition between

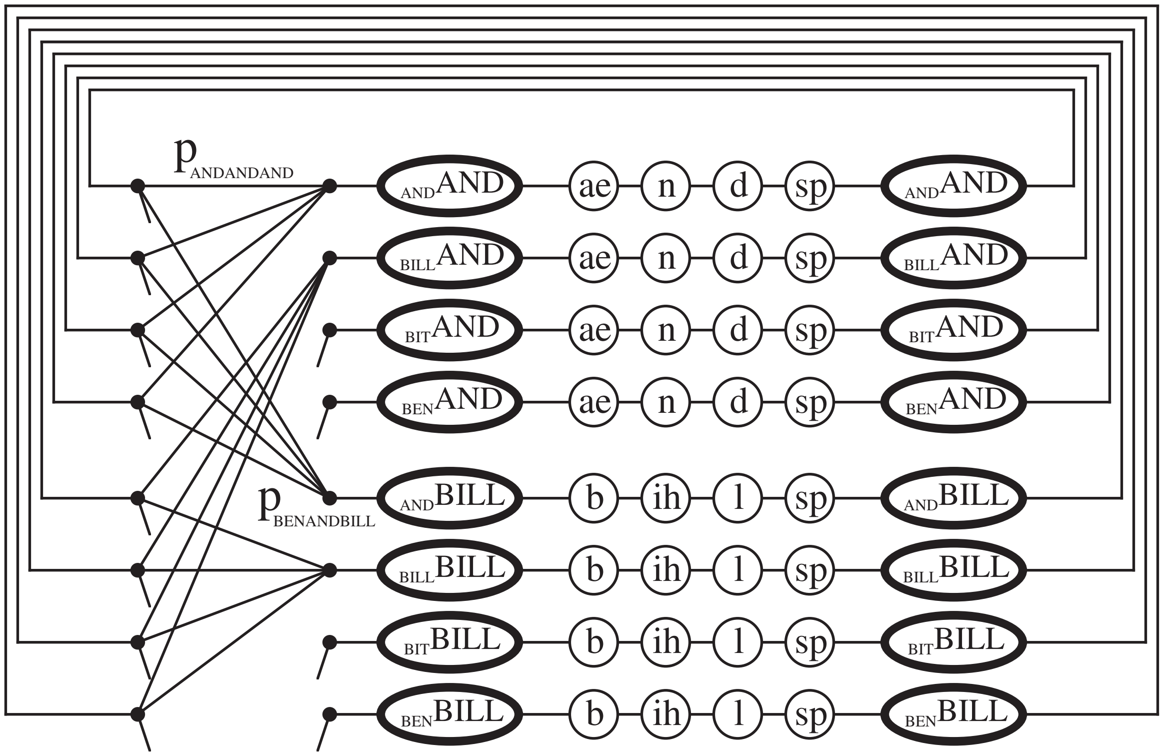

Figure 4.4: Part of a network using a trigram language model.

words has a unique two word history. Figure 4.4 shows a portion of the network from figure 4.2 expanded to allow the use of trigram language model likelihoods. When trigram probabilities exist for every word triple this means that a duplicate instance of every word is needed for each word in the vocabulary. This will increase the total number of model instances in the network by a factor that is the number of words in the vocabulary. Even for small vocabularies and relatively sparse language models this is impractical.

4.2.5 Back-Off Implementation

When back-off bigram language models are used, many of the language model probabilities are calculated using the unigram probabilities together with the back-off weights. Whenever these back-off values are used it is not necessary to know the identity of the previous word and the current word simultaneously to calculate the transition probability. The two components that make up the value can be applied independently and doing this allows the network structure to be simplified.

Each back-off transition is divided in two and a back-off node introduced between the new transitions. The likelihood of the transition into the back-off node is calculated using the back-off weight of the previous word and the likelihood of transition from the back-off node is calculated using the unigram probability of the following word [37].

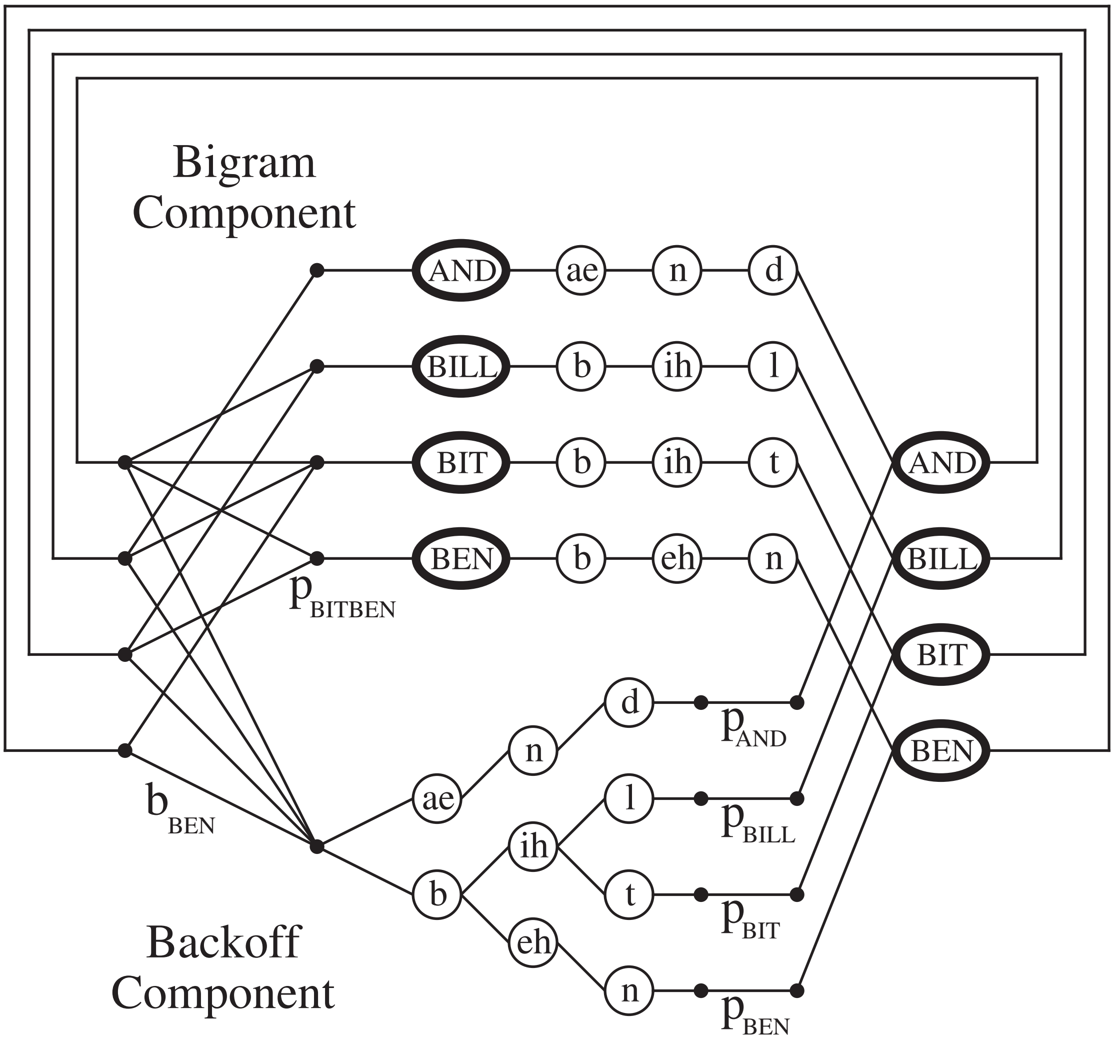

Figure 4.5: A bigram network with tree structured back-off component.

By itself this simplification does not simplify the network. However, merging the back-off nodes and sharing a single node with transitions into all word beginnings reduces the total number of transitions [3]. This introduces a back-off transition for all word pairs including those with an explicit bigram probability. In these cases, the most likely path will be taken when strictly only the bigram transition should exist. Normally the bigram transition is more likely but this is not always the case. However, in practice this approximation has little effect on overall accuracy [73].

The number of active between word transitions is reduced since only the transitions into and out of the single back-off node and explicit bigram transitions are needed for each active word end. Previously each active word end had a transition to the start of every word. This reduction saves a certain amount of computation because, for most words, bigram transitions exist for only a small proportion of the vocabulary. However more significant computational savings can be found by further restructuring of the network.

As mentioned in section 4.2.2, the majority of active models occur in the first few phones of each word [53]. Since explicit bigram transitions are only a small proportion of the between word connections the majority of word initial models will be active because of a back-off connection to an active word end. Separating the back-off section from the remainder of the network and effectively duplicating each word in the network doubles the size of the network. However, because the proportion of explicit bigram transitions is small this will only slightly increase the total number of active models.

The only transitions into the back-off section of the network are from the single back-off node and by delaying the application of the unigram until the end of the words the network can be simplified still further. The initial stages of the back-off network are identical for words which share initial model sequences. Consequently, multiple instances of these models are no longer required and the network can be tree structured so that words share initial model instances.

The network architecture is shown in Figure 4.5. The majority of active models in a beam pruned search are the word initial phones in the back-off section of the network and the use of tree structuring greatly reduces this number.

Delaying the application of the unigram language model until the end of words alters the likelihood of tokens in the back-off part of the network relative to the bigram part and this may lead to additional (or at least different) search errors. However, it is not necessary to delay the application as the likelihoods can be factored into transitions within the tree structured network. The likelihood of the most likely of the words sharing each model can be assigned to that instance and these likelihoods factored into the transitions into each model instance. This ensures that the likelihood of the tokens in any shared instance is the same as those in the most likely of the merged original instances. The likelihood of the top of the beam is then unaffected by the re-structuring and no extra pruning will occur (potentially causing additional search errors). In fact, fewer search errors may result since unlikely words could be saved from pruning if they initially share model instances with more likely words.

4.3 Best First Decoding

Time-synchronous searches are essentially simple breadth first searches. All hypotheses are advanced together every frame, independent of their relative likelihood (apart from the effects of pruning). This step by step approach is well suited to the frame by frame processing inherent in hidden Markov models and simple beam pruning schemes are effective in reducing the search space. However a great deal of computation is wasted on relatively unlikely paths which do not form part of the most likely path merely because their relative likelihood can change significantly later in the utterance.

Best first searches start by extending the most likely partial path and only later extend less likely hypotheses if extensions to the best path become unpromising. This asynchronous approach involves comparing hypotheses covering different portions of an utterance to find which is most likely. Time-synchronous approaches only need to compare hypotheses that cover the same number of frames. The likelihood of each frame will tend to be similar and so the likelihood of a partial path generally varies in a linear fashion as more frames are processed. In order to accurately compare hypotheses covering different numbers of frames an estimate is required for the likelihood of the portion of the utterance which is not covered by both hypotheses.

This *lookahead* gives best first decoders their main advantage. Because paths tend to merge at word boundaries, and identical continuations have the same likelihood, lookahead to the next word boundary will allow the total likelihood of each hypothesis to be compared. As explained in section 2.5, the beam used during a time-synchronous search must be wide enough to allow for variations in the relative likelihood of the continuations of different partial paths. Since the best first decoder uses an approximation to the whole utterance likelihood to compare paths and determine which to extend, the effective beam width can be much lower and far fewer paths need to be extended.

In the limit, the exact complete hypothesis likelihood is used to compare partial paths. This is the ideal case since only partial paths which form part of the globally optimal hypothesis will be extended and no computation is wasted on other paths. However, the computational burden has effectively just shifted to the lookahead calculations which need to be as complex as the original decoding problem if exact lookahead values are required [38].

When accurate lookahead values are available, perhaps from an initial time-synchronous pass, best first searches are the ideal choice since they will minimise the computational and storage requirements of the decoder [52].

4.3.1 A* Decoding

A best first search can be implemented with the A* algorithm [55].

This is an admissible search guaranteed to find the most likely hypothesis and uses an estimate of the likelihood of the whole utterance (the total likelihood) to decide on which partial paths should be extended. Once the partial path with the highest total likelihood spans the whole utterance it forms the most likely utterance hypothesis and decoding is complete.

The estimate of the total likelihood, $h^*(p_q(t))$, of the partial path, $p_q(t)$ is composed of two portions.

- $f(p_q(t))$.

  The exact log likelihood of the partial path $p_q(t)$ from the beginning of the utterance to time t calculated using the appropriate sequence of models in the same way as $\phi(t)$ is found in a time-synchronous decoder.

- $g^*(p_q(t))$.

  This is an estimate of the log likelihood of the most likely completion to path $p_q(t)$ from time t to the end of the utterance.

By ensuring that $g^*(p_q(t))$ is an upper bound on the exact likelihood of the remainder of the utterance, $g(p_q(t))$, the decoding process is admissible. The exact likelihood of the whole path $h(p_q(t)) = f(p_q(t)) + g(p_q(t))$ will always be less than the estimate since $g^*(p_q(t)) > g(p_q(t))$

until the path spans the whole utterance and $g^*(p_q(T)) = 0$ and $h^*(p_q(T)) = h(p_q(T))$. The search is admissible and finds the hypothesis with the highest likelihood because no partial path that could be extended to have a higher likelihood over the whole utterance will exist. If such a partial path did exist it would already have been extended because its estimated likelihood $h^*(p_q(t))$ would be greater than its exact value and thus higher than the completed hypothesis.

The search is initialised with a single null partial path with $f(p_q(0)) = 0$ and $g^*(p_q(0))$ an estimated likelihood for the whole sentence. This null path will then be expanded with all possible one word extensions. The exact likelihood of each of these words will be calculated and combined with the approximate likelihood of the rest of the sentence and then used to select which partial path will be expanded next.

The search can proceed without using any lookahead information by using a constant upper bound for $g^*(p_q(t))$. However this will force the search to behave in a time-synchronous fashion extending the shortest partial path at each opportunity.

When only the most likely hypothesis is required computation can be saved by merging equivalent paths. When a set of partial paths will be extended in the same way and the equivalent extensions will be equally likely, the relative likelihood of the paths will remain constant throughout the remainder of the utterance. Only the most likely of the set can form part of the globally optimal path and only this path needs to be extended [70].

Only a single entry is required for a set of equivalent paths. When a partial path is produced that is equivalent to but less likely than one already present it may be discarded since it does not form part of the optimal hypothesis. When the new partial path is more likely, it replaces the one already present.

An A* search is capable of generating an ordered list of all likely hypotheses as well as the most likely one. This is accomplished by not stopping with the first partial path that spans the whole utterance but continuing to extend paths until the required number of completed paths have been found. However, the above condition can no longer be used to allow path merging since the less likely of a set of equivalent paths can form part of one of the N most likely hypotheses although not the most likely one.

The major computational load in such a search (in addition to the calculation of path and lookahead likelihoods) is the manipulation of the set of partial paths. The operations that need to be performed on this set are;

- Expansion. Find the most likely partial path and evaluate its extensions.
- Insertion. Insert a new partial path created during path extension.
- Relaxation. Replacing a partial path with an equivalent but more likely one.

These operations can be very efficiently implemented if the set of partial paths is stored as a *Fibonacci heap* rather than a stack or ordered list ([15]). When stored in this way the computation required by all of the above operations rises less than linearly with the total number of partial paths considered (in fact expansion and relaxation scale with logarithm of the number and insertion is independent). For a stack or list, the insertion and relaxation operations require computation that scales linearly with the total number of partial paths. For very large searches this can become a significant computational overhead.

A* searches are particularly well suited to searches through networks and lattices. They are often used as second (or subsequent) passes during decoding once an initial time-synchronous pass has calculated values for $g^*(p_q(t))$ and reduced the search space to a lattice of word hypotheses.

4.3.2 The Stack Decoder for Speech Recognition

Best first searches can be implemented with a stack decoder. Partial paths are stored on a stack which is sorted in likelihood order. The decoder operates by removing the most likely partial hypothesis from the stack, extending it by one word and inserting the resultant hypotheses back into the stack [67]. Normally fast but approximate matches are used to reduce the number of extensions which need to be considered by the detailed match.

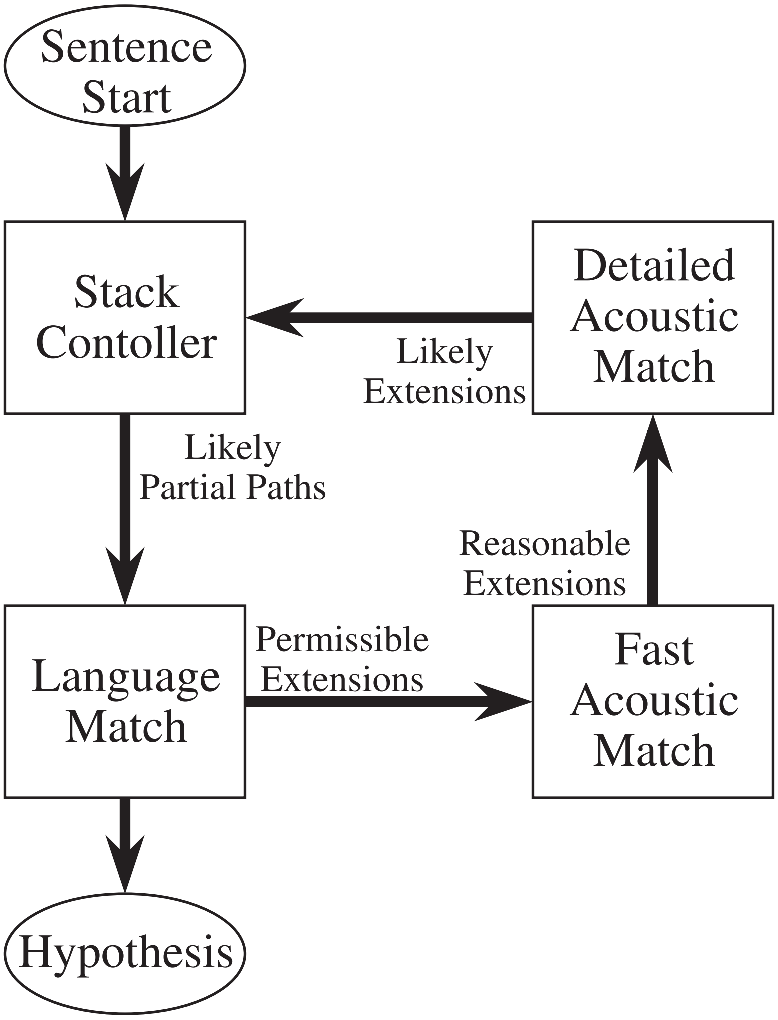

Figure 4.6: Dataflow in a Stack Decoder.

Figure 4.6 shows how this process operates. The stack is initialised with an empty beginning of sentence hypothesis. The most likely partial path is removed from the top of the stack and the language model queried to find the likelihood of all the permissible one word extensions. An approximate (fast) acoustic match is performed to determine which of the extensions are likely and these are rescored with detailed acoustic models. Likely extended paths are then placed into the stack for later extension.

This structure allows a great deal of control over the complexity of the search.

- Easy to incorporate lookahead information.

  The order that items are considered can be influenced by lookahead for the remainder of the sentence. The order of partial hypotheses in the stack can be based upon an estimate of the likelihood of the whole utterance. Rather than always extending the most likely path which may only have unlikely extensions the estimated whole utterance likelihood can be used to extend a path that is currently less likely but has more likely extensions.

- Easy control over search complexity.

  The partial paths are considered in a best first manner and so the number of paths that need to be extended (and the number of extensions considered) can be controlled. This can be used to place an upper limit on the computational complexity (and thus worst case performance) which is difficult with likelihood based beam pruning.

However this architecture does have a number of drawbacks. Partial paths are extended a word at a time. This is well matched to the language model but is a poor fit with the acoustic models which calculate likelihoods frame by frame. For continuous speech the position of word boundaries is not known and so each partial path takes the form of a word level history and a set of likelihoods for different end times. This becomes even more complicated when cross word context dependent acoustic models are used since the acoustic likelihood of a word is dependent upon following words. In practice this means that paths must be rescored and new likelihoods calculated once the appropriate contexts are known. This can effect the admissibility of the search as well as lead to duplicated computation for evaluating a single path.

The biggest advantage of this scheme is the ability to use the approximate match to limit the number of extensions considered by the detailed models. However fast matches tend to be more approximate and lead to search errors when only a few extensions are considered whilst more accurate matches which need consideration of fewer extensions require more computation themselves. Efficient but accurate fast matches are essential to avoid repeated (and therefore wasted) computation [8].

Finally the stack itself can become a drawback. When the number of likely hypotheses is high the stack can grow very large. With a fixed size static network the number of hypotheses that exist at one time is limited to one per state in the network as hypotheses are advanced one frame at a time. The stack can contain hypotheses from many times and its size is not well bounded. For N-Best decoding simple path merging is not possible and a single stack becomes unwieldy and inefficient. A secondary stack is used to store alternative hypotheses which only need be accessed once the most likely answer has been found [86].

However the stack decoder provides an easy way to incorporate detailed acoustic and long span language models in a efficient rescoring pass when hypotheses are extended. So the first attempt to build a decoder capable of using cross word triphones and long span language models was based on a stack decoder which operated in a frame synchronous fashion linked to the acoustic models.

4.4 A Hybrid Approach

An initial attempt to build an efficient decoder capable of handling large vocabularies, cross word context dependent models and long span language models was based on a stack decoder.

This architecture has been used successfully with discrete density cross word context dependent hidden Markov models. Complex fast match schemes are used to reduce the number of word extensions considered when each partial path was extended. None of these methods was suitable for use with continuous density models and so a beam pruned search of a tree structured network of monophone models was used to perform an initial approximate but computationally simple match. The words which are relatively likely with these models are then rescored with the cross word triphone models.

The decoder functions in a semi time-synchronous fashion using frame by frame token passing to evaluate both the fast and subsequent detailed matches. The stack stores the likelihood of each partial path at word boundaries. When paths are extended they are rescored with the detailed models and likelihoods from the stack are fed into a newly created context dependent network of triphone models. This triphone network is advanced asynchronously until it catches up with the main network. Once it has reached the same point in the utterance, tokens within it can advance time-synchronously with the other models and it rejoins the main network.

Initially context dependent cross word triphones were only used for the initial stages of the network when the context is fully specified. The final phone before the tree structured fast match network and the whole of the tree structured network used context independent monophone models.

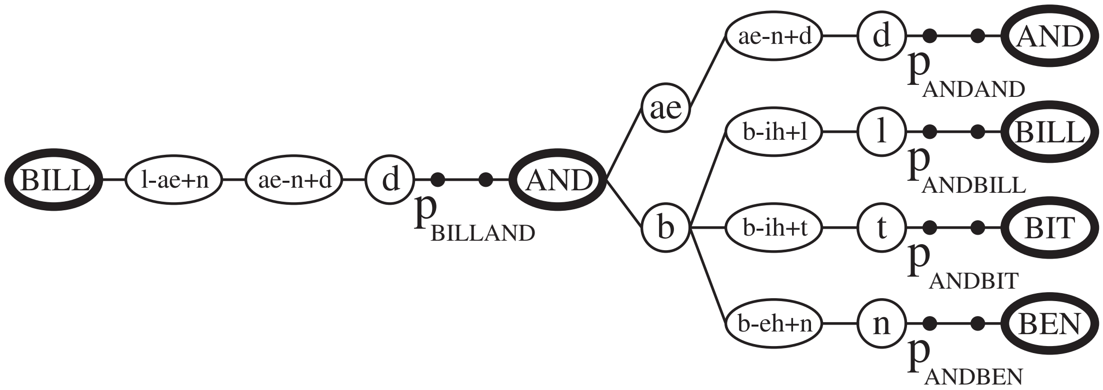

Figure 4.7: Structure of the network in the hybrid decoder.

However the monophone models used for the initial approximate match are a relatively poor approximation for the detailed triphone models used later on. This meant that the beam used for pruning the fast match needed to be relatively wide to ensure that few search errors were made and decoding proceeded slowly because many paths needed to be re-evaluated with the triphone models.

Within words, between the first and last phones, sufficient context is always available to choose a triphone model. The fast match can become more accurate by making use of triphone models wherever possible within the fast match network. This does not greatly increase the total computational load because the increased accuracy and specificity of the triphone models offsets the increased computational complexity due to the increased network size.

As before, when the network is extended at word boundaries the path is re-evaluated and context dependent triphone models used for all but the last phone before the tree structured fast match network. The tree structured fast match network uses a mixture of context dependent and context independent models. The initial and final phones of each word are represented with monophone models but within words context dependent models are used. Figure 4.7 shows this structure.

The use of monophone models for the first phone ensures that the initial stages of the tree structured network are still very compact. Since many of these models will be active this ensures that the computation required by the fast match remains relatively small. Using context dependent models for the rest of each word increases the accuracy of the approximate match allowing relatively narrow beam widths to be used without introducing search errors.

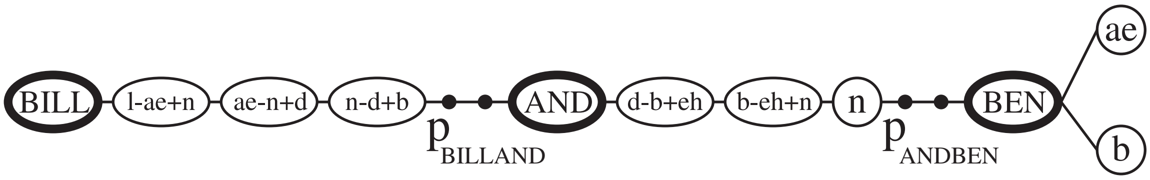

Figure 4.8: Continuation of the hybrid network.

