CHAPTER 9

Acoustic Modeling

After years of research and development, accuracy of automatic speech recognition remains one of the most important research challenges. A number of well-known factors determine accuracy; those most noticeable are variations in context, in speaker, and in environment. Acoustic modeling plays a critical role in improving accuracy and is arguably the central part of any speech recognition system.

For the given acoustic observation $\mathbf{X} = X_1 X_2 ... X_n$, the goal of speech recognition is to find out the corresponding word sequence $\hat{\mathbf{W}} = w_1 w_2 ... w_m$ that has the maximum posterior probability $P(\mathbf{W} \mid \mathbf{X})$ as expressed by Eq. (9.1).

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{w}} P(\mathbf{W} \mid \mathbf{X}) = \arg\max_{\mathbf{w}} \frac{P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})}{P(\mathbf{X})} \tag{9.1}$$

Since the maximization of Eq. (9.1) is carried out with the observation **X** fixed, the above maximization is equivalent to maximization of the following equation:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{w}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W}) \tag{9.2}$$

The practical challenge is how to build accurate acoustic models, $P(\mathbf{X} \mid \mathbf{W})$, and language models, $P(\mathbf{W})$, that can truly reflect the spoken language to be recognized. For large-vocabulary speech recognition, since there are a large number of words, we need to decompose a word into a subword sequence. Thus $P(\mathbf{X} \mid \mathbf{W})$ is closely related to phonetic modeling. $P(\mathbf{X} \mid \mathbf{W})$ should take into account speaker variations, pronunciation variations, environmental variations, and context-dependent phonetic coarticulation variations. Last, but not least, any static acoustic or language model will not meet the needs of real applications. So it is vital to dynamically adapt both $P(\mathbf{W})$ and $P(\mathbf{X} \mid \mathbf{W})$ to maximize $P(\mathbf{W} \mid \mathbf{X})$ while using the spoken language systems. The decoding process of finding the best matched word sequence $\mathbf{W}$ to match the input speech signal $\mathbf{X}$ in speech recognition systems is more than a simple pattern recognition problem, since in continuous speech recognition you have an infinite number of word patterns to search, as discussed in detail in Chapters 12 and 13.

In this chapter we focus on discussing solutions that work well in practice. To highlight solutions that are effective, we use the Whisper speech recognition system [49] developed at Microsoft Research as a concrete example to illustrate how to build a working system and how various techniques can help to reduce speech recognition errors.[1] We hope that by studying what worked well in the past we can illuminate the possibilities for further improvement of the state of the art.

The hidden Markov model we discussed in Chapter 8 is the underpinning for acoustic phonetic modeling. It provides a powerful way to integrate segmentation, time warping, pattern matching, and context knowledge in a unified manner. The underlying technologies are undoubtedly evolving, and the research community is aggressively searching for more powerful solutions. Most of the techniques discussed in this chapter can be readily derived from the fundamentals discussed in earlier chapters.

9.1. VARIABILITY IN THE SPEECH SIGNAL

The research community has produced technologies that, with some constraints, can accurately recognize spoken input. Admittedly, today's state-of-the-art systems still cannot match humans' performance. Although we can build a very accurate speech recognizer for a particular speaker, in a particular language and speaking style, in a particular environment, and limited to a particular task, it remains a research challenge to build a recognizer that can essentially understand anyone's speech, in any language, on any topic, in any free-flowing style, and in almost any speaking environment.

[1] Most of the experimental results used here are based on a development test set for the 60,000-word speaker-independent continuous dictation task. The training set consists of 35,000 utterances from about 300 speakers. The test set consists of 410 utterances from 10 speakers that were not used in the training data. The language model is derived from 2 billion words of English text corpora.

Accuracy and robustness are the ultimate measures for the success of speech recognition algorithms. There are many reasons why existing algorithms or systems did not deliver what people want. In the sections that follow we summarize the major factors involved.

9.1.1. Context Variability

Spoken language interaction between people requires knowledge of word meanings, communication context, and common sense. Words with widely different meanings and usage patterns may have the same phonetic realization. Consider the challenge represented by the following utterance:

> *Mr. Wright should write to Ms. Wright right away about his Ford or four door Honda.*

For a given word with the same pronunciation, the meaning could be dramatically different, as indicated by *Wright*, *write*, and *right*. What makes it even more difficult is that *Ford or* and *Four Door* are not only phonetically identical, but also semantically relevant. The interpretation is made within a given word boundary. Even with smart linguistic and semantic information, it is still impossible to decipher the correct word sequence, unless the speaker pauses between words or uses intonation to set apart these semantically confusable phrases.

In addition to the context variability at word and sentence level, you can find dramatic context variability at phonetic level. As illustrated in Figure 9.1, the acoustic realization of phoneme /*ee*/ for word *peat* and *wheel* depends on its left and right context. The dependency becomes more important in fast speech or spontaneous speech conversation, since many phonemes are not fully realized.

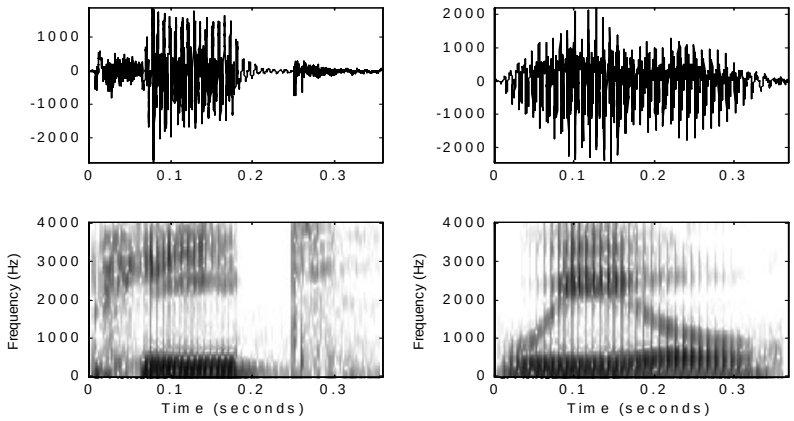


Figure 9.1 Waveforms and spectrograms for words *peat* (left) and *wheel* (right). The phoneme `/ee/` is illustrated with two different left and right contexts. This illustrates that different contexts may have different effects on a phone.

9.1.2. Style Variability

To deal with acoustic realization variability, a number of constraints can be imposed on the use of the speech recognizer. For example, we can have an *isolated* speech recognition system, in which users have to pause between each word. Because the pause provides a clear boundary for the word, we can easily eliminate errors such as *Ford or* and *Four Door*. In addition, isolated speech provides a correct silence context to each word so that it is easier to model and decode the speech, leading to a significant reduction in computational complexity and error rate. In practice, the word-recognition error rate of an isolated speech recognizer can typically be reduced by more than a factor of three (from 7% to 2%) as compared with to a comparable continuous speech recognition system [5]. The disadvantage is that such an isolated speech recognizer is unnatural to most people. The throughput is also significantly lower than that for continuous speech.

In continuous speech recognition, the error rate for casual, spontaneous speech, as occurs in our daily conversation, is much higher than for carefully articulated read-aloud speech. The rate of speech also affects the word recognition rate. It is typical that the higher the *speaking rate* (words/minute), the higher the error rate. If a person whispers, or shouts, to reflect his or her emotional changes, the variation increases more significantly.

9.1.3. Speaker Variability

Every individual speaker is different. The speech he or she produces reflects the physical vocal tract size, length and width of the neck, a range of physical characteristics, age, sex, dialect, health, education, and personal style. As such, one person's speech patterns can be entirely different from those of another person. Even if we exclude these interspeaker differences, the same speaker is often unable to precisely produce the same utterance. Thus, the shape of the vocal tract movement and rate of delivery may vary from utterance to utterance, even with dedicated effort to minimize the variability.

For *speaker-independent* speech recognition, we typically use more than 500 speakers to build a combined model. Such an approach exhibits large performance fluctuations among new speakers because of possible mismatches in the training data between exiting speakers and new ones [50]. In particular, speakers with accents have a tangible error-rate increase of 2 to 3 times.

To improve the performance of a speaker-independent speech recognizer, a number of constraints can be imposed on its use. For example, we can have a user enrollment that requires the user to speak for about 30 minutes. With the *speaker-dependent* data and training, we may be able to capture various speaker-dependent acoustic characteristics that can significantly improve the speech recognizer's performance. In practice, speaker-dependent speech recognition offers not only improved accuracy but also improved speed, since decoding can be more efficient with an accurate acoustic and phonetic model. A typical speaker-dependent speech recognition system can reduce the word recognition error by more than 30% as compared with a comparable speaker-independent speech recognition system.

The disadvantage of speaker-dependent speech recognition is that it takes time to collect speaker-dependent data, which may be impractical for some applications such as an automatic telephone operator. Many applications have to support walk-in speakers, so speaker-independent speech recognition remains an important feature. When the amount of speaker-dependent data is limited, it is important to make use of both speaker-dependent and speaker-independent data using *speaker-adaptive* training techniques, as discussed in Section 9.6. Even for speaker-independent speech recognition, you can still use speaker-adaptive training based on recognition results to quickly adapt to each individual speaker during the usage.

9.1.4. Environment Variability

The world we live in is full of sounds of varying loudness from different sources. When we interact with computers, we may have people speaking in the background. Someone may slam the door, or the air conditioning may start humming without notice. If speech recognition is embedded in mobile devices, such as PDAs (personal digital assistants) or cellular phones, the spectrum of noises varies significantly because the owner moves around. These external parameters, such as the characteristics of the environmental noise and the type and placement of the microphone, can greatly affect speech recognition system performance. In addition to the background noises, we have to deal with noises made by speakers, such as lip smacks and noncommunication words. Noise may also be present from the input device itself, such as the microphone and A/D interference noises.

In a similar manner to speaker-independent training, we can build a system by using a large amount of data collected from a number of environments; this is referred to as *multistyle training* [70]. We can use adaptive techniques to normalize the mismatch across different environment conditions in a manner similar to speaker-adaptive training, as discussed in Chapter 10. Despite the progress being made in the field, environment variability remains as one of the most severe challenges facing today's state-of-the-art speech systems

9.2. HOW TO MEASURE SPEECH RECOGNITION ERRORS

It is critical to evaluate the performance of speech recognition systems. The word recognition error rate is widely used as one of the most important measures. When you compare different acoustic modeling algorithms, it is important to compare their relative error reduction. Empirically, you need to have a test data set that contains more than 500 sentences (with 6 to 10 words for each sentence) from 5 to 10 different speakers to reliably estimate the recognition error rate. Typically, you need to have more than 10% relative error reduction to consider adopting a new algorithm.

As a sanity check, you may want to use a small sample from the training data to measure the performance of the *training set*, which is often much better than what you can get from testing new data. Training-set performance is useful in the development stage to identify potential implementation bugs. Eventually, you need to use a *development set* that typi-

cally consists of data never used in training. Since you may tune a number of parameters with your development set, it is important to evaluate performance of a *test set* after you decide the optimal parameter setting. The test set should be completely new with respect to both training and parameter tuning.

There are typically three types of word recognition errors in speech recognition:

- *Substitution*: an incorrect word was substituted for the correct word
- *Deletion*: a correct word was omitted in the recognized sentence
- *Insertion*: an extra word was added in the recognized sentence[2]

For instance, a speech recognition system may produce an incorrect result as follows, where substitutions are ***bold***, insertions are underlined, and deletions are denoted as **. There are four errors in this example.

> **Correct:** *Did mob mission area of the Copeland ever go to m4 in nineteen eighty one*
> **Recognized:** *Did mob mission area ** the **copy** land ever go to m4 in nineteen **east** one*

To determine the minimum error rate, you can't simply compare two word sequences one by one. For example, suppose you have utterance *The effect is clear* recognized as *Effect is not clear*. If you compare word to word, the error rate is 75% (*The* vs. *Effect*, *effect* vs. *is*, *is* vs. *not*). In fact, the error rate is only 50% with one deletion (*The*) and one insertion (*not*). In general, you need to align a recognized word string against the correct word string and compute the number of substitutions (*Subs*), deletions (*Dels*), and insertions (*Ins*). The *Word Error Rate* is defined as:

$$\text{Word Error Rate} = 100\% \times \frac{Subs + Dels + Ins}{\text{No. of word in the correct sentence}} \tag{9.3}$$

This alignment is also known as the *maximum substring matching* problem, which can be easily handled by the dynamic programming algorithm discussed in Chapter 8.

Let the correct word string be $w_1 w_2 \cdots w_n$, where w_i denotes the ith word in the correct word string, and the recognized word string be $\hat{w}_1 \hat{w}_2 \cdots \hat{w}_m$, where $\hat{w}_i$ denotes the ith word in the recognized word string. We denote $R[i, j]$ as the minimum error of aligning substring $w_1 w_2 \cdots w_n$ against substring $\hat{w}_1 \hat{w}_2 \cdots \hat{w}_m$. The optimal alignment and the associated word error rate $R[n, m]$ for correct word string $w_1 w_2 \cdots w_n$ and the recognized word string $\hat{w}_1 \hat{w}_2 \cdots \hat{w}_m$ are obtained via the dynamic programming algorithm illustrated in ALGORITHM 9.1. The accumulated cost function $R[i, j]$ progresses from $R[1, 1]$ to $R[n, m]$ corresponding to the minimum distance from (1, 1) to (n, m). We store the back pointer information $B[i,$

[2] Even for isolated speech recognition, you may still have the insertion error, since the word boundary needs to be detected in most applications. It is possible that one isolated utterance is recognized into two words.

j] as we move along. When we reach the final grid (n, m), we back trace along the optimal path to find out if there are substitutions, deletions, or insertions on the matched path, as stored in $B[i, j]$.

ALGORITHM 9.1: THE ALGORITHM TO MEASURE THE WORD ERROR RATE

Step 1: *Initialization* $R[0,0] = 0$ $R[i, j] = \infty$ if $(i < 0)$ or $(j < 0)$ $B[0,0] = 0$

Step 2: *Iteration*

for $i = 1, \ldots, n$ {

for $j = 1, \ldots, m$ {

$$R[i, j] = \min \begin{bmatrix} R[i-1, j]+1 \text{ (deletion)} \\ R[i-1, j-1] \text{ (match)} \\ R[i-1, j-1]+1 \text{ (substitution)} \\ R[i, j-1]+1 \text{ (insertion)} \end{bmatrix}$$

$$B[i, j] = \begin{cases} 1 & \text{if deletion} \\ 2 & \text{if insertion} \\ 3 & \text{if match} \\ 4 & \text{if substitution} \end{cases} \quad \} \}$$

Step 3: *Backtracking and termination*

$$\text{word error rate} = 100\% \times \frac{R(n,m)}{n}$$

$$\text{optimal backward path} = (s_1, s_2, \ldots, 0)$$

$$\text{where } s_1 = B[n,m],\ s_t = \begin{bmatrix} B[i-1, j] \text{ if } s_{t-1} = 1 \\ B[i, j-1] \text{ if } s_{t-1} = 2 \\ B[i-1, j-1] \text{ if } s_{t-1} = 3 \text{ or } 4 \end{bmatrix} \text{ for } t = 2, \ldots \text{ until } s_t = 0$$

For applications involved with rejection, such as word confidence measures as discussed in Section 9.7, you need to measure both false rejection rate and false acceptance rate. In speaker or command verification, the false acceptance of a valid user/command is also referred to as Type I error, as opposed to the false rejection of a valid user/command (Type II) [17]. A higher false rejection rate generally leads to a lower false acceptance rate. A plot of the false rejection rate versus the false acceptance rate, widely used in communication theory, is called the *receiver operating characteristic* (ROC) curve.

9.3. SIGNAL PROCESSING—EXTRACTING FEATURES

The role of a signal processing module, as illustrated in Figure 1.2, is to reduce the data rate, to remove noises, and to extract salient features that are useful for subsequent acoustic

matching. Using as building blocks the topics we discussed in earlier chapters, we briefly illustrate here what is important in modeling speech to deal with variations we must address. More advanced environment normalization techniques are discussed in Chapter 10.

9.3.1. Signal Acquisition

Today's computers can handle most of the necessary speech signal acquisition tasks in software. For example, most PC sound cards have direct memory access, and the speech can be digitized to the memory without burdening the CPU with input/output interruptions. The operating system can correctly handle most of the necessary AD/DA functions in real time.

To perform speech recognition, a number of components—such as digitizing speech, feature extraction and transformation, acoustic matching, and language model-based search—can be pipelined time-synchronously from left to right. Most operating systems can supply mechanisms for organizing pipelined programs in a multitasking environment. Buffers must be appropriately allocated so that you can ensure time-synchronous processing of each component. Large buffers are generally required on slow machines because of potential delays in processing an individual component. The right buffer size can be easily determined by experimentally tuning the system with different machine load situations to find a balance between resource use and relative delay.

For speech signal acquisition, the needed buffer typically ranges from 4 to 64 kB with 16-kHz speech sampling rate and 16-bit A/D precision. In practice, 16-kHz sampling rate is sufficient for the speech bandwidth (8 kHz). Reduced bandwidth, such as telephone channel, generally increases speech recognition error rate. Table 9.1 shows some empirical relative word recognition error increase using a number of different sampling rates. If we take the 8-kHz sampling as our baseline, we can reduce the word recognition error with a comparable recognizer by about 10% if we increase the sampling rate to 11 kHz. If we further increase the sampling rate to 16 kHz, the word recognition error rate can be further reduced by additional 10%. Further increasing the sampling rate to 22 kHz does not have any additional impact on the word recognition errors, because most of the salient speech features are within 8-kHz bandwidth.

Table 9.1 Relative error rate reduction with different sampling rates.

Sampling Rate	Relative Error-Rate Reduction
8 kHz	Baseline
11 kHz	+10%
16 kHz	+10%
22 kHz	+0%

9.3.2. End-Point Detection

To activate speech signal capture, you can use a number of modes including either *push to talk* or *continuously listening.* The push-to-talk mode uses a special push event to activate or deactivate speech capturing, which is immune to the potential background noise and can eliminate unnecessary use of processing resources to detect speech events. This mode sometimes also requires you to *push and hold while talking*. You push to indicate speech's beginning and then release to indicate the end of speech capture. The disadvantage is the necessity to activate the application each time the person speaks.

The continuously listening model listens all the time and automatically detects whether there is a speech signal or not. It needs a so-called *speech end-point detector*, which is typically based on an extremely efficient two-class pattern classifier. Such a classifier is used to filter out obvious silence, but the ultimate decision on the utterance boundary is left to the speech recognizer. In comparison to the push-to-talk mode, the continuously listening mode requires more processing resources, also with potential classification errors.

The endpoint detector is often based on the energy threshold that is a function of time. The logarithm of the energy threshold can be dynamically generated based on the energy profiles across a certain period of time. Constraints on word duration can also be imposed to better classify a sequence of frames so that extremely short spikes can be eliminated.

It is not critical for the automatic end-point detector to offer exact end-point accuracy. The key feature required of it is a low rejection rate (i.e., the automatic end-point detector should not interpret speech segments as silence/noise segments). Any false rejection leads to an error in the speech recognizer. On the other hand, a possible false acceptance (i.e., the automatic end-point detector interprets noise segments as speech segments) may be rescued by the speech recognizer later if the recognizer has appropriate noise models, such as specific models for clicks, lip smacks, and background noise.

