Automatic Speech Recognition: From Study to Practice

by:
Sara Sharifzadeh

Supervisor:
Javier Serrano Garcia

Submitted to the Department of Microelectronics

in partial fulfillment of the requirements for the degree of Master of Science in Multimedia Technologies
at the UNIVERSITY OF AUTONOMA DE BARCELONA

July 2010

Abstract

Today, automatic speech recognition (ASR) is widely used for different purposes such as robotics, multimedia, medical and industrial application. Although many researches have been performed in this field in the past decades, there is still a lot of room to work. In order to start working in this area, complete knowledge of ASR systems as well as their weak points and problems is inevitable. Besides that, practical experience improves the theoretical knowledge understanding in a reliable way. Regarding to these facts, in this master thesis, we have first reviewed the principal structure of the standard HMM-based ASR systems from technical point of view. This includes, feature extraction, acoustic modeling, language modeling and decoding. Then, the most significant challenging points in ASR systems is discussed. These challenging points address different internal components characteristics or external agents which affect the ASR systems performance. Furthermore, we have implemented a Spanish language recognizer using HTK toolkit. Finally, two open research lines according to the studies of different sources in the field of ASR has been suggested for future work.

To My Husband, Ehsan

Acknowledgments

Thanks to my adviser, Dr. Javier Serrano , for his guidance for introducing me to the problem of speech recognition. Thanks to my thesis committee members. Thanks to my colleague Héctor Delgado Flores for his kind help in implementation problems. Thanks to my husband, and my family for their help and support. This research was funded by the microelectronic department , University of Autonoma De Barcelona.

Contents

List of Tables

List of Figures

1 Introduction

The problem of converting speech to text which is known as automatic speech recognition (ASR), has been an area of great interest and activity to the signal processing, artificial intelligence and human language processing scientists, over the past several decades. Despite its valuable achievements, ASR still remains far from being a solved problem. No doubt, there has been much progress and ASR technology is now in widespread use; however, there is still a considerable gap between human and machine performance, particularly in adverse conditions.

The rapid wide spread use of ASR systems in variety of applications from multimedia to medical and industrial applications, has motivated many researchers to work in this field. Therefore, lots of research has been performed to increase these systems performance and capabilities. Today we are witness of the distribution of ASR amazing applications in intelligent houses, cars, mobile and etc.

1.1 ASR History

The earliest approach to ASR was based on template method [29]. One key idea of this method is to formulate a unit like a word or phoneme model by a training paradigm e.g. vector quantization. The unit model is built in a training phase. The second key idea is to employ some form of dynamic programming, which is often called time wrapping algorithm (DTW) to evaluate the similarity between the utterance and the template. The complexity of this problem is related to the speaking rates across talkers and the vocabulary size. Although this method yields good performance for a variety of practical applications, it has a number of deficiencies. For example multi-unit templates are required for high recognition accuracy which requires considerable memory storage which was a significant limitation in the early days of ASR systems. In addition, substantial computation is required to reach an optimal DTW path. This is the most vulnerable drawback of the conventional template approaches.

In the 1980s, the hidden Markov model (HMM) became widely applied in speech recognition community which is still the most dominated approach in ASR. The basic idea behind the HMM is that the speech signal can be characterized by a parametric stochastic process. Essentially, a HMM can be thought of as a finite-state machine where the transitions between the states are governed by probabilistic laws. In HMMs, the sequence of acoustic events belonging to a specific class is treated as if it were the output of a process modeled by the corresponding HMM. Variations between observation sequences of the same class are modeled by the underlying stochastic nature of the HMM. In this sense, HMMs do not directly involve with time alignment. The key parameters to be determined in an HMM are the number of states in the model, the state-transition probability distribution, the observation symbol probability distribution, and the initial-state distribution. One prominent and crucial characteristic of an HMM-based ASR is that large amounts of training data are required to obtain acceptable estimates of the model parameters to recognize the observation sequences in best way.

Another approach to ASR is to apply neural networks to solve sub-tasks of the speech recognition problems. In this manner, neural network paradigms and conventional ASR technology are integrated to provide satisfying solutions efficiently. Basically, a neural network is a massively parallel distributed processor. Among the many appealing properties of a neural network, the property that is of

prime significance is the ability of the neural network to learn concepts from given numerical data inductively. A neural network improves its performance by adjusting its synaptic weights. A feed forward neural network with sufficient hidden nodes was proved to be a universal approximator. These networks are often called multi-layer perceptrons (MLPs) or backpropagation networks (BPNs). A major limitation of standard MLPs is that they are structured to learn input-output mappings that are static. Unfortunately, speech signals are inherently dynamic in nature. Hence, some modifications of the standard MLPs is performed in order to make them responsive to time-varying signals. Numerous paradigms and structures are investigated in this case. The drawbacks of MLPs is also the training time.

While limitation in computer resources like the level of computation and memory storage was a significant problem, due to improvements in computer technology and the ever increasing level of speed and memory capacity, today, huge repositories of vocabulary and training data sets could be used to increase the quality of training and therefore, the over all performance of the ASR systems.

1.2 New ASR Research Areas

Regarding to considerable improvements in ASR systems, recently, new research areas has been introduced [30,31]. These topics are mostly depend on exploitation of the knowledge about human speech perception, understanding and cognition. This rich area is based on psychological and physiological processes in human. One specific case, is that the human brain processes the spoken language so that it adapt to nonnative accent and reacquaint itself to variations in language. This amazing human capability in adaptation, have motivated researchers to improve current poor ASR models in this case. Studies from the literature show that the future research in ASR is highly motivated for investing in self-adaptive or self learning systems. But, this is possible if sufficient understanding of human speech processing mechanism, be obtained.

1.3 Objectives and Structure of the Thesis

The most important objectives of this master thesis are:

1. To study different components of the standard HMM-based ASR systems in detail and also the most important parameters that make them useful for large vocabulary continues speech recognition.
2. To study the most important challenges in ASR and classify different commonly used methods to deal with them
3. To build a Spanish language recognizer using HTK toolkit and perform different experiments
4. To suggest research lines for future work in the field of ASR

There are reasons behind any of these objectives that we will discuss here shortly.

HMM-based ASR is the most dominated method that has been used for speech recognition and most successful speech recognizer have been built using this technique. Also, as we are motivated to continue working in this area, its useful to know the most challenging points and their solution methods. Besides that, it is necessary to implement the theoretical knowledge for having deeper understanding. HTK is a well known powerful speech processing toolkit which supports variety ASR techniques. Therefore, it is a good means for starting the implementation task. Finally, due to our studies of

different ASR materials, it is worth suggesting some open research lines for future work which can highlight a direct goal more specifically.

In this thesis, we will start in chapter two with a deep study about the principal components of a HMM-based ASR model. This includes, feature extraction, acoustic modeling, language modeling and decoding. Then, the most significant challenging points in ASR systems will be discussed. The first discussed case is the feature projection and discriminative training algorithms, which are used to improve the recognition task for large vocabulary continues speech recognizers (LVCR). In addition, robust speech recognition problem will be introduced and a brief review about the common approaches in this field will be presented. Adaptation, is another commonly used method to improve the recognition task. We will also explain different methods of adaptation.

In chapter three, a Spanish language recognizer will be developed using HTK toolkit. This implementation increases the understanding of different steps of ASR. First different feature extraction methods which are supported by HTK, are used and the recognition results are compared. Then, HMM mono-phone models as well as triphone single Gaussian models (SGM) and Gaussian mixture models (GMM) are built in acoustic modeling stage. Another recognition task on word grammar test data will be performed to compare these models. At the next step, language models will be built to be used together with HMM models, to perform discriminative training. Finally at the decoding stage, results of recognition will be compared for different models.

Chapter four suggests two open research lines according to the studies of different sources in the field of ASR. The first one is about robust speech recognition using microphone arrays for the noise speaker problem. The next one is about a recent introduced approach for pattern based feature extraction and parts-based models for feature classification.

Chapter five is the conclusion of these thesis.

Chapter Summary

In this chapter, the significance and capabilities of ASR has been introduced. The main techniques of speech recognition has been briefly described. Also, the recent interests in ASR problem was discussed. Then, we have described the thesis objectives and the other chapters content shortly.

2 State of the art Automatic Speech Recognition

In this chapter, Standard architecture of state of the art Automatic Speech Recognizers (ASR) will be described. Although in different works, many different techniques have been used, most of them are based on a basic method. They are almost, variations or extensions for target applications or improvement purposes. Furthermore, the most important current challenges of ASR systems associated to their applications and limitations will be discussed.

2.1 Automatic Speech Recognition

ASR systems can be considered as pattern classification systems [1]. In these systems, a model is built for each unit of sound like words, phoneme, etc. Using these models, the system attempts to estimate the correct sequence of patterns, to change the speech sound into text. Today, the most widely used speech recognizers are based on Hidden Markov Models (HMM). The principal components of a large vocabulary continuous speech recognizer are illustrated in Figure 2.1 .

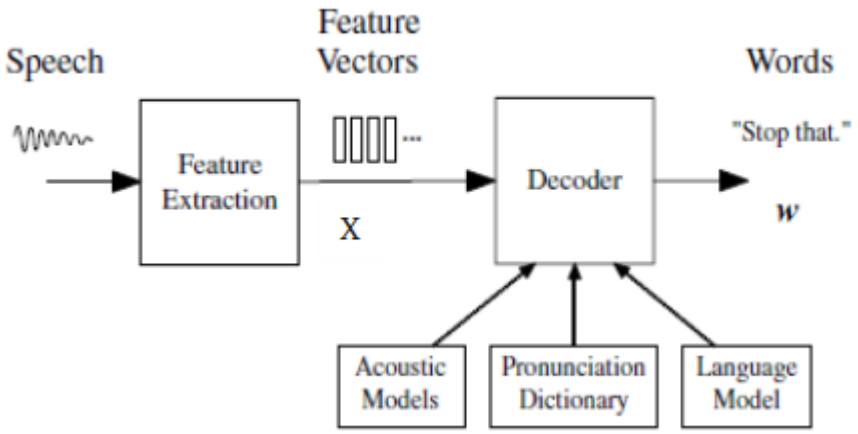


Figure 2.1: Architecture of a HMM-based Recognizer

Speech recognition is started with framing the discrete audio signal at a high rate like 40 f/Sec to obtain a number of overlapped frames. Then, speech feature vectors are extracted from the samples of each frame. If $X_{1:T} = \{x_1, x_2, ...x_T\}$ be considered as a sequence of extracted features from which, we would like to hypothesize a sequence of words, $W_{1:L} = \{w_1, w_2, ...w_M\}$. The desired output,$\hat{W}$, is the word string with the highest probability, given the features. This is the maximum a posteriori (MAP) decision that the decoder tries to find:

$$\hat{W} = \arg\max_{w} P(W|X) \tag{2.1}$$

However as its difficult to model and calculate this probability, in most cases Bayes rule is used to transform this equation into a more manageable equation:

$$\hat{W} = \arg\max_{w} \frac{P(X|W)P(W)}{P(X)} = \arg\max_{w} P(X|W)P(W) \tag{2.2}$$

Note that the denominator of this equation $P(X)$ is not considered as it has no effect in maximization. The likelihood $P(X|W)$ is determined by an *acoustic model* which models how the acoustic measurements,X , relate to the word string, W. The prior $P(W)$ is determined by a *language model*. [1] Thus, the process of recognizing an utterance of speech can be divided into two main stages, *feature extraction*, where a speech signal is parametrized into a sequence of feature vectors, and *decoding*, in which the most likely word sequence is hypothesized based on the observed features, using acoustic model and language model as shown in 2.1. In the following sections, we will introduce these components in more details.

2.1.1 Feature Extraction

The input audio signal to an ASR system is usually a stream of waveform samples with the rate of 16,000 samples per second. The initial stage of feature extraction changes the signal using signal processing methods so that, it could be statistically modeled. A good representation should reduce the dimensionality, preserve relevant information, and put it in a form well suited to the statistical modeling techniques (e.g., decorrelating the features)[2]. Mel-frequency cepstral coefficients (MFCC) have become a standard method for feature extraction in ASR. Several steps to achieve MFCC features are shown in Figure 2.2. They are also explained in the following items.

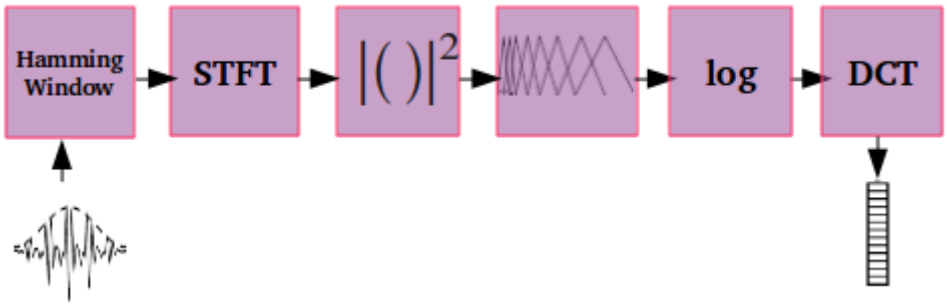


Figure 2.2: Flow chart depicting generation of mel-frequency cepstral coefficients from a frame of speech.

[1] In practice, the acoustic model is not normalized and the language model is often scaled by an empirically determined constant and a word insertion penalty is added i.e., in the log domain the total likelihood is calculated as $\log P(X|w) + \alpha \log(P(w)) + \beta |w|$ where α is typically in the range 8–20 and β is typically in the range 0 – (−20).

1. Framing. 25ms hamming windows placed every 10ms are applied to the digital signal. Each window will be referred to as a frame.
2. STFT. The frame is windowed and then transformed to the frequency domain using a Short-Time Fourier Transform (STFT) which result in $DFTY_{\hat{n}}[k]$ for analysis time $\hat{n}$. Then, the magnitude squared of the STFT is computed.
3. Mel Filter bank. Each frame's spectrum from previous step is multiplied to a bank of triangular filters. These filters are overlapping triangular weighting functions called *mel filters* (see Figure 2.3). They are equally distributed along the mel frequency scale with a 50% overlap between consecutive triangles. They are spaced in frequency approximately linearly at low frequencies up to the frequency of 1000 Hz and logarithmically at higher frequencies. This reduces the frequency resolution and warps frequencies to be spaced logarithmically (according to the Mel scale). The mel spectrum of the frame is computed as a vector whose components represent the energy in each of the mel Filters. To approximate human auditory processing more closely, the natural logarithm of each of the elements in the mel spectral vector is then computed, producing the log mel spectrum of the frame, resulting vector (roughly 40 dimensions) which in frequency domain is referred to as a set of Mel-frequency spectral coefficients (MFSCs). (The time series of MFSCs will be referred to as a Mel-spectrogram.)

$$MF_{\hat{n}}[r] = \frac{1}{A_r} \sum_{K=L_r}^{U_r} |V_r[K]\, Y_{\hat{n}}[K]\,|^2 \tag{2.3}$$

$$A_r = \sum_{K=L_r}^{U_r} |V_r[K]\,|^2 \tag{2.4}$$

where $V_r[K]$ is the triangular weighting function for the rth filter ranging from DFT index L_r to U_r, and A_r is a normalizing factor for the rth mel-filter. This normalization is built into the weighting functions of Figure 2.3. It is needed so that a perfectly flat input Fourier spectrum will produce a flat mel-spectrum.

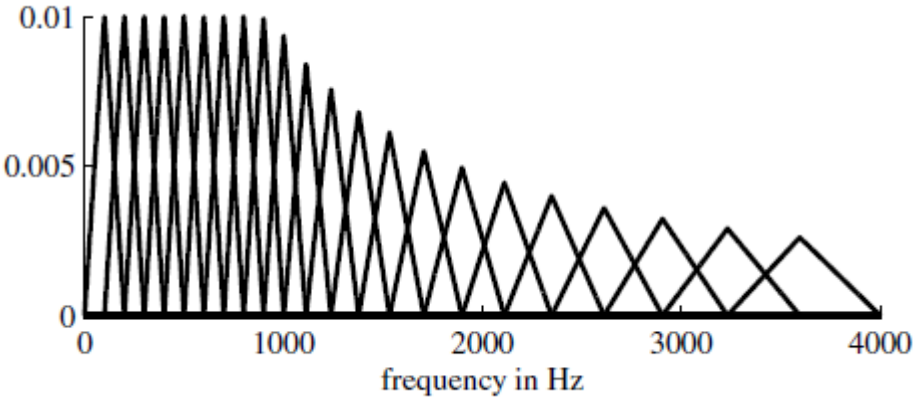


Figure 2.3: Weighting functions for Mel-frequency filter bank.

4. DCT. In each frame, a discrete cosine transform (DCT) of the MFSCs is taken. Typically, $mfcc_{\hat{n}}[m]$ is evaluated for a number of coefficients, *Nmfcc*, that is less than the number of

mel-filters, e.g., *Nmfcc* = 13 and R = 22. This helps to decorrelate the features, reduce dimensionality, and effectively keep only the smooth information from the spectral profile.

$$mfcc_{\hat{n}}[m] = \frac{1}{R}\sum_{r=1}^{R} log(MF_{\hat{n}}[r])\cos\left(\frac{2\pi}{R}\left(r+\frac{1}{2}\right)m\right) \tag{2.5}$$

5. Delta Features. The computation of MFCCs is completed after the DCT, but typically the final feature vector also includes some information from a larger time span than single frame. This is often done with delta features: time derivatives which measure the change in feature values. One typical way [3] to define delta features for a feature time series, $x[n]$, is,

$$d_N\{x\}[n] = \frac{\sum_{\tau=1}^{N}\tau\left(x[n+\tau]-x[n-\tau]\right)}{2\sum_{\tau=1}^{N}\tau^2} \tag{2.6}$$

Where N controls the amount of neighboring context. A common feature choice is that of MFCC+△+△△: MFCCs with first and second derivatives stacked at each frame. For example, 13 dimensional MFCCs with derivatives computed over ±1 frame $(\tau = 1)$ will result in a final 39 dimensional representation of (ignoring constant scaling factors)

$$\begin{aligned}[x, d_1\{x\}, d_1\{d_1\{x\}\}] \quad = \quad & [x[n], \\ & x[n+1]-x[n-1], \\ & 2x[n+2]+x[n+1]-x[n-1]-2x[n-2]]\end{aligned} \tag{2.7}$$

A popular alternative to MFCCs are so-called perceptual linear prediction (PLP) coefficients. The PLP technique attempts to model several aspects of human perception to create an auditory spectral representation which is then modeled with an all-pole filter [3]. Both MFCCs and PLPs are based on a short-time Fourier Transform (STFT) and can be considered alternative ways to derive a low-dimensional representation of the short-time spectral shape.

2.1.2 Acoustic Modeling

The feature vectors, extracted from training data, are used to build a HMM model for each sound unit (phoneme or word). Then, these models are used in recognition phase using the feature vectors of test data, for recognition task. Figure 2.4 represents a 5 state HMM model.

An HMM can be characterized by the following[4]:

- a finite number of states
- a state-transition probability distribution which specify the probability of making a transition from state i to state j at each frame, thereby defining the time sequence of the feature vectors over the duration of the word.
- an output probability distribution function associated with each state

The HMM shown in Figure 2.4, has five states. As shown by the solid arrows, the only allowable transitions in this HMM are back to the current state or to the state immediately to the right. All other state transitions have the probability of zero. The initial and final states are non-emitting states. In these states, no observations (feature vectors) are generated as there are no probability distributions associated with these states. The final state is also an absorbing state, in that when this state is reached, no further transitions are permitted.

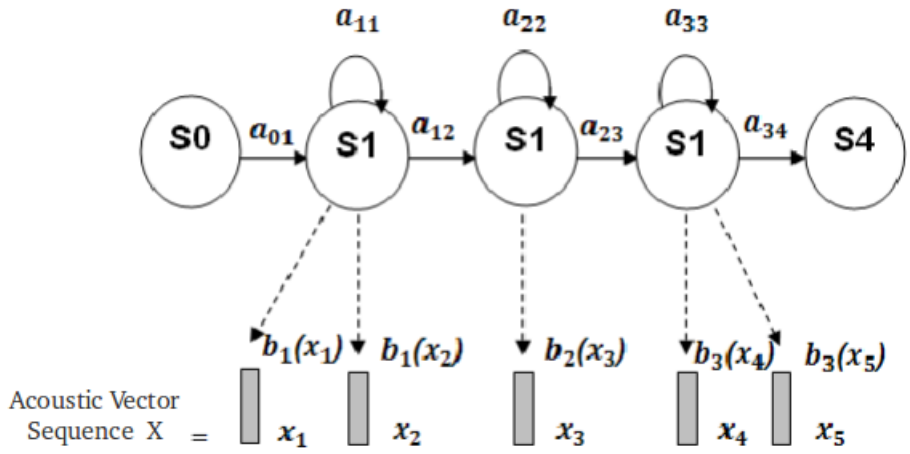


Figure 2.4: An example of a 5-state left-to-right HMM. The initial and final states are non-emitting. No observations are associated with these states.

2.1.2.1 Training HMM models

In order to train the HMM (i.e., learn the optimal model parameters) for each word (or sub-word) unit, a labeled training set of sentences (transcribed into words and sub-word units) is used to guide an efficient training procedure known as the *Baum–Welch* algorithm [1]. This algorithm aligns each of the various words (or sub-word units) with the spoken inputs and then estimates the appropriate means, covariances and mixture gains for the distributions in each model state.

Considering a sequence of M word models, $W = \{w_1, w_2, ..., w_M\}$ with a sequence of feature vectors, $X = x_1, x_2, ..., x_T$. The resulting alignment procedure is illustrated in Figure2.5. The sequence of feature vectors are represented along the horizontal axis and the concatenated sequence of word states along the vertical axis. An optimal alignment procedure determines the exact best matching sequence between word model states and feature vectors such that the first feature vector, x_1, aligns with the first state in the first word model, and the last feature vector, x_T , aligns with the last state in the Mth word model. The procedure for obtaining the best alignment between feature vectors and model states is based on either using the *Baum–Welch* statistical alignment procedure (in which we evaluate the probability of every alignment path and add them up to determine the probability of the word string), or a *Viterbi alignment* procedure for which the single best alignment path is determined and then the probability score along that path is used as the probability measure for the current word string.

2.1.2.2 Calculating the Acoustic Likelihood

In recognition phase, the function of the trained acoustic model is to assign probabilities to the acoustic realizations of a sequence of words, given the observed acoustic vectors. For instance, we need to compute the probability that the acoustic vector sequence $X_{1:T} = \{x_1, x_2, ...x_T\}$ came from the word sequence $W_{1:L} = \{w_1, w_2, ...w_M\}$ (assuming each word is represented as an HMM) and perform this computation for all possible word sequences. This requires to use the statistical behavior of the corresponding HMM model which is dependent on its state transition probabilities and the output distributions of its constituent states.

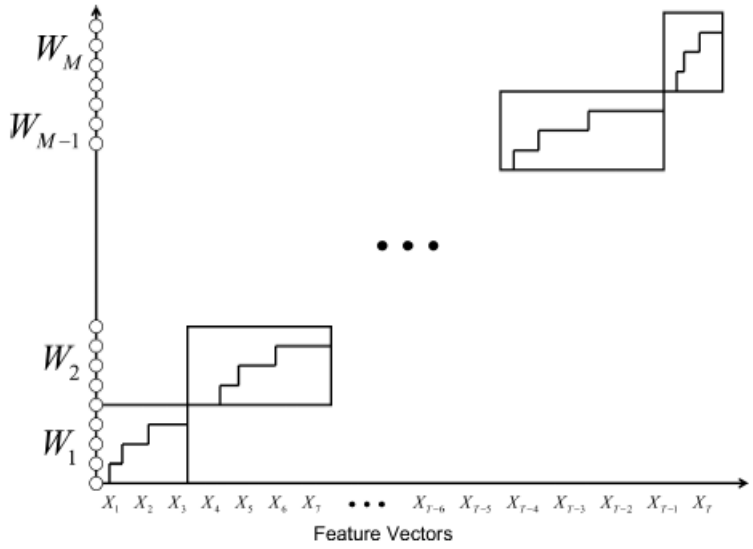


Figure 2.5: Alignment of concatenated HMM word models with acoustic feature vectors based on either a Baum–Welch or Viterbi alignment procedure for training HMMs.

The above probability is the acoustic model in Equation2.2. It involves assigning individual speech frames to the appropriate word model in an utterance which is based on an optimal alignment process between the concatenated sequence of word models and the sequence of feature vectors of the spoken input utterance being recognized.

This alignment process is illustrated in Figure 2.6 which shows the set of T feature vectors (frames) along the horizontal axis, and the set of M words (and word model states) along the vertical axis. The optimal segmentation of these feature vectors (frames) into the M words is shown by the sequence of boxes, each of which corresponds to one of the words in the utterance and its set of optimally matching feature vectors. The probability density of each state, and for each word, is learned during a training phase of the recognizer.

