Chapter 8

LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

8.1 INTRODUCTION

Throughout this book we have developed a wide range of tools, techniques, and algorithms for attacking several fundamental problems in speech recognition. In the previous chapter we saw how the different techniques came together to solve the connected word recognition problem. In this chapter we extend the concepts to include issues needed to solve the large vocabulary, continuous speech recognition problem. We will see that the fundamental ideas need modification because of the use of subword speech units; however, a great deal of the formalism for recognition, based on word units, is still preserved.

The standard approach to large vocabulary continuous speech recognition is to assume a simple probabilistic model of speech production whereby a specified word sequence, W, produces an acoustic observation sequence Y, with probability $P(W, Y)$. The goal is then to decode the word string, based on the acoustic observation sequence, so that the decoded string has the maximum a posteriori (MAP) probability, i.e.,

$$\hat{W} \ni P(\hat{W}|Y) = \max_{W} P(W|Y). \tag{8.1}$$

Using Bayes' Rule, Equation (8.1) can be written as

$$P(W|Y) = \frac{P(Y|W)P(W)}{P(Y)}. \tag{8.2}$$

Since $P(Y)$ is independent of W, the MAP decoding rule of Eq. (8.1) is

$$\hat{W} = \arg\max_{W} P(Y|W)P(W). \tag{8.3}$$

The first term in Eq. (8.3), $P(Y|W)$, is generally called the acoustic model, as it estimates the probability of a sequence of acoustic observations, conditioned on the word string. The way in which we compute $P(Y|W)$, for large vocabulary speech recognition, is to build statistical models for subword speech units, build up word models from these subword speech unit models (using a lexicon to describe the composition of words), and then postulate word sequences and evaluate the acoustic model probabilities via standard concatenation methods. Such methods are discussed in Sections 8.2–8.4 of this chapter.

The second term in Eq. (8.3), $P(W)$, is generally called the language model, as it describes the probability associated with a postulated sequence of words. Such language models can incorporate both syntactic and semantic constraints of the language and the recognition task. Often, when only syntactic constraints are used, the language model is called a grammar and may be of the form of a formal parser and syntax analyzer, an N-gram word model ($N = 2, 3, \ldots$), or a word pair grammar of some type. Generally such language models are represented in a finite state network so as to be integrated into the acoustic model in a straightforward manner. We discuss language models further in Section 8.5 of this chapter.

We begin the chapter with a discussion of subword speech units. We formally define subword units and discuss their relative advantages (and disadvantages) as compared to whole-word models. We next show how we use standard statistical modeling techniques (i.e., hidden Markov models) to model subword units based on either discrete or continuous densities. We then show how such units can be trained automatically from continuous speech, without the need for a bootstrap model of each of the subword units. Next we discuss the problem of creating and implementing word lexicons (dictionaries) for use in both training and recognition phases. To evaluate the ideas discussed in this chapter we use a specified database access task, called the DARPA Resource Management (RM) task, in which there is a word vocabulary of 991 words (plus a silence or background word), and any one of several word grammars can be used. Using such a system, we show how a basic set of subword units performs on this task. Several directions for creating subword units which are more specialized are described, and several of these techniques are evaluated on the RM task. Finally we conclude the chapter with a discussion of how task semantics can be applied to further constrain the recognizer and improve overall performance.

8.2 SUBWORD SPEECH UNITS

We began Chapter 2 with a discussion of the basic phonetic units of language and discussed the acoustic properties of the phonemes in different speech contexts. We then argued that the acoustic variability of the phonemes due to context was sufficiently large and not well understood, that such units would not be useful as the basis for speech models for recognition. Instead, we have used whole-word models as the basic speech unit, both for

isolated word recognition systems and for connected word recognition systems, because whole words have the property that their acoustic representation is well defined, and the acoustic variability occurs mainly in the region of the beginning and the end of the word. Another advantage of using whole-word speech models is that it obviates the need for a word lexicon, thereby making the recognition structure inherently simple.

The disadvantages of using whole-word speech models for continuous speech recognition are twofold. First, to obtain reliable whole-word models, the number of word utterances in the training set needs to be sufficiently large, i.e., each word in the vocabulary should appear in each possible phonetic context several times in the training set. In this way the acoustic variability at the beginning and at the end of each word can be modeled appropriately. For word vocabularies like the digits, we know that each digit can be preceded and followed by every other digit; hence for an 11-digit vocabulary (zero to nine plus oh), there are exactly 121 phonetic contexts (some of which are essentially identical). Thus with a training set of several thousand digit strings, it is both realistic and practical to see every digit in every phonetic context several times. Now consider a vocabulary of 1000 words with an average of 100 phonetic contexts for both the beginning and end of each word. To see each word in each phonetic context exactly once requires $100 \times 1000 \times 100 = 10$ million carefully designed sentences. To see each combination 10 times requires 100 million such sentences. Clearly, the recording and processing of such homogeneous amounts of speech data is both impractical and unthinkable. Second, with a large vocabulary the phonetic content of the individual words will inevitably overlap. Thus storing and comparing whole-word patterns would be unduly redundant because the constituent sounds of individual words are treated independently, regardless of their identifiable similarities. Hence some more efficient speech representation is required for such large vocabulary systems. This is essentially the reason we use subword speech units.

There are several possible choices for subword units that can be used to model speech. These include the following:

- **Phonelike units** (PLUs) in which we use the basic phoneme set (or some appropriately modified set) of sounds but recognize that the acoustic properties of these units could be considerably different than the acoustic properties of the "basic" phonemes [1–7]. This is because we define the units based on linguistic similarity but model the unit based on acoustic similarity. In cases in which the acoustic and phonetic similarities are roughly the same (e.g., stressed vowels) then the phoneme and the PLU will be essentially identical. In other cases there can be large differences and a simple one-to-one correspondence may be inadequate in terms of modeling accuracy. Typically there are about 50 PLUs for English.
- **Syllable-like units** in which we again use the linguistic definition of a syllable (namely a vowel nucleus plus the optional initial and final consonants or consonant clusters) to initially define these units, and then model the unit based on acoustic similarity. In English there are approximately 10,000 syllables.
- **Dyad or demisyllable-like units** consisting of either the initial (optional) consonant cluster and some part of the vowel nucleus, or the remaining part of the vowel nucleus and the final (optional) consonant cluster [8]. For English there is something on the

order of 2000 demisyllable-like units.

- **Acoustic units**, which are defined on the basis of clustering speech segments from a segmentation of fluent, unlabeled speech using a specified objective criterion (e.g., maximum likelihood) [9]. Literally a codebook of speech units is created whose interpretation, in terms of classical linguistic units, is at best vague and at worst totally nonexistent. It has been shown that a set of 256–512 acoustic units is appropriate for modeling a wide range of speech vocabularies.

Consider the English word *segmentation*. Its representation according to each of the above subword unit sets is

- **PLUs**: /s/ /ɛ/ /g/ /m/ /ə/ /n/ /t/ /e^y/ /sh/ /ə/ /n/ (11 units)
- **syllables**: /seg/ /men/ /ta/ /tion/ (4 syllables)
- **demisyllables**: /sɛ/ /ɛg/ /mə/ /ən/ /te^y/ /e^{y}sh/ /shə/ /ən/ (8 demisyllables)
- **acoustic units**: 17 111 37 3 241 121 99 171 37 (9 acoustic units).

We see, from the above example, that the number of subword units for this word can be as small as 4 (from a set of 10,000 units) or as large as 11 (from a set of 50 units).

Since each of the above subword unit sets is capable of representing any word in the English language, the issues in the choice of subword unit sets are the context sensitivity and the ease of training the unit from fluent speech. (In addition, for acoustic units, an issue is the creation of a word lexicon since the units themselves have no inherent linguistic interpretation.) It should be clear that there is no ideal (perfect) set of subword units. The PLU set is extremely context sensitive because each unit is potentially affected by its predecessors (one or more) and its followers. However, there is only a small number of PLUs and they are relatively easy to train. On the other extreme are the syllables which are longest units and are the least context sensitive. However, there are so many of them that they are almost as difficult to train as whole-word models.

For simplicity we will initially assume that we use PLUs as the basic speech units. In particular we use the set of 47 PLUs shown in Table 8.1 (which includes an explicit symbol for silence –h#). For each PLU we show an orthographic symbol (e.g., aa) and a word associated with the symbol (e.g., father). (These symbols are essentially identical to the ARPAPET alphabet of Table 2.1; lowercase symbols are used throughout this chapter for consistency with the DARPA community.) Table 8.2 shows typical pronunciations for several words from the DARPA RM task in terms of the PLUs in Table 8.1. A strong advantage of using PLUs is the ease of creating word lexicons of the type shown in Table 8.2 from standard (electronic) dictionaries. We will see later in this chapter how we exploit the advantages of PLUs, while reducing the context dependencies, by going to more specialized PLUs which take into consideration either the left or right (or both) contexts in which the PLU appears.

One problem with word lexicons of the type shown in Table 8.2 is that they don't easily account for variations in word pronunciation across different dialects and in the context of a sentence. Hence a simple word like "a" is often pronounced as /ey/ in isolation (e.g., the

TABLE 8.1. Set of basic PLUs for speech.

Number	Symbol	Word	Number	Symbol	Word
1	h#	silence	26	k	*k*ick
2	aa	father	27	l	*l*ed
3	ae	b*a*t	28	m	*m*om
4	ah	b*u*tt	29	n	*n*o
5	ao	b*ough*t	30	ng	si*ng*
6	aw	b*ough*	31	ow	b*oa*t
7	ax	*a*gain	32	oy	b*oy*
8	axr	din*er*	33	p	*p*op
9	ay	b*i*te	34	r	*r*ed
10	b	*b*ob	35	s	*s*is
11	ch	*ch*urch	36	sh	*sh*oe
12	d	*d*ad	37	t	*t*ot
13	dh	*th*ey	38	th	*th*ief
14	eh	b*e*t	49	uh	b*oo*k
15	el	bott*le*	40	uw	b*oo*t
16	en	butt*on*	41	v	*v*ery
17	er	b*ir*d	42	w	*w*et
18	ey	b*ai*t	43	y	*y*et
19	f	*f*ief	44	z	*z*oo
20	g	*g*ag	45	zh	mea*s*ure
21	hh	*h*ag	46	dx	bu*tt*er
22	ih	b*i*t	47	nx	ce*n*ter
23	ix	ros*es*			
24	iy	b*ea*t			
25	jh	*j*udge			

TABLE 8.2. Typical word pronunciations (word lexicon) based on context-independent PLUs.

Word	Number of phones	Transcription				
a	1	ax				
above	4	ax	b	ah	v	
bad	3	b	ae	d		
carry	4	k	ae	r	iy	
define	5	d	iy	f	ay	n
end	3	eh	n	d		
gone	3	g	ao	n		
hours	4	aw	w	axr	z	

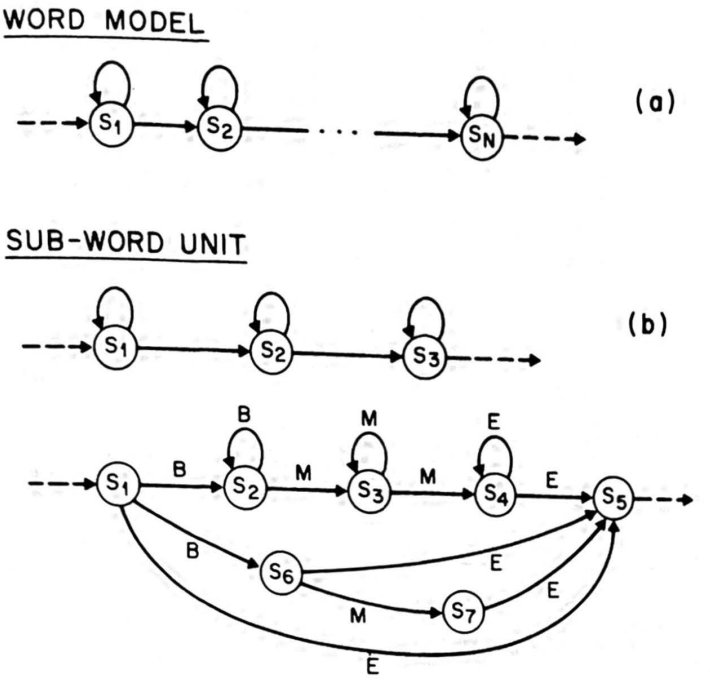


Figure 8.1 HMM representations of a word (a) and a subword unit (b).

letter A), but is pronounced as /ax/ in context. Another example is a word like "data," which can be pronounced as /d ey t ax/ or /d ae t ax/ depending on the speaker's dialect. Finally words like "you" are normally pronounced as /y uw/ but in context often are pronounced as /jh ax/ or /jh uh/. There are several ways of accounting for word pronunciation variability, including multiple entries in the word lexicon, use of phonological rules in the recognition grammar, and use of context dependent PLUs. We will discuss these options later in this chapter.

8.3 SUBWORD UNIT MODELS BASED ON HMMS

As we have shown several times in this book, the most popular way in which speech is modeled is as a left-to-right hidden Markov model. As shown in Figure 8.1a, a whole-word model typically uses a left-to-right HMM with N states, where N can be a fixed value (e.g., 5–10 for each word), or can be variable with the number of sounds (phonemes) in the word, or can be set equal to the average number of frames in the word. For subword units, typically, the number of states in the HMM is set to a fixed value, as shown in Figure 8.1b where a three-state model is used. This means that the shortest tokens of each subword unit must last at least three frames, a restriction that seems reasonable in practice. (Models that use jumps to eliminate this restriction have been studied [2].)