Explicit end-point detectors work reasonably well with recordings exhibiting a signal-to-noise ratio of 30 dB or greater, but they fail considerably on noisier speech. As discussed, speech recognizers can be used to determine the end points by aligning the vocabulary words preceded and followed by a silence/noise model. This scheme is generally much more reliable than any threshold-based explicit end-point detection, because recognition can jointly detect both the end points and words or other explicit noise classes, but requires more computational resources. A compromise is to use a simple adaptive two-class (speech vs. silence/noise) classifier to locate speech activities (with enough buffers at both ends) and notify the speech recognizer for subsequent processing. For the two-class classifier, we can use both the log-energy and delta log-energy as the feature. Two Gaussian density functions, $\{\Phi_1, \Phi_2\} = \Phi$, can be used to model the background stationary noise and speech, respectively. The parameters of the Gaussian density can be estimated using the labeled speech and noise data or estimated in an unsupervised manner.

When enough frames are classified as speech segments by the efficient two-class classifier, the speech recognizer is notified to start recognizing the signal. As shown in Figure 9.2, we should include enough frames before the beginning frame, t_b, for the speech recognizer to minimize the possible detection error. In the same manner, when enough

noise/silence frames are detected at t_e, we should keep providing the speech recognizer with enough frames for processing before declaring that the end of the utterance has been reached.

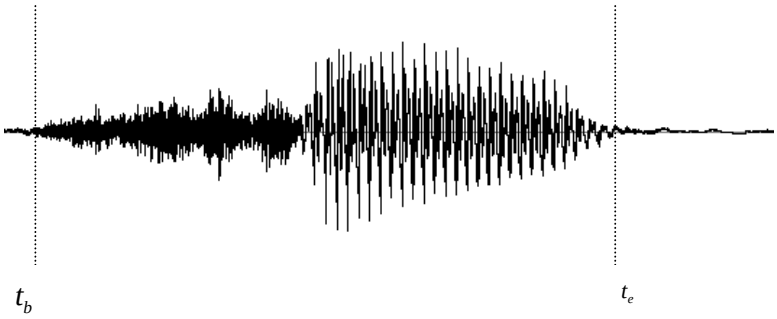


Figure 9.2 End-point detection boundary t_b and t_e may need extra buffering for subsequent speech recognition.

Since there are only two classes, these parameters can be dynamically adapted using the EM algorithm during runtime. As discussed in Chapter 4, the EM algorithm can iteratively estimate the Gaussian parameters without having a precise segmentation between speech and noise segments. This is very important, because we need to keep the parameters dynamic for robust end-point detection in constantly changing environments.

To track the varying background noises, we use an exponential window to give weight to the most recent signal:

$$w_k = \exp(-\alpha k) \tag{9.4}$$

where α is a constant that controls the adaptation rate, and k is the index of the time. In fact, you could use different rates for noise and speech when you use the EM algorithm to estimate the two-class Gaussian parameters. It is advantageous to use a smaller time constant for noise than for speech. With such a weighting window, the means of the Gaussian density, as discussed in Chapter 4, can be rewritten as:

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=-\infty}^{t} w_i \frac{c_k P(\mathbf{x}_i \mid \Phi_k)\mathbf{x}_i}{\sum_{k=1}^{2} P(\mathbf{x}_i \mid \Phi_k)}}{\sum_{i=-\infty}^{t} w_i \frac{c_k P(\mathbf{x}_i \mid \Phi_k)}{\sum_{k=1}^{2} P(\mathbf{x}_i \mid \Phi_k)}}, k \in \{0,1\} \tag{9.5}$$

9.3.3. MFCC and Its Dynamic Features

The extraction of reliable features is one of the most important issues in speech recognition. There are a large number of features we can use. However, as discussed in Chapter 4, the *curse-of-dimensionality* problem reminds us that the amount of training data is always limited. Therefore, incorporation of additional features may not lead to any measurable error reduction. This does not necessarily mean that the additional features are poor ones, but rather that we may have insufficient data to reliably model those features.

The first feature we use is the speech waveform itself. In general, time-domain features are much less accurate than frequency-domain features such as the mel-frequency cepstral coefficients (MFCC) discussed in Chapter 6 [23]. This is because many features such as formants, useful in discriminating vowels, are better characterized in the frequency domain with a low-dimension feature vector.

As discussed in Chapter 2, temporal changes in the spectra play an important role in human perception. One way to capture this information is to use *delta coefficients* that measure the change in coefficients over time. Temporal information is particularly complementary to HMMs, since HMMs assume each frame is independent of the past, and these dynamic features broaden the scope of a frame. It is also easy to incorporate new features by augmenting the static feature.

When 16-kHz sampling rate is used, a typical state-of-the-art speech system can be build based on the following features.

- 13th-order MFCC $\mathbf{c}_k$;
- 13th-order 40-msec 1st-order delta MFCC computed from $\Delta\mathbf{c}_k = \mathbf{c}_{k+2} - \mathbf{c}_{k-2}$;
- 13th-order 2nd-order delta MFCC computed from $\Delta\Delta\mathbf{c}_k = \Delta\mathbf{c}_{k+1} - \Delta\mathbf{c}_{k-1}$;

The short-time analysis Hamming window of 256 ms is typically used to compute the MFCC $\mathbf{c}_k$. The window shift is typically 10 ms. Please note that $c_k[0]$ is included in the feature vector, which has a role similar to that of the log power. The feature vector used for speech recognition is typically a combination of these features

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{c}_k \\ \Delta\mathbf{c}_k \\ \Delta\Delta\mathbf{c}_k \end{pmatrix} \tag{9.6}$$

The relative error reduction with a typical speech recognition system is illustrated in Table 9.2. As you can see from the table, the 13th-order MFCC outperforms 13th-order LPC cepstrum coefficients, which indicates that perception-motivated mel-scale representation indeed helps recognition. In a similar manner, perception-based LPC features such as PLP can achieve similar improvement. The MFCC order has also been studied experimentally for speech recognition. The higher-order MFCC does not further reduce the error rate in comparison with the 13th-order MFCC, which indicates that the first 13 coefficients already con-

tain most salient information needed for speech recognition. In addition to mel-scale representation, another perception-motivated feature such as the first- and second-order delta features can significantly reduce the word recognition error, while the higher-order delta features provide no further information.

Feature extraction in these experiments is typically optimized together with the classifier, since there are a number of modeling assumptions, such as the diagonal covariance in the Gaussian density function, that are closely related to what features to use. It is possible that these relative error reductions would vary if a different speech recognizer were used.

Table 9.2 Relative error reduction with different features.

Feature set	Relative error reduction
13th-order LPC cepstrum coefficients	Baseline
13th-order MFCC	+10%
16th-order MFCC	+0%
+1st- and 2nd-order dynamic features	+20%
+3rd-order dynamic features	+0%

9.3.4. Feature Transformation

Before you use feature vectors such as MFCC for recognition, you can preprocess or transform them into a new space that alleviates environment noise, channel distortion, and speaker variations. You can also transform the features that are most effective for preserving class separability so that you can further reduce the recognition error rate. Since we devote Chapter 10 completely to environment and channel normalization, we briefly discuss here how we can transform the feature vectors to improve class separability.

To further reduce the dimension of the feature vector, you can use a number of dimension reduction techniques to map the feature vector into more effective representations. If the mapping is linear, the mapping function is well defined and you can find the coefficients of the linear function so as to optimize your objective functions. For example, when you combine the first- and second-order dynamic features with the static MFCC vector, you can use the *principal-component analysis* (PCA) (also known as *Karhunen-Loeve* transform) [32] to map the combined feature vector into a smaller dimension. The optimum basis vectors of the principal-component analysis are the eigenvectors of the covariance matrix of a given distribution. In practice, you can compute the eigenvectors of the autocorrelation matrix as the basis vectors. The effectiveness of the transformed vector, in terms of representing the original feature vector, is determined by the corresponding eigenvalue of each value in the vector. You can discard the feature with the smallest eigenvalue, since the mean-square error between the transformed vector and the original vector is determined by the eigenvalue of each feature in the vector. In addition, the transformed feature vector is uncorrelated. This is particularly suitable for the Gaussian probability density function with a diagonal covariance matrix.

The recognition error is the best criterion for deciding what feature sets to use. However, it is hard to obtain such an estimate to evaluate feature sets systematically. A simpler approach is to use within-class and between-class scatter matrices to formulate criteria of class separability, which is also called as *Linear Discriminant Analysis* (LDA) transformation. We can compute the within-class scatter matrix as:

$$S_w = \sum_{\mathbf{x}\in\omega_i} P(\omega_i) E\{(\mathbf{x}-\boldsymbol{\mu}_i)(\mathbf{x}-\boldsymbol{\mu}_i)^t \mid \omega_i\} = \sum_{\mathbf{x}\in\omega_i} P(\omega_i)\Sigma_i \tag{9.7}$$

where the sum is for all the data **x** within the class ω_i. This is the scatter of samples around their respective class mean. On the other hand, the between-class scatter matrix is the scatter of the expected vectors around the mixture mean:

$$S_B = \sum_{\boldsymbol{\mu}_i\in\omega_i} P(\omega_i)(\boldsymbol{\mu}_i - \mathbf{m}_0)(\boldsymbol{\mu}_i - \mathbf{m}_0)^t \tag{9.8}$$

where $\mathbf{m}_0$ represents the expected mean vector of the mixture distribution:

$$\mathbf{m}_0 = E\{\mathbf{x}\} = \sum_{\omega_i} P(\omega_i)\mathbf{m}_i \tag{9.9}$$

To formulate criteria to transform feature vector **x**, we need to derive the linear transformation matrix **A**. One of the measures can be the trace of $S_w^{-1}S_B$:

$$J = tr(S_w^{-1}S_B) \tag{9.10}$$

The trace is the sum of the eigenvalues of $S_w^{-1}S_B$ and hence the sum of the variances in the principal directions. The number is larger when the between-class scatter is large or the within-class scatter is small. You can derive the transformation matrix based on the eigenvectors of $S_w^{-1}S_B$. In a manner similar to PCA, you can reduce the dimension of the original input feature vector by discarding the smallest eigenvalues [16, 54].

Researchers have used the LDA method to measure the effectiveness of several feature vectors for speaker normalization [41]. Other feature processing techniques designed for speaker normalization include neural-network-based speaker mapping [51], frequency warping for vocal tract normalization (VTN) via mel-frequency scaling [67, 100], and bilinear transformation [2].

To reduce interspeaker variability by a speaker-specific frequency warping, you can simply shift the center frequencies of the mel-spaced filter bank. Let $k\Delta f_{mel}$, k = 1, …, K, denote the center frequencies in mel-scale. Then the center frequencies in hertz for a warping factor of α are computed by Eq. (9.11) before the cosine transformation of the MFCC feature vector.

$$f_{Hz}^{\alpha}(k\Delta f_{mel}) = 700(10^{k\Delta f_{mel}/2595} - 1)/\alpha \tag{9.11}$$

The warping factor is estimated for each speaker by computing the likelihood of the training data for feature sets obtained with different warping factors using the HMM. The relative error reduction based on the feature transformation method has been limited, typically under 10%.

9.4. PHONETIC MODELING—SELECTING APPROPRIATE UNITS

As discussed in Chapter 2, the phonetic system is related to a particular language. We focus our discussion on language-independent technologies but use English in our examples to illustrate how we can use the language-independent technologies to model the salient phonetic information in the language. For general-purpose large-vocabulary speech recognition, it is difficult to build whole-word models because:

- Every new task contains novel words without any available training data, such as proper nouns and newly invented jargons.
- There are simply too many words, and these different words may have different acoustic realizations, as illustrated in Chapter 2. It is unlikely that we have sufficient repetitions of these words to build context-dependent word models.

How to select the most basic units to represent salient acoustic and phonetic information for the language is an important issue in designing a workable system. At a high level, there are a number of issues we must consider in choosing appropriate modeling units.

- The unit should be *accurate*, to represent the acoustic realization that appears in different contexts.
- The unit should be *trainable*. We should have enough data to estimate the parameters of the unit. Although words are accurate and representative, they are the least trainable choice in building a working system, since it is nearly impossible to get several hundred repetitions for all the words, unless we are using a speech recognizer that is domain specific, such as a recognizer designed for digits only.
- The unit should be *generalizable*, so that any new word can be derived from a predefined unit inventory for task-independent speech recognition. If we have a fixed set of word models, there is no obvious way for us to derive the new word model.

A practical challenge is how to balance these selection criteria for speech recognition. In this section we compare a number of units and point out their strengths and weaknesses in practical applications.

9.4.1. Comparison of Different Units

What is a unit of language? In English, words are typically considered as a principal carrier of meaning and are seen as the smallest unit that is capable of independent use. As the most natural unit of speech, whole-word models have been widely used for many speech recognition systems. A distinctive advantage of using word models is that we can capture phonetic coarticulation inherent within these words. When the vocabulary is small, we can create word models that are context dependent.

For example, if the vocabulary is English digits, we can have different word models for the word *one* to represent the word in different contexts. Thus each word model is dependent on its left and right context. If someone says *three one two*, the recognizer uses the word model *one* that specifically depends on the left context *three* and right context *two*. Since the vocabulary is small (10), we need to have only 10*10*10=1000 word models, which is achievable when you collect enough training data. With context-dependent, or even context-independent, word models, a wide range of phonological variations can be automatically accommodated. When these word models are adequately trained, they usually yield the best recognition performance in comparison to other modeling units. Therefore, for small vocabulary recognition, whole-word models are widely used, since they are both *accurate* and *trainable*, and there is no need to be *generalizable*.

While words are suitable units for small-vocabulary speech recognition, they are not a practical choice for large-vocabulary continuous speech recognition. First, each word has to be treated individually, and data cannot be shared across word models; this implies a prohibitively large amount of training data and storage. Second, for some task configurations, the recognition vocabulary may consist of words that never appeared in the training data. As a result, some form of word-model composition technique is required to generate word models. Third, it is very expensive to model interword coarticulation effects or adapt a word-based system for a new speaker, a new channel, or new context usage.

To summarize, word models are *accurate* if enough data are available. Thus, they are *trainable* only for small tasks. They are typically not *generalizable*.

Alternatively, there are only about 50 phones in English, and they can be sufficiently trained with just a few hundred sentences. Unlike word models, phonetic models provide no training problem. Moreover, they are also vocabulary independent by nature and can be trained on one task and tested on another. Thus, phones are more *trainable* and *generalizable*. However, the phonetic model is inadequate because it assumes that a phoneme in any context is identical. Although we may try to say each word as a concatenated sequence of independent phonemes, these phonemes are not produced independently, because our articulators cannot move instantaneously from one position to another. Thus, the realization of a phoneme is strongly affected by its immediately neighboring phonemes. For example, if context-independent phonetic models are used, the same model for *t* must capture various events, such as flapping, unreleased stops, and realizations in */t s/* and */t r/*. Then, if */t s/* is the only context in which *t* occurs in the training, while /*t r*/ is the only context in the testing, the model used is highly inappropriate. While word models are not generalizable, phonetic models overgeneralize and, thus, lead to less accurate models.

A compromise between the word and phonetic model is to use larger units such as *syllables*. These units encompass phone clusters that contain the most variable contextual effects. However, while the central portions of these units have no contextual dependencies, the beginning and ending portions are still susceptible to some contextual effects. There are only about 1200 tone-dependent syllables in Chinese and approximately 50 syllables in Japanese, which makes syllable a suitable unit for these languages. Unfortunately, the large number of syllables (over 30,000) in English presents a challenge in terms of trainability.

9.4.2. Context Dependency

If we make units context dependent, we can significantly improve the recognition accuracy, provided there are enough training data to estimate these context-dependent parameters. Context-dependent phonemes have been widely used for large-vocabulary speech recognition, thanks to its significantly improved accuracy and trainability. A context usually refers to the immediately left and/or right neighboring phones.

A *triphone* model is a phonetic model that takes into consideration both the left and the right neighboring phones. If two phones have the same identity but different left or right contexts, they are considered different triphones. We call different realizations of a phoneme *allophones*. Triphones are an example of allophones.

The left and right contexts used in triphones, while important, are only two of many important contributing factors that affect the realization of a phone. Triphone models are powerful because they capture the most important coarticulatory effects. They are generally much more consistent than context-independent phone models. However, as context-dependent models generally have increased parameters, trainability becomes a challenging issue. We need to balance the trainability and accuracy with a number of parameter-sharing techniques.

Modeling interword context-dependent phones is complicated. For example, in the word *speech*, pronounced /*s p iy ch*/, both left and right contexts for /*p*/ and /*iy*/ are known, while the left context for /s/ and the right context for /*ch*/ are dependent on the preceding and following words in actual sentences. The juncture effect on word boundaries is one of the most serious coarticulation phenomena in continuous speech, especially with short function words like *the* or *a*. Even with the same left and right context identities, there may be significantly different realizations for a phone at different word positions (the *beginning*, *middle*, or *end* of a word). For example, the phone /*t*/ in *that rock* is almost extinct, while the phone /*t*/ in the middle of *theatrical* sounds like /*ch*/. This implies that different word positions have effects on the realization of the same triphone.

In addition to the context, stress also plays an important role in the realization of a particular phone. Stressed vowels tend to have longer duration, higher pitch, and more intensity, while unstressed vowels appear to move toward a neutral, central *schwa*-like phoneme. Agreement about the phonetic identity of a syllable has been reported to be greater in stressed syllables for both humans and automatic phone recognizers. In English, *word-level* stress is referred to as *free stress*, because the stressed syllable can take on any position within a word, in contrast to *bound stress* found in languages such as French and Polish,

where the position of the stressed syllable is fixed within a word. Therefore, stress in English can be used as a constraint for lexical access. In fact, stress can be used as a unique feature to distinguish a set of word pairs, such as ***im****port* vs. *im****port***, and ***ex****port* vs. *ex****port.*** For example, the phone set used for Whisper, such as /*er*/-/*axr*/ and /*ah*/-/*ix*/-/*ax*/, describes these stressed and unstressed vowels. One example illustrating how stress can significantly affect the realization of phone is demonstrated in Figure 9.3, where phone /t/ in word *Italy* vs *Italian* is pronounced differently in American English due the location of the stress, albeit the triphone context is identical for both words.

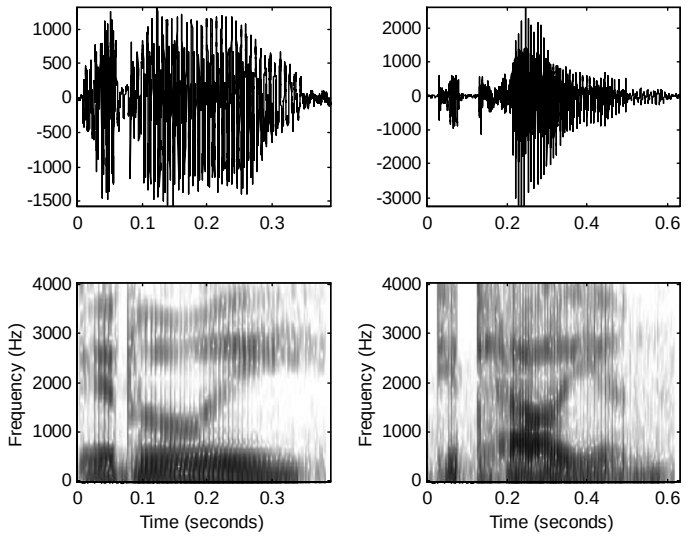


Figure 9.3 The importance of stress is illustrated in *Italy* vs. *Italian* for phone /t/.