Figure 4.8 shows how the network from figure 4.7 would be extended from the word BEN. The network up to the final phone of the word BEN has been rebuilt with context dependent triphone models. The tokens fed into the word AND from the word end BILL are taken from the stack. These are only stored for the period during which the word end of BILL was both active and relatively likely, to minimise storage requirements.

Unfortunately tuning this system to minimise the total number of errors (including both search and modelling errors) was time consuming and difficult. To maximise speed without introducing search errors required the use of different beam widths to determining which paths to extend, which tokens to propagate during the initial match and which tokens to propagate during the rescoring with the context dependent models. Choosing these beam widths required multiple experiments which needed to be repeated when substantial changes were made to either the detailed or fast match systems. However this architecture has been shown to work well when there is no need to perform a separate fast match [28].

These drawbacks resulted in a desire for a simple one-pass decoder which only used the detailed set of models removing the need for careful system integration during decoding with the speed/accuracy trade-off controlled by a few easily chosen parameters.

4.5 Summary

This chapter has outlined the basic methods for word based continuous speech recognition and has described the token passing method for implementing the Viterbi algorithm. It has then explained the problems encountered when expanding this scheme to use cross word context dependent acoustic models, large vocabularies and long span language models. A method for implementing a particular type of recognition system using tree structuring was explained to show the way in which sharing computation can be used to speed up decoder operation. Best first approaches to decoding were outlined including an explanation of A* searches and the stack decoder. Finally a hybrid stack/Viterbi decoder was described and the problems associated with this hybrid approach were explained.

The preliminary work in designing a decoder indicated that decoder simplicity is a very desirable characteristic especially for system development work when accuracy is of prime importance. This desire for simplicity led to the goal of producing a decoder capable of functioning in a single pass, utilising the best acoustic and language models throughout the search to enable the use of tight pruning to speed up the search without introducing excessive numbers of search errors.

The next chapter describes a decoder capable of utilising cross word context dependent acoustic and long span language models in conjunction with very large vocabularies in a single pass producing a lattice of likely word hypotheses as well as the single most likely hypothesis.

Chapter 5

A One-Pass Dynamic Tree Structured Network Decoder

This chapter begins by outlining the philosophy behind the development of a one-pass decoder.

It explains a one-pass method for time-synchronous decoding of continuous speech suitable for large vocabularies using the cross word context dependent acoustic and long span language models required for greatest accuracy. This decoder uses a dynamically constructed tree structured network to allow efficient operation using reasonable amounts of storage.

The decoder can be extended to generate a lattice of likely word hypotheses without significantly increasing the computation required. Methods for generating, pruning and assessing the accuracy of such lattices are described. Finally the way in which these lattices can be used to reduce the computational requirements during system development is discussed.

5.1 Philosophy

It is desirable for any recogniser to perform efficiently and to decode each utterance as quickly as possible. However fast decoding can result in search errors if the decoder fails to find the most likely hypothesis. This is undesirable as it makes it difficult to accurately estimate the performance of the underlying speech recognition system.

Normally a balance must be struck between speed and search accuracy. In practical systems real-time performance is often a design requirement. However, evaluation of different acoustic and language modelling techniques requires accurate comparison of systems, and so the number of search errors should be relatively small.

Maximising the speed of a recogniser requires minimising the total number of calculations during decoding. The number of calculations can be reduced in two ways.

- Simplifying the evaluation of each partial path.
- Reducing the number of partial paths that are evaluated.

The use of less complex models simplifies the evaluation of each partial path but tends to increase the number of modelling errors. Similarly, just reducing the number of partial paths considered (for instance by reducing beam widths) will tend to increase the number of search errors.

To maintain high accuracy as well as efficient decoding a combination of these techniques is needed. The last chapter described how fast but approximate matches using simple models can be used to reduce the number of partial paths that are evaluated with computationally expensive detailed models. Similar approaches are used in stack decoders and other multi-pass schemes [11]. However fast matches do have some disadvantages (especially for system development).

- Fast matches are approximate.

  A mismatch exists between the simple models used in the fast match and the detailed models used in the final system. The number of hypotheses that the detailed match must consider in order to avoid search errors increases as the fast match becomes simpler. This offsets the reduction in computation due to the simpler fast match models.

- Multiple systems are needed.

  It is necessary to generate and optimise not only the system that is being tested but also a fast match system to perform the lookahead. Generating the fast match system can be time consuming and computationally expensive.

However, the major problem is the need to closely integrate the two systems and ensure that relatively few search errors are made whilst maintaining decoding efficiency. This can be a complex process requiring the optimisation of many parameters. When implementing real systems this is not a problem because it only needs to be performed once. However the need to perform detailed comparisons between systems during system development on a one off basis makes a simple one-pass decoder, needing only one model set with few parameters to optimise, seem very attractive.

As discussed in the previous chapter, simple time-synchronous one-pass decoding techniques are not suited to the use of cross word context dependent models, long span language models and very large vocabularies. There are two main problems;

- Size.

  The size of the standard linear network becomes prohibitively large when tasks and the models used become more complex.

- Computation.

  A high proportion of the model instances in the network are active every frame and the computation needed for decoding using these methods becomes prohibitive.

However, such one-pass decoders simplify both training and testing. Using the most accurate models from the outset allows the size of the search space to be minimised without introducing search errors. The simplicity of one-pass time-synchronous approaches makes them attractive if they can be performed in a computationally tractable way.

The decoding strategy described in this chapter overcomes the above problems in the following ways.

- Dynamic network construction.

  Since the majority of the network tends to be inactive during a beam pruned search allocating space for the active portion of the network dynamically will minimise the memory requirements.

- Early application of knowledge.

  Knowledge sources are applied as early as possible to allow the size of the search space to be reduced without introducing search errors.

- Sharing of computation.

  The asymmetry in a beam pruned search together with the similarity of many words in a large vocabulary can be exploited to reduce the computation needed. Tree structured recognition networks have been shown to be particularly effective in reducing the total number of active models.

5.2 Network Architecture

The proportion of words actively considered during a beam pruned search decreases through the word, with a high proportion of word initial model instances active and only a small proportion of word final ones. Left to right tree structuring of the network, allowing words with common initial phone sequences to share model instances, greatly reduces the total number of word initial models. Because of the asymmetry in activity, word initial model instances represent a high proportion of the total computation and thus tree structuring substantially decreases the total number of active models [53].

However this simple tree structuring is only possible when a language model with unigram dependencies is used (such as the backoff portion of a backoff bigram language model, as described in section 4.2.5). In order to calculate bigram language model probabilities it is necessary to know the identity of both the current and previous words. This is possible in a linear network with a single instance of each word (as shown in figure 4.2) because a distinct transition exists between each pair of words.

With a tree structured network, many words initially share model instances and so distinct transitions do not exist for each word pair because the identity of the following word is not well defined. In fact, because of the presence of words which are homophones, the identity of every word cannot be uniquely determined until its end. This makes it impossible to apply a language model on the transition into a word, it must wait until the end of the word, when the identity of the word can be uniquely determined.

A bigram language model requires the identity of both words in the bigram pair. If the language model application must wait until the end of the second word, it is necessary to have a distinct network for each possible initial word to ensure the bigram transition is specified uniquely. This requires a copy of the tree structured network for every word in the vocabulary and the size of the network is multiplied by the size of the vocabulary. When longer span language models are used more tree copies are needed since a unique copy is required for every word history that the language model considers distinct. For a trigram the language model probabilities are conditioned on the previous two words (rather than one in the case of a bigram) and so tree copies are required for each distinct two word history.

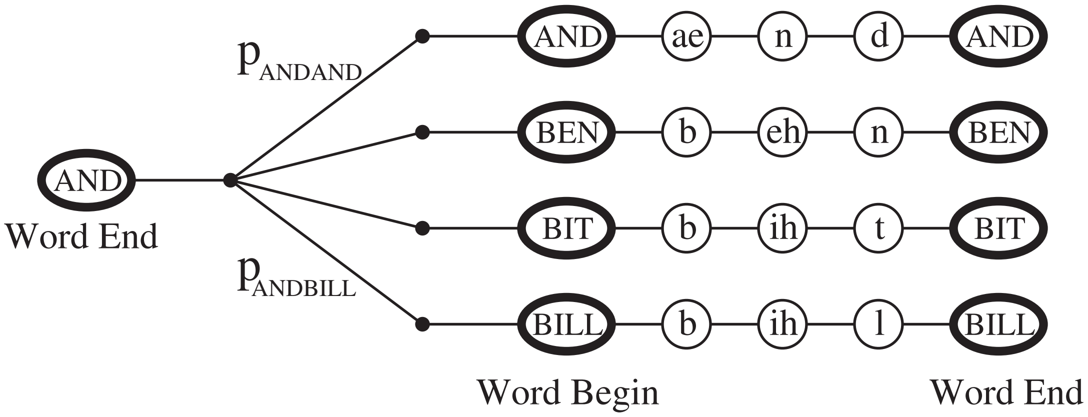

Figure 5.1: A linear network.

Although this restructuring increases the network size dramatically, the computational load is dependent upon the number of models that are actively considered rather than the total size of the search space. Because of the asymmetric pattern of model activity in a beam pruned search, tree structuring reduces the average number of active models despite the increase in total network size. This is due to the relatively small number of active word end models, and hence the number of tree copies being actively considered, compared to the number of word initial models active in a linear network. Table 4.1 showed that only a few tens of word ends are active (and this can be further reduced by word end pruning as described in section 5.3.2). Tree structuring the network can reduce the number of word initial model instances by more than an order of magnitude and hence reduce the total number of active models. It also means that the total computational load is relatively independent of vocabulary size. For large vocabularies, the size of the initial portion of the tree structured network is almost independent of the total number of words. The number of active word ends (and hence the number of tree copies) tends to be a function of the confusability inherent in the task which is only a weak function of vocabulary size. This also means that the increase in network size needed by longer span language models may not increase the number of active models as the rising size of the network will often be offset by the gain in search locality resulting from the improved accuracy of the language model.

However the total size of the search space has increased dramatically and so efficient pruning and memory allocation are relatively much more important.

This restructuring can be viewed as taking a linear network following a particular word end (as shown in figure 5.1) and delaying the application of the language model until the end of each word. This allows common initial model sequences to be shared between words (as shown in figure 5.2). This delay is undesirable because it reduces the locality of the search and leads to increased numbers of search errors at a particular beam width, and furthermore it is unnecessary [2].

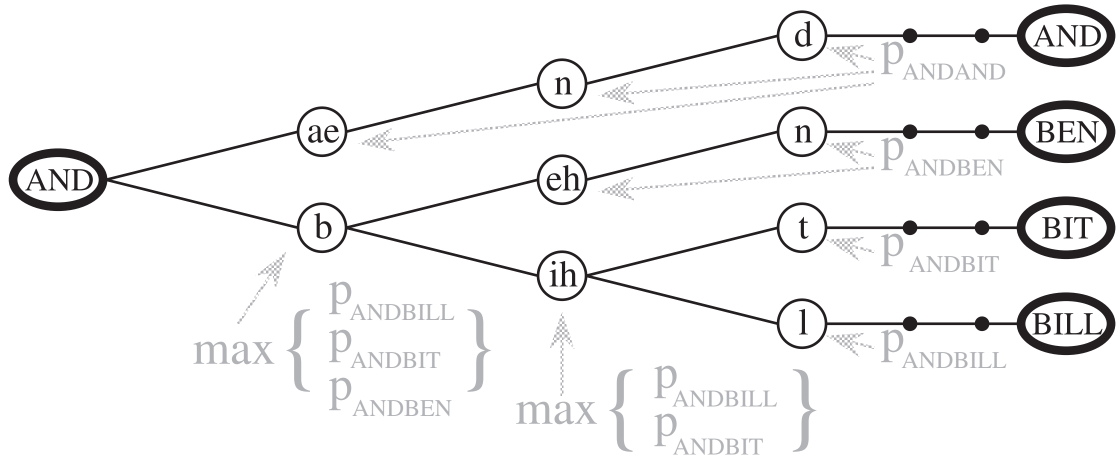

Figure 5.2: A tree structured network.

Section 4.2.5 described how the probabilities may be factored into the transitions within a tree structured network. The likelihood of the most likely word sharing each model instance is associated with the instance and is used to calculate the transition probabilities in the network.

This is implemented by storing a language model likelihood with each instance and combining this with the token likelihoods for pruning purposes. This removes the need to calculate and store a likelihood with each transition and allows a uniform network structure in which the language model likelihoods are eventually added to the tokens at the end of words. Figure 5.2 shows the network structure and the language model likelihoods associated with each model instance (in grey).

This early, but approximate, application of the language model allows relatively narrow beam widths to be used without increasing the number of search errors. The progressive addition of language model likelihoods for relatively unlikely words can reduce the number of search errors because the large step changes in token likelihood (which occur if the likelihoods are added during a single transition) are avoided.

5.2.1 Context dependency

Word internal context dependent models can be incorporated into a conventional linear network without increasing its size or the computation required during decoding. When such models are used with tree structured networks, the number of different word initial model sequences increases and the amount of sharing possible decreases. This increases the size of the network and the computation required for recognition.

However, the tree structured network is better suited to the use of cross word context dependent models than the linear network because of a side effect of the need to make tree copies. Incorporating cross word triphones into a linear network results in an order of magnitude increase in the size of the network. The word initial portion of the network (which dominates the computational load) is expanded to be dependent on the last phone of the previous word and the computation required during decoding rises by a similar amount.

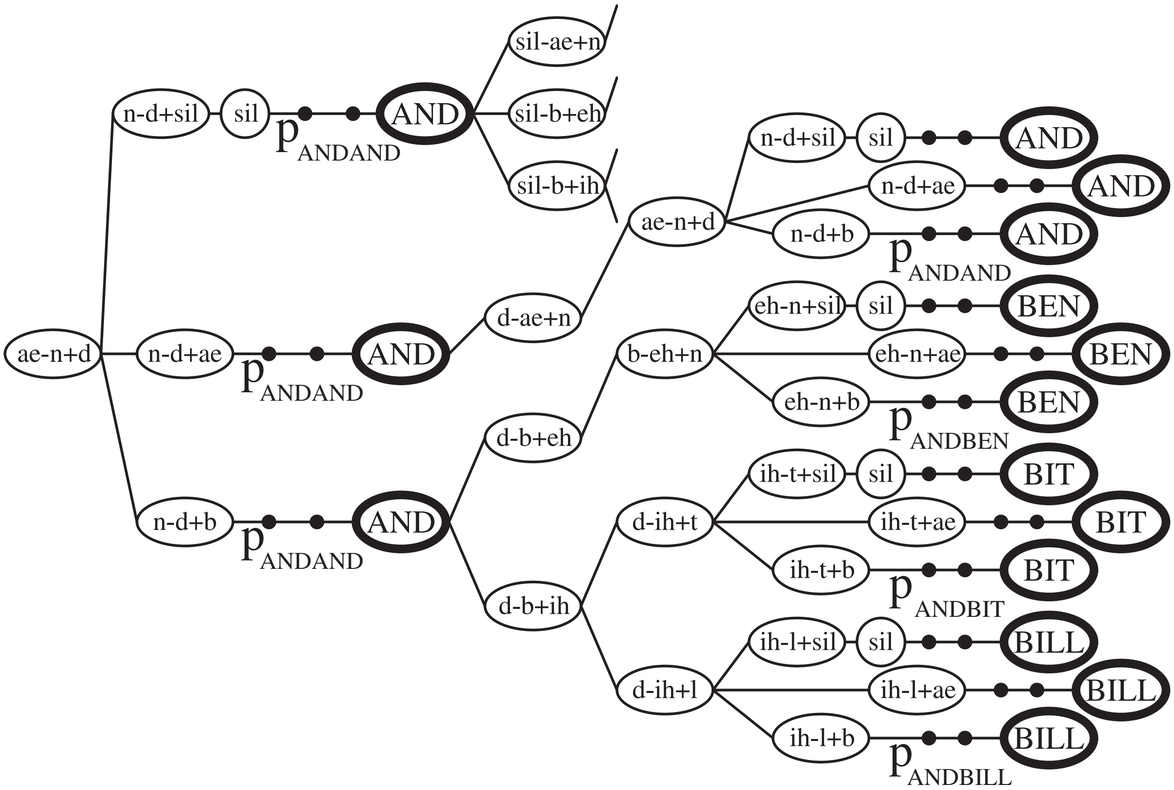

Figure 5.3: Cross word triphones in a tree structured network.

Using cross word context dependent models in the tree structured network increases the computation required during decoding by only a small amount compared to a network using models dependent upon word internal context. A distinct tree copy is generated for each preceding word and when each word has only a single phonetic pronunciation, the phonetic context of each tree will be unique. Adding cross word context dependency will only expand the final part of each word since the initial part has effectively already been expanded when the tree copies are made. Since only a small proportion of word end models tend to be active in a beam pruned search, the total computation required does not increase substantially. Of course, when multiple pronunciations exist for a word they may require additional tree copies to preserve the unique previous phonetic context of each tree. However, only a few words will normally have more than one pronunciation.

The increase in model accuracy will also increase search locality, so although the potential size of the network increases with the use of cross word context dependent models, the computation needed for decoding may actually decrease compared to using word internal models.

Often the recogniser needs to allow an optional silence between words and this is accomplished by adding an optional silence model at the end of the pronunciation for each word. This effectively doubles the number of pronunciations in a given dictionary because the phonetic context will depend upon whether the silence is present. In this case, the computation will increase significantly since both 'pronunciations' will be almost equally likely and the number of tree copies could potentially double.

Figure 5.3 shows the network from figure 5.2 expanded to incorporate cross word context dependent models. It shows how the initial part of each tree is expanded due to the following (word internal) context but that the cross word context dependencies only expand the final phone of each word. This substantially increases the total size of the network but since only a few of the word final models will be active, it does not significantly increase the computation beyond that required for a system that only uses word internal context dependent models. Note also that the optional silence, `sil`, model at the end of each word results in two tree copies for the different phonetic contexts.

5.2.2 Network Structure

Apart from the left to right tree structuring, which ensures that each node in the network has only one predecessor although it may have many successors, the network is otherwise very similar in structure to the standard linear network described in section 4.2.1. Consequently, the same token passing paradigm is used for both networks to calculate the likelihood of partial paths.

Two types of node appear in the tree structured network.

- Model Instances.

  These represent a particular phone in context and are associated with a specific model. Each instance holds one token for each state of the associated model, including the non-emitting entry and exit states. This token represents the most likely path ending at the state at the current time. The likelihood of each token is updated after every observation in the same way as in the conventional static network. Traceback information is not updated within the word unless state or phone alignments are needed. The instance also stores a language model likelihood. This is the likelihood of the most likely of the words sharing the instance and is added to that of the most likely token in the instance to provide an instance likelihood used for pruning.

  Unless a transition exists between the non-emitting entry and exit states tokens will be updated with at least one observation likelihood whilst passing through model instances.

- Word End Instances.

  Word end nodes are associated with a particular word (or more accurately with a particular pronunciation of a particular word). They represent the point in the network where the word identity becomes unique. At this point, the language model likelihood of the word is added to the token likelihood and the token's traceback information is updated to allow later recovery of the word sequence. The likelihood of the token after addition of the language model likelihood is used to determine if the node should be pruned.

Tokens pass through a word end instance without processing any observations.

It is not feasible to pre-compile the tree structured network due to its potentially huge size. The network must be built dynamically, similar to the way in which partial paths are extended on demand in a stack decoder [58]. This also makes it easy to use any type of finite state word network as a grammar (including one generated in a lookahead pass) without incurring any additional computational or storage overheads.

5.2.3 Token Passing and Network Growth

The most likely path through the tree structured network is calculated using the same token passing paradigm described in section 4.2.1 for a linear network. However, since the network must be constructed dynamically during decoding to minimise storage requirements, token propagation also controls the growth of the network. When active tokens (which fall within the beam) propagate from a node in the network which has not yet been expanded it may be necessary to add nodes to the following part of the network.

The list of different pronunciations sharing the node is processed to determine the set of model instances and word end nodes which should follow. Word end instances are created and linked into the network if the end of the pronunciation has been reached. Within words, the context is used to determine which model should be used to represent the next phone and an instance of this model is created and linked into the network.

At the ends of words, the permissible extensions to the current partial path are needed. When cross word context dependent models are used, these extensions may also be necessary to determine the following context for the last few phones in each word. In this case, each pronunciation will be split dependent upon following context and several models used to represent the final phone or phones of a single pronunciation. The list of permissible extensions together with their likelihoods are obtained from the combined language model and grammar. When large vocabularies are used with fully connected grammars, it is not feasible to perform the network expansion individually for each word and instead pre-compiled prototypes are used to guide the process. With smaller vocabularies or heavily constrained finite state syntaxes, this pre-compilation is not needed and the actual list of words (and associated pronunciations) is used directly.

If the exact procedure above was followed, the network would always be extended in complete layers (which consists of all possible followers for a given node). However there will be significant variation in the likelihood within the set of following nodes due to their differing language model likelihoods. Sometimes some of the newly created nodes will not fall within the current beam and would be destroyed during pruning only to be recreated the following frame. To avoid this, the combined likelihood of the node to be created (as used for pruning) is checked to ensure it falls within the current beam. If it does not, the node is not created immediately and the network layer is not fully expanded. The network growth is complicated a little but since the final layer of the network represents a significant proportion of the active instances substantial computational and memory savings result.

The presence in the network of nodes which can be traversed without processing any observations, such as word ends and tee models, means that every frame the active models must be processed in the correct order to ensure correct operation of the recogniser. Before a token can propagate from the exit state of an instance into the entry state of its followers it must be updated for the current frame. Essentially this means that the whole of the network should be processed from left to right and each instance processed before any of its followers.

Fortunately this is the sequence in which the network was created. Keeping the list of active instances in the order in which they were created and always processing the models in this order ensures that token propagation occurs correctly for all models.

The network is initialised by creating a single beginning-of-sentence word end instance holding a single token with a log likelihood of zero and a null path. This token immediately propagates from the word end instance and begins to generate the recognition network starting with the distinct beginning of utterance word, *!SENT_START*, which will normally consist of a single silence model. The network continues to grow in the manner described above once a token has passed through the *!SENT_START* instance and the likely portions of the recognition network will be generated.

Once the final frame of the utterance has been processed, the network is searched to find all occurrences of the unique utterance final word *!SENT_END*. The most likely token in any of the corresponding word end instances is found and the recognition hypothesis found by traceback from this token.

When multiple systems need to be run in parallel, for instance when separate gender dependent models are being used, several networks are needed. Each network is associated with a particular system and is composed only of models from that system. Recognition begins by initialising several networks although at the end of the utterance only the single most likely hypothesis is found (together with the identity of the system that generated it). Frequently the network for less likely systems will be completely destroyed by pruning and the total computation required for decoding with several systems in parallel will be only slightly higher than for a single system.

5.2.4 Path Merging

The number of reasonable hypotheses for an utterance increases exponentially with its duration. To prevent the computation involved in decoding the utterance increasing exponentially as well, it is necessary to share the computation of similar portions of different hypotheses.

In the standard network this is accomplished by tokens recombining (so that only one survives) at the beginning of words. When many partial paths to the beginning of a certain word exist, it is only necessary to extend the most likely. The most likely path through the remainder of the sentence from that point in the network is independent of the previous path and so the relative likelihoods of the different partial paths will remain fixed. In best first decoders path merging occurs explicitly by checking for equivalent partial paths and only extending the most likely one.

Path merging must also be performed in the dynamic net decoder to ensure that the computation does not rise throughout an utterance.

Three requirements need to be met for a set of partial paths to be considered equivalent;

- Identical network structure.

The allowable extensions to each partial path must be identical.

- Identical acoustic likelihood.

  Identical extensions to each partial path should be composed of identical acoustic models to ensure that the relative likelihood of the extended paths remains unchanged.

- Identical language model likelihood.

  The partial paths should appear identical to the language model so that identical extensions will have equal language model likelihoods.

With the static network these conditions are encoded within the topology of the network by defining the points at which tokens recombine. When the network is constructed dynamically it is necessary to determine equivalent paths dynamically in a similar manner to that used in a best first decoder.

The language model likelihoods are only known at word ends and so the final requirement can only be met easily at transitions from word end instances. Consequently path merging occurs at word ends and when word end instances are created they are linked with equivalents to form a partial path equivalence chain.