To represent the spectral density associated with the states of each subword unit, one of three approaches can be used. These approaches are illustrated in Figure 8.2. Perhaps the simplest approach is to design a VQ-based codebook for all speech sounds (as shown in part a of the figure). For this approach the probability density of the observed

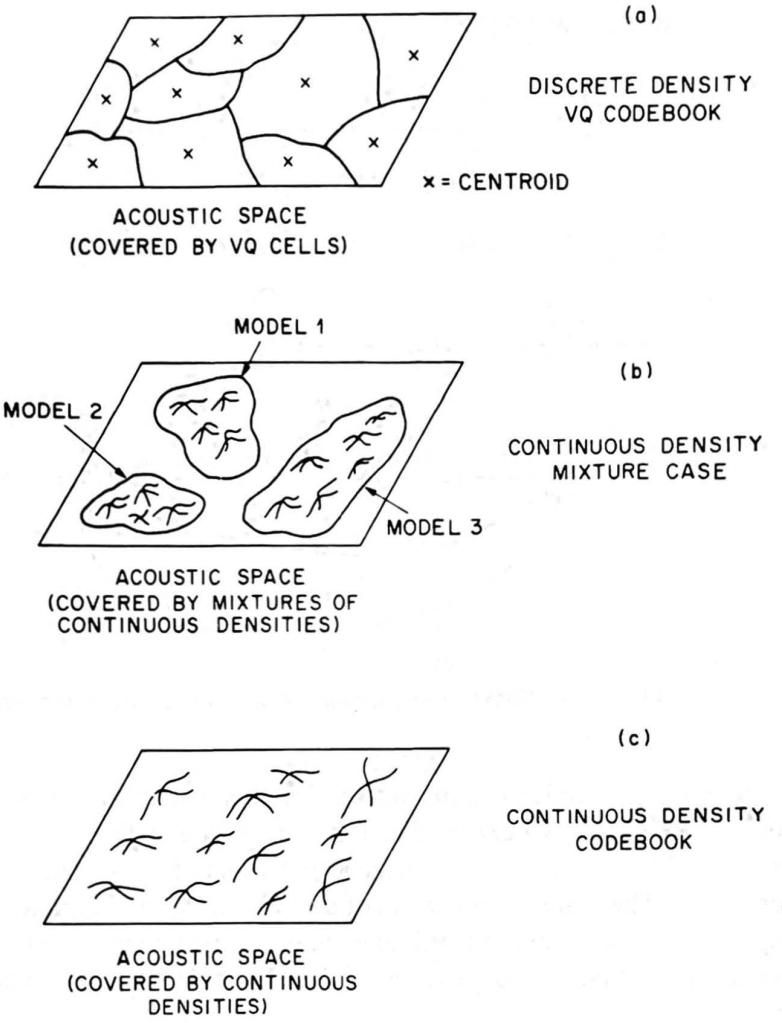


Figure 8.2 Representations of the acoustic space of speech by (a) partitioned VQ cells, (b) sets of continuous mixture Gaussian densities, and (c) a continuous-density codebook (after Lee et al. [7]).

spectral sequence within each state of each PLU is simply a discrete density defined over the codebook vectors. The interpretation of the discrete density within a state is that of implicitly isolating the part of the acoustic space in which the spectral vectors occur and assigning the appropriate codebook vector (over that part of the space) a fixed probability for spectral vectors within each isolated region regardless of its proximity to the corresponding codebook vector. A second alternative, illustrated in part b of Figure 8.2, is to represent the continuous probability density in each subword unit state by a mixture density that explicitly defines the part of the acoustic space in which the spectral vectors occur. Each mixture component has a spectral mean and variance that is highly dependent on the spectral characteristics of the subword unit (i.e., highly localized in the acoustic space). Hence the models for different subword units usually do not have substantial overlap in the acoustic space. Finally, a third alternative is to design a type of continuous density codebook over the entire acoustic space, as illustrated in part c of Figure 8.2. Basically the entire acoustic

space is covered by a set of independent Gaussian densities, derived in much the same way as the discrete VQ codebook, with the resulting set of means and covariances stored in a codebook. This alternative is a compromise between the previous two possibilities. It differs from the discrete density case in the way the probability of an observation vector is computed; instead of assigning a fixed probability to any observation vector that falls within an isolated region, it actually determines the probability according to the closeness of the observation vector to the codebook vector (i.e., it calculates the exponents of the Gaussian distributions). For each state of each subword unit, the density is assumed to be a mixture of the fixed codebook densities. Hence, even though each state is characterized by a continuous mixture density, one need only estimate the set of mixture gains to specify the continuous density completely. Furthermore, since the codebook set of Gaussian densities is common for all states of all subword models, one can precompute the likelihoods associated with an input spectral vector for each of the codebook vectors, and ultimately determine state likelihoods using only a simple dot product with the state mixture gains. This represents a significant computational reduction over the full mixture continuous density case. This mixed density method has been called the tied mixture approach [10, 28] as well as the semicontinuous modeling method [11] and has been applied to the entire acoustic space as well as to pieces of the acoustic space for detailed PLU modeling. This method can be further extended to the case in which a set of continuous density codebooks is designed, one for each state of each basic (context independent) speech unit. One can then estimate sets of mixture gains appropriate to context dependent versions of each basic speech unit and use them appropriately for recognition. We will return to this issue later in this chapter.

8.4 TRAINING OF SUBWORD UNITS

Implicitly it would seem that training of the models for subword units would be extremely difficult, because there is no simple way to create a bootstrap model of such short, imprecisely defined, speech sounds. Fortunately, this is not the case. The reason for this is because of the inherent tying of subword units across words and sentences—that is, every subword unit occurs a large number of times in any reasonable size training set. Hence estimation algorithms like the forward-backward procedure, or the segmental k-means algorithm, can start with a uniform segmentation (flat or random initial models) and rapidly converge to the best model estimates in just a few iterations.

To illustrate how models of subword units are estimated, assume we have a labeled training set of speech sentences, where each sentence consists of the speech waveform and its transcription into words. (We assume that waveform segmentation into words is not available.) We further assume a word lexicon is available that provides a transcription of every word in the training set strings in terms of the set of subword units being trained. We assume that silence can (but needn't) precede or follow any word within a sentence (i.e., we allow pauses in speaking), with silence at the beginning and end of each sentence the most likely situation. Based on the above assumptions, a typical sentence in the training set can be transcribed as

$$S_W: \; W_1 \; W_2 \; W_3 \; \ldots \; W_I,$$

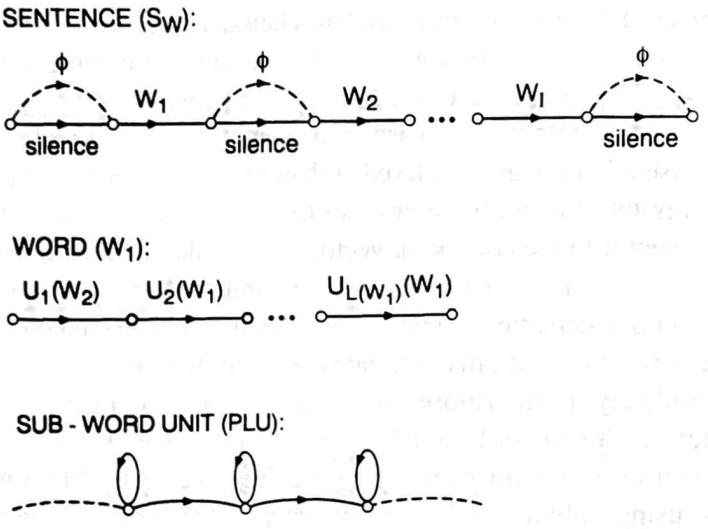


Figure 8.3 Representation of a sentence, word, and subword unit in terms of FSNs.

in which each W_i, $1 \leq i \leq I$, is a word in the lexicon. Hence the sentence "show all alerts" is a three-word sentence with W_1 = show, W_2 = all, and W_3 = alerts. Each word can be looked up in the lexicon to find its transcription in terms of subword units. Hence the sentence S can be written in terms of subword units as

$$S_U: \quad U_1(W_1)U_2(W_1)\ldots U_{L(W_1)}(W_1) \oplus U_1(W_2)U_2(W_2)\ldots U_{L(W_2)}(W_2) \oplus U_1(W_3)U_2(W_3)\ldots U_{L(W_3)}(W_3) \oplus \cdots \oplus U_1(W_I)U_2(W_I)\ldots U_{L(W_I)}(W_I),$$

where $L(W_1)$ is the length (in units) of word W_1, etc. Finally we replace each subword unit by its HMM (the three-state models shown in Figure 8.1) and incorporate the assumptions about silence between words to give an extended HMM for each sentence.

The above process is illustrated (in general) in Figure 8.3. We see that a sentence is represented as a finite-state network (FSN) where the arcs are either words or silence or null arcs (where a null (ϕ) transition is required to skip the alternative silence). Each word is represented as an FSN of subword units and each subword unit is represented as a three-state HMM.

Figure 8.4 shows the process of creating the composite FSN for the sentence "Show all alerts," based on a single-word pronunciation lexicon. One feature of this implementation is the use of a single-state HMM for the silence word. This is used (rather than the three-state HMMs used for each PLU), since silence is generally stationary and has no temporal structure to exploit.

When there are multiple representations of words in the lexicon (e.g., for two or more distinct pronunciations) it is easy to modify the FSN of Figures 8.3 and 8.4 to add parallel paths for the word arcs. (We will see that only one path is chosen in training, namely the best representation of the actual word pronunciation in the context of the spoken sentence.) Furthermore, multiple models of each subword unit can be used by introducing parallel paths in the word FSNs and then choosing the best version of each subword unit in the decoding process.

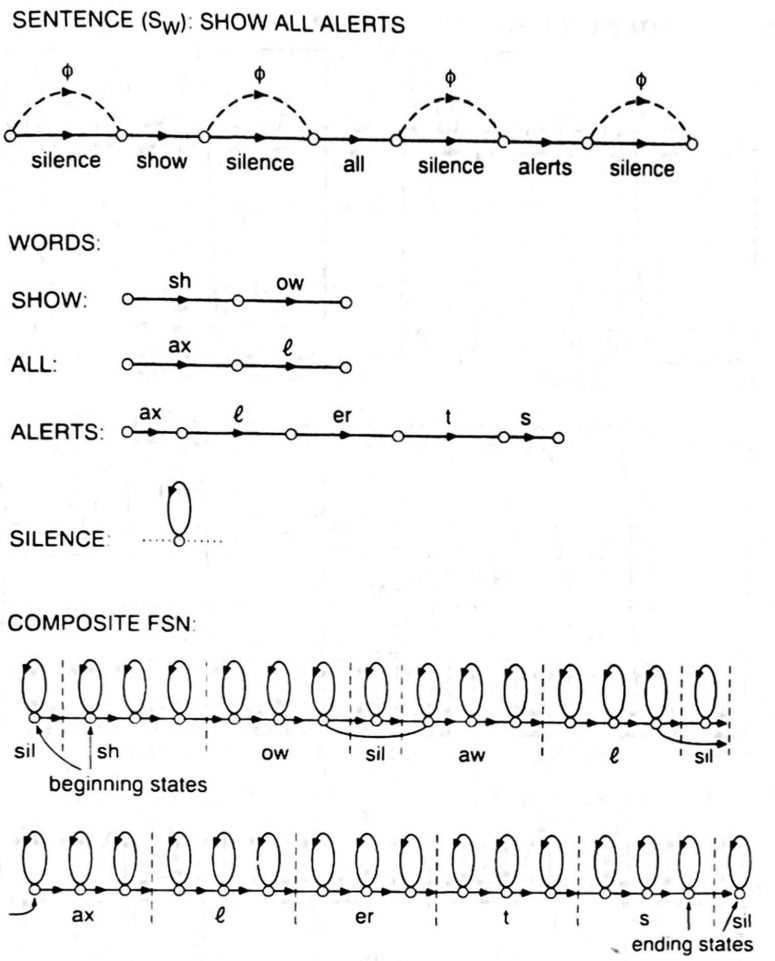


Figure 8.4 Creation of composite FSN for sentence "Show all alerts."

Once a composite sentence FSN is created for each sentence in the training set, the training problem becomes one of estimating the subword unit model parameters which maximize the likelihood of the models for all the given training data. The maximum likelihood parameters can be solved for using either the forward-backward procedure (see Ref. [2] for example) or the segmental k-means training algorithm. The way in which we use the segmental k-means training procedure to estimate the set of model parameters (based on using a mixture density with M mixtures/state) is as follows:

1. **Initialization**: Linearly segment each training utterance into units and HMM states assuming no silence between words (i.e., silence only at the beginning and end of each sentence), a single lexical pronunciation of each word, and a single model for each subword unit. Figure 8.5, iteration 0, illustrates this step for the first few units of one training sentence. Literally we assume every unit is of equal duration initially.
2. **Clustering**: All feature vectors from all segments corresponding to a given state (i) of a given subword unit are partitioned into M clusters using the k-means algorithm. (This step is iterated for all states of all subword units.)
3. **Estimation**: The mean vectors, $\boldsymbol{\mu}_{ik}$, the (diagonal) covariance matrices, $\mathbf{U}_{ik}$, and the

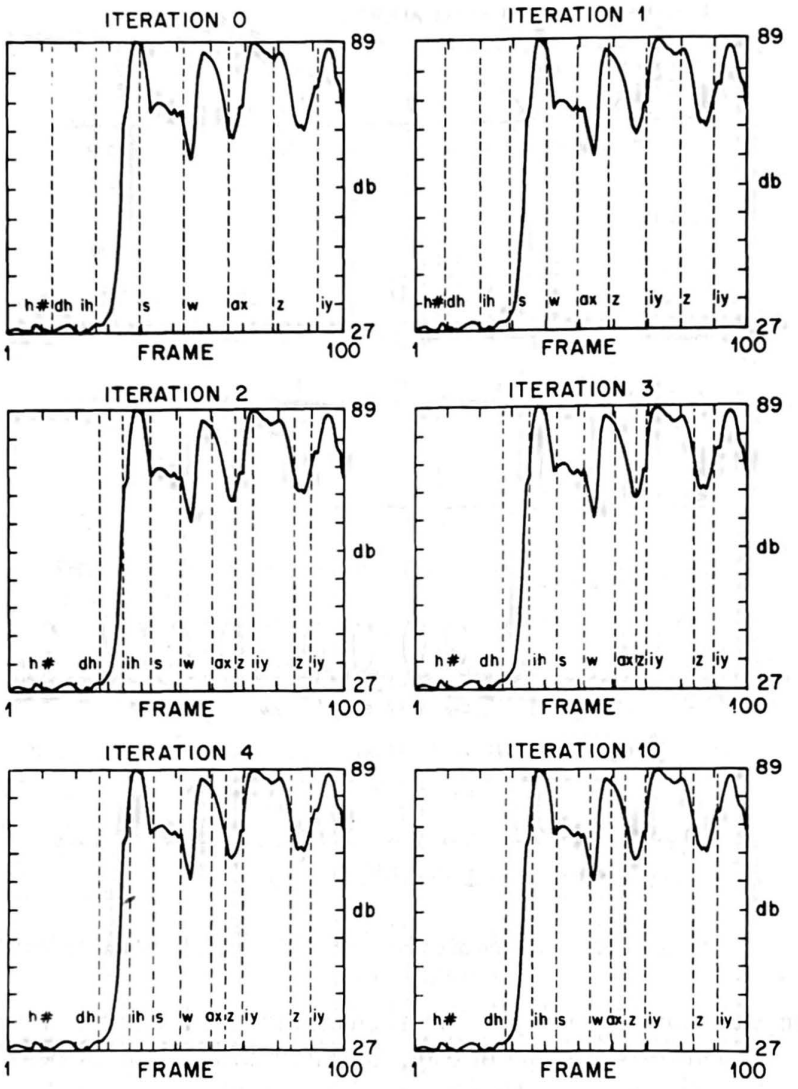


Figure 8.5 Segmentations of a training utterance resulting from the segmental k-means training for the first several iterations (after Lee et al. [7]).

mixture weights, c_{ik}, are estimated for each cluster k in state i. (This step is iterated for all states of all subword units.)

4. **Segmentation**: The updated set of subword unit models (based on the estimation of step 3) is used to resegment each training utterance into units and states (via Viterbi decoding). At this point multiple lexical entries can be used for any word in the vocabulary. Figure 8.5 shows the result of this resegmentation step for iterations 1–4 and 10 for one training utterance. It can be seen that by iteration 2 the segmentation into subword units is remarkably stable.
5. **Iteration**: Steps 2–4 are iterated until convergence (i.e., until the overall likelihoods stop increasing).