Sentence-level stress, on the other hand, represents the overall stress pattern of continuous speech. While sentence-level stress does not change the meaning of any particular lexicon item, it usually increases the relative prominence of portions of the utterance for the purpose of contrast or emphasis. *Contrastive stress* is normally used to coordinate constructions such as *there are* ***import*** *records and there are* ***domesti****c ones*, as well as for the purpose of correction, as in *I said* ***import****, not* ***export.*** *Emphatic* stress is commonly used to distinguish a sentence from its negation, e.g., *I* ***did*** *have dinner*. Sentence-level stress is very hard to model without incorporating high-level semantic and pragmatic knowledge. In most state-of- the-art speech recognition systems, only word-level stress is used for creating allophones.

9.4.3. Clustered Acoustic-Phonetic Units

Triphone modeling assumes that every triphone context is different. Actually, many phones have similar effects on the neighboring phones. The position of our articulators has an important effect on how we pronounce neighboring vowels. For example, /b/ and /p/ are both labial stops and have similar effects on the following vowel, while /r/ and /w/ are both liquids and have similar effects on the following vowel. Contrary to what we illustrate in Figure 9.1, Figure 9.4 illustrates this phenomenon. It is desirable to find instances of similar contexts and merge them. This would lead to a much more manageable number of models that can be better trained.

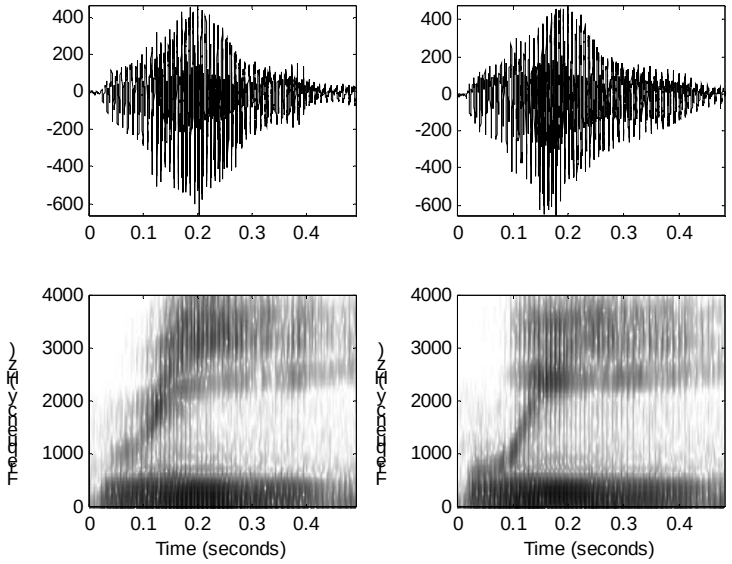


Figure 9.4 The spectrograms for the phoneme /iy/ with two different *left-contexts* are illustrated. Note that /r/ and /w/ have similar effects on /iy/. This illustrates that different left-contexts may have similar effects on a phone.

The trainability and accuracy balance between phonetic and word models can be generalized further to model subphonetic events. In fact, both phonetic and subphonetic units have the same benefits, as they share parameters at unit level. This is the key benefit in comparison to the word units. Papers by [11, 45, 57, 66, 111] provide examples of the application of this concept to cluster hidden Markov models. For subphonetic modeling, we can treat the state in phonetic HMMs as the basic subphonetic unit. Hwang and Huang further generalized clustering to the state-dependent output distributions across different phonetic models [57]. Each cluster thus represents a set of similar Markov states and is called a *senone* [56]. A subword model is thus composed of a sequence of senones after the cluster-

ing is finished. The optimal number of senones for a system is mainly determined by the available training corpus and can be tuned on a development set.

Each allophone model is an HMM made of states, transitions, and probability distributions. To improve the reliability of the statistical parameters of these models, some distributions can be tied. For example, distributions for the central portion of an allophone may be tied together to reflect the fact that they represent the stable (context-independent) physical realization of the central part of the phoneme, uttered with a stationary configuration of the vocal tract. Clustering at the granularity of the state rather than the entire model can keep the dissimilar states of two models apart while the other corresponding states are merged, thus leading to better parameter sharing.

Figure 9.5 illustrates how state-based clustering can lead to improved representations. These two HMMs come from the same phone class with a different right context, leading to very different output distributions in the last state. As the left contexts are identical, the first and second output distributions are almost identical. If we measure the overall model similarity based on the accumulative overall output distribution similarities of all states, these two models may be clustered, leading to a very inaccurate distribution for the last state. Instead, we cluster the first two output distributions while leaving the last one intact.

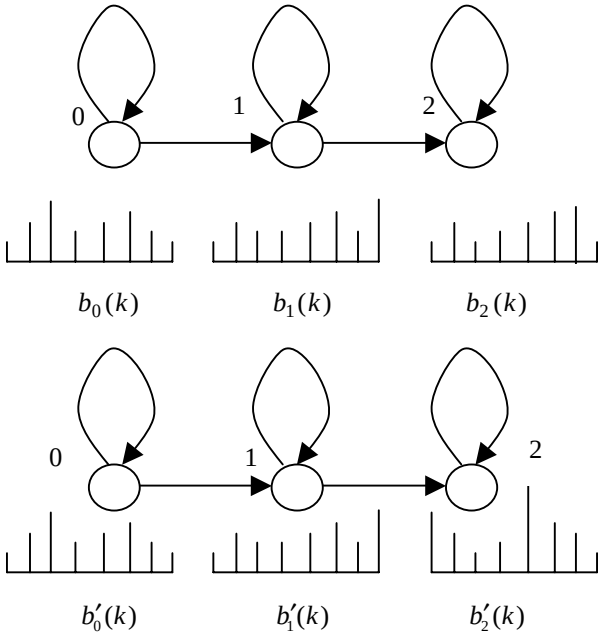


Figure 9.5 State-based vs. model-based clustering. These two models are very similar ,as both the first and the second output distributions are almost identical. The key difference is the output distribution of the third state. If we measure the overall model similarity, which is often based on the accumulative output distribution similarities of all states, these two models may be clustered, leading to a very inaccurate distribution for the last state. If we cluster output distributions at state level, we can cluster the first two output distributions while leaving the last ones intact, leading to more accurate representations.

Table 9.3 Some example questions used in building senone trees.

Questions	Phones in Each Question Category
Aspseg	*hh*
Sil	*sil*
Alvstp	*d t*
Dental	*dh th*
Labstp	*b p*
Liquid	*l r*
Lw	*l w*
S/Sh	*s sh*
Sylbic	*er axr*
Velstp	*g k*
Affric	*ch jh*
Lqgl-B	*l r w*
Nasal	*m n ng*
Retro	*r er axr*
Schwa	*ax ix axr*
Velar	*ng g k*
Fric2	*th s sh f*
Fric3	*dh z zh v*
Lqgl	*l r w y*
S/Z/Sh/Zh	*s z sh zh*
Wglide	*uw aw ow w*
Labial	*w m b p v*
Palatl	*y ch jh sh zh*
Yglide	*iy ay ey oy y*
High	*ih ix iy uh uw y*
Lax	*eh ih ix uh ah ax*
Low	*ae aa ao aw ay oy*
Orstp2	*p t k*
Orstp3	*b d g*
Alvelr	*n d t s z*
Diph	*uw aw ay ey iy ow oy*
Fric1	*dh th s sh z zh v f*
Round	*uh ao uw ow oy w axr er*
Frnt-R	*ae eh ih ix iy ey ah ax y aw*
Tense	*iy ey ae uw ow aa ao ay oy aw*
Back-L	*uh ao uw ow aa er axr l r w aw*
Frnt-L	*ae eh ih ix iy ey ah ax y oy ay*
Back-R	*uh ao uw ow aa er axr oy l r w ay*
Orstp1	*b d g p t k ch jh*
Vowel	*ae eh ih ix iy uh ah ax aa ao uw aw ay ey ow oy er axr*
Son	*ae eh ih ix iy ey ah ax oy ay uh ao uw ow aa er axr aw l r w y*
Voiced	*ae eh ih ix iy uh ah ax aa ao uw aw ay ey ow oy l r w y er axr m n ng jh b d dh g v z zh*

There are two key issues in creating trainable context-dependent phonetic or subphonetic units:

- We need to enable better parameter sharing and smoothing. As Figure 9.4 illustrates, many phones have similar effects on the neighboring phones. If the acoustic realization is indeed identical, we tie them together to improve trainability and efficiency.
- Since the number of triphones in English is very large (over 100,000), there are many new or unseen triphones that are in the test set but not in the training set. It is important to map these unseen triphones into appropriately trained triphones.

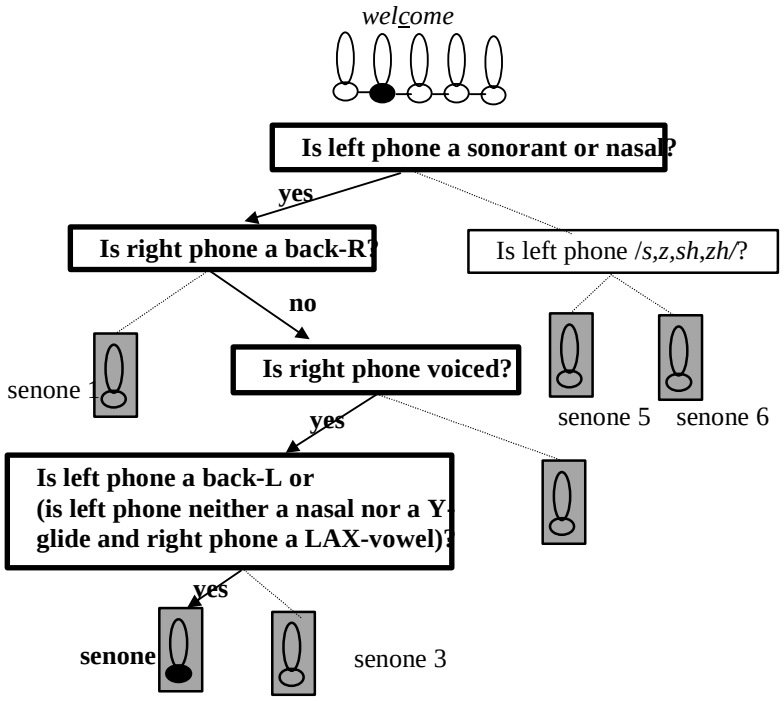


Figure 9.6 A decision tree for classifying the second state of *K*-triphone HMMs [48].

As discussed in Chapter 4, a decision tree is a binary tree to classify target objects by asking binary questions in a hierarchical manner. Modeling unseen triphones is particularly important for *vocabulary independence*, since it is difficult to collect a training corpus which covers enough occurrences of every possible subword unit. We need to find models that are accurate, trainable, and especially generalizable. The senonic decision tree classifies Markov states of triphones represented in the training corpus by asking linguistic questions composed of conjunctions, disjunctions, and/or negations of a set of predetermined simple categorical linguistic questions. Examples of these simple categorical questions are: *Is the left-context*

phone a fricative? Is the right-context phone a front vowel? The typical question set used in Whisper to generate the senone tree is shown in Table 9.3. So, for each node in the tree, we check whether its left or right phone belongs to one of the categories. As discussed in Chapter 4, we measure the corresponding entropy reduction or likelihood increase for each question and select the question that has the largest entropy decrease to split the node. Thus, the tree can be automatically constructed by searching, for each node with the best question that renders the maximum entropy decrease. Alternatively, complex questions can be formed for each node for improved splitting.

When we grow the tree, it needs to be pruned using cross-validation as discussed in Chapter 4. When the algorithm terminates, the leaf nodes of the tree represent the senones to be used. Figure 9.6 shows an example tree we built to classify the second state of all /*k*/ triphones seen in a training corpus. After the tree is built, it can be applied to the second state of *any /k/* triphone, thanks to the generalizability of the binary tree and the general linguistic questions. Figure 9.6 indicates that the second state of the */k/* triphone in *welcome* is mapped to the second senone, no matter whether this triphone occurs in the training corpus or not.

In practice, senone models significantly reduce the word recognition error rate in comparison with the model-based clustered triphone models, as illustrated in Table 9.4. It is the senonic model's significant reduction of the overall system parameters that enables the continuous mixture HMMs to perform well for large-vocabulary speech recognition [56].

Table 9.4 Relative error reductions for different modeling units.

Units	Relative Error Reductions
Context-independent phone	Baseline
Context-dependent phone	+25%
Clustered triphone	+15%
Senone	+24%

9.4.4. Lexical Baseforms

When appropriate subword units are used, we must have the correct pronunciation for each word so that concatenation of the subword unit can accurately represent the word to be recognized. The dictionary represents the standard pronunciation used as a starting point for building a workable speech recognition system. We also need to provide alternative pronunciations to words such as *tomato* that may have very different pronunciations. For example, the COMLEX dictionary from LDC has about 90,000 baseforms that cover most words used in many years of *The Wall Street Journal*. The CMU Pronunciation Dictionary, which was optimized for continuous speech recognition, has about 100,000 baseforms.

In continuous speech recognition, we must also use phonologic rules to modify interword pronunciations or to have reduced sounds. Assimilation is a typical coarticulation phenomenon—a change in a segment to make it more like a neighboring segment. Typical examples include phrases such as *did you /d ih jh y ah/, set you /s eh ch er/, last year / l ae s ch*

iy r/, because you've /b iy k ah zh uw v/, etc. Deletion is also common in continuous speech. For example, /t/ and /d/ are often deleted before a consonant. Thus, in conversational speech, you may find examples like *find him /f ay n ix m/, around this /ix r aw n ih s/,* and *Let me in /l eh m eh n/*.

Dictionaries often don't include proper names. For example, the 20,000 names included in the COMPLEX dictionary are a small fraction of 1–2 million names in the USA. To deal with these new words, we often have to derive their pronunciation automatically. These new words have to be added on the fly, either by the user or through interface from speech-aware applications. Unlike Spanish or Italian, the rule-based letter-to-sound (LTS) conversion for English is often impractical, since so many words in English don't follow the phonological rules. A trainable LTS converter is attractive, since its performance can be improved by constantly learning from examples so that it can generalize rules for the specific task. Trainable LTS converters can be based on neural networks, HMMs, or the CART described in Chapter 4. In practice, the CART-based LTS has a very accurate performance [10, 61, 71, 89].

When the CART is used, the basic YES-NO question for LTS conversion looks like: *Is the second right letter 'p'?* or: *Is the first left output phone /ay/?* The question for letters and phones can be on either the left or the right side. The range of question positions should be long enough to cover the most important phonological variations. Empirically, the 10-letter window (5 for left letter context and 5 for right letter context) and 3-phone window context is generally sufficient. A primitive set of questions can include all the singleton questions about each letter or phone identity. If we allow the node to have a complex question—that is, a combination of primitive questions—the depth of the tree can be greatly reduced and performance improved. For example, a complex question: *Is the second left letter 't' and the first left letter 'i' and the first right letter 'n'?* can capture *o* in the common suffix *tion* and convert it to the correct phone. Complex questions can also alleviate possible data-fragmentation problems caused by the greedy nature of the CART algorithm.

Categorical questions can be formed in both the letter and phone domains with our common linguistic knowledge. For example, the most often used set includes the letter or phone clusters for vowels, consonants, nasals, liquids, fricatives, and so on. In growing the decision tree, the context distance also plays a major role in the overall quality. It is very important to weight the entropy reduction according to the distance (either letter or phoneme) to avoid overgeneralization, which forces the tree to look more carefully at the *nearby* context than at the *far-away* context. Each leaf of the tree has a probability distribution for letter-to-phoneme mapping.

There are a number of ways to improve the effectiveness of the decision tree. First, pruning controls the tree's depth. For example, certain criteria have to be met for a node to be split. Typically splitting requires a minimum number of counts and a minimum entropy reduction. Second, the distribution at the leaves can be smoothed. For example, a leaf distribution can be interpolated with the distributions of its ancestor nodes using deleted-interpolation. Finally, we can partition the training data and build multiple trees with different prediction capabilities. These trees accommodate different phonological rules with different language origins.

When the decision tree is used to derive the phonetic pronunciation, the phonetic conversion error is about 8% for the *Wall Street Journal* newspaper text corpora [61]. These errors can be broadly classified into two categories. The first includes errors of proper nouns and foreign words. For example, *Pacino* can be mistakenly converted *to /p ax s iy n ow* / instead of /*p ax ch iy n ow*/. The second category includes generalization errors. For example, *shier* may be converted to /*sh ih r*/ instead of the correct pronunciation /*sh ay r*/ if the word *cashier* /*k ae sh ih r*/ appears in the training data. The top three phone confusion pairs are /*ix/ax/*, */dx/t/*, and */ae/ax/*. The most confusing pair is */ix/ax/*. This is not surprising, because */ix/ax/* is among the most inconsistent transcriptions in most of the published dictionaries. There is no consensus for */ix/ax/* transcription among phoneticians.

Although automatic LTS conversion has a reasonable accuracy, it is hardly practical if you don't use an exception dictionary. This is especially true for proper nouns. In practice, you can often ask the person who knows how to pronounce the word to either speak or write down the correct phonetic pronunciation, updating the exception dictionary if the correct one disagrees with what the LTS generates. When acoustic examples are available, you can use the decision tree to generate multiple results and use these results as a language model to perform phone recognition on the acoustic examples. The best overall acoustic and LTS probability can be used as the most likely candidate in the exception dictionary. Since there may be many ways to pronounce a word, you can keep multiple pronunciations in the dictionary with a probability for each possible one. If the pronunciation probability is inaccurate, an increase in multiple pronunciations essentially increases the size and confusion of the vocabulary, leading to increased speech recognition error rate.

Even if you have accurate phonetic baseform, pronunciations in spontaneous speech differ significantly from the standard baseform. Analysis of manual phonetic transcription of conversational speech reveals a large number (> 20%) of cases of genuine ambiguity: instances where human labelers disagree on the identity of the surface form [95]. For example, the word *because* has more than 15 different pronunciation variations, such as */b iy k ah z/, /b ix k ah z/, /k ah z/, /k ax, z/, /b ix k ax z/, /b ax k ah z/, /b ih k ah z/, /k s/, /k ix z/, /k ih z/,/b iy k ah s/, /b iy k ah/, /b iy k ah zh/, /ax z/*, etc., in the context of conversational speech [39]. To characterize the acoustic evidence in the context of this ambiguity, you can partly resolve the ambiguity by deriving a suitable phonetic baseform from speech data [29, 95, 97]. This is because the widespread variation can be due either to a lexical fact (such as that the word *because* can be *'cause* with informal speech) or to the dialect differences. African American Vernacular English has many vowels different from general American English.

To incorporate widespread pronunciations, we can use a probabilistic finite state machine to model each word's pronunciation variations, as shown in Figure 9.7. The probability with each arc indicates how likely that path is to be taken, with all the arcs that leave a node summing to 1. As with HMMs, these weights can be estimated from real corpus for improved speech recognition [20, 85, 102, 103, 110]. In practice, the relative error reduction of using probabilistic finite state machines is very modest (5–10%).