We will calculate the acoustic likelihood probability for one word, w with the feature vector of $X_w = \{x_1, ...x_m\}$. In the HMM model of the word w (HMM_w), the transition probabilities are represented by a transition matrix, A_w. The elements of this matrix, $a_w(i, j)$ represent the probability of transiting to state j at time $t+1$ given that state i is occupied at time t. Usually the self-transitions, a_{ii} are large (close to 1.0), and the jump transitions, $a_{12}, a_{23}, a_{34}, a_{45}$, in the model, are small (close to 0). If the HMM for word w has N states, then the summation of all transition probabilities of each state which corresponds to one row of A_w will be:

$$\sum_{j=1}^{N} a_w(i, j) = 1 \tag{2.8}$$

In speech recognition the state output probability distribution functions are usually modeled as Gaussians or mixtures of Gaussians. This is because, the observed feature vectors are not symmetric and uni-modal. In practice, depending on the speaker, accent and gender differences the data is asymmetric and multi-modal and its distribution can be best modeled by a mixture of Gaussians. Typically, in order to improve computational efficiency, the Gaussians are assumed to have diagonal covariance matrices. Thus, the output probability of a feature vector x belonging to the state i of an HMM for

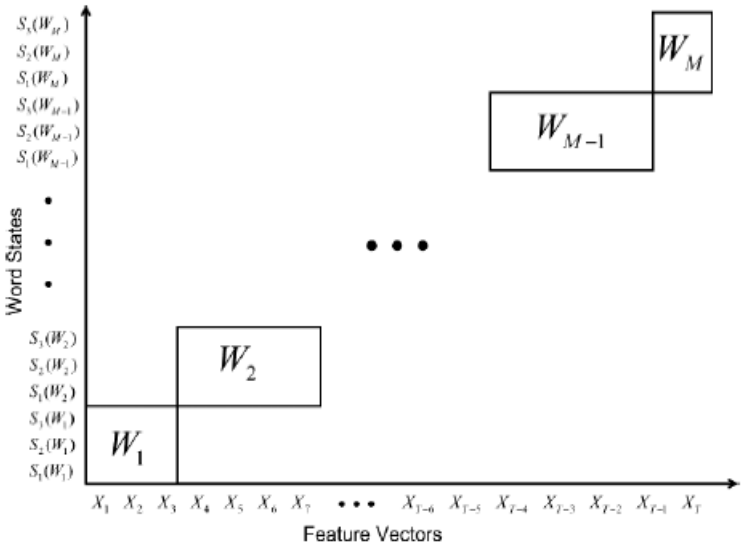


Figure 2.6: Illustration of time alignment process between unknown utterance feature vectors and set of M concatenated word models.

word w, is represented as,

$$b_w(x,i) = \sum_k \alpha_{i_k}^w N\left(x; \mu_{i_k}^w, \Sigma_{i_k}^w\right) \tag{2.9}$$

where $\alpha_{i_k}^w$, $\mu_{i_k}^w$ and $\Sigma_{i_k}^w$ are the mixture weight, mean vector and covariance matrix associated with the kth Gaussian in the mixture density of state i of the HMM of word w (HMM_w). We define B_w as the set of parameters $\left\{\alpha_{i_k}^w, \mu_{i_k}^w, \Sigma_{i_k}^w\right\}$ for all mixture components for all states in (HMM_w). We can then define $\lambda_w = (A_w, B_w)$ as the complete set of statistical parameters that define the (HMM_w).
Given the (HMM_w), with the feature vector set X_w, we denote $S = \{s_1, s_2, ..., s_N\}$ as the set of all possible state sequences of length N , then the acoustic likelihood is given by:

$$P(X_w|w) = \sum_{s \in S} P(X_w, s|w) = \sum_{s \in S} P(X_w|s)\, P(s|w) \tag{2.10}$$

The expression $P(s|w)$ represents the probability of a particular state sequence and is computed from the state transition matrix A_w. The expression $P(X_w|s)$ represents the probability of a particular sequence of feature vectors given a state sequence, and is computed from the state output probability distributions using Equation 2.9. Thus, we can rewrite Equation 2.10 as:

$$P(X_w|w) = \sum_{s \in S} \left(\prod_{t=1}^{N} a_w(s_t, s_{t+1}) \right) \left(\prod_{t=1}^{N} b_w(X_t, s_t) \right) \tag{2.11}$$

where X_t refers to the features vector set associated to each state $(X_t \in X_w)$. The acoustic model parameters $\lambda_w = (A_w, B_w)$ can be efficiently estimated from a corpus of training utterances using the *forward–backward algorithm* (also called Baum–Welch algorithm) which is an example of *expectation maximization (EM)*.

Commonly, the acoustic models are built for phonemes instead of words and all speech utterances are represented by concatenating a sequence of phone models together. The major problem with this is

that decomposing each vocabulary word into a sequence of context-independent base phones fails to capture the very large degree of context-dependent variation that exists in real speech.

A simple way to mitigate this problem is to use a unique phone model for every possible pair of left and right neighbors. The resulting models are called *triphones* [5] and if there are N base phones, there are N^3 potential triphones. To reduce this, the complete set of logical triphones L can be mapped to a reduced set of physical models P by clustering and tying together the parameters in each cluster. This mapping process is illustrated in Figure 2.7 and the parameter tying is illustrated in Figure 2.8 where the notation x − q + y denotes the triphone corresponding to phone q spoken in the context of a preceding phone x and a following phone y. Each base phone pronunciation q is derived by simple look-up from the pronunciation dictionary, these are then mapped to logical phones according to the context, finally the logical phones are mapped to physical models.

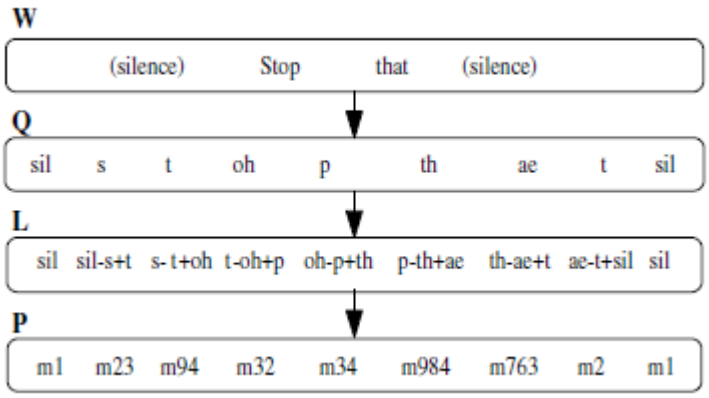


Figure 2.7: Context dependent phone modeling.

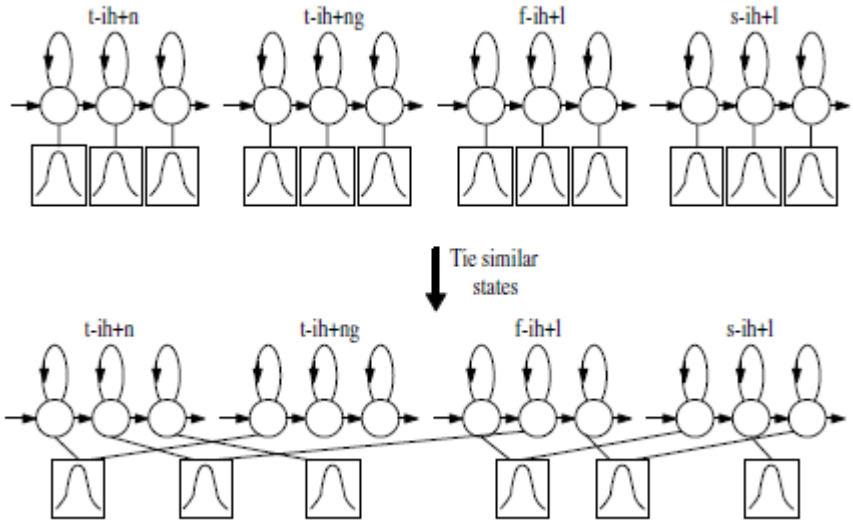


Figure 2.8: Formation of tied-state phone models.

The choice of which states to tie is commonly made using decision trees. Figure 2.9 illustrates this

tree-based clustering. In the figure, the logical phones s-aw+n and t-aw+n will both be assigned to leaf node 3 and hence they will share the same central state of the representative physical model.

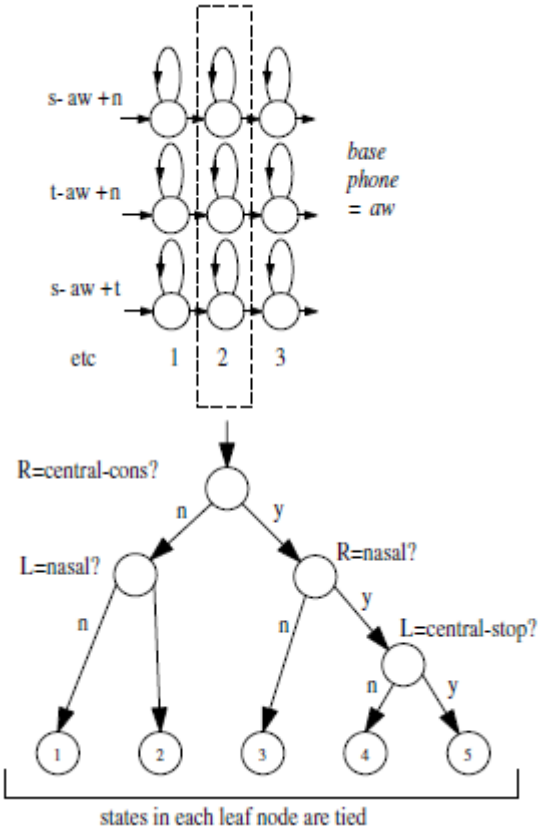


Figure 2.9: Decision tree clustering.

2.1.3 Language Modeling

In Equation 2.2, it is mentioned that the term $P(W)$ is the language model (LM). In order to build a LM, a very large training set of text data is required. If we consider the word sequence $W = \{w_1, w_2, ..., w_M\}$ in Equation 2.2, the prior probability is given by:

$$P(W) = \prod_{i=1}^{M} P(w_i | w_{i-1}, ..., w_1) \tag{2.12}$$

For large vocabulary recognition, the conditioning word history in Equation 2.12 is usually truncated to $N-1$ words to form an N-gram language model

$$P(W) = \prod_{i=1}^{M} P(w_i | w_{i-1}, w_{i-2} ..., w_{i-N+1}) \tag{2.13}$$

where N is typically in the range 2–4. An N-gram LM is used to predict each symbol in the sequence given its $n-1$ predecessors. It is built on the assumption that the probability of a specific N-gram occurring in some unknown test text can be estimated from the frequency of its occurrence in some given training text [3].

2.1.3.1 Perplexity

A speaker emitting language can be considered to be a discrete information source which is generating a sequence of words $W = \{w_1, w_2, ..., w_M\}$ from a vocabulary set, $\mathscr{L}$. The probability of a symbol w_i is dependent upon the previous symbols $w_1, .., w_{i-1}$. The information source's inherent per-word entropy H represents the amount of non-redundant information provided by each new word on average, defined in bits as:

$$H = -\lim_{M\to\infty}\frac{1}{M}\sum_{w_1,w_2,...,w_M}(P(w_1,...w_M))\log_2(P(w_1,...w_M)) \tag{2.14}$$

$$\approx -\lim_{M\to\infty}\frac{1}{M}\log_2(P(w_1,w_2,...,w_M)) \tag{2.15}$$

As this summation is over all possible sequences of words, and the source is ergodic[2] then the summation over all possible word sequences has been discarded. It is reasonable to assume ergodicity on the basis that we can disambiguate words on the basis of only the recent parts of a conversation or piece of text. Having assumed this ergodic property, it follows that given a large enough value of M, H can be approximated with:

$$H = -\frac{1}{M}\log_2(P(w_1,w_2,...,w_M)) \tag{2.16}$$

LMs are often assessed in terms of their *perplexity*. In [5], H is defined as the perplexity of LM. But in [3], it is defined as a measure related to entropy, H to assess the actual performance of a language model and is defined such that:

$$PP = 2^{\hat{H}} = \hat{P}(w_1,w_2,...,w_M)^{\frac{-1}{M}} \tag{2.17}$$

Perplexity can be considered to be a measure of on average how many different equally most probable words can follow any given word. Lower perplexities represent better language models, although this simply means that they 'model language better', rather than necessarily work better in speech recognition systems – perplexity is only loosely correlated with performance in a speech recognition system since it has no ability to note the relevance of acoustically similar or dissimilar words [3].

2.1.3.2 Robust Model Estimation

Even to have a large training set to calculate N-grams for LM, there are word sequences which were not observed in the training text, but, cannot be assumed to represent impossible sequences. Therefore, some probability mass must be reserved for these unseen sequences.

Estimates of probabilities in N-gram models are commonly based on maximum likelihood (ML) estimates. This is done by counting events in context on some given training text. For example, if $C(w_{i-2}w_{i-1}w_i)$ represent the number of occurrences of the three words $w_{i-2}w_{i-1}w_i$ and similarly for $C(w_{i-2}w_{i-1})$, then

$$P(w_i|w_{i-1}w_{i-2}) \approx \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})} \tag{2.18}$$

[2]Ergodic theory describes the behavior of a dynamic system when it is allowed to run for a long time. The system that evolves for a long time, ¨forgets¨ its initial state.

As mentioned above, the major problem with this simple ML estimation scheme is data sparsity and unobserved events. This can be mitigated by a combination of *discounting* and *backing-off*.

$$P(w_i|w_{i-1}w_{i-2}) = \begin{cases} d\frac{C(w_{i-2},w_{i-1},w_i)}{C(w_{k-2},w_{k-1})} & if\, 0 < C \leqslant C' \\ \frac{C(w_{i-2},w_{i-1},w_i)}{C(w_{k-2},w_{k-1})} & if\, C > C' \\ \alpha(w_{i-2},w_{i-1})P(w_i|w_{i-1}) & if\, C = 0 \end{cases} \tag{2.19}$$

where C' is a count threshold which is called *cut-off*, C is short-hand for $C(w_{i-2}w_{i-1}w_i)$, d is a discount coefficient and α is a normalization constant. Thus, when the N-gram count exceeds the cut-off threshold, the ML estimate is used. When the count is small the same ML estimate is used but discounted slightly. The discounted probability mass is then distributed to the unseen N-grams which are approximated by a weighted version of the corresponding bi-gram. This idea can be applied recursively to estimate any sparse N-gram in terms of a set of back-off weights and (N −1)-grams.

2.1.3.3 Class N-gram Models

An alternative approach to robust language model estimation is to use *class-based* models in which for every word w_i there is a corresponding class c_i. Then,

$$P(W) = \prod_{i=1}^{M} (P(w_i|c_i)P(c_i|c_{i-1}, c_{i-2}..., c_{i-N+1})) \tag{2.20}$$

Where M is the number of classes. As for word-based models, the class N-gram probabilities are estimated using ML but, since there are far fewer classes (typically a few hundred) data sparsity is much less of an issue. The classes themselves are chosen to optimize the likelihood of the training set assuming a bi-gram class model. It can be shown that when a word is moved from one class to another, the change in perplexity depends only on the counts of a relatively small number of bi-grams. Hence, an iterative algorithm can be implemented which repeatedly scans through the vocabulary, testing each word to see if moving it to some other class would increase the likelihood [5].

Although the basic principle of an N-gram LM is very simple, in practice there are usually many more potential N-grams than can ever be collected in a training text in sufficient numbers to yield robust frequency estimates. Furthermore, for any real application such as speech recognition, the use of an essentially static and finite training text makes it difficult to generate a single LM which is well-matched to varying test material. For example, an LM trained on newspaper text would be a good predictor for dictating news reports but the same LM would be a poor predictor for personal letters or a spoken interface to a flight reservation system. A final difficulty is that the vocabulary of an N-gram LM is finite and fixed at construction time. Thus, if the LM is word-based, it can only predict words within its vocabulary and furthermore new words cannot be added without rebuilding the LM [3].

2.1.4 Decoding

The problem of decoding is to calculate the maximum probability that a sequence of feature vectors X_w (from input audio signal) correspond to the word $\hat{w}$, in a higher level, the same problem exists for a sequence of words or in lower level for a sequence of phonemes. Considering the knowledge of acoustic models and language models, this problem has been formulated in 2.1 and as follows:

$$\hat{w} = \arg\max_w P(w|X_w) = arg\max_w P(X_w|w)P(w)$$

By substitution of the acoustic likelihood from Equation 2.11 we have:

$$\hat{w} = \arg\max_{w} \left\{ P(w) \sum_{s \in S} \left(\prod_{t=1}^{N} a_w(s_t, s_{t+1}) \right) \left(\prod_{t=1}^{N} b_w(X_t, s_t) \right) \right\} \tag{2.21}$$

However, for computational efficiency, most HMM speech recognition systems estimate the best state sequence, i.e. the state sequence with the highest likelihood, associated with the estimated hypothesis. Thus, recognition is actually performed as:

$$\hat{w} = \arg\max_{w, s \in S} \left\{ P(w) \left(\prod_{t=1}^{N} a_w(s_t, s_{t+1}) \right) \left(\prod_{t=1}^{N} b_w(X_t, s_t) \right) \right\} \tag{2.22}$$

While this is for single words recognition, the HMM framework can easily be expanded to model strings of words, $W = \{w_1, w_2, ..., w_M\}$ by concatenating the HMMs of the constituent words. An example of this is shown in Figure 2.10 for an utterance composed of three words. On the other hand, in most cases, the phonemes are modeled individually. In this case, however, recognition becomes significantly more computationally demanding (for large vocabularies and high average word branching factor language models) and in fact impractical, because Equation 2.22 would have to be evaluated for every possible phone sequence to build a word and then sequence of words etc. As a result, the Viterbi algorithm, is used to obtain a locally optimal estimate.

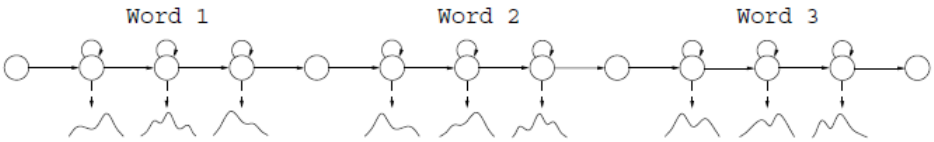


Figure 2.10: An HMM for a sequence of words can be built from the individual HMMs of its constituent words.

For solving the Equation 2.22 in higher levels, its more efficient to generate not just the most likely hypothesis but the N-best set of hypotheses. This is extremely useful since it allows multiple passes over the data without the computational expense of repeatedly solving 2.22. A compact and efficient structure for storing these hypotheses is the *word lattice*. A word lattice consists of a set of nodes representing points in time and a set of spanning arcs representing word hypotheses. An example is shown in Figure 2.11 part (a). In addition to the word IDs shown in the figure, each arc can also carry score information such as the acoustic and language model scores.

Lattices can also be compacted into a very efficient representation called a *confusion network*. This is illustrated in Figure 2.11 part (b) where the "-" arc labels indicate null transitions. In a confusion network, the nodes no longer correspond to discrete points in time, instead they simply enforce word sequence constraints. Thus, parallel arcs in the confusion network do not necessarily correspond to the same acoustic segment. However, it is assumed that most of the time the overlap is sufficient to enable parallel arcs to be regarded as competing hypotheses. A confusion network has the property that for every path through the original lattice, there exists a corresponding path through the confusion network. Each arc in the confusion network carries the posterior probability of the corresponding word w. This is computed by finding the link probability of w in the lattice using a forward–backward procedure, summing over all occurrences of w and then normalizing so that all competing word arcs in the confusion network sum to one.

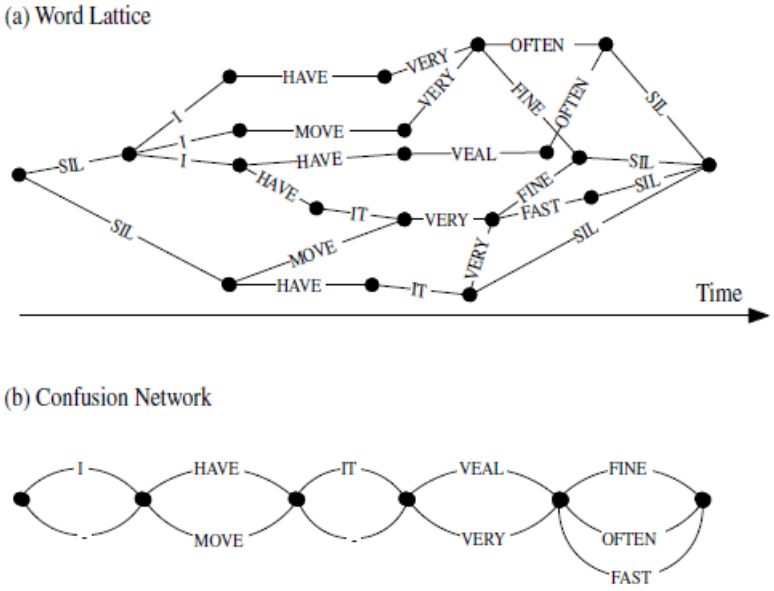


Figure 2.11: Example lattice and confusion network.

In practice, a direct implementation of the Viterbi algorithm becomes unmanageably complex for continuous speech where the topology of the models, the language model constraints and the need to bound the computation must all be taken into account [5]. N-gram language models and cross-word triphone contexts are particularly problematic since they greatly expand the search space. To deal with this, a number of different architectural approaches have evolved. For Viterbi decoding, the search space can either be constrained by maintaining multiple hypotheses in parallel or it can be expanded dynamically as the search progresses.

Alternatively, a completely different approach can be taken where the Viterbi algorithm is replaced by a depth-first search. This class of recognizers are called *stack decoders*. Stack decoding is a variant of tree search. While the Viterbi search finds the optimal state sequence, Stack decoding focuses on the optimal word sequence. The main idea is that, if some heuristics are available to guide the decoding, the search can be done in a depth-first fashion around the best path. This avoids wasting computation on unpromising paths via time synchronous decoding. However, such a heuristic function is very difficult to attain in speech recognition since it must combine elements of acoustic and language model scoring. Also, as its necessary to compare hypotheses of different lengths, the run-time search characteristics can be difficult to control.

Finally, through the use of methods from the field of Finite State Automata Theory, *Finite State Network* (FSN) methods have evolved that reduce the computational cost, thereby enabling exact maximum likelihood solutions in computationally feasible times, even for very large speech recognition problems. The basic concept of a finite state network transducer is illustrated in Figure 2.12 which shows a word pronunciation network for the word /data/ [1]. Each arc corresponds to a phoneme in the word pronunciation network, and the weight is an estimate of the probability that the arc is utilized in the pronunciation of the word in context. As its shown, for the word /data/ there are four total pronunciations, namely (along with their (estimated) pronunciation probabilities):

1. /D/ /EY/ /D/ /AX/ — probability of 0.32
2. /D/ /EY/ /T/ /AX/ — probability of 0.08

3. /D/ /AE/ /D/ /AX/ — probability of 0.48
4. /D/ /AE/ /T/ /AX/ — probability of 0.12.

The combined FSN of the 4 pronunciations is a lot more efficient than using 4 separate enumerations of the word since all the arcs are shared among the 4 pronunciations and the total computation for the full FSN for the word /data/ is close to $\frac{1}{4}$ the computation of the 4 variants of the same word.

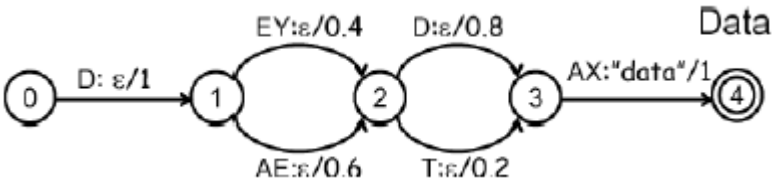


Figure 2.12: Word pronunciation transducer for four pronunciations of the word /data/.

Its possible to continue the process of creating efficient FSNs for each word in the task vocabulary (the speech dictionary or lexicon), and then combine word FSNs into sentence FSNs using the appropriate language model. Further, the process could be continued down to the level of HMM phones and HMM states, making the process even more efficient. Ultimately, a very large network of model states, model phones, model words, and even model phrases can be compiled into a much smaller network via the method of *weighted finite state transducers* (WFST), which combine the various representations of speech and language and optimize the resulting network to minimize the number of search states (and, equivalently, thereby minimize the amount of duplicate computation). A simple example of such a WFST network optimization is given in Figure 2.13.

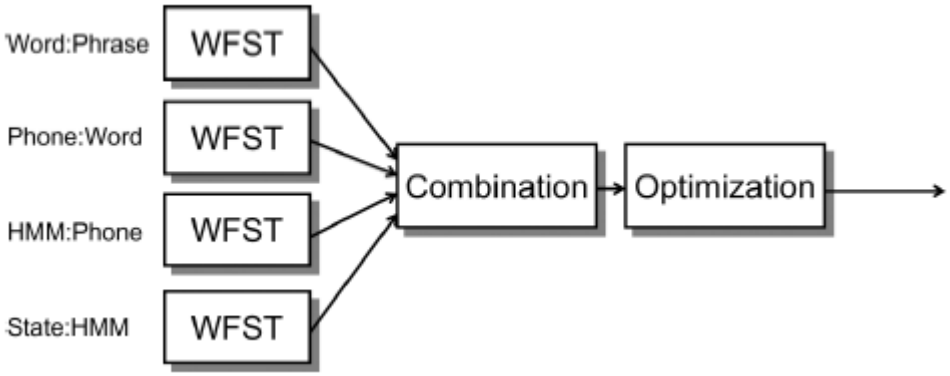


Figure 2.13: Use of WFSTs to compile a set of FSNs into a single optimized network to minimize redundancy in the network.

Using the techniques of network combination (which include network composition, determination, minimization, and weight pushing) and network optimization, the WFST uses a unified mathematical framework to efficiently compile a large network into a minimal representation that is readily searched using standard Viterbi decoding methods. Using these methods, an unoptimized network with 10^{22} states (the result of the cross product of model states, model phones, model words, and model phrases) is compiled down to a mathematically equivalent model with 10^8 states. This approach offers both flexibility and efficiency and is therefore extremely useful for both research and practical applications.

2.2 A Review of ASR Challenges

Many researches in the field of ASR and its application have been performed from more than 40 years ago. Although its difficult to study an categorize all these works, in this section, some most important challenges of ASR will be explained and different approaches in the past works will be reviewed briefly.