Figure 8.6 illustrates the resulting segmentation of the first few units of the utterance

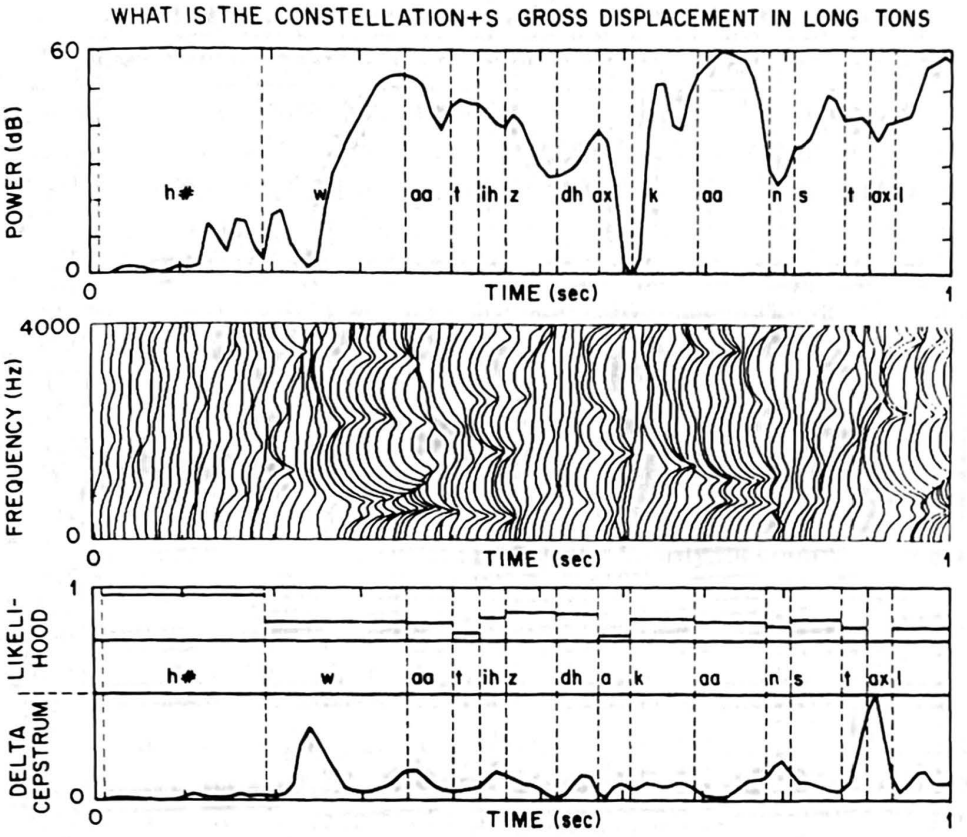


Figure 8.6 Segmentation of an utterance into PLUs (after Lee et al. [7]).

"What is the constellation. . . ." Shown in this figure are the power contour in dB (upper panel), the running LPC spectral slices (the middle panel), and the likelihood scores and delta-cepstral values (lower panel) for the first second of the sentence. The resulting segmentations are generally remarkably consistent with those one might manually choose based on acoustic-phonetic criteria. Since we use an acoustic criterion for choice of segmentation points, the closeness of PLU units to true phonetic units is often remarkable, especially in light of the phonetic variability in word pronunciation discussed previously.

In summary we have shown how one can use a training set of speech sentences that have only word transcriptions associated with each sentence and optimally determine the parameters of a set of subword unit HMMs. The resulting parameter estimates are extremely robust to the training material as well as to details of word pronunciation as obtained from the word lexicon. The reason for this is that a common word lexicon (with associated word pronunciation errors) is used for both training and recognition; hence errors in associating proper subword units to words are consistent throughout the process and are less harmful than they would be in alternative methods of estimating parameters of subword models.

The results of applying the segmental k-means training procedure to a set of 3990 training sentences from 109 different talkers, in terms of PLU counts and PLU likelihood scores are shown in Table 8.3. A total of 155,000 PLUs occurred in the 3990 sentences with silence (h#) having the most occurrences (10,638 or 6.9% of the total) and nx (flapped

TABLE 8.3. PLU statistics on count and average likelihood score.

PLU	Count	%	Average likelihood	(Rank)
h#	10638	6.9	18.5	(1)
r	8997	5.8	8.4	(45)
t	8777	5.7	9.7	(37)
ax	8715	5.6	7.1	(47)
s	8625	5.6	15.4	(3)
n	8478	5.5	8.3	(46)
ih	6542	4.2	9.9	(35)
iy	5816	3.7	12.0	(17)
d	5391	3.5	8.5	(44)
ae	4873	3.1	13.3	(10)
ℓ	4857	3.1	8.9	(41)
z	4733	3.0	12.4	(14)
eh	4604	3.0	11.2	(21)
k	4286	2.8	10.6	(27)
p	3793	2.4	14.3	(6)
m	3625	2.3	8.5	(43)
ao	3489	2.2	10.4	(32)
f	3276	2.1	17.7	(2)
ey	3271	2.1	14.5	(5)
w	3188	2.1	10.2	(34)
ix	3079	2.0	8.7	(42)
dh	2984	1.9	11.8	(18)
v	2979	1.9	12.0	(16)
aa	2738	1.8	10.3	(33)
b	2138	1.4	10.7	(25)
y	2137	1.4	13.1	(11)
uw	2032	1.3	10.6	(26)
sh	1875	1.2	13.1	(12)
ow	1875	1.2	10.9	(24)
axr	1825	1.2	9.5	(38)
ah	1566	1.0	11.3	(20)
dx	1548	1.0	10.4	(31)
ay	1527	1.0	13.9	(8)
en	1478	0.9	9.1	(40)
g	1416	0.9	9.8	(36)
hh	1276	0.8	11.4	(19)
th	924	0.6	14.1	(7)
ng	903	0.6	9.1	(39)
ch	885	0.6	12.5	(13)
el	863	0.6	11.0	(23)
er	852	0.5	10.6	(29)
jh	816	0.5	10.6	(28)
aw	682	0.4	13.6	(9)
uh	242	0.2	11.0	(22)
zh	198	0.1	12.2	(15)
oy	130	0.1	15.3	(4)
nx	57	0.04	10.4	(30)

n) having the fewest occurrences (5 or 0.04% of the total). In terms of average likelihood scores, silence (h#) had the highest score (18.5) followed by f (17.7) and s (15.4), while ax had the lowest score (7.1), followed by n (8.3) and r (8.4). (Note that, in this case, a higher average likelihood implies less variation among different renditions of the particular sound.) It is interesting to note that the PLUs with the three lowest average likelihood scores (ax, n, and r) were among the most frequently occurring sounds (r was second, n sixth, and ax fourth in frequency of occurrence). Similarly, some of the sounds with the highest likelihood scores were among the least occurring sounds (e.g., oy was fourth according to likelihood score but 21st according to frequency of occurrence).

8.5 LANGUAGE MODELS FOR LARGE VOCABULARY SPEECH RECOGNITION

Small vocabulary speech-recognition systems are used primarily for command-and-control applications where the vocabulary words are essentially acoustic control signals that the system has to respond to. (See Chapter 9 for a discussion of command-and-control applications of speech recognition.) As such, these systems generally do not rely heavily on language models to accomplish their selected tasks. A large vocabulary speech-recognition system, however, is generally critically dependent on linguistic knowledge embedded in the input speech. Therefore, for large vocabulary speech recognition, incorporation of knowledge of the language, in the form of a "language" model, is essential. In this section we discuss a statistically motivated framework for language modeling.

The goal of the (statistical) language model is to provide an estimate of the probability of a word sequence W for the given recognition task. If we assume that W is a specified sequence of words, i.e.,

$$W = w_1 w_2 \ldots w_Q, \tag{8.4}$$

then it would seem reasonable that $P(W)$ can be computed as

$$P(W) = P(w_1 w_2 \ldots w_Q) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\ldots P(w_Q|w_1 w_2 \ldots w_{Q-1}). \tag{8.5}$$

Unfortunately, it is essentially impossible to reliably estimate the conditional word probabilities, $P(w_j|w_1 \ldots w_{j-1})$ for all words and all sequence lengths in a given language. Hence, in practice, it is convenient to use an N-gram word model, where we approximate the term $P(w_j|w_1 \ldots w_{j-1})$ as

$$P(w_j|w_1 w_2 \ldots w_{j-1}) \approx P(w_j|w_{j-N+1} \ldots w_{j-1}), \tag{8.6}$$

i.e., based only on the preceding $N-1$ words. Even N-gram probabilities are difficult to estimate reliably for all but $N = 2$ or possibly 3. Hence, in practice, it is often convenient to use a word pair model that specifies which word pairs are valid in the language through the use of a binary indicator function, i.e.,

$$P(w_j|w_k) = \begin{cases} 1 & \text{if } w_k w_j \text{ is valid} \\ 0 & \text{otherwise} \end{cases} \tag{8.7}$$

Another simple language model, often called the no-grammar model, assumes $P(w_j|w_k) = 1$ for all j and k, so that every word is assumed capable of being followed by every other word in the language. In the next section we show how the word pair and the no-grammar models can be implemented as finite state networks so as to be integrated simply into a recognition decoding algorithm.

Alternative language models include formal grammars (e.g., context free or context dependent grammar), N-grams of word classes (rather than words) etc. These types of grammars provide more realistic models for natural language input to machines than the artificial N-grams or words, or the word pair grammars. However, they are somewhat more difficult to integrate with the acoustic decoding and hence will not be discussed here.

8.6 STATISTICAL LANGUAGE MODELING

In large vocabulary speech recognition, in which word sequences W are uttered to convey some message, the language model $P(W)$ is of critical importance to the recognition accuracy as shown in Eq. (8.3). In most cases, the language model $P(W)$ has to be estimated from a given (large) text corpus. In this section we discuss how to construct such a statistical language model from a (textual) training corpus.

For practical reasons, the word sequence probability $P(W)$ is approximated by

$$P_N(W) = \prod_{i=1}^{Q} P(w_i|w_{i-1}, w_{i-2}, \ldots, w_{i-N+1}), \tag{8.8}$$

which is called an N-gram language model. The conditional probabilities $P(w_i|w_{i-1}, \ldots, w_{i-N+1})$ can be estimated by the simple relative frequency approach,

$$\hat{P}(w_i|w_{i-1}, \ldots, w_{i-N+1}) = \frac{F(w_i, w_{i-1}, \ldots, w_{i-N+1})}{F(w_{i-1}, \ldots, w_{i-N+1})}, \tag{8.9}$$

in which F is the number of occurrences of the string in its argument in the given training corpus. Obviously, in order for the estimate in Eq. (8.9) to be reliable, $F(w_i, w_{i-1}, \ldots, w_{i-N+1})$ has to be substantial in the given corpus. The implications of this are that the size of the training corpus may be prohibitively large and that $F(w_i, w_{i-1}, \ldots, w_{i-N+1}) = 0$ for many possible word strings, $w_i, w_{i-1}, \ldots, w_{i-N+1}$, due to the limited size of the corpus.

One way to circumvent this problem is to smooth the N-gram frequencies as suggested by Jelinek et al. [12]. Consider $N = 3$, the trigram model. The smoothing is done by interpolating trigram, bigram and unigram relative frequencies

$$\hat{P}(w_3|w_1, w_2) = p_1 \frac{F(w_1, w_2, w_3)}{F(w_1, w_2)} + p_2 \frac{F(w_1, w_2)}{F(w_1)} + p_3 \frac{F(w_1)}{\sum F(w_i)}, \tag{8.10}$$

in which the nonnegative weights satisfy $p_1 + p_2 + p_3 = 1$ and $\sum F(w_i)$ is the size of the corpus. The weights depend on the values of $F(w_1, w_2)$ and $F(w_1)$ and can be obtained by applying the principle of cross-validation [12].

8.7 PERPLEXITY OF THE LANGUAGE MODEL

Having constructed a language model from a training corpus, one may ask how well the language model will perform in the context of speech recognition. This can be answered based on the concept of source of information in information theory. To provide such a measure of performance, we must first discuss several concepts, including entropy, estimated entropy, and perplexity.

Consider an information source that puts out sequences of words (symbols) w_1, w_2, ..., w_Q, each of which is chosen from a vocabulary $\bar{V}$ with size $|\bar{V}|$, according to some stochastic law. The entropy of the source can be defined as

$$H = -\lim_{Q\to\infty}\left(\frac{1}{Q}\right)\left\{\sum P(w_1, w_2, \ldots, w_Q)\log\ P(w_1, w_2, \ldots, w_Q)\right\}, \tag{8.11}$$

in which $P(\ )$ is the probability of the argument string the source will put out according to the stochastic law and the summation is over all string sequences $w_1, w_2, \ldots, w_Q$. If the words in the string sequence are generated by the source in an independent manner

$$P(w_1, w_2, \ldots, w_Q) = P(w_1)P(w_2)\ldots P(w_Q), \tag{8.12}$$

then

$$H = -\sum_{w\in\bar{V}} P(w)\log\ P(w), \tag{8.13}$$

which is sometimes referred to as the first-order entropy of the source (even if Eq. (8.12) is not true).

The quantity H of Eq. (8.11) can be considered as the average information of the source when it puts out a word w. Equivalently, a source of entropy H is one that has as much information content as a source which puts out words equiprobably from a vocabulary of size 2^H.

If the source is ergodic (meaning its statistical properties can be completely characterized in a sufficiently long sequence that the source puts out), the entropy of Eq. (8.11) is equivalent to

$$H = -\lim_{Q\to\infty}\left(\frac{1}{Q}\right)\log\ P(w_1, w_2, \ldots, w_Q). \tag{8.14}$$

In other words, we can compute the entropy from a "typical" (long) sequence of words generated by the source. The length of this typical sequence (i.e., the corpus) has to approach infinity, which is of course unattainable. We often compute H based on a finite but sufficiently large Q; i.e.,

$$H = -\left(\frac{1}{Q}\right)\log\ P(w_1, w_2, \ldots, w_Q). \tag{8.15}$$

An interesting interpretation of H from the perspective of speech recognition is that it is the degree of difficulty that the recognizer encounters, on average, when it is to determine a word from the same source. This difficulty, or uncertainty, is based on the actual probability $P(w_1, w_2, \ldots, w_Q)$ which, for natural languages, is usually not known

beforehand and thus has to be estimated. (We do not include acoustic uncertainty in the present context of language modeling.)

One way to estimate H is to use $P(W) = P(w_1, w_2, \ldots, w_Q)$ from the language model. For example, if the N-gram language model $P_N(W)$ (Eq. (8.8)) is used, an estimate of H of Eq. (8.15) is thus

$$H_p = -\frac{1}{Q} \sum_{i=1}^{Q} \log P(w_i | w_{i-1}, w_{i-2}, \ldots, w_{i-N+1}). \tag{8.16}$$

In general,

$$H_p = -\frac{1}{Q} \log \hat{P}(w_1, w_2, \ldots, w_Q), \tag{8.17}$$

where $\hat{P}(w_1, w_2, \ldots, w_Q)$ is an estimate of $P(w_1, w_2, \ldots, w_Q)$. The quantity H_p is an estimated entropy as calculated from a sufficiently long sequence based on a language model. If the source is ergodic and $Q \to \infty, H_p \geq H$. Intuitively, this can be easily verified by the fact that knowledge of the true probability $P(w_1, w_2, \ldots, w_Q)$ is the best a recognizer can use and any other probability estimate or language model can never make the task easier for the recognizer. Since H_p is an indication of the recognition difficulty lower-bounded by H, a language model that achieves a lower H_p (i.e., closer to H) is therefore considered a better model than another language model which leads to a higher H_p.