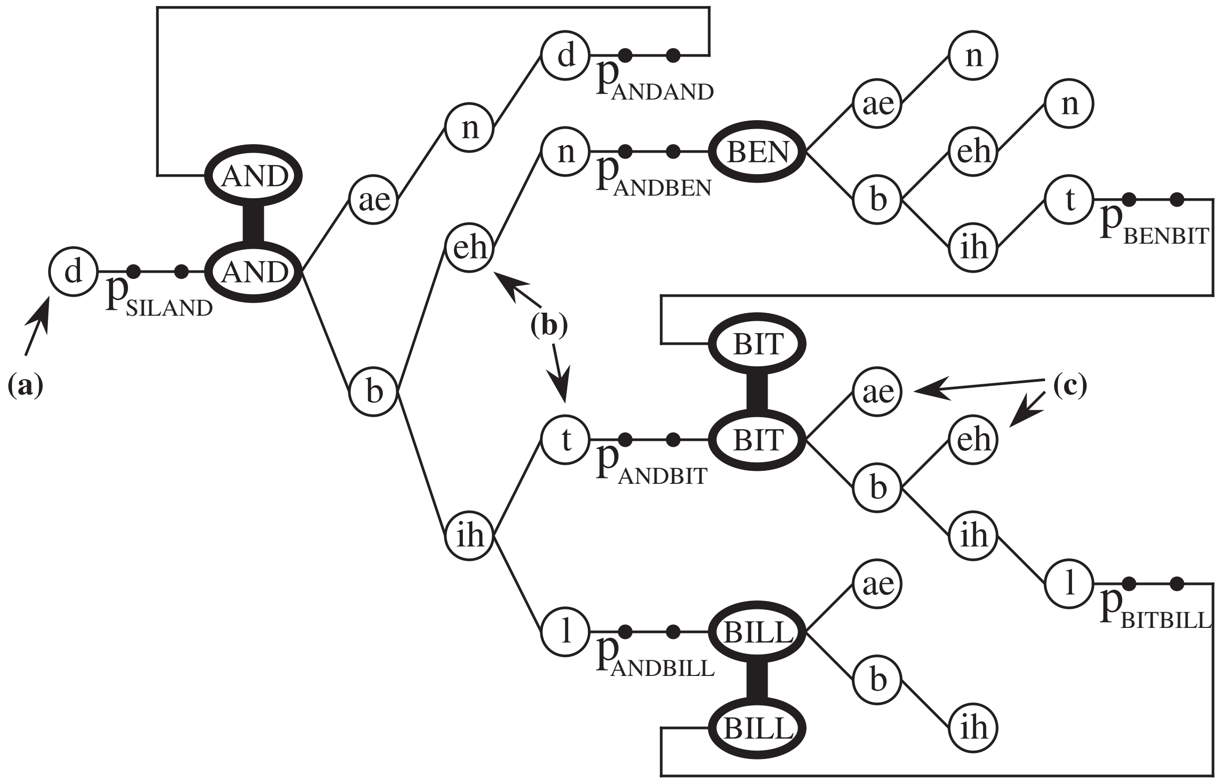

Figure 5.4: An example of the structure of a tree structured network.

Figure 5.4 shows an example network. The network is based on monophone models and does not include any optional inter-word silence models. The heavy links between word end instances are the equivalence chains and represent the points at which partial paths merge.

There are two ways in which the paths can be merged to prevent the unnecessary extension of equivalent partial paths.

- Path domination.

  Extension of the network is explicitly blocked for the less likely partial paths. This is the way in which equivalent partial paths merge in stack decoders.

- Token recombination.

  Identical partial paths share a single successor network. Tokens recombine before propagating into the successor network and only the most likely one survives. This is the way in which paths merge in a conventional static linear network decoder.

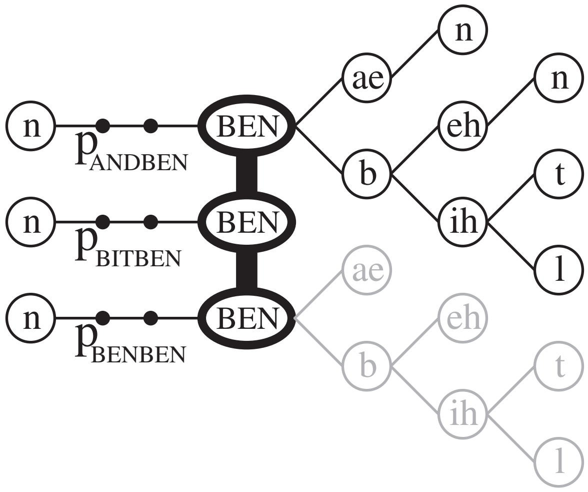

Figure 5.5: Path merging using path domination.

The first technique, domination, can produce several successor networks to one set of equivalent partial paths if their relative likelihood changes. Figure 5.5 shows this. Initial *AND BEN* is more likely than either *BIT BEN* or *BEN BEN* and this path is extended whilst the others are blocked. Later, when *BEN BEN* becomes more likely its block is removed and the network shown in grey is created. *BIT BEN* remains blocked.

Token recombination explicitly creates a single network and results in smaller networks requiring less computation. Figure 5.6 shows how the partial paths from figure 5.5 are merged by sharing a single successor network.

Despite potentially producing larger and less efficient networks, path domination ensures that the left to right tree structure of the network is maintained. Each network node has only a single predecessor and this ensures that correct token propagation is simple because several tokens never propagate into a single model.

This has two benefits.

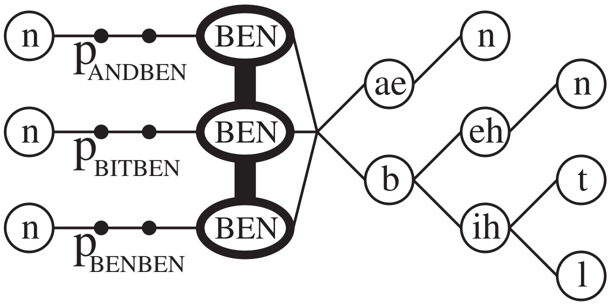

Figure 5.6: Path merging using token recombination.

- It is easy to maintain the network in the order required for correct token propagation. When paths recombine the order in which the network is processed may need to be changed because a word end instance is created and linked to followers that were created before it.
- The traceback information needed to recover the most likely sequence of words can be stored at the network (model) level rather than explicitly with every token. This can reduce the total memory required for the network by between 5% and 20%.

These benefits must be offset against the wasted computation when multiple networks are generated from a single set of equivalent partial paths. In practice this occurs rarely and typically results in the number of active instances in the network increasing by around 5% to 20%. This increase in network size may not lead to a similar increase in total computation because the smaller size of each instance can lead to better cache performance and increased CPU efficiency.

However, the memory overhead of the token recombination method is relatively small (the network rarely represents more than half the total process size) and its use makes it easier to implement efficient N-best lattice recognition (see section 5.5.1). Consequently most of the experimental work has used the token recombination method of merging equivalent paths.

5.3 Pruning

Pruning accomplishes two tasks during decoding.

- Avoids (wasted) computation for paths that are relatively unlikely.
- Reclaims (unused) storage to minimise total memory requirements.

When decoding using a standard static network the second of these operations is of minor importance. The structure of the network is left intact and instances which fall out of the beam are made inactive to prevent token passing occurring within them. For the dynamically created network, the second operation is at least as important as the first. The potential size of a fully expanded tree structured network is huge and so space is only allocated for instances which are active. Consequently, both computation and storage requirements are controlled by pruning of the network.

Standard beam pruning is used in which tokens which are relatively unlikely are assumed to have zero probability. All pruning is performed at the model level rather than the state level since the network is allocated at model level. State based pruning would not reduce total storage requirements and would have minimal impact on the overall computational complexity.

The combined likelihood, which is the sum of the likelihood of the most likely token in each instance together with the language model likelihood of the instance, is used for pruning. The instance with the highest combined likelihood is used to set the top of the beam and any instances more than a fixed beam width less likely are pruned from the search.

Since the network is created dynamically it is not always sensible to remove instances when they are pruned because it can be difficult to reconnect them to the rest of the network if they need to be recreated. Consequently three different operations are performed during pruning depending on where the instance occurs in the network.

- Erased.

  A node with no predecessors will never receive active tokens and can be completely removed from the network and the storage reclaimed. If the model instance labelled (a) in figure 5.4 is pruned it will be erased.

- Deleted.

  A node with no followers can be removed from the network in a manner that allows it to be recreated later should it become relatively more likely. The instances labelled (c) in figure 5.4 would be deleted.

- Halted.

  A network node with both predecessors and followers cannot easily be recreated and reconnected to the network. If such an instance is pruned it is not removed from the network but token passing within it is halted until it is reactivated or it can be erased. This saves computation but not memory. If the model instances labelled (b) in figure 5.4 were relatively unlikely they would be halted.

Pruning is performed every frame and the instances are processed in the same order as used for token propagation. When an instance is relatively unlikely and falls out of the beam it is pruned as described above. Processing the instances in this order ensures that a sequence of models can be erased in one pass.

5.3.1 Maximum Model Pruning

Due to the potentially very large size of the network, there are periods for which the number of active models can be very high, often between ten to a hundred times the average. This has little effect on the total computation since these periods occur rarely and represent only a small

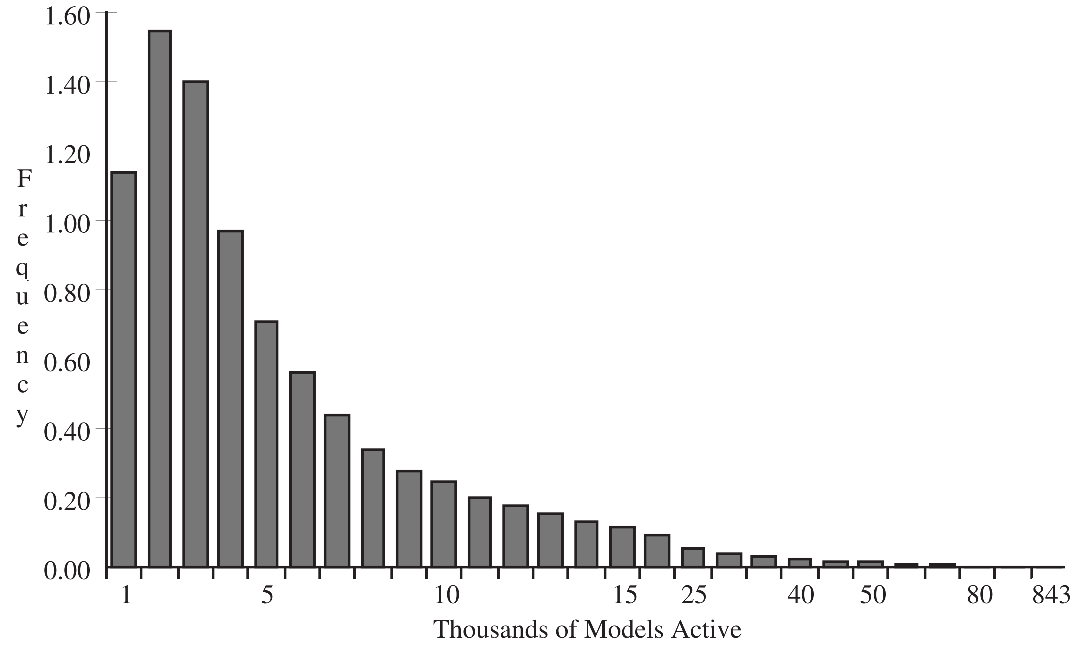

Figure 5.7: Number of models active during decoding.

proportion of the speech. However, the memory required to store this many active instances may be unacceptable.

Figure 5.7 shows a histogram of the number of active model instances during a typical 5k WSJ test. The peak number of active models is over ninety times greater than the average number.

During the periods when many models are active the uncertainty about word identity is relatively high and many partial paths will have similar likelihoods. Because so many hypotheses are being actively considered, the beam width used (in terms of likelihood) can be lowered without introducing a significant number of search errors.

During periods of greatest uncertainty rather than using a beam width defined by likelihood differences one based on the number of active models is used instead. Thus, an upper limit is placed on both computational and memory requirements. This has little effect on the total computation required but allows a limit to be placed on the size of the network and the storage requirements of the decoder. Limiting the process size in this way allows efficient use of the computational resources (by avoiding excessive thrashing of the virtual memory when the process becomes larger than physical memory).

It was found that this *maximum model pruning* could be incorporated into the recogniser without significantly increasing error rates (because of additional search errors caused by the increased pruning) whilst dramatically reducing the peak memory requirements.

5.3.2 Word End Pruning

When using the recogniser with fully connected grammars and backoff language models, a third type of pruning was implemented.

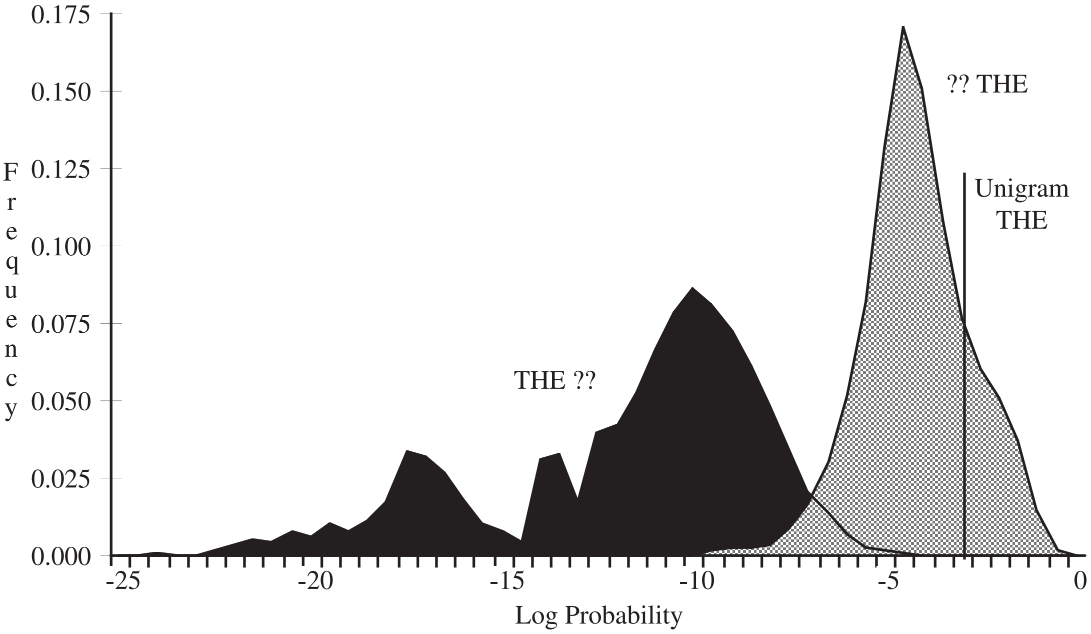

Figure 5.8: Variation in the probability of bigrams including the word *THE*.

Studies of the language model revealed that variations in the probability of a specific word were much lower than the variations over all words. Figure 5.8 shows how the variation in log probability for the word *THE* over the official 5k bigram language model compares with the variation of the probabilities of words following *THE*. The variation in the probability of a particular word over different histories is much lower than the general variation independent of word identity. For this language model, the average over all words of the standard deviation in the natural logarithm of the probability of the word is 1.32 whereas the average standard deviation of the probabilities independent of the word identity is 5.28.

This reduced variability together with the fact that the uncertainty in the identity of the current word is lower at the end of words than at the start [41] suggest that a separate tighter beam could be used to control extension of paths at word boundaries. This type of pruning can also be used with linear networks and in this case also substantially reduces the number of tokens which need to be propagated between words [71]. The major source of variation in the likelihood of a particular word is due to the language model and this variation tends to be relatively small compared to the beam widths employed during recognition. Consequently when several word ends are active, it is only necessary to propagate tokens from the relatively likely ones.

This *word end pruning* was implemented by using a separate beam to control propagation of tokens from word end instances. Extension of the network from word end instances that are more than a fixed likelihood less likely than the most likely word end instance is blocked. Word end pruning can significantly reduce the number of active word end instances without increasing the number of search errors.

5.4 Algorithm

The overall algorithm can be summarised as;

- Load parameterised speech file.
- Initialise a network for each system. (For example a male one and a female one).
- For each observation in utterance;
  - Determine current beam width.

    This will either be a fixed likelihood or chosen to limit the number of active models to a predetermined maximum.
  - Prune network according to current beam.

    Each node in the network has a combined log likelihood which is used to determine which instances should be pruned. The combined log likelihood is the sum of the log likelihood of the most likely token in the instance with its language model log likelihood. When the combined log likelihood of an instance is more than the beam width less likely than the most likely instance it is pruned.
  - Perform token propagation within model instances.

    The tokens in each state of each model instance are updated after processing the next observation. The model with the highest combined likelihood is found and this likelihood sets the top of the beam for the next frame.
  - Perform token propagation between instances.

    The token in the exit state of each instance is updated and propagated into following models which may need to be reactivated or created. At word ends, the language model likelihood is added to the token and the traceback information updated to reflect the end of the current word. When necessary the network is extended and new model instances and word ends are created although this process can be blocked at word ends because of word end pruning or path merging through domination.
- Find most likely utterance final word end instance.
- Perform traceback and recover most likely hypothesis.
- Free network and other storage.

5.5 Lattices

Despite the relatively high efficiency of the decoder the recognition process is still computationally expensive. During development it is necessary to first optimise and then measure the accuracy of systems with a set of development test data to allow the performance of different modelling techniques to be accurately estimated. This means that the decoder will be run several times over the same set of development data with only relatively small changes made to the system each time. It is possible to speed up system development by initially producing a lattice of likely hypotheses for each utterance rather than just finding the single most likely hypothesis. This lattice can then be used to constrain (and therefore speed up) the decoder when it next processes that utterance. This scheme can also be used to decode sentences with computationally expensive models. An initial pass using (for instance) triphone acoustic and bigram language models can generate a lattice of hypotheses that can then be rescored using a larger language model and more detailed acoustic models which may be too computationally expensive to use in a single pass.

The decoder is well suited to generating lattices of hypotheses that are both deep and accurate with little modification and virtually no computational overhead compared to finding only the most likely hypothesis.

5.5.1 Lattice Generation

Only a few simple modifications are needed to modify the decoder to generate a lattice of hypotheses.

Rather than discarding all but the most likely partial path when these merge at word ends it is possible to retain information about them all to allow lattice traceback [27]. This procedure is easiest to implement when the decoder merges paths using token recombination.

When only the most likely hypothesis is required the language model likelihoods are added to the tokens in word end instances and the traceback information updated. The tokens from equivalent partial paths recombine and only the most likely survives to propagate into the following network. When a lattice of hypotheses are needed the less likely tokens are not discarded but are linked to the most likely one and the combined structure propagates into the following network. The calculations in the remainder of the network are only performed on the most likely token but when traceback occurs at the end of the sentence all of the tokens are used to construct a lattice of hypotheses

Since calculations are only performed on the most likely token, it is used to decide the most likely state sequence through the following words and the other tokens are assumed to follow the same path. This is equivalent to assuming that the position of the word boundary is the same for paths from each of the word end instances. For a bigram network only instances of the same word will be linked into an equivalence chain and so the assumption is the same as the word pair assumption described in Section 4.2.3.

At the end of each utterance traceback proceeds separately through each of the linked tokens and a lattice of alternative hypotheses is constructed.

A separate node is produced for each combined set of tokens and each of the tokens creates a

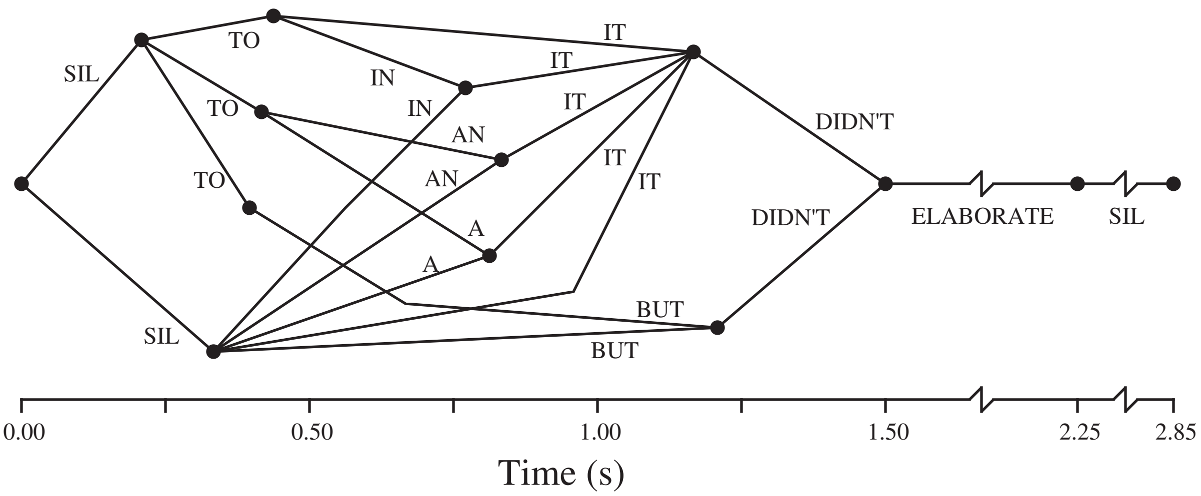

Figure 5.9: An example lattice.

link into that node. Each node has an associated time which is the time at which the combined token propagated into the following network and is the end time of the various word instances that were linked together. Each arc has an associated word (and pronunciation) identity, acoustic likelihood and language model likelihood and forms a link between two nodes which define the start and end time of the word hypothesis. This ensures that the different acoustic and language model contexts are incorporated in the topology of the network.

Because of the different contexts in which a word can occur, there can be many arcs representing similar hypotheses for the same word covering a similar portion of the utterance. These will have differing acoustic scores (because of different phonetic contexts represented with different model sequences) and/or different language model likelihoods. This replication can be avoided by using acoustic models which are only dependent on the word internal context and short span language models. However this would result in lower recognition accuracy (in terms of both the one-best and lattice-best word error rates) and must be offset against the reduced lattice size.

An example lattice generated with cross word triphones and a bigram language model is shown in Figure 5.9. Note the multiple instances of the word *TO* due to different right phonetic contexts, `ih` for *IT* and *IN*, `ax` for *AN* and *A* and `b` for *BUT*, each with different acoustic likelihoods and word boundary times.

5.5.2 Lattice Accuracy

The quality of the lattices (and the model set that generated them) can be measured by determining the error rate of the lattices with respect to the actual sentence spoken in a similar way to that in which the accuracy of a single hypotheses is assessed. The most accurate path through the lattice is found using a best first dynamic programming search that minimises the total number of word errors. When the correct sentence exists in the lattice, it will be found otherwise the total number of insertion, deletion and substitution errors will be calculated. The number of word errors and number of sentence errors can both be used as performance figures.

The presence of out-of-vocabulary (OOV) words makes it difficult to perform comparisons across test sets as these will often be the dominant cause of errors in the lattice.

To enable comparisons between test sets, bounds on word and sentence error rates can be used instead. Both upper and lower bounds are found by first performing the dynamic programming search to find the path which minimises the number of word errors. The lower bound is found by only counting differences between the reference and the lattice path as errors when the reference word occurs in the recognition vocabulary. This assumes that OOV words would be recognised correctly if they were added to the recognition vocabulary and this provides a best case estimate of the error rate of an unlimited vocabulary system. The upper bound is found by counting all differences as errors and this is the actual error rate in the lattice. However, the upper bound can be heavily influenced by the number of out-of-vocabulary words in the test set and so the lower bound tends to be a more accurate indication of the relative difficulty of test sets.

These figures will be used when lattice error rates are quoted. When only a single figure is quoted, this is the upper bound which makes no allowance for out-of-vocabulary words. Otherwise, a range will be quoted with the lower bound taking out-of-vocabulary words into account and the upper bound making no allowance.

5.5.3 Lattice Pruning

The lattices generated by the decoder are potentially very large but a significant proportion of each one will be relatively unlikely. These portions will rarely form part of the spoken sentence or be part of the most likely path when the lattice is rescored. Consequently the lattice can often be significantly reduced in size without reducing the lattice accuracy or altering the results obtained if they are rescored.

Since the lattice contains likelihoods over the whole of the utterance the likelihood of complete paths can be used to accurately determine which parts of the lattice are most likely. The log likelihood of the most likely path through a particular arc in the lattice can be found with an efficient three pass scheme.

- The forward log likelihood of each node is found. This is the likelihood of the most likely path from the start of the lattice to the node.

- The backward log likelihood of each node is found. This is the likelihood of the most likely path from the end of the lattice to the node.

- Finally the log likelihood of the most likely path through each arc is found. This is the sum of the arc acoustic and language model log likelihoods together with the forward log likelihood of its start node and the backward log likelihood of its end node.

The initial two passes use best first searches to minimise the computation required. This is similar to the final phase in some decoding schemes (which use time-synchronous searches to provide the lookahead information available from the lattice) [1]. The resulting log likelihood of each arc is compared with the log likelihood of the most likely sentence hypothesis. Any arcs which are more than a predetermined beam width less likely are removed from the lattice as are any nodes which become disconnected as a result.

Since pruning is based on the likelihood over the whole utterance it can be relatively tight without substantially reducing the accuracy of the lattices. The following chapter gives some results which show the effect of pruning on the accuracy, generality and size of lattices.

5.5.4 N-Best Generation

Some recognition techniques require a hypothesis for the whole of an utterance in order to calculate its likelihood. For example it may not be possible to calculate this likelihood word by word because of long span effects such as average speaking rate. These techniques are commonly implemented by augmenting the acoustic and language model likelihoods for the N most likely hypotheses with the new likelihoods and then reordering the hypotheses to find the overall most likely. The lattices generated by the decoder can be used to generate very deep N-best lists for such schemes.

An A* search through the lattice which does not merge paths will generate N-best lists many thousands deep using relatively small amounts of computation and storage. Often, only different word sequences are required and, since the lattice contains arcs for different pronunciations, hypotheses with the same word sequence but different pronunciations need to be merged.

5.6 Rescoring

Often these lattices will be of sufficient quality to allow lattice rescoring to be used for evaluation of new acoustic and language models without needing to perform a complete utterance decode. The lattice is used to constrain the number of paths that are considered and can substantially reduce the amount of computation required. The use of lattices in this way is similar to the final rescoring phase of some progressive multi-pass decoding techniques [4].

This allows rapid system development as acoustic rescoring of a lattice is more than an order of magnitude faster than full decoding and language model rescoring is almost three orders of magnitude faster.

However care must be taken to ensure that the system used to rescore the lattices is not too different from the one used to generate them and that the lattices provide adequate variety to ensure that the results obtained will be representative of those that would be obtained during a full decode.