2.2.1 Improved ASR Systems for LVCR

Although the simple HMM models are appropriate for small vocabulary tasks, they do not work well in the case of more complex and large vocabulary applications such as transcribing broadcast news. In order to improve the over all performance of ASR systems in such cases, different methods have been proposed so that some of them have successfully made significant improvements and become a common method in recognition algorithms.

2.2.1.1 Feature Projection

In architecture of an HMM-Based Recognizer, dynamic first and second differential parameters, the so-called delta and delta–delta parameters, were added to the static feature parameters to overcome the limitations of the conditional independence assumption associated with HMMs. Furthermore, the DCT was assumed to approximately decorrelate the feature vector to improve the diagonal covariance approximation and reduce the dimensionality of the feature vector [5].

It is also possible to reduce the dimension and also decorrelate the features using some projection and transformation methods. For example, if we consider the source features vector as $\tilde{x_t}$ with in d dimension, it is possible to reduce the dimensionality to p and have x_t feature vector by applying a $p \times d$ linear transformation matrix $A_{[p]}$ as follows:

$$x_t = A_{[p]}\tilde{x_t} \tag{2.23}$$

It has been demonstrated that the best class label for $\tilde{x_t}$ is Gaussian component labels and using the class labels in a supervised fashion yields better projections and discrimination between classes but is computationally high cost than unsupervised methods that just used the general attributes of observations such as their variances.

The choice of the transformation criteria in Equation 2.23 is a key point. The simplest criteria is *principal component analysis* (PCA) which is an unsupervised projection. It is based on finding the p rows of the orthonormal matrix A that maximizes:

$$F_{pca}(\lambda) = \log\left(|A_{[p]}\tilde{\sum}_g A_{[p]}^T|\right) \tag{2.24}$$

where $\tilde{\sum}_g$ is the total, or global, covariance matrix of the original data, $\tilde{x_t}$. This selects the orthogonal projections of the data that maximizes the total variance in the projected subspace. But, selecting subspaces that yield large variances does not necessarily yield subspaces that discriminate between the classes.

The *linear discriminant analysis* (LDA) is a supervise approach can be used as a solution in this case. In LDA the objective is to increase the ratio of the between class variance to the average within class variance for each dimension. This criterion may be expressed as:

$$F_{lda}(\lambda) = \log\left(\frac{|A_{[p]}\tilde{\Sigma}_b A_{[p]}^T|}{|A_{[p]}\tilde{\Sigma}_w A_{[p]}^T|}\right) \tag{2.25}$$

where $\tilde{\Sigma}_b$ is the between-class covariance matrix and $\tilde{\Sigma}_w$ the average within-class covariance matrix where each distinct Gaussian component is usually assumed to be a separate class. This criterion yields an orthonormal transform such that the average within-class covariance matrix is , which should improve the diagonal covariance matrix assumption.

The alternative refinements of LDA are heteroscedastic discriminant analysis (HDA) and heteroscedastic LDA (HLDA). While, LDA projections are best suited to classifier models where class distributions have equal variance, it is not the optimal transform when the class distributions are heteroscedastic. This is because LDA assigns the same average within-class covariance $\tilde{\Sigma}_w$ to all classes [6].

By using the actual class covariance matrices rather than using the averages, HDA solves this limitation. But , as the transformation matrix A is not constrained to be orthonormal, and does not perform the decorrelating of the data associated with each Gaussian component, a separate decorrelating transform must be added. In contrast, HLDA yields the best projection whilst simultaneously generating the best transform for improving the diagonal covariance matrix approximation. But, the level of computation in HLDA is high compared to other approaches.

2.2.1.2 Discriminative Criteria for Parameter Estimation

In the HMM-based recognizer model introduced in 2.1.2, the estimation of HMM model parameters, λ , was performed based on the maximum likelihood. The idea is to build the HMM models that are most likely to generate the same training data. In other words, as much as this probability increases, we will obtain more accurate models.

Given a sequence of features of training data $x^{(1)}, x^{(2)}, ...x^{(R)}$, the maximum likelihood (ML) training criteria is expressed as:

$$F_{ml} = \frac{1}{R}\sum_{r=1}^{R} \log\left(P\left(x^{(r)}|w_{ref}^{(r)};\lambda\right)\right) \tag{2.26}$$

Where, $x^{(r)}$ is the r th training utterance feature with the corresponding transcription $w_{ref}^{(r)}$. This optimization criteria is calculated using *EM* algorithm. The popularity of MLE is due to its ability to produce accurate systems that can be quickly trained using the globally convergent Baum-Welch algorithm. There are some assumptions that MLE algorithm performance is based on their validity. According to these assumption, observations are from a known family of distributions (typically Gaussian), training data is infinite, and the true language model is known [7]. But as none of these assumptions in practice are true, alternative discriminative criteria have been developed. In these approaches, the goal is not to estimate the model parameters so that, most likely they can generate the training data. Instead, the objective is to modify the model parameters so that hypotheses generated by the recognizer on the training data more closely "match" the correct word-sequences, whilst generalizing to unseen test data.

For speech recognition, three main forms of discriminative training have been examined, all of which can be expressed in terms of the posterior of the correct sentence [5]:

1. **Maximum Mutual Information(MMI)**
   In MMI, the goal is to maximize the mutual information between the word-sequence, w, and

the information extracted by a recognizer with parameters λ from the associated observation sequence, x , $I(w,x;\lambda)$. As the joint distribution of the word-sequences and observations is unknown, it is approximated by the empirical distributions over the training data. This can be expressed as [8]:

$$I(w,x;\lambda) = \frac{1}{R}\sum_{r=1}^{R} \log\left(\frac{P\left(w_{ref}^{(r)}, x^{(r)};\lambda\right)}{P\left(w_{ref}^{(r)}\right)P\left(x^{(r)};\lambda\right)}\right) \tag{2.27}$$

As the $P\left(w_{ref}^{(r)}\right)$is fixed, in order to find the model parameters to maximize the average log-posterior probability of the correct word sequence, the following criterion should be maximized:

$$F_{mmi}(\lambda) = \frac{1}{R}\sum_{r=1}^{R} \log\left(P\left(w_{ref}^{(r)}|x^{(r)};\lambda\right)\right) \tag{2.28}$$

$$= \frac{1}{R}\sum_{r=1}^{R} \log\left(\frac{P\left(x^{(r)}|w_{ref}^{(r)};\lambda\right)P\left(w_{ref}^{(r)}\right)}{\sum_w P\left(x^{(r)}|w;\lambda\right)P(w)}\right) \tag{2.29}$$

To maximize the Equation 2.29, the numerator must be increased while the denominator be decreased. The first term in the numerator is identical to the objective function for MLE. Just like MLE, MMIE will try and maximize the likelihood of each observation given the training transcriptions. The difference in MMIE is the denominator term which can be made smaller by reducing the probabilities of other possible word sequences. Therefore, MMIE attempts to both make the correct hypothesis more probable, while at the same time making incorrect hypotheses less probable.
The most important limitation of MMIE is that it is computationally expensive to maximize objective function specially for large vocabulary sytems and it yields poor generalization to unseen data.

2. **Minimum Classification Error(MCE)**
MCE is a smooth measure of the error which is based on a smooth function of the difference between the log-likelihood of the correct word sequence and all other competing sequences, and a sigmoid is often used for this purpose. The MCE criterion may be expressed in terms of the posteriors as:

$$F_{mce}(\lambda) = \frac{1}{R}\sum_{r=1}^{R}\left(1+\left[\frac{P\left(w_{ref}^{(r)}|x^{(r)};\lambda\right)}{\sum_{w\neq w_{ref}^{(r)}} P\left(w|x^{(r)};\lambda\right)}\right]^{\rho}\right)^{-1} \tag{2.30}$$

There differences between MCE and MMI is that the denominator term does not include the correct word sequence, Also, the posteriors (or log-likelihoods) are smoothed with a sigmoid function, which introduces an additional smoothing term ρ. When $\rho = 1$ then:

$$F_{mce}(\lambda) = 1 - \frac{1}{R}\sum_{r=1}^{R} P\left(w_{ref}^{(r)}|x^{(r)};\lambda\right) \tag{2.31}$$

Both MMIE and MCE have been successfully applied to small vocabulary speech recognition tasks and it has been reported that MCE outperforms MMIE. There are also other methods like MWE and MPE, that outperform them for the most difficult large vocabulary tasks [9].

3. **Minimum Bayes' Risk (MBR)**
In MBR, The basic notion is the same as other discriminative objective functions such as MMI, which is training the acoustic parameters by forcing the acoustic model to recognize the training data correctly [10]. In MBR training, rather than trying to model the correct distribution, as in the MMI criterion, the expected loss during recognition is minimized [5]. To approximate the expected loss during recognition, the expected loss estimated on the training data is used:

$$F_{mbr}(\lambda) = \frac{1}{R}\sum_{r=1}^{R}\sum_{w} P\left(w|x^{(r)};\lambda\right) L\left(w, w_{ref}^{(r)}\right) \tag{2.32}$$

where, $L\left(w, w_{ref}^{(r)}\right)$ is the loss function of word sequence w against the reference for sequence r, $w_{ref}^{(r)}$. There are a number of loss functions that have been examined.

1/0 loss: For continuous speech recognition this is equivalent to a sentence-level loss function

$$L\left(w, w_{ref}^{(r)}\right) = \begin{cases} 1; & w \neq w_{ref}^{(r)} \\ 0; & w = w_{ref}^{(r)} \end{cases} \tag{2.33}$$

In Equation 2.30, when $\rho = 1$ MCE and MBR training with a sentence cost function are the same.

Word: The loss function directly related to minimizing the expected word error rate. The *minimum word error* (**MWE**) objective function attempts to minimize the number of word level errors. It is normally computed by minimizing the Levenshtein edit distance.[3]

Phone: For large vocabulary speech recognition not all word sequences will be observed. To assist generalization, the loss function is often computed between phone sequences, rather than word sequences. In the literature this is known as *minimum phone error* (**MPE**) training. While MWE was shown to give better training set performance, MPE results in better test set performance.

Phone frame error: When using the phone loss-function, the number of possible errors to be corrected is reduced compared to the number of frames. This can cause generalization issues. To address this *minimum phone frame error* (**MPFE**) may be used where the phone loss is weighted by the number of frames associated with each frame. This is the same as the Hamming distance described in [11].

2.2.1.2.1 Parameter Estimation In discriminative training, EM cannot be used to estimate parameters. To explain the reason, we consider the MMI criterion which can be expressed as the difference of two log-likelihood expressions:

$$F_{mmi}(\lambda) = \frac{1}{R}\sum_{r=1}^{R} \log\left(P\left(x^{(r)}|w_{ref}^{(r)};\lambda\right) P\left(w_{ref}^{(r)}\right)\right) - \log\left(\sum_{w} P\left(x^{(r)}|w;\lambda\right) P(w)\right) \tag{2.34}$$

The first term is referred to as the *numerator term* and the second, the *denominator term*. The numerator term is identical to the standard ML criterion (the language model term, $P\left(w_{ref}^{(r)}\right)$, does not

[3] the Levenshtein distance is a metric for measuring the amount of difference between two sequences. The term edit distance is often used to refer specifically to Levenshtein distance. The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being *insertion*, *deletion*, or *substitution* of a single character.

influence the ML-estimate). The denominator may also be rewritten in a similar fashion to the numerator by producing a composite HMM with parameters λ^{den} . Hence, although auxiliary functions may be computed for each of these, the difference of two lower-bounds is not itself a lower- bound and so standard EM cannot be used. To handle this problem, the extended Baum-Welch (EBW) criterion was proposed. In this case, standard EM-like auxiliary functions are defined for the numerator and denominator but stability during re-estimation is achieved by adding scaled current model parameters to the numerator statistics.

For large vocabulary speech recognition systems, where thousands of hours of training data may be used, it is important that the training procedure be as efficient as possible. For discriminative techniques one of the major costs is the accumulation of the statistics for the denominator since this computation is equivalent to recognizing the training data. To improve efficiency, it is common to use lattices as a compact representation of the most likely competing word sequences and only accumulate statistics at each iteration for paths in the lattice.

2.2.1.2.2 Generalization Compared to ML training, discriminative criteria tend to generalize less-well [5]. To mitigate this, a number of techniques have been developed that improve the robustness of the estimates. As a consequence of the conditional independence assumptions, the posterior probabilities computed using the HMM likelihoods tend to have a very large dynamic range and typically one of the hypotheses dominates. To address this problem, the acoustic model likelihoods are often raised to a fractional power, referred to as acoustic deweighting. Thus when accumulating the statistics, the posteriors are based on

$$P\left(w_{ref}|x^{(r)};\lambda\right) = \frac{P\left(x^{(r)}|w_{ref}^{(r)};\lambda\right)^{\alpha} P\left(w_{ref}^{(r)}\right)^{\beta}}{\sum_{w} P\left(x^{(r)}|w;\lambda\right)^{\alpha} P\left(w\right)^{\beta}} \tag{2.35}$$

In practice β is often set to one and α is set to the inverse of the language model scale-factor described in footnote 1 in 2.1. The form of the language model used in training should in theory match the form used for recognition. However, it has been found that using simpler models, uni-grams or heavily pruned bi-grams, for training despite using trigrams or four-grams in decoding improves performance. By weakening the language model, the number of possible confusions is increased allowing more complex models to be trained given a fixed quantity of training data. To improve generalization, "robust" parameter priors may be used when estimating the models.

2.2.2 Robust Speech Recognition

State-of-the-art ASR systems could work much better when the speech signals are captured in a noise-free environment using a close-talking microphone worn near the mouth of the speaker [12]. However, these systems performance may significantly degrade if different variability's have not been taken in to account. For example, many target applications for this technology do not take place in noise-free environments. Furthermore, it is often inconvenient for the speaker to wear a close-talking microphone.

Environmental noise is the main variable factor in the input speech. There are also other factors, such as distortion in the transmission channel, speaker variability, distant-talking interaction, etc. Moreover, in many applications of ASR systems, the user can not hold the microphone in an ideal mode. In this condition, the environmental (in general non stationary) noise, the reverberation effects, and a time-varying distance between the user and the microphone would introduce new dimensions

on which robustness is necessary. Therefore, when the ASR component is embedded in a spoken dialog system, a wide range of techniques and parameters are involved to make the recognition task robust.

Speech recognition systems achieve best performance when the models are trained and operated in matched environments. For most applications, this is impractical, as the operating environment varies with time and space, and therefore, some form of signal enhancement and noise compensation must be employed.

Many different methods and techniques have been so far explored in order to reduce the impact of noise and reverberation on ASR system performance. They could be categorized in three main approaches:

1. To clean or enhance the input signal
2. To derive robust acoustic features
3. To train robust statistical models

In the two latter cases, a reduced mismatch between training and test conditions is generally pursued by applying techniques aimed to "clean" the acoustic features (feature-based compensation) or to update the statistical models (model-based compensation and adaptation).

2.2.2.1 Input signal Enhancement

Signal enhancement before feature extraction can be accomplished by using a single microphone such as a traditional noise-canceling microphone or multi-microphones particularly microphone arrays.

In order to de-reverberate the speech signal with a single microphone, the microphone signal is filtered with the inverse of the transfer function of the path between speaker and microphone. As typical room impulse responses are non-minimum-phase, a causal, exact inverse filter is not stable and therefore can only be approximated [13].

Microphone arrays record the speech signal simultaneously over a number of spatially separated channels. Many array-signal-processing techniques have been developed to combine the signals in the array to achieve a substantial improvement in the signal-to-noise ratio (SNR) of the output signal. Microphone array-based speech recognition is performed in two independent stages: array processing and recognition. During Array-processing the captured waveforms is processed and enhanced, and the output waveforms are passed to the speech recognition system. It is implicitly assumed that the best recognition performance is obtained with the array processing methods which provide the best enhancement. The recognition systems just extract a set of features from the enhanced waveforms. As a result, improvements in the quality of the output waveform may not necessarily translate into improvements in the quality of the recognized features and, improvements in recognition performance.

Some most microphone array based approaches are:

- **Fixed Bean Forming** - Beam-forming is the most widely used array-processing method [14]. Beam-forming refers to any method that algorithmically (rather than physically) steers the sensors in the array toward a target signal. The direction the array is steered is called the "look direction". Beam-forming algorithms can either be fixed, meaning that the array-processing parameters are "hardwired" and do not change over time, or they may be adaptive, so that parameters are time varying and adjusted to track changes in the target signal and environment. Time Delay Estimation (TDE) between different microphones, plays important role in beam-forming. It is the first step in many speaker localization and tracking algorithms and have a significant impact in the next calculations. The most common form of fixed beam-forming is

the delay-and-sum beam-forming (DSB) method based on the temporal re-alignment of the signals. In delay-and-sum, signals from the various microphones are first time-aligned to adjust for the delays caused by path length differences between the target source and each of the microphones, using a variety of methods. A coherent addition of the time-aligned signal originating from the desired direction is achieved, while the signals originating from other directions add incoherently and are therefore attenuated. Thus the interfering signals can be attenuated relative to the desired signal. Because of the limited directionality, DSB achieves only a moderate reverberation reduction at low frequencies.

- **Adaptive Bean Forming** - In adaptive beam-forming, the array-processing parameters are dynamically adjusted according to some optimization criterion. Each channel is filtered by an adaptive filter which is adjusted to optimize a certain cost function. For example, the minimum variance distortion-less beam-former minimizes the signal energy under the constraint of undistorted response in the look direction. Many implementations have been proposed, starting from the original Generalized Side-lobe Canceler described in [15], a fixed beam-former and an adaptive beam-former are combined to obtain the desired target signal. As the adaptive-filter methods are based on the assumption that the desired signal and the noise are de-correlated, they suffer from signal cancellation in the case if reflected copies of the target signal. For example this is the case for speech signals in a reverberant environment. This seriously degrades the quality of the output signal and results in poor speech recognition performance. It is that desired signal cancellation in adaptive filtering methods can be reduced somewhat by adapting the parameters only during silence regions when no speech is present in the signals. While adaptive beam-forming is very successful for attenuating interferes, the reverberation reduction capability is only slightly better than that of the DSB.
- **De-Reverberation Techniques -** Reverberation is one of the undesired factors which may cause poor speech recognition. It may also affect microphone arrays performance. Much research has been focused on de-reverberation to solve the limitations of beam-forming methods dealing with this problem. In most de-reverberation methods, the effect of the room on the target signal is modeled as the response of the room. However, room impulse responses are generally non-minimum phase which causes the inverted de-reverberation filter to be unstable. It has been shown that under some specific situations it is possible to perform exact inverse filtering, even though the involved impulse responses are non-minimum-phase. This requires to use multiple channels and the room transfer functions of all channels be known. Also, the transfer functions should not have common zeros. However, concerns about the numerical stability and hence, practicality, of this method have been raised because of the large matrix inversions it requires. A different approach in this case has been introduced in [16], where the transfer function of the source-to-sensor room response for each microphone in the array is estimated and then, a truncated, time-reversed version of this estimate is used as a matched-filter for that source-sensor pair. The matched filters are used in a filter-and sum manner to process the array signals. This method is able to reduce the effects of reverberation significantly and obtain recognition improvements in highly reverberant environments. The main problem is that the room impulse responses need to be known exactly and even small inaccuracies can lead to significant deviations from the optimum solution. This requires the use of additional hardware to estimate the impulse responses and assumes that the transfer functions are fixed, which implies the location of the talker and the environmental conditions in the room will not change over time.
- **Blind Source Separation** - In the general Blind source separation (BSS) framework, observed

signals from multiple sensors are assumed to be the result of a combination of source signals and some unknown mixing matrix. In one family of BSS techniques, called independent component analysis (ICA), by using iterative optimization methods, the inverse of the unknown mixing matrix is estimated in the frequency domain for each DFT bin independently. This inverse matrix is used in the next step, to separate the microphone signals on a frequency-component basis, and then recombine them to form the output signal. Also, in [17] a framework for multi-channel blind signal processing is proposed, which can be used for blind dereverberation.
These methods are limited by this assumption that the number of competing sound sources is known and identical to the number of microphones present. Additionally, these methods assume that the sources are mutually independent point sources and are unable to process target signals in correlated or diffuse noise, while both of them are not commonly true in microphone array recordings.

- **Auditory Model- Based Array Processing -** These methods are based on processing of arrays and are capable to isolate target signals in extremely difficult acoustic conditions. In auditory model-based methods, no output waveform is produced; instead, some representation of the combined signal that models processing believed to occur in the auditory system. From this auditory representation, features can be extracted and used directly in speech recognition. Sullivan [18] devised such a scheme in which the speech from each microphone was band-pass filtered and then the cross-correlations among all the microphones in each sub-band were computed. The peak values of the cross-correlation outputs were used to derive a set of speech recognition features. While the method was quite promising in pilot work, the speech recognition performance on real speech was only marginally better than conventional DBS techniques and was much more computationally expensive.
- **Integrated Techniques** - The main idea in this method [12], is to improve the performance of microphone array-based speech recognition using the information from the statistical models of the recognition system. This feedback is used to tune the parameters of the array processing scheme. In this technique, the microphone-array speech recognition system will be treated as a single closed-loop system so that, not two independent entities cascaded together. In this way, information from the recognition system is integrated into the design of the array processing strategy. By adaptation of the coefficients of a filter-and-sum-beam-former (FSB), the probability of the correct transcription is maximized. Because of the nonlinear relationship between the filter coefficients and the cost function, the adaptation of the filter coefficients is very challenging.
- **phase-error based filters -** Phase-error based filters have been used successfully to enhance dual-microphone speech-signal [19]. It involves obtaining time-varying, or alternatively, time-frequency (TF), phase-error filters based on the time difference of arrival (TDOA) of the speech source of interest and the phases of the signals recorded by the microphones. It is shown that by masking the TF representation of the speech signals, the noise components are distorted beyond recognition while the speech source of interest maintains its perceptual quality. This has the aim of maintaining the spectral structure of the speech source of interest, and thereby, the main contents of that speech source, while damaging the spectral contents of other sources, hopefully beyond recognition. This technique requires knowledge regarding the time difference of arrival (TDOA) of the speech source of interest. More details about this algorithm will be explained in chapter four.

2.2.2.2 Feature-Based Compensation

It is well-known that MFCC is not robust enough in noisy environments, which suggests that the MFCC still has insufficient sound representation capability, especially at low SNR. In some algorithms, the signal enhancement is performed in feature extraction stage.

In [20], the enhancement of speech quality is performed at the first stage for Mel-cepstra based recognition systems. In which, the noise robustness is improved by the perceptual wavelet packet (PWP) based de-noising algorithm, which decompose the input speech signal into critical sub-band signals. Such a PWPT is designed to match the psychoacoustic model and to improve the performance of speech de-noising.

De-noising is performed by thresholding algorithm as a powerful tool in De-noising signals degraded by additive white noise. De-noising procedure is divided into two steps: firstly, threshold is estimated by penalized threshold algorithm , and secondly, two types of thresholding algorithms are applied, soft thresholding algorithm and modified soft thresholding (Mst) algorithm, to determine which of these algorithm is more efficient to improve recognition accuracy. Finally, these thresholded wavelet coefficients are constructed to obtain the enhanced speech samples by the inverse perceptual wavelet packet transform (IPWPT).

Then, the feature extraction is performed with Mel-frequency product spectrum cepstral coefficients (MFPSCCs). This is defined as the product of the power spectrum and the group delay function (GDF). It combines the magnitude spectrum and the phase spectrum. This is because, recently it has been shown that the phase spectrum is useful in human speech perception. While traditionally it is believed that the human auditory system ignores phase spectrum and uses only magnitude spectrum for speech perception.

In [21] an approach for cepstrum-domain feature compensation in ASR is proposed, which exploits noisy speech decomposition techniques that were originally developed for speech enhancement. The work relies on a minimum mean squared error log spectral amplitude estimator (MMSE-LSA). The MMSE-LSA estimator of a clean speech magnitude spectrum is represented as a noisy speech magnitude spectrum multiplied by a frequency dependent spectral gain function that is derived from the estimate of noise spectrum, signal-to-noise ratio (SNR), and speech absence probability. As a result, the estimated log spectrum of clean speech becomes a sum of the log spectra of noisy speech and the gain function. By converting these log spectra into cepstra, it turns out to be that estimated noise and clean speech can be considered to be additive in the cepstrum domain.

In [24], a technique for dynamic, frame-by-frame compensation of the Gaussian variances in the hidden Markov model (HMM) is proposed. In order to improve noise-robust speech recognition, it exploits the feature variance or uncertainty estimated during the speech feature enhancement process. The work provides an alternative to the Bayesian predictive classification (BPC) decision rule by carrying out an integration over the feature space instead of over the model-parameter space, offering a simpler system implementation and dynamic compensation capabilities at the frame level. The computation of the feature enhancement variances is carried out using a probabilistic and parametric model of speech distortion. Dynamic compensation of the Gaussian variances in the HMM recognizer is derived, which is simply enlarging the HMM Gaussian variances by the feature enhancement variances.

In [25], a data driven compensation technique has been proposed that modifies on-line the incoming spectral representation of degraded speech to approximate the features of high quality speech used to train a classifier. The Bayesian inference framework is applied to the degraded spectral coefficients based on modeling clean speech linear-spectrum with appropriate non-Gaussian distributions that allow maximum a-posteriori (MAP) closed form solution to be set. MAP solution leads to a

soft threshold function applied and adapted to the spectral characteristics and noise variance of each spectral band.

2.2.2.3 Model Based Compensation

Model compensation techniques have proved to be a superior alternatives to feature subtraction in many cases. Parallel model combination (PMC) and vector Taylor series (VTS) approaches have been used for model compensations in several works. They use a model of clean speech as the starting point and then adapt this model to fit a new noise environment. PMC and VTS have been widely used in robust speech recognition. The original versions of PMC and VTS do not compensate the models continuously to fit the dynamic noise conditions. Furthermore, their performance relies on the noise estimate obtained during speech pauses indicated by a voice activity detector.