Associated with H_p is a quantity called perplexity (often called the average word branching factor of the language model) defined as

$$B = 2^{H_p} = \hat{P}(w_1, w_2, \ldots, w_Q)^{-1/Q}. \tag{8.18}$$

Note that H_p is the average difficulty or uncertainty in each word based on the language model. When the recognizer uses this language model for the task, the difficulty it faces is equivalent to that of recognizing a text generated by a source that chooses words from a vocabulary size of B independently of each other and with equal probability. Another way to view perplexity is to consider it as the average number of possible words following any string of $(N-1)$ words in a large corpus based on an N-gram language model. Perplexity is an important parameter in specifying the degree of sophistication in a recognition task, from the source uncertainty to the quality of the language model.

8.8 OVERALL RECOGNITION SYSTEM BASED ON SUBWORD UNITS

A block diagram of the overall continuous speech-recognition system based on subword speech units is shown in Figure 8.7. The first step in the processing is spectral analysis to derive the feature vector used to characterize the spectral properties of the speech input. For the most part, we will consider spectral vectors with 38 components consisting of 12 cepstral components, 12 delta cepstral components, 12 delta-delta cepstral components, delta log energy, and delta-delta log energy. (Systems with the first 12 and the first 24

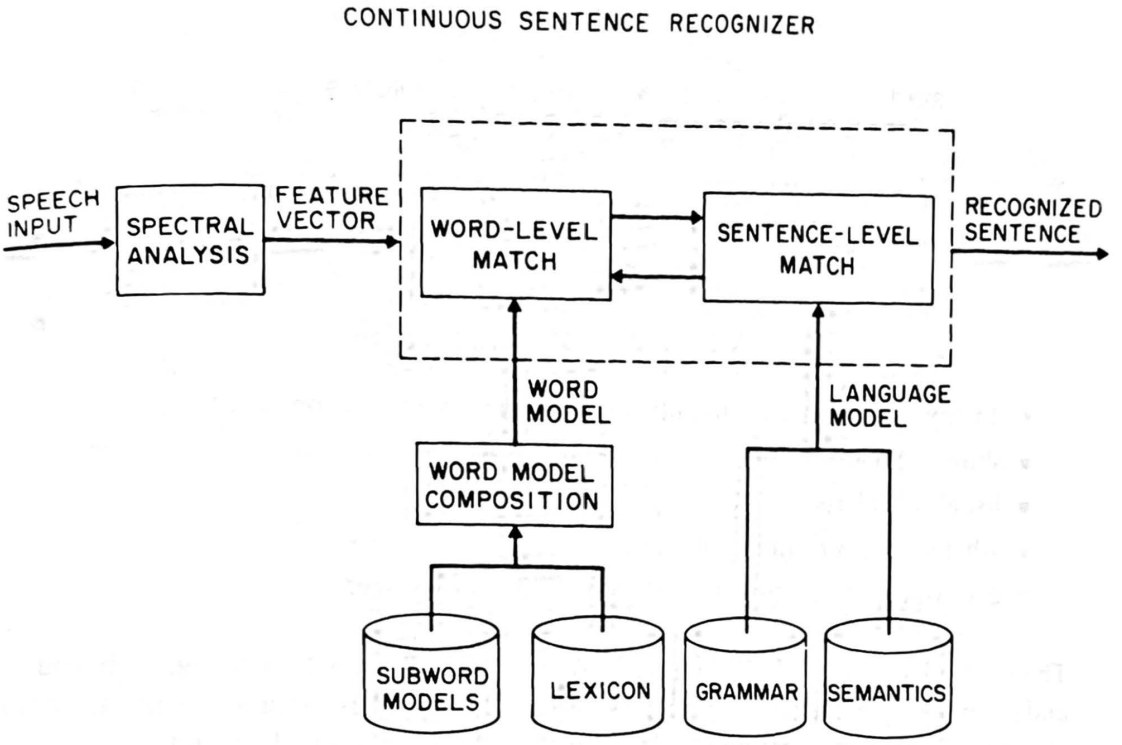


Figure 8.7 Overall block diagram of subword unit based continuous speech recognizer.

features were also studied, but results on such systems will not be presented here.)

The second step in the recognizer is a combined word-level/sentence-level match. The way this is accomplished is as follows. Using the set of subword HMMs and the word lexicon, a set of word models (HMMs) is created by concatenating each of the subword unit HMMs as specified in the word lexicon. At this point, the system is very similar to the connected word recognizers of Chapter 7. The way in which the sentence-level match is done is via an FSN realization of the word grammar (the syntax of the system) and the semantics as expressed in a composite FSN language model. The implementation of the combined word-level match/sentence-level match is via any of the structures described in Chapter 7. In particular, most systems use structures similar to the frame synchronous level-building method (usually with some type of beam search to restrict the range of paths) to solve for the "best" recognition sentence.

Consider using the recognizer of Figure 8.7 for a database management task called the Naval Resource (Battleship) Management Task—as popularly defined within the DARPA community [13]. This task, which has a 991-word vocabulary (plus a separate silence word), can be used to query a database as to locations, attributes, constraints, history, and other information about ships within the database. Typical examples of sentences used to query the database include

- what is mishawaka's percent fuel
- total the ships that will arrive in diego-garcia by next month

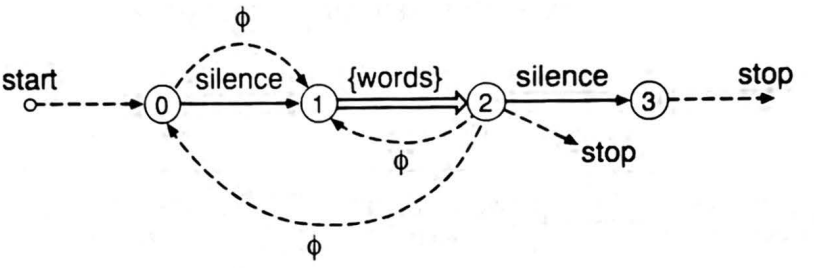


Figure 8.8 FSN for the NG syntax.

- do any vessels that are in gulf of tonkin have asw mission area of m4
- show the names of any submarines in yellow sea on twenty eight october
- list all the alerts
- what's jason's m-rating on mob
- give t-lam vessels that weren't deployed in november.

The vocabulary thus includes many jargon words, such as m4, m-rating, mob, and t-lam, and several long-content words, such as mishawaka's, diego-garcia, submarines, november, etc., and many short-function words, such as is, the, by, do, in, of, and on.

A wide range of sentences can be constructed from the 991-word vocabulary to query this database. It is possible to construct a finite-state network representation of the full grammar associated with all such sentences. The perplexity (average word branching factor) (see Section 8.7) of the full grammar network is computed to be about 9. However, such a network is rather large (because of the high degree of constraint among words within the vocabulary which form syntactically valid and semantically meaningful sentences) with upward of 50,000 arcs and 20,000 nodes, and cannot easily be implemented as a practical system. Instead, several types of FSN approximations to the full grammar have been constructed.

Perhaps the least constraining grammar (and the simplest to implement) is the no grammar (NG) case, in which any word in the vocabulary is allowed to follow any word in the vocabulary. Such an FSN has the property that, although its coverage of valid sentences is perfect, its overcoverage of the language (i.e., the ratio of sentences generated by the grammar to valid sentences within the task language) is extremely large. The perplexity of the FSN for the NG case is 991, since each word can follow every word in the grammar (assuming all words are essentially equiprobable). The FSN for the NG case is shown in Figure 8.8. (Note that the FSN of Figure 8.8 allows arbitrary phrasing, i.e., groups of words spoken together followed by a pause, because of the silence model and the null arcs.)

A second FSN form of the task syntax is to create a word pair (WP) grammar that specifies explicitly which words can follow each of the 991 words in the vocabulary. The perplexity of this grammar is about 60, and the overcoverage, while significantly below that of the NG case, is still very high. Although the network of Figure 8.8 could be used for the WP grammar (by explicitly including the word pair information at node 2), a somewhat more efficient structure exploits the fact that only a subset of the vocabulary occurs as the first word in a sentence (*B* or beginning words), and only a subset of the vocabulary occurs

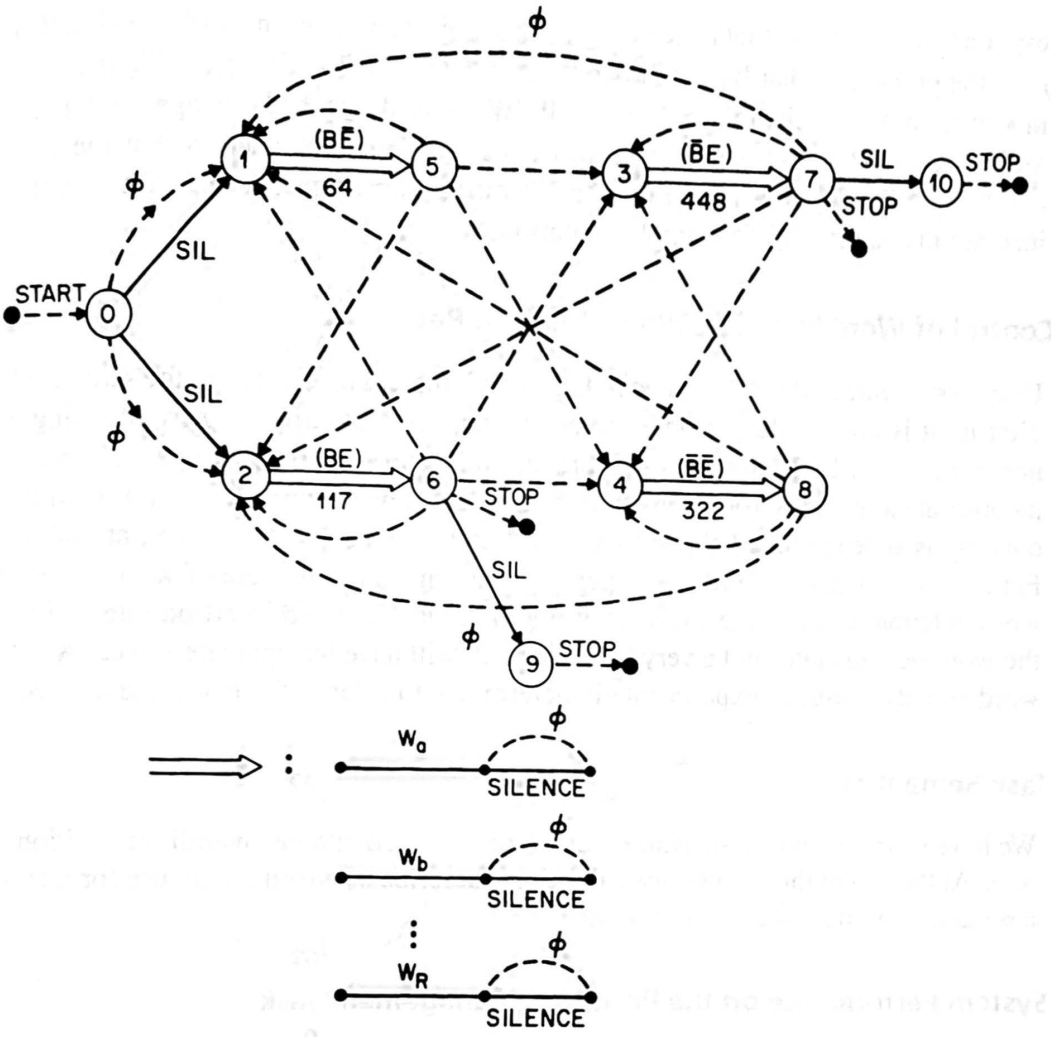


Figure 8.9 FSN of the WP syntax.

as the last word in a sentence (E or ending words); hence we can partition the vocabulary into four nonoverlapping sets of words, namely

$\{BE\}$=set of words that can either begin or end a sentence, $|BE| = 117$
$\{B\overline{E}\}$=set of words that can begin a sentence but cannot end a sentence, $|B\overline{E}| = 64$
$\{\overline{B}E\}$=set of words that cannot begin a sentence but can end a sentence, $|\overline{B}E| = 488$
$\{\overline{BE}\}$=set of words that cannot begin or end a sentence, $|\overline{BE}| = 322$.

The resulting FSN, based on this partitioning scheme, is shown in Figure 8.9. This network has 995 real arcs and 18 null arcs. To account for silence between words (which is optional), each word arc bundle (e.g., nodes 1 to 4) is expanded to individual words followed by optional silence, as shown at the bottom of Figure 8.9. Hence the overall FSN allows recognition of sentences of the form

S : (silence) $-$ $\{B\overline{E}, BE\}$ $-$ (silence) $-$ $(\{W\})\ldots(\{W\})$ $-$ (silence) $-$ $\{\overline{B}E, BE\}$ $-$ (silence).

Finally, one could construct a task syntax based on statistical word bigram (or even

trigram) probabilities—that is, we assign a probability, p_{ij}, to each word pair (W_i, W_j) where p_{ij} is the probability that W_i is followed immediately by W_j. That is, if W_n is the n^{th} word in a string of words, then $p_{ij} = P(W_n = W_j | W_{n-1} = W_i)$ is the language model according to Eq. (8.6). The advantage of the word bigram (WB) approach is that the perplexity is reduced considerably (to 20) for the Resource Management task, with essentially no increase in complexity of the implementation.

8.8.1 Control of Word Insertion/Word Deletion Rate

Using a structure of the type shown in Figure 8.9, there is no control on the sentence length. That is, it is possible to generate sentences that are arbitrarily long by inserting a large number of short-function words. To prevent this from occurring, it is a simple matter to incorporate a word insertion penalty into the Viterbi decoding, such that a fixed negative quantity is added to the likelihood score at the end of *each* word arc (i.e., at nodes 5–8 in Figure 8.9). By adjusting the word penalty, we can control the rate of word insertion and word deletion; a very large word penalty will reduce the word insertion rate and increase the word deletion rate, and a very small penalty will have the opposite effect. A value for word penalty is usually experimentally determined to balance these adverse effects.

8.8.2 Task Semantics

We have discussed how task syntax can be incorporated into the overall recognition structure. At the end of this chapter we will briefly describe a general procedure for integrating a semantic component into the recognizer.

8.8.3 System Performance on the Resource Management Task

Using the segmental k-means training algorithm, the set of 47 PLUs of Table 8.1 were trained using a set of 4360 sentences from 109 talkers. The likelihood scores were essentially unchanged after two iterations of the k-means loop. The number of mixtures per state was varied from 1 to 256 in multiples of 2 to investigate the effects of higher acoustic resolution on performance.

To evaluate the recognizer performance, five different sets of test data were used, including:

train 109	A randomly selected set of 2 sentences from each of the 109 training talkers; this set was used to evaluate the ability of the algorithm to recognize the training material
feb 89	A set of 30 sentences from each of 10 talkers, none of whom was in the training set; this set was distributed by DARPA in February of 1989 to evaluate performance
oct 89	A second set of 30 sentences from each of 10 additional talkers, none of whom was in the training set; this set was distributed by DARPA in October of 1989
jun 90	A set of 120 sentences from each of 4 new talkers, none of whom was in the training set (distributed by DARPA in June of 1990)
feb 91	A set of 30 sentences from each of 10 new talkers, none of whom was in the training set (distributed by DARPA in February of 1991).