5.6.1 Acoustic Rescoring

Although acoustic rescoring could be performed (probably more efficiently) with an A* style search using the lattice to provide lookahead information, instead, for simplicity, the standard decoder is used. The lattice is used to provide a constraining grammar and optionally language model likelihoods.

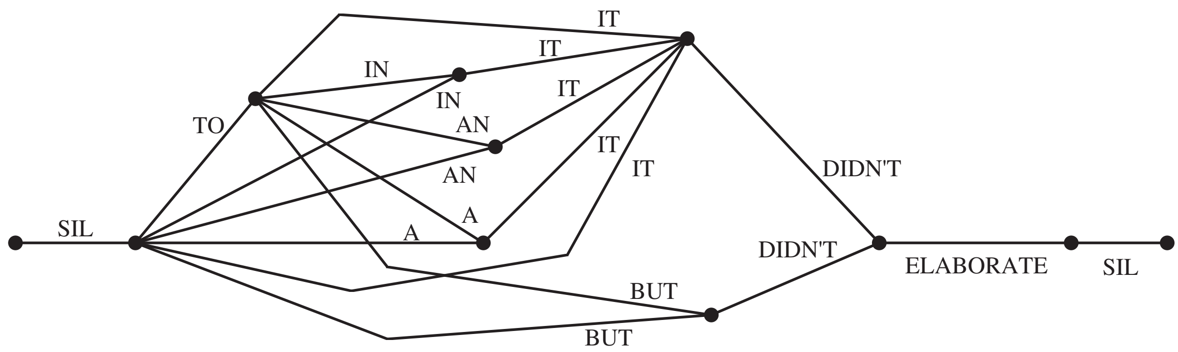

Figure 5.10: A lattice without acoustic context.

The lattice is rebuilt to remove unnecessary acoustic contextual dependencies in a recursive merging scheme in which equivalent arcs representing the same word that have different pronunciations or phonetic contexts are merged. This preserves the language model likelihoods because these are dependent only on the word sequence and allows the acoustic rescoring to use language model likelihoods taken from the lattice. However both new language and acoustic models can be used in a single pass with the lattice just providing a constraining grammar. Figure 5.10 shows the lattice from figure 5.9 with acoustic context dependencies removed. Note the multiple instances of *SIL* and *TO* which had different end times and right contexts have been replaced with single arcs. Such lattices no longer contain meaningful acoustic likelihoods or word boundary times.

The lattice is used to provide the decoder with a list of possible word extensions to each partial path as well as indicating the points at which differing partial paths may merge (when they end at the same lattice node). The reduction in the number of hypotheses that need to be considered dramatically reduces the computation required to evaluate new acoustic models, normally by more than an order of magnitude. When the acoustic models are well matched and relatively accurate, the difference in overall word error rate between lattice rescoring and unconstrained decoding will be only a small fraction of a percent and normally can be considered insignificant.

5.6.2 Language Model Reapplication

The lattices can also be used to evaluate new language models in a similar way. The rescoring could just involve variation in the weighting between the acoustic and language model likelihoods, adding a word insertion penalty or using a completely different language model, such as a trigram in place of a bigram.

Language model rescoring can be accomplished by an A* search in which the acoustic scores are used to provide the lookahead whilst the new language model generates the language model likelihoods that are combined with the acoustic likelihoods from the lattice as paths are expanded. To ensure that this is efficient for large lattices, the language model can also provide information about which paths it considers to be equivalent (although this is only possible for some types of language model). This allows path merging to occur and reduces the number of distinct paths that must be considered. However path merging makes it impossible to generate N-best hypotheses at the same time as applying a new language model.

Instead of performing an A* search to find the best hypothesis, it is often more useful to be able to rebuild the lattice to incorporate likelihoods from a new language model with acoustic likelihoods from the original lattice. This resultant lattice can then be used to generate N-best lists or for acoustic rescoring using new language model likelihoods from the rebuilt lattice.

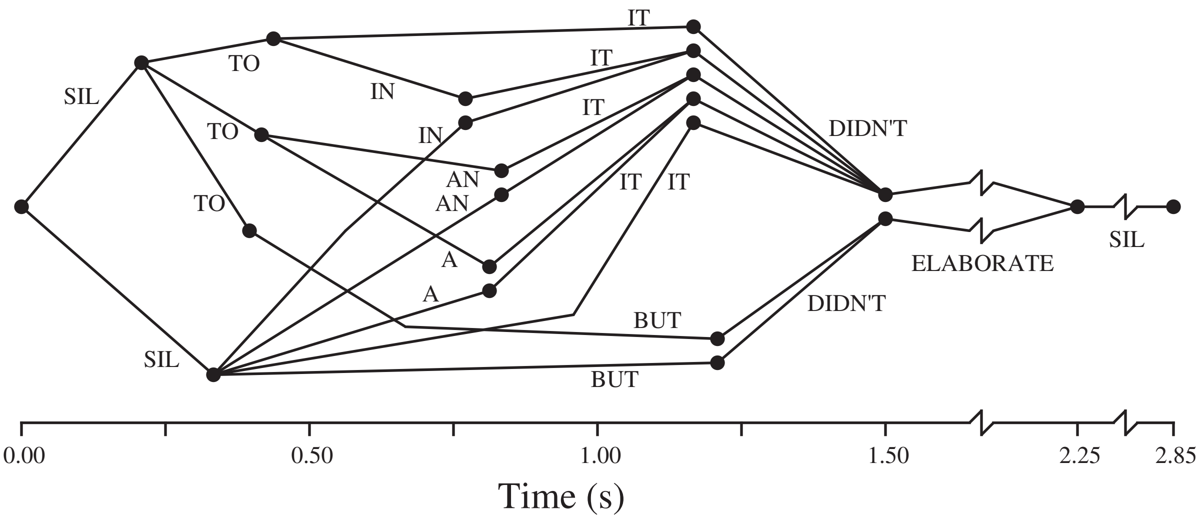

Figure 5.11: A lattice incorporating a trigram language model.

Rebuilding can be accomplished by a simple recursive procedure similar to that used to convert a lattice into the finite state syntax needed for acoustic rescoring. Normally the lattice will expand as it needs to incorporate the dependencies present in the new language model as well as those already present in the lattice. For instance, expanding a lattice generated with a bigram using a trigram language model will duplicate arcs in order to ensure that each node will have the unique two word history needed to calculate trigram likelihoods.

The lattice from figure 5.9 is shown after expansion with a trigram in figure 5.11. Note the single node where all the instances of IT combined in figure 5.9 has been duplicated five times for the trigram contexts *SIL IT*, *TO IT*, *IN IT*, *AN IT* and *A IT*. This allows the different language model probabilities for *DIDN'T* (which are dependent on the previous two words) to be placed on distinct arcs in the lattice. Each of these arcs will have the same acoustic score and time span since they were created by duplicating the single arc representing *DIDN'T* following *IT* in the bigram lattice.

5.7 Summary

This chapter has described a decoding architecture suitable for large vocabulary continuous speech recognition. The decoder can use cross word context dependent acoustic and long span language models in a single pass generating a lattice of likely hypotheses for each utterance.

A variety of pruning techniques in addition to standard beam pruning were developed to improve decoder efficiency.

Several lattice manipulation tools have been produced to allow the pruning, accuracy measurement and rescoring of lattices. These have reduced the time required for system development by an order of magnitude allowing a wide variety of acoustic and language modelling techniques to be evaluated.

Results in the next chapter will show that the decoder used in conjunction with the acoustic modelling techniques described in chapter 3 can produce recognition systems with state of the art accuracy.

Chapter 6

Experimental Results

This chapter contains details of recognition experiments performed with the dynamic network decoder using models constructed by decision tree based clustering. Both the decoder and the tree based models have been used for a variety of continuous speech recognition tasks ranging from small vocabulary speaker dependent systems through to very large vocabulary speaker independent ones. The use of decision trees enabled the construction of models for unseen contexts allowing recognition of arbitrary words and the use of cross word context dependent modelling techniques giving state of the art performance. The novel decoder design has allowed the efficient use of these models and the lattice based decoding approach has enabled rapid evaluation of the accuracy of different modelling techniques.

This chapter contains results for two of the tasks attempted. Initially the medium vocabulary DARPA Naval Resource Management task was used for developing the acoustic modelling techniques. These techniques were then applied to the large vocabulary North American Business News domain based on the Wall Street Journal Database. Details of these tasks can be found in appendix A.

6.1 Recognition System Architecture

All recognition systems were based on hidden Markov models using continuous density diagonal covariance mixture Gaussian output probability distributions and fixed state transition probabilities. The output probability distributions could be shared at the state level but there was no sharing of mixture components. (The models were continuous density rather than tied-mixture or semi-continuous).

Model parameters were estimated using embedded Baum-Welch re-estimation in which a composite model for each complete sentence was used to probabilistically assign observations to states and then update the model parameters (see section 2.6).

The complexity of the models was increased in an incremental fashion using the mixture splitting technique described in section 3.5.2. Experiments suggested there was little benefit in varying the number of components in different distributions and so the number of components in the mixture Gaussian distributions does not vary across speech models. (Although, as described below, the number of components of the non-speech distributions can be different).

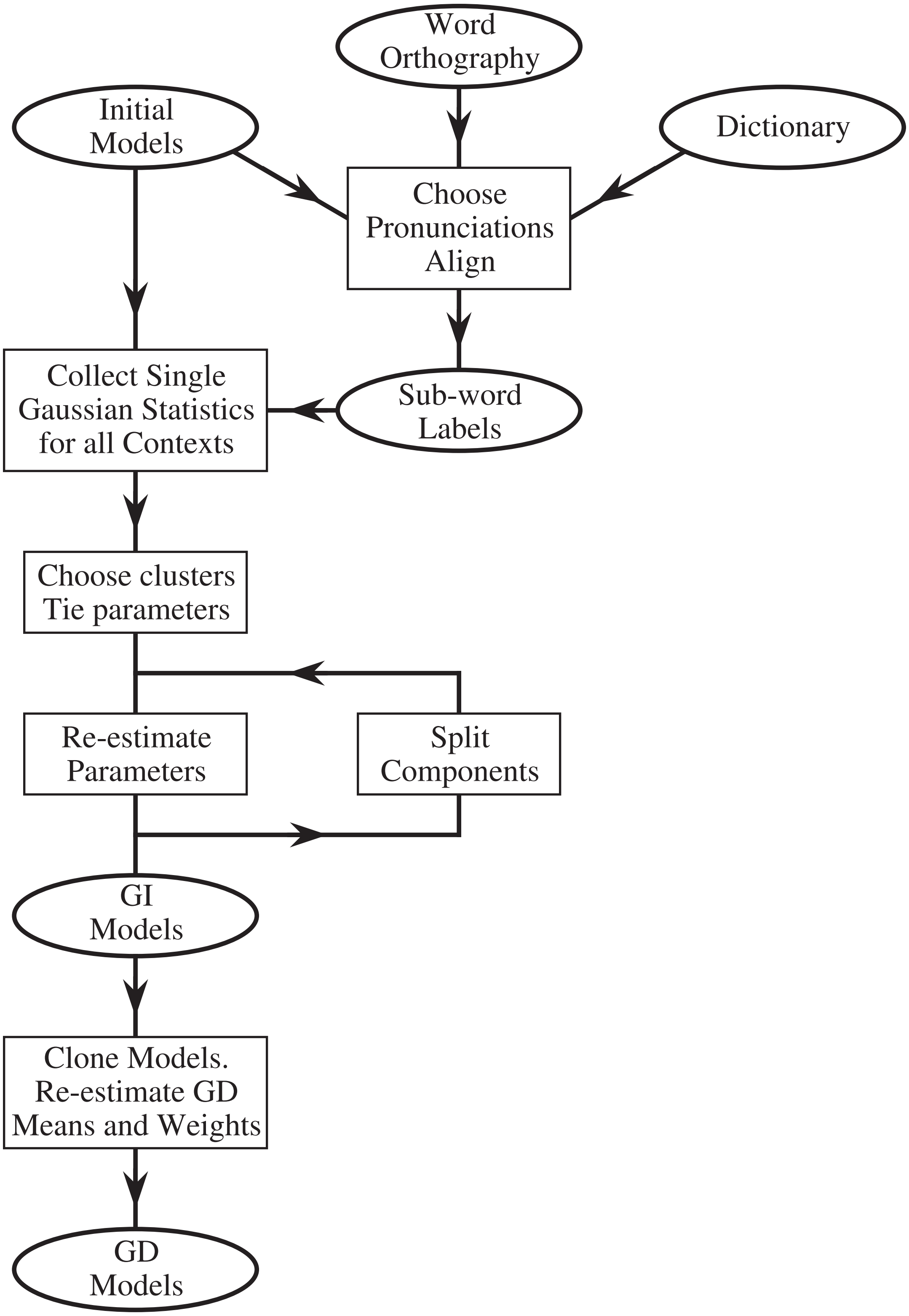

Figure 6.1: System building procedure.

All systems were constructed in approximately the same manner. An initial set of models was used to compute statistics for the different contexts occurring in the training data. These statistics were then used in a clustering procedure which decided how state distributions would be shared and then produced an initial set of tied state models. The complexity of these models was increased in a step by step fashion until performance plateaus or a fixed level of complexity has been reached. This results in a set of gender independent (GI) models which can be used to produce gender dependent (GD) ones by cloning the system and then re-estimating the component means and weights of each system using data from only one gender. The whole of this training procedure is shown in figure 6.1.

Most models employ a simple architecture with three emitting states. Each emitting state has a self transition and a transition to the following state. This enforces a minimum duration of three frames since at least one observation must be generated by each state.

The only speech models with a different topology are the optionally released stops in the Resource Management dictionary (`dd`, `kd`, `pd` and `td`). For these models an extra transition from the second emitting state to the exit state is added. This transition allows the final emitting state (notionally representing the release) to be skipped and reduces the minimum duration of a path through the model to two frames. Figure 6.2 shows the different topologies employed (including those of the the silence models discussed next).

Models are also required to represent the periods of an utterance without speech. These periods, such as the beginning and end of the utterance and pauses between words, are often described as silence as there is often little airflow or acoustic energy. However, a variety of phenomena must be represented in addition to real silence. These include breath noise, tongue clicks and general background noise. Initially (in a similar fashion to [91]) two models were used. A `sil` model represented silence at the beginning and end of an utterance and used the three state left-to-right architecture used for most speech models. A `sp` model represented optional pauses between words and used a single state model which could be entirely skipped (by using a tee transition). However this simple silence modelling, in which the silence models were treated much like any other phone, was found to be inadequate when the word internal context dependent systems were modified to use cross word context dependent models.

Silence models needed to be treated differently from speech models for several reasons.

- Always context independent.

  There is no reason for supposing that acoustic characteristics of the periods of non-speech are significantly influenced by the surrounding phonetic context. (Although these periods may effect the production of the surrounding phones).

- No temporal structure.

  The periods of silence are of arbitrary duration with no obvious progression between periods of silence, breath noise or other phenomena. The left to right topology employed for speech models is unsuitable as less structure is needed.

- Greater detail.

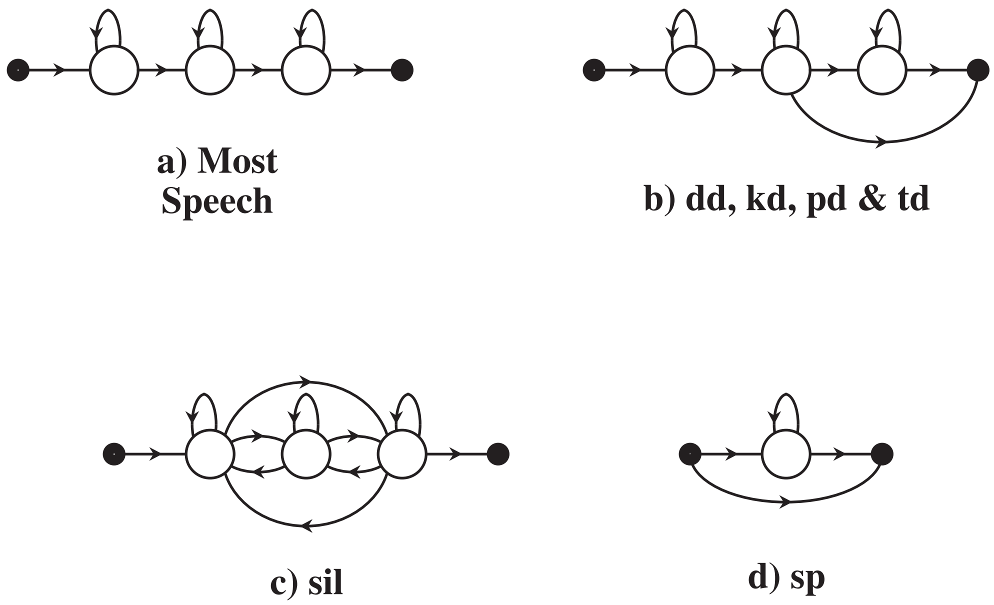

Figure 6.2: Model topology.

Because silence models are context independent, represent a large proportion of the data and need to model a variety of phenomena it is desirable to use distributions with more components than the speech models. Typically twice as many components are used in each non-speech state distribution as in the speech models. Further increases in the number of components did not lead to further improvements in accuracy.

This led to the use of two silence models which represent different periods of non-speech.

- `sil`

  `sil` represents longer periods of non-speech including the start and end of an utterance (where it always occurs) and long pauses between words (which are optional). `sil` is considered as a separate phonetic context since the articulators will have a chance to return to rest positions during long pauses between words and will start and finish in rest positions at utterance boundaries. `sil` is a three state model but allows transitions between all emitting states as there is no temporal structure to the periods it represents. However the only transition from the entry state is to the first emitting state and the only transition to the exit state if from the last emitting state thus enforcing a minimum duration of two frames. A longer minimum duration may be better, however, it is convenient in the decoder to limit the number of states in all models.

- `sp`

  `sp` represents brief pauses between words. It consists of a single state which may be skipped and is ignored when expanding context since during the short pauses which it represent the articulators will not be able to move significantly. This means that the phone sequence `sil ih z sp dh ax sil` would be expanded to the triphone sequence `sil sil-ih-z ih-z+dh sp z-dh+ax dh-ax+sil sil`. This architecture allows `sp` to be present at all word boundaries (except those where `sil` occurs) and thus simplifies the decoding network architecture.

It was found that these changes reduced the number of spurious insertion errors in which words were recognised if brief noises occurred during long periods of silence. This significantly improved recognition accuracy.

6.2 Resource Management Experiments

Experiments on decision tree clustered models using the tree structured dynamic network decoder were performed on the DARPA Resource Management task. Appendix A describes both the task and the associated corpus in detail.

All systems were trained on the *SI109* section of the database and tested on the four official test sets using the standard word pair grammar (with a perplexity of around 60).

Top down decision tree based clustering restricts the sharing of parameters because of the finite set of questions that can be used to split each node. Previous work has indicated that using knowledge based approaches to group contexts in a restricted fashion can lead to poorer modelling accuracy than bottom-up approaches that do not restrict the way in which parameters can be shared [32]. The principal advantage of the decision tree approach is the ability to construct models for unseen contexts. This is vital when producing cross word context dependent systems as the majority of contexts appear very few, if any, times. Bottom-up approaches are not suited to this task. Consequently the effect of the decision tree constraints on the accuracy of the acoustic modelling was assessed using a system based on word-internal triphone models. The training portion of the database was designed to ensure coverage of the different words and so all word internal contexts occurred in the training data.

A recognition system for the Resource Management database was described in [92] which used tied-state fully continuous mixture Gaussian word internal triphone models. The system was constructed using the bottom-up agglomerative clustering technique described in section 3.5.2. This approach requires the same set of statistics as that needed by the decision tree based clustering approach described in section 3.7. Consequently it is possible to produce both decision tree clustered and agglomeratively clustered state tied systems from the same set of untied triphone models. Results from [92] on the various Resource Management evaluation test sets are reproduced in table 6.1 labelled “agglomerative”. This system used a set of 7111 triphone models constructed from 1655 tied states each composed of a 6 component mixture Gaussian probability distribution.

A decision tree based system was also constructed from the same triphone statistics. The architecture of this system was not optimised separately but was as similar as possible to the original system. The same minimum occupancy was used for the state distributions and the threshold used to decide when to stop splitting nodes was chosen to ensure the systems were of comparable size. The decision tree based system used 1581 tied state distributions each also consisting of a 6 component mixture Gaussian and so both systems had approximately 750,000 acoustic parameters. Neither system used cepstral means subtraction (the parameterisation of the data is described further in section A.1).

System	Feb'89	Oct'89	Feb'91	Sep'92
Agglomerative	4.10%	4.84%	3.78%	8.05%
Decision Tree	3.87%	4.99%	3.74%	7.31%

Table 6.1: Word error rates for agglomeratively and decision tree tied state systems.

As table 6.1 shows the performance of both systems is similar. The constrained clustering procedure using decision trees does not adversely effect performance and would, if required, allow the construction of models for unseen contexts.

The ability to produce models for unseen contexts makes it easy to produce systems incorporating cross word triphone models. Since these explicitly model co-articulation effects across word boundaries they should provide a more consistent and accurate representation of speech and thus produce more accurate results.

Much of the previous work using decision trees ([6], [29]) had performed the clustering at the model rather than the state level. However experience with word internal systems indicated that clustering at the state level allows greater flexibility and leads to more accurate models. To evaluate the effects of clustering at the state rather than the model level two cross word triphone systems were constructed. Both had similar complexity with approximately 2400 state distributions each a 4 component mixture Gaussian for a total of around 800,000 parameters. The model tied system had 800 generalised triphone models whereas the state tied system had over 11,300 distinct triphone models. Again neither system used cepstral means subtraction for data normalisation.

Clustering was performed gender independently (so each cluster was evaluated for a single Gaussian per state) with the same initial set of statistics for each context in the training data. The model based system was made by tying the trees for each of the three states together and evaluating each question by the increase in likelihood summed over all three states. The state based system used the optimal question for each state and produced three decision trees for each phone.

System	Feb'89	Oct'89	Feb'91	Sep'92
Model tied	3.71%	4.58%	4.19%	7.03%
State tied	3.12%	3.76%	3.38%	6.25%

Table 6.2: Word error rates for state and model based decision tree systems.

The results of both systems are shown in table 6.2. Allowing tying at the state level (and so producing more distinct models which can more accurately capture consistent contextual variation) improves the accuracy by about 15%. Similar results occur with the larger Wall

Street Journal Task [60]. The state tied system used more distinct models and so less merging could be accomplished by tree structuring the network and recognition was computationally more expensive. However the increased accuracy of the models results in better search locality and partially offsets the increase in network size. Consequently the difference in decoding complexity between the model and state tied systems is relatively small (approximately 25%).

The architecture of the cross word triphone system was further improved. A system with 1778 tied states each with a 6 component mixture Gaussian distribution was constructed. This system did use cepstral means subtraction during both training and recognition. The word error rate for this system on the four evaluation tests are shown in table 6.3 together with the word internal triphone system from [92] and the context independent monophone system from [91]. The addition of cross word context dependency resulted in a 32% drop in word error rate compared to the system with word internal context dependency and over 50% compared to the monophone system. (Note that these figures and all other comparisons between error rates refer to relative improvements unless otherwise stated).

If this cross word context dependent system were used with a conventional static network decoder, recognition would be much more computationally expensive than one based on word internal triphones. However, the tree structured network means that even for this task, which does not benefit greatly from sharing as each word has on average only 60 possible followers, the number of active models required (shown in the final column of table 6.3) only rises by about a factor of two. (Although the number of models active is approximately doubled the actual computation only rises by 50% due to improved efficiency of the dynamic decoder). Recognition requires around twenty seconds per utterance on a HP735/125 with beam widths chosen to avoid any search errors.

System	Feb'89	Oct'89	Feb'91	Sep'92	(Models)
Monophone	5.7%	7.3%	6.0%	9.7%	
Word Internal Triphones	4.10%	4.84%	3.78%	8.05%	(760)
Cross Word Triphones	3.05%	2.91%	2.46%	5.78%	(1600)

Table 6.3: Word error rates for optimised word internal and cross word triphone systems.

6.3 Decoding Complexity

In order to compare the computational complexity of different tasks, a normalised measure of the computation required to decode speech is needed. When only a single type of computer is used the ratio of actual processing time to the duration of the speech would allow accurate comparisons. However, since many types of computer were used, a different quantity is needed to give an indication of the relative computational load of the different experiments.