In [22] a noise estimation algorithm is proposed that produces an estimate of the noise spectrum in every frame. This allows the speech models compensated with PMC or VTS methods to be time-varying. The noise estimation produces biased noise estimates and therefore, a method to compensate the bias is also proposed.

In [23], Several noise compensation schemes for speech recognition in impulsive and non-impulsive noise are compared. These noise compensation schemes are spectral subtraction, HMM-based Wiener filters, noise-adaptive HMM's, and a front-end impulsive noise removal.

In spectral subtraction, an estimate of the speech spectra is obtained by subtracting an estimate of the average noise spectra from the noisy speech. Spectral subtraction utilizes only an estimate of the noise power spectrum. As a result, spectral subtraction performs poorly when compared with methods such as Wiener filters and model adaptation that utilize estimates of the power spectra of the signal and the noise processes.

Two implementation methods for using Wiener filters with HMM's to improve speech recognition in noisy conditions are described, namely, state-dependent Wiener filters and state-integrated Wiener filters. In state-dependent Wiener filtering, when the noisy speech is processed by the filter sequence derived from the correctly hypothesized HMM, the effect of noise is reduced. However, when the filter sequence is derived from an incorrectly hypothesized HMM, no significant noise reduction occurs. This process increases the likelihood of accurate speech recognition. Figure 2.14 illustrates this methods.

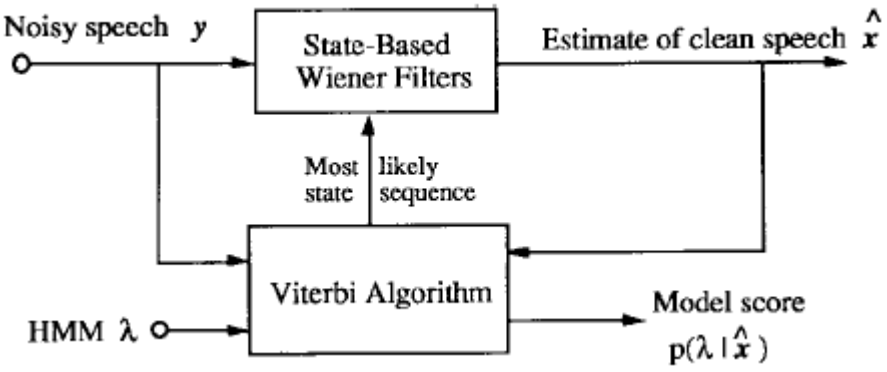


Figure 2.14: Illustration of HMM's with state-dependent Wiener filters for noisy speech recognition.

A drawback of this method is that it relies on the accuracy of the state sequence from which the

state dependent Wiener filters are derived. The problem is that the accuracy of the state sequence estimators deteriorates rapidly with increasing noise. An advantage of the state-integrated Wiener filters implementation technique over the state-based Wiener filtering is that it does not rely on the accuracy of the maximum likelihood state sequence estimated from the noisy speech. Experimentally, this method resulted in substantial improvement to recognition performance.

A deficiency of conventional filtering methods, such as spectral subtraction, is that crucial speech information may be removed during the filtering process. For noisy speech recognition, an alternative to filtering is to adapt the parameters of the speech models to include the effects of noise in an attempt to obtain models that would have been obtained under matched training and testing conditions. Adaptation of speech model parameters depends on the choice of the speech features.

For linear speech features such as power spectral or correlation features, the statistics of the noisy speech are given as the sum of the statistics of the speech and the additive noise. There are methods for the noise adaptation of LPC-based speech features.

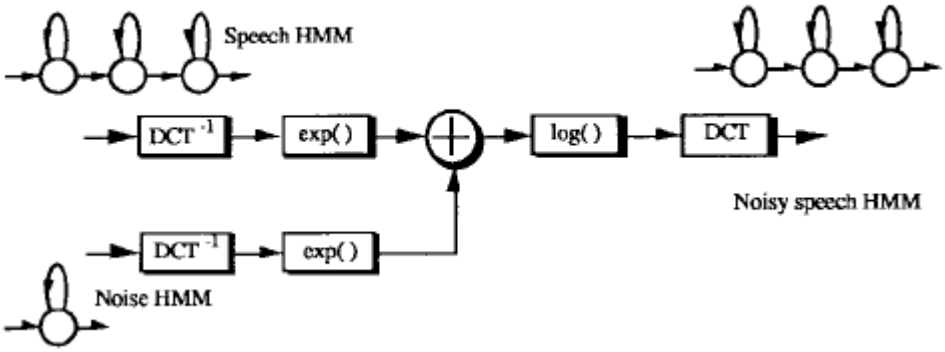


Figure 2.15: Block diagram configuration of adaptation system

For cepstral speech features, the nonlinear, logarithmic transformation from the spectral domain to the cepstral domain affects the adaptation. There are methods that use noise-adaptive speech models trained on log-power spectral features. In these models, it is assumed that at any given time, each speech spectral band is dominated either by the signal energy or by the noise energy. Also, a model combination method is proposed in which a speech HMM and a noise HMM are combined to produce a model of the noisy signal. Figure 2.15 outlines the stages involved in the model adaptation process, where the HMM state observation parameters are converted from the cepstral domain into the log spectral domain using an inverse DCT. The log spectral features are then mapped to linear power spectral features. Noise adaptation takes place on the power spectral features, and the combined noise and speech model is mapped back to the cepstral domain.

2.2.3 Adaptation

In speech recognition task, it is highly probable a new speaker appears who is poorly presented in the training data and also new unseen environments. Adaptation is the solution to these cases. Human routinely get adapted in responding and recognizing speech signals. But, currently, ASR systems deliver significantly degraded performance when they encounter audio signals that differ from the limited conditions under which they were originally developed and trained. One research line is to concentrate on creating and developing systems that would be much more robust against

variability and shifts in acoustic environments, reverberation, external noise sources, communication channels , speaker characteristics (e.g., speaking style, nonnative accents, emotional state), and language characteristics (e.g., formal/informal styles, dialects, vocabulary, topic domain). As described in the previous section, robust ASR systems are used to handle environmental noise. New techniques and architectures are also proposed to treat the variation in speakers. This challenging problem can productively exploits the knowledge from related disciplines, including natural-language processing, information retrieval, and cognitive science.

One simplest way for current state-of-the-art recognition systems to improve performance on a given task is to increase the amount of task-relevant training data from which its models are constructed [30]. System capabilities have progressed directly along with the amount of speech corpora available to cover all variations in speech nature. Despite all the speech databases that have been exploited so far, system performance consistently improves when more relevant data are available. This indicates that more data are needed to capture crucial information in the speech signal. This is especially important in increasing the facility with which we can learn, understand, and subsequently automatically recognize a wide variety of languages. To have more powerful systems capable of understanding the nature of speech itself, we must collect Well-labeled speech corpora. Indeed, the single most popular English speech database available from the Linguistic Data Consortium (LDC) is TIMIT, a very compact acoustic-phonetic database. There is also an European Language Resources Association (ELRA) which provides languages resources for language engineering and to evaluate language engineering technologies.

On the other hand, since it is not possible to predict and collect separate data for any and all types of speech, topic domains, and so on, it is important to enable automatic systems to learn and generalize even from single instances (episodic learning) or limited samples of data, so that new or changed signals (e.g., accented speech, noise adaptation) could be correctly understood. It has been well demonstrated that adaptation in automatic speech systems is very beneficial. A key human cognitive characteristic is the ability to learn and adapt to new patterns and stimuli. This is similar to human capability as mentioned above.

Adaptation may be *supervised*, which means that accurate transcriptions are available for all of the adaptation data, or it can be *unsupervised* that is the case when the required transcriptions must be hypothesized [5]. The main adaptation approaches are feature based and linear transformation based approaches.

2.2.3.1 Feature-Based Schemes

Feature-Based Schemes only depend on the acoustic features. Different approaches in this case are:

- **Mean and Variance Normalization**- Cepstral mean normalization (CMN) removes the average feature value of the feature vector from each observation which reduces sensitivity to channel variation, provided that the channel effects do not significantly vary with time or the amplitude of the signal. Cepstral variance normalization (CVN) scales each individual feature coefficient to have a unit variance and this reduces the sensitivity to additive noise.
- **Gaussianization**- It is an extension to normalize the higher order statistics for the data from, for example, the overall distribution of all the features from a specific speaker is Gaussian. This so-called Gaussianization, which is performed by finding a transform $x = \phi(\tilde{x})$that yields

$$x \sim N(o, I) \tag{2.36}$$

  where 0 is the d-dimensional zero vector and I is a d-dimensional identity matrix. Performing this for the complete feature vector is highly complex and each element of the feature vector is

treated independently. When the normalization data set is small an M-component GMM on the data is estimated instead.
Both these forms of Gaussianization have assumed that the dimensions are independent of one another. To reduce the impact of this assumption Gaussianization is often applied after some form of decorrelating transform.

- **Vocal Tract Length Normalization (VTLN)-** Speech signal carries information about vocal tract length (VTL). For example, the formant frequencies of vowels decrease as the VTL increases. typical female speaker exhibits formant frequencies around 20% higher than those of a male speaker. Warping the frequency axis for specific speakers (or speaker cluster-specific) could compensate for such difference in VTL, resulting in a speech recognition system with a reduced word error rate (WER) [26].
To implement VTLN two issues need to be addressed: definition of the scaling function and estimation of the appropriate scaling function parameters for each speaker [5]. Figure 2.16-a shows a simple linear mapping which results in a problem at high frequencies where female voices have no information in the upper frequency band and male voices have the upper frequency band truncated. This can be mitigated by using a piece-wise linear function of the form shown in Figure 2.16-b. The speaker-specific warp factor α is usually obtained by maximizing

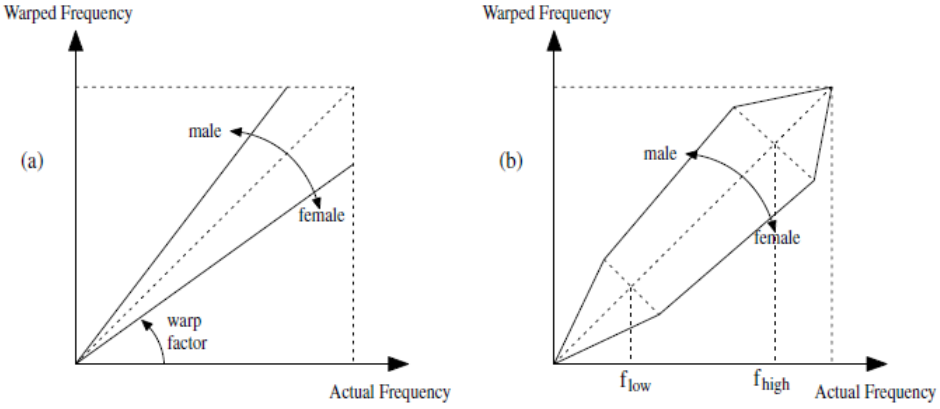


Figure 2.16: Vocal tract length normalization

the likelihood (ML) with respect to the model so that the probability of recognizing an utterance given a particular acoustic model is maximized. VTLN is particularly effective for telephone speech where speakers can be clearly identified. It is less effective for other applications such as broadcast news transcription where speaker changes must be inferred from the data.

2.2.3.2 Linear Transform-Based Schemes

For cases where adaptation data is limited, linear transformation based schemes are currently the most effective form of adaptation. These differ from feature-based approaches that use the acoustic model parameters and require a transcription of the adaptation data.

Maximum Likelihood Linear Regression (MLLR) - In MLLR, a set of linear transforms are used to map an existing model set into a new adapted model set such that the likelihood of the adaptation data is maximized. MLLR is generally very robust and well suited to unsupervised incremental adaptation [5]. Two main variants of MLLR are: unconstrained and constrained. In unconstrained MLLR,

separate transforms are trained for the means and variances:

$$\hat{\mu}^{(sm)} = A^{(s)}\mu^{(m)} + b^{(s)} \quad ; \quad \hat{\Sigma}^{(sm)} = H^{(s)}\Sigma^{(m)}H^{(s)T} \tag{2.37}$$

where s indicates the speaker. For MLLR there are no constraints between the adaptation applied to the means and the covariances. If the two matrix transforms are constrained to be the same, then a linear transform related to the feature-space transforms described earlier may be obtained. This is constrained MLLR (CMLLR):

$$\hat{\mu}^{(sm)} = \tilde{A}^{(s)}\mu^{(m)} + \tilde{b}^{(s)} \quad ; \quad \hat{\Sigma}^{(sm)} = \tilde{A}^{(s)}\Sigma^{(m)}\tilde{A}^{(s)T} \tag{2.38}$$

CMLLR is efficient where the speaker (or acoustic environment) changes rapidly.

Both forms of linear transforms require transcriptions of the adaptation data in order to estimate the model parameters. For supervised adaptation, the transcription is known and may be directly used without further consideration. When used in unsupervised mode, the transcription must be derived from the recognizer output and in this case, MLLR is normally applied iteratively to ensure that the best hypothesis for estimating the transform parameters is used. This is repeated until convergence is achieved. This can be refined using lattices or N-best list methods.

Linear transforms can also be estimated using discriminative criteria for supervised adaptation, but in the case of unsupervised adaptation, for example in BN transcription systems, there is an additional concern. In the absense of training transcription, discriminative training schemes have to use hypothesized transcription to modify the parameters so that the posterior of the transcription (or a function thereof) is improved. This will cause errors. In practice, discriminative unsupervised adaptation is not commonly used [5].

A powerful feature of linear transform-based adaptation is that it allows all the acoustic models to be adapted using a variable number of transforms. When the amount of adaptation data increases, instead of using a global transform, the HMM state components can be grouped into regression classes with each class having its own transform and therefore transforms can be increased correspondingly and even they can be determined automatically using a ***regression class tree*** as illustrated in Figure 2.17. Each node represents a regression class, i.e., a set of Gaussian components which will share a single transform [5].

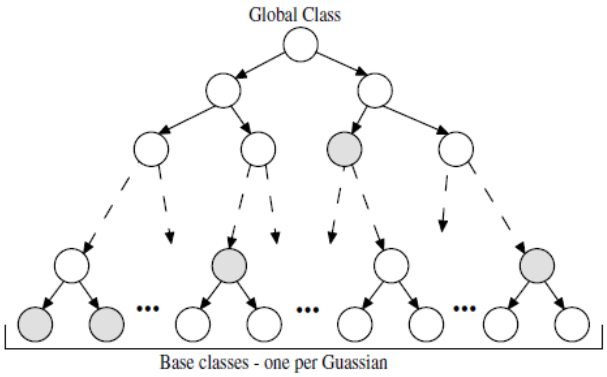


Figure 2.17: A regression class tree

2.2.3.3 Maximum a Posteriori (MAP) Adaptation

For adaptation purpose, it is also possible to use standard statistical approaches to obtain robust parameter estimates. One common approach is maximum a posteriori (MAP) adaptation where in addition to the adaptation data, a prior over the model parameters, $P(\lambda)$, is used to estimate the model parameters. Given some adaptation data $x^{(1)}, x^{(2)}, ...x^{(R)}$ and a model with parameters λ, MAP-based parameter estimation seeks to maximize the following objective function:

$$\begin{aligned} F_{map} &= F_{m1}(\lambda) + \frac{1}{R}\log(P(\lambda)) \\ &= \left(\frac{1}{R}\sum_{r=1}^{R}\log P\left(x^{(r)}|w_{ref}^{(r)};\lambda\right)\right) + \frac{1}{R}\log(P(\lambda)) \end{aligned} \tag{2.39}$$

MAP is especially useful for porting a well-trained model set to a new domain for which there is only a limited amount of data. A major drawback of MAP adaptation is that every Gaussian component is updated individually. If the adaptation data is sparse, then many of the model parameters will not be updated.

Chapter Summary

In this chapter different components of the standard HMM based speech Recognition were described. They are feature extraction, acoustic modeling, language modeling and decoding. Then, some challenges of ASR systems were discussed. The first challenging point is to improve the primitive ASR system for Large vocabulary continuous speech recognition (LVCR) which requires feature projection and discriminative training techniques. Robust speech recognition and adaptation are the two other challenges that has been described and common used methods for each one were introduced.

3 Building a Spanish Recognizer

In this chapter different steps of building a Spanish recognizer using HTK toolkit and Albayzin Spanish acoustic corpus will be described. HTK is primarily designed for building HMM-based speech processing tools, particularly recognizers. The Albayzin corpus contains limitted number of training audio recordings and their corresponding transcriptions. As shown in Figure 3.1, there are two major processing stages [3]. First, the HTK training tools are used to estimate the parameters of a set of HMMs using training utterances and their associated transcriptions from the corpus. Secondly, unknown utterances are transcribed using the HTK recognition tools. The recognition task may be performed on a simple grammar with limited statements and vocabularies using the acoustic models, or it may be used in more complicated steps for the whole language using the acoustic and language models.

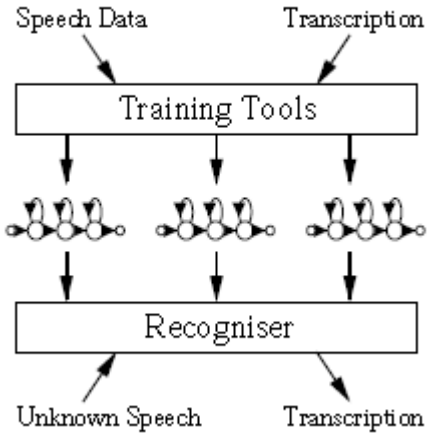


Figure 3.1: The general view of HTK processing steps in recognition.

3.1 Training HMM models

This section presents the most important points to implement different speech feature extraction methods using HTK. In [3], different steps for training the acoustic model and testing it, has been explained. Also, different techniques for speech analysis and feature extraction has been described. We are going to use these techniques as well as comparing them in this section. As mentioned before, the Albayzin Spanish acoustic corpus is used for training, containing 4800 audio files and their corresponding transcriptions, and a phoneme dictionary.

3.1.1 Data Preparation

The first stage of any recognizer development project is data preparation. In the case of the training data, the prompt scripts will be used in conjunction with a pronunciation dictionary to provide the initial phone level transcriptions needed to start the HMM training process. We have processed the dictionary file so that, it has no repetition unless there are two different phonetic forms for one word.

3.1.1.1 Word Level

First, the word label files from the transcription files of the corpus are built. This will convert the transcriptions into the word level master label file (MLF). In word MLF file, the prompt labels of the corpus are converted into path names, each word is written on a single line and each utterance is terminated by a single period on its own. Figure 3.2-a shows a sample word.mlf file.

3.1.1.2 Phone Level

Once the word level MLF has been created, phone level MLFs can be generated by replaceing each word in words.mlf by the corresponding pronunciation in the dictionary file. Also a silence model sil is inserted at the start and end of every utterance. Figure 3.2-b shows a sample phone.mlf file.

(a)

```
#!MLF!#
"*/AAFA0001.SES.lab"
francia
suiza
y
hungría
ya
hicieron
causa
común
.
"*/AAFA0002.SES.lab"
mi
primer
profesor
de
lengua
.
.
.
```

(b)

```
#!MLF!#
"*/AAFA0001.SES.lab"
sil
f
r
a
n
T
j
a
s
w
i
T
a
.
.
.
```

Figure 3.2: (a) word MLF file (b) Phone MLF file

3.1.2 Feature Extraction

In feature extraction stage, the raw speech waveforms is parametrized into sequences of feature vectors. HTK support both FFT-based and LPC-based analysis. Different feature extraction methods that are supported, are MFCC (Mel-Frequency Cepstrum Coefficients) that has been described in 2.1.1, LPC (Linear Prediction Analysis), LPCEPSTRA (LPC Cepstral Coefficients), LPREFC (Linear Prediction Reflection Coefficients), PLP (Perceptual Linear Prediction). In order to perform any of these feature extraction methods, a special configuration file should be used.

Independent of which method is required, there are some simple pre-processing operations that can be applied prior to performing the actual signal analysis.

- First, the DC mean can be removed from the source waveform. This is useful when the original analogue-digital conversion has added a DC offset to the signal. It is applied to each window individually.
- Secondly, it is common practice to pre-emphasize the signal by applying the first order difference equation:
$$s'_n = ks_n - s_{n-1} \tag{3.1}$$
to the samples$\{s_n; n = 1, N\}$ in each window. Here k is the pre-emphasis coefficient which should be in the range $0 < k < 1$.
- Finally, it is usually beneficial to taper the samples in each window so that discontinuities at the window edges are attenuated. For this, the following transformation is applied to the samples $\{s_n; n = 1, N\}$ in the window:
$$s'_n = \left\{0.54 - 0.46\cos\frac{2\pi(n-1)}{(n-1)}\right\} s_n \tag{3.2}$$
When both pre-emphasis and Hamming windowing are enabled, pre-emphasis is performed first. In practice, all three of the above are usually applied [3].

3.1.2.1 Linear Prediction (LP) Analysis

For explaining the LP idea, we consider the source/system model for speech production of human where the sampled speech signal is modeled as the output of a linear, slowly time-varying system excited by either quasi-periodic impulses (during voiced speech), or random noise (during unvoiced speech) [1]. The particular form of the source/system model implied by linear predictive analysis is shown in Figure 3.3. For over short time intervals, the linear system is described by an all-pole system function of the form:
$$H(z) = \frac{S(z)}{E(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{3.3}$$
In linear predictive analysis, the excitation is defined implicitly by the vocal tract system model. So that, the excitation is whatever is needed to produce $s[n]$ at the output of the system. For the system of Figure3.3 with the vocal tract model of Equation 3.3, the speech samples $s[n]$ are related to the excitation $e[n]$ by the difference equation:
$$s[n] = \sum_{k=1}^{p} a_k s[n-k] + Ge[n] \tag{3.4}$$
A linear predictor with prediction coefficients, α_k is defined as a system whose output is:

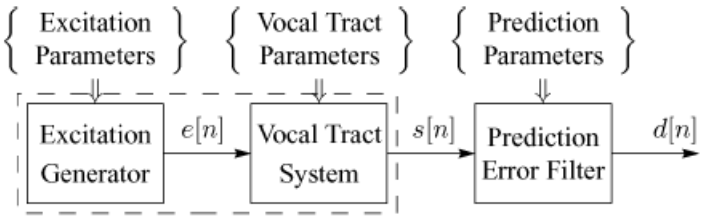


Figure 3.3: Model for linear predictive analysis of speech signals.

$$\tilde{s}[n] = \sum_{k=1}^{p} \alpha_k s[n-k] \tag{3.5}$$

and the prediction error, defined as the amount by which $\tilde{s}[n]$ fails to exactly predict sample $s[n]$, is:

$$d[n] = s[n] - \tilde{s}[n] = s[n] - \sum_{k=1}^{p} \alpha_k s[n-k] \tag{3.6}$$

From Equation 3.6 it follows that the prediction error sequence is the output of an FIR linear system whose system function is:

$$A(z) = 1 - \sum_{k=1}^{p} \alpha_k z^{-k} = \frac{D(z)}{S(z)} \tag{3.7}$$

It can be seen by comparing Equation 3.4 and 3.6 that if the speech signal obeys the model of Equation 3.4 exactly, and if $\alpha_k = a_k$, then $d[n] = Ge[n]$. Thus, the prediction error filter, $A(z)$, will be an inverse filter for the system, $H(z)$, of Equation 3.3, so that:

$$H(z) = \frac{G}{A(z)} \tag{3.8}$$

The basic problem of linear prediction analysis can be described as determining the set of predictor coefficients $\{\alpha_k\}$ directly from the speech signal in order to obtain a useful estimate of the time-varying vocal tract system through the use of Equation 3.8. The basic approach is to find a set of predictor coefficients that will minimize the mean-squared prediction error over a short segment of the speech waveform. The resulting parameters are then assumed to be the parameters of the system function $H(z)$ in the model for production of the given segment of the speech waveform. This process is repeated periodically at a rate appropriate to track the phonetic variation of speech (i.e., order of 50–100 times per second).

In order to understand the LP concept better, it could be described in other ways. One way is to recall that, if $\alpha_k = a_k$, then $d[n] = Ge[n]$. For voiced speech this means that $d[n]$ would consist of a train of impulses, i.e., $d[n]$ which would be small except at isolated samples spaced by the current pitch period, P_0. Thus, finding α_ks that minimize the mean-squared prediction error seems consistent with this observation.

A second motivation for this approach follows from the fact that if a signal is generated by Equation 3.4 with non-time-varying coefficients and excited either by a single impulse or by a stationary white noise input, then it can be shown that the predictor coefficients that result from minimizing the mean-squared prediction error (over all time) are identical to the coefficients of Equation 3.4.

The short-time average prediction error is defined as:

$$E_{\hat{n}} = \left\langle d_{\hat{n}}^2 [m] \right\rangle = \left\langle \left(s_{\hat{n}} [m] - \sum_{k=1}^{p} \alpha_k s_{\hat{n}} [m-k] \right)^2 \right\rangle \tag{3.9}$$

where $s_{\hat{n}} [m]$ is a segment of speech that has been selected in a neighborhood of the analysis time $\hat{n}$:

$$s_{\hat{n}} [m] = s [m + \widehat{n}] \ , -M_1 \leq m \leq M_2 \tag{3.10}$$

That is, the time origin of the analysis segment is shifted to sample $\hat{n}$ of the entire signal. The notation $\langle\rangle$ denotes averaging over a finite number of samples.