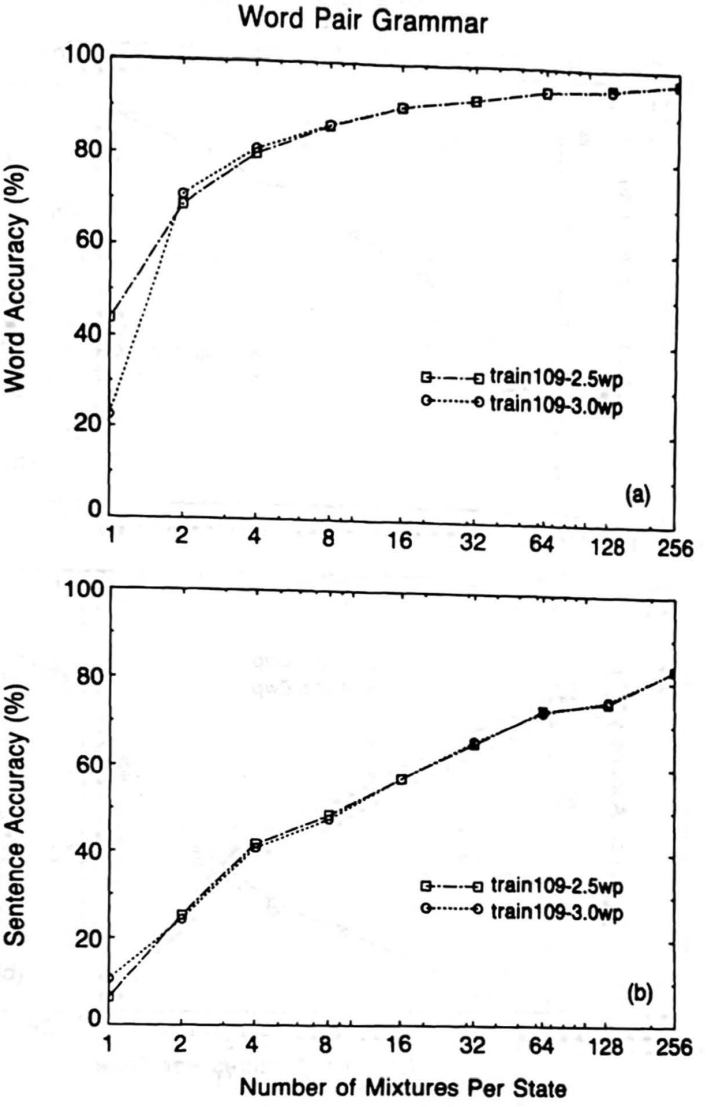


Figure 8.10 Word and sentence accuracies versus number of mixtures per state for the training subset using the WP syntax.

The recognizer performance was evaluated for each of the test sets, using both WP and NG syntax, and with different word penalties. For all cases, evaluations were made using models with from 1 to 256 mixtures per state for each PLU.

The recognition results are presented in terms of word accuracy (percentage words correct minus percentage word insertions) and sentence accuracy as a function of the number of mixtures per state for each PLU model. The alignment of the text of the recognized string with the text of the spoken string was performed using a dynamic programming alignment method as specified by DARPA.

The recognition results on the training subset (train 109) are given in Figures 8.10 (for the WP syntax) and 8.11 (for the NG syntax). The upper curves show word accuracy (in percentage) versus number of mixtures per state (on a logarithmic scale) for two different values of the word penalty, and the lower curves show sentence accuracy for the same

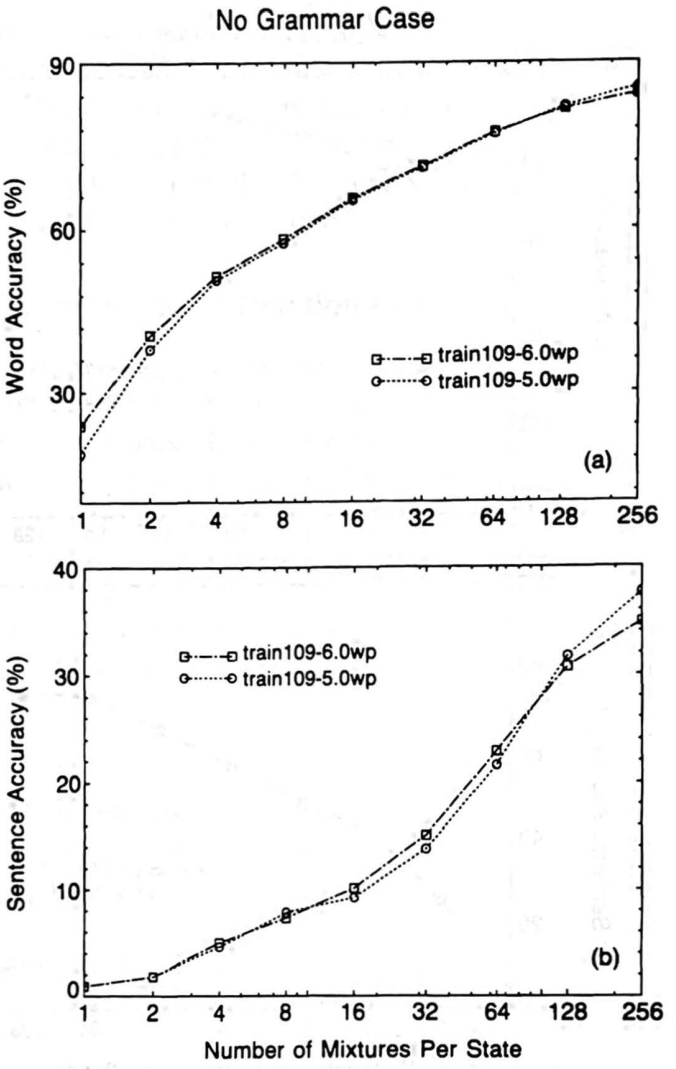


Figure 8.11 Word and sentence accuracies versus number of mixtures per state for the training subset using the NG syntax.

parameters. A sharp and steady increase in accuracy is obtained as the number of mixtures per state increases, going from about 43.6% word accuracy (10.6% sentence accuracy) for 1 mixture per state to 97.3% word accuracy (83% sentence accuracy) for 256 mixtures per state for the WP syntax using a word penalty of 2.5. For the NG syntax (using a word penalty of 6.0), the comparable results were 24% word accuracy (0.9% sentence accuracy) for 1 mixture per state and 84.2% word accuracy (34.9% sentence accuracy) for 256 mixtures per state.

The recognition results on the independent test sets are given in Figures 8.12 (for the WP syntax) and 8.13 (for the NG syntax). Although there are detailed differences in performance among the different test sets (especially for small numbers of mixtures per state), the performance trends are essentially the same for all the test sets. In particular we see that for the WP syntax, the range of word accuracies for 1 mixture per state is 42.9%

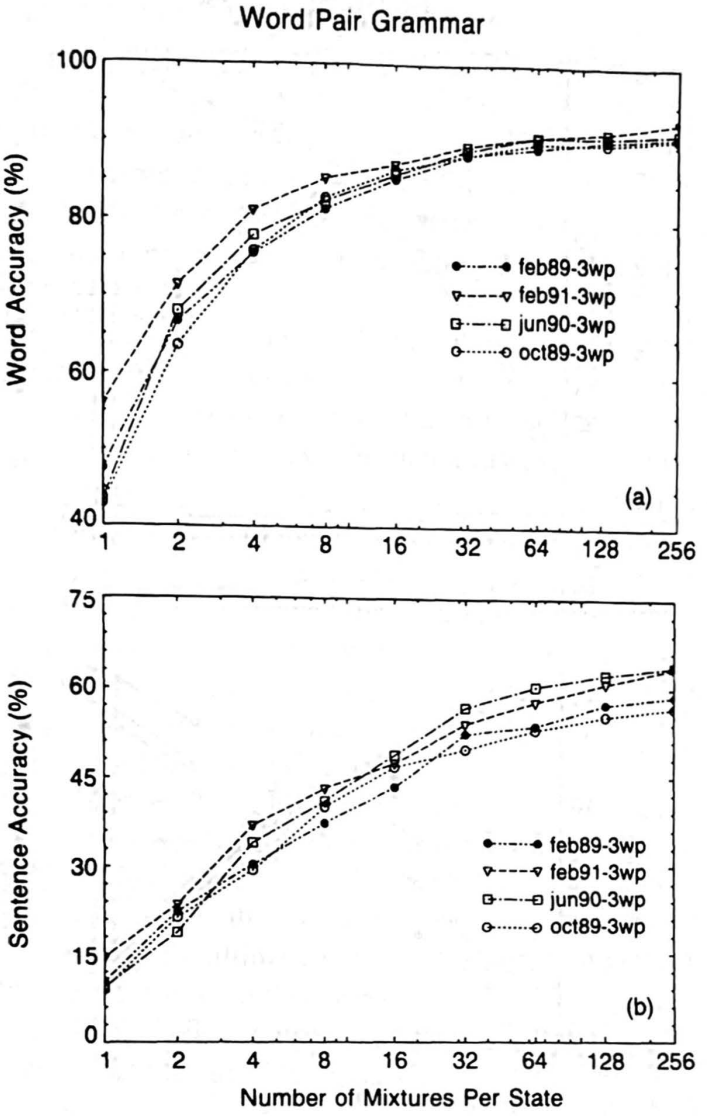


Figure 8.12 Word and sentence accuracies versus number of mixtures per state for the four test sets using the WP syntax.

(for feb 89) to 56.0% (for jun 90), whereas for 256 mixtures per state the range is 90.9% (for feb 89) to 93.0% (for jun 90). For the NG syntax, the range of word accuracies for 1 mixture per state is 20.1% (for feb 91) to 28.5% (for jun 90) and for 256 mixtures per state it is 68.5% (for oct 89) to 70.0% (for feb 91).

Perhaps the most significant aspect of the performance is the difference in accuracies between the test sets and the training subset. Thus there is a gap of 4–7% in word accuracy for the WP syntax at 256 mixtures per state, and a gap of 14.2–15.7% for the NG syntax at 256 mixtures per state. Such gaps are indicative of the ability of the training procedure to overtrain (learn details) on the training set, thereby achieving significantly higher recognition accuracy on this set than on any other representative test set.

The results presented in this section show that a simple set of context-independent PLUs can be trained for a continuous speech large vocabulary recognition task, using

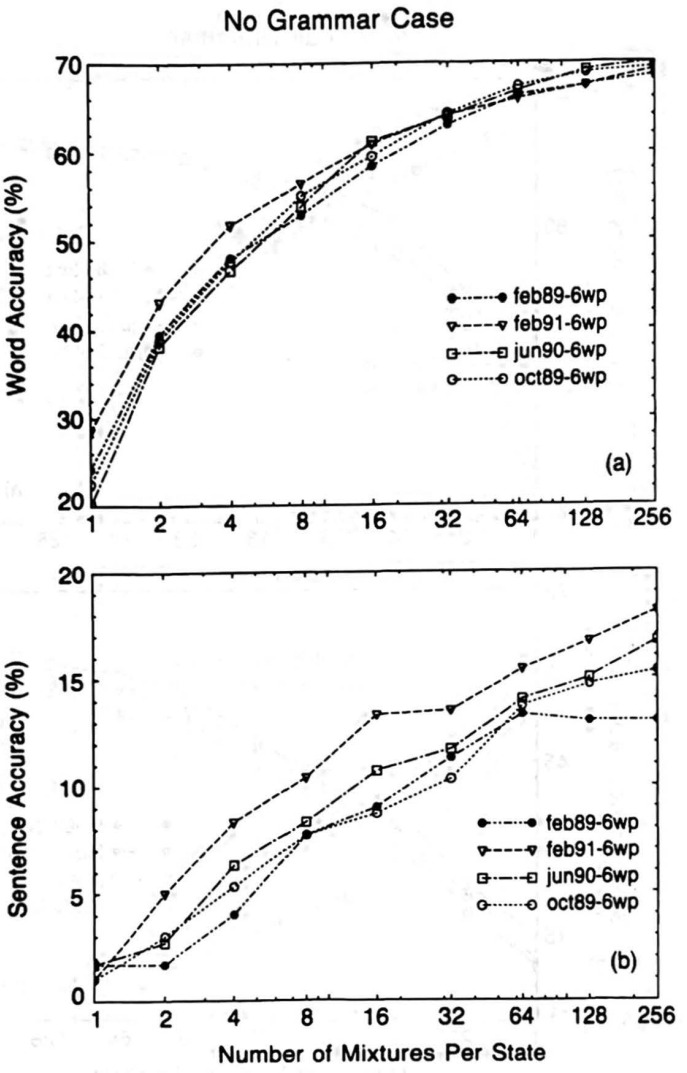


Figure 8.13 Word and sentence accuracies versus number of mixtures per state for the four test sets using the NG syntax.

standard Viterbi training procedures, and be used to provide reasonably good recognition accuracy for a moderately complex task. The key issue now is what can be done, in meaningful ways to improve recognizer performance. To answer this question, we will examine several possible extensions of the basic recognition system in the next few sections.

8.9 CONTEXT-DEPENDENT SUBWORD UNITS

There are several advantages to using a small basic set of context-independent subword units for large vocabulary speech recognition. First of all we have shown that the models of these subword units are easily trained from fluent speech, with essentially no human decisions as to segmentation and labeling of individual sections of speech. Second, the

resulting units are generalizable to new contexts (word vocabularies, tasks with different syntax and semantics) with no extra effort. Finally, the resulting models are relatively insensitive to the details of the context from which the training tokens are extracted. By this we mean that, in theory, we can derive the subword unit model parameters from two arbitrary but sufficiently large training sets of fluent speech (hopefully of the same size and general linguistic content but not necessarily the same vocabulary words and sentences) to obtain essentially the same parameter estimates for each model. In practice, this is almost the case.

However, there are situations in which the subword unit model parameters are extracted from a training set whose linguistic content matches the test set precisely—that is, when the training set is a set of sentences drawn from the recognition task (with the same vocabulary, syntax, and semantics). In such a case, the resulting subword units are somewhat "word sensitive" (showing higher likelihood scores than in the general case) and typically provide higher recognition performance than equivalent model sets derived from arbitrary input speech. In particular, for the Resource Management task discussed in the previous section, "word-sensitive" subword unit models, trained on task-specific training sentences, give about 10% higher word recognition accuracy than the same set of subword unit models trained on arbitrary sentences of comparable size. If the training set is increased in size by a factor of about 3, the word accuracy of the text-independent models approaches that of the word-sensitive models.

Obviously, this performance difference can be attributed to the fact that context-independent subword unit models are not adequate in representing the spectral and temporal properties of the speech unit in all contexts. (By context we mean the effects of the preceding and following sounds as well as the sound stress and information, and even the word in which the sound occurs.) The ultimate effect is a decrease in performance in word and sentence accuracy on speech-recognition tasks.