The number of models which were active during each frame gives an indication of the relative computation. The complexity of most of the tasks involved in decoding (token propagation, pruning and network construction) varies linearly with this number. However, the evaluation

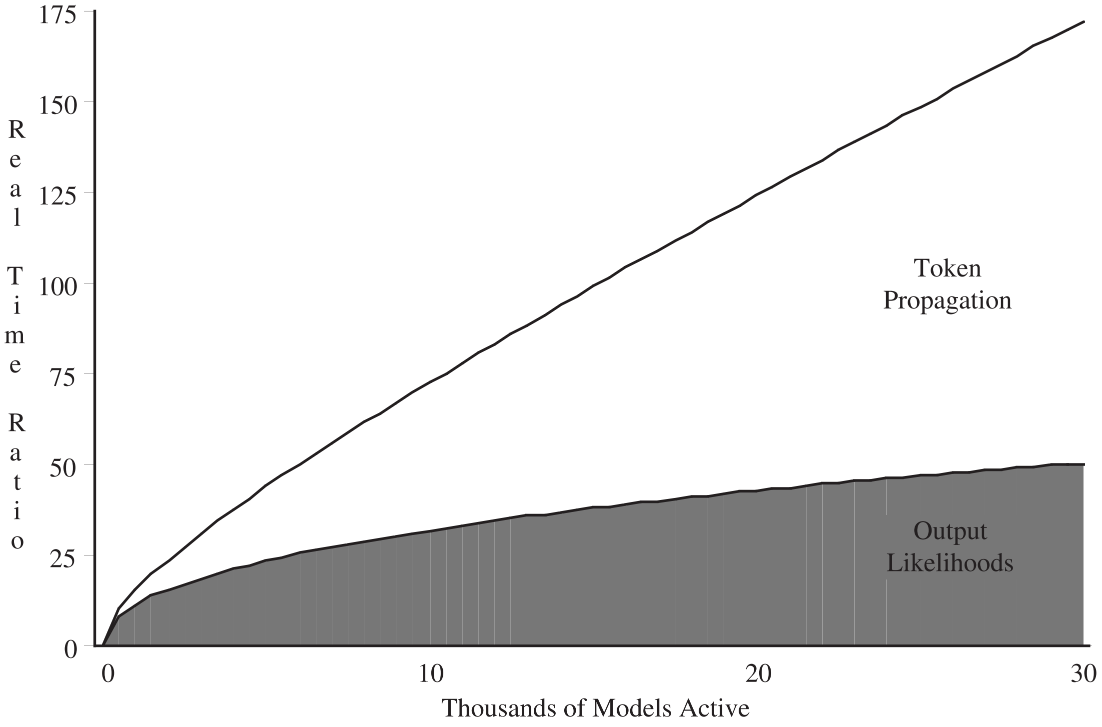

Figure 6.3: Variation in run time with number of active models: Light pruning.

of the output likelihoods does not scale linearly because of the tied-state architecture of the models. The output likelihood calculations can be shared and need only be calculated the first time a distribution is used each frame. This value is recorded and any states sharing the distribution also share the computation. The calculation of output likelihoods typically represents a significant (although normally not dominant) proportion of the total computation (typically between 30% and 60%). Consequently the number of models active each frame only gives an approximate indication of the total computation. The evaluation of the output likelihoods will depend upon the degree of tying present in the system. The computation this requires must be estimated from the system size and task complexity. For large systems with complex distributions and few models active the output likelihood calculations will represent about 60% of the total computation. For fewer, simpler distributions or a large number of active models the corresponding figure is nearer 30%. In practice the evaluation of output likelihoods may not be a significant computational overhead as they are amenable to a variety of speed-up techniques (including both vector processing and vector quantisation [11]).

Figures 6.3 and 6.4 show how the actual run time of a 5k WSJ experiment (see next section) varied with the number of active models. These experiments were performed on a SUN20 capable of SPEC-FP 106.0 and SPEC-INT 93.0. (Note the model set used had relatively complex probability distributions with 12 components in each of 6,400 shared distributions. Resource management systems tend to have fewer, simpler distributions and the output likelihood calculations are correspondingly less significant at a particular number of active models. For example the most accurate Resource Management system described above was almost twice as fast as the 6,400 state system for the same number of active models (1600). This is due to the smaller number of states (1778) which used simpler 6 component distributions requiring less computation for output likelihood calculations.). As these figures show, the computation does not vary in direct proportion to the number of active model as the computation of output likelihoods becomes more significant when the number of active models decreases. However in the regime in which experiments are typically performed (chosen to avoid search errors) the contribution of output likelihood calculations is reasonably static at between 30% and 40%. Similar experiments on an HP735/125 capable of SPEC-FP 201.0 and SPEC-INT 103.0 ran approximately two to three times as fast. On this machine (excluding output likelihood calculations) approximately 500 active models can be processed each frame (10ms) in real-time.

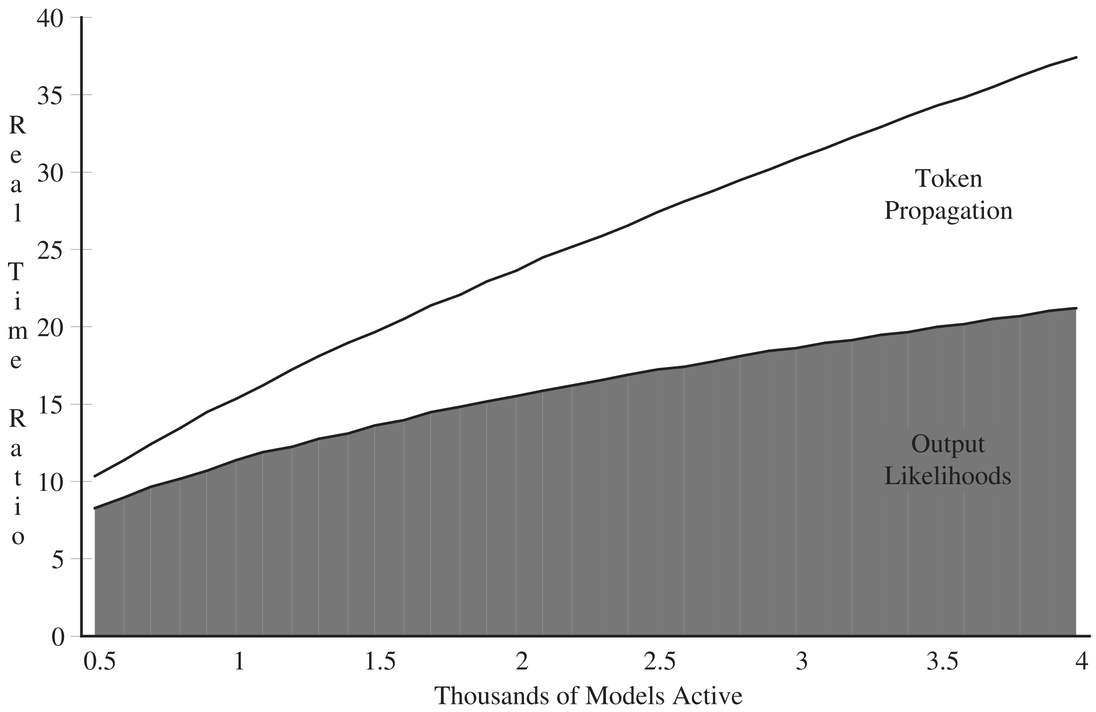

Figure 6.4: Variation in run time with number of active models: Heavy pruning.

6.4 Wall Street Journal Experiments

The resource management task used a relatively small vocabulary and heavily constrained grammar. Realistic tasks must process a larger vocabulary with a less constrained grammar.

The ARPA Wall Street Journal task was the successor to Resource Management and used a larger vocabulary as well as a fully connected grammar. Appendix A describes the task as well as the associated database.

For larger tasks, computational complexity becomes increasingly important and additional pruning is required to maintain efficiency.

6.4.1 Pruning

The computation required for decoding is controlled by pruning of the search space. Simple beam pruning, which discards tokens that are more than a fixed beam width less likely than the most likely token for the frame, is the primary method of controlling the complexity of the search. However two additional forms of pruning were introduced to further reduce the computational load.

Maximum model pruning (section 5.3.1) specifically limits the complexity of the search (in both size and computation required) by placing an upper limit on the number of active models. This was required to ensure that the decoding process would fit in the physical memory of the computer used. Since a great many alternatives are being actively considered reducing the effective likelihood beam (because of the limit on models) should not introduce search errors.

Word end pruning (section 5.3.2) introduces a separate beam for word end tokens. This word end beam can be tighter than the general beam for two reasons. Firstly there is a greater degree of certainty about word identity at word ends rather than word starts. Secondly the range in language model likelihoods for a particular word over different histories is much lower than the general beam width.

Each of these three forms of pruning was investigated to determine how many search errors are produced under different pruning conditions. These experiments (described below) all used the same set of cross word triphone models with 6,400 tied states. These were produced for the November 94 Evaluation and described in detail in 6.4.3 as 94'HMM-1. The figures described in section 6.3 were produced using these models and so these graphs can be used to directly determine the run time of the experiments described below. The remainder of the experiments in this chapter investigate modelling accuracy rather than search performance and use very conservative values for pruning. Main beam widths (chosen to minimise the number of search errors at a reasonable computational load) were typically between 275 and 350. The word end beam width (150) and maximum model limit (100k for five thousand word recognition or 200k for larger tasks) were set to levels shown not to effect accuracy.

Beam Pruning

The choice of beam width is the major factor controlling the computational load and number of search errors made during decoding.

Wide beam widths lead to many paths being considered and a high computational load. Many of these paths will be both locally and globally unlikely and so beam pruning the locally unlikely paths can reduce the amount of computation required during decoding. However some of the paths which are locally unlikely may form part of the globally most likely path and

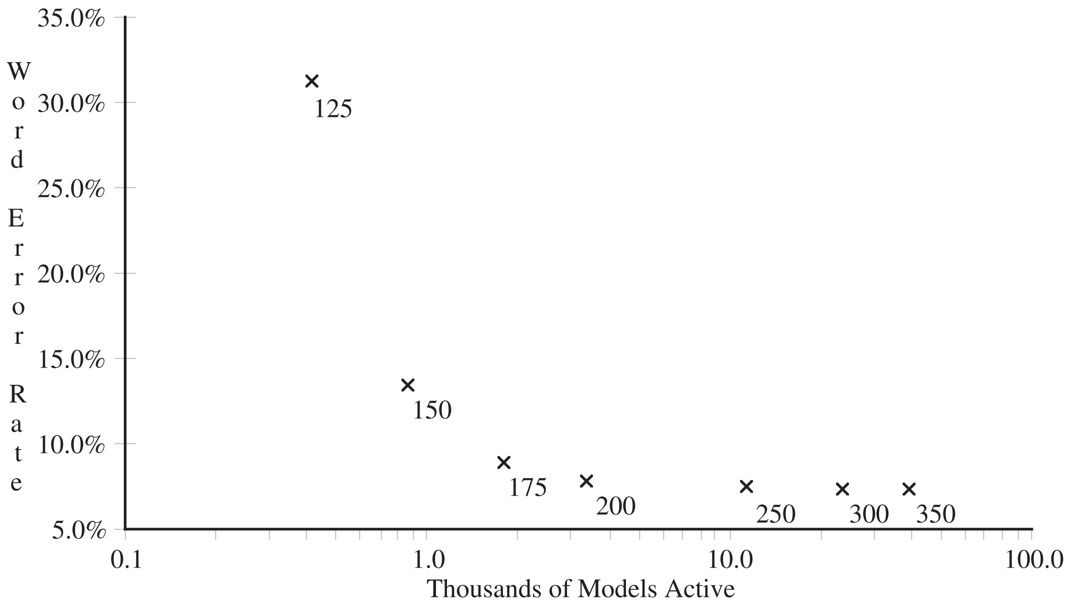

Figure 6.5: Variation in active models and word error rate with overall beam width.

pruning them will result in search errors. The choice of beam width depends on the relative importance of decoding speed and accuracy. For system evaluation accurate estimation of the word error rate of a system is required and beam widths are chosen to minimise the number of search errors (whilst maintaining reasonable decoding times).

Figure 6.5 shows how the number of active models per frame (and thus the computation required for decoding) and the percentage word error rate vary for particular beam widths. (Both maximum model and word end pruning were used for these experiments, but at thresholds which did not lead to search errors.)

As this shows, for beam widths greater than 250 the performance of the system is not affected by the choice of beam width and this indicates that there are no search errors being made. At a threshold of 250, search errors account for only 2% of the total error rate. Experiments were typically performed with a beam pruning threshold of between 275 and 350 in order to minimise the search error rate. Using a beam width of 250 compared to 300 halves the number of active models, almost doubling decoding speeds, and using 200 results in a factor of seven reduction in the number of models and only increases the error rate by 7%. Reducing the beam width still further dramatically increases the error rate and although it is possible to reach real-time performance (excluding output likelihood calculations) by just decreasing the beam-width the error rate of over 30% is dominated by search errors.

Maximum Model Pruning

Although the total running time of an experiment is proportional to the average number of models active each frame, the size of the process is dependent upon the peak number. Figure 5.7 showed how the number of active models varied widely with the maximum almost two orders of magnitude higher than the average. If no attempt is made to constrain the maximum number of models, the size of the decoding process can easily exceed the amount of physical memory present and as a result the speed of decoding falls dramatically.

In order to constrain the process size, additional pruning was introduced to limit the number of active models. This is used to ensure that the decoder will always operate from physical memory (to ensure consistent performance) and speeds up decoding during periods of greatest uncertainty (as there is an upper bound on the computation required for each frame).

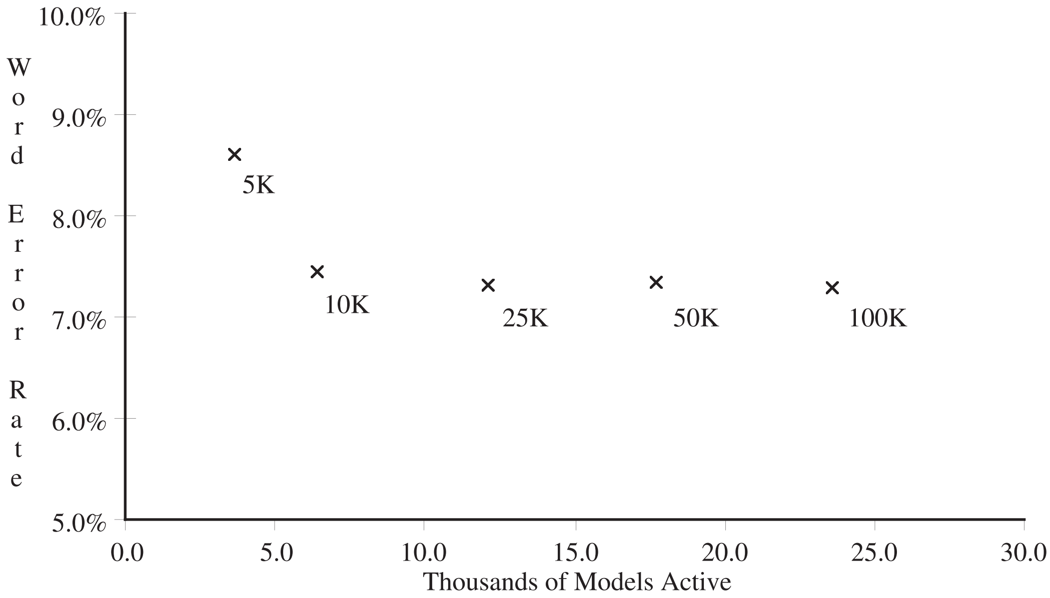

Figure 6.6: Variation in active models and word error rate with maximum model limit.

Figure 6.6 shows the variation in the average number of models active and the word error rate with the threshold chosen for "maximum model pruning". Beam pruning (with a threshold of 300) and word end pruning (with a threshold of 150) were also used although the thresholds were chosen to ensure they did not introduce search errors.

At relatively high maximum model limits (more than a few times the average number of active models) only a small number of frames are effected and the error rate is unaltered. As the maximum model beam is reduced further more frames are subject to this form of pruning and the average number of active models reduces substantially. Even so this has little effect on the error rate and reducing the maximum number of active models by a factor of four to

25k only introduces 2% relative (0.15% absolute) search errors. However further reductions in the limit on the number of models active do increase the error rate and better results (at a particular computational load) are produced by lowering the main beam width.

Word End Pruning

As described in section 5.3.2 the range of language model likelihoods is much smaller over different histories than over different words. This, coupled with the greater degree of certainty of word identity at words end, leads to the use of a different beam width for word end tokens.

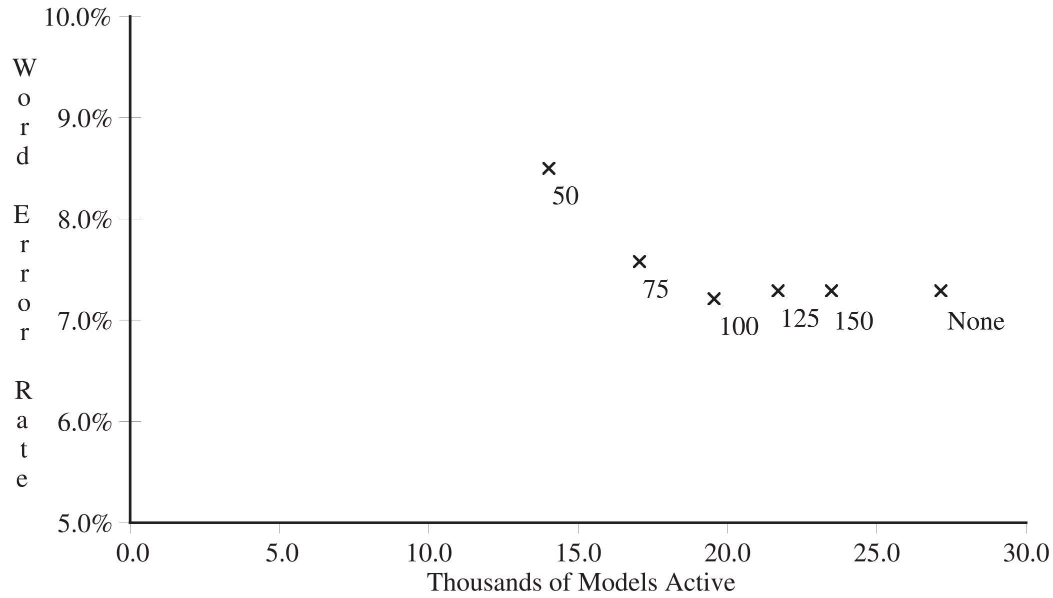

Figure 6.7: Variation in active models and word error rate with word end beam width.

Figure 6.7 shows how the word error rate and average number of active models is affected by a separate word end beam width. Maximum model pruning (with a limit of 100k models) and beam pruning (with a width of 300) were enabled at levels which did not cause search errors.

Word end pruning had less effect on the overall computational load than either beam or maximum model pruning but it was able to reduce the number of active models by up to 30% without introducing any additional search errors. Search errors only occurred when the word end beam was reduced to 1/3 of the width (in log likelihood) of the main beam. The actual beam width, in this case 75, is equivalent to between three and four standard deviations of the probability of a word over all distinct histories (i.e. the standard deviation given in section 5.3.2, 1.32, multiplied by the language model scaling factor, 16.0)

Combined Pruning

The best results (in terms of fewest search errors for a fixed amount of computation or least computation for a particular accuracy) come from combining all three forms of pruning.

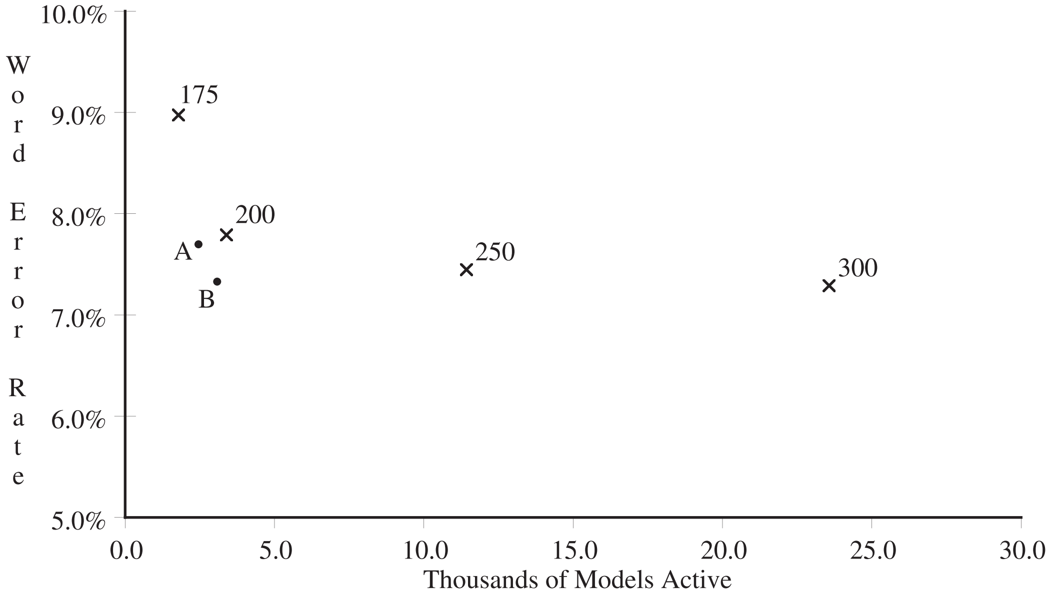

Figure 6.8: Variation in active models and word error rate with combined pruning.

Figure 6.8 shows how all three forms of pruning can be combined to provide better performance than just varying the main beam width. The points labelled A and B have beam pruning enabled at 200 (for A), 220 (for B), word end pruning at 100 and maximum model pruning at 10k. The remaining points are a subset of those in figure 6.5 showing how accuracy and complexity vary with the main beam. In the first case (with around 3000 active models per frame) only 1% of the 7.34% error rate is due to search errors and in the second case (with around 2200 active models per frame) the error rate only rises to 7.71% (5% relative search errors). Since there are relatively few models active each frame and the output likelihood distributions are complex (6400 distributions each with 12 components) the proportion of the computation consumed by the output likelihood calculations rises to 60%.

These figures show that this simple unoptimised decoder cannot reach real-time performance for an unconstrained large vocabulary task. However it comes close enough to suggest that with optimised code, accelerated output likelihood calculations and a small amount of lookahead real-time performance should be possible on a 100-200 M-FLOPS machine. A demonstration system using an unmodified decoder with simplified acoustic models requires a few times real-time for 5k recognition and between five and ten times real-time for 20k on an HP735/125 capable of around 100 M-FLOPS.

Training	Type	LM	Eval'92	Dev'93.s6	Dev'93.odd	Eval'93	(Models)
SI84	wint/gi	bg	8.11%	10.39%	12.40%	12.53%	(9.6K)
SI84	xwrd/gi	bg	6.86%	9.52%	10.48%	8.51%	(21.7K)
SI84	xwrd/gd	bg	6.58%	9.13%	9.67%	8.67%‡	(21.9K)
SI284	xwrd/gd	bg	5.14%	6.63%	7.58%	6.77%	(23.4K)
SI284	xwrd/gd	tg	3.19%	5.27%	6.09%	4.90%†	(19.8K)

Table 6.4: Word error rates for 5k systems (H2) used in the Nov'93 evaluation.

6.4.2 November 1993 Evaluation

Systems using these techniques were submitted to the ARPA Continuous Speech Recognition Evaluation in November 1993. The evaluation took the form of a once-through test in which results had to be generated on the unseen test data in one run. The systems could not be optimised for the new data. The various tests and conditions are further explained in section A.3.1.

- H1. Open vocabulary (64K quality filtered) recognition.
  - H1-C1. Recognition with 20k vocabulary and trigram language model with *SI37* or *SI284* training.
  - H1-C2. Recognition with 20k vocabulary and bigram language model with *SI37* or *SI284* training.
- H2. 5k closed vocabulary recognition.
  - H2-P0. Recognition with any language model and any training
  - H2-C1. Recognition with bigram language model and *SI12* or *SI84* training

Three sets of acoustic models were produced for the November 1993 evaluation, two with gender dependent variants. All were produced using gender independent clustering with only one Gaussian per cluster.

The simplest, a set of word-internal context dependent triphones, was produced only in gender independent form. The models were trained on the *SI84* section of the training data and consisted of 8453 distinct triphone models. These were constructed from 3701 tied state distributions, each an eight component mixture Gaussian giving a total of around 2.3 million parameters. The techniques used to construct and use this system were broadly similar to the HTK system submitted in the Sept'92 RM evaluation ([91]) and provided a baseline against which further improvements could be measured. However this system was restricted to recognition with a vocabulary of 5k and a bigram language model because of limitations in the conventional static network decoder. The differing test and training vocabularies (and as a result word-internal triphones occurring in the test dictionary which did not appear in the training data) meant that it was not possible to use the same bottom-up agglomerative clustering procedure. Instead the decision tree based method described in chapter 3 was used for its ability to construct models for unseen contexts.

A system of cross-word context dependent triphones trained on the *SI84* section of the database was also produced for the H2-C1 test. Both gender-dependent and gender-independent versions were produced (by cloning the gender independent system and re-estimating gender dependent mixture component means and weights but using the gender independent variances). The gender independent system contained 3820 tied state eight component mixture Gaussians distributions (for a total of 2.4 million parameters) chosen using the decision tree clustering method. The gender dependent version had around 3.6 million parameters. The tree structured dynamic network decoder was used for recognition and so both bigram and trigram language models could be used.

Finally a cross-word system trained on the *SI284* database was produced for the H2-P0, H1-C1 and H1-C2 tests. Again this was produced in both gender independent and dependent versions with the gender independent version constructed from 7558 ten component mixture Gaussian distributions giving a total of 6.0 million parameters. The dynamic net decoder allowed recognition to be performed with both 5k and 20k vocabularies using bigram or trigram language models in a single pass.

Table 6.4 shows the error rates of these systems for 5k (actually 4986) word recognition. The accuracy of the word internal system is substantially lower than for the Resource Management task, an average word error rate of 10.9% compared to 3.55%. This is due to the larger vocabulary and greater connectivity of the grammar giving more confusability between words and a higher task perplexity (106 rather than 60).

Using models which are cross word context dependent reduced the word error rate by an average of 13% on the 5k development data and by over 30% on the evaluation test data. This contrast may be due to the greater fluency and faster speaking rate of the evaluation data. After the evaluation NIST highlighted the effect of speaking rate on the error rate and how the presence of outlier speakers can have a disproportionate effect on the average error rate [65].