We can find the values of α_k that minimize $E_{\hat{n}}$ in Equation 3.9 by setting $\frac{\partial E_{\hat{n}}}{\partial \alpha_i} = 0$, for $i = 1, 2, ..., p$, thereby obtaining the equations:

$$\sum_{k=1}^{p} \tilde{\alpha}_k \left\langle s_{\hat{n}} [m-i] \, s_{\hat{n}} [m-k] \right\rangle = \left\langle s_{\hat{n}} [m-i] \, s_{\hat{n}} [m] \right\rangle \ , 1 \leq i \leq p \tag{3.11}$$

where the $\tilde{\alpha}$ are the values of α_k that minimize $E_{\hat{n}}$ in Equation 3.9. (Since the $\tilde{\alpha}$ are unique, we will drop the tilde and use the notation α_k to denote the values that minimize $E_{\hat{n}}$. If we define,

$$\varphi_{\hat{n}} [i,k] = \left\langle s_{\hat{n}} [m-i] \, s_{\hat{n}} [m-k] \right\rangle \tag{3.12}$$

then Equation 3.11 can be written more compactly as[1]

$$\sum_{k=1}^{p} \alpha_k \varphi_{\hat{n}} [i,k] = \varphi_{\hat{n}} [i,0] \ i = 1, 2, ..., p \tag{3.13}$$

If we know $\varphi_{\hat{n}} [i,k]$ for $, 1 \leq i \leq p$ and $, 0 \leq k \leq p$, this set of p equations in p unknowns, which can be represented by the matrix equation:

$$\Phi \alpha = \psi, \tag{3.14}$$

can be solved for the vector $\alpha = \{\alpha_k\}$ of unknown predictor coefficients that minimize the average squared prediction error for the segment $s_{\hat{n}} [m]$.[2] Using Equation 3.9 and 3.11, it is demonstrated that, the minimum mean-squared prediction error can be shown as:

$$E_{\hat{n}} = \varphi_{\hat{n}} [0,0] - \sum_{k=1}^{p} \alpha_k \varphi_{\hat{n}} [0,k] \tag{3.15}$$

Thus, the total minimum mean-squared error consists of a fixed component equal to the mean-squared value of the signal segment minus a term that depends on the predictor coefficients that satisfy Equation 3.13, which are, the optimum coefficients reduce $E_{\hat{n}}$ in Equation3.15 the most.

To solve for the optimum predictor coefficients, we must first compute the quantities$\varphi_{\hat{n}} [i,k]$ for $, 1 \leq i \leq p$ and $, 0 \leq k \leq p$. Once this is done we only have to solve Equation 3.13 to obtain the α_ks.

[1] The quantities $\varphi_{\hat{n}} [i,k]$ are in the form of a correlation function for the speech segment $s_{\hat{n}} [m]$. The details of the definition of the averaging operation used in Equation 3.12 have a significant effect on the properties of the prediction coefficients that are obtained by solving Equation 3.13.

[2] Although the α_ks are functions of $\hat{n}$ (the time index at which they are estimated) this dependence will not be explicitly shown. The subscripts $\hat{n}$ on $E_{\hat{n}}$, $s_{\hat{n}} [m]$, and $\varphi_{\hat{n}} [i,k]$ when no confusion will result.

Thus, in principle, linear prediction analysis is very straightforward. However, there are some details of the computation of $\varphi_{\hat{n}}[i,k]$ and the subsequent solution of the equations are more complex. For example, we have not been explicit about the meaning of the averaging notation $\langle\rangle$ used to define the mean-squared prediction error in Equation 3.9. As the procedures are in short time analysis , the averaging must be in a finite interval. There are two methods for linear predictive analysis which emerge out of a consideration of the limits of summation and the definition of the waveform segment $s_{\hat{n}}[m]$. These two methods, applied to the same speech signal yield slightly different optimum predictor coefficients. They are *Covariance* method and *Auto-correlation* method. The second method will be explained in more detail, as HTK uses this Auto-correlation method.

The Auto-correlation method:

Auto-correlation is the most widely used method of linear predictive analysis because, the covariance function $\varphi_{\hat{n}}[i,k]$ needed in Equation 3.13 reduces to the short time auto-correlation function (STACF) $\phi_{\hat{n}}[|i-k|]$. In the auto-correlation method, the analysis segment $s_{\hat{n}}[m]$ is defined as [1]:

$$s_{\hat{n}}[m] = \begin{cases} s[n+m]\,w[m] & -M_1 \le m \le M_2 \\ 0 & otherwise \end{cases} \tag{3.16}$$

where the analysis window $w[m]$ is used to taper the edges of the segment to zero. Since the analysis segment is defined by the windowing of Equation 3.16 to be zero outside the interval $-M_1 \le m \le M_2$, it follows that the prediction error sequence $d_{\hat{n}}[m]$ can be nonzero only in the range $-M_1 \le m \le M_2 + p$. Therefore, $E_{\hat{n}}$ is defined as:

$$E_{\hat{n}} = \sum_{m=-M_1}^{M_2+p} (d_{\hat{n}}[m])^2 = \sum_{-\infty}^{\infty} (d_{\hat{n}}[m])^2 \tag{3.17}$$

The windowing of Equation 3.16 allows us to use the infinite limits to signify that the sum is over all nonzero values of $d_{\hat{n}}[m]$.

On the other hand, The quantities $\varphi_{\hat{n}}[i,k]$ defined in Equation 3.12 inherit the same definition of the averaging operator,as $E_{\hat{n}}$:

$$\varphi_{\hat{n}}[i,k] = \sum_{m=-M_1}^{M_2} s_{\hat{n}}[m-i]\,s_{\hat{n}}[m-k] \quad \begin{cases} 1 \le i \le p \\ 0 \le k \le p \end{cases} \tag{3.18}$$

recalling $s_{\hat{n}}[m] = s[m+\widehat{n}]$ over the range $-M_1-p \le m \le M_2$. By changes of index of summation, Equation3.18 can be expressed in the equivalent forms:

$$\varphi_{\hat{n}}[i,k] = \sum_{m=-M_1-i}^{M_2-i} s_{\hat{n}}[m]\,s_{\hat{n}}[m+i-k] \tag{3.19a}$$

$$\sum_{m=-M_1-k}^{M_2-k} s_{\hat{n}}[m]\,s_{\hat{n}}[m+k-i] \tag{3.19b}$$

from which it follows that $\varphi_{\hat{n}}[i,k] = \varphi_{\hat{n}}[k,i]$. Applying the same notion to the Equation 3.19a and 3.19b leads to the conclusion that:

$$\varphi_{\hat{n}}[i,k] = \sum_{m=-\infty}^{\infty} s_{\hat{n}}[m]\,s_{\hat{n}}[m+|i-k|] = \phi_{\hat{n}}[|i-k|] \tag{3.20}$$

Thus, $\varphi_{\hat{n}}[i,k]$ is a function only of $|i-k|$. Therefore, we can replace $\varphi_{\hat{n}}[i,k]$ by $\phi_{\hat{n}}[|i-k|]$, which is the STACF :

$$\phi_{\hat{n}}[k] = \sum_{m=-\infty}^{\infty} s_{\hat{n}}[m]\, s_{\hat{n}}[m+k] = \phi_{\hat{n}}[-k] \tag{3.21}$$

The resulting set of equations for the optimum predictor coefficients is therefore:

$$\sum_{k=1}^{p} \alpha_k \phi_{\hat{n}}[|i-k|] = \phi_{\hat{n}}[i]\ , i = 1,2,...,p \tag{3.22}$$

Figure 3.4 shows the sequences that are involved in computing the optimum prediction coefficients using the auto-correlation method. The upper plot shows the sampled speech signal $s_{\hat{n}}[m]$ with a Hamming window centered at time index $\hat{n}$. The middle plot shows the result of multiplying the signal $s_{\hat{n}}[\hat{n}+m]$ by the window $w[m]$ and redefining the time origin to obtain $s_{\hat{n}}[m]$. Note that the zero-valued samples outside the window are shown with light shading. The third plot shows the prediction error computed using the optimum coefficients. Note that for this segment, the prediction error (which is implicit in the solution of Equation 3.13 is nonzero over the range $-M_1 \leq m \leq M_2 + p$. Also note the lightly shaded p samples at the beginning. These samples can be large due to the fact that the predictor must predict these samples from the zero-valued samples that precede the windowed segment $s_{\hat{n}}[m]$. At least one of these first p samples of the prediction error must be nonzero. Similarly, the last p samples of the prediction error can be large due to the fact that the predictor must predict zero-valued samples from windowed speech samples. It can also be seen that at least one of these last p samples of the prediction error must be nonzero. For this reason, it follows that $E_{\hat{n}}$, being the sum of squares of the prediction error samples, must always be strictly greater than zero. For this reason, a tapering window is generally used in the auto-correlation method.

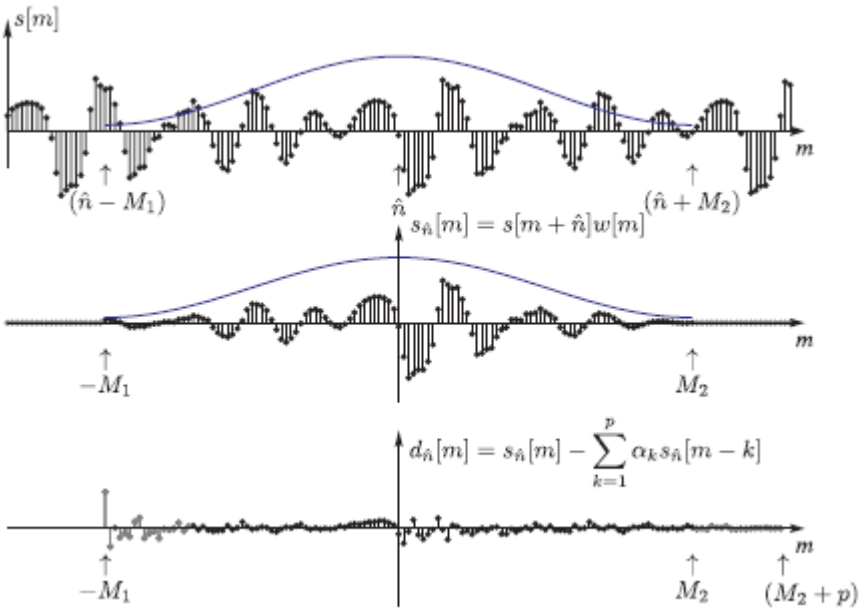


Figure 3.4: Illustration of windowing and prediction error for the auto-correlation method.

LPC Spectrum:

The frequency-domain interpretation of linear predictive analysis provides an informative link to the STFT and cepstrum analysis. The auto-correlation method is based on the short-time auto-correlation function, $\phi_{\hat{n}}[m]$, which is the inverse discrete Fourier transform of the magnitude-squared of the STFT, $\left|S_{\hat{n}}\left(e^{j\hat{w}}\right)\right|^2$, of the windowed speech signal $s_{\hat{n}}[m] = s[n+m]\,w[m]$. The values $\phi_{\hat{n}}[m]$ for $m = 0, 1, ..., p$ are used to compute the prediction coefficients and gain, which in turn define the vocal tract system function $H(z)$ in Equation 3.3. Therefore, the magnitude-squared of the frequency response of this system, obtained by evaluating $H(z)$ on the unit circle at angles $\frac{2\pi f}{f_s}$, is of the form:

$$\left|h\left(e^{j2\pi f/f_s}\right)\right|^2 = \left|\frac{G}{1-\sum_{k=1}^{p}\alpha_k e^{-j2\pi f/f_s}}\right|^2 \tag{3.23}$$

and can be thought of as an alternative short-time spectral representation. Figure3.5 shows a comparison between short-time Fourier analysis and linear predictive spectrum analysis for segments of voiced and unvoiced speech. Figures 3.5(a) and 3.5(c) show the STACF, with the first 23 values plotted with a heavy line. These values are used to compute the predictor coefficients and gain for an LPC model with $p = 22$. The frequency responses of the corresponding vocal tract system models are computed using Equation 3.23 where the sampling frequency is $f_s = 16kHz$. These frequency responses are superimposed on the corresponding STFTs (shown in gray).

The rapid variations with frequency in the STFT are due primarily to the excitation, while the overall shape is assumed to be determined by the effects of glottal pulse, vocal tract transfer function, and radiation. In cepstrum analysis, the excitation effects are removed by low-pass lifting the cepstrum. In linear predictive spectrum analysis, the excitation effects are removed by focusing on the low-time auto-correlation coefficients. The amount of smoothing of the spectrum is controlled by the choice of p. Figure 3.5 shows that a linear prediction model with p = 22 matches the general shape of the short-time spectrum, but does not represent all its local peaks and valleys, and this is exactly what is desired.

In a particular application, the prediction order is generally fixed at a value that captures the general spectral shape due to the glottal pulse, vocal tract resonances, and radiation. From the acoustic theory of speech production, it follows that the glottal pulse spectrum is low-pass, the radiation filtering is high-pass, and the vocal tract imposes a resonance structure that, for adult speakers, is comprised of about one resonance per kilohertz of frequency [1]. For the sampled speech signal, the combination of the low-pass glottal pulse spectrum and the high-pass filtering of radiation are usually adequately represented by one or two additional complex pole pairs. When coupled with an estimate of one resonance per kilohertz, this leads to a rule of thumb of p = 4 + fs/1000. For example, in Figure 3.5 the sampling rate was $f_s = 16000Hz$, a predictor order of $p = 22$ gave a good representation of the overall shape and resonance structure of the speech segment over the band from 0 to $8000Hz$.

3.1.2.2 Perceptual Linear Prediction (PLP)

The PLP feature extraction uses LPC coefficients and usually transfer them to LPC-cepstrum [28]. Assuming the signal $s_{\hat{n}}[m]$, its auto-correlation $\phi_{\hat{n}}[m]$ is the inverse Fourier transform of the power spectrum $|S(w)|^2$ of the signal. The continuous -frequency Fourier transform can not be computed easily, but its possible to take an *FFT* to obtain the $S[k]$ and then the auto-correlation could be calculated as the inverse transform of $|S[k]|^2$.

PLP uses this method, but replaces $|S[k]|^2$ by a perceptually motivated power spectrum. The most important aspect is the non-linear frequency scaling, which can be achieved through a set of filterbanks similar to those described in 2.1.1, so that, the critical-band power spectrum can be computed

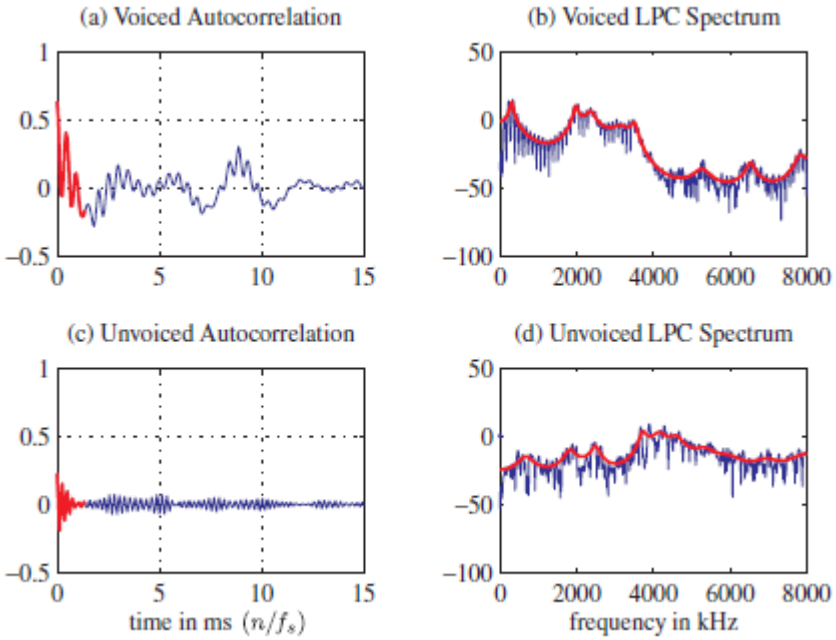


Figure 3.5: Comparison of short-time Fourier analysis with linear predictive analysis.

around central frequencies with increasing bandwidths. Another difference is that, instead of taking the logarithm on the filter-bank energy outputs, a different non-linearity compression is used, often cubic root. It is demonstrated that the use of this different non-linearity is beneficial for speech recognizers in noisy conditions.

3.1.3 HMM Modeling

This step is to create well-trained set of HMM models. First single-Gaussian mono-phone HMMs are created. This means each phoneme of Spanish language is modeled with a three-sate HMM model with single Gaussians. Then cross-word triphone models will be built by tying the proper states. By splitting the Gaussians, mixture Gaussian HMM models would be made Finally. Before starting to build the models, the evaluation of the recognition results will be explained to underestand the out put information of the recognition step.

3.1.3.1 Evaluating Recognition Results

Once the test data has been processed by the recogniser, the next step is to analyse the results. HTK, compares the transcriptions output with the original reference transcriptions and then outputs various statistics. It matches each of the recognised and reference label sequences by performing an optimal string match using dynamic programming. The optimal string match works by calculating a score for the match with respect to the reference such that identical labels match with score 0, a label insertion carries a score of 7, a deletion carries a score of 7 and a substitution carries a score of 10. The optimal string match is the label alignment which has the lowest possible score. Once the optimal alignment has been found, the number of substitution errors (S), deletion errors (D) and insertion errors (I) can

be calculated. The percentage correct is then:

$$PercentCorrect = \frac{N-D-S}{N} \times 100\% \tag{3.24}$$

where N is the total number of labels in the reference transcriptions. this measure ignores insertion errors. For many purposes, the percentage accuracy defined as:

$$PercentCorrect = \frac{N-D-S-I}{N} \times 100\% \tag{3.25}$$

This is a more representative figure of recogniser performance. HTK outputs both of the above measures.

3.1.3.2 Creating Mono-phone HMMs

First, a flat global speech means and variances will be calculated through the training data features. For this purpose, a prototype HMM model is defined. The parameters of this model are not important. It is just used to define the three state right to left topology of the model. In this *proto* file, the selected feature extraction method information should be incorporated. For instance, a sample *proto* file for MFCC is shown in Figure 3.6-a. This proto file is used to build the flat start HMM model which is an estimate of global mean and variance. Two different files are generated at this step. Figure 3.6-b shows these 2 files.

```
<BeginHMM>
 <NumStates> 5
 <State> 2
    <Mean> 39
      0.0 0.0 0.0 ...
    <Variance> 39
      1.0 1.0 1.0 ...
 <State> 3
    <Mean> 39
      0.0 0.0 0.0 ...
    <Variance> 39
      1.0 1.0 1.0 ...
 <State> 4
    <Mean> 39
      0.0 0.0 0.0 ...
    <Variance> 39
      1.0 1.0 1.0 ...
 <TransP> 5
  0.0 1.0 0.0 0.0 0.0
  0.0 0.6 0.4 0.0 0.0
  0.0 0.0 0.6 0.4 0.0
  0.0 0.0 0.0 0.7 0.3
  0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

(a)

macros

```
~o
   <VecSize> 39
   <MFCC_0_D_A>
~v "varFloor1"
   <Variance> 39
      0.0012 0.0003 ...
```

hmmdefs

```
~h "aa"
   <BeginHMM> ...
   <EndHMM>
~h "eh"
   <BeginHMM> ...
   <EndHMM>
... etc
```

(b)

Figure 3.6: (a) A sample proto file (b) Typical *hmmdefs* and *macros* files

After re-estimating these models parameters for three or four times, we use them to build a separate initial HMM model for each phoneme of the language.

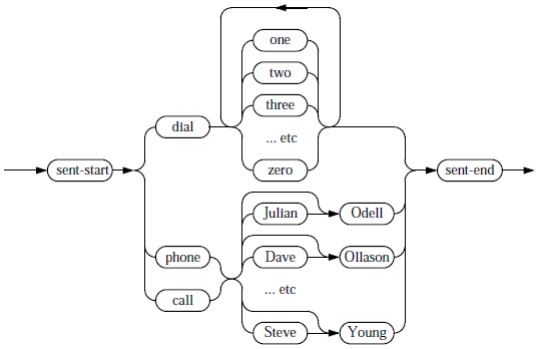
The dictionary contains multiple pronunciations for some words, while the phone level file that has been build so far, considers just the first form of the pronunciation of such words in the dictionary. At this step, a new phone level transcription will be built, that considers all possible pronunciations for this kind of words. Therefore, this step is called *realigning*. Then, the re-estimation of the HMM parameters should be done again for two times using the new aligned mlf file. The first set of the simple HMM single Gaussian models for mono-phones is ready to be tested.

Experimental Results for Mono-phone Simple HMMs:

As mentioned before, different feature extraction methods that are supported by HTK, are used and in each case mono-phone HMM models are built. Then these models are tested in recognition phase. For performing a simple recognition task, a context free grammar may be used which is easier for recognition than a test set from the whole language. In a context free grammar, a limited number of sentences are used which are all listed in a corresponding dictionary. HTK provides a grammar definition language for specifying such a simple task grammars. It consists of a set of variable definitions followed by a regular expression describing the words to recognize. For example Figure 3.7-a shows a grammar of an English voice dialing application. where the vertical bars denote alternatives, the square brackets denote optional items and the angle braces denote one or more repetitions. The complete grammar can be depicted as a network shown in Figure 3.7-b .