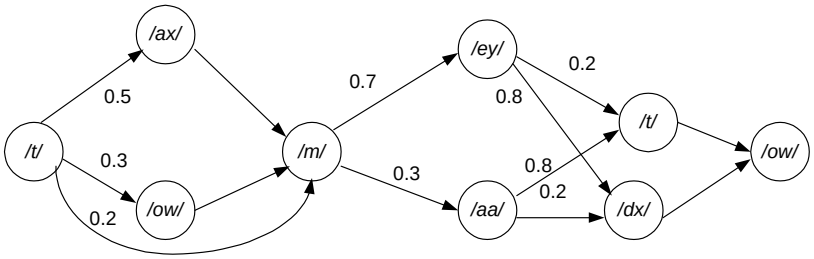


Figure 9.7 A possible pronunciation network for word *tomato*. The vowel /*ey*/ is more likely to flap, thereby having a higher transition probability into /*dx*/.

9.5. ACOUSTIC MODELING—SCORING ACOUSTIC FEATURES

After feature extraction, we have a sequence of feature vectors, **X**, such as the MFCC vector, as our input data. We need to estimate the probability of these acoustic features, given the word or phonetic model, **W**, so that we can recognize the input data for the correct word. This probability is referred to as acoustic probability, $P(\mathbf{X} \mid \mathbf{W})$. In this section we focus our discussion on the HMM. As discussed in Chapter 8, it is the most successful method for acoustic modeling. Other emerging techniques are discussed in Section 9.8.

9.5.1. Choice of HMM Output Distributions

As discussed in Chapter 8, you can use discrete, continuous, or semicontinuous HMMs. When the amount of training data is sufficient, parameter tying becomes unnecessary. A continuous model with a large number of mixtures offers the best recognition accuracy, although its computational complexity also increases linearly with the number of mixtures. On the other hand, the discrete model is computationally efficient, but has the worst performance among the three models. The semicontinuous model provides a viable alternative between system robustness and trainability.

When either the discrete or the semicontinuous HMM is employed, it is helpful to use multiple codebooks for a number of features for significantly improved performance. Each codebook then represents a set of different speech parameters. One way to combine these multiple output observations is to assume that they are independent, computing the output probability as the product of the output probabilities of each codebook. For example, the semicontinuous HMM output probability of multiple codebooks can be computed as the product of each codebook:

$$b_i(\mathbf{x}) = \prod_m \sum_{k=1}^{L^m} f^m(\mathbf{x}^m \mid o_k^m) b_i^m(o_k^m) \tag{9.12}$$

where superscript *m* denotes the codebook-*m* related parameters. Each codebook consists of L^m -mixture continuous density functions.

Following our discussion in Chapter 8, the re-estimation algorithm for the multiple-codebook-based HMM could be extended. Since multiplication of the output probability density of each codebook leads to several independent terms in the *Q*-function, for codebook *m*, $\zeta_t(j,k^m)$ can be modified as follows:

$$\zeta_t(j,k^m) = \frac{\sum_i \alpha_{t-1}(i) a_{ij} b_j^m(k^m) f^m(\mathbf{x}_t \mid v_k^m) \prod_{m \neq n} \sum_k b_j^n(k^n) f^n(\mathbf{x}_t \mid v_k^n) \beta_t(j)}{\sum_k \alpha_T^m(k)} \tag{9.13}$$

Other intermediate probabilities can also be computed in a manner similar to what we discussed in Chapter 8.

Multiple codebooks can dramatically increase the representation power of the VQ codebook and can substantially improve speech recognition accuracy. You can typically build a codebook for $\mathbf{c}_k$, $\Delta\mathbf{c}_k$, and $\Delta\Delta\mathbf{c}_k$, respectively. As energy has a very different dynamic range, you can further improve the performance by building a separate codebook for $c_k[0]$, $\Delta c_k[0]$, and $\Delta\Delta c_k[0]$. In comparison to building a single codebook for $\mathbf{x}_k$ as illustrated in Eq. (9.6), the multiple-codebook system can reduce the error rate by more than 10%.

In practice, the most important parameter for the output probability distribution is the number of mixtures or the size of the codebooks. When there are sufficient training data, relative error reductions with respect to the discrete HMM are those shown in Figure 9.8.

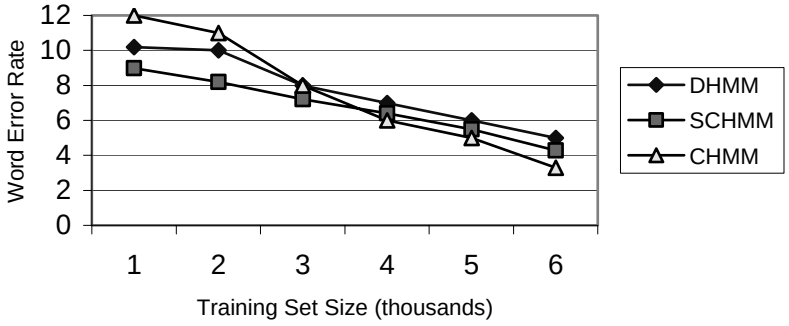


Figure 9.8 Continuous speaker-independent word recognition error rates of the discrete HMM (DHMM), SCHMM, and the continuous HMM (CHMM) with respect to the training set sizes (thousands of training sentences). Both the DHMM and SCHMM have multiple codebooks. The CHMM has 20 mixture diagonal Gaussian density functions.

As you can see from Figure 9.8, the SCHMM offers improved accuracy in comparison with the discrete HMM or the continuous HMM when the amount of training data is limited.

When we increase the training data size, the continuous mixture density HMM starts to outperform both the discrete and the semicontinuous HMM, since the need to share the model parameters becomes less critical.

The performance is also a function of the number of mixtures. With a small number of mixtures, the continuous HMM lacks the modeling power and it actually performs worse than the discrete HMM across the board. Only after we dramatically increase the number of mixtures does the continuous HMM start to offer improved recognition accuracy. The SCHMM can typically reduce the discrete HMM error rate by 10–15% across the board. The continuous HMM with 20 diagonal Gaussian density functions performed worse than either the discrete or the SCHMM when the size of training data was small. It outperformed either the discrete HMM or the SCHMM when sufficient amounts of training data became available. When the amount of training data is sufficiently large, it can reduce the error rate of the semicontinuous HMM by 15–20%.

9.5.2. Isolated vs. Continuous Speech Training

If we build a word HMM for each word in the vocabulary for isolated speech recognition, the training or recognition can be implemented directly, using the basic algorithms introduced in Chapter 8. To estimate model parameters, examples of each word in the vocabulary are collected. The model parameters are estimated from all these examples using the forward-backward algorithm and the reestimation formula. It is not necessary to have precise end-point detection, because the silence model automatically determines the boundary if we concatenate silence models with the word model in both ends.

If subword units,[3] such as phonetic models, are used, we need to share them across different words for large-vocabulary speech recognition. These subword units are concatenated to form a word model, possibly adding silence models at the beginning and end, as illustrated in Figure 9.9.

To concatenate subword units to form a word model, you can have a null transition from the final state of the previous subword HMM to the initial state of the next subword HMM, as indicated by the dotted line in Figure 9.9. As described in Chapter 8, you can estimate the parameters of the concatenated HMM accordingly. Please notice that the added null transition arc should satisfy the probability constraint with the transition probability of each phonetic HMM. The self-loop transition probability of the last state in each individual HMM has the topology illustrated in Figure 9.9. If we estimate these parameters with the concatenated model, the null arc transition probability, a_{ij}^{ε}, should satisfy the constraint $\sum_j (a_{ij} + a_{ij}^{\varepsilon}) = 1$ such that the self-loop transition probability of the last state is no longer equal to 1. For interword concatenation or concatenation involving multiple pronunciations, you can use multiple null arcs to concatenate individual models together.

[3] We have a detailed discussion on word models vs. subword models in Section 9.4.1.

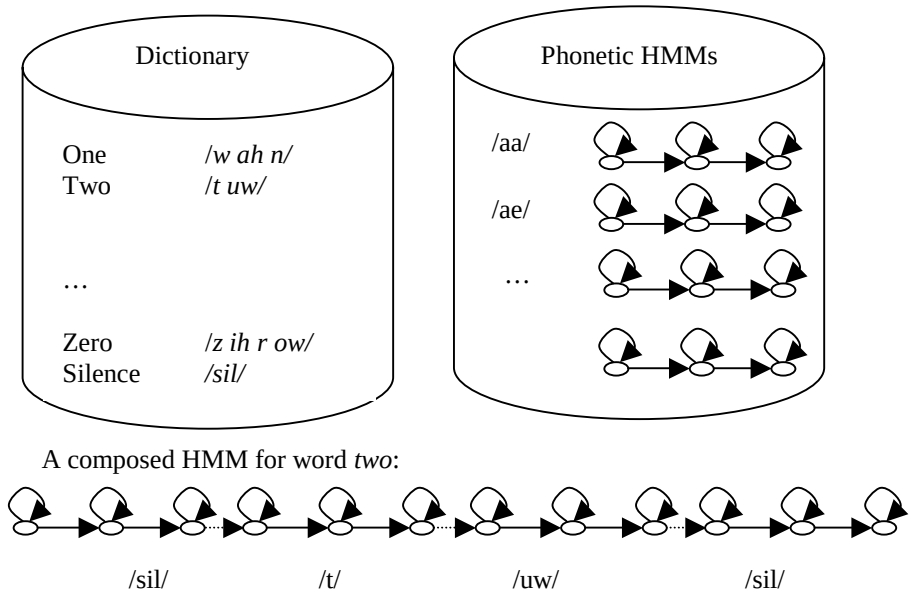


Figure 9.9 The construction of an isolated word model by concatenating multiple phonetic models based on the pronunciation dictionary.

In the example given here, we have ten English digits in the vocabulary. We build an HMM for each English phone. The dictionary provides the information on each word's pronunciation. We have a special wordn, *Silence*, that maps to a /*sil*/ HMM that has the same topology as the standard phonetic HMM. For each word in the vocabulary we first derive the phonetic sequence for each word from the dictionary. We link these phonetic models together to form a word HMM for each word in the vocabulary. The link between two phonetic models is shown in the figure as the dotted arrow.

For example, for word *two*, we create a word model based on the beginning silence /*sil*/, phone /*t*/, phone /*uw*/, and ending silence /*sil*/. The concatenated word model is then treated in the same manner as a standard large composed HMM. We use the standard forward-backward algorithm to estimate the parameters of the composed HMM from multiple sample utterances of the word *two*. After several iterations, we automatically get the HMM parameters for /*sil*/, /*t*/, and /*uw*/. Since a phone can be shared across different words, the phonetic parameters may be estimated from acoustic data in different words.

The ability to automatically align each individual HMM to the corresponding unsegmented speech observation sequence is one of the most powerful features in the forward-backward algorithm. When the HMM concatenation method is used for continuous speech, you need to compose multiple words to form a sentence HMM based on the transcription of the utterance. In the same manner, the forward-backward algorithm absorbs a range of possible word boundary information of models automatically. There is no need to have a precise segmentation of the continuous speech.

In general, to estimate the parameters of the HMM, each word is instantiated with its concatenated word model (which may be a concatenation of subword models). The words in the sentence are concatenated with optional silence models between them. If there is a need to modify interword pronunciations due to interword pronunciation change, such as *wan**t** **y**ou*, you can add a different optional phonetic sequence for ***t-y*** in the concatenated sentence HMM.

In the digit recognition example, if we have a continuous training utterance *one three*, we compose a sentence HMM, as shown in Figure 9.10, where we have an optional silence HMM between the words *one* and *three*, linked with a null transition from the last state of the word model *one* to the thirst state of the word model *three*. There is also a direct null arc connection between the models *one* and *three* because a silence may not exist in the training example. These optional connections ensure that all the possible acoustic realizations of the natural continuous speech are considered, so that the forward-backward algorithm can automatically discover the correct path and accurately estimate the corresponding HMM from the given speech observation.

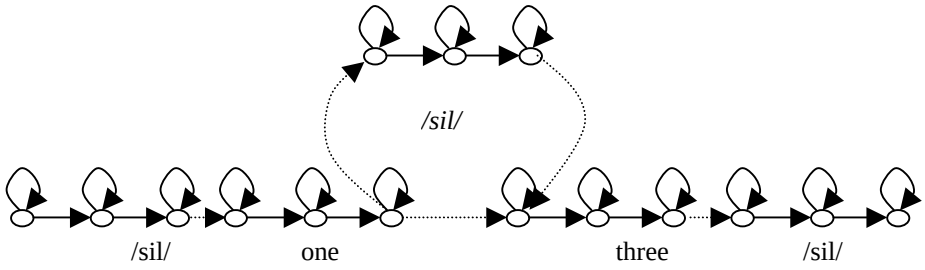


Figure 9.10 A composed sentence HMM. Each word can be a word HMM or a composed phonetic word HMM, as illustrated in Figure 9.9.

In general, the concatenated sentence HMM can be trained using the forward-backward algorithm with the corresponding observation sequence. Since the entire sentence HMM is trained on the entire observation sequence for the corresponding sentence, most possible word boundaries are inherently considered. Parameters of each model are based on those state-to-speech alignments. It does not matter where the word boundaries are. Such a training method allows complete freedom to align the sentence model against the observation, and no explicit effort is needed to find word boundaries.

In speech decoding, a word may begin and end anywhere within a given speech signal. As the word boundaries cannot be detected accurately, all possible beginning and end points have to be accounted for. This converts a linear search (as for isolated word recognition) to a tree search, and a polynomial recognition algorithm to an exponential one. How to design an efficient decoder is discussed in Chapters 12 and 13.

9.6. ADAPTIVE TECHNIQUES—MINIMIZING MISMATCHES

As Figure 1.2 illustrated, it is important to adapt both acoustic models and language models for new situations. A decent model can accommodate a wide range of variabilities. However, the mismatch between the model and operating conditions always exists. One of the most important factors in making a speech system usable is to minimize the possible mismatch dynamically with a small amount of *calibration data*. Adaptive techniques can be used to modify system parameters to better match variations in microphone, transmission channel, environment noise, speaker, style, and application contexts. As a concrete example, speaker-dependent systems can provide a significant word error-rate reduction in comparison to speaker-independent systems if a large amount of speaker-dependent training data exists [50]. Speaker-adaptive techniques can bridge the gap between these two configurations with a small fraction of the speaker-specific training data needed to build a full speaker-dependent system. These techniques can also be used incrementally as more speech is available from a particular speaker. When speaker-adaptive models are build, you can have not only improved accuracy but also improved speed and potentially reduced model parameter sizes because of accurate representations, which is particularly appealing for practical speech recognition.

There are a number of ways to use adaptive techniques to minimize mismatches. You can have a nonintrusive adaptation process that works in the background all the time. This is typically unsupervised, using only the outcome of the recognizer (with a high confidence score, as discussed in Section 9.7) to guide the model adaptation. This approach can continuously modify the model parameters so that any nonstationary mismatches can be eliminated. As discussed in Chapter 13, systems that are required to transcribe speech in a non-real-time fashion may use multiple recognition passes. You can use unsupervised adaptation on the test data to improve the models after each pass to improve performance for a subsequent recognition pass.

Since the use of recognition results may be imperfect, there is a possibility of divergence if the recognition error rate is high. If the error rate is low, the adaptation results may still not be as good as supervised adaptation in which the correct transcription is provided for the user to read, a process referred to as the *enrollment process*. In this process you can check a wide range of parameters as follows:

- Check the background noise by asking the user not to speak.
- Adjust the microphone gain by asking the user to speak normally.
- Adapt the acoustic parameters by asking the user to read several sentences.
- Change the decoder parameters for the best speed with no loss of accuracy.
- Compose dynamically new enrollment sentences based on the user-specific error patterns.

The challenge for model adaptation is that we can use only a small amount of observable data to modify model parameters. This constraint requires different modeling strategies from the ones we discussed in building the baseline system, as the amount of training data is

generally sufficient for offline training. In this section we focus on a number of adaptive techniques that can be applied to compensate either speaker or environment variations. Most of these techniques are model-based, since the acoustic model parameters rather than the acoustic feature vectors are adapted. We use speaker-adaptation examples to illustrate how these techniques can be used to improve system performance. We can generalize to environment adaptation by using environment-specific adaptation data and a noise-compensation model, which we discuss in Chapter 10. In a similar manner, we can modify the language model as discussed in Chapter 11.

9.6.1. Maximum a Posteriori (MAP)

Maximum a posteriori (MAP) estimation, as discussed in Chapter 4, can effectively deal with data-sparse problems, as we can take advantage of prior information about existing models. We can adjust the parameters of pretrained models in such a way that limited new training data would modify the model parameters guided by the prior knowledge to compensate for the adverse effect of a mismatch [35]. The prior density prevents large deviations of the parameters unless the new training data provide strong evidence.

More specifically, we assume that an HMM is characterized by a parameter vector Φ that is a random vector, and that prior knowledge about the random vector is available and characterized by a prior probability density function $p(\Phi)$, whose parameters are to be determined experimentally.

With the observation data $\mathbf{X}$, the MAP estimate is expressed as follows:

$$\hat{\Phi} = \arg\max_{\Phi}[p(\Phi \mid \mathbf{X})] = \arg\max_{\Phi}[p(\mathbf{X} \mid \Phi)p(\Phi)] \tag{9.14}$$

If we have no prior information, $p(\Phi)$ is the uniform distribution, and the MAP estimate becomes identical to the ML estimate. We can use the EM algorithm as the ML to estimate the parameters of HMMs. The corresponding Q-function can be defined as:

$$Q_{MAP}(\Phi, \hat{\Phi}) = \log p(\hat{\Phi}) + Q(\Phi, \hat{\Phi}) \tag{9.15}$$

The EM algorithm for the ML criterion can be applied here directly. The actual expression depends on the assumptions made about the prior density. For the widely used continuous Gaussian mixture HMM, there is no joint conjugate prior density. We can assume different components of the HMM to be mutually independent, so that the optimization can be split into different subproblems involving only a single component of the parameter set. For example, the prior density function for the mixture Gaussian can be as follows:

$$p_{b_i}(\mathbf{c}_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = p_{c_i}(\mathbf{c}_i) \prod_k p_{b_{ik}}(\boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}) \tag{9.16}$$

where $p_{c_i}(\mathbf{c}_i)$ is a Dirichlet prior density for the mixing coefficient vector of all mixture components in the Markov state i, and $p_{b_{ik}}(\boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik})$ denotes the prior density for parameters

of the *k*th Gaussian component in the state *i*. The Dirichlet prior density $p_{c_i}(\mathbf{c}_i)$ is characterized by a vector $\boldsymbol{\upsilon}_i$ of positive hyperparameters such that:

$$p_{c_i}(\mathbf{c}_i) \propto \prod_k c_{ik}^{\upsilon_{ik}-1} \tag{9.17}$$

For full covariance *D*-dimensional Gaussian densities, the prior density can be a normal-Wishart density parameterized by two values $\eta > D-1, \tau > 0,$ the vector $\boldsymbol{\mu}_{nw}$, and the symmetric positive definite matrix $\mathbf{S}$ as follows:

$$\begin{aligned} &p_{b_{ik}}(\boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}) \propto \\ &\sqrt{\det(\boldsymbol{\Sigma}_{ik})^{D-\eta}} \exp(-\frac{\tau}{2}(\boldsymbol{\mu}_{ik}-\boldsymbol{\mu}_{nw})\boldsymbol{\Sigma}_{ik}^{-1}(\boldsymbol{\mu}_{ik}-\boldsymbol{\mu}_{nw})^t - \frac{1}{2}tr(\mathbf{S}\boldsymbol{\Sigma}_{ik}^{-1})) \end{aligned} \tag{9.18}$$

We can apply the same procedure as the MLE Baum-Welch reestimation algorithm. For example, with the *Q*-function defined in Eq. (9.15), we can apply the Lagrange method to derive the mixture coefficients as follows:

$$\begin{cases} \dfrac{\partial}{\partial \hat{c}_{ik}}(\log p_{c_i}(\hat{\mathbf{c}}_i) + \sum_k \sum_t \xi_t(i,k) \log \hat{c}_{ik}) + \lambda = 0, \forall k \\ \sum_k \hat{c}_{ik} = 1 \end{cases} \tag{9.19}$$

Based on Eqs. (9.17) and (9.19), the solution is:

$$\hat{c}_{ik} = \frac{\upsilon_{ik} - 1 + \sum_t \xi_t(i,k)}{\sum_l (\upsilon_{il} - 1 + \sum_t \xi_t(i,l))} \tag{9.20}$$

A comparison between Eq. (9.20) and the ML estimate Eq. (8.58) shows that the MAP estimate is a weighted average between the mode of the prior density and the ML estimate with proportions given by $\upsilon_{ijk} - 1$ and $\sum_t \xi_t(i,k)$, respectively.