The solution to this problem is basically a simple, straightforward one—namely, to extend the set of subword units to include context-dependent units (either in addition to or as a replacement for context-independent units) in the recognition system. In theory, the only change necessary in either training or recognition is to modify the word lexicon to be consistent with the final set of subword units. Consider the word "above." Based on using (1) context-independent units, (2) triphone (left and right context) units, (3) multiple-phone models, and (4) word-dependent units, we could have the following lexical representations:

(1) above:	ax	b	ah	v	Context-Independent Units
(2) above:	$-ax-b	ax-b-ah	b-ah-v	ah-v-$	Triphones (Context Dependent)
(3) above:	ax2	b2	ah1	v1	Multiple Phone Units
(4) above:	ax (above)	b (above)	ah (above)	v (above)	Word-Dependent Units.

In representation (2), using triphone units, the number of units needed for all sounds in all words is very large (on the order of 10–20,000). In practice, only a small percentage of such triphone units are used, since most units are seen rarely, if at all, in a finite training set. (We discuss this issue below in more detail.) In representation (3), using multiple models of each subword unit, the idea is to cluster common contexts together so as to reduce the

number of context-dependent models. This leads to problems in defining lexical entries for words. (We discuss this issue further in a later section of this chapter.) Finally, the use of word-dependent units is most effective for modeling short-function words (like a, the, in, of, an, and, or) whose spectral variability is significantly greater than that of long-content words like aboard and battleship. (We discuss the modeling of function words in a later section of this chapter.) Finally, it is both reasonable and meaningful to combine all four types of units in a common structure. In theory, as well as in practice, the training and recognition architectures can handle subword unit sets of arbitrary size and complexity. We now discuss each of these issues in more detail.

8.9.1 Creation of Context-Dependent Diphones and Triphones

Consider the basic set of context-independent PLUs in which we use the symbol p to denote an arbitrary PLU. We can define a set of context-dependent (CD) diphones as

$$\begin{array}{ll} p_L - p - \$ & \text{left context (LC) diphone} \\ \$ - p - p_R & \text{right context (RC) diphone,} \end{array}$$

in which p_L is the PLU immediately preceding p (the left context sound), p_R is the PLU immediately following p (the right context sound), and \$ denotes a don't care (or don't know) condition.

Similarly we can define a set of context-dependent triphones as

$$p_L - p - p_R \qquad \text{left-right context (LRC) triphone.}$$

In theory, the potential number of left (or right) context diphones is 46×45 (for a basic set of 47 PLUs and excluding silence) or about 2070 left context diphone units. The potential number of left-right context triphone units is $45 \times 46 \times 45$ or 93,150 units. In practice, the actual number of context-dependent PLUs actually seen in a finite training set of sentences is significantly smaller than these upper bounds.

To better understand these concepts, consider the RM task (991 word vocabulary) with a training set of 3990 sentences. To use diphone and triphone context-dependent units, we first convert the lexicon to such units using the rule that the initial sound becomes a right context diphone, the middle sounds become left-right context diphones, and the final sound becomes a left context diphone. Hence the word "above" is converted to the set of units \$–ax–b, ax–b–ah, b–ah–v, ah–v–\$. (We must use diphone units at the beginnings and ends of words because we do not know the preceding or following words.) The above rule is modified to eliminate triphone middles for words with only two PLUs (e.g., in, or) and to revert to the context-independent PLU for words with only one PLU (e.g., a). Using the above method of creating the lexicon, one can count the number of left-right context-dependent (LRC) units (1778), the number of left-context (LC) units (279), the number of right-context (RC) units (280), and the number of context-independent (CI) units (3) for a total of 2340 PLUs in the training set. This number of units, although significantly smaller than the maximum possible number of context-dependent units, is deceiving because many of the units occur only a small number of times in the training set, and therefore it would be difficult to reliably estimate model parameters for such models.

TABLE 8.4. Number of intra-word CD units as a function of count threshold, T.

Count Threshold (T)	Number of LRC PLUs	Number of LC PLUs	Number of RC PLUs	Number of CI PLUs	Total Number of PLUs
50	378	158	171	47	754
40	461	172	188	47	868
30	639	199	205	47	1090
20	952	212	234	46	1444
10	1302	243	258	44	1847
5	1608	265	270	32	2175
1	1778	279	280	3	2340

To combat the difficulties due to the small number of occurrences of some context-dependent units, one can use one of three strategies. Perhaps the simplest approach is to eliminate all models that don't occur sufficiently often in the training set. More formally we define $c(\cdot)$ as the occurrence count for a given unit. Then, given a threshold T on the required number of occurrences of a unit (for reliable model estimation), a reasonable Unit Reduction Rule is

If $c(p_L - p - p_R) < T$, then

1. $p_L - p - p_R \rightarrow \$ - p - p_R$ if $c(\$ - p - p_R) \geq T$
2. $p_L - p - p_R \rightarrow p_L - p - \$$ if $c(p_L - p - \$) > T$
3. $p_L - p - p_R \rightarrow \$ - p - \$$ otherwise.

The tests above are made sequentially until one passes and the procedure terminates. To illustrate the sensitivity of the CD PLU set to the threshold T, Table 8.4 shows the counts of LRC PLUs, LC PLUs, RC PLUs, CI PLUs, and the total PLU count for the 3990 sentence training set. For a threshold of 50, which is generally adequate for estimating model parameters, there are only 378 LRC PLUs (almost a 5-to-1 reduction over the number with a count threshold of 1) and a total of 754 PLUs. We will see later that although such CD PLU sets do provide improvements in recognition performance over CI PLU sets, the amount of context dependency achieved is small and alternative techniques are required to create CD PLU sets.

8.9.2 Using Interword Training to Create CD Units

Although the lexical entry for each word uses right or left context diphone units for the first and last sound of each word, both in training and in scoring, one can utilize the known (or postulated) sequence of words to replace these diphone units with the triphone unit appropriate to the words actually (or assumed) spoken. Hence the sentence "Show all ships" would be represented as

$$\$\text{–sh–ow}\quad \text{sh–ow–}\$\quad \$\text{–aw–}\ell\quad \text{aw–}\ell\text{–}\$\quad \$\text{–sh–i}\quad \text{sh–i–p}\quad \text{i–p–s}\quad \text{p–s–}\$$$

using only intraword units, whereas the sentence would be represented as

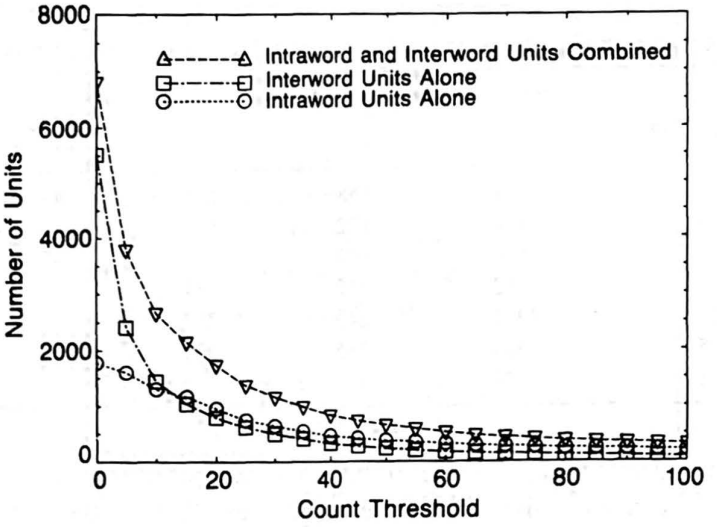


Figure 8.14 Plots of the number of intraword units, interword units, and combined units as a function of the count threshold.

$–sh–ow sh–ow–aw ow–aw–ℓ aw–ℓ–sh ℓ–sh–i sh–i–p i–p–s p–s–$

using both intraword and interword units. From this simple example we see that, whereas there were only two triphones based on intraword units, there are six triphones based on intraword and interword units—that is, a threefold increase in context-dependent triphone units. (We are assuming no silence between words; it is straightforward to handle the cases when silence actually occurs between words.) To illustrate this effect, Figure 8.14 shows a plot of the number of intraword units, the number of interword units, and the combined count, as a function of the count threshold, for the 1990 sentence DARPA training set. More than 5000 interword triphone units occur one or more times versus less than 2000 intraword units for the same count threshold.

Even when using interword units, the problems associated with estimating model parameters from a small number of occurrences of the units is the major issue. In the next sections we discuss various ways of smoothing and interpolating context dependent models, created from small numbers of occurrences in the training set, with context-independent models, created from large numbers of occurrences in the training set.

8.9.3 Smoothing and Interpolation of CD PLU Models

As shown above, we are faced with the following problem. For a training set of reasonable size, there is sufficient data to reliably train context-independent unit models. However, as the number of units becomes larger (by including more context dependencies) the amount of data available for each unit decreases and the model estimates become less reliable. Although there is no ideal solution to this problem (short of increasing the amount of training data ad infinitum), a reasonable compromise is to exploit the reliability of the estimates of the higher level (e.g., CI) unit models to smooth or interpolate the estimates of the lower level (CD) unit models. There are many ways in which such smoothing or

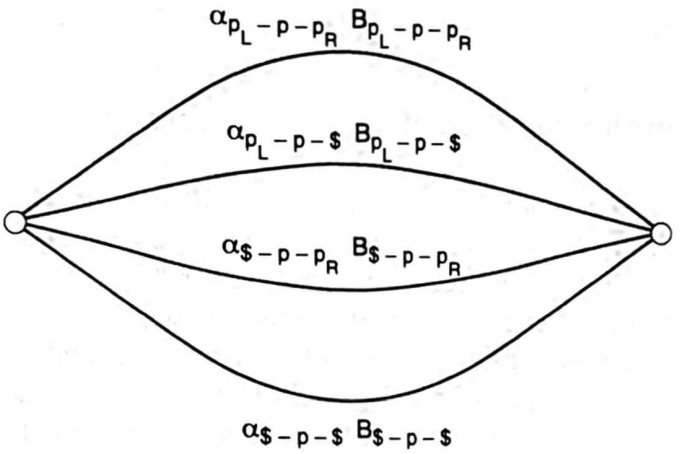


Figure 8.15 Deleted interpolation model for smoothing discrete density models.

interpolation can be achieved.

The simplest way to smooth the parameter estimates for the CD models is to interpolate the spectral parameters with all higher (less context dependency) models that are consistent with the model [12]. By this we mean that the model for the CD unit $p_L - p - p_R$ (call this $\lambda_{p_L-p-p_R}$) should be interpolated with the models for the units $\$-p-p_R$ ($\lambda_{\$-p-p_R}$), $p_L - p - \$$ ($\lambda_{p_L-p-\$}$) and $\$ - p - \$$ ($\lambda_{\$-p-\$}$). Such an interpolation of model parameters is meaningful only for discrete densities, within states of the HMM, based on a common codebook. Thus if each model λ is of the form (A, B, π) where B is a discrete density over a common codebook, then we can formulate the interpolation as:

$$\begin{aligned}\hat{B}_{p_L-p-p_R} = {} & \alpha_{p_L-p-p_R} B_{p_L-p-p_R} + \alpha_{p_L-p-\$} B_{p_L-p-\$} \\ & + \alpha_{\$-p-p_R} B_{\$-p-p_R} + \alpha_{\$-p-\$} B_{\$-p-\$},\end{aligned} \tag{8.19}$$

where $\hat{B}_{p_L-p-p_R}$ is the interpolated density. We constrain the αs to add up to 1; hence

$$\alpha_{p_L-p-p_R} + \alpha_{p_L-p-\$} + \alpha_{\$-p-p_R} + \alpha_{\$-p-\$} = 1. \tag{8.20}$$

The way in which the αs are determined is according to the deleted interpolation algorithm discussed in Section 6.13. We review the ideas, as they apply to these speech unit models, here. Each of the discrete densities, $B_{p_L-p-p_R}$, $B_{p_L-p-\$}$, $B_{\$-p-p_R}$, and $B_{\$-p-\$}$, is estimated from the training data where a small percentage (e.g., 20%) is withheld (deleted). Using the withheld data, the αs are estimated using a standard forward-backward approach based on the HMM shown in Figure 8.15. The interpretation of the αs is essentially the probability weighted percentage of new data (unseen in training) that favors each of the distributions over the others. Hence, for well-trained detailed models we get $\alpha_{p_L-p-p_R} \to 1$, whereas for poorly trained models we get $\alpha_{p_L-p-p_R} \to 0$ (i.e., the LRC model is essentially obtained from interpolating higher-level, lower context dependency models that are better trained than the detailed CD model).

Other smoothing methods include empirical estimates of the αs based on occurrence counts, co-occurrence smoothing based on joint probabilities of pairs of codebook symbols [14], and use of fuzzy VQs in which an input spectral vector is coded into two or more

codebook symbols.

8.9.4 Smoothing and Interpolation of Continuous Densities

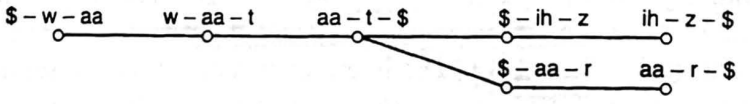
When one uses continuous density modeling of PLUs it is very difficult to devise a good smoothing or interpolation algorithm because the acoustic space of different units is inherently different. There are two reasonable ways to handle this problem. One is to exploit the so called semicontinuous or tied mixture modeling approach discussed earlier in which each PLU uses a fixed set (a codebook) of mixture means and variances, and the only variables are the mixture gains for each model. In this case it is trivial to exploit the method of deleted interpolation on the mixture gains in a manner virtually identical to the one discussed in the previous section.

An alternative modeling approach, and one more in line with independent continuous density modeling of different sounds, is to use a tied-mixture approach on the CI unit level; that is, we design a separate (large) codebook of densities for each CI PLU and then constrain each derived CD unit to use the same mixture means and variances but with independent mixture gains. Again we can use the method of deleted interpolation to smooth mixture gains in an optimal manner.