```
$digit = ONE | TWO | THREE | FOUR | FIVE |
         SIX | SEVEN | EIGHT | NINE | OH | ZERO;
$name  = [ JOOP ] JANSEN |
         [ JULIAN ] ODELL |
         [ DAVE ] OLLASON |
         [ PHIL ] WOODLAND |
         [ STEVE ] YOUNG;
( SENT-START ( DIAL <$digit> | (PHONE|CALL) $name) SENT-END )
```

(a)



(b)

Figure 3.7: context free grammar (a) a simple English voice dialing application (b) the corresponding word network

We have used a test set which is a Domotica context free grammar with 150 test recording files and a dictionary of the size 30. The sentences are simply about the home devices. It consists of audio files, transcription files, the word-net file, the script file of wave file addresses and a dictionary. Figure 3.8 illustrates the different components in grammar recognition task.

For doing the recognition task, the final HMM model, the dictionary of the test set and also the list of Spanish phonemes. HTK provides complete information to evaluate the recognition results. In the out put table, the percentage of correct recognition of sentences and words will be reported. It lists total number of correctly recognized options (H), the total number of all options (N), the number of deletion errors (D), the number of substitutions (S) and the insertion error (I). The *%corr* ignores the insertion errors while the *ACC* considers them and therefore it is smaller. Table 3.1, presents the recognition results for the five feature extraction methods.

As illustrated in Figure 3.9 , the PLP and MFCC methods offer the much higher performances in

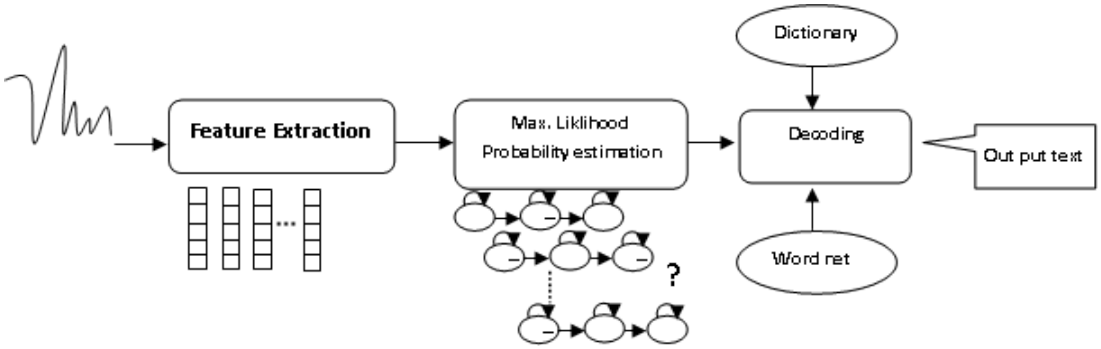


Figure 3.8: grammar recognition process

Table 3.1: Results of Recognition for different feature extraction methods

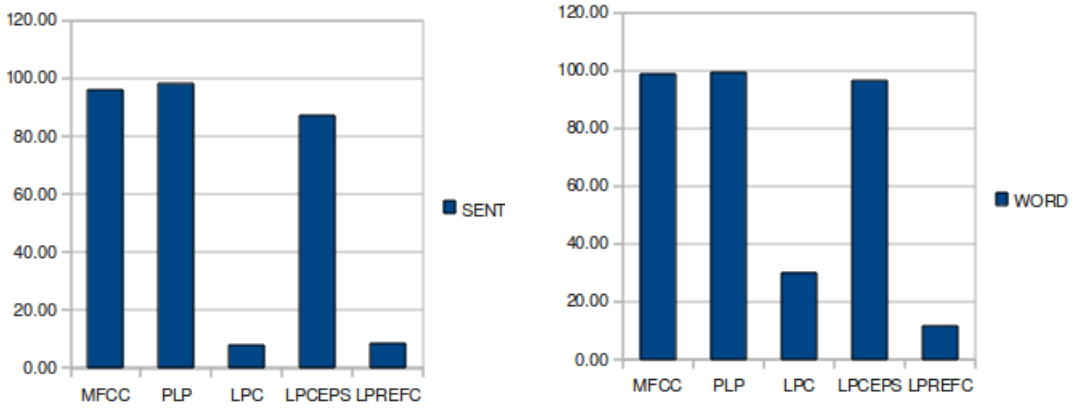
	SENT				WORD						
	%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
MFCC	96.00	144	6	150	98.92	98.92	547	0	6	0	553
PLP	98.00	147	3	150	99.46	99.46	550	0	3	0	553
LPC	8.00	12	138	150	30.20	26.04	167	126	260	23	553
LPCEPSTRA	87.33	131	19	150	96.38	96.38	533	0	20	0	553
LPREFC	8.67	13	137	150	11.57	11.39	64	234	255	1	553

recognition task than other methods. Therefore, for developing the Spanish recognizer we will use the standard MFCC method.

3.1.3.3 Creating Tri-Phone HMMs

In order to build the tri-phone HMM models, its necessary to start from the initial set of HMM models that has been built which has not yet been realigned. The different steps are:

- **Realigning-** The realigning process will be done with the same idea as explained before to consider words with different pronunciation.
- **Fixing the Silence Models-** while a HMM for the silence model *sil* has been generated before, the idea here is to make the model more robust by allowing individual states to absorb the various impulsive noises in the training data. Therefore the *sil* model will be modified as shown in Figure 3.10 . Also, a 1 state short pause *sp* model is created. This is a tee-model which has a direct transition from entry to exit node. This *sp* has its emitting state tied to the central state of the silence model. After these modifications, again HMM parameters are re-estimated.
- **Triphone model-list and training label file creation-** A set of training cross word labels and a tri-phones model list will be created. Figure 3.11 shows the tri-phones list and the training data mlf file of tri-phones.
- **Initial models-** An initial set of triphone HMM models are created by cloning the mono-phone HMM models and tying their transition matrices. These models parameters are re-estimated for 2 times.
- **Building state-clustered models-** The outcome of the previous stage is a set of triphone HMMs with all tri-phones in a phone set sharing the same transition matrix. When estimating these



(a) (b)

Figure 3.9: Comparison of feature extraction methods (a) Sentence performance(b) Word performance

models, many of the variances in the output distributions will have been floored since there will be insufficient data associated with many of the states. The last step in the model building process is to tie states within triphone sets in order to share data and thus be able to make robust parameter estimates. In the previous step, all members of a set of transition matrices were explicitly tied together. However, the choice of which states to tie requires is a bit more complex since the performance of the recognizer depends crucially on how accurate the state output distributions capture the statistics of the speech data.
In HTK there are two mechanisms which allow states to be clustered and then each cluster tied. The first is data-driven and uses a similarity measure between states. The second uses decision trees and is based on asking questions about the left and right contexts of each triphone. The decision tree attempts to find those contexts which make the largest difference to the acoustics and which should therefore distinguish clusters.
We will use the decision trees. The created tri-phones models will be state clustered. This allows the synthesis of unseen tri-phones and thus makes it possible to produce cross-word context dependent systems. The clustering is used by sharing only possible within the same state of the same base phone. However the clustering proceeds in a top down manner by initially grouping all contexts and then splitting on the basis of questions about context. The questions used are chosen to maximize the likelihood of the training data whilst ensuring that each tied-state has a minimum occupancy. Figure 3.12 shows a part of a Spanish decision tree. For a triphone system, it is necessary to include questions referring to both the right and left contexts of a phone. The questions should progress from wide, general classifications (such as consonant, vowel, nasal, diphthong, etc.) to specific instances of each phone. The decision tree is used to build both the set of tri-phones of the training data and all of the new previously unseen tri-phones. Finally, the HMM parameters are re-estimated for another tow rounds.

- **Multiple mixture state-clustered tri-phones-**Multiple mixture models for the state-clustered cross-word tri-phones are built for 2 mixture, 4 mixture, 16 mixture, 32 mixture and then 64 mixture models, and at each stage 4 iterations for parameter re-estimation will be performed.

Experimental Results for Tri-phone HMMs:

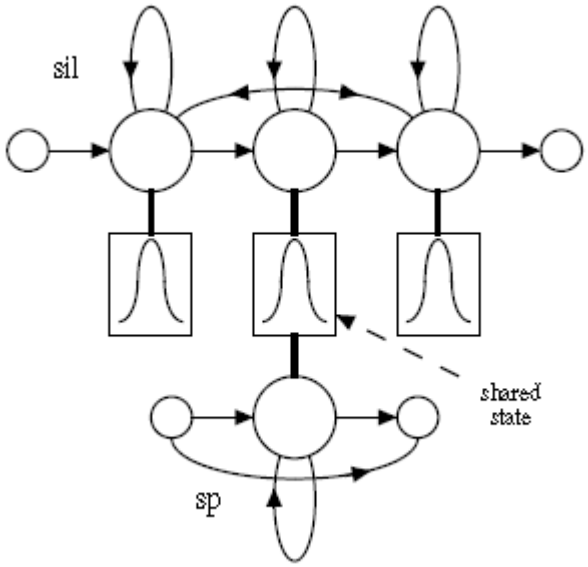


Figure 3.10: Silence Models

Two sets of experiments have been performed with domotica word grammars, one in poor quality and another in high quality. Both Single Gaussian models (SGM) and mixture Gaussian models (GMM) of tri-phones HMMs has been tested. Table 3.2 and 3.3 present the recognition results for poor quality and high quality test sets. Also Figure 3.13 and 3.14 show the comparison of the results in each set. Comparing the results does not show a homogeneous flow. One important problem that affects seriously our models is the insufficient training data. It has been proved that having enough number of training data in a large corpus plays an important role to achieve strong acoustic models, while the Albayzin corpus does not provide enough training data. This lead to have warnings during the splitting of Gaussian models to build mixture models. The warnings indicate that there is not enough training examples and model is not updated. Therefor, our test results suffer from the consequences of this fact. Clearly, in both poor quality test set and high quality test set, the performances of different mixture models are more or less the same. Due to the warnings during the mixture Gaussians modeling, we believe that the 2Gaussian mixture model is a better model with lower computation cost than other higher mixtures which have been poorly modeled.

Table 3.2: Recognition Results for Poor Quality Test Grammar

	SENT				WORD						
	%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
SGM	60.00	27	18	45	83.45	78.62	118	4	23	4	145
2GMM	68.89	31	14	45	84.83	82.76	123	4	18	3	145
4GMM	66.67	30	15	45	83.45	80.69	121	4	20	4	145
16GMM	77.78	35	10	45	86.90	84.83	126	4	15	3	145
32GMM	66.67	30	15	45	82.07	78.62	119	4	22	5	145
64GMM	82.22	37	8	45	88.28	86.21	128	4	13	3	145

```
sil
sil-f+r
f-r+a
r-a+n
a-n+T
n-T+j
T-j+a
j-a+s
sp
a-s+w
s-w+i
w-i+T
i-T+a
T-a+i
a-i+u
i-u+N
u-N+g
N-g+r
g-r+i
r-i+a
.
.
.
```

(a)

```
#!MLF!#
"*/AAFA0001.SES.lab"
sil
sil-f+r
f-r+a
r-a+n
a-n+T
n-T+j
T-j+a
j-a+s
sp
a-s+w
s-w+i
w-i+T
i-T+a
T-a+i
sp
a-i+u
sp
i-u+N
.
.
.
```

(b)

Figure 3.11: (a) tri-phones list (b) training tri-phone transcription mlf file.

```
QS 'R_NonBoundary' { "*+*" }
QS 'R_Silence' { "*+sil" }
QS 'R_Stop' { "*+g","*+k","*+d","*+t","*+b","*+p" }
QS 'R_Nasal' { "*+N","*+J","*+n","*+m" }
QS 'R_Fricative' { "*+tS","*+f","*+z","*+s" }
QS 'R_Aproximante' { "*+G","*+D","*+B" }
QS 'R_Liquid' { "*+rr","*+L","*+r","*+l" }
QS 'R_Vowel' { "*+i","*+u","*+o","*+e","*+a" }
QS 'R_C-Front' { "*+B","*+f","*+m","*+b","*+p" }
```

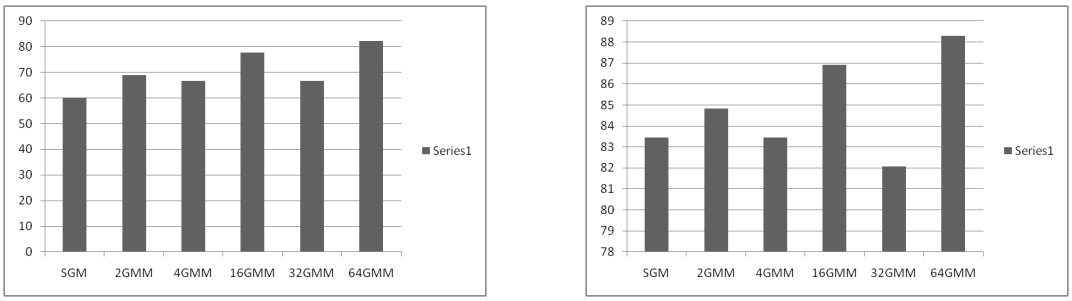
Figure 3.12: A Spanish language decision tree

3.1.3.4 Adapting the HMMs

HTK supports both supervised adaptation, where the true transcription of the data is known and unsupervised adaptation where the transcription is hypothesized. In HTK, supervised adaptation is performed offline using maximum likelihood linear transformations like MLLR or CMLLR or maximum a-posteriori (MAP) techniques to estimate a series of transforms or a transformed model set, that reduces the mismatch between current model set and the adaptation data. Unsupervised adaptation is provided, using just linear transformations. We are going to implement an offline supervised adaptation using MLLR.

Table 3.3: Recognition Results for High Quality Test Grammar

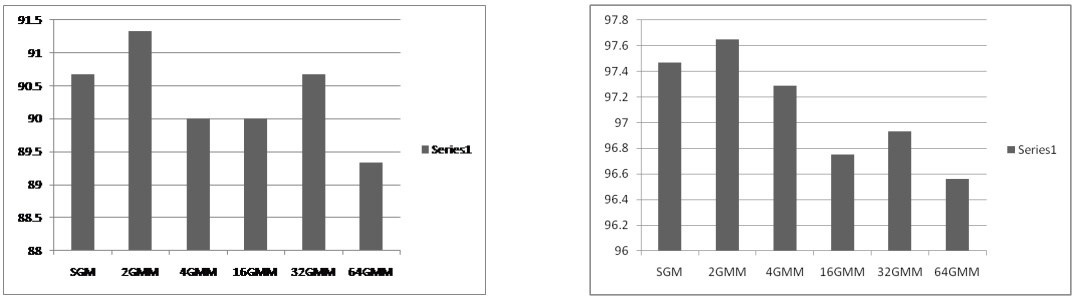
	SENT				WORD						
	%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
SGM	90.67	136	14	150	97.47	97.47	539	0	14	0	553
2GMM	91.33	137	13	150	97.65	97.65	540	0	13	0	553
4GMM	90.00	135	15	150	97.29	97.29	538	0	15	0	553
16GMM	90.00	135	15	150	96.75	96.75	532	2	16	0	553
32GMM	90.67	136	14	150	96.93	96.93	536	2	15	0	553
64GMM	89.33	134	16	150	96.56	96.56	534	2	17	0	553



(a) (b)

Figure 3.13: Comparison of poor quality test set results for different Gaussian models (a) Sentence performance(b) Word performance

- **Data Preparation -** The amount of adaptation data required will normally be found empirically, but a performance improvement should be observable after just 30 seconds of speech. In this case, around 20 utterances should be sufficient. Assuming to have the script files of the source and output files for the adaptation and test data respectively, then, both sets of speech will be coded using Melfrequency Cepstral Analysis. The final stage of preparation involves generating context dependent phone transcriptions of the adaptation data and word level transcriptions of the test data for use in adapting the models and evaluating their performance. To minimize the problem of multiple pronunciations the phone level transcriptions of the adaptation data can be obtained using a forced alignment of the adaptation data.
- **Generating the Transforms -** Regression class trees can be used to dynamically specify the number of transformations to be generated, or the number may be pre-determined using a set of base-classes. The base classes and the regression class tree is built and stored along with a set of baseclasses at this step.
  HTK supports static adaptation, where all the adaptation data is processed in a single block. We use MLLR as the form of linear adaptation. First, a global adaptation is performed using the 2-Gaussian mixture model. Then, this global transformation is used as an input transformation, to transform the model set, producing better frame/state alignments which are then used to estimate a set of more specific transforms, using a regression class tree.
- **Evaluation of the Adapted System -** To evaluate the performance of the adaptation, a word grammar test set is used. All audio files are recorded by a specific user. The recognition is performed using the models before and after adaptation. Table 3.4 and 3.5 show the results of



(a) (b)

Figure 3.14: Comparison of high quality test set results for different Gaussian models (a) Sentence performance(b) Word performance

these tests. In contrast to our expectation, obtained results before and after the recognition task are the same with high performance. Analyzing the information of the recognition show that there was just one substitution error and the program successfully recognized all the sentences. We interpret these results to our limited test sets that could be accurately recognized.

Table 3.4: Recognition results using 2-Gaussian HMM model before adaptation

SENT				WORD						
%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
95.00	19	1	20	98.53	98.53	67	0	1	0	68

Table 3.5: Recognition results using adapted 2-Gaussian HMM model

SENT				WORD						
%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
95.00	19	1	20	98.53	98.53	67	0	1	0	68

3.2 Language Modeling

In 2.1.3, the basic idea of Language modeling has been explained. As discussed before, the first step in language modeling is to form N-gram files. An N-gram is a sequence of N symbols (usually words) and an N-gram (LM) is used to predict each symbol in the sequence given its $N-1$ predecessors. It is built on the assumption that **the probability of a specific N-gram occurring in some unknown test text can be estimated from the frequency of its occurrence in some given training text** [3]. As illustrated in Figure 3.15, Language modeling is a three stage process. Firstly, the training text is scanned and its N-grams are counted and stored in a database of gram files. In the second stage some words may be mapped to an out of vocabulary class or other class mapping may be applied, and then in the final stage the counts in the resulting gram files are used to compute N-gram probabilities which are stored in the language model file. Lastly, the goodness of a language model can be estimated by the perplexity measure on a previously unseen test set. In general, the better a language model then the lower its test-set perplexity.

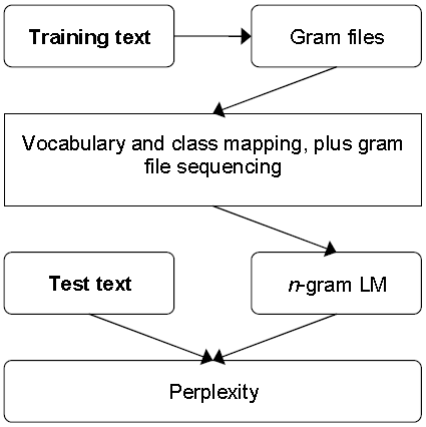


Figure 3.15: Language modeling Steps in HTK

3.2.1 Data Preparation

The first stage of language modeling is data preparation. The text data used in our work is from wikipedia Spanish articles.

Text Conditioning- Text conditioning stage is necessary to transform the raw text into its most common and useful representation (e.g. number conversions, abbreviation expansion and punctuation filtering).

Text Partitioning- Then, the conditioned texts have been partitioned into training and test material (in our case we have a 131.5 MB training and 48 KB test data) and reside in the train and test subdirectories respectively.

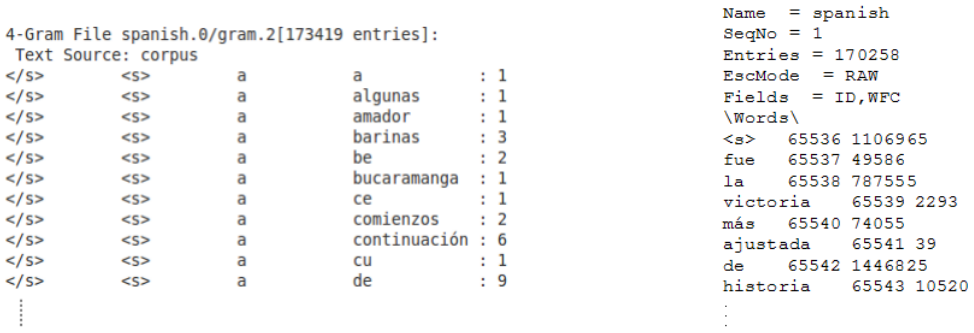
Sentence Start and End Labeling- When training a language model you need to include sentence start and end labeling because the tools cannot otherwise infer this. Its not mandatory to have the entire input text on a single line. The default sentence start and sentence end tokens of <s> and </s> are used – if different tokens are used for , they should be passed in configuration parameters to the HTK tools. Figure 3.16 shows a part of prepared training data for language modeling.

```
<s> fue la victoria más ajustada de la historia de la selección americana
de baloncesto desde que empezó a utilizar jugadores nba y un triple de
jasikevicius en el último segundo pudo haberles eliminado </s>
<s> en los juegos olímpicos de atenas dos mil cuatro consiguió
desquitarse de la derrota de sidney anotando veintiocho puntos que fueron
claves para la victoria contra los estados unidos </s>
<s> olimpija  mil novecientos noventa y nueve menos dos mil </s>
<s> tres euroligas  dos mil tres punto y coma dos mil cuatro y dos mil
cinco </s>
:
```

Figure 3.16: prepared wikipedia training text for Spanish language modeling

Generating Word Map and Gram Files- The training text is scanned and a preliminary set of sorted N-gram files are generated. Also, the HTK tools maintain a cumulative word map to which every new word is added and assigned a unique id. This means that you can add future N-gram files without having to rebuild existing ones so long as you start from the same word map, thus ensuring that each

id remains unique. Figure 3.17-a shows a part of a gram file and 3.17-b displays the generated word map.



```
4-Gram File spanish.0/gram.2[173419 entries]:
 Text Source: corpus
</s>        <s>         a           a           : 1
</s>        <s>         a           algunas     : 1
</s>        <s>         a           amador      : 1
</s>        <s>         a           barinas     : 3
</s>        <s>         a           be          : 2
</s>        <s>         a           bucaramanga : 1
</s>        <s>         a           ce          : 1
</s>        <s>         a           comienzos   : 2
</s>        <s>         a           continuación : 6
</s>        <s>         a           cu          : 1
</s>        <s>         a           de          : 9
```

(a)

```
Name  = spanish
SeqNo = 1
Entries = 170258
EscMode  = RAW
Fields  = ID,WFC
\Words\
<s>    65536 1106965
fue    65537 49586
la     65538 787555
victoria     65539 2293
más    65540 74055
ajustada     65541 39
de     65542 1446825
historia     65543 10520
```

(b)

Figure 3.17: (a) a grammar file (b) the initial generated word map

3.2.2 Mapping Out of Vocabulary (OOV) words

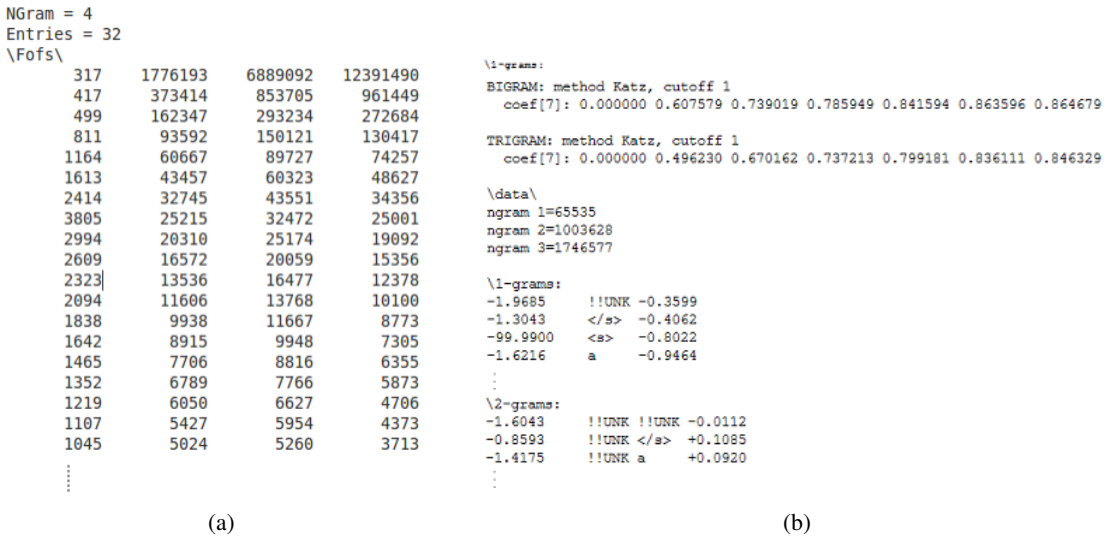
An important step in building a language model is to decide on the system's vocabulary. We have supplied a word list of 64 KB of words. Once the system's vocabulary has been specified, all out-of-vocabulary (OOV) words will be filtered out . To achieve this, the 64KB word list is used as a special case of a class map which maps all OOVs into members of the "unknown" word class. The unknown class symbol defaults to !!UNK.

All N-grams containing OOVs will be extracted from the input files and the OOV words mapped to the unknown symbol. A new word map containing the new class symbols (!!UNK in this case) and only words in the vocabulary will be created. However, any N-grams containing OOV words will be discarded since these are no longer in the word map.

3.2.3 Language Model Generation

The first step in generating a language model is to produce a frequency of frequency (FoF) table for the chosen vocabulary list. This file can generate before building the LM and then the result can be passed to LM phase. This has only a negligible effect on computation time, but the result is interesting in itself because it provides useful information for setting cut-offs. Cut-offs are where we choose to discard low frequency events from the training text – this is for decreasing the model size, or ignoring unimportant events. Figure 3.18-a shows a FOF table. A FoF file contains a list of the number of times that an N-gram occurred just once, twice, three times, . . . , n times. Its format is similar to a word map file. The data part contains a list of the uni-grams, bi-grams, . . . , N-grams occurring exactly $\boldsymbol{k}$ times, where $\boldsymbol{k}$ is the number of the row of the table – for example in Figure 3.18-a, the first row shows the number of N-grams occurring exactly one time, as shown, there are 64832 uni-grams that occurred just one time in the text data.

Then, actual language model will be built for uni-gram, bi-gram and trigram. Figure 3.18-b, displays a LM with cut-off one in this file uni-gram, bi-gram and trigram probabilities are listed.



```
NGram = 4
Entries = 32
\Fofs\
       317    1776193    6889092   12391490
       417     373414     853705     961449
       499     162347     293234     272684
       811      93592     150121     130417
      1164      60667      89727      74257
      1613      43457      60323      48627
      2414      32745      43551      34356
      3805      25215      32472      25001
      2994      20310      25174      19092
      2609      16572      20059      15356
      2323      13536      16477      12378
      2094      11606      13768      10100
      1838       9938      11667       8773
      1642       8915       9948       7305
      1465       7706       8816       6355
      1352       6789       7766       5873
      1219       6050       6627       4706
      1107       5427       5954       4373
      1045       5024       5260       3713
```

(a)

```
\1-grams:
BIGRAM: method Katz, cutoff 1
  coef[7]: 0.000000 0.607579 0.739019 0.785949 0.841594 0.863596 0.864679

TRIGRAM: method Katz, cutoff 1
  coef[7]: 0.000000 0.496230 0.670162 0.737213 0.799181 0.836111 0.846329

\data\
ngram 1=65535
ngram 2=1003628
ngram 3=1746577

\1-grams:
-1.9685     !!UNK -0.3599
-1.3043     </s>  -0.4062
-99.9900    <s>   -0.8022
-1.6216     a     -0.9464