We can optimize Eq. (9.15) with respect to mean and covariance parameters in a similar fashion. For example, the solution of these estimates is:

$$\hat{\boldsymbol{\mu}}_{ik} = \frac{\tau_{ik}\boldsymbol{\mu}_{nw_{ik}} + \sum_{t=1}^{T} \zeta_t(i,k)\mathbf{x}_t}{\tau_{ik} + \sum_{t=1}^{T} \zeta_t(i,k)} \tag{9.21}$$

$$\hat{\boldsymbol{\Sigma}}_{ik} = \frac{\mathbf{S}_{ik} + \tau_{ik}(\hat{\boldsymbol{\mu}}_{ik} - \boldsymbol{\mu}_{nw_{ik}})(\hat{\boldsymbol{\mu}}_{ik} - \boldsymbol{\mu}_{nw_{ik}})^t + \sum_{t=1}^{T} \zeta_t(i,k)(\mathbf{x} - \hat{\boldsymbol{\mu}}_{ik})(\mathbf{x} - \hat{\boldsymbol{\mu}}_{ik})^t}{\eta_{ik} - D + \sum_{t=1}^{T} \zeta_t(i,k)} \tag{9.22}$$

where τ_{ik} is the parameter in the normal-gamma density for the corresponding state *i*.

Thus, the reestimation formula for the Gaussian mean is a weighted sum of the prior mean with the ML mean estimate mean $\sum_{t=1}^{T} \zeta_t(i,k)\mathbf{x}_t / \sum_{t=1}^{T} \zeta_t(i,k)$. τ_{ik} is a balancing factor between prior mean and the ML mean estimate. When τ_{ik} is large, the variance of the prior knowledge is small and the value of the mean $\boldsymbol{\mu}_{nw_{ik}}$ is assumed to have high certainty, leading to the dominance of the final estimate. When the amount of adaptation data increases, the MAP estimate approaches the ML estimate, as the adaptation data overwrite any important prior that may influence the final estimate. Similarly, the covariance estimation formula has the same interpretation of the balance between the prior and new data.

One major limitation of the MAP-based approach is that it requires an accurate initial guess for the prior $p(\Phi)$, which is often difficult to obtain. We can use the already trained initial models that embody some characteristics of the original training conditions. A typical way to generate an initial Gaussian prior is to cluster the initial training data based on speaker or environment similarity measures. We can derive a set of models based on the partition, which can be seen as a set of observations drawn from a distribution having $p(\Phi)$. We can, thus, estimate the prior based on the sample moments to derive the corresponding prior parameters.

Another major limitation is that the MAP-based approach is a local approach to updating the model parameters. Namely, only model parameters that are observed in the adaptation data can be modified from the prior value. When the system has a large number of free parameters, the adaptation can be very slow. Thus in practice we need to find correlations between the model parameters, so that the unobserved or poorly adapted parameters can be altered [3, 22]. Another possibility is to impose structural information so the model parameters can be shared for improved adaptation speed [96].

The MAP training can be iterative, too, which requires an initial estimate of model parameters. A careful initialization for the Gaussian densities is also very important. Unlike the discrete distributions, there is no such a thing as a uniform density for a total lack of information about the value of the parameters. We need to use the same initialization procedure as discussed in Chapter 8.

For speaker-adaptive speech recognition, it has been experimentally found that τ_{ik} can be a fixed constant value for all the Gaussian components across all the dimensions. Thus the MAP HMM can be regarded as an interpolated model between the speaker-independent and speaker-dependent HMM. Both are derived from the standard ML forward-backward algorithm. Experimental performance of MAP training is discussed in Section 9.6.3.

9.6.2. Maximum Likelihood Linear Regression (MLLR)

When the continuous HMM is used for acoustic modeling, the most important parameter set to adapt is the output Gaussian density parameters, i.e., the mean vector and the covariance matrix. We can use a set of linear regression transformation functions to map both means and covariances in order to maximize the likelihood of the adaptation data [68]. The maximum likelihood linear regression (MLLR) mapping is consistent with the underlying criterion for building the HMM while keeping the number of free parameters under control. Since the transformation parameters can be estimated from a relatively small amount of adaptation data, it is very effective for rapid adaptation. MLLR has been widely used to obtain adapted models for either a new speaker or a new environment condition.

More specifically, in the mixture Gaussian density functions, the *k*th mean vector $\boldsymbol{\mu}_{ik}$ for each state *i* can be transformed using following equation:

$$\tilde{\boldsymbol{\mu}}_{ik} = \mathbf{A}_c \boldsymbol{\mu}_{ik} + \mathbf{b}_c \tag{9.23}$$

where $\mathbf{A}_c$ is a regression matrix and $\mathbf{b}_c$ is an additive bias vector associated with some broad class *c*, which can be either a broad phone class or a set of tied Markov states. The goal of Eq. (9.23) is to map the mean vector into a new space such that the mismatch can be eliminated. Because the amount of adaptation data is small, we need to make sure the number of broad classes *c* is small so we have only a small number of free parameters to estimate. Equation (9.23) can be simplified into:

$$\tilde{\boldsymbol{\mu}}_{ik} = \mathbf{W}_c \boldsymbol{\mu}_{ik} \tag{9.24}$$

where $\boldsymbol{\mu}_{ik}$ is extended as $[1, \boldsymbol{\mu}_{ik}^t]^t$ and $\mathbf{W}_c$ is the extended transform, $[\mathbf{b}, \mathbf{A}]$.

This mapping approach is based on the assumption that $\mathbf{W}_c$ can be tied for a wide range of broad phonetic classes so that the overall free parameters are significantly less than the number of the mean vectors. Therefore, the same transformation can be used for several distributions if they represent similar acoustic characteristics.

To estimate these transformation parameters in the MLE framework, we can use the same *Q*-function we discussed in Chapter 8. We need to optimize only

$$\sum_i \sum_{k=1}^{M} Q_{b_i}(\Phi, \hat{\mathbf{b}}_{ik}) \tag{9.25}$$

with respect to $\mathbf{W}_c$. Maximization of $Q_{b_i}(\Phi, \hat{\mathbf{b}}_{ik})$ with respect to $\mathbf{W}_c$ can be achieved by computing the partial derivatives. For the Gaussian mixture density function, the partial derivative with respect to $\mathbf{W}_c$ is:

$$\frac{\partial \hat{b}_{ik}(\mathbf{x})}{\partial \mathbf{W}_c} = N(\mathbf{x}, \tilde{\boldsymbol{\mu}}_{ik}, \hat{\boldsymbol{\Sigma}}_{ik}) \boldsymbol{\Sigma}_{ik}^{-1} (\mathbf{x} - \mathbf{W}_c \boldsymbol{\mu}_{ik}) \boldsymbol{\mu}_{ik}^t \tag{9.26}$$

Let us denote the set of Gaussian components forming the broad transformation classes as *C*; we use $b_{ik} \in C$ to denote that the *k*th Gaussian density in state *i* belongs to the class. We can expand the *Q*-function with the partial derivatives and set it to zero, leading to the following equation:

$$\sum_{t=1}^{T} \sum_{b_{ik} \in C} \zeta_t(i,k) \boldsymbol{\Sigma}_{ik}^{-1} \mathbf{x}_t \boldsymbol{\mu}_{ik}^t = \sum_{t=1}^{T} \sum_{b_{ik} \in C} \zeta_t(i,k) \boldsymbol{\Sigma}_{ik}^{-1} \mathbf{W}_c \boldsymbol{\mu}_{ik} \boldsymbol{\mu}_{ik}^t \tag{9.27}$$

We can rewrite Eq. (9.27) as:

$$\mathbf{Z} = \sum_{b_{ik} \in C} \mathbf{V}_{ik} \mathbf{W}_c \mathbf{D}_{ik} \tag{9.28}$$

where

$$\mathbf{Z} = \sum_{t=1}^{T} \sum_{b_{ik} \in C} \zeta_t(i,k) \boldsymbol{\Sigma}_{ik}^{-1} \mathbf{x}_t \boldsymbol{\mu}_{ik}^t , \tag{9.29}$$

$$\mathbf{V}_{ik} = \sum_{t=1}^{T} \zeta_t(i,k) \boldsymbol{\Sigma}_{ik}^{-1} , \tag{9.30}$$

and

$$\mathbf{D}_{ik} = \boldsymbol{\mu}_{ik} \boldsymbol{\mu}_{ik}^t . \tag{9.31}$$

Estimating $\mathbf{W}_c$ for Eq. (9.28) is computationally expensive, as it requires solving simultaneous equations. Nevertheless, if we assume that the covariance matrix is diagonal, we can have a closed-form solution that is computationally efficient. Thus, we can define

$$\mathbf{G}_q = \sum_{b_{ijk} \in C} v_{qq} \mathbf{D}_{ik} \tag{9.32}$$

where v_{qq} denotes the *q*th diagonal element of matrix $\mathbf{V}_{ik}$. The transformation matrix can be computed row by row. So for the *q*th row of the transformation matrix $\mathbf{W}_q$, we can derive it from the *q*th row of $\mathbf{Z}_q$ [defined in Eq. (9.29)] as follows:

$$\mathbf{W}_q = \mathbf{Z}_q \mathbf{G}_q^{-1} \tag{9.33}$$

Since $\mathbf{G}_q$ may be a singular matrix, we need to make sure we have enough training data for the broad class. Thus, if the amount of training data is limited, we must tie a number of transformation classes together.

We can run several iterations to maximize the likelihood for the given adaptation data. At each iteration, transformation matrices can be initialized to identity transformations. We can iteratively repeat the process to update the means until convergence is achieved. We can

also incrementally adapt the mean vectors after each observation sequence or set of observation sequences while the required statistics are accumulated over time. Under the assumption that the alignment of each observation sequence against the model is reasonably accurate, we can accumulate these estimated counts over time and use them incrementally. In order to deal with the tradeoff between specificity and robust estimation, we can dynamically generate regression classes according to the senone tree. Thus, we can incrementally increase the number of regression classes when more and more data become available.

MLLR adaptation can be generalized to include the variances with the ML framework, although the additional gain after mean transformation is often less significant (less than relative 2% error reduction). When the user donates about 15 sentences for enrollment training, Table 9.5 illustrates how the MLLR adaptation technique can be used to further reduce the word recognition error rate for a typical dictation application. Here, there is only one context-independent phonetic class for all the context-dependent Gaussian densities. As we can see, most of the error reduction came from adapting the mean vector.

We can further extend MLLR to *speaker-adaptive training* (SAT) [6, 74]. In conventional speaker-independent training, we simply use data from different speakers to build a speaker-independent model. An inherent difficulty in this approach is that spectral variations of different speakers give the speaker-independent acoustic model higher variance than the corresponding speaker-dependent model. We can include MLLR transformation in the process of training to derive the MLLR parameters for each individual speaker. Thus the training data are transformed to maximize the likelihood for the overall speaker-independent model. This process can be run iteratively to reduce mismatches of different speakers. By explicitly accounting for the interspeaker variations during training and decoding, SAT reduces the error rate by an additional 5–10%.

Table 9.5 Relative error reductions with MLLR methods.

Models	Relative Error Reduction
CHMM	Baseline
MLLR on mean only	12%
MLLR on mean and variance	+2%
MLLR SAT	+8%

9.6.3. MLLR and MAP Comparison

The MLLR method can be combined with MAP. This guarantees that with the increased amount of training data, we can have, not only a set of compact MLLR transformation functions for rapid adaptation, but also directly modified model parameters that converge with ML estimates. We can use MAP to adapt the model parameters and then add MLLR to transform these adapted models. It is also possible to incorporate the MAP principle directly into MLLR [18, 19].

As an example, the result of a 60,000-word dictation application using various adaptation methods is shown in Figure 9.11.[4] The speaker-dependent model used 1000 utterances. Also included as a reference is the speaker-independent result, which is used as the starting point for adaptive training. When the speaker-independent model is adapted with about 200 utterances, the speaker-adaptive model has already outperformed both speaker-independent and speaker-dependent systems. The results clearly demonstrate that we have insufficient training data for speaker-dependent speech recognition, as MAP-based outperform ML-based models. This also illustrates that we can make effective use of speaker-independent data for speaker-dependent speech recognition. Also, notice that the MLLR method has a faster adaptation rate than the MAP method. The MLLR method has context-independent phonetic classes. So, when the amount of adaptation data is limited, the MLLR method offers better overall performance.

However, the MAP becomes more accurate when the amount of adaptation data increases to 600 per speaker. This is because we can modify all the model parameters with the MAP training, and the MLLR transformation can never have the same degree of freedom as the MAP method. When the MLLR is combined with MAP, we can have not only rapid adaptation but also superior performance over either the MLLR or MAP method across a wide range of adaptation data points. There are a number of different ways to combine both MLLR and MAP for improved performance [4, 98].

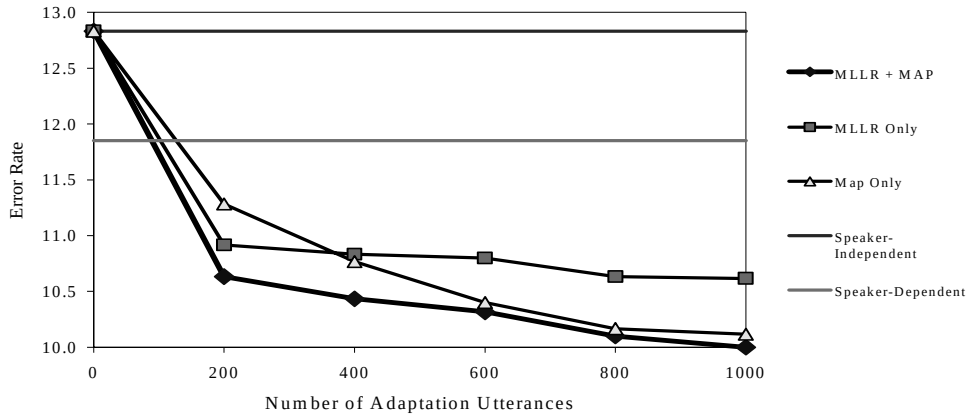


Figure 9.11 Comparison of Whisper with MLLR, MAP, and combined MLLR and MAP. The error rate is shown for a different amount of adaptation data. The speaker-dependent and -independent models are also included. The speaker-dependent model was trained with 1000 sentences.

[4] In practice, if a large, well-trained, speaker-independent model is used, the baseline performance may be very good and hence the relative error reduction from speaker adaptation may be smaller than for smaller and simpler models.

9.6.4. Clustered Models

Both MAP and MLLR techniques are based on using an appropriate initial model for adaptive modeling. How accurate we make the initial model directly affects the overall performance. An effective way to minimize the mismatch is, thus, to cluster similar speakers and environments in the training data, building a set of models for each cluster that has minimal mismatch for different conditions. When we have enough training data, and enough coverage for a wide range of conditions, this approach ensures a significantly improved robustness.

For example, we often need a set of clustered models for different telephone channels, including different cellular phone standards. We also need to build gender-dependent models or speaker-clustered models for improved performance. In fact, when we construct speaker-clustered models, we can apply MMLR transformations or neural networks to minimize speaker variations such that different speakers can be mapped to the same golden speaker that is the representative of the cluster.

Speaker clusters can be created based on the information of each speaker-dependent HMM. The clustering procedure is similar to the decision-tree procedure discussed in Section 9.4.3. Using clustered models increases the amount of computational complexity. It also fragments the training data. Clustering is often needed to combine other smoothing techniques, such as deleted interpolation or MLLR transformation, in order to create clustered models from the pooled model. We can also represent a speaker as a weighted sum of individual speaker cluster models with the cluster adaptive training [33] or eigenvoice techniques [64].

When we select an appropriate model, we can compute the likelihood of the test speech against all the models and select the model that has the highest likelihood. Alternatively, we can compute likelihoods as part of the decoding process and prune away less promising models dynamically without significantly increasing the computational load. When multiple models are plausible, we can compute the weighted sum of the clustered models with pretrained mixing coefficients for different clusters, much as we train the deleted interpolation weights.

Traditionally speaker clustering is performed across different speakers without considering phonetic similarities across different speakers. In fact, clustered speaker groups may have very different degrees of variations for different phonetic classes. You can further generalize speaker clustering to the subword or subphonetic level [62]. With multiple instances derived from clustering for each subword or subphonetic unit, you can model speaker variation explicitly across different subword or subphonetic models.

In practice, gender-dependent models can reduce the word recognition error by 10%. More refined speaker-clustered models can further reduce the error rate, but not as much as the gain from gender-dependent models, unless we have a large number of clustered speakers. If the new user happens to be similar to one of these speaker clusters, we can approach speaker-dependent speech recognition without enrollment. For environment-dependent models, clustering is more critical. The challenge is to anticipate the kind of environment or channel distortions the system will have to deal with. Since this is often unpredictable, we

need to use adaptive techniques such as MAP and MLLR to minimize the mismatch. We discuss this in more detail in Chapter 10.

9.7. CONFIDENCE MEASURES: MEASURING THE RELIABILITY

One of the most critical components in a practical speech recognition system is a reliable *confidence measure*. With an accurate confidence measure for each recognized word, the conversational back end can repair potential speech recognition errors, can reject out-of-vocabulary words, and can identify key words (perform word spotting) that are relevant to the back end. In a speaker-dependent or speaker-adaptive system, the confidence measure can help user enrollment (to eliminate mispronounced words). It is also critical for unsupervised speaker adaptation, allowing selective use of recognition results so that transcriptions with lower confidence can be discarded for adaptation.

In theory, an accurate estimate of $P(\mathbf{W} \mid \mathbf{X})$, the posterior probability, is itself a good confidence measure for word $\mathbf{W}$ given the acoustic input $\mathbf{X}$. Most practical speech recognition systems simply ignore $P(\mathbf{X})$, as it is a constant in evaluating $P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W}) / P(\mathbf{X})$ across different words. $P(\mathbf{W} \mid \mathbf{X})$ can be expressed:

$$P(\mathbf{W} \mid \mathbf{X}) = \frac{P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})}{P(\mathbf{X})} = \frac{P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})}{\sum_{\mathbf{w}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})} \tag{9.34}$$

Equation (9.34) essentially provides a solid framework for measuring confidence levels. It is the ratio between the score for the word hypothesis $P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$ and the acoustic probability $\sum_{\mathbf{w}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$. In the sections that follow we discuss a number of ways to model and use such a ratio in practical systems.