Using separate gender dependent models in parallel and choosing the most likely hypothesis from either set of models gave a 5% reduction in error rate on the development data but showed a 2% rise in the error rate on the evaluation data. This difference may be due to the gender independent cluster not enforcing a minimum quantity of training data per distribution for each gender separately. This can lead to data insufficiency in the gender dependent system. Overall the gender dependent cross word context dependent system was 18% more accurate than the gender independent word internal system on the development data and 32% on the evaluation. This system (marked with a ‡ in table 6.4) was submitted for the H2-C1 test in the Nov'93 evaluation.

Increasing the amount and variety of the training data by using the speaker independent portion of both the WSJ0 and WSJ1 sections of the database allowed a larger system to be constructed. This system had two and a half times the number of parameters of the *SI84* trained system. Despite the large amount of data (over 55 hours of speech) used to estimate the model parameters the time taken to perform the decision tree clustering was only a small fraction of the total time required to train the system. The improvement in modelling detail

Speaker	Number of Sentences	Words Correct	Substitution Errors	Deletion Errors	Insertion Errors	Word Errors	Sentence Errors
4OA	22	99.2%	0.52%	0.26%	0.78%	1.56%	18.2%
4OB	21	97.5%	1.48%	0.99%	0.25%	2.72%	33.3%
4OC	23	95.2%	4.51%	0.25%	1.25%	6.02%	60.9%
4OD	22	98.8%	1.20%	0.00%	0.48%	1.67%	22.7%
4OE	23	96.0%	3.67%	0.31%	2.14%	6.12%	47.8%
4OF	20	89.1%	8.24%	2.65%	0.59%	11.47%	60.0%
4OG	20	97.0%	2.99%	0.00%	0.60%	3.58%	40.0%
4OH	21	96.0%	2.25%	1.75%	0.25%	4.25%	52.4%
4OI	22	96.4%	3.12%	0.45%	1.34%	4.91%	50.0%
4OJ	21	93.0%	4.02%	3.02%	0.75%	7.79%	76.2%
Overall	215	95.93%	3.11%	0.96%	0.83%	4.90%	46.0%
Mean	21	95.8%	3.20%	0.97%	0.84%	5.01%	46.2%
Deviation	1	3.0%	2.19%	1.12%	0.58%	3.03%	18.0%

Table 6.5: Speaker by speaker results for H1-P0 system used in the Nov'93 evaluation.

and accuracy led to a consistent reduction in error rate of around 23% for both development and evaluation data.

The final line of the table shows the effect of the use of longer span language models. The novel decoder architecture employed allows the trigram language model to be integrated directly into a single pass search in place of the bigram language model. This reduced the task perplexity from around 106.1 to 61.5 and the word error rate by an average of 25% on both the development and evaluation test data.

Differences in performance between the test sets are probably due to wide variation in performance over speakers. Table 6.5 shows speaker by speaker performance of the system used in the H1-P0 test (highlighted with a † in table 6.4).

The final column of table 6.4 also shows the average number of models active for each frame during decoding. This provides an approximate measure of the computational cost of recognition for each system. Increases in system complexity improve the accuracy without huge increases in the computational load. The only substantial increase occurs with the introduction of cross word context dependent models and this is much smaller than would occur with a static linear network decoder. Use of the trigram language model actually reduces the average number of active models per frame despite a substantial increase in the size of the search space. This disparity is due to the increased accuracy of the trigram language model allowing the search to be more localised.

Table 6.6 shows results for tests on the open (64K quality filtered) vocabulary. Error rates for 20k tests are higher because of additional confusable words as well as out-of-vocabulary words. On average each out-of-vocabulary word produced 1.6 word errors and since the OOV rate was 1.8% these errors represent over 20% of the final error rate. Use of the trigram

Training	Type	LM	Eval'92	Dev'93.odd	Eval'93	(Models)
SI284	xwrd/gd	bg	11.08%	16.17%	14.45%	(30.7K)
SI284	xwrd/gd	tg	9.46%	13.71%	12.67%	(29.3K)

Table 6.6: Word error rates for 20k systems (H1) used for the Nov'93 evaluation.

produces less improvement at 20k (14%) than at 5k (25%). However the decrease in perplexity is much smaller at 142.7 for the trigram verses 222.9 for the bigram. Both of these systems were submitted to the Nov'93 CSR Evaluation, the bigram system in the H1-C2 test and the trigram one in the H1-C1 test.

The performance of the decoder at 20k is also shown in the table. Despite the factor of four increase in the size of the vocabulary the number of active models has only increased by 50% compared to 5k tests. The small size of this increase is due to the tree structuring which ensures that since most 'new' words will share initial phones with words already appearing in the vocabulary they will contribute little to the computation.

Speaker	Number of Sentences	Words Correct	Substitute Errors	Delete Errors	Insert Errors	Word Errors	Sentence Errors
4OA	20	96.2%	3.53%	0.29%	2.35%	6.18%	55.0%
4OB	21	92.4%	6.96%	0.95%	1.27%	9.18%	57.1%
4OC	22	93.7%	6.02%	0.29%	1.72%	8.02%	50.0%
4OD	22	93.1%	6.61%	0.55%	3.03%	9.92%	68.2%
4OE	22	93.0%	6.76%	0.24%	2.17%	9.18%	63.6%
4OF	24	78.8%	18.41%	2.81%	2.56%	23.79%	83.3%
4OG	20	93.4%	5.51%	1.10%	1.10%	7.72%	55.0%
4OH	21	85.6%	13.22%	1.15%	2.30%	16.67%	76.2%
4OI	20	93.7%	6.31%	0.00%	0.66%	6.98%	50.0%
4OJ	21	77.2%	19.44%	3.66%	2.82%	25.92%	90.5%
Overall	213	89.4%	9.42%	1.16%	2.09%	12.67%	65.3%
Mean	21	89.7%	9.19%	1.13%	2.03%	12.35%	64.9%
Deviation	1	6.7%	5.56%	1.27%	0.82%	7.20%	14.3%

Table 6.7: Speaker by speaker results for 20k trigram (H1-C1) system used in the Nov'93 evaluation.

In the November 1993 ARPA CSR Evaluation, five systems were officially entered and submitted to NIST for scoring. The word internal system was submitted to provide a baseline to show the effect of the improvements in modelling accuracy made by CU-HTK since the Sept'92 Resource Management Evaluation. In three (H1-C2, H2-P0 and H2-C1) of the four tests for which systems were entered the CU-HTK systems produced the lowest word error rates with better performance than systems from BBN [98], BU [62], CMU [33], Cambridge University Connectionist Group [78], Dragon [83], ICSI [51], MIT Lincoln Labs [66], Philips [2] and SRI [17]. In the remaining test (H1-C1) only a system from LIMSI [23] produced a lower word error rate.

6.4.3 November 1994 Evaluation

For the ARPA Continuous Speech Recognition Evaluation the following year, the decoder had been extended to generate N-Best lattices and to use these as a constraining grammar for faster recognition. Gender independent cross word triphone systems together with bigram language models were used to generate lattices for the various test sets. These could be used to constrain recognition and allow additional (more complex systems) to be evaluated efficiently.

The LIMSI dictionary was made available in 1994 and experiments showed that use of this dictionary led to a 4% reduction in word error rate compared to systems which used the Dragon dictionary. Systems produced after the November 1993 evaluation were based on the LIMSI dictionary.

Vocabulary	Test Set	OOV Rate	Bigram	Trigram	Fourgram
20k'94	Dev'94	2.68%	201.2	128.8	
20k'94	Eval'94$^+$	2.38%	208.6	131.2	
65k	Dev'94	0.31%	240.9	145.3	133.2
65k	Eval'94$^+$	0.65%	232.3	143.9	130.5

Table 6.8: Test set perplexities and OOV rates for various Nov'94 language models.

The Nov'93 evaluation had shown that the presence of out-of-vocabulary words in the test set produced a substantial proportion of the errors. For the H1-P0 test in the Nov'94 evaluation any language model and vocabulary could be used. Preliminary experiments indicated that the addition of words to the vocabulary rarely resulted in additional errors because these additional words tended to be relatively low probability poly-syllables. Such words are rarely confused with other more common ones already in the vocabulary, although they themselves may not be recognised correctly. Consequently a vocabulary size of 65k was chosen since exceeding a limit of 2^{16} words would have dramatically increased the amount of storage required by the language model.

System	Eval'92	Dev'93.odd	Eval'93
Eval'93 *SI284* GD	9.46%	13.71%	12.67%
HMM-1 *SI284* GD	9.45%	12.90%	12.04%
HMM-2 *SI284* GD	8.19%	12.34%	11.61%

Table 6.9: Comparison at 20k of the Nov'93 evaluation system and the Nov'94 systems.

The 65k vocabulary was chosen by selecting the most common 65464 real words (not spelling mistakes or other errors) which occurred in a corpus of North American Business news. This consisted of the 237 million words supplied by the Linguistic Data Consortium for generating language models, supplemented by the more recent language model development test data. This development test data was added several times to ensure that any words which occurred more than once in the development test data would be included in the final list. This larger vocabulary was chosen in this way in an attempt to minimise the OOV rate on the unseen test data which had been drawn from articles occurring immediately after the development test epoch. Table 6.8 gives the OOV rates and test set perplexities for the language models used in the Nov'94 evaluation. Note these values are given for the preliminary reference answers (indicated as Eval'94$^+$) which only contain a single transcript for each sentence. One-best word error rates are given for the final post-adjudication reference answers which contained multiple alternatives complicating lattice accuracy and perplexity calculations. For the final reference answers the OOV rates dropped to 1.87% at 20k and 0.42% at 65k.

This data was also used to generate bigram, trigram and fourgram language models for the 65k vocabulary. In this case the language modelling development test data was added sufficient times to ensure that all bigrams which appeared were included in the final language model but not all trigrams or fourgrams.

The first pass of the two pass decoding strategy used a cross word triphone system (similar to that used in the Nov'93 evaluation) with a bigram language model to generate lattices of word hypotheses. This system (94'HMM-1) consisted of 6399 shared 12 component mixture Gaussian state distributions giving a total of 6 million parameters. Comparative results for this system and the system used in the 1993 evaluation are shown in table 6.9.

Test	Lattice Density	Sentence Error Rate	OOV rate	Word Error Rate
Dev'94 H1-C1	265	12.3%	2.68%	0.80-4.05%
Eval'94$^+$ H1-C1	327	8.9%	2.38%	0.50-3.44%

Table 6.10: Lattice quality for unlimited vocabulary test data using a 20k recognition vocabulary, cross word triphones and a bigram language model.

Lattice error rates for the the Nov'94 development and evaluation test data using the standard 20k'94 vocabulary and bigram language model are shown in table 6.10. The word error rate is range dependent upon the treatment of OOV words (as described in section 5.5.2). However the sentence error rate is only calculated for sentences which do not contain any out of vocabulary words. The lattice density gives the number of hypotheses appearing in the complete set of lattices divided by the number of words actually spoken. This gives an indication of the size of the lattices although duplication of words due to differing phonetic and language model contexts means that the number of different words hypothesised for each spoken word is more than an order of magnitude smaller.

Corresponding figures for the 65k vocabulary system are shown in table 6.11. A comparison

Test	Lattice Density	Sentence Error Rate	OOV rate	Word Error Rate
Dev'94 H1-P0	289	16.2%	0.31%	1.15-1.53%
Eval'94^{+} H1-P0	341	10.7%	0.65%	0.60-1.50%

Table 6.11: Lattice quality for unlimited vocabulary test data using a 65k recognition vocabulary, cross word triphones and a bigram language model.

of these results shows that the presence of out of vocabulary words in the test is a significant source of lattice errors. However, unlike the one best error rates, which show an average of 1.6 errors for each out of vocabulary word, lattices contain closer to one error for each out of vocabulary word.

The use of these lattices allowed a second recognition pass with models having a greater degree of context dependency. This system used the same method of state tying as the triphone system, with decision trees used to synthesise models for all possible contexts. However the questions used to split each node during construction were extended in two ways.

- Longer distance.

  The questions were not limited to just the immediately preceding and following phones ($+/-1$ context) but also the phones adjacent to them. This $+/-2$ context effectively allows 'quinphone' rather than 'triphone' context to be represented where this has a significant effect upon the acoustic realisation of a phone.

- Word boundaries.

  Questions concerning the position of the boundary of the current word were also added. These allow different representations for phones which include a word boundary in their surrounding context.

These extended context questions ($+/-2^{*}$ context) were used to construct a system referred to as 94'HMM2. Table 3.1 showed the very large number of distinct contexts that occur at this level and so the unified Baum-Welch method for collecting statistics about the different contexts was not suitable. Instead a cross word triphone system (similar but not identical to the 94'HMM1 system) was used to perform a state level Viterbi alignment. This deterministic alignment was used to generate statistics about all the contexts for each base phone in turn. The decision trees were then generated in the gender robust fashion described in section 3.8.3. Separate means and variances were used for male and female data and the likelihoods used to select questions were summed over both two distributions. This prevented the use of virtual gender questions and allowed checks for both genders on the occupancy of the tied distributions. The resultant model set estimated from the *SI284* data consisted of 9358 state distributions each consisting of a 14 component mixture Gaussian distribution. The total number of parameters (10.5 million for the gender independent version) was significantly higher than that used for similar triphone based systems. However the triphone systems showed no improvement in accuracy as their number of parameters was increased further. This difference is probably due to the increased number of contexts allowing commonly occurring triphones (which would previously have been represented by a single model) to be represented more consistently and in greater detail by several models.

Comparative results for this system, the triphone system 94'HMM-1 and the system used in the 1993 evaluation are shown in table 6.9. On these test sets the word error rates of the 94'HMM-2 system are approximately 6% lower than 94'HMM-1 and 11% lower than the equivalent (*SI284* trained gender dependent) Nov'93 evaluation system.

System	Dev'94	Eval'94
94'HMM-1 GI	12.51%	11.67%
94'HMM-1 GD	12.01%	11.30%
94'HMM-2 GI	11.76%	10.58%
94'HMM-2 GD	11.52%	10.49%†

Table 6.12: Word error rates for 20k trigram (H1-C1) systems used for the Nov'94 evaluation.

For the 20k (H1-C1) test each sentence had to be processed independently. Three systems were run in parallel, gender independent, male and female and the most likely answer chosen. Because this decision was made independently for each sentence this meant that occasionally the wrong gender choice could be made. Often in these cases the gender independent models would be used although for a single sentence the incorrect gender dependent models were used. It was noted that these incorrect choices of gender normally occurred with very short sentences and did not effect the overall error rate. Error rates for the Nov'94 development and evaluation test data are shown in table 6.12 for the 94'HMM-1 and 94'HMM-2 systems in both gender dependent and gender independent form. The additional context modelled by 94'HMM-2 provides an average of 6% improvement over the triphone system and the use of gender dependent models an average improvement of only 1.5%.

The 94'HMM-2 GD result (10.5%) was the lowest reported error rate for the H1-C1 Nov'94 test. A. T. & T. [49], Bolt Beranek & Newman [54], Boston University [63], Carnegie Mellon University [13], Centre de Recherche Informatique de Montreal [56], Cambridge University Connectionist Group [28], Dragon [81], I. B. M. [5], Karlsruhe University [79], CNRS-LIMSI [24], M. I. T. Lincoln Laboratory [68], New York University [85], Philips [20] and S. R. I. International [18] participated in this evaluation and submitted systems.

The H1-P0 test allows knowledge of the session (speaker) boundaries to be used. This makes it possible to apply a consistent choice of gender onwards through a session as well as to adapt the acoustic and language models to the particular speaker and topic.

The CU-HTK system made use of incremental adaptation of the acoustic models but made no attempt to adapt the language model. To avoid spurious changes of gender during a session and reduce the total amount of computation required for the evaluation the choice of speaker gender was based only on the first two sentences. All three model sets (male, female and independent) were used to decode the first two sentences (for which adaptation was not used)

System	Language Model	Dev'94	Eval'94
94'HMM-1 GI	Trigram	9.52	9.18
94'HMM-1 GD	Trigram	9.17	8.57
94'HMM-2 GI	Trigram	9.14	8.41
94'HMM-2 GD	Trigram	8.68	8.22
94'HMM-2 GD	Fourgram	8.26	7.93
94'HMM-2 GD Adapted	Fourgram	7.24	7.18†

Table 6.13: Word error rates for 65k word (H1-P0) systems used for the Nov'94 evaluation.

and the system used for the remainder of the session (for which adaptation was used) determined by their relative likelihoods. When a single gender dependent system was most likely in both cases, this system was used for the remainder of the session. However if different choices were made or the gender independent system was more likely for either of the first two sentences the gender independent system was used for the rest of the utterances. This did not occur with either the development or the evaluation test data and only the gender dependent models were used.

The scheme used for adaptation, mentioned in section 3.2 and described in detail in [46] was incorporated directly into the decoder. This enabled transparent adaptation throughout a session with the recogniser selecting the most likely hypothesis for each sentence using the current set of models. This hypothesis was passed to the adaptation library which performed the necessary Baum-Welch frame state alignment to collect statistics about the speaker characteristics. After two files had been recognised the model parameters were updated and these speaker adapted models used to decode the next sentence. This updating procedure was repeated every other sentence throughout the session. (The updating was not performed every sentence as it was computationally expensive since all mixture means need to be adjusted and it did not provide significantly better performance).

Table 6.13 gives results at 65k for both the development and evaluation data. Again the use of wider context models and gender dependency gives small improvements in accuracy of about 10% but for the 94'H1-P0 test further improvements were made by using a fourgram language model (4%) and incremental unsupervised speaker adaptation (11%). The adaptation was particularly effective for speakers with relatively poor performance. Speaker by speaker results for the H1-P0 system are shown in table 6.14. This system (marked with the † in table 6.13) gave the lowest error rate of any of the systems which took part in the evaluation.

6.5 Summary

This chapter has presented results which show that both the decoding and acoustic modelling techniques developed in this thesis are capable of state of the art performance. The accuracy of the decision tree clustering technique was shown to be comparable to bottom-up techniques

Speaker	Number of Sentences	Words Correct	Substitute Errors	Delete Errors	Insert Errors	Word Errors	Sentence Errors
4t0	15	92.6%	6.11%	1.31%	1.31%	8.73%	73.3%
4t1	21	95.4%	4.23%	0.37%	0.55%	5.15%	57.1%
4t2	15	96.3%	3.71%	0.00%	1.98%	5.69%	80.0%
4t3	16	98.0%	2.04%	0.00%	0.26%	2.30%	43.8%
4t4	16	94.0%	5.71%	0.30%	0.90%	6.91%	68.8%
4t5	15	96.6%	2.42%	0.97%	0.73%	4.12%	60.0%
4t6	15	84.0%	13.12%	2.89%	2.89%	18.90%	80.0%
4t7	15	97.3%	2.53%	0.21%	0.84%	3.59%	46.7%
4t8	16	92.5%	5.64%	1.88%	1.25%	8.78%	62.5%
4t9	17	96.5%	2.70%	0.81%	0.81%	4.31%	47.1%
4ta	15	93.9%	5.01%	1.06%	0.26%	6.33%	60.0%
4tb	15	90.7%	8.10%	1.25%	2.49%	11.84%	86.7%
4tc	15	95.0%	4.31%	0.72%	0.24%	5.26%	60.0%
4td	17	80.8%	13.65%	5.58%	1.54%	20.77%	82.4%
4te	16	96.3%	3.50%	0.23%	1.86%	5.59%	43.8%
4tg	15	95.3%	4.45%	0.30%	2.67%	7.42%	53.3%
4th	16	98.0%	1.57%	0.39%	0.79%	2.76%	43.8%
4ti	15	95.4%	4.64%	0.00%	1.55%	6.18%	60.0%
4tj	15	96.2%	3.51%	0.29%	0.00%	3.80%	60.0%
4tk	16	96.4%	3.08%	0.51%	1.28%	4.87%	50.0%
Overall	316	94.0%	5.01%	0.99%	1.18%	7.18%	60.8%
Mean	15	94.0%	5.00%	0.95%	1.21%	7.16%	61.0%
Deviation	1	4.4%	3.26%	1.30%	0.84%	4.89%	13.7%

Table 6.14: Speaker by speaker results for the 65k fourgram speaker adaptive (H1-P0) system used for the Nov'94 evaluation.

with the added advantage of being able to synthesise accurate models for unseen contexts. This allows the use of cross word triphone (or longer distance context dependent) models for substantially increased accuracy. A decoder architecture suitable for these types of models and very large vocabularies has been investigated. The use of maximum model and word end pruning techniques in addition to beam pruning has improved the efficiency of the recognition and reduced the required computational resources. Further development of the decoder has allowed lattice generation and rescoring which increase the efficiency of system evaluation and optimisation by an order of magnitude and enable the use of more complex acoustic and language models. Although the decoder is incapable of real time operation, optimisation of the code, faster output likelihood calculation and the addition of some form of lookahead should allow real time operation.

Chapter 7

Conclusions

This dissertation has addressed a number of key issues involved in the design of a large vocabulary speech recogniser based on hidden Markov models. In particular, it has investigated the problems posed by the use of context dependent models in such recognisers. A method of building cross word context dependent models has been developed in conjunction with a decoder capable of using these models for large vocabulary recognition. The techniques developed have been applied to a variety of large vocabulary recognition tasks and the performance analysed both in terms of the computational complexity and the recognition accuracy.

7.1 Review of the Work

Speech is inherently highly variable, however, a significant proportion of this variability is due to consistent contextual effects. This thesis has shown that enhanced recognition performance is obtained when the models account for these consistent differences.

In order to exploit context dependent models in a large vocabulary recogniser, two problems must be solved. Firstly, a method must be found to train the models which takes due account of the sparseness of the training data and the fact that many of the models required for recognition will not occur at all. Secondly, a decoder must be designed which can incorporate these context dependencies, not just within words but across word boundaries. The latter is particularly important when dealing with fluent speech when the co-articulation effects are substantial.

To solve the first problem, decision trees are used to determine significant and consistent differences across contexts. These decision trees are then used to cluster the states of hidden Markov models. Sharing parameters in this way increases the trainability and robustness of the resulting recogniser and allows the models to capture fine contextual differences. The decision trees are generated using statistics from the training data in conjunction with linguistic knowledge supplied in the form of questions. The linguistic information is used to supplement the training data when there are few (or no) examples of a particular context. The structure of the trees is chosen to maximise the likelihood of the training data and the resulting algorithm allows the construction of accurate recognition systems in a computationally efficient manner. The ability of this procedure to generate accurate models for unseen contexts allows the effects of cross word context dependency to be captured. In a typical database only a small proportion of the possible contexts occur. As a result, alternative methods of system construction, which do not have the ability to produce models for unseen contexts, are less accurate.

Although the acoustic models are more accurate when the effects of cross word context dependency are taken into account, their use complicates the decoding process. Conventional decoding approaches make use of a static network in which each word is individually represented. Such approaches are unsuited to the use of cross word context dependent models, long span language models or very large vocabularies, yet all of these features are necessary to produce accurate speech recognisers. Many decoding schemes now make use of multiple passes in which computationally simple but relatively inaccurate models are used initially to constrain the search space of later passes using more accurate models. Such schemes can be complicated to implement because of the need to carefully integrate the different systems used in the multiple passes.

Hence, the solution proposed here for the decoding problem is to adopt a simple one-pass strategy. A novel decoder design, that uses accurate acoustic and language models from the outset, has therefore been proposed. This decoder uses dynamic network construction and model sharing, through tree structuring of the network, to reduce the computational load. Early application of the most accurate acoustic and language models allows the search space to be heavily constrained, without introducing search errors. The architecture employed by this decoder also allows the generation of multiple hypotheses in the form of a lattice with little computational overhead. These lattices can then be used to speed system development and allow the evaluation of more detailed but computationally expensive models.

Experiments were initially performed on the medium vocabulary Resource Management task to evaluate the accuracy of the decision tree based modelling approach. These experiments showed that the approach was as accurate as other state clustering techniques, whilst allowing generation of models for unseen contexts. This ability enabled the construction of cross word triphone models and their use substantially increased recognition accuracy. The new decoder architecture allowed the use of such models at the expense of a small increase in computational complexity. As the size of the recognition vocabulary and complexity of the recognition task are increased, the efficiency of the decoder and the discriminative ability of the models becomes more important. Improved modelling accuracy was possible with the larger Wall Street Journal database by extending the amount of context considered by the decision tree. Efficient decoding using quinphone models was possible though the use of lattices generated using triphone models.

Together the new decoder design and the decision tree synthesised models were used to produce a recognition system with state of the art performance.

7.2 Suggestions for Further Work

All the work in this thesis has used speech recorded in quiet laboratory conditions. Performance of the recognition systems degrades substantially when the source of the speech is not highly controlled. In particular, the presence of noise and distortion, which barely effect human recognition performance dramatically reduce the accuracy of typical automatic speech recognisers. To be useful in practical applications, speech recognisers must be more robust to background noise and distortion.

Similarly all the speech was read from prepared texts. Recognition performance drops substantially for spontaneous or conversational speech. This is due to the fluent nature of such speech not matching either the acoustic or the sentence based language models. In particular accounting for restarts and hesitations will probably need some form of dialogue modelling and require that the speech recogniser interact with the user when it needs to obtain clarification.