8.9.5 Implementation Issues Using CD Units

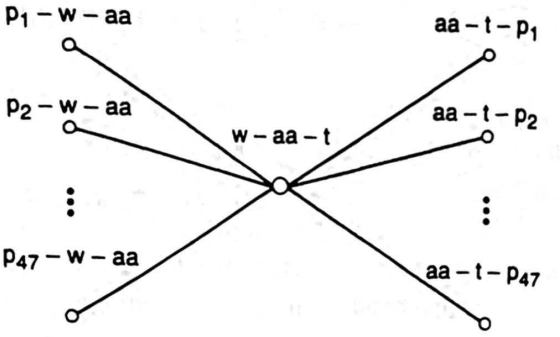
The FSN structure of Figure 8.9 is used to implement the continuous speech-recognition algorithm based on a given vocabulary and task syntax ([15–19]). The structure is straightforward to implement when using strictly intraword units because there is no effect of context at word boundaries. Hence the models (HMMs) for each word can be constructed independently and concatenated at the appropriate point of the processing. This is illustrated below for the recognition of the string "what {is, are}" based on intraword units, where the individual words are represented in the lexicon as

$$\begin{aligned} \text{what} &- \{\$\text{–w–aa},\ \text{w–aa–t},\ \text{aa–t–}\$\} \\ \text{is} &- \{\$\text{–ih–z},\ \text{ih–z–}\$\} \\ \text{are} &- \{\$\text{–aa–r},\ \text{aa–r–}\$\} \end{aligned}$$

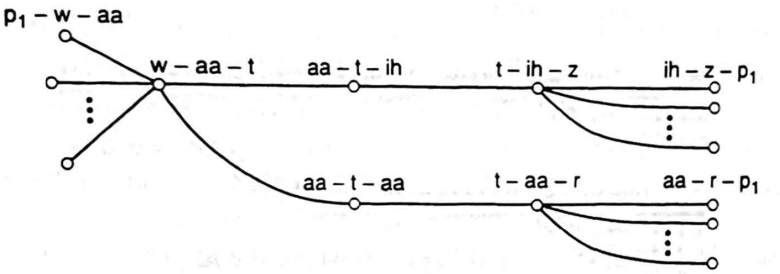


(If we allow silence between words there is a trivial modification to include a silence node after the word /what/.) When we include interword units in the recognition stage the FSN becomes considerably more complicated because the first unit of each word (which we call the head unit) is variable depending on the last unit of each possible preceding word; similarly, the last unit of each word (which we call the tail unit) is variable depending on the first unit of each possible following word. (We call the set of units between the head unit and the tail unit, the body units.) Thus, in theory, a word like "what" consists of (up

to) 47 head units and (up to) 47 tail units, and would be represented as



In practice many, if not most, of the head as well as tail units don't exist; hence, the structure is generally considerably less complex. Thus the FSN network of the strings "what {is, are}" becomes



that is, a considerably more complex network results. (Interestingly, the inclusion of silence adds only a single extra path to each branch of the network.) The bookkeeping associated with such networks can easily get out of hand and dominate the overall computation. Fortunately, several network architectures have been devised for efficiently handling the bookkeeping associated with such networks [15]. Interesting special cases occur when the number of units within a word falls below three. When there are exactly two units in a word, there are no body units so the variable head units merge with the variable tail units. When there is only a single unit within a word, there are no body units nor is there a tail unit. Effectively, the bookkeeping must look at both the preceding word set of tail units and the following word set of head units to handle this case. The three cases described above—namely, implementations of words with three or more units, words with two units, and words with one unit—are illustrated in Figure 8.16.

8.9.5.1 Word Junction Effects

The assumption that is made when training interword units is that in continuous speech, words are pronounced similar to the way they are pronounced in isolation. In most cases this assumption is reasonable in that the coarticulation phenomena at word boundaries only lead to small (soft) changes in the word pronunciation and therefore can readily be modeled by interword units based on the concatenation of the tail unit from one word with the appropriate head unit from the following word. However, in some cases, the pronunciation changes are radical (hard) changes in which a boundary sound (tail or head)

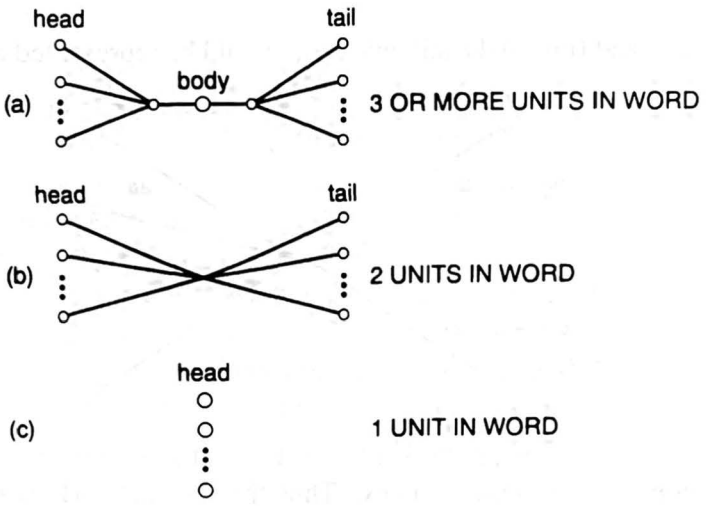


Figure 8.16 FSN representation of words with three or more units (a), two units (b), and one unit (c).

is completely deleted or replaced with a totally different sound. Examples of such hard changes include the strings "what time" and "did you," among others. In the string "what time," the double stop consonant (t followed by t) between words is replaced with a single occurrence of t; hence one of the *t*s is deleted. In the string "did you," the standard phonetic transcription would be /d ih d y uw/; however, in continuous speech the actual transcription would be /d ih j h uw/ (or even /d ih jh ax/) where the /d y/ boundary phones are changed to the single sound /jh/.

Phonologically predictable changes of the type shown above (the so-called hard sound changes) cannot easily be learned from the training procedure because they occur infrequently and they lead to radically different sounds than would be predicted from the concatenation of boundary sounds of the relevant words as spoken in isolation. To handle these hard changes correctly, a set of phonological rules has to be superimposed on both the training and recognition networks (in a relatively straightforward (brute force) manner). There are about 11 such rules that handle most of the known phonological changes in English [16].

Some typical phonological rules include the following:

Rule 1 Geminate deletion. If a word ends in a consonant and the following word begins with the same consonant, the ending consonant is deleted; e.g., the final /t/ is deleted in the pair "what time."

Rule 2 Palatization. If a word ends in a /d/, and the following word begins with a /y/, then (optionally), the final sound can be converted to a /jh/, and the initial sound of the following word is deleted. Thus the words "did you" can be spoken as either /d ih d y uw/ or /d ih jh uw/.

Rule 3 Plosive deletion. If a word ends in the nasal /n/ followed by a plosive sound, and the following word begins in a plosive sound, then the final plosive in the initial word is deleted. Thus the words "went down" can be spoken as /w eh n d aw n/.

The complete set of rules is available in Reference [15].

8.9.5.2 Variance Estimation Problems

Perhaps the most difficult problem, in training, when using continuous density HMMs, is the estimation of mixture variances when the amount of training data is small (as is almost always the case when using context-dependent units). The problem is that to maximize the likelihood on the training data, the estimation procedure often tries to make the variance very small (i.e., choosing data samples that are very close to each other in value). Although this leads to good training likelihood scores, it often provides poor matches to independent test data. Hence, some protection against variances getting too small in training is required.

Several proposals have been made as to how to realistically control the variance of the estimates to prevent such effects from occurring. One simple one is to tie variances across units, states, and even words, ultimately leading to a grand variance for each spectral component that is independent of the unit, state, and word. This idea is reasonable and has been shown to work well in practice [20]. An alternative is to set a floor on the variance of each spectral component that is based on a statistical analysis of the range of values of the variance component for different sounds, states, etc. Thus rather than using a grand variance, the concept of setting a variance clipping threshold at an appropriate point of the distribution (e.g., 2 sigma below the mean) preserves the (reasonable) range of variance estimates while at the same time preventing the variance from getting unreasonably small. (One could also argue that a high clipping threshold would prevent the variance from getting unreasonably large; in practice, the process of maximizing likelihood prevents this from happening. See Section 6.5 for a more complete discussion of these concepts.)

8.9.6 Recognition Results Using CD Units

A key issue in continuous speech recognition is the total number of subword units used in the system. We have already discussed several different types of subword units, including context-independent units, intraword context-dependent units, interword context-dependent units, and various combinations of these. In later sections of this chapter we will extend the unit classes to include position-dependent units, function word-dependent units, and even function phase-dependent units.

On the one hand, it seems clear that as we add more units with greater context dependency, the performance of the recognition should continue to improve. On the other hand, for a fixed training set, the amount of training data available for estimating model parameters of context-dependent units becomes smaller as the number of units increases. Hence the reliability of the estimates of model parameters decreases and therefore recognition performance falls. (To combat this second effort, various smoothing and interpolation procedures have been devised.) The overall result is that recognition performance is maximized for a finite size subword unit set whose size depends on the training data, the recognition vocabulary, the task syntax, and the method for creating the context-dependent units. In this section we present several results illustrating this trade-off between number of subword units and overall word accuracy of the recognizer.

TABLE 8.5. Word error rates as a function of occurrence threshold for the feb 89 test set using intraword units with a 38 component/vector analysis.

Threshold	∞	30[a]	25[a]	20[a]	15[a]	10[a]
Number of Units	47	1090	1215	1444	1694	1874
WP Word Error (%)	14.0	6.7	7.0	7.1	7.4	7.6
NG Word Error (%)	40.0	25.0	24.8	25.0	25.2	25.6

[a]Sixteen mixtures per state were used for these model sets.

TABLE 8.6. Word error rates as a function of occurrence threshold for the feb 89 test set using both intraword and interword units (independently) with a 38 component/vector analysis.

Threshold	∞	30[a]	25[a]	20[a]	15[a]	10[a]
Number of Units	47	1769	2125	2534	2985	3863
WP Word Error (%)	9.1	4.6	4.7	4.6	4.7	5.3
NG Word Error (%)	32.7	20.8	19.8	19.4	20.6	20.9

[a]Sixteen mixtures per state were used for these model sets.

For evaluating speech-recognition performance, we use the feb 89 test set of 300 sentences spoken by 10 adult male and female talkers (30 sentences per talker). Using, as a baseline system, the recognizer based on the 47 context-independent units with 256 mixtures per state, the unit reduction rule was used at several thresholds to generate unit sets with up to 1874 intraword units (no interword units were used here), and the tests were done using cepstral plus differential cepstral (delta and delta-delta) with differential energy (first and second order) parameters (38/vector) with both the word pair (WP) and no grammar (NG) syntaxes. The word recognition accuracies for these systems are given in Table 8.5. A significant improvement in performance is achieved when adding context-dependent intraword units (e.g., from 14% to 9.2% error rate for the WP case); however, increasing the total number of units from 638 to 915 or 1759 or 2340 does not reduce word error rate but instead increases it slightly. This is the tradeoff referred to above. For the NG syntax a similar trend is observed, although the recognizer performance is relatively flat for a large range of units.

The results when using both intraword and interword units (independently, see next section) are shown in Table 8.6. For this test, we again used the feb 89 test set; however, we used the full 38 component/vector analysis frame (including delta-delta cepstral values, delta energy, and delta-delta energy). The effects of the enhanced analysis frame are seen in the improved performance of the 47 PLU set where the error rate falls from 14% to 9.1% for the WP syntax, and from 40% to 32.7% for the NG syntax. Similarly the use of interword units (along with the enhanced analysis) reduced the error rate to 4.6% for the WP case (using 1769 units) and to 19.4% for the NG case (using 2534 units). Again

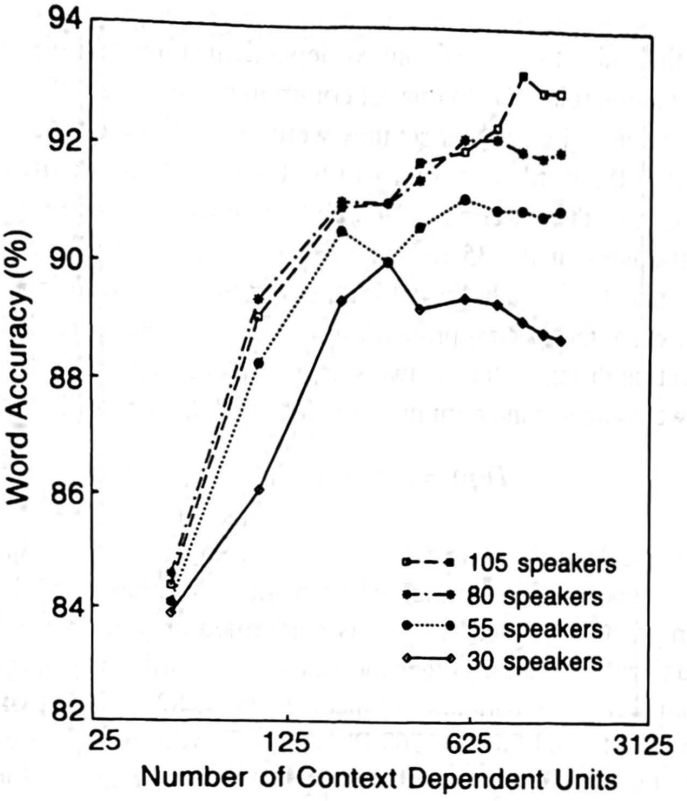


Figure 8.17 Word accuracy (%) as a function of the number of generalized triphone models for several training set sizes (after Lee [2]).

we clearly see the saturation in performance as the number of units increases due to the problems in reliably estimating parameters of the context-dependent models from a finite training set.

To illustrate the effects of training set size on recognition performance even more dramatically, Figure 8.17 (due to K. F. Lee) shows a plot of word accuracy versus number of units (called generalized triphone models in Lee's notation) for different size training sets (measured in terms of the number of speakers in the set). For the smallest training set (30 speakers), the word accuracy peaks at around 300 units and then falls dramatically beyond this point. For the 55-speaker training set the word accuracy peaks at around 625 units and then falls slightly. For the 80 and 105 speaker sets the performance peaks at about 1000 and 2000 units. These results dramatically illustrate the difficulties in creating subword unit sets with a large number of context-dependent units.

8.9.7 Position Dependent Units

When using both intraword and interword units, it is natural and reasonable to combine occurrences of the same unit independent of whether they occurred within the word or across words. It has been observed that phones within words are significantly more stable, acoustically, than phones occurring at word boundaries. Thus it seems plausible that the spectral behavior of the same intraword and interword unit could be considerably different.

To illustrate this point, two sets of context-dependent subword units were created using the same unit reduction rule [21]. In one set common occurrences of intraword and interword units were combined; in the other set they were modeled independently according to their positions within the words or across words (thus the name position dependent). Using a threshold of 30, there were 1282 combined units, including 1101 left-right context units, 99 left-context units, 35 right-context units, and 47 CI units, and 1769 separate position-dependent units including 913 intraword units and 856 interword units.

To show that the spectral properties of these two sets were different, the histograms of unit separation distances for the two sets were computed as follows. For each unit, λ_p, in each set, we computed the minimum distance (likelihood separation) as

$$D(p) = \min_{q \neq p} \{L(Y_p|\lambda_p) - L(Y_p|\lambda_q)\}$$

where Y_p represents the training data segments used to estimate λ_p. $D(p)$ represents the smallest likelihood score difference when using any other model than the one created from Y_p. (In practice, the computation is performed only for models, λ_q, which had the same base unit as λ_p since all other models gave significantly larger difference scores.) The histograms of unit separation distance for the 1282 PLU set of combined intraword and interword units, and for the 1769 PLU set of position dependent units are shown in Figure 8.18. For the 1769 PLU set almost *all* of the unit separation distances are larger than 2.0 (difference in log likelihoods), including the cases where the same unit occurred in both intraword and interword contexts. The average unit separation distance for this set is about 9.0. For the 1282 PLU set the histogram is skewed to the left, showing many small unit separations, with an average distance on the order of 4–5. The results clearly show that the spectral properties of context-dependent units are often significantly different within words than when they occur at word boundaries.

8.9.8 Unit Splitting and Clustering

We have shown in previous sections that it is relatively simple to train models for a small set of context-independent units from a training set of a reasonable size. The problem is that the recognizer performance is not good enough for large vocabulary continuous speech-recognition tasks. We also discussed one simple way to train models for a large set of context-dependent units from the same training set. Here the problem is the inadequacy of training data, which leads to poor estimates of model parameters for all but a small subset of the units observed in a typical training set. The result of the poor model estimates is that recognition performance saturates for about 1000–2000 context-dependent units and either remains constant or decreases as the number of units trained increases.