9.7.1. Filler Models

You can compute $P(\mathbf{X})$ in Eq. (9.34) with a general-purpose recognizer. It should be able to recognize anything such that it can *fill the holes* of the grammar in the normal speech recognizer. The filler model has various forms [7, 63]. One of the most widely used is the so-called all-phone network, in which all the possible phonetic and nonspeech HMMs are connected to each other, and with which any word sequence can be recognized.

In addition to evaluating $P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$ as needed in normal speech recognition, a separate decoding process is used to evaluate $\sum_{\mathbf{w}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$. Here $\mathbf{W}$ is either a phonetic or a word model. You can also apply phonetic *n*-gram probabilities that are derived from a lexicon targeted for possible new words. The best path from the all-phone network is compared with the best path from the normal decoder. The ratio between the two, as ex-

pressed in Eq. (9.34), is used to measure the confidence for either word or phone. In the decoding process (see Chapters 12 and 13), you can accumulate the phonetic ratio derived from Eq. (9.34) on a specific word. If the accumulative $P(\mathbf{W} \mid \mathbf{X})$ for the word is less than a predetermined threshold, the word is rejected as either a new word or a nonspeech event.

Both context-independent and context-dependent phonetic models can be used for the fully connected network. When context-dependent phonetic models are used, you need to make sure that only correct contextual phonetic connections are made. Although context-dependent models offer significant improvement for speech recognition, the filler phonetic network seems to be insensitive to context-dependency in empirical experiments.

There are *word-spotting* applications that need to spot just a small number of key words. You can use the filler models described here for word spotting. You can also build antiword models trained with all the data that are not associated with the key words of interest. Empirical experiments indicate that large-vocabulary speech recognition is the most suitable choice for word spotting. You can use a general-purpose *n*-gram to generate recognition results and identify needed key words from the word lattice. This is because a large-vocabulary system provides a better estimate of $\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$ with a more accurate language model probability. In practice, we don't need to use all the hypotheses to compute $\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$. Instead, *n*-best lists and scores [40] can be used to provide an effective estimate of $\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$.

9.7.2. Transformation Models

To determine the confidence level for each word, subword confidence information is often helpful. Different phones have different impact on our perception of words. The weight for each subword confidence score can be optimized from the real training data. If a word *w* has *N* phones, we can compute the confidence score of the word as follows:

$$CS(w) = \sum_{i=1}^{N} \wp_i(x_i) / N \tag{9.35}$$

where *CS(w)* is the confidence score for word *w*, x_i is the confidence score for subword unit *i* in word *w*, and $\wp_i$ is the mapping function that may be tied across a number of subword units. The transformation function can be defined as:

$$\wp_i(x) = ax + b \tag{9.36}$$

We can use discriminative training to optimize the parameters *a* and *b*, respectively. A cost function can be defined as a sigmoid function of *CS*(*w*). As shown in Figure 9.12, the optimal transformation parameters vary substantially across different phones. The weight for consonants is also typically larger than that of vowels.

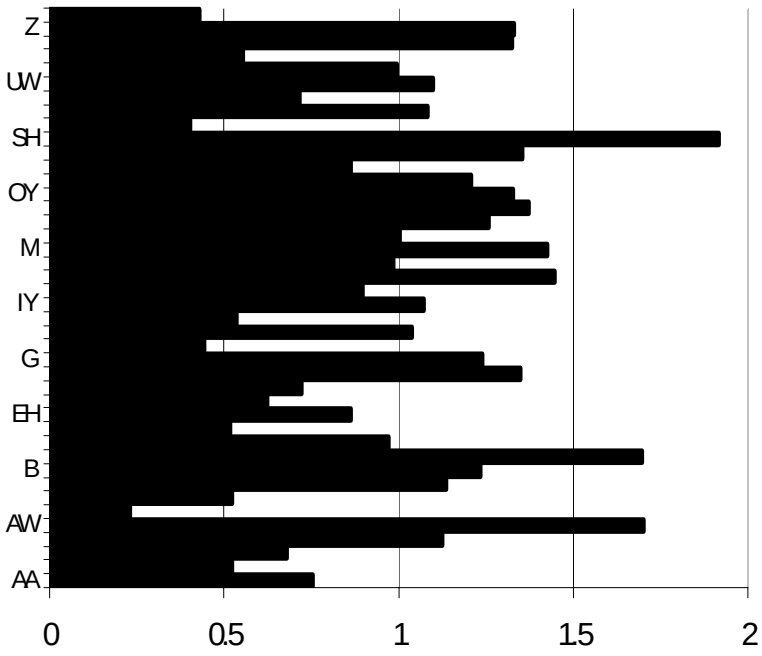


Figure 9.12 Transformation parameter *a* for each context-independent phone class. The weight of consonants is typically larger than that of vowels [63].

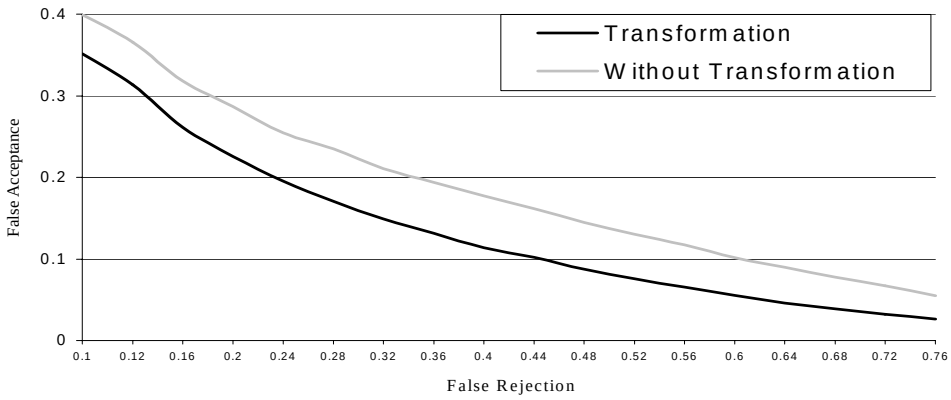


Figure 9.13 The ROC curve of phonetic filler models with and without optimal feature transformation [63].

The transformation function can be context dependent. Figure 9.13 illustrates the ROC curve of the context-dependent transformation model in comparison with the corresponding phonetic filler model. The filler model essentially has a uniform weight across all the phones in a given word. The estimated transformation model has 15–40% false-acceptance error reduction at various fixed false-rejection rates. The false-acceptance rate of the transformation model is consistently lower than that of the filler model [63].

9.7.3. Combination Models

In practical systems, there are a number of features you can use to improve the performance of confidence measures. For example, the following features are helpful:

- Word stability ratio from different language model weights (*WrdStabRatio*). This is obtained by applying different language weights to see how stably each word shows up in the recognition *n*-best list.
- Logarithm of the average number of active words around the ending frame of the word (*WrdCntEnd*).
- Acoustic score per frame within the word normalized by the corresponding active senone scores (*AscoreSen*).
- Logarithm of the average number of active words within the word (*WrdCntW*).
- Acoustic score per frame within the word normalized by the phonetic filler model (*AscoreFiller*).
- Language model score (*LMScore*).
- Language model back-off (trigram, bigram, or unigram hit) for the word (*LMBackOff*).
- Logarithm of the average number of active states within the word (*StateCnt*).
- Number of phones in the word (*Nphones*).
- Logarithm of the average number of active words around the beginning frame of the word (*WrdCntBeg*).
- Whether the word has multiple pronunciations (*Mpron*).
- Word duration (*WordDur*).

To clarify each feature's relative importance, Table 9.6 shows its linear correlation coefficient against the correct/incorrect tag for each word in the training set. Word stability ratio (*WrdStabRatio*) has the largest correlation value.

Several kinds of classifiers can be used to compute the confidence scores. Previous research has shown that the difference between classifiers, such as linear classifiers, generalized linear models, decision trees, and neural networks, is insignificant. The simplest linear classifier based on the discriminative training performs well in practice. As some features are highly correlated, you can iteratively remove features to combat the curse of dimensionality.

The combination model can have up to 40–80% false-acceptance error reduction at fixed false-rejection rate in comparison to the single-feature approach.

Table 9.6 Correlation coefficients of several features against correct/incorrect tag.

Feature	Correlation
WrdStabRatio	0.590
WrdCntW	–0.223
LMBackOff	0.171
AscoreSen	0.250
LMScore	0.175
Nphones	0.091
WordDur	0.012
WrdCntEnd	–0.321
AscoreFiller	0.219
StateCnt	–0.155
Mpron	0.057
WrdCntBeg	–0.067

9.8. OTHER TECHNIQUES

In addition to HMMs, a number of interesting alternative techniques are being actively investigated by researchers. We briefly review two promising methods here.

9.8.1. Neural Networks

You have seen both single-layer and multilayer neural nets in Chapter 4 for dealing with static patterns. In dealing with nonstationary signals, you need to address how to map an input sequence properly to an output sequence when two sequences are not synchronous, which should include proper alignment, segmentation, and classification. The basic neural networks are not well equipped to address these problems in a unified way.

Recurrent neural networks have an internal state that is a function of the current input and the previous internal state. A number of them use time-step delayed recurrent loops on the hidden or output units of a feedforward network, as discussed in earlier chapters. For sequences of finite numbers of delays, we can transform these networks into equivalent feedforward networks by unfolding them over the time period. They can be trained with the standard back propagation procedure, with the following modifications:

- The desired outputs are functions of time, and error functions have to be computed for every copy of the output layer. This requires the selection of an appropriate time-dependent target function, which is often difficult to define.

- All copies of the unfolded weights are constrained to be identical during the training. We can compute the correction terms separately for each weight and use the average to update the final estimate.

In most of these networks, you can have a partially recurrent network that has feedback of the hidden and output units to the input layer. For example, the feedforward network can be used in a set of local feedforward connections with one time-step delay. These networks are usually implemented by extending the input field with additional feedback units containing both the hidden and output values generated by the preceding input. You can encode the past nonstationary information that is often required to generate the correct output, given the current input, as illustrated in Figure 9.14.

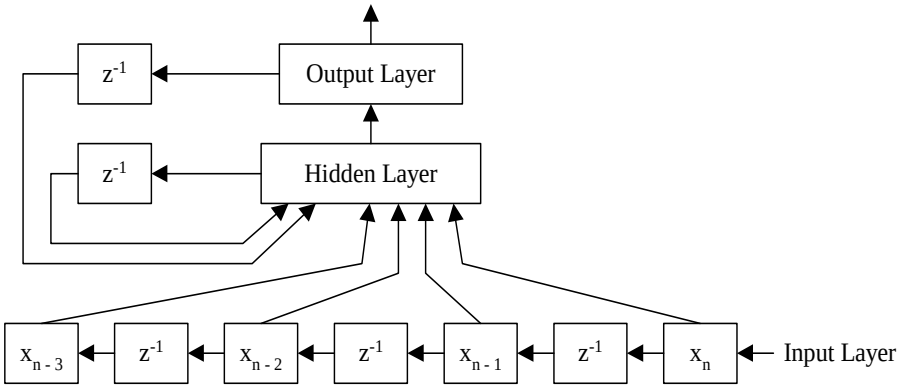


Figure 9.14 A recurrent network with contextual inputs, hidden vector feedback, and output vector feedback.

One of the popular neural networks is the *Time Delay Neural Network* (TDNN) [105]. Like static networks, the TDNN can be trained to recognize a sequence of predefined length (defined by the width of the input window). The activation in the hidden layer is computed from the current and multiple time-delayed values of the preceding layer, and the output units are activated only when a complete speech segment has been processed. A typical TDNN is illustrated in Figure 9.15. The TDNN has been successfully used to classify presegmented phonemes.

All neural networks have been shown to yield good performance for small-vocabulary speech recognition. Sometimes they are better than HMMs for short, isolated speech units. By recurrence and the use of temporal memory, they can perform some kind of integration over time. It remains a challenge for neural networks to demonstrate that they can be as effective as HMMs for dealing with nonstationary signals, as is often required for large-vocabulary speech recognition.

To deal with the continuous speech, the most effective solution is to integrate neural nets with HMMs [91, 113]. The neural network can be used as the output probabilities to replace the Gaussian mixture densities. Comparable results can be obtained with the inte-

grated approach. These HMM output probabilities could be estimated by applying the Bayes rule to the output of neural networks that have been trained to classify HMM state categories. The neural networks can consist either of a single large trained network or of a group of separately trained small networks [21, 31, 75, 90].

A number of techniques have been developed to improve the performance of training these networks. Training can be embedded in an EM-style process. For example, dynamic programming can be used to segment the training data. The segmented data are then used to retrain the network. It is also possible to have Baum-Welch style training [14, 42].

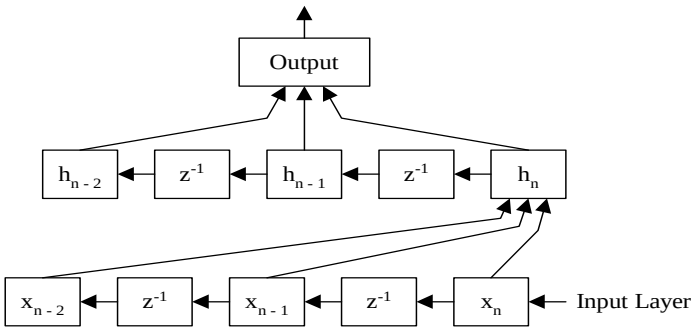


Figure 9.15 A time-delay neural network (TDNN), where the box h_t denotes the hidden vector at time *t*, the box x_t denotes the input vector at time *t*, and the box z^{-1} denotes a delay of one sample.

9.8.2. Segment Models

As discussed in Chapter 8, the HMM *output-independence assumption* results in a piecewise stationary process within an HMM state. Although the nonstationary speech may be modeled sufficiently with a large number of states, the states in which salient speech features are present are far from stationary [25, 99]. While the use of time-derivative features (e.g., delta and/or delta-delta features) alleviates these limitations, the use of such longer-time-span features may invalidate the conditional independence assumption.

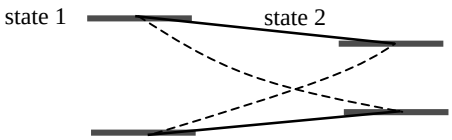


Figure 9.16 Diagram illustrating that HMM's output observation can hop between two unexpected quasi-stationary states [46].

The use of Gaussian mixtures for continous or semicontinuous HMMs, as described in Chapter 8, could introduce another potential problem, where arbitrary transitions among the Gaussian mixture components between adjacent HMM states are allowed [59]. Figure 9.16 illustrates two HMM states with two mixture components. The solid lines denote the valid trajectories actually observed in the training data. However, in modeling these two trajectories, the Gaussian mixtures inadvertently allow two *phantom* trajectories, shown in dashed lines in Figure 9.16, because no constraint is imposed on the mixture transitions across the state. It is possible that such phantom trajectories degrade recognition performance, because the models can overrepresent speech signals that should be modeled by other acoustic units. Segment models can alleviate such HMM modeling deficiencies [77, 79].

In the standard HMM, the output probability distribution is modeled by a quasi-stationary process, i.e.,

$$P(\mathbf{x}_1^L \mid s) = \prod_{i=1}^{L} b_s(\mathbf{x}_i) \tag{9.37}$$

For the segment model (SM), the output observation distribution is modeled by two stochastic processes:

$$P(\mathbf{x}_1^L \mid s) = P(\mathbf{x}_1^L \mid s, L)P(L \mid s) \tag{9.38}$$

The first term of Eq. (9.38) is no longer decomposable in the absence of the output-independence assumption. The second term is similar to the duration model described in Chapter 8. In contrast to the HMM whose quasi-stationary process for each state s generates one frame $\mathbf{x}_i$, a state in a segment model can generate a variable-length observation sequence $\{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_L\}$ with random length L.

Since the likelihood evaluation of segment models cannot be decomposed, the computation of the evaluation is not shareable between different segments (even for the case where two segments differ only by one frame). This results in a significant increase in computation for both training and decoding [77]. In general, the search state space is increased by a factor of $L_{\max}$, the maximum segment duration. If segment models are used for phone segments, $L_{\max}$ could be as large as 60. On top of this large increase in search state space, the evaluation of segment models is usually an order of magnitude more expensive than for HMM, since the evaluation involves several frames. Thus, the segment model is often implemented in a multipass search framework, as described in Chapter 13.

Segment models have produced encouraging performance for small-vocabulary or isolated recognition tasks [25, 44, 79]. However, their effectiveness on large-vocabulary continuous speech recognition remains an open issue because of necessary compromises to reduce the complexity of implementation.

9.8.2.1. Parametric Trajectory Models

Parametric trajectory models [25, 37] were first proposed to model a speech segment with curve-fitting parameters. They approximate the *D*-dimensional acoustic observation vector $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ by a polynomial function. Specifically, the observation vector $\mathbf{x}_t$ can be represented as

$$\mathbf{x}_t = \mathbf{C} \times F_t + \mathbf{e}_t(\Sigma) = \begin{pmatrix} c_1^0 & c_1^1 & \cdots & c_1^N \\ c_2^0 & c_2^1 & \cdots & c_2^N \\ \vdots & \vdots & \vdots & \vdots \\ c_D^0 & c_D^1 & c_D^2 & c_D^N \end{pmatrix} \begin{pmatrix} f_0(t) \\ f_1(t) \\ \vdots \\ f_N(t) \end{pmatrix} + \mathbf{e}_t(\Sigma) \tag{9.39}$$

where the matrix $\mathbf{C}$ is the trajectory parameter matrix, F_t is the family of *N*th-order polynomial functions, and $\mathbf{e}_t(\Sigma)$ is the residual fitting error. Equation (9.39) can be regarded as modeling the time-varing mean in the output distribution for an HMM state. To simplify computation, the distribution of the residual error is often assumed to be an independent and identically distributed random process with a normal distribution $N(0,\Sigma)$. To accommodate diverse durations for the same segment, the relative linear time sampling of the fixed trajectory is assumed [37].

Each segment *M* is characterized by a trajectory parameter matrix $\mathbf{C}_m$ and covariance matrix Σ_m. The likelihood for each frame can be specified [46] as

$$P(\mathbf{x}_t \mid \mathbf{C}_m, \Sigma_m) = \frac{\exp\left\{-tr\left[(\mathbf{x}_t - \mathbf{C}_m F_t)\Sigma_m^{-1}(\mathbf{x}_t - \mathbf{C}_m F_t)^t\right]/2\right\}}{(2\pi)^{D/2} \mid \Sigma_m \mid^{1/2}} \tag{9.40}$$

If we let $\mathbf{F}_T = (F_0, F_1, \cdots, F_{T-1})^t$, then the likelihood for the whole acoustic observation vector can be expressed as

$$P(\mathbf{X} \mid \mathbf{C}_m, \Sigma_m) = \frac{\exp\left\{-tr\left[(\mathbf{X} - \mathbf{C}_m \mathbf{F}_T)\Sigma_m^{-1}(\mathbf{X} - \mathbf{C}_m \mathbf{F}_T)^t\right]/2\right\}}{(2\pi)^{DT/2} \mid \Sigma_m \mid^{T/2}} \tag{9.41}$$

Multiple mixtures can also be applied to SM. Suppose segment *M* is modeled by *K* trajectory mixtures. The likelihood for the acoustic observation vector $\mathbf{X}$ becomes

$$\sum_{k=1}^{K} w_k \, p_k(\mathbf{X} \mid \mathbf{C}_k, \Sigma_k) \tag{9.42}$$

Hon et al. [47] showed that only a handful of target trajectories are needed for speaker-independent recognition, in contrast to the many mixtures required for continuous Gaussian HMMs. This should support the phantom-trajectory argument involved in Figure 9.16.