Whilst the recognition system described in this thesis gives high accuracy there is room for further refinement of the acoustic models. Improvements in modelling accuracy may require that even more contextual features be taken into account. For instance, none of the systems used for this work have taken account of variation in stress. Both stress and prosody form important cues for human speech recognition. It may be possible to investigate such features in the decision tree framework by extending the type of questions that are used during construction. When consistent differences are due to such effects, the additional questions will be used during tree construction and the resulting models will be able to capture these differences. Accuracy may also be increased if the trees could be constructed in a way that maximises the ability of the models to distinguish between different words. Such discriminative training schemes tend to be computationally expensive but have demonstrated that they can improve the ability of the models to make fine distinctions. This ability becomes more important as vocabulary size and task complexity increase.

Perhaps more importantly, the decoder can be improved still further, particularly with regard to its computational efficiency.

Two of ways in which the efficiency of the decoder could be increased have been mentioned in the text. The calculation of the output likelihoods represents a significant proportion of the total computation, especially at the tighter beam widths necessary for fast recognition. No attempt was made to decrease the computational cost of these calculations, beyond caching the likelihoods for each tied-state distribution. These computations could be pruned in a similar manner to that used during the beam search. Each frame, many of the mixture components will be relatively unlikely and their exact likelihoods are not important. If a computationally efficient way to approximate these likelihoods could be found, this could be used to determine the most likely components. Calculation of the exact output likelihoods would then be restricted to this set and decoding efficiency improved. Vector quantisation provides an efficient way to divide the input space into a finite number of regions. Exact output likelihood calculations could be restricted to the components that fall within the most likely region of the input space.

Lookahead has also been suggested as a way of improving decoding efficiency. A substantial proportion of the computation during decoding with the tree structured network is involved in the final layer of the network. Restricting the size of this layer by not considering in detail the unlikely models could substantially decrease the computation. If a fast but approximate estimate of the relative likelihood of each model over the duration of the next phone could be found, then these values could be used as lookahead to constrain network growth.

Improvements in decoding efficiency may also be possible through further refinement of the network architecture. In the current decoder little use is made of the back-off nature of the language models. Path merging only occurs at the end of a word when the language model probabilities of all following words are the same. For instance, when a trigram language model is used, paths only merge if they have a common two word history (unless only bigram probabilities are used following a particular word pair). Normally explicit trigram probabilities will exist for only a small proportion of possible following words. If these words were treated separately, the remaining backed-off paths (which only use bigram probabilities) could be merged with paths with the same phonetic context ending in the same word (rather than the same last two words, the criteria normally required to merge paths using a trigram). This is similar to the way in which a linear network can be divided into back-off and bigram portions described in section 4.2.5. This additional path merging should reduce the number of model instances by allowing more sharing of computation.

7.3 Conclusions

The use of context dependent models to capture consistent contextual variation has been investigated. A method of constructing a context dependent hidden Markov model based continuous speech recogniser for unlimited vocabularies has been developed. This recogniser uses a new decoding architecture to enable efficient use of cross word context dependent acoustic and long span language models with large vocabularies. In the 1994 ARPA continuous speech recognition evaluation, this recogniser produced the lowest error rate of any of the systems submitted.

Appendix A

Tasks and Databases

Experiments were performed on a variety of standardised continuous speech recognition tasks. As well as the Resource Management and Wall Street Journal tasks (described below) the techniques developed in this work have been used on both American conversational speech (Switchboard [59]) and read speech from other languages [75].

A.1 Parameterisation

In order to use hidden Markov Models the continuous speech data must be converted into a series of discrete observations or frames in a process called *parameterisation* or *coding*. For this work all speech data was parameterised in the same way. This coding had been used successfully with continuous density distributions models on both the TIMIT ([93]) and Resource Management ([91] and see the following section) tasks.

Twelve mel-frequency cepstral coefficients (*MFCCs*) were calculated on a 25ms window advanced by 10ms each frame. The speech data was pre-emphasised with the filter $1 - z^{0.97}$ and then a Hamming window applied to split the continuous stream into frames. A fast Fourier transform is performed to calculate a magnitude spectrum for the frame. This spectrum is averaged into twenty-four triangular bins arranged at equal mel-frequency intervals (where $f_{mel} = 2595 \log_{10}(1 + \frac{f}{700})$). Finally the following cosine transformation and lifter were performed to calculate the twelve MFCCs.

$$c'_i = \sum_{j=1}^{24} m_j \cos\left(\frac{\pi i}{24}(j - 0.5)\right) \tag{A.1}$$

$$c_i = \left(1 + \frac{22}{2} sin(\frac{\pi n}{22})\right) c'_i \tag{A.2}$$

The normalised log energy is also found. The actual acoustic energy in each 25ms period is calculated and the maximum found. All values are then normalised with respect to the maximum (which is set to +10dB) and values below a silence floor (set to -50dB) clamped to that floor.

This thirteen dimensional vector is stored on disk but is expanded upon loading to produce the thirty-nine dimensional parameter vector upon which the models are trained. The vector is extended by appending first and second order differences of the static coefficients. The differences are found using the following regression calculation, first to find the first order differences from the static parameters and then applied to the first order differences to find the second order ones.

$$\delta p_i = \frac{\sum_{\tau=1}^{2} \tau(p_i(t+\tau) - p_i(t-\tau))}{2\sum_{\tau=1}^{2} \tau^2} \tag{A.3}$$

For the first two and last two frames of the utterance, these equations cannot be used and simple differences are calculated instead.

Basic channel normalisation can be performed by subtracting the average of the static cepstral coefficients over an utterance in a process of *cepstral means subtraction* [21]. This was also performed automatically upon loading the file but was not used for all systems. When it was not used this will be specifically mentioned in the system description.

A.2 Resource Management

The DARPA Resource Management Database is a collection of speaker dependent and speaker independent continuous speech data for both training and testing speech recognition systems [74]. The speech comes from a variety of speakers of North American English collected by Texas Instruments in quiet environments using close talking, noise cancelling, head mounted microphones.

The subject of the material is the management of Naval resources and gives the corpus its name. The sentences used as prompts were generated from a list of 900 different template queries. Because of this small number of sentences, the vocabulary and number of different word sequences is relatively limited (with approximately 1000 words and a task perplexity of around 9).

The corpus is split into speaker independent and speaker dependent sections. This work was concerned speaker independent recognition and so the speaker independent, *SI109*, section of the database was used for training. This consists of 3990 sentences from 109 different speakers (hence *SI109*) totalling approximately 3.3 hours of speech data. 2830 of the utterances were from the 78 male speakers and remaining 1160 utterances from the 31 female speakers.

A.2.1 Test Data and Conditions

The speaker independent test data was that used in the four official DARPA Resource Management evaluations occurring in February 1989, October 1989, February 1991 and September 1992. Each of the test sets, which are referred to by the date of the evaluation, consists of 30 sentences from 10 new speakers giving a total of 300 utterances per test.

Two testing conditions were defined, both less constrained than the actual source of the data.

- No Grammar (NG).

	Name	Utterances	Speakers	Words	Duration
Training	SI109	3990	109	34722	3.3 Hours
Test	Feb'89	300	10	2561	16.4 Mins
	Oct'91	300	10	2684	17.0 Mins
	Feb'91	300	10	2484	16.3 Mins
	Sept'92	300	10	2559	16.3 Mins

Table A.1: Summary of the Resource Management database.

No language model is used and the grammar is fully connected. All 991 words can follow any other word and all are equiprobable.

- Word Pair Grammar (WP).

  A word pair syntax defined by the word pairs which occurred in the template queries provides a grammar for constrained recognition. For each word in the vocabulary, a set of equiprobable followers is defined. The average branching factor of this grammar on the test data is approximately 60.

Although the perplexity of both tasks seems relatively high, the complexity of the task is much lower because of the ability of the acoustic models to learn the true source language model (which has very low perplexity) directly from the training data [69].

A.3 Wall Street Journal

The Wall Street Journal Corpus is a larger and more varied database of spoken North American English produced by ARPA as a successor to the Resource Management corpus. The prompting texts were drawn from articles appearing in the Wall Street Journal (a daily American financial newspaper). Initially the articles used for both the training and testing portions of the database were *quality filtered* to limit the vocabulary to the 64,000 most frequently occurring words in the whole database (which consists of approximately 37 million words of text). Recently however the task has moved away from a specific newspaper and a fixed vocabulary towards the wider subject area of North American business news and unlimited vocabulary (with no articles removed because of unseen words).

The data was collected using close talking head mounted Sennheiser microphones (data was also collected simultaneously from a variety of desktop microphones but the experiments described in chapter 6 have only used the close talking data).

This collection was performed in two stages producing a two part database.

- WSJ0.

  For the initial section of the database speakers were asked to say the prompts exactly as given, with any punctuation symbols and numbers expanded as individual words. Half the prompts were *verbalised punctuation* in which the punctuation symbols (such as *COMMA*) were included and said as any other words. The remaining *non-verbalised punctuation* section had all punctuation symbols removed.

This section was split into three sections of almost equal size.

- Longitudinal Speaker Dependent, *LSD*.
- Long term speaker independent (or short term speaker dependent), *SI12*.
- Short term speaker independent, *SI84*.

Of these only the *SI84* section was used for training. This consisted of 7193 sentences from 84 different speakers (42 male and 42 female) for a total around 12.2 hours of speech.

- WSJ1.

  For the second larger section of the database prompt texts were no longer pre-filtered to specify the way in which numbers, punctuation and abbreviations should be pronounced. The individual speakers used their normal speaking style. The resulting data was similar to the non-verbalised WSJ0 data although some of the punctuation, such as *QUOTE*, *UNQUOTE* and *OPEN PARENTHESES* was spoken.

  This was split into two sections of almost equal size.

  - Long term speaker independent, *SI25*.
  - Short term speaker independent, *SI200*.

  Again only the short term speaker independent data *SI200* was used. This contained 29320 sentences from 200 new speakers (100 male and 100 female) for a total of 45.1 hours of speech.

For some of the experiments both the short term speaker independent portions of the database, *SI84* and *SI200*, were used for training and these are collectively referred to as *SI284*. Experiments indicated that supplementing this with the long term speaker independent data (*SI12* and *SI25* referred to as *SI37*) did not noticeably improve accuracy.

The supplied data contained substantial amounts of silence at the beginning and end of each utterance (in total around 15.3 Hours). This quantity of silence data was unnecessary and led to over training of the silence model.

The beginning and end of each sentence tended to be much quieter than within sentence periods of silence (which included non-speech noises such as breathing and tongue clicks). Since these represented the majority of the data the silence model became a better fit for these quiet portions and a worse fit for other phenomena. This resulted in insertion errors occurring if any sounds occurred during periods of ‘silence’. To prevent this effect the training data was stripped of excess silence by limiting the period of silence at the beginning and end of sentence period to 200ms. This had the beneficial side effect of substantially decreasing the storage needed for the acoustic training data as well as reducing the computation required in training.

Language Models

Unlike the Resource Management task the grammar of the sentences was not constrained and any sequence of words could appear in a sentence. With such a simple fully connected grammar a language model is necessary to reduce the task perplexity. The language model provides a probability for each sequence of words. This work has only made use of backed-off N-gram language models [37]. In an N-gram language model the probability of the current (N^{th}) word is assumed to be dependent only upon the identity of the previous $N-1$ words.

So

$$p(w_i|w_1...w_{i-1}) = p(w_i|w_{i-n+1}...w_{i-1}) \quad \text{(A.4)}$$

$$p(w_1...w_i) = \prod_{j=1}^{i} p(w_j|w_{j-n+1}...w_{j-1}) \quad \text{(A.5)}$$

However there are still too many probabilities to be able to estimate these directly from a corpus of text and so a backing-off procedure is used to estimate the probability of rare or unseen events. For example a trigram requires the estimation of *(number of words in vocabulary)*^{3} probabilities. Only a fraction of these can be accurately estimated from the number of occurrences in the corpus for example using

$$Prob(w_i|w_{i-2}w_{i-1}) = Func\left(\frac{Count(w_{i-2}w_{i-1}w_i)}{Count(w_{i-2}w_{i-1})}\right) \quad \text{(A.6)}$$

When there is not enough data for this to be a reliable estimate, backing-off occurs and instead a less specific distribution is used to estimate the probability of the i^{th} word.

$$Pr(w_i|w_{i-2}w_{i-1}) = Prob(w_i|w_{i-2}w_{i-1}) \quad \textit{if known} \quad \text{(A.7)}$$

$$else \quad Bowt(w_{i-2}w_{i-1}) * Pr(w_i|w_{i-1}) \quad \text{(A.8)}$$

and

$$Pr(w_i|w_{i-1}) = Prob(w_i|w_{i-1}) \quad \textit{if known} \quad \text{(A.9)}$$

$$else \quad Bowt(w_{i-1}) * Prob(w_i) \quad \text{(A.10)}$$

Where $Bowt()$ are normalisation constants to ensure that

$$\sum_{word \subset vocabulary} Prob(word|history) = 1.0 \quad \text{(A.11)}$$

Some standard language models were available and these were used unaltered in the Nov'93 evaluation. However for the primary test in Nov'94 sites were given the opportunity to choose their own vocabulary and generate their own language models from a large corpus of text; The North American Business News (NAB) corpus containing over 200 million words of text. Table A.2 shows the sources of this text and highlights with a † the WSJ data used to estimate the language models for the 5k and 20k'93 vocabularies.

Generating language models provided the opportunity to fix some deficiencies that had become apparent in the standard language models [89]. Comparison between the way in which the text had been processed and the way in which prompt texts had been spoken (in the WSJ1 training data) revealed two main problems.

Source	Description	Period	Words
Wall Street Journal	Financial Services Newspaper	1987-1989	38M†
		1990-1992	32M
Dow Jones Information Services	Financial Information Service	1992-1994	40M
Associated Press	General News Wire Service	1988-1990	110M
San Jose Mercury	General Business News	1992	11M

Table A.2: Source of texts for North American Business News corpus.

- Numbers.

  The perl scripts which converted numbers to words never inserted the word *AND* between clauses whereas people often did. For instance the number *123* would be converted to *ONE HUNDRED TWENTY THREE* but would normally be said as *ONE HUNDRED AND TWENTY THREE* or *A HUNDRED AND TWENTY THREE*. This was corrected by inserting the word *AND* or changing occurrences of *ONE* to *A* in the processed text a certain proportion of the times these situations occurred. However no attempt was made to allow for the fact that people occasionally said *1234* as *TWELVE HUNDRED AND THIRTY FOUR* rather than *ONE THOUSAND TWO HUNDRED AND THIRTY FOUR*.

- Abbreviations.

  Some words were pronounced as abbreviations so *CORPORATION* could be said as *CORP.* and *INCORPORATED* as *INC.*. This was corrected by using the language model entries of the full form for the abbreviated word. For simplicity this was actually implemented by manipulating the dictionary (and adding additional pronunciations which resulted in different symbols) rather than altering the language model.

Finally table A.3 gives details of the language models used for experiments and evaluations together with an indication of the perplexity and proportion of out of vocabulary words (measured on the evaluation test data and using the preliminary reference answers in the case of the Nov'94 figures).

A.3.1 Test Data and Conditions

Official ARPA Continuous Speech Recognition evaluations were conducted in Nov'92, Nov'93 and Nov'94. Cambridge University HTK systems (CU-HTK) using decision tree clustered state models (chapter 3) and the tree structured dynamic network decoder (chapter 5) were submitted to the tests in 1993 and 1994 and chapter 6 gives the full results.

For each of these evaluations new test data was recorded. This included development test data which was supplied a few months before the November evaluation as well as the evaluation data (which could only be used for a single run to produce the official results).

Name	Vocab	Type	Size	Words	Test	OOV Rate	Perplexity
5bg	5k	bigram	0.8M	37M	H2_C2 Eval'93	0.29%	106.1
5tg	5k	trigram	3.7M	37M	H2_C1 Eval'93	0.29%	61.5
20bg93	20k'93	bigram	1.5M	37M	H1_C2 Eval'93	1.82%	222.9
20tg93	20k'93	trigram	6.5M	37M	H1_C1 Eval'93	1.82%	142.7
20bg94	20k'94	bigram	5.0M	237M	Eval'94	2.38%	208.6
20tg94	20k'94	trigram	11.2M	237M	H1_C1 Eval'94	2.38%	131.2
65bg94	65k'94	bigram	6.2M	237M	Eval'94	0.65%	232.3
65tg94	65k'94	trigram	16.3M	237M	Eval'94	0.65%	143.9
65fg94	65k'94	fourgram	26.4M	237M	H1_P0 Eval'94	0.65%	130.5

Table A.3: Language models for North American Business News.

Test Set	Domain	Vocabulary	CD-ROM Name	Sentences	Words
Eval'92	WSJ	5k	si_evl5.nvp	330	5353
Dev'93.odd	WSJ	5k	si_dt_05 (subset)	248	4074
Dev'93.s6	WSJ	5k	si_dt_s6	202	3319
Eval'93	WSJ	5k	si_et_h2	215	3851
Eval'92	WSJ	64K QF	si_evl20.nvp	330	5643
Dev'93.odd	WSJ	64K QF	si_dt_20 (subset)	252	4069
Eval'93	WSJ	64K QF	si_et_h1	213	3446
Dev'94	NAB	Unlimited	csrnab1_dt_h1	310	7388
Eval'94	NAB	Unlimited	csrnab1_et_h1	316	8190

Table A.4: Summary of the Wall Street Journal database test sets.

Table A.4 gives details of the various test sets including the vocabulary, official name as well as the name used in chapter 6. The domain refers to the source of the texts either the Wall Street Journal or the more general North American Business News corpus.

For each test in the evaluations, the CC Coordinating Committee defined a set of test conditions. These included specifying the data that could be used to train a system, the language model used and what additional side information is available to modify the system's behaviour.

- Nov'93 Evaluation

  The 1993 evaluation consisted of two core or *hub* tests and several spoke tests [40]. The hub tests investigated continuous speech recognition of North American English speakers in a noise free environment. The spoke tests investigated the effect of noise and adaptation on the accuracy of such systems. Standard bigram and trigram language models at both 5k and 20k were estimated on 37 million words of WSJ text and supplied by MIT-LL.

The hub tests differed in the size of the test vocabulary;

The first hub (H1) used data quality filtered at 65k and defined two open vocabulary tests investigating acoustic modelling which used standard 19979 word (20k) vocabulary bigram (H1-C2) and trigram (H1-C1) language models. A primary test which allowed any vocabulary and training was not entered.

The second (H2) used data from a closed 4986 word (5k) vocabulary and again provided standard bigram. For the H2-C1 test the bigram language model and *SI84* acoustic training had to be used although for the H1-P0 test any language model and acoustic training data were allowed. The full official results can be found in [65].

Test	Vocabulary	Language Model	Acoustic Training	Side Info
93'H1-C1	20k'93	20tg93	*SI284* or *SI37*	None
93'H1-C2	20k'93	20bg93	*SI284* or *SI37*	None
93'H2-P0	5k	Any	Any	Session
93'H2-C1	5k	5bg	*SI84* or *SI12*	None
94'H1-P0	Any	Any	Any	Session
94'H1-C2	20k'94	20tg94	*SI284* or *SI37*	None

Table A.5: Summary of the Nov'93 and Nov'94 evaluation test conditions.

- Nov'94 Evaluation

  The November 1994 Evaluation also consisted of two hub tests and several spoke tests [39]. This time the second hub was over a telephone channel rather than a close-talking microphone and CU-HTK did not take part. However variants of the systems used in the hub tests were submitted for the spoke tests investigating recognition in noise and speaker adaptation. Again only a single run was allowed for systems officially submitted. Full results of all tests can be found in [64].

  For the primary test (H1-P0) any data could be used for training acoustic and language models, however, this had to be produced prior to the testing epoch (16 June 1994 onwards) to ensure it was not possible to 'train on the test data'. Session boundaries and utterance order were known to allow the use of incremental speaker adaptation schemes.

  The contrast condition (H1-C1) investigated differences in acoustic modelling and so defined a standard 20k vocabulary, trigram language model (provided by CMU [80]) and specified the acoustic training data (either *SI284* or *SI37*).

These conditions are summarised in table A.5.

Appendix B

Dictionaries and Phonetic Questions

B.1 Dictionaries

Phone based speech recognisers require a dictionary or lexicon to specify pronunciations for each word. The choice of phone set is usually governed by the availability of machine readable dictionaries (which are suitable for speech recognition) that cover the task vocabulary. However the choice is important as the phone set and dictionary have a significant effect on recognition accuracy.

A single dictionary was used for Resource Management experiments. This was based on the dictionary appearing in [42] (using a set of 48 phones) which contained a single pronunciation for each of the 991 words appearing in the database. The vocabulary for Resource Management is fixed and so these pronunciations covered all words that appeared in any test or training data.

The number of words appearing in both the training and testing portions of the Wall Street Journal database was much higher than for the Resource Management task (over 13000 different words appear in the *SI284* acoustic training data). The size of the vocabulary makes the availability of a suitable dictionaries which covered the required vocabulary of prime importance as it would require substantial effort to generate one by hand.

Three separate dictionaries were used for the Wall Street Journal experiments. The choice of dictionary was governed initially by availability and coverage of the required vocabulary and secondly by system performance.

- Dragon.

  Dragon Systems Inc. made available in 1992 a dictionary that covered the *SI284* training data as well as the 5K and 20K Nov'1993 test vocabularies. This dictionary contained a total of over 29000 words with around 1.1 pronunciations for each word. The original dictionary contained three levels of stress for the vowels as well as syllable boundary markers. For the systems described in chapter 6 stress marking were ignored and the syllable markers deleted to give a dictionary composed of 44 phones.

- LIMSI.

ARPABET	LIMSI	Example	IPA	ARPABET	LIMSI	Example	IPA
Vowels				Plosives			
aa	a	bott	/ɑ/	b	b	bet	/b/
ae	@	bat	/æ/	d	d	debt	/d/
ah	^	but	/ʌ/	g	g	get	/g/
ao	c	bought	/ɔ/	k	k	cat	/k/
aw	W	bout	/aʊ/	p	p	pet	/p/
ax	x	about	/ə/	t	t	tat	/t/
axr	X	butter	/ɚ/	Fricatives			
ay	Y	bite	/aɪ/	dh	D	that	/ð/
eh	E	bet	/ɛ/	th	T	thin	/θ/
er	R	bird	/ɜr/	f	f	fan	/f/
ey	e	bait	/eɪ/	v	v	van	/v/
ih	I	bit	/ɪ/	s	s	sue	/s/
ix	—	dating	/ɜ/	sh	S	shoe	/ʃ/
iy	i	beet	/i/	z	z	zoo	/z/
ow	o	boat	/o/	zh	Z	measure	/ʒ/
oy	O	boy	/ɔɪ/	Affricates			
uh	U	book	/ʊ/	ch	C	cheap	/ʧ/
uw	u	boot	/u/	jh	J	jeep	/ʤ/
Glides				Nasals			
l	l	led	/l/	m	m	met	/m/
el	L	bottle	/l̩/	em	M	bottom	/m̩/
r	r	red	/r/	n	n	net	/n/
w	w	wed	/w/	en	N	button	/n̩/
y	y	yet	/j/	ng	G	thing	/ŋ/
hh	h	hat	/h/				

Table B.1: The LIMSI dictionary phone set.

The following year LIMSI made available the lexicon they used in the 1993 Evaluation. Experiments indicated that the LIMSI dictionary provided better consistency and higher accuracy than the Dragon dictionary and so once it had been made available this dictionary was used (in particular the systems used in the Nov'94 ARPA CSR evaluation were based on the LIMSI dictionary). Approximately 22,700 words were present in the

LIMSI	Dragon	KFL	LIMSI	Dragon	KFL
Vowels			Plosives		
aa	aw,ah	aa	b	b	b
ae	aa	ae	d	d	d,dd
ah	uh	ah	g	g	g
ao	awh	ao	k	k	k,kd
aw	ow	aw	p	p	p,pd
ax	-	ax	t	t	t,td
axr	-	-	Fricatives		
ay	ay	ay	dh	dh	dh
eh	eh,ae	eh	th	th	th
er	ur	er	f	f	f
ey	ey	ey	v	v	v
ih	ih	ih	s	s	s
ix	-	ix	sh	sh	sh
iy	ee	iy	z	z	z
ow	oh	ow	zh	zh	-
oy	oy	oy	Affricates		
uh	oo	uh	ch	ch	ch
uw	ooh	uw	jh	j	jh
Glides			Nasals		
l	l	l	m	m	m
el	ul	-	em	um	-
r	r	r	n	n	n
w	w	w	en	un	en
y	y	y	ng	ng	ng
hh	h	hh			

Table B.2: Equivalences between different phone sets

dictionary (again with around 1.1 pronunciations per word) and since the dictionary did not contain lexical stress information the phone set (of 45 phones) was used unchanged.

- Truetalk Text-to-Speech System.

  The A. T. & T. text-to-speech system, Truetalk, was used to supplement the LIMSI dictionary and provide pronunciations when the vocabulary was extended beyond that covered by the LIMSI dictionary. Pronunciations generated by TrueTalk were mapped to the LIMSI phone set (using the equivalences in table B.2). This method of extending the dictionary was another reason for using the LIMSI dictionary as it was easier to map the phone set used in the speech synthesiser to that of LIMSI rather than that of Dragon.

The original dictionary supplied by LIMSI used a single character to represent each phone and this had to be modified for use by the training tools and the recogniser. Consequently a single case alphabetic representation similar to the *ARBAPET* style symbols used in TIMIT as well as the KFL Resource Management and Dragon WSJ dictionaries was used instead.

The ARPABET symbols together are shown in table B.1 together with the original LIMSI ones. The table also gives an example of a word containing each phone (with the letters realised as the phone underlined) and the equivalent standard International Phonetic Alphabet symbol.