\2-grams:
-1.6043     !!UNK !!UNK -0.0112
-0.8593     !!UNK </s>  +0.1085
-1.4175     !!UNK a     +0.0920
```

(b)

Figure 3.18: (a) a FOF table (b) a language model

3.2.3.1 Testing the LM perplexity

Once the language models have been generated, their "goodness" can be evaluated by computing the perplexity of previously unseen text data. This won't necessarily tell us how well the language model will perform in a speech recognition task because it takes no account of acoustic similarities or the vagaries of any particular system, but it will reveal how well a given piece of test text is modeled by your language model. As mentioned before, the low perplexity shows a better LM.

Table 3.6 and 3.7 list the results on our test set for bi-gram and trigram LMs. For each test, the first part of the result gives general information such as the number of utterances and tokens encountered, words predicted and OOV statistics. The second part of the results gives explicit access statistics for the back off model. For example, in the bi-gram model test, the total number of words predicted is 7963. From this number, 88.9% were found as explicit bi-grams in the model, 10.9% were computed by backing off to the respective uni-grams and 0.2% were simply computed as uni-grams by shortening the word context. In the same way, for the trigram model test, the total number of words predicted is 7895. From this number, 56.8% were found as explicit trigrams in the model, 22.0% were computed by backing off to the respective bi-grams and 21.3% were simply computed as bi-grams by shortening the word context. The perplexity for trigram test is smaller than bi-gram which shows better modeling. Clearly, the OOV rate is very low as we have used a strong word-list (dictionary) of 64 KB.

Table 3.6: Bi-gram LM Test results

2-gram						
perplexity	var.	utterances	words predicted	num. tokens	OOV	OOV rate
176.5502	11.9621	438	7963	8561	80	0.98%
Lang model	requested	exact	backed	n/a	mean	stdev
bi-gram	7963	88.9%	10.9%	0.2%	-5.17	3.46
trigram	0	0.0%	0.0%	0.0%	0.00	0.00

Table 3.7: Trigram LM Test Results

3-gram						
perplexity	var.	utterances	words predicted	num. tokens	OOV	OOV rate
117.7408	13.3613	438	7895	8561	80	0.98%
Lang model	requested	exact	backed	n/a	mean	stdev
bi-gram	3414	74.3%	25.3%	0.4%	-6.94	3.82
trigram	7895	56.8%	22.0%	21.3%	-4.77	3.66

These perplexity tests do not include the prediction of words from context which includes OOVs. To include such N-grams in the calculation another test for trigram has been performed. Table 3.8 shows the results for this test. As clear, the number of predicted words has increased from 7895 to 8043.

Table 3.8: Trigram LM Test Results including predictions from OOV

3-gram						
perplexity	var.	utterances	words predicted	num. tokens	OOV	OOV rate
119.2902	13.3565	438	8043	8561	80	0.98%
Lang model	requested	exact	backed	n/a	mean	stdev
bi-gram	3473	74.3%	25.2%	0.4%	-6.95	3.83
trigram	8043	56.8%	22.1%	21.1%	-4.78	3.65

3.2.3.2 Count-Based Language Models

The language models generated in the previous section are static in terms of their size and vocabulary. For example, in order to evaluate a trigram model with cut-offs 2 (bi-gram) and 2 (trigram) the we should rebuild the bi-gram and trigram stages of the model. When large amounts of text data are used this can be a very time consuming operation. The HTK toolkit provides the capabilities to generate and manipulate a more generic type of model, called a count-based models, which can be dynamically adjusted in terms of its size and vocabulary. we can set cut-off parameters which control the initial size of the model, but if so then once the model is generated only higher cut-off values may be specified in the subsequent operations. For example during generating a full trigram model no intermediate file like the uni-gram and bi-gram models files are produced. The generated model can be used in perplexity tests and different model sizes can be obtained by specifying new cut-off values. Table 3.9 lists the perplexity test results for a trigram LM with cut-offs (2,2). Also Table 3.10 shows the results for a trigram LM with cut-offs (3,3). Comparing the results from two tables show that the perplexity in smaller cut-off is better but worth than the previous models with the defaults cut-off of one.

Table 3.9: count-based Trigram LM Test Results with (2,2) cut-off

perplexity	var.	utterances	words predicted	num. tokens	OOV	OOV rate
130.4351	13.8806	438	8043	8561	80	0.98%
Lang model	requested	exact	backed	n/a	mean	stdev
bi-gram	3814	71.7%	26.8%	1.4%	-6.92	3.84
trigram	8043	52.6%	21.5%	25.9%	-4.87	3.73

Table 3.10: count-based Trigram LM Test Results with (3,3) cut-off

perplexity	var.	utterances	words predicted	num. tokens	OOV	OOV rate
137.8614	14.2183	438	8043	8561	80	0.98%
Lang model	requested	exact	backed	n/a	mean	stdev
bi-gram	4022	69.7%	28.0%	2.3%	-6.91	3.85
trigram	8043	50.0%	21.6%	28.4%	-4.93	3.77

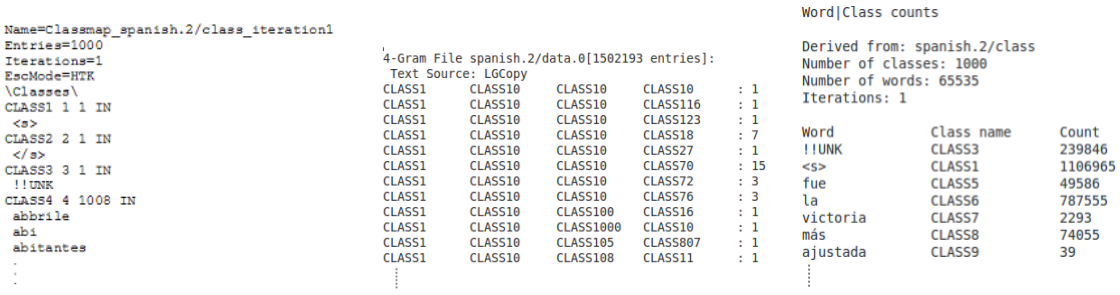
3.2.3.3 Class-Based Language Models

The idea of class-based language modeling has been explained in Section 2.1.3.3. It is similar to a word-based N-gram as both of them store probabilities N- tuples of tokens – except in the class model case these tokens consist of word classes instead of words. Thus building a class model involves constructing class N-grams. A second component of the model calculates the probability of a word given each class. This has be formulated in 2.20 as:

$$P(W) = \prod_{i=1}^{M} (P(w_i|c_i)\,P(c_i|c_{i-1}, c_{i-2}..., c_{i-N+1}))$$

The HTK tools only support deterministic class maps, so each word can only be in one class. Class language models use a separate file to store each of the two components – the word-given-class probabilities $P(w_i|c_i)$ and the class N-grams $P(c_i|c_{i-1}, c_{i-2}..., c_{i-N+1})$– as well as a third file which points to the two component files. Alternatively, the two components can be combined together into a standalone separate file. We are going to list class-based LM modeling steps:

- **class map**- Before a class model can be built it is necessary to construct a class map which defines which words are in each class. In the case of large number of classes, the execution time is long and measured in hours. In many systems class models are combined with word models to give further gains. a decision should to be made as to how many separate classes are required. This depends on the modeling purpose and whether it will be purely interpolated with a word model or not. In the latter case, for example, a sensible number of classes is often around the 1000 mark when using a 64K word vocabulary. Figure 3.19-a shows the class map file.



(a) (b) (c)

Figure 3.19: (a) a class map file (b) class N-grams component (c) Word-given- Class Component

- **Class-Ngram Components-** the class N-grams component is built. Figure 3.19-b shows this file.
- **Word-given- Class Component-** the word-given-class component is built. Figure 3.19-c shows this file.
- **Class N-gram Language Model-** Given the two language model components we can now link them together to make our overall class N-gram language model.

The class-based language model has been tested. The class trigram model performs worse than the word trigram (with higher perplexity). Results are shown in Table 3.11. But this is not a surprise since this is true of almost every reasonably-sized test set – the class model is less specific. Interpolating the two often leads to further improvements. Table 3.12 presents the improved results after interpolation.

Table 3.11: Class based LM Test Results

3-gram						
perplexity	var.	utterances	words predicted	num. tokens	OOV	OOV rate
137.4564	12.0400	438	7895	8561	80	0.98%
Lang model	requested	exact	backed	n/a	mean	stdev
bi-gram	1896	94.3%	5.7%	0.0%	-5.80	2.30
trigram	7895	76.0%	15.3%	8.7%	-4.92	3.47

Table 3.12: Class based LM Test Results after interpolation

3-gram						
perplexity	var.	utterances	words predicted	num. tokens	OOV	OOV rate
106.6306	11.5383	438	8043	8561	80	0.98%
Access statistics for word trigram LM						
Lang model	requested	exact	backed	n/a	mean	stdev
bi-gram	3473	74.3%	25.2%	0.4%	-6.95	3.83
trigram	8043	56.8%	22.1%	21.1%	-4.78	3.65
Access statistics for class trigram LM						
Lang model	requested	exact	backed	n/a	mean	stdev
bi-gram	1931	94.3%	5.7%	0.0%	-5.82	2.31
trigram	8043	76.0%	15.5%	8.6%	-4.94	3.47

3.3 Discriminative Training

HTK supports discriminative training for both the Maximum Mutual Information (MMI) and Minimum Phone Error (MPE) training criteria [3]. In both cases the aim is to estimate the HMM parameters in such a way as to approximately reduce the error rate on the training data. Hence the using criteria is not only the actual word-level transcription of the training data but also "confusable" hypotheses which increases the language model / acoustic model log likelihoods. The form of MMI criterion to be maximized may be expressed in Equation 2.28 and 2.32 as follows:

$$
\begin{aligned}
F_{mmi}(\lambda) &= \frac{1}{R}\sum_{r=1}^{R}\log\left(P\left(w_{ref}^{(r)}|x^{(r)};\lambda\right)\right) \\
&= \frac{1}{R}\sum_{r=1}^{R}\log\left(\frac{P\left(x^{(r)}|w_{ref}^{(r)};\lambda\right)P\left(w_{ref}^{(r)}\right)}{\sum_{w}P\left(x^{(r)}|w;\lambda\right)P(w)}\right)
\end{aligned}
$$

Thus, the average log-posterior of the reference, w_{ref} , is maximized. Here the summation for w is over all possible word sequences. In practice this is restricted to the set of confusable hypotheses, which will be defined by a lattice. Also, The MPE training criterion which is an example of minimum Bayes' risk training has been defined in 2.2.1.2. The general expression to be minimized has been expressed in Equation 2.32 as:

$$
F_{mpe}(\lambda) = \frac{1}{R}\sum_{r=1}^{R}\sum_{w}P\left(w|x^{(r)};\lambda\right)L\left(w,w_{ref}^{(r)}\right)
$$

Where $L\left(w,w_{ref}^{(r)}\right)$ is the "loss" between the hypothesis w and the reference, w_{ref} . In general, there are various forms of loss function that may be used. However, in MPE training, the loss function is measured in terms of the the Levenshtein edit distance between the phone sequences of the reference and the hypothesis. In HTK, rather than minimizing this expression, the normalized average phone accuracy is maximized. This may be expressed as

$$
\hat{\lambda} = \arg\max_{\lambda}\left\{1 - \frac{1}{\sum_{r=1}^{R}Q^{r}}F_{mpe}(\lambda)\right\} \tag{3.26}
$$

where Q_r is the number of phones in the transcription for training sequence r.

During the implementation, the language model scores, including the grammar scale factor are combined into the acoustic models to yield a numerator acoustic model, M_r^{num} , and a denominator acoustic model, M_r^{den} for utterance r. In this case the MMI criterion can be expressed as:

$$
F_{mmi}(\lambda) = \sum_{r=1}^{R}\log\left(\frac{P\left(x^{(r)}|M_r^{num}\right)}{P\left(x^{(r)}|M_r^{den}\right)}\right) \tag{3.27}
$$

and the MPE criterion is expressed as:

$$
F_{mpe}(\lambda) = \sum_{r=1}^{R}\sum_{w}\left(\frac{P\left(x^{(r)}|M_w\right)}{P\left(x^{(r)}|M_r^{den}\right)}\right)L\left(w,w_{ref}^{(r)}\right) \tag{3.28}
$$

where M_w is the acoustic model for hypothesis H. In practice, approximate forms of the MMI and normalized average phone accuracy criteria are optimized.

For both MMI and MPE training the estimation of the model parameters are based on variants of the Extended Baum-Welch (EBW) algorithm as mentioned before in 2.2.1.2. More details about the estimation of the means and covariance matrices could be found in [3].

3.3.1 Lattice-Based Discriminative Training

For both the MMI and MPE training criteria a set of possible hypotheses for each utterance must be considered. To get these confusable hypotheses the training data must first be recognized. HTK uses

Lattice Based Discriminative Training, in which word lattices are first created and then these lattices are used for all iterations of discriminative training. To make this operation more efficient, the times of the HMM/phone boundaries are also marked in the lattices. This creates so-called phone-marked lattices. For each utterance used for discriminative training, two lattices need to be created. The first is a phone-marked lattice that represents the correct word sequence (also known as a "numerator" lattice). The second is a phone-marked lattice for competing hypotheses: the "denominator" lattice. These names derive from the MMI objective function, but the same phone-marked lattices are also required for MPE. The numerator lattice is found by generating phone-level alignments in lattice form the correct word level transcription, while the denominator lattice uses a phone-marked form of the lattice representing confusable hypotheses.

3.3.2 Implementing Discriminative Training

Discriminative Training approach to HMM parameter estimation is a further refinement to the acoustic models. In order to perform discriminative training over training data, a cross-word triphone HMM model set, a pronunciation dictionary and lattices are required.

1. **Generation of Initial Maximum Likelihood Models** - A cross-word triphone set of HMMs must be initially trained using standard maximum likelihood estimation. We will use the 2 Gaussian mixture models that we have generated in 3.1.3.3 because they offer higher performance.
2. **Training Data LM Creation** - A weak language model, i.e. a uni-gram or bi-gram, must be created for use in discriminative training. To do this, the transcripts of the acoustic training data are used. It is essential that the vocabulary includes (at least) the words in the correct word-level transcripts. For language modeling, the same steps in 3.2 are used. The cut-off parameter should be selected so that a bi-gram of a suitable size is obtained.
3. **Word Lattice Creation -** Two sets of "phone-marked" lattices, called the denominator and numerator lattices, are required for discriminative training. The first stage in generating these phone-marked lattices is to produce word lattices. The denominator word lattices represent the set of most likely word sequences for a particular training sentence according to Equation 2.32. Numerator word-level lattices include language model log probabilities.
4. **Phone Marking of Numerator and Denominator Lattices -** The word-level lattices are further processed using the initial models and the speech data to produce the phone-marked lattices used for discriminative training. Before the phone-marked denominator lattices can be created, the denominator word lattices must be made deterministic. A check should be made on the phone-marked lattices to ensure that are created that all expected lattices are exist before continuing. If some have failed the pruning parameters can be altered or a new list of successful training files created for subsequent stages.
5. **Generating Discriminatively Trained Models** - Having generated the required numerator and denominator phone-marked lattices, then its possible to discriminatively train the HMMs. A number (typically 4-8) of iterations of the Extended Baum-Welch (EBW) algorithm are run. There are a number of configuration options that allow the choice of objective function to be varied; the amount and type of smoothing; learning rate in EBW updates etc. By changing the parameters, the type of discriminative training like MMI or MPE could be selected.

3.4 Decoding

We have prepared acoustic model, language model and also discriminatively trained models. In this step, speech recognition on test recordings from wikipedia ill be performed. The training corpus used in language modeling was also from wikipedia Spanish transcriptions.

3.4.1 Recognition using LM and 2-Gaussian Triphone HMM Model

The first recognition test will be performed using the initial LM, Count-based LM and Class-based LM. The 2-Gaussian triphone HMM acoustic models are used as according to the previous tests in 3.1.3.3 they are modeled better than higher mixtures due to insufficient training data from Albayzin corpus. Repeating the recognition with different scale factors and word insertion penalties, we have found that the scale factor of 13 and the insertion penalty of 0 yields highest performances. Table 3.13 presents the recognition results with these parameter values. Table 3.14 and 3.15 show the results for count-based models and class based models with the same scale factor value. These results, indicate that the recognition performance of initial LM and count-based models are the same and they are much better than class-based LM. We think this is because, the classification criteria does not cover all instances of triphone cases and it is more probable that a wrong combination obtains higher probability to be selected as the recognition result.

Table 3.13: Recognition results for initial LM and 2 Gaussian triphone model - scale factor=13

SENT				WORD						
%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
11.76	2	15	17	82.78	81.57	274	14	43	4	331

Table 3.14: Recognition results for count-based LM and 2-Gaussian triphone model - scale factor=13

SENT				WORD						
%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
11.76	2	15	17	82.78	81.57	274	14	43	4	331

Table 3.15: Recognition results for class-based LM and 2-Gaussian triphone model - scale factor=13

SENT				WORD						
%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
0.00	0	17	17	17.82	-87.92	59	4	268	350	331

3.4.2 Recognition using LM and discriminatively trained 2-Gussian Triphone HMM Model

According to the recognition results of the previous section, the initial LM has been chosen to be used together with discriminatively trained acoustic models in recognition task at this step. Both MMI and MPE discriminative training methods were applied on 2-Gaussian triphone HMM model. Table 3.16 and 3.17 shows the recognition results. For MMI test the scale factor of 13 got the best performance while for MPE it was 14.

Table 3.16: Recognition results for initial LM and MMI discriminatively trained 2-Gaussian triphone model - scale factor=13

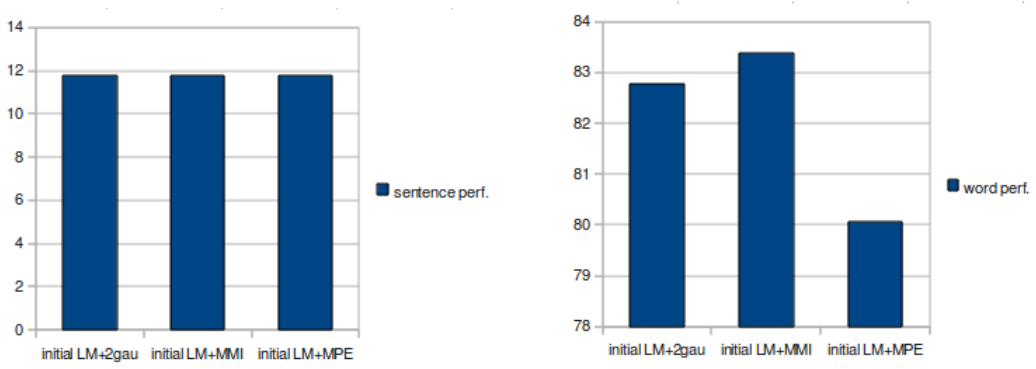
SENT				WORD						
%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
11.76	2	15	17	83.38	82.18	276	13	42	4	331

Table 3.17: Recognition results for initial LM and MPE discriminatively trained 2-Gaussian triphone model - scale factor=14

SENT				WORD						
%CORR	H	S	N	%CORR	ACC	H	D	S	I	N
11.76	2	15	17	80.06	77.04	265	14	52	10	331

3.4.3 Comparision of the Results

Regarding the obtained recognition results, we can now compare the methods. Figure 3.20 represent the comparision of the results for three different acoustic models. In cotntrast to our expectation, the MPE gets worse result than MMI. The processing time during the recognition depends on the pruning factor that we adjust it. With smaller values, it will be done faster but more less probable hypothesises are ignored which reduce the performance to some extent.



(a) Sentence Performance

(b) Word Performance

Figure 3.20: Comparision of the recognition results for three different modesls

Chapter Summary

In this chapter we have developed a Spanish language recognizer using HTK toolkit. First, we have used different feature extraction methods that HTK supports. Using these features, single Gaussian HMMs for Spanish phonemes have been developed and the models were tested on a domotica word grammar. Results, show that MFCC and PLP are the two efficient feature extraction methods, which gain higher recognition performance. Therefore, MFCC have been used for the rest of experiments. Cross-word triphone SGM and GMM HMMs were built using these features as well as adapted models. Our models suffer from the insufficient training data and therefore, during the splitting the Gaussian models there were warnings of un-updated models. As a result, there weren't prominent changes in the performances of the experimental results. Therefore, the 2-Gaussian HMM which has been modeled better and requires less computation has been used for the rest of experiments. Also, language modeling has been performed as well as discriminative training using Spanish wikipedia transcriptions. Comparing the results of perplexity tests of initial LM, count-based LM and class-based LM show that class-based LM offers better perplexity. Finally, the acoustic HMMs and LM have been used in decoding phase for recognition of wikipedia test recordings. MMI discriminatively trained acoustic models improved the performance, but MPE did not unexpectedly. Also, class-based model has worse performance than initial LM.

4 Future Research in ASR

In chapter 2 different components of HMM-based ASR systems and some challenges in this case were studied. Using this information, in this chapter, two potential fields in ASR will be discussed as suggestions for performing more researches in future. The first field is robust speech recognition which is an important and active challenging area in ASR and has been investigated in different aspects. Considering robust speech recognition approaches that have been described briefly in chapter 2, we are going to make some suggestions for future work. The second case is a novel approach in ASR which is based on completely different strategies in feature extraction and modeling. There has been just a few research about it in the past.

4.1 Optimized Robust Integrated Phase-Dependent ASR

In ASR systems, one of the key factors for a successful speech recognition is the good quality of input signal. In practice, speech signal may be affected by noise which include Gaussian noise, speech noise (unrelated conversation) and reverberation.

Multi-microphone based speech enhancement approaches have been used successfully in ASR. They are specially used in environments where a close talking microphone is not applicable and speech is captured by an array of microphone located some distance from the user. As the distance between the microphone and the user increases, the speech signal is increasingly distorted by the effect of additive noise, reverberation or even competitive speaker which in turn, degrades the speech signal. The key idea behind array processing is that, by using an array of microphones rather than a single microphone, it is possible to achieve spatial selectivity, reinforcing speech sources propagating from a particular direction, while attenuating sources propagating from other directions. Therefore, many microphone array-based processing algorithms are proposed to manipulate the received signal according to various-level criteria in order to generate an enhanced output signal.

4.1.1 Integrated Microphone Array-Based Techniques for ASR

One of the successful microphone array-based techniques for ASR systems is proposed in [4]. In this work, the main idea is that, microphone array-processing methods for signal enhancement, do not necessarily result in an improvement in the feature extraction step of ASR. Because, microphone array processing is a signal processing problem while the speech recognition is more a pattern classification problem. In ASR, speech waveform is converted into a sequence of feature vectors and the recognizer compares these vectors to a statistical class models of sound units (usually GMMs in HMM model states) and the output is a label corresponding to the sound class or a sequence of sound classes that has the maximum likelihood of generating the observed vectors. Thus, it is suggested that to improve the speech recognition performance or in other words, to maximize the likelihood of the correct class, the parameters of microphone arrays should be found optimally so that, features extracted from the input signal and the manner in which the features are processed be optimized.

This can be considered as an integrated system in which, according to the results of ASR system, the parameters of the microphone arrays are changed in order to reach to the optimum results. In [4], this idea is implemented for beam forming microphone array-based algorithm which is capable of resulting good performance in environments with significant level of additive noise and reverberation, but the competitive speaker scenario has not yet been addressed.

4.1.1.1 Likelihood Maximizing by Microphone Array Parameter Estimation

Speech recognition systems operate by finding the word string most likely to generate the observed sequence of feature vectors, as measured by the acoustic models and language models of the recognition system. This has has been formulated in Equation2.2 as :

$$\hat{W} = \arg\max_{w} P(X|w)P(w)$$

When speech is captured by a microphone array, the feature vectors X are a function of both the incoming speech and the array processing parameters ξ. In an integrated microphone array based ASR system, the goal is to tune array parameters ξ to optimize the recognition task. One logical approach is to choose the array parameters that maximize the likelihood of the correct transcription of the utterance that was spoken. Assume that the correct transcription of the utterance, which we notate as w_C, is known. Then the Equation 2.2 can be maximized for the array parameters , as:

$$\hat{\xi} = \arg\max_{w} P(X(\xi)|w_C)P(w_C) = \arg\max_{w} P(X(\xi)|w_C) \tag{4.1}$$

Which can be described as acoustic model maximization regarding to microphone array parameters. This is computed as the total likelihood of all possible paths, i.e. state sequences, through HMM of the word w_C HMM_{wC} . If S_C represents the set of all possible state sequences through HMM_{wC} , and s represents one such state sequence, then the maximum likelihood estimate of ξ can be written as:

$$\hat{\xi} \approx \arg\max_{\xi, s_i \epsilon S_C} \prod_i P(x_i(\xi)|s_i)P(s_i|s_{i-1}, w_C) \tag{4.2}$$

It will be more convenient and entirely equivalent to maximize the log likelihood rather than the likelihood itself. Thus, we have:

$$\hat{\xi} \approx \arg\max_{\xi, s \epsilon S_C} \left\{ \sum_i \log\left(P(x_i(\xi)|s_i)\right) + \sum_i \log\left(P(s_i|s_{i-1}, w_C)\right) \right\} \tag{4.3}$$

According to Equation 4.3, in order to find $\hat{\xi}$, the likelihood of the correct transcription must be jointly optimized with respect to both the array parameters and the state sequence. The second one is achieved by the Viterbi algorithm which is known as forced alignment or Viterbi alignment. The array parameter optimization which is the first component will be discussed later.

4.1.2 Phase-Dependent Time Frequency Masking