The estimation of segment parameters can be accomplished by the EM algorithm described in Chapter 4. Assume a sample of L observation segments $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_L$, with corresponding duration $T_1, T_2, \cdots, T_L$, are generated by the segment model M. The MLE formulae using the EM algorithm are:

$$\gamma_k^i = \frac{w_k p_k(\mathbf{X}_i \mid \mathbf{C}_k, \Sigma_k)}{P(\mathbf{X}_i \mid \Phi_m)} \tag{9.43}$$

$$\hat{w}_k = \frac{1}{L}\sum_{i=1}^{L} \frac{w_k p_k(\mathbf{X}_i \mid \mathbf{C}_k, \Sigma_k)}{P(\mathbf{X}_i \mid \Phi_{\mathrm{m}})} \tag{9.44}$$

$$\hat{\mathbf{C}}_k = \left[\sum_{i=1}^{L} \gamma_k^i \mathbf{X}_i \mathbf{F}_{T_i}^t\right] / \left[\sum_{i=1}^{L} \gamma_k^i \mathbf{F}_{T_i} \mathbf{F}_{T_i}^t\right] \tag{9.45}$$

$$\hat{\Sigma}_k = \sum_{i=1}^{L} \gamma_k^i (\mathbf{X}_i - \mathbf{C}_k \mathbf{F}_{T_i})(\mathbf{X}_i - \mathbf{C}_k \mathbf{F}_{T_i})^t \Big/ \sum_{i=1}^{L} \gamma_k^i T_i \tag{9.46}$$

Parametric trajectory models have been successfully applied to phone classification [25, 46] and word spotting [37], which offers a modestly improved performance over HMMs.

9.8.2.2. Unified Frame- and Segment-Based Models

The strengths of the HMM and the segment-model approaches are complementary. HMMs are very effective in modeling the subtle details of speech signals by using one state for each quasi-stationary region. However, the transitions between quasi-stationary regions are largely neglected by HMMs because of their short durations. In contrast, segment models are powerful in modeling the transitions and longer-range speech dynamics, but might need to give up the detailed modeling to assure trainability and tractability. It is possible to have a unified framework to combine both methods [47].

In the unified complementary framework, the acoustic model $p(\mathbf{X}_1^T \mid \mathbf{W})$, can be considered as two joint hidden processes, as in the following equation:

$$\begin{aligned} p(\mathbf{X} \mid \mathbf{W}) &= \sum_{\mathbf{q}^h} \sum_{\mathbf{q}^s} p(\mathbf{X}, \mathbf{q}^h, \mathbf{q}^s \mid \mathbf{W}) \\ &= \sum_{\mathbf{q}^h} \sum_{\mathbf{q}^s} p(\mathbf{X} \mid \mathbf{q}^h, \mathbf{q}^s) p(\mathbf{q}^s \mid \mathbf{q}^h) p(\mathbf{q}^h \mid \mathbf{W}) \end{aligned} \tag{9.47}$$

where $\mathbf{q}^h$ represents the hidden process of the HMM and $\mathbf{q}^s$, the segment model. The conditional probability of the acoustic signal $p(\mathbf{X} \mid \mathbf{q}^s, \mathbf{q}^h)$ can be further decomposed into two separate terms:

$$p(\mathbf{X} \mid \mathbf{q}^s, \mathbf{q}^h) = p(\mathbf{X} \mid \mathbf{q}^h)\, p(\mathbf{X} \mid \mathbf{q}^s)^a \tag{9.48}$$

where a is a constant that is called *segment-model weight*. The first term is the contribution from normal frame-based HMM evaluation. We further assume for the second term that recognition of segment units can be performed by detecting and decoding a sequence of salient events in the acoustic stream that are statistically independent. In other words,

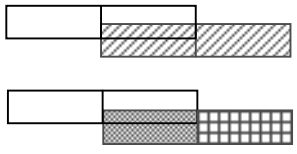
$$p(\mathbf{X} \mid \mathbf{q}^s) = \prod_i p(\mathbf{X}_i \mid q_i^s) \tag{9.49}$$

where $\mathbf{X}_i$ denotes the ith segment.

We assume that the phone sequence and the phone boundaries hypothesized by HMMs and segment models agree with each other. Based on the independent-segment assumption, this leads to a segment duration model as

$$p(\mathbf{q}^s \mid \mathbf{q}^h) = \prod_i p(t_i, t_{i+1} - 1 \mid \mathbf{X}_i) \tag{9.50}$$

By treating the combination as a hidden-data problem, we can apply the decoding and iterative EM reestimation techniques here. This unified framework enables both frame- and segment-based models to *jointly* contribute to optimal segmentations, which leads to more efficient pruning during the search. The inclusion of the segment models does not require massive revisions in the decoder, because the segment model scores can be handled in the same manner as the language model scores; whereas the segment evaluation is performed at each segment boundary.



(a) phone-pair segment models

(b) two phone (monophone or gen. triphone) segment models

Figure 9.17 Overlapped evaluation using (a) a phone-pair segment model, or (b) back-off to two phone units when the phone-pair (e_{i-1}, e_i) segment model does not exist [47].

Since subphonetic units are often used in HMMs to model the detailed quasi-stationary speech region, the segment units should be used to model long-range transition. As studies have shown that phone transitions play an essential role in humans' perception, the phone-pair segment unit that spans over two adjacent phones can be used [47]. Let e_i and t_i denote the phone and the starting time of the ith segment, respectively. For a phone-pair (e_{i-1}, e_i) segment between t_i and t_{i+1}, the segment likelihood can be computed as follows:

$$p(\mathbf{X}_i \mid q_i^s) = p(\mathbf{x}_{t_{i-1}}^{t_{i+1}} \mid e_{i-1}, e_i) \tag{9.51}$$

Rather than applying segment evaluation for every two phones, an *overlapped evaluation* scheme can be used, as shown in Figure 9.17 (a), where a phone-pair segment model

evaluation is applied at each phone boundary. The overlapped evaluation implies that each phone is evaluated twice in the overall score. Most importantly, the overlapped evaluation places constraints on overlapped regions to assure consistent trajectory transitions. This is an important feature, as trajectory mixtures prohibit *phantom* trajectories within a segment unit, but there is still no mechanism to prevent arbitrary trajectory transitions between adjacent segment units.

Some phone-pairs might not have sufficient training data. Units containing silence might also have obscure trajectories due to the arbitrary duration of silence. As a result, a phone-pair unit ($e_{i\text{-}1}$, e_i) can be backed off with two phone units as shown in Figure 9.17 (b). The phone units can be context independent or context dependent [46]. Thus, the back-off segment-model evaluation becomes:

$$p(\mathbf{X}_i \mid q_i^s) = \beta * p(\mathbf{x}_{t_{i-1}}^{t_i} \mid e_{i-1})\, p(\mathbf{x}_{t_i}^{t_{i+1}} \mid e_i) \tag{9.52}$$

where β is the back-off weight, generally smaller than 1.0. The use of back-off weight has the effect of giving more preference to phone-pair segment models than to two-phone-based back-off segment models.

The phone-pair segment model outperformed the phone-pair HMM by more than 20% in a phone-pair classification experiment [46]. The unified framework achieved about 8% word-error-rate reduction on the WSJ dictation task in comparison to the HMM-based Whisper [47].

9.9. CASE STUDY: WHISPER

Microsoft's Whisper engine offers general-purpose speaker-independent continuous speech recognition [49]. Whisper can be used for command and control, dictation, and conversational applications. Whisper offers many features such as continuous speech recognition, speaker-independence with adaptation, and dynamic vocabulary. Whisper has a unified architecture that can be scaled to meet different application and platform requirements**.**

The Whisper system uses MFCC representations (see Chapter 6) and both first- and second-order delta MFCC coefficients. Two-mean cepstral normalization discussed in Chapter 10 is used to eliminate channel distortion for improved robustness.

The HMM topology is a three-state left-to-right model for each phone. Senone models discussed in Section 9.4.3 are derived from both inter- and intraword context-dependent phones. The generic shared density function architecture can support either semicontinuous or continuous density hidden Markov models.

The SCHMM has a multiple-feature front end. Independent codebooks are built for the MFCC, first-order delta MFCC, second-order delta MFCC, and power and first and second power, respectively. Deleted interpolation is used to interpolate output distributions of context-dependent and context-independent senones. All codebook means and covariance matrices are reestimated together with the output distributions except the power covariance matrices, which are fixed.

The overall senone models can reduce the error rate significantly in comparison to the triphone or clustered triphone model. The shared Markov state also makes it possible to use continuous-density HMMs efficiently for large-vocabulary speech recognition. When a sufficient amount of training data becomes available, the best performance is obtained with the continuous-density mixture HMM. Each senone has 20 mixtures, albeit such an error reduction came at the cost of significantly increased computational complexity.

We can further generalize sharing to the level of each individual Gaussian probability density function. Each Gaussian function is treated as the basic unit to be shared across any Markov state. At this extreme, there is no need to use senones or shared states any more, and the shared probability density functions become the acoustic kernels that can be used to form any mixture function for any Markov state with appropriate mixture weights. Parameter sharing is, thus, advanced from a phone unit to a Markov state unit (senones) to a density component unit.

Regarding lexicon modeling, most words have one pronunciation in the lexicon. For words that are not in the dictionary, the LTS conversion is based on the decision tree that is trained form the existing lexicon. For the purpose of efficiency, the dictionary is used to store the most frequently used words. The LTS is only used for new words that need to be added on the fly.

For speaker adaptation, the diagonal variances and means are adapted using the MAP method. Whisper also uses MLLR to modify the mean vectors only. The MLLR classes are phone dependent. The transition probabilities are context independent and they are not modified during the adaptation stage.

The language model used in Whisper can be either the trigram or the context-free grammar. The difference is largely related to the decoder algorithm, as discussed in Chapter 13. The trigram lexicon has 60,000 most-frequent words extracted from a large text corpus. Word selection is based on both the frequency and the word's part-of-speech information. For example, verbs and the inflected forms have a higher weight than proper nouns in the selection process.

Whisper's overall word recognition error rate for speaker-independent continuous speech recognition is about 7% for the standard DARPA business-news dictation test set. For isolated dictation with similar materials, the error rate is less than 3%. If speaker-dependent data are available, we can further reduce the error rate by 15–30%, with less than 30 minutes speech from each person. The performance can be obtained real-time on today's PC systems.

9.10. HISTORICAL PERSPECTIVE AND FURTHER READING

The first machine to recognize speech was a commercial toy named Radio Rex manufactured in the 1920s. Fueled by increased computing resources, acoustic-phonetic modeling has progressed significantly since then. Relative word error rates have been reduced by 10% every year, as illustrated in Figure 9.18, thanks to the use of HMMs, the availability of large speech and text corpora, the establishment of standards for performance evaluation, and advances in computer technology. Before the HMM was established as the standard, there were

many competing techniques, which can be traced back to the 1950s. Gold and Morgan's book provides an excellent historical perspective [38].

The HMM is powerful in that, with the availability of training data, the parameters of the model can be estimated and adapted automatically to give optimal performance. There are many HMM-based state-of-the-art speech recognition systems [1, 12, 27, 34, 49, 55, 72, 73, 93, 108, 109, 112]. Alternatively, we can first identify speech segments, then classify the segments and use the segment scores to recognize words. This approach has produced competitive recognition performance that is similar to HMM-based systems in several small- to medium-vocabulary tasks [115].

Speech recognition systems attempt to model the sources of variability in several ways. At the level of signal representation, in addition to MFCC, researchers have developed representations that emphasize perceptually important speaker-independent features of the signal, and deemphasize speaker-dependent characteristics [43]. Other methods based on linear discriminant analysis to improve class separability [28, 54] and speaker normalization transformation to minimize speaker variations [51, 67, 86, 106, 107, 114] have achieved limited success. Linear discriminant analysis can be traced back to *Fisher's linear discriminant* [30], which projects a dimensional vector onto a single line that is oriented to maximize the class separability. Its extension to speech recognition can be found in [65].

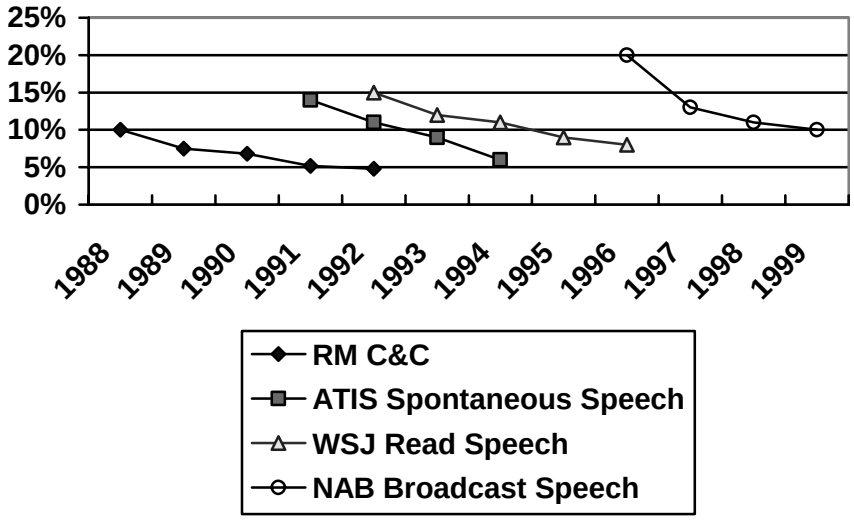


Figure 9.18 History of DARPA speech recognition word-error-rate benchmark evaluation results from 1988 to 1999. There are four major tasks: the Resource Management command and control task (RM C&C, 1000 words), the Air Travel Information System spontaneous speech understanding task (ATIS, 2000 words), the *Wall Street Journal* dictation task (WSJ, 20,000 words), and the Broadcast News Transcription Task (NAB, 60,000 words) [80-84].

At the level of acoustic-phonetic modeling, we need to provide an accurate distance measure of the input feature vectors against the phonetic or word models from the signal-

processing front end. Before the HMM was used, the most successful acoustic-phonetic model was based on the speech template where the feature vectors are stored as the model and dynamic-programming-based time warping was used to measure the distance between the input feature vectors and the word or phonetic templates [88, 94]. The biggest problem for template-based systems is that they are not as trainable as HMMs, since it is difficult to generate a template that is as representative as all the speech samples we have for the particular units of interest.

Another approach that attracted many researchers is the knowledge-based one that originated from the Artificial Intelligence research community. This approach requires extensive knowledge engineering, which often led to many inconsistent rules. Due to the complexity of speech recognition, rule-based approaches generally cannot match the performance of data-driven approaches such as HMMs, which can automatically extract salient rules from a large amount of training data [105].

Senones are now widely used in many state-of-the-art systems. Word models or allophone models can also be built by concatenation of basic structures made by states, transitions, and distributions such as *fenones* [8, 9] or *senones* [58].

Segment models, as proposed by Roucos and Ostendorf [79, 92], assume that each variable-length segment is mapped to a fixed number of representative frames. The resulting model is very similar to the HMM with a large number of states. Ostendorf published a comprehensive survey paper [77] on segment models. The parametric trajectory segment model was introduced by Deng et al. [25] and Gish et al. [37] independently. Gish's work is very similar to our description in Section 9.8.2.1, which is based on Hon et al. [46, 47], where the evaluation and estimation are more efficient because no individual polynormial fitting is required for likelihood computation. In addition to the phone-pair units described in this chapter, segment models have also been applied to phonetic units [25], subphonetic units [25], diphones [36], and syllables [78]. The dynamic model [24, 26] is probably the most aggressive attempt to impose a global transition constrain on the speech model. It uses the phonetic target theories on unobserved vocal-tract parameters, which are fed to an MLP to produce the observed acoustic data.

Today, it is not uncommon to have tens of thousands of sentences available for system training and testing. These corpora permit researchers to quantify the acoustic cues important for phonetic contrasts and to determine parameters of the recognizers in a statistically meaningful way. While many of these corpora were originally collected under the sponsorship of the U.S. Defense Advanced Research Projects Agency (ARPA) to spur human language technology development among its contractors [82], they have nevertheless gained international acceptance. Recognition of the need for shared resources led to the creation of the Linguistic Data Consortium (LDC)[5] in the United States in 1992 to promote and support the widespread development and sharing of resources for human language technology. The LDC supports various corpus development activities and distributes corpora obtained from a variety of sources. Currently, LDC distributes about twenty different speech corpora including those cited above, comprising many hundreds of hours of speech. The availability of a large body of data in the public domain, coupled with the specification of evaluation standards,

[5] http://www.cis.upenn.edu/ldc

has resulted in uniform documentation of test results, thus contributing to greater reliability in monitoring progress.

To further improve the performance of acoustic-phonetic models, we need a robust system so that performance degrades gracefully (rather than catastrophically) as conditions diverge from those under which it was trained. The best approach is likely to have systems continuously adapted to changing conditions (new speakers, microphone, task, etc). Such adaptation can occur at many levels in systems, subword models, word pronunciations, language models, and so on. We also need to make the system portable, so that we can rapidly design, develop, and deploy systems for new applications. At present, systems tend to suffer significant degradation when moved to a new task. In order to retain peak performance, they must be trained on examples specific to the new task, which is time consuming and expensive. In the new task, system users may not know exactly which words are in the system vocabulary. This leads to a certain percentage of out-of-vocabulary words in natural conditions. Currently, systems lack a very robust method of detecting such out-of-vocabulary words. These words often are inaccurately mapped into the words in the system, causing unacceptable errors.

An introduction to all aspects of acoustic modeling can be found in *Spoken Dialogues with Computers* [76] and *Fundamentals of Speech Recognition* [87]. A good treatment of HMM-based speech recognition is given in [52, 60, 105]. Bourlard and Morgan's book [15] is a good introduction to speech recognition based on neural networks. There are a range of applications such as predictive networks that estimate each frame's acoustic vector, given the history [69, 104] and nonlinear transformation of observation vectors [13, 53, 101].

You can find tools to build acoustic models from Carnegie Mellon University's speech open source Web site.[6] This site contains the release of CMU's Sphinx acoustic modeling toolkit and documentation. A version of Microsoft's Whisper system can be found in the Microsoft Speech SDK.[7]

REFERENCES

[1] Abrash, V., *et al.*, "Acoustic Adaptation Using Nonlinear Transformations of HMM Parameters" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1996, Atlanta, pp. 729-732.

[2] Acero, A., *Acoustical and Environmental Robustness in Automatic Speech Recognition*, 1993, Boston, Kluwer Academic Publishers.

[3] Ahadi-Sarkani, S.M., *Bayesian and Predictive Techniques for Speaker Adaptation*, Ph. D. Thesis 1996, Cambridge University, .

[4] Ahn, S., S. Kang, and H. Ko, "Effective Speaker Adaptations For Speaker Verification," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1081-1084.

[5] Alleva, F., *et al.*, "Can Continuous Speech Recognizers Handle Isolated Speech?," *Speech Communication*, 1998, **26**, pp. 183-189.

[6] http://www.speech.cs.cmu.edu/sphinx/

[7] http://www.microsoft.com/speech

[6] Anastasakos, T., *et al.*, "A Compact Model for Speaker Adaptive Training," *Int. Conf. on Spoken Language Processing*, 1996, Philadelphia pp. 1137-1140.