Table B.2 shows how the phone sets from the Dragon and Kai-Fu Lee dictionaries are related to the LIMSI phones. Most of the phones have direct one to one equivalents but there are some exceptions.

- Kai-Fu Lee, Resource Management

  This phone set contains additional phones for occurrences of `d`, `k`, `p` and `t` when they may are optionally released (`dd`, `kd`, `pd` and `td`) as well as an explicit model of the compound `t s` (`ts`) and the alveolar flap (`dx`). It does not have equivalents for the reduced form of `er` (`axr`), for two of the syllabics (`el` and `em`) or for `zh`.

- Dragon WSJ

  The Dragon WSJ phone set does not have any explicit models for reduced vowels (`ax`, `ix` and `axr`) but has two extra vowels not included in the LIMSI phone set (`ae` which is very similar to `eh` but is always followed by `r` and `ah` which is similar to `aw`).

General Questions	
Feature	Phones
Stop	b d g k p t
Nasal	em en m n ng
Fricative	ch dh f jh s sh th v z zh
Liquid	el hh l r w y
Vowel	aa ae ah ao aw ax axr ay eh er ey ih ix iy ow oy uh uw
Front	ae b eh em f ih ix iy m p v w
Central	ah ao axr d dh el en er l n r s t th z zh
Back	aa ax ch g hh jh k ng ow sh uh uw y

Table B.3: General questions.

B.2 Phonetic Questions

The questions used to construct the decision trees are chosen to incorporate linguistic knowledge into the clustering procedure by ensuring that unseen contexts are grouped with those which one would expect to be linguistically similar ([14], [88]).

Simple position independent binary questions concerning phonetic features were used. These were of the form *feature[off]* meaning "Does the phone at offset *off* have the feature *feature* ?". The complete set of questions is defined by the range of offsets used ($+/-1$ for triphones or $+/-2$ for quinphones) and the set of phonetic features. The question is answered by checking if the phone at the offset is a member of the set of phones which has the phonetic feature. For instance in the phone string `w ih dh y uw` the question Liquid[+1] concerning `dh` would be true because the phone at offset +1, `y`, is a member of the set of liquids, `el hh l r w y`. Whereas the question Liquid[-1] would be false because `ih` does not appear in the set.

Other questions concerning non-phonetic features such as the position of word boundaries and the speaker gender were also used. These are described together with the experiments in which they were used.

Vowel Questions	
Feature	Phones
Front Vowel	ae eh ih ix iy
Central Vowel	aa ah ao axr er
Back Vowel	ax ow uh uw
Long	ao aw el em en iy ow uw
Short	aa ah ax ay eh ey ih ix oy uh
Dipthong	aw axr ay el em en er ey oy
Front Start	aw axr er ey
Fronting	ay ey oy
High	ih ix iy uh uw
Medium	ae ah ax axr eh el em en er ey ow
Low	aa ae ah ao aw ay oy
Rounded	ao ow oy uh uw w
Unrounded	aa ae ah aw ax axr ay eh el em en er ey hh ih ix iy l r y
Reduced	ax axr ix
IVowel	ih ix iy
EVowel	eh ey
AVowel	aa ae aw axr ay er
OVowel	ao ow oy
UVowel	ah ax el em en uh uw

Table B.4: Vowel questions.

Preliminary experiments showed that the addition of further linguistically well motivated questions increased the accuracy of the resultant models, whilst reducing the number of questions so that only broad phonetic distinctions remained reduced the accuracy. Consequently a single large set containing questions about a wide variety of phonetic features was used for all experiments. The clustering procedure automatically chooses the most important questions and so the broad class distinctions tend to be used most often [95].

The set of phones for each phonetic feature depends upon the phone set used. Tables B.3, B.5 and B.4 show the features together with the set LIMSI phones with the feature. These sets can be transformed to use the Dragon or Resource Management phones using the equivalences shown in Table B.2.

As well as these features, a question was included for each phone. These are not shown but just consist of the feature with the name "phone" with one member `phone`. This allows each phonetic context to be treated separately when enough data is available.

Consonant Questions	
Feature	Phones
Unvoiced	ch f hh k p s sh t th
Voiced	b d dh el em en g jh l m n ng r v w y
Front Consonant	b em f m p v w
Central Consonant	d dh el en l n r s t th z zh
Back Consonant	ch g hh jh k ng sh y
Fortis	ch f k p s sh t th
Lenis	b d dh g jh v z zh
Neither Fortis or Lenis	el em en hh l m n ng r w y
Coronal	ch d dh el en jh l n r s sh t th z zh
Non Coronal	b em f g hh k m ng p v w y
Anterior	b d dh el em en f l m n p s t th v w z
Non Anterior	ch g hh jh k ng r sh y zh
Continuent	dh el em en f hh l m n ng r s sh th v w y z zh
Non Continuent	b ch d g jh k p t
Positve Strident	ch jh s sh z zh
Negative Strident	dh f hh th v
Neutral Strident	b d el em en g k l m n ng p r t w y
Syllabic	axr el em en er
Voiced Stop	b d g
Unvoiced Stop	p t k
Front Stop	b p
Central Stop	d t
Back Stop	g k
Voiced Fricative	ch dh v z zh
Unvoiced Fricative	ch f s sh th
Front Fricative	f v
Central Fricative	dh s th z
Back Fricative	ch jh sh zh
Affricate	ch jh
Not Affricate	dh f s sh th v z zh

Table B.5: Consonant questions.

Bibliography

[1] Alleva F., Huang X., Hwang M-Y. 'An Improved Search Algorithm Using Incremental Knowledge for Continuous Speech Recognition.', *Proceedings ICASSP, Minneapolis, pages 307-310*, 1993.

[2] Aubert X., Dugast C., Ney H., Steinbiss V. 'Large Vocabulary Continuous Speech Recognition of Wall Street Journal Data.', *Proceedings ICASSP, Adelaide, pages 129-132*, 1994.

[3] Austin S., Peterson P., Placeway P., Schwartz R., Vandergrift J. 'Towards a Real-Time Spoken Language System Using Commercial Hardware.', *Proceedings DARPA Speech and Natural Language Workshop, Hidden Valley, pages 72-77*, 1990.

[4] Austin S., Schwartz R., Placeway P. 'The Forward-Backward Search Algorithm.', *Proceedings ICASSP, Toronto, pages 697-700*, 1991.

[5] Bahl L. R., Balakrishnan-Aiyer S., Franz M., Gopalakrishnan P. S., Gopinath R., Novak M., Padmanabhan M., Roukos S. 'The IBM Large Vocabulary Continuous Speech Recognition System for the ARPA NAB News Task.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 121-126*, 1995.

[6] Bahl L. R., de Souza P. V., Gopalakrishnan P. S., Nahamoo D., Picheny M. A. 'Context Dependent Modeling of Phones in Continuous Speech Using Decision Trees.', *Proceeding DARPA Speech and Natural Language Processing Workshop, Pacific Grove, pages 264-268*, 1991.

[7] Bahl L. R., de Souza P. V., Gopalakrishnan P. S., Nahamoo D., Picheny M. A. 'Decision Trees for Phonological Rules in Continuous Speech.', *Proceedings ICASSP, Toronto, pages 185-188*, 1991.

[8] Bahl L. R., de Souze P. V., Gopalakrishnan P. S., Nahamoo D., Picheny M. A. 'A Fast Match for Continuous Speech Recognition Using Allophonic Models.', *Proceedings ICASSP, San Fransisco, pages 17-20*, 1992.

[9] Baum L. E. 'An Inequality and Associated Maximisation Technique in Statistical Estimation for Probabilistic Functions of Markov Processes.', *Inequalities 3, pages 1-8*, 1972.

[10] Bellegarda J. R., Nahamoo D. 'Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Word Recognition.', *Proceedings ICASSP, Glasgow, pages 13-16*, 1989.

[11] Bocchieri 'A Study of the Beam-Search Algorithm for Large Vocabulary Continuous Speech Recognition and Methods for Improved Efficiency.', *Proceedings Eurospeech, Berlin, pages 1521-1524*, 1993.

[12] Brown P. 'The Acoustic-Modelling Problem in Automatic Speech Recognition.', *PhD Thesis, IBM T. J. Watson Research Center*, 1987.

[13] Chase L., Rosenfeld R., Hauptmann A., Ravishanhkar M., Thayer E., Placeway P., Weide R., Lu C. 'Improvements in Language, Lexical and Phonetic Modelling in Sphinx-II.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 60-65*, 1995.

[14] Connor J. D. 'Phonetics.', *Penguin Books*, 1973.

[15] Cormen T. H., Leiserson C. E., Rivest R. L. 'Introduction to Algorithms.', *Massachusetts Institute of Technology Press*, 1990.

[16] Davis S. B., Mermelstein P. 'Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences.', *IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 28, No. 4, pages 357-366*, 1980.

[17] Digalakis V., et al. 'SRI November 1993 CSR Hub Evalulation.', *Oral Presentation at ARPA Workshop on Spoken Language Technology, Merrill Lynch Conference Centre*, 1994.

[18] Digalakis V., Weintraub M., Sankar A., Franco H., Neumeyer L., Murveit H. 'Continuous Speech Dictation on ARPA's North American Business News Domain.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 88-93*, 1995.

[19] Downey S., Russell M. J. 'A Decision Tree Approach to Task Independent Speech Recognition.', *Proceedings IOA Autumn Conference on Speech and Hearing, Windermere, pages 181-188*, 1992.

[20] Dugast C., Kneser R., Aubert X., Ortmanns S., Beulen K., Ney H. 'Continuous Speech Recognition Tests and Results for the NAB'94 Corpus.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 156-161*, 1995.

[21] Furui S. 'Cepstral Analysis Technique for Automatic Speaker Verification.', *IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 29, No. 2, pages 254-272*, 1981.

[22] Gauvain J-L., Lee C-H. 'MAP Estimation of Continuous Density HMM: Theory and Applications.', *Proceedings DARPA Speech and Natural Language Workshop, pages 185-190*, 1992.

[23] Gauvain J. L., Lamel L. F., Adda G., Adda-Decker M. 'The LIMSI Continuous Speech Dictation System: Evaluation on the ARPA Wall Street Journal Task.', *Proceedings ICASSP, Adelaide, pages 557-560*, 1994.

[24] Gauvain J.-L., Lamel L., Adda-Decker M. 'Developments in Large Vocabulary Dictation: The LIMSI Nov94 NAB System.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 131-138*, 1995.

[25] Giachin E. P., Rosenberg A. E., Lee C-H. 'Word Juncture Modeling Using Phonological Rules for HMM-Based Continuous Speech Recognition.', *Proceedings ICASSP, Alberqueque, pages 737-740*, 1990.

[26] Gotoh Y., Hochberg M. M., Silverman H. F. 'Using MAP Estimated Parameters to Improve HMM Speech Recognition Performance.', *Proceedings ICASSP, Adelaide, pages 229-232*, 1994.

[27] Hetherington I. L., Phillips M. S., Glass J. R., Zue V. W. 'A* Word Network Search for Continuous Speech Recognition.', *Proceedings Eurospeech, Berlin, pages 1533-1536*, 1993.

[28] Hochberg M. M., Cook G. D., Renal S. J., Robinson A. J., Schechtman R. S. 'The 1994 ABBOT Hybrid Connectionist-HMM Large-Vocabulary Recognition System.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 170-177*, 1995.

[29] Hon H-W., Lee K-F. 'Recent Progress in Robust Vocabulary-Independent Speech Recognition.', *Proceeding DARPA Speech and Natural Language Processing Workshop, Pacific Grove, pages 258-263*, 1991.

[30] Huang X. D., Lee K-F. 'On Speaker Indpendent, Speaker Dependent and Speaker Adaptive Speech Recognition.', *Proceedings ICASSP, Toronto, pages 877-880*, 1991.

[31] Hwang M-Y., Huang X. 'Subphonetic Modeling for Speech Recognition.', *Proceedings DARPA Speech and Natural Language Workshop, New York, pages 174-179*, 1992.

[32] Hwang M-Y., Huang X., Alleva F. 'Prediciting Unseen Triphones with Senones.', *Proceedings ICASSP, Minneapolis, pages 311-314*, 1993.

[33] Hwang M., Rosenfeld R., Thayer E., Mosur R., Chase L., Weide R., Huang X. 'Improving Speech Recognition Performance via Phone-Dependent VQ Codebooks and Adaptive Language Models in SPHINX-II.', *Proceedings ICASSP, Adelaide, pages 549-552*, 1994.

[34] Jelinek F., Mercer R. L. 'Interpolated Estimation of Markov Source Parameters from Sparse Data.', *Pattern Recognition in Practice, North-Holland, pages 381-397*, 1980.

[35] Juang B. H., Levinson S. E., Sondhi M. M. 'Maximum Likelihood Estimation for Multivariate Mixture Observations of Markov Chains.', *IEEE Transactions on Information Theory, Vol . 32, No. 2, pages 307-309*, 1986.

[36] Kannan A., Ostendorf M., Rohlicek J.R. 'Maximum Likelihood Clustering of Gaussians for Speech Recognition.', *IEEE Transactions on Speech and Audio Processing, Vol.2 No. 3, pages 453-455*, 1994.

[37] Katz S. M. 'Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recogniser.', *IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 35, No. 3, pages 400-401*, 1987.

[38] Kenny P., Hollan R., Gupta V., Lennig M., Mermelstein P., O'Shaughnessy D. 'A* - Admissible Heuristics for Rapid Lexical Access.', *Proceedings ICASSP, Toronto, pages 689-692*, 1991.

[39] Kubala F. 'Design of the 1994 CSR Benchmark Tests.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Barton Creek, pages 41-46*, 1995.

[40] Kubala F., Bellegarda J., Cohen J., Pallett D., Paul D., Phillips M., Rajasekaran R., Richardson F., Riley M., Rosenfeld R., Roth B., Weintraub M. 'The Hub and Spoke Paradigm for CSR Evaluation.', *Proceedings ARPA Workshop on Human Language Technology, Merrill Lynch Conference Centre, pages 31-36*, 1994.

[41] Lacouture R., Normandin Y. 'Efficient Lexical Access Strategies.', *Proceedings Eurospeech, Berlin, pages 1537-1540*, 1993.

[42] Lee K-F. 'Automatic Speech Recognition: The Development of the SPHINX System.', *Kluwer Acedemic Press*, 1989.

[43] Lee K-F. 'Context-Dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition.', *IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 38, No 4, pages 599-609*, 1990.

[44] Lee K-F. 'Large Vocabulary Speaker Independent Continuous Speech Recognition.', *PhD Thesis, Carnegie Mellon University*, 1988.

[45] Lee K-F., Hon H-W. 'Speaker-Independent Phone Recognition Using Hidden Markov Models.', *IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 37, No 11, pages 1641-1648*, 1989.

[46] Leggetter C. J., Woodland P. C. 'Flexible Speaker Adaptation Using Maximum Likelihood Linear Regression.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Barton Creek, pages 110-115*, 1995.

[47] Leggetter C. J., Woodland P. C. 'Speaker Adaptation of Continuous Density HMMs using Multi-variate Linear Regression.', *Proceedings International Conference on Spoken Language Processing, Yokohama, pages 451-454*, 1994.

[48] Levinson S. E. 'Continuously Variable Duration Hidden Markov Models for Automatic Speech Recognition.', *Computer Speech and Language, Vol. 1, No. 1, pages 29-45*, 1986.

[49] Ljolje A., Riley M., Hindle D., Pereira F. 'The AT&T 60,000 Word Speech-to-Text System.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 162-165*, 1995.

[50] Lowerre B., Reddy R. 'The Harpy Speech Understanding System.', *Trends In Speech Recognition, Prentice Hall Publishers, pages 340-360*, 1980.

[51] Morgan N., et al. 'Scaling a Hybrid HMM/MLP System for Large Vocabulary CSR.', *Oral Presentation at ARPA Workshop on Spoken Language Technology, Merrill Lynch Conference Centre*, 1994.

[52] Murveit H., Butzberger J., Digalakis V., Weintraub M. 'Large-Vocabulary Dictation Using SRI's Decipher Speech Recognition System: Progressive Search Techniques.', *Proceedings ICASSP, Minneapolis, pages 319-322*, 1993.

[53] Ney H., Haeb-Umbach R., Tran B-H., Oerder M. 'Improvements in Beam Search for 10000 Word Continuous Speech Recognition.', *Proceedings ICASSP, San Francisco, pages 9-12*, 1992.

[54] Nguyen L., Anastasakos T., Kubala F., LaPre C., Makhoul J., Schwartz R., Yuan N., Zavaliagkos G., Zhao Y. 'The 1994 BBN/BYBLOS Speech Recognition System.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 77-81*, 1995.

[55] Nilsson N. J. 'Principles of Artificial Intelligence.', *Springer-Verlay*, 1982.

[56] Normandin Y., Bowness D., Cardin R., Drouin C., Lacouture R., Lzarides A. 'CRIM's November 94 Continuous Speech Recognition System.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 153-155*, 1995.

[57] Odell J. J. 'The Use of Decision Trees With Context Sensitive Phoneme Modelling.', *MPhil Thesis, Cambridge University, Engineering Dept*, 1992.

[58] Odell J. J., Valtchev V., Woodland P. C., Young S. J. 'A One-Pass Decoder Design for Large Vocabulary Recognition.', *Proceedings ARPA Workshop on Human Language Technology, Merrill Lynch Conference Centre, pages 405-410*, 1994.

[59] Odell J. J., Valtchev V., Woodland P. C., Young S. J. 'Recent Developments in the HTK Large Vocabulary Continuous Speech Recognition System.', *Proceedings IOA Autumn Conference on Speech and Hearing, Windermere, pages 39-46*, 1993.

[60] Odell J. J., Woodland P. C., Young S. J. 'Tree-Based State Clustering for Large Vocabulary Speech Recognition.', *International Symposium on Speech Image Processing and Neural Networks, Hong Kong, pages 690-693*, 1994.

[61] Ostendorf M., Bechwati I., Kimball O. 'Context Modeling with the Stochastic Segment Model.', *Proceedings ICASSP, San Fransisco, pages 389-392*, 1992.

[62] Ostendorf M., et al. 'Stochastic Segment Modelling for Continuous Speech Recognition: Wall Street Journal Benchmark Report.', *Oral Presentation at ARPA Workshop on Spoken Language Technology, Merrill Lynch Conference Centre*, 1994.

[63] Ostendorf M., Richardson F., Iyer R., Kannan A., Ronen O., Bates R. 'The 1994 BU NAB News Benchmark System.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 139-143*, 1995.

[64] Pallet D. S., Fiscus J. G., Fisher W. M., Garofolo J. S., Lund B. A., Przybocki M. A. '1994 Benchmark Tests for the ARPA Spoken Language Program.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 5-38*, 1995.

[65] Pallett D. S., Fiscus J. G., Fisher W. M., Garofolo J. S., Lund B. A., Przybocki M. A. '1993 Benchmark Tests for the ARPA Spoken Language Program.', *Proceedings ARPA Workshop on Human Language Technology, Merrill Lynch Conference Centre, pages 49-74*, 1994.

[66] Paul D. B. 'The Lincoln Large-Vocabulary Stack-Decoder Based HMM CSR.', *Proceedings ARPA Workshop on Human Language Technology, Merrill Lynch Conference Centre, pages 399-404*, 1994.

[67] Paul D. B. 'An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model.', *Proceedings ICASSP, San Fransisco, pages 25-28*, 1992.

[68] Paul D. B. 'New Developments in the Lincoln Stack-Decoder Based Large-Vocabulary CSR System.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 143-147*, 1995.

[69] Paul D. B., Baker J. K., Baker J. M 'On the Interaction Between True Source, Training and Testing Language Models.', *Proceedings ICASSP, Toronto, pages 569-572*, 1991.

[70] Paul D. B. 'Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder.', *Proceedings ICASSP, Toronto, pages 693-696*, 1991.

[71] Pieraccini R., Lee C-H., Giachin E., Rabiner L. R. 'Complexity Reduction in a Large Vocabulary Speech Recogniser.', *Proceedings ICASSP, Toronto, pages 729-732*, 1991.

[72] Pieraccini R., Lee C. H., Giachin E., Rabiner L. R. 'Implementation Aspects of Large Vocabulary Recognition Based on Intraword and Interword Phonetic Units.', *Proceedings DARPA Speech and Natural Language Workshop, Hidden Valley, pages 311-318*, 1990.

[73] Placeway P., Schwartz R., Fung., Nguyen L. 'The Estimation of Powerful Language Models from Small and Large Corpora.', *Proceedings ICASSP, Minneapolis, pages 33-36*, 1993.

[74] Price P. J., Fischer W., Bernstein J., Pallett D. 'A Database for Continuous Speech Recognition in a 1000 Word Domain.', *Proceedings ICASSP, New York, pages 651-654*, 1988.

[75] Pye D., Woodland P. C., Young S. J. 'Large Vocabulary Multilingual Speech Recognition Using HTK.', *Proceedings Eurospeech (Forthcoming), Madrid*, 1995.

[76] Rabiner L. R. 'A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.', *Proceedings of the IEEE, Vol. 77, No. 2, pages 257-285*, 1989.

[77] Robinson A. J. 'An Application of Recurrent Nets to Phone Probability Estimation.', *IEEE Transactions on Neural Networks, Vol. 5, No 2, pages 298-305*, 1994.

[78] Robinson T., Hochberg M., Renals S. 'IPA: Improved Phone Modelling with Recurrent Neural Networks.', *Proceedings ICASSP, Adelaide, pages 37-40*, 1994.

[79] Rogina I., Waibel A. 'The JANUS Speech Recogniser.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 166-169*, 1995.

[80] Rosenfeld R. 'The CMU Statistical Language Modeling Toolkit and its use in the 1994 ARPA CSR Evalulation.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Barton Creek, pages 57-50*, 1995.

[81] Roth R., Gillick L., Orloff J., Scattone F., Gao G., Wegmann S., Baker J. 'Dragon Systems' 1994 Large Vocabulary Continuous Speech Recogniser.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 116-120*, 1995.

[82] Sakoe H., Chiba S. 'Dynamic Programming Algorithm Optimization for Spoken Word Recognition.', *Readings in Speech Recognition, Morgan Kaufmann Publishers, pages 159-165*, 1978.

[83] Scattone F., et al. 'Dragon's Large Vocabulary Speech Recognition System.', *Oral Presentation at ARPA Workshop on Spoken Language Technology, Merrill Lynch Conference Centre*, 1994.

[84] Schwartz R., Austin S. 'Efficient, High-Performance Algorithms for N-Best Search.', *Proceedings DARPA Speech and Natural Language Workshop, Hidden Valley, pages 6-11*, 1990.

[85] Sekine S., Steling J., Grishman R. 'NYU/BBN 1994 CSR Evaluation.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Austin, pages 148-152*, 1995.

[86] Soong F. K., Huang E-F. 'A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition.', *Proceedings ICASSP, Toronto, pages 705-708*, 1991.

[87] Viterbi A. J. 'Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm.', *IEEE Transactions on Information Theory, Vol. 13, No. 2, pages 260-269*, 1967.

[88] Wells J. C. 'Accents of English 3: Beyond the British Isles.', *Cambridge University Press*, 1982.

[89] Woodland P. C., Leggetter C. J., Odell J. J., Valtchev V., Young S. J. 'The Development of the 1994 HTK Large Vocabulary Speech Recognition System.', *Proceedings ARPA Spoken Language Systems Technology Workshop, Barton Creek, pages 110-115*, 1995.

[90] Woodland P. C., Odell J. J., Valtchev V., Young S. J. 'Large Vocabulary Continuous Speech Recognition Using HTK.', *Proceedings ICASSP, Adelaide, pages 125-128*, 1994.

[91] Woodland P. C., Young S. J. 'Benchmark DARPA RM Results with the HTK Portable HMM Toolkit.', *Proceedings DARPA Continuous Speech Recognition Workshop, Stanford, pages 71-76*, 1992.

[92] Woodland P. C., Young S. J. 'The HTK Tied-State Continuous Speech Recogniser.', *Proceedings Eurospeech, Berlin, pages 2207-2210*, 1993.

[93] Young S. J. 'The General Use of Tying in Phoneme-Based HMM Speech Recognisers.', *Proceedings ICASSP, San Fransisco, pages 569-572*, 1992.

[94] Young S. J. 'The HTK Hidden Markov Model Toolkit: Design and Philosophy.', *Cambridge University Engineering Dept. Technical Report No TR152*, 1993.

[95] Young S. J., Odell J. J., Woodland P. C. 'Tree-Based Tying for High Accuracy Acoustic Modelling.', *Proceedings ARPA Workshop on Human Language Technology, Merrill Lynch Conference Centre, pages 286-291*, 1994.

[96] Young S. J., Russell N. H. 'Token Passing: A Simple Conceptual Model for Continuous Speech Recognition Systems.', *Cambridge University Engineering Dept., Technical Report No. 38*, 1989.

[97] Young S. J., Woodland P. C., Bynre W. J. 'HTK Version 1.5: User, Reference and Programmer Manuals.', *Cambridge University Engineering Dept & Entropic Research Laboratories Inc*, 1993.

[98] Zavaliagkos G. 'BBN Hub System and Results.', *Oral Presentation at ARPA Workshop on Spoken Language Technology, Merrill Lynch Conference Centre*, 1994.

[99] Ney H.,Mergel D., Noll A., Paeseler A. 'A Data-Driven Organisation of the Dynamic Programming Beam Search for Continuous Speech Recognition.', *Proceedings ICASSP, Dallas, pages 833-836*, 1987.