In [27], a phased-based dual microphone technique for robust speech enhancement was proposed. Using the same approach, in [19] a robust digit recognition algorithm has been built which was based on a short-time filtering strategy, alternatively known as time-frequency masking. It was shown that such a technique can be specifically useful when a mixed signal with one speaker of interest and

several noise speakers may be inseparable in either time or frequency domain, they are separable to some extent in time- frequency domain.

As the time domain representation of speech signal does not show useful information, usually short time Fourier transform (STFT) is applied to represent the harmonic an formant information of the signal in frequency domain. Considering signal S , taking the STFT, we have

$$\begin{aligned} |S| &= \begin{bmatrix} |F_0(0)| & \dots & |F_M(0)| \\ |F_0(1)| & & \\ \dots & & \\ |F_0(N)| & & |F_M(N)| \end{bmatrix} \\ \angle S &= \begin{bmatrix} \angle F_0(0) & \dots & \angle F_M(0) \\ \angle F_0(1) & & \\ \dots & & \\ \angle F_0(N) & & \angle F_M(N) \end{bmatrix} \end{aligned} \tag{4.4}$$

Spectograms show the information about the amplitude while the phase information $\angle S$ are ignored. The phase information have mostly been used in multi-microphone based approaches for time delay estimation.

Assuming two microphones and a sound source, the sound waves will arrive at two microphones at different times. Considering the sound speed, the time difference well known as the time difference of arrival (TDOA) is very small. Therefore, the spectograms of the two channels are very similar but the time-frequency phase images are very different and can be used to estimate the TDOA of the sound signal between the two microphones.

There are many different algorithms for estimating TDOA. The most frequent used is general cross correlation class which attempts to filter the cross correlation between two received signals in an optimal and suboptimal manner and then selects the time index of the peak of the result to be the TDOA estimate. Considering the two signal channels as:

$$x_1(t) = h_1.s(t) + n_1(t) \tag{4.5}$$

$$x_2(t) = h_2.s(t-\tau) + n_2(t) \tag{4.6}$$

where h_1 and h_2 are the impulse responses associated with the speech source for the first and second microphone, respectively, $x_1(t)$ and $x_2(t)$ are the signals obtained by the microphones, $n_1(t)$ and $n_2(t)$ are the noise signal associated with each microphone. The two microphones receive a time-delayed and scaled version of the source signal $s(t)$ without modeling reverberations. The goal of TDOA estimation is to estimate τ given the microphone signals. It is demonstrated that TDOA estimation algorithm results in:

$$\tilde{\tau} = \arg\max_{\beta} \sum_{K=1}^{N} \sum_{w=-w_s}^{w_s} \cos\left(\theta_{\beta,k}(w)\right) \tag{4.7}$$

Where N is the number of signal segments used to calculate the TDOA, w_s is the highest frequency of interest and $\theta_{\beta,k}(w)$ is the phase error of the k th signal segment at time index β and frequency of w and is defined as:

$$\theta_{\beta,k}(w) = \angle X_{1,k}(w) - \angle X_{2,k}(w) - w\beta \tag{4.8}$$

where $X_{1,k}(w)$ and $X_{2,k}(w)$ are the representation of signal segments of two channels in the frequency domain. Equation 4.7 involves a maximization that will be achieved when the appropriate choice of β equals the TDOA τ , resulting in a decreased phase error for most frequencies. As a result, the phase transform can be (approximately) represented as a phase error minimization technique:

$$\tilde{\tau} = \arg\min_{\beta} \psi_{\beta} \tag{4.9}$$

where ψ_{β} is the following phase variance corresponding to the TDOA β:

$$\psi_{\beta} = \sum_{K=1}^{N} \sum_{w=-w_s}^{w_s} \theta_{\beta,k}^2(w) \tag{4.10}$$

At the correct time delay τ, Equation 4.9 will have a minimized phase variance ψ_{τ} of

$$\psi_{\tau} = \min_{\beta} \sum_{K=1}^{N} \sum_{w=-w_s}^{w_s} \theta_{\beta,k}^2(w) \tag{4.11}$$

Ideally, ψ_{τ} should be equal to 0. However, due to the presence of noises, reverberations, and the effects of a finite-duration window, the minimum phase variance, or MPV, will be nonzero. As an example, consider the case when

$$x_1(t) = s(t) + n_1(t) \tag{4.12}$$

$$x_2(t) = s(t-\tau) + n_2(t) \tag{4.13}$$

where $n_1(t)$ and $n_2(t)$ are independent Gaussian signals. Results from experiments show that a decrease in the SNR or increase in reverberation will increase the minimum phase variance MPV. Therefore, MPV is a good indicator of the level of noise or reverberation. Because its the sum-square for all TF blocks. A TF block, corresponds to a specific frequency component in a specific time component. Science we may have several time segments indexed by k and several frequencies indexed by w , we call this component the kth time and wth frequency block. The block based phase error is obtained by Equation 4.8. This will be used as a metric for the amount of noise in an individual block. It has been proved that the phase error defines an upper bound for the signal to noise ratio of a given TF block. Therefore, the phase error is used as a reward punish criteria for noise removal. The TF blocks with large phase errors would be punished which means that their amplitude will be scaled down, while the low phase error blocks would not be changed.

An appropriate reward-punish technique should not damage the clear signal while having strong effect on on noisy signal. In other words its behavior should be a good function of signal to noise ratio.

For this, $\eta_k(w)$, is considered as a reward punish parameter between 0 to 1 for treating the TF block of the kth time and wth frequency. Then the overall SNR R, is defined as

$$R = \frac{\sum_w \sum_k |S_k(w)|^2}{\sum_w \sum_k |N_k(w)|^2 \eta_k(w)^2 + |S_k(w)|^2 (1-\eta_k(w))^2} \tag{4.14}$$

Where $S_k(w)$ is the coefficient of the kth time block of frequency w and $N_k(w)$ is the noise component of the same TF block.

Clearly, the goal is to maximize the SNR, therefore, differentiating with respect to $\eta_k(w)$ and equating to 0, the optimal scaling equation is obtained as follows:

$$\eta_k(w) = \frac{R_k(w)}{1 + R_k(w)} \tag{4.15}$$

where $R_k(w)$ is defined as the SNR of the $\{w,k\}$ TF block:

$$R_k(w) = \frac{\left|S_{j,k}(w)\right|^2}{\left|N_{j,k}(w)\right|^2} \tag{4.16}$$

This equation is also known as *Weiner Filter*. As the block scaling bound in Equation 4.15 is a function of a block -SNR, and also, we are interested to relate the phase error to the masking function, first the relationship of the block-SNR with the phase error is found. It is demonstrated that the block-SNR is bound so that:

$$R_k(w) \leq \frac{1}{\sin^2 \frac{\theta_k(w)}{2}} \tag{4.17}$$

Using this in Equation 4.15 we have:

$$\eta_k(w) \leq \frac{1}{\sin^2 \frac{\theta_{\tau,k}(w)}{2}} \tag{4.18}$$

Where $\eta_k(w)$ is the upper bound for the scaling of a given TF. One proposed equation as a parametrize scaling strategy for each TF block is:

$$\eta_k(w) \leq \frac{1}{1 + \gamma\theta^2_{\tau,k}(w)} \tag{4.19}$$

Where $\eta_k(w)$ is the TF block attenuation function and γ is a fixed constant. Figure 4.1 illustrates output SNRs obtained using the attenuation function of Equation 4.19 with different values of γ. In [27], finally $\gamma = 5$ was selected for boundary scale function, as a trade off for the treatment between punishment of low SNRs and rewards of high SNRs.

Obviously, a more reasonable technique for adjusting γ according to the variation of SNR would be more successful. For example, defining γ as a function of SNR, is a better treatment, but, as there is not direct information about SNR at the enhancement stage, a measure should be find to show the quality of the signal or its effect in recognition task according which the γ could be chosen.

4.1.3 Integrating Masking Function with ASR System

Regarding to the above discussion for defining $\eta_k(w)$ and the idea from [4] which was explained in 4.1.1, we suppose presumably, its possible to use phase information in an integrated system to address the noise speaker problem. Consequently, it is necessary:

- To investigate a better selection criteria for $\eta_k(w)$ due to its dependency to $\theta_{\tau,k}(w)$.
- To research for addressing the problem of noise competitive speakers by an integrated microphone array-based system using phase information.

To obtain more accurate information about these questions, we focus to the approach of integration which was proposed in [4] in more detail. The integration was started by defining the MFCC feature vectors X as a function of microphone array parameters ξ in 4.1. The problem of finding the optimized array parameters was lead to 4.3. Where ξ appears in its first component:

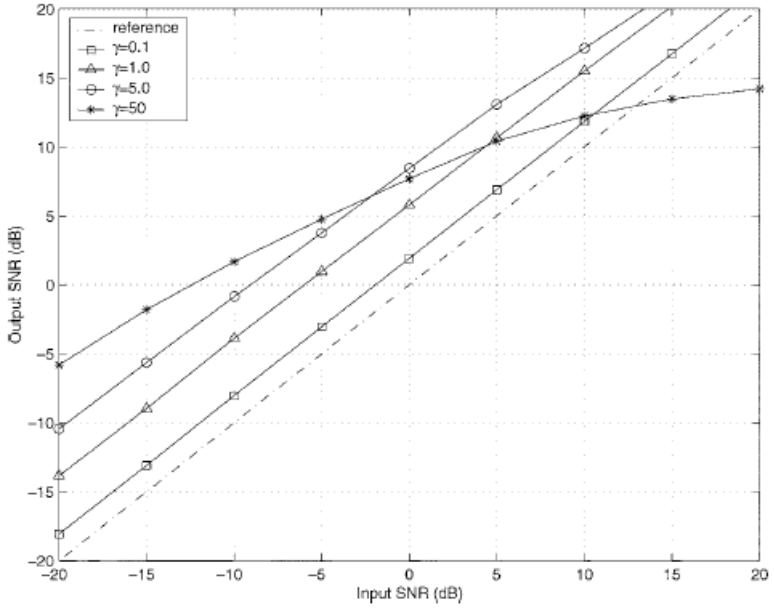


Figure 4.1: SNR ratio improvement using the attenuation function of Equation 4.19

$$\hat{\xi} \approx \arg\max_{\xi} \left\{ \sum_{i} \log\left(P(x_i(\xi)\,|s_i)\right) \right\}$$

Where $X = \{x_1, x_2, ..., x_T\}$ is the observation feature vectors as a function microphone array parameters ξ . Now we should see how MFCC features could be a function of array parameters. In [4], the microphone array processing approach of beamforming concept, is used as a generalized definition of *filter and- sum beamformer* where, each microphone signal has an associated filter and the captured signals are filtered before they are combined. If we consider $z_m[n]$ as the signal captured by the mth microphone in the array, then the output of a filter-and sum beamforming system $y[n]$ can be expressed as:

$$y[n] = \sum_{m=0}^{m-1} \sum_{p=0}^{p-1} h_m(p)\, z_m[n-p] \tag{4.20}$$

Where m is the number of microphones in the array , p is the length of the FIR filters used to process the array signals and $h_m(p)$ is the pth tap of the filter associated with microphone m. Not that, the output signal is then segmented into a series of overlapping frames from which the mel-frequency cepstral feature vectors $X(\xi)$ will be extracted. Therefore, ξ is the vector of length $m.p$ composed of all filter parameters for all microphones, expressed as :

$$\xi = [h_0[0], h_0[1], ..., h_{m-1}[p-2], h_m[p-1]]^T \tag{4.21}$$

This will help us to obtain closer understanding about the relationship of MFCC vectors to array parameters. A non-linear optimization methods has been employed to solve the maximization problem. For more information refer to [4]. A gradient-based approach has been used to find the optimal value of $\hat{\xi}$. For convenience, $L(\xi)$ is defined as the total log likelihood of the observation vectors given an HMM state sequence. Then the problem is defined as:

$$L(\xi) = \sum_{i} \log\left(P(x_i(\xi)\,|s_i)\right) \qquad ;x_i(\xi) = MFCC(y_i)$$

Thus, the $\nabla L(\xi)$ is computed with respect to the array parameters ξ :

$$\nabla_{\xi} L(\xi) = \left[\frac{\partial L(\xi)}{\partial h_0[0]}, \frac{\partial L(\xi)}{\partial h_0[1]}, \cdots \frac{\partial L(\xi)}{\partial h_{m-1}[p-1]} \right] \tag{4.23}$$

It is now clear that the relationship between MFCC feature vector and array parameters are due to is $h_m(p)$ coefficients, which lead to the successful optimization by gradient function.

4.1.4 Objectives of Research in Phase-Dependent ASR

In order to cancel the effect of noise speaker, our goal is to relate the phase information of the signal to the array coefficients. But, **There is not any direct information of phase in MFCC feature vectors**. This means that, although there are some useful information in phase, ASR systems do not utilize them. Therefore, some important points should be considered:

- In order to integrate noise canceling criteria or in other words the reward-punish function $\eta_k(w)$ to maximum likelihood criteria, we should extract a phase related variable in likelihood Equation 4.3.
- One alternative may be to focus on the ASR models to develop another feature extraction method which utilizes the phase information of the signal. This may affect the whole ASR architecture which may even cause a revolution in the whole model.

4.2 Part-Based Models and Local Features for ASR

In [2], a novel approach for feature extraction and modeling in ASR systems is proposed. However, this works in its very initial stage and there is a high potential for research to use its general ideas and expand it to be used for ASR systems.

The most important reason for proposing such a new approach is the limitation of current HMM-based ASR systems performance compared to human. In fact, despite of many researches for several decades, the progress in ASR has been little and no prominent change have been made in the underlying models of speech. The main argue is that, well known phonetic cues, crucial to human speech perception are not modeled effectively in standard ASR systems, and such cues should be modeled explicitly rather than implicitly.

The idea is that phonetic cues can be described as patterns in an appropriate time-frequency (T-F) representation. These patterns are often localized to small T-F regions instead of spanning the full frequency region. Such cues include phenomena such as formant transitions, bursts in particular frequency bands and voicing information. This idea is illustrated in Figure 4.2. Instead of the traditional model of speech, a sequence of spectral profiles (frame-based features and a discrete sequence of states), in this work, it is proposed to model phonetic cues as a collection of local acoustic phonetic cues arranged in a flexible configuration. It supports the idea to construct models able to capture this kind of information directly.

4.2.1 Privileges of Local Features

The current well-developed ASR models work on features extracted from the whole frequency bands while the proposed method just considers particular frequencies that could be used to distinguish each phoneme. The most advantages of these features are:

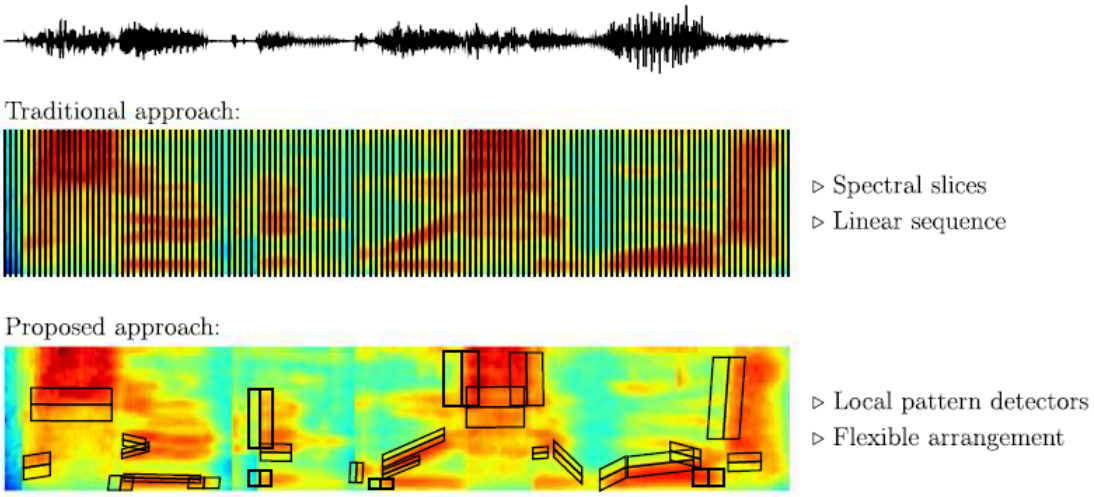


Figure 4.2: High-level illustration of the goals of this thesis. Typical ASR approaches model speech as a linear sequence of states represented by short-time spectral profiles. Our goal is to explore approaches based on collections of local time-frequency pattern detectors.

- **Noise Robustness-** If there is a noise source corrupting one frequency band, all MFCCs are affected. It has been shown that local features approach makes great improvements in noise robust ASR by ignoring (or rather marginalizing out) T-F regions which can be identified as corrupted. These methods will not work on non-localized features.
- **Speech Perception-** In [2], it is mentioned that according to many reasons, human speech perception relies heavily on processing information in limited frequency bands. The "glimpsing" model of speech perception [32] suggests that humans can robustly decode noise-corrupted speech by taking advantage of local T-F regions having high SNR, and is supported by empirical evidence and computational models.
- **Auditory Neuroscience-** Recent work in auditory neuroscience also provides evidence indicating that most neurons have a strongest response at a particular frequency. (Also, recent research seeking to characterize the behavior of individual neurons in the mammalian auditory cortex has resulted in models in which cortical neurons act as localized spectro-temporal pattern detectors.

According to these facts, in [2], this feature extraction approach is used with part-based modeling framework to model different letters.

4.2.2 Part-Based Models

Part-based models have been widely used in machine vision algorithms and utilizes graphical modeling. A prototypical vision application of parts-based models is that of face detection. Instead of trying to build a single pattern detector which locates a face as a whole, faces are modeled as a collection of individual parts. Local detectors can independently look for the parts (eyes, nose, mouth, etc.), while the overall system attempts to find the set of parts in roughly the correct configuration (e.g., a nose should be in-between and below two eyes, a mouth should be below a nose, etc.). The high-level idea for creating a parts-based model (PBM) of speech is illustrated in Figure .The "parts" will be localized detectors, each trying to detect a specific phonetic cue. Figure 4.3-a highlights several cues

for the letter B (the diphone /b iy/). Figure 4.3-b shows how these cues might be encoded with a collection of local patch-based part detectors. The patches detect individual cues, while the spring connections between patches indicate that their relative placement has some flexibility. Therefore, it represents a deformable template.

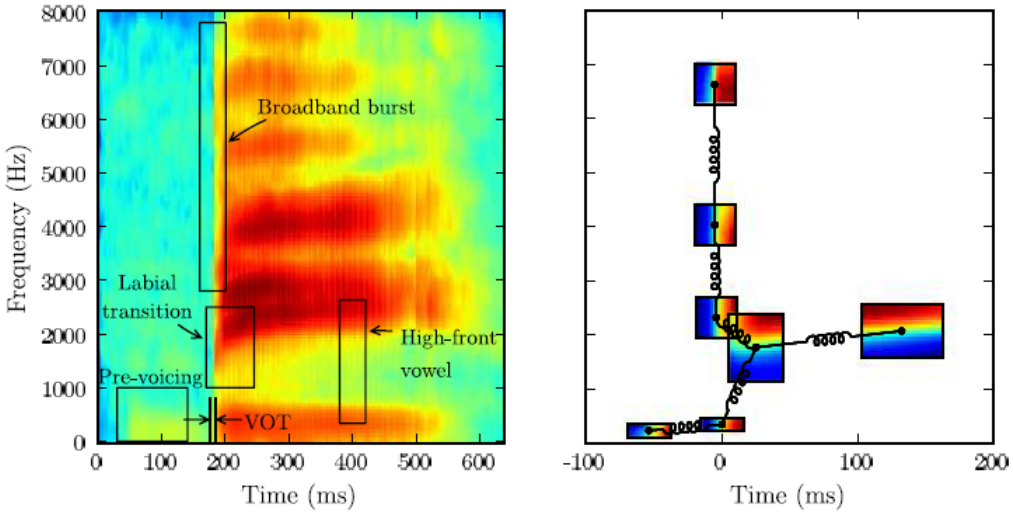


Figure 4.3: High-level description of parts-based modeling for speech. (a) Labeled phonetic cues for the diphone (letter) B. (b) The visualization of a parts-based model to capture these cues with localized patch detectors.

4.2.3 Objectives of Research in Pattern-Based Feature Extraction and Part-Based Models

The work described in previous section, was just applied on ISOLET corpus of isolated spoken letters. Therefore, there is a lot of room to research in this case. Some important lines for research are:

- To study the structure and characteristics of pattern_based features known as local features
- To research about part-based models and their use in pattern-based features modeling
- To study about the application of classification techniques, specially vector machine classification in ASR and investigating their use in pattern_based feature classification
- To apply the part-based model ASR systems on LVCR problem

Chapter Summary

In this chapter, two areas for research in the field of ASR were introduced. The first suggested subject was in the case of robust ASR where noise speaker exists. Our studies show that, no integrated ASR system was modeled for this problem. As the HMM-based ASR does not use the phase information, there is not a direct way to integrate a phase-based noise canceling algorithm to the ASR system. More studies should be performed to formulize the relationship between phase information and ASR indirectly or to propose a phase dependent ASR system to address this problem. The second suggested topic to investigate is the pattern-based feature extraction approach which was recently proposed but not yet used in large vocabulary continuous speech recognition (LVCR). Parts-based models were

applied on these features in early work but stronger classifiers should be used to deal with LVCR problem.

5 Conclusion

In this thesis, the basic principal components of HMM-based ASR systems has been studied and the most significant methods to improve the system performance for LVCR has been explained. The most challenges of these systems have been also described.

In order to have deeper understanding of ASR problem, a Spanish language large vocabulary system has been developed. During the implementation, different feature extraction methods were compared as well as different HMM models. Results show higher performance with MFCC and PLP feature extraction methods. According to this, MFCC feature extraction method has been used in all experiments. Cross-word triphone SGM and GMM HMMs were built using these features as well as adapted models. Our models suffer from the insufficient training data and therefore, during the splitting the Gaussian models there were warnings that indicated the models weren't updated. As a result, the performances of the experimental results did not changed sharply using different mixture models. Therefore, the 2-Gaussian HMM which has been modeled better and requires less computation has been used for the rest of experiments. But we believe that this is because of our limited corpus used in acoustic modeling and with a larger corpus, higher mixture models should be more efficient. In addition, language modeling was performed as well as discriminative training. Results show that, general language models outperform the class-based language models in recognition task. This is because of general nature of these models that make it high probable that wrong hypothesised words obtain higher performance among the other words of that class to be selected. Comparing the results also show that, discriminative training improves the recognition performance.

Finally, we have suggested two open research lines for future. According to our studies, no integrated ASR system for the problem of the noise speaker has been suggested. The current solutions are using phase-based microphone arrays methods. But, no integrated ASR system was modeled for this problem. As the HMM-based ASR does not use the phase information, there is not a direct way to integrate a phase-based noise canceling algorithm to the ASR system. More studies should be performed to formulize the relationship between phase information and ASR indirectly or to propose a phase dependent ASR system to address this problem. Another rich area for research is the pattern-based feature extraction approach, which was recently proposed but not yet used in large vocabulary continuous speech recognition (LVCR). There is a lot of room to work in this case to find a strong classification method and prepare the models for applying on LVCR.

Bibliography

[1] L. Rabiner and B.-H. Juang. Fundamentals of speech recognition. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] K. T. Schutte. Phd Thesis: Parts-based Models and Local Features for Automatic Speech Recognition. Massachusetts Institute of Technology(MIT), USA, 2009.

[3] S. Young, G. Evermann, M. Gales, T. Hain, D. Kerhsaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. The HTK Book. Cambridge University Engineering Department, 2002.

[4] M. L. Seltzer. Phd Thesis: Microphone Array Processing for Robust Speech Recognition. Carnegie Mellon University, USA, 2003.

[5] M. Gales, S. Young. The Application of Hidden Markov Models in Speech Recognition. Now Publishers Inc. Hanover, MA, USA ,2008.

[6] N. Kumar, A. Andreou. Heteroscedastic analysis and reduced rank HMMs for improved speech recognition, Speech Commun. vol. 26, no. 4, pp. 283–297, 1998.

[7] P. C. Woodland, D. Povey. Large scale discriminative training of hidden Markov models for speech recognition. Computer Speech and Language , vol. 16, pp. 25–47, 2002.

[8] P. F. Brown. PhD thesis: The Acoustic-Modelling Problem in Automatic Speech Recognition. Carnegie Mellon University, USA, 1987.

[9] D. Povey, P.C. Woodland. Minimum Phone error and I-smoothing for Improved Discriminative Training. In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing. p I-105-8 vol.1, 2002.

[10] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, G. Zweig. FMPE: Discriminatively Trained Features for Speech Recognition. In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 1, pp. I/961-4 , 2005.

[11] B. Tasker. Phd Thesis: Learning Structured Prediction Models: A Large Margin Approach. Stanford University, USA, 2004.

[12] M. L. Seltzer. Phd Thesis: Microphone Array Processing for Robust Speech Recognition. Carnegie Mellon University, USA, 2003.

[13] S. Neely, J. Allen. Invertibility of a room impulse response. Journal of the Acoustical Society of America, vol. 66, no. 1, pp. 165-169, 1979.

[14] [6] D. H. Johnson, D. E. Dudgeon, Array Signal Processing: Concepts and Techniques. New Jersey: Prentice Hall, 1993.

[15] L. Gri_ths and C. Jim. An alternative approach to linearly constrained adaptive beamforming. IEEE Trans. on Antennas and Propagation, vol. 30, no. 1, pp. 27-34, 1982.

[16] P. Raghavan, R. J.Renomeron, C.Che, D. S.Yuk, J. L.Flanagan. Speech recognition in a reverberant environment using matched filter array (MFA) processing and linguistic tree maximum

likelihood linear regression (LT-MLLR) adaptation. in Proc. ICASSP '99, Phoenix, pp.777-780,Arizona, 1999.

[17] H. Buchner, R. Aichner, W. Kellermann. Trinicon: A versatile framework for multichannel blind signal processing. In Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Canada, 2004.

[18] T.M.Sullivan. Phd Thesis: Multi-microphone correlation-based processing for robust automatic speech recognition. Carnegie Mellon University, USA, 1996.

[19] G. Shi, P. Aarabi. Robust Digit Recognition Using Phase-Dependent Timefrequency Masking. In Proc. of ICASSP, Hong Kong,2003.

[20] M.C. Amara Korba, D. Messadeg, R. Djemili, H. Bourouba. Robust Speech Recognition Using Perceptual Wavelet Denoising and Mel-frequency Product Spectrum Cepstral Coefficient Features, Informatica, vol 32 pp. 283-288, 2008.

[21] H. K. Kim, R. C. Rose. Cepstrum-Domain Acoustic Feature Compensation Based on Decomposition of Speech and Noise for ASR in Noisy Environments. IEEE Transactions on Speech and Audio Processing, Vol. 11, No. 5, 2003.

[22] M. Myllymaki and T. Virtanen. Non-Stationary Noise Model Compensation in Voice Activity Detection. , EUSIPCO, Scotland, 2009.

[23] S. V. Vaseghi, B. P. Milner. Noise Compensation Methods for Hidden Markov Model Speech Recognition in Adverse Environments. IEEE Transactions on Speech and Audio Processing, Vol. 5, No. 1, 1997.

[24] L. Deng, J. Droppo, A. Acero. Dynamic Compensation of HMM Variances Using the Feature Enhancement Uncertainty Computed From a Parametric Model of Speech Distortion. IEEE Transactions on Speech and Audio Processing, Vol. 13, No. 3, 2005.

[25] I. Potamitis, N. Fakotakis, G. Kokkinakis. Bayesian Noise Compensation of Time Trajectories of Spectral Coefficients for Robust Speech Recognition. V. Matousek et al. (Eds.), pp. 214–221, Springer-Verlag Berlin Heidelberg, 2001.

[26] G. Garau, S. Renals. Applying Vocal Tract Length Normalization to Meeting Recordings. In Proc. Interspeech 2005, 265-268, Portugal, 2005.

[27] P. Aarabi, G. Shi. Phase-Based Dual-Microphone Robust Speech Enhancement. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, Vol. 34, No. 4, 2004.

[28] X. Huang, A. Acreo, H. W. Hon. Spocken Language Processing. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2001.

[29] M. Ch. Su, Ch. T. Hsieh, Ch. Ch. Chin. A neuro-fuzzy approach to speech recognition without time alignmeent. Elsvier, Fuzzy Sets and Systems98. pp. 33-41-1998.

[30] J. M. Baker, L. Deng, J. Glass, S. Khudanpur, Ch.H. Lee, N. Morgan, and D. O'Shaughnessy. Research Developments and Directions in Speech Recognition and Understanding, Part 1. IEEE Signal Processing Magazin, pp 75-80, May, 2009.

[31] J. M. Baker, L. Deng, S. Khudanpur, Ch.H. Lee, J. R. Glass, N. Morgan, and D. O'Shaughnessy. Updated MINDS Report on Speech Recognition and Understanding, Part 2. IEEE Signal Processing Magazin, pp 78-84, July, 2009.

[32] M. Cooke. A glimpsing model of speech perception in noise, Jurnal of Acoustical Society of America, Vol. 119, No. 3,1562–1573, March, 2006