Thus a key issue in the design and implementation of large vocabulary continuous speech recognizers is how to efficiently determine the number and character of the context-dependent units that give best recognition performance for a given training set. Unfortunately, there is no simple answer to this question. In this section we discuss several proposed methods based on the concepts of either starting from a small set of context-independent units and iteratively splitting the units, or of starting from a large set

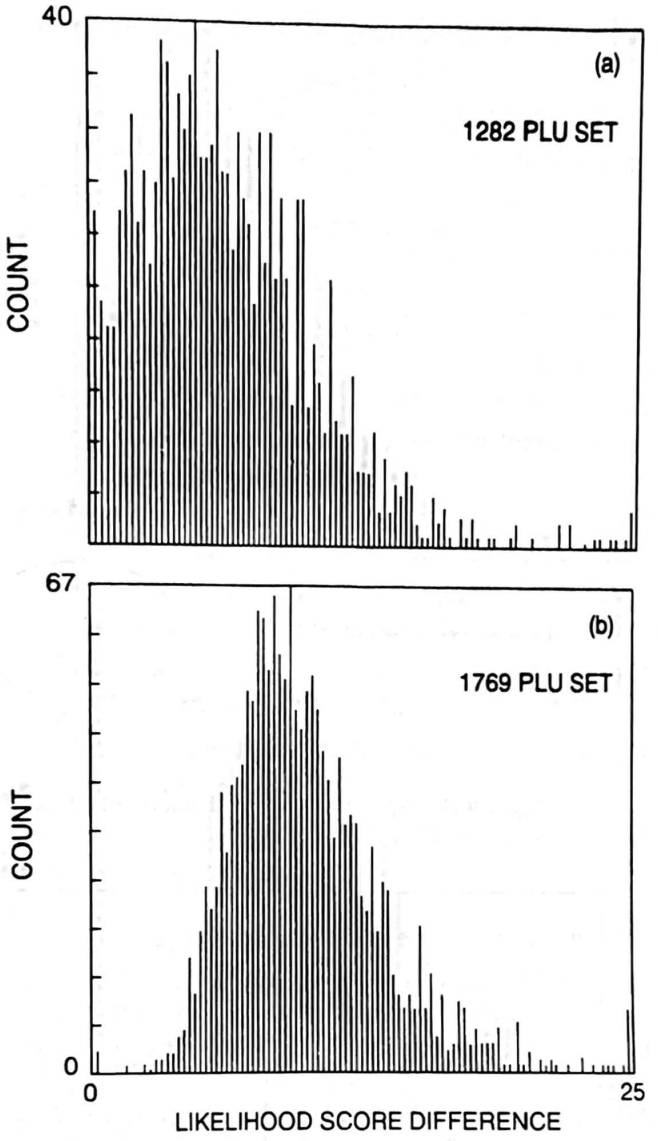


Figure 8.18 Histograms of the unit separation distance for (a) combined intraword and interword units (1282 PLUs) and (b) separate intraword and interword units (1769 PLUs) (after Lee et al. [19]).

of context-dependent units and merging similar units to reduce the number of units based on some type of clustering procedure.

8.9.8.1 Splitting of Subword Units

The basic idea of subword unit splitting is illustrated in Figure 8.19. We assume that for each subword unit p_i (with model λ_i), representing a context-independent unit, there is some inherent internal distribution of training tokens that naturally clusters into two or more clusters. (Within the figure we show three clusters, namely p_i^1, p_i^2, and p_i^3.) The clusters represent classes of sounds that are all labeled as p_i, but which have different spectral

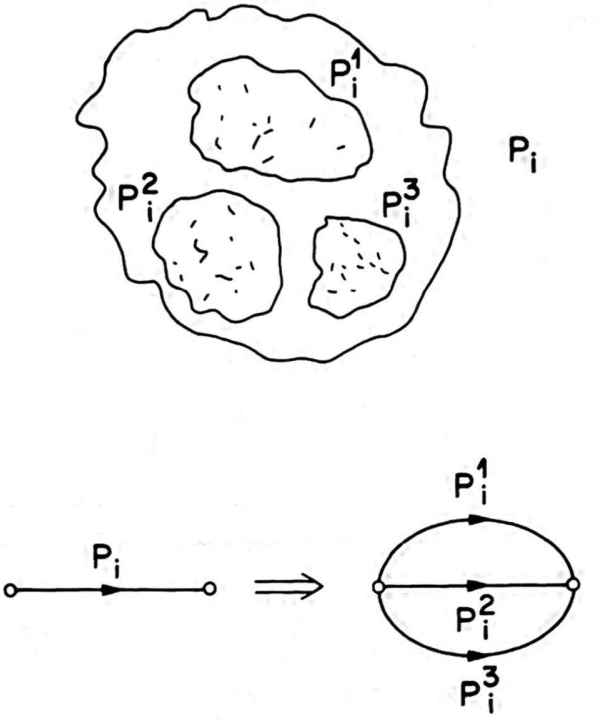


Figure 8.19 Splitting of subword unit p_i into three clusters (after Lee et al. [7]).

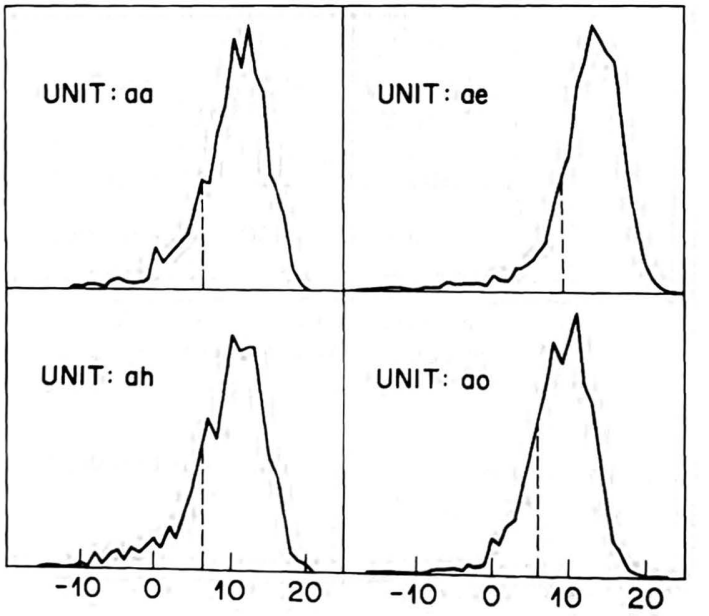


Figure 8.20 Histograms of likelihood score for four context-independent units (vowels) (after Lee et al. [7]).

properties depending on the context in which they occur. Once the separation of p_i into clusters is achieved, we have effectively created multiple models of the context-independent subword unit, as shown at the bottom of Figure 8.19.

There are several ways to create the clusters for each unit, but a particularly simple (and meaningful) one is based on the following argument. If we examine the histogram of (log) likelihood scores for each training token which is labeled p_i we get curves similar to those shown in Figure 8.20. The likelihood score histograms show that for a large percentage of the training tokens, good scores are obtained using the context-independent unit. These training tokens are (relatively) well represented by λ_i and do not need to be split off. Instead the low tail of the histogram (below the dashed lines) represents training tokens whose likelihood scores are relatively low and these tokens need an alternative representation (model) to be well represented.

Based on the above discussion, a simple procedure for splitting off training tokens with low likelihood scores and creating a new model from these tokens is as follows:

1. For each subword unit, p_i, which is to be split (not every unit need be split), all training tokens whose likelihood scores fall below a threshold are split off and used to estimate an additional model for that unit.
2. The segmental k-means training procedure is iterated on the split-off tokens until the new model reaches convergence.
3. The above procedure (steps 1 and 2) is iterated until the desired number of models, for each subword unit, is obtained.

The results (in terms of average likelihood score over the entire set of units) of applying the above splitting procedure to the 47 PLU set of context-independent units is shown in Figure 8.21. The results are shown, as a function of iteration number, for splitting each of the 47 models into 2, 3, and 4 models. The procedure converges rapidly and provides small but consistent increases in average likelihood scores.

The above model splitting procedure leads to one major difficulty, namely, How do we modify the word pronunciation dictionary to account for the presence of multiple versions of each subword unit? The inherent problem is illustrated in Figure 8.22, which shows the networks for a complete set of word models assuming every version of each sound in the word can follow every version of every other sound in the word (part a), or that instead we determine one or two best representations of each word via some type of word learning procedure (part b). The problem with the network of part a is that a word with N sounds (e.g., "often" has three sounds, /ao, f, en/) has 2^N representations when we use the complete network (e.g., eight versions of "often") with two models for each sound. This means not only more computation, but even worse, more chances to cause word insertion or substitution in the recognition phase because there are so many more ways in which the words can occur. The network of part b, in which we explicitly enumerate the version of each unit used for each word, is far more viable; however, the problem is how to estimate the best sequence of units for each word in the lexicon. To do this properly we need occurrences, within the training set, of each word in the vocabulary from which we use the network of part a and backtrack to get the best sequences of the type shown in part b.

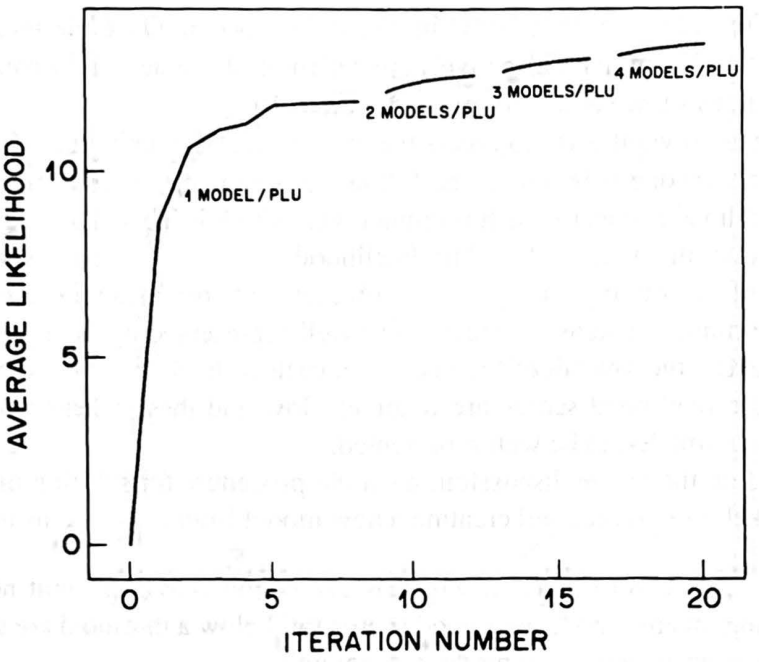


Figure 8.21 Average likelihood scores for sets of PLU models obtained from model splitting (after Lee et al. [7]).

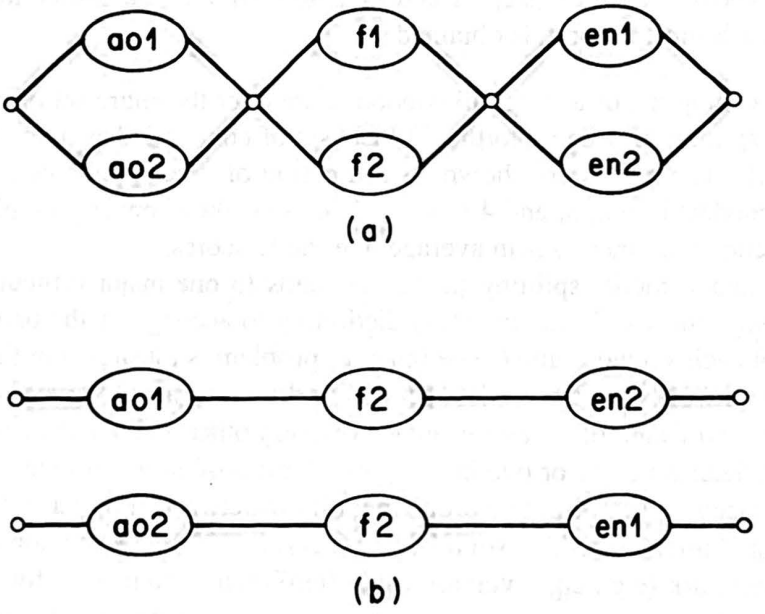


Figure 8.22 Word networks based on all combinations of units (a), or selected combinations of units (b) (after Lee et al. [7]).

For words that don't occur in the training set, some canonic representation must be relied on (e.g., use the primary model for each sound as a default). The necessity of having each word occur in the training makes this type of splitting method less viable than alternative procedures.

8.9.8.2 Clustering of Context Dependent Units

The alternative to model splitting (a top-down approach) is model clustering (a bottom-up approach) in which we initially start with the complete set of context-dependent units (as many as exist in the training set based on a count threshold of 1) and then sequentially merge units (actually the training tokens associated with the units) so that the decrease in likelihood score is minimized at each step. (In practice, we merge only units whose contexts are comparable, e.g., unit $p_i - p_j - p_k$ could be merged with units $p_i - p_j - p_\ell$, $\ell \neq k$, or $p_\ell - p_j - p_k$, or $\$ - p_j - p_k$, or $p_i - p_j - \$$, or $\$ - p_j - \$$.) This procedure is iterated either until a desired number of units is reached or until the resulting decrease in likelihoods gets too large.

A key advantage to model clustering is that it is trivial to modify the word lexicon to account for the decrease in units from merging. The procedure is basically to change each occurrence of both $p_i - p_j - p_k$ and $p_\ell - p_j - p_n$ to the merged unit, call it $\hat{p}_i - p_j - p_k$, whenever they occur in the lexicon. Thus model clustering is inherently simpler to implement than model splitting and therefore has been used more widely in practical systems.

Many variations on model clustering have been proposed, including knowledge-based allophonic clustering [22], in which specific knowledge of the vowel and consonant contexts is explicitly used to guide the clustering procedure, and CART-based phonetic clusters in which a decision tree is used to choose the most reasonable clustering sequence based on phonetic considerations.

8.9.9 Other Factors for Creating Additional Subword Units

In practice, the training methods for creating robust, complete sets of subword unit models for representing continuous speech are up against hard physical limits, including amount of training data and ability to reliably estimate model parameters from insufficient training. To obtain improvements in recognition performance, subject to the above constraints, several ideas have emerged for creating specialized units and models. For completeness, we briefly outline several interesting proposals that have been advanced along these lines.

A key source of difficulty in continuous speech recognition are the so-called function words, which include words like a, and, for, in, and is. These function words have the following properties:

1. They are generally unstressed in speech.
2. They are poorly articulated in continuous speech.
3. They are highly variable in pronunciation depending on context.

4. They account for a large percentage of the word recognition errors in continuous speech (upward of 50–70% in some tests).

To combat these problems, one simple idea is to represent function words independently of the rest of the training set, using either whole-word models, multiple pronunciations in the lexicon (e.g., the, thee), or special subword units, called function word dependent units, trained directly from occurrences of the function words within the training set. Experience shows small but consistent improvements in recognition performance when function word dependent units are added to the standard set of subword units.

The idea of representing function words can be extended to the representation of function phrases such as "in the," or "what is." Thus, specialized units can be created for these combinations in much the same way as for individual function words. Again there are small performance gains that are achieved when using function phrase units.

Another interesting idea is to create separate sets of units for both male and female talkers. The