[7] Asadi, A., R. Schwartz, and J. Makhoul, "Automatic Modeling for Adding New Words to a Large-Vocabulary Continuous Speech Recognition System" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1991, Toronto, pp. 305-308.

[8] Bahl, L.R., *et al.*, "Multonic Markov Word Models for Large Vocabulary Continuous Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(3), pp. 334-344.

[9] Bahl, L.R., *et al.*, "A Method for the Construction of Acoustic Markov Models for Words," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(4), pp. 443-452.

[10] Bahl, L.R., *et al.*, "Automatic Phonetic Baseform Determination," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto pp. 173-176.

[11] Bahl, L.R., *et al.*, "Decision Trees for Phonological Rules in Continuous Speech," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, Canada pp. 185-188.

[12] Bellegarda, J.R., *et al.*, "Experiments Using Data Augmentation for Speaker Adaptation," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit pp. 692-695.

[13] Bengio, Y., *et al.*, "Global Optimization of a Neural Network-Hidden Markov Model Hybrid," *IEEE Trans. on Neural Networks*, 1992, **3**(2), pp. 252--259.

[14] Bourlard, H., "A New Training Algorithm for Statistical Sequence Recognition with Applications to Transition-Based Speech Recognition," *IEEE Signal Processing Letters*, 1996, **3**, pp. 203-205.

[15] Bourlard, H. and N. Morgan, *Connectionist Speech Recognition - A Hybrid Approach*, 1994, Boston, MA, Kluwer Academic Publishers.

[16] Brown, P.F., *The Acoustic-Modeling Problem in Automatic Speech Recognition*, PhD Thesis in *Computer Science Department* 1987, Carnegie Mellon University, Pittsburgh, PA.

[17] Campbell, J., "Speaker Recognition: A Tutorial," *Proc. of the IEEE*, 1997, **85**(9), pp. 1437-1462.

[18] Chesta, C., O. Siohan, and C.H. Lee, "Maximum A Posteriori Linear Regression for Hidden Markov Model Adaptation," *Eurospeech*, 1999, Budapest pp. 211-214.

[19] Chou, W., "Maximum A Posteriori Linear Regression with Elliptically Symmetric Matrix Priors," *Eurospeech*, 1999, Budapest pp. 1-4.

[20] Cohen, M., *Phonological Structures for Speech Recognition*, Ph.D. Thesis 1989, University of California, Berkeley, .

[21] Cook, G. and A. Robinson, "Transcribing Broadcast News with the 1997 Abbot System," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle, WA pp. 917-920.

[22] Cox, S., "Predictive Speaker Adaptation in Speech Recognition," *Computer Speech and Language*, 1995, **9**, pp. 1-17.

[23] Davis, S. and P. Mermelstein, "Comparison of Parametric Representations for Monosyllable Word Recognition in Continuously Spoken Sentences," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1980, **28**(4), pp. 357-366.

[24] Deng, L., "A Dynamic, Feature-based Approach to the Interface Between Phonology and Phonetics for Speech Modeling and Recognition," *Speech Communication*, 1998, **24**(4), pp. 299-323.

[25] Deng, L., *et al.*, "Speech Recognition Using Hidden Markov Models with Polynomial Regression Functions as Nonstationary States," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**(4), pp. 507-520.

[26] Digalakis, V., *Segment-based Stochastic Models of Spectral Dynamics for Continuous Speech Recognition*, Ph.D. Thesis in *Electrical Computer System Engineering* 1992, Boston University, .

[27] Digalakis, V. and H. Murveit, "Genones: Optimizing the Degree of Mixture Tying in a Large Vocabulary Hidden Markov Model Based Speech Recognizer" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1994, Adelaide, Australia, pp. 537-540.

[28] Doddington, G.R., "Phonetically Sensitive Discriminants for Improved Speech Recognition," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1989, Glasgow, Scotland pp. 556-559.

[29] Eichner, M. and M. Wolff, "Data-Driven Generation of Pronunciation Dictionaries In The German Verbmobil Project - Discussion of Experimental Results," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1687-1690.

[30] Fisher, R., "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, 1936, **7**(1), pp. 179-188.

[31] Fritsch, J. and M. Finke, "ACID/HNN: Clustering Hierarchies of Neural Networks for Context-Dependent Connectionist Acoustic Modeling," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle, WA pp. 505-508.

[32] Fukunaga, K., *Introduction to Statistical Pattern Recognition*, 2nd ed, 1990, Orlando, FL, Academic Press.

[33] Gales, M., "Cluster Adaptive Training for Speech Recognition," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia pp. 1783-1786.

[34] Gauvain, J.L., L. Lamel, and M. Adda-Decker, "Developments in Continuous Speech Dictation using the ARPA WSJ Task," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit, MI pp. 65-68.

[35] Gauvain, J.L. and C.H. Lee, "Bayesian Learning of Gaussian Mixture Densities for Hidden Markov Models," *Proc. of the DARPA Speech and Natural Language Workshop*, 1991, Palo Alto, CA pp. 272-277.

[36] Ghitza, O. and M.M. Sondhi, "Hidden Markov Models with Templates as Non-Stationary States: An Application To Speech Recognition," *Computer Speech and Language*, 1993, **7**(2), pp. 101-120.

[37] Gish, H. and K. Ng, "A Segmental Speech Model With Applications to Word Spotting," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1993, Minneapolis, MN pp. 447-450.

[38] Gold, B. and N. Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2000, New York, John Wiley.

[39] Greenberg, S., D. Ellis, and J. Hollenback, "Insights into Spoken Language Gleaned from Phonetic Transcription of the Switchboard Corpus," *Int. Conf. on Spoken Language Processing*, 1996, Philadelphia, PA pp. addendum 24-27.

[40] Gunawardana, A., H.W. Hon, and L. Jiang, "Word-Based Acoustic Confidence Measures for Large-Vocabulary Speech Recognition," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia pp. 791-794.

[41] Haeb-Umbach, R., "Investigations on Inter-Speaker Variability in the Feature Space," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1999, Phoenix, AZ.

[42] Hennebert, J., *et al.*, "Estimation of Global Posteriors and Forward-Backward Training of Hybrid HMM/ANN Systems," *Proc. of the Eurospeech Conf.*, 1997, Rhodes, Greece pp. 1951-1954.

[43] Hermansky, H., "Perceptual Linear Predictive (PLP) Analysis of Speech," *Journal of the Acoustical Society of America*, 1990, **87**(4), pp. 1738-1752.

[44] Holmes, W. and M. Russell, "Probabilistic-Trajectory Segmental HMMs," *Computer Speech and Language*, 1999, **13**, pp. 3-37.

[45] Hon, H.W. and K.F. Lee, "CMU Robust Vocabulary-Independent Speech Recognition System," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto pp. 889-892.

[46] Hon, H.W. and K. Wang, "Combining Frame and Segment Based Models for Large Vocabulary Continuous Speech Recognition," *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999, Keystone, CO.

[47] Hon, H.-W. and K. Wang, "Unified Frame and Segment Based Models for Automatic Speech Recognition," *Int. Conf. on Acoustic, Signal and Speech Processing*, 2000, Istanbul, Turkey, IEEE pp. 1017-1020.

[48] Huang, X., *et al.*, "From Sphinx II to Whisper: Making Speech Recognition Usable" in *Automatic Speech and Speaker Recognition*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds. 1996, Norwell, MA, pp. 481-508, Kluwer Academic Publishers.

[49] Huang, X., *et al.*, "From Sphinx-II to Whisper - Make Speech Recognition Usable" in *Automatic Speech and Speaker Recognition*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds. 1996, Norwell, MA, Kluwer Academic Publishers.

[50] Huang, X. and K.-F. Lee, "On Speaker-Independent, Speaker-Dependent, and Speaker-Adaptive Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(2), pp. 150-157.

[51] Huang, X.D., "Speaker Normalization for Speech Recognition," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1992, San Francisco pp. 465-468.

[52] Huang, X.D., Y. Ariki, and M.A. Jack, *Hidden Markov Models for Speech Recognition*, 1990, Edinburgh, U.K., Edinburgh University Press.

[53] Huang, X.D., K. Lee, and A. Waibel, "Connectionist Speaker Normalization and its Applications to Speech Recognition," *IEEE Workshop on Neural Networks for Signal Processing*, 1991, New York pp. 357-366.

[54] Hunt, M.J., *et al.*, "An Investigation of PLP and IMELDA Acoustic Representations and of Their Potential for Combination," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto pp. 881-884.

[55] Huo, Q., C. Chan, and C.-H. Lee, "On-Line Adaptation of the SCHMM Parameters Based on the Segmental Quasi-Bayes Learning for Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1996, **4**(2), pp. 141-144.

[56] Hwang, M.Y., X. Huang, and F. Alleva, "Predicting Unseen Triphones with Senones," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1993, Minneapolis pp. 311-314.

[57] Hwang, M.Y. and X.D. Huang, "Acoustic Classification of Phonetic Hidden Markov Models" in *Proc. of Eurospeech* 1991.

[58] Hwang, M.Y. and X.D. Huang, "Shared-Distribution Hidden Markov Models for Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(4), pp. 414-420.

[59] Iyer, R., *et al.*, "Hidden Markov Models for Trajectory Modeling," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia.

[60] Jelinek, F., *Statistical Methods for Speech Recognition*, 1998, Cambridge, MA, MIT Press.

[61] Jiang, L., H.W. Hon, and X. Huang, "Improvements on a Trainable Letter-to-Sound Converter," *Proc. of Eurospeech*, 1997, Rhodes, Greece pp. 605-608.

[62] Jiang, L. and X. Huang, "Subword-Dependent Speaker Clustering for Improved Speech Recognition," *Int Conf. on Spoken Language Processing*, 2000, Beijing, China.

[63] Jiang, L. and X.D. Huang, "Vocabulary-Independent Word Confidence Measure Using Subword Features," *Int. Conf. on Spoken Language Processing*, 1998, Syndey, Austrilia.

[64] Kuhn, R., *et al.*, "Eigenvoices for Speaker Adaptation," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia pp. 1771-1774.

[65] Kumar, N. and A. Andreou, "Heteroscedastic Discriminant Analysis and Reduced Rank HMMs for Improved Speech Recognition," *Speech Communication*, 1998, **26**, pp. 283-297.

[66] Lee, K.F., *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System*, Ph.D. Thesis in *Computer Science Dept.* 1988, Carnegie Mellon University, Pittsburgh.

[67] Lee, L. and R. Rose, "Speaker Normalization Using Efficient Frequency Warping Procedures," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1996, Atlanta, GA pp. 353-356.

[68] Leggetter, C.J. and P.C. Woodland, "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models," *Computer Speech and Language*, 1995, **9**, pp. 171--185.

[69] Levin, E., "Word Recognition Using Hidden Control Neural Architecture," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1990, Albuquerque pp. 433-436.

[70] Lippmann, R.P., E.A. Martin, and D.P. Paul, "Multi-Style Training for Robust Isolated-Word Speech Recognition," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1987, Dallas, TX pp. 709-712.

[71] Lucassen, J.M. and R.L. Mercer, "An Information-Theoretic Approach to the Automatic Determination of Phonemic Baseforms," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1984, San Diego pp. 42.5.1-42.5.4.

[72] Lyu, R.Y., *et al.*, "Golden Mandarin (III) - A User-Adaptive Prosodic Segment-Based Mandarin Dictation Machine For Chinese Language With Very Large Vocabulary" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1995, Detroit, MI, pp. 57-60.

[73] Matsui, T., T. Matsuoka, and S. Furui, "Smoothed N-best Based Speaker Adaptation for Speech Recognition" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1997, Munich, Germany, pp. 1015--1018.

[74] McDonough, J., *et al.*, "Speaker-Adapted Training on the Switchboard Corpus" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1997, Munich, Germany, pp. 1059--1062.

[75] Morgan, N. and H. Bourlard, *Continuous Speech Recognition: An Introduction to Hybrid HMM/Connectionist Approach*, in *IEEE Signal Processing Magazine*, 1995. pp. 25-42.

[76] Mori, R.D., *Spoken Dialogues with Computers*, 1998, London, Academic Press.

[77] Ostendorf, M., V.V. Digalakis, and O.A. Kimball, "From HMM's to Segment Models: a Unified View of Stochastic Modeling for Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1996, **4**(5), pp. 360-378.

[78] Ostendorf, M. and K. Ross, "A Multi-Level Model for Recognition of Intonation Labels" in *Computing Prosody*, Y. Sagisaka, W.N. Campell, and N. Higuchi, eds. 1997, New York, pp. 291-308, Springer Verlag.

[79] Ostendorf, M. and S. Roukos, "A Stochastic Segment Model for Phoneme-Based Continuous Speech Recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1989, **37**(1), pp. 1857-1869.

[80] Pallett, D., J.G. Fiscus, and J.S. Garofolo, "DARPA Resource Management Benchmark Test Results June 1990" in *Proc. of the DARPA Speech and Natural Language Workshop* 1990, Hidden Valley, PA, pp. 298-305, Pallett.

[81] Pallett, D., J.G. Fiscus, and J.S. Garofolo, "DARPA Resource Management Benchmark Test Results," *Proc. of the DARPA Speech and Natural Language Workshop*, 1991, Morgan Kaufmann Publishers pp. 49-58.

[82] Pallett, D.S., *et al.*, "The 1994 Benchmark Tests for the ARPA Spoken Language Program" in *Proc. of the ARPA Spoken Language Technology Workshop* 1995, Austin, TX, pp. 5-38.

[83] Pallett, D.S., *et al.*, "1997 Broadcast News Benchmark Test Results: English and Non-English," *Proc. of the Broadcast News Transcription and Understanding Workshop*, 1998, Landsdowne, Virginia, Morgan Kaufmann Publishers.

[84] Pallett, D.S., J.G. Fiscus, and M.A. Przybocki, "1996 Preliminary Broadcast News Benchmark Tests," *Proc. of the DARPA Speech Recognition Workshop*, 1997, Chantilly, VA, Morgan Kaufmann Publishers.

[85] Pereira, F., M. Riley, and R. Sproat, "Weighted Rational Transductions and Their Application to Human Language Processing," *Proc. of the ARPA Human Language Technology Workshop*, 1994, Plainsboro, NJ pp. 249-254.

[86] Pye, D. and P.C. Woodland, "Experiments in Speaker Normalization and Adaptation for Large Vocabulary Speech Recognition" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1997, Munich, Germany, pp. 1047--1050.

[87] Rabiner, L.R. and B.H. Juang, *Fundamentals of Speech Recognition*, May, 1993, Prentice-Hall.

[88] Rabiner, L.R. and S.E. Levinson, "Isolated and Connected Word Recognition - Theory and Selected Applications," *IEEE Trans. on Communication*, 1981, **COM-29**(5), pp. 621-659.

[89] Riley, M. and A. Ljolje, eds. *Automatic Generation of Detailed Pronunciation Lexicons*, in Automatic Speech and Speaker Recognition, ed. C. Lee, F. Soong, and K. Paliwal, 1996, Kluwer Academic Publishers.

[90] Robinson, A., "An Application of Recurrent Nets to Phone Probability Estimation," *IEEE Trans. on Neural Networks*, 1994, **5**, pp. 298-305.

[91] Robinson, A.J., *et al.*, "A Neural Network Based, Speaker Independent, Large Vocabulary," *Proc. of the European Conf. on Speech Communication and Technology*, 1999, Berlin pp. 1941--1944.

[92] Roucos, S., *et al.*, "Stochastic Segment Modeling Using the Estimate-Maximize Algorithm," *Int. Conf. on Acoustic, Speech and Signal Processing*, 1988, New York pp. 127-130.

[93] Sagisaka, Y. and L.S. Lee, "Speech Recognition of Asian Languages" in *Proc. IEEE Automatic Speech Recognition Workshop* 1995, Snowbird, UT, pp. 55-57.

[94] Sakoe, H. and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1978, **26**(1), pp. 43-49.

[95] Saraclar, M. and S. Khudanpur, "Pronunciation Ambiguity vs. Pronunciation Variability in Speech Recognition," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1679-1682.

[96] Shinoda, K. and C. Lee, "Structural MAP Speaker Adaptation Using Hierarchical Priors," *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997, Santa Barbara, CA pp. 381-388.

[97] Singh, R., B. Raj, and R. Stern, "Automatic Generation of Phone Sets and Lexical Transcriptions," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1691-1694.

[98] Siohan, O., C. Chesta, and C. Lee, "Joint Maximum A Posteriori Estimation of Transformation and Hidden Markov Model Parameters," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 965-968.

[99] Siu, M., *et al.*, "Parametric Trajectory Mixtures for LVCSR," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia.

[100] Sixtus, A., *et al.*, "Recent Improvements of the RWTH Large Vocabulary Speech Recognition System on Spontaneous Speech," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1671-1674.

[101] Sorenson, H., "A Cepstral Noise Reduction Multi-Layer Network," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto pp. 933-936.

[102] Sproat, R. and M. Riley, "Compilation of Weighted Finite-State Transducers from Decision Trees," *ACL-96*, 1996, Santa Cruz pp. 215-222.

[103] Tajchman, G., E. Fosler, and D. Jurafsky, "Building Multiple Pronunciation Models for Novel Words Using Exploratory Computational Phonology," *Eurospeech*, 1995 pp. 2247-2250.

[104] Tebelskis, J. and A. Waibel, "Large Vocabulary Recognition Using Linked Predictive Neural Networks," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1990, Albuquerque, NM pp. 437-440.

[105] Waibel, A.H. and K.F. Lee, *Readings in Speech Recognition*, 1990, San Mateo, CA, Morgan Kaufman Publishers.

[106] Watrous, R., "Speaker Normalization and Adaptation Using Second-Order Connectionist Networks," *IEEE Trans. on Neural Networks*, 1994, **4**(1), pp. 21-30.

[107] Welling, L., S. Kanthak, and H. Ney, "Improved Methods for Vocal Tract Normalization," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1999, Phoenix, AZ.

[108] Wilpon, J.G., C.H. Lee, and L.R. Rabiner, "Connected Digit Recognition Based on Improved Acoustic Resolution," *Computer Speech and Language*, 1993, **7**(1), pp. 15--26.

[109] Woodland, P.C., *et al.*, "The 1994 HTK Large Vocabulary Speech Recognition System," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit pp. 73-76.

[110] Wooters, C. and A. Stolcke, "Multiple-Pronunciation Lexical Modeling in a Speaker Independent Speech Understanding System," *Proc. of the Int. Conf. on Spoken Language Processing*, 1994, Yokohama, Japan pp. 1363-1366.

[111] Young, S.J. and P.C. Woodland, "The Use of State Tying in Continuous Speech Recognition," *Proc. of Eurospeech*, 1993, Berlin pp. 2203-2206.

[112] Zavaliagkos, G., R. Schwartz, and J. Makhoul, "Batch, Incremental and Instantaneous Adaptation Techniques for Speech Recognition," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit pp. 676-679.

[113] Zavaliagkos, G., *et al.*, "A Hybrid Segmental Neural Net/Hidden Markov Model System for Continuous Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**, pp. 151-160.

[114] Zhan, P. and M. Westphal, "Speaker Normalization Based on Frequency Warping" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1997, Munich, Germany, pp. 1039--1042.

[115] Zue, V., *et al.*, "The MIT SUMMIT System: A Progress Report," *Proc. of DARPA Speech and Natural Language Workshop*, 1989 pp. 179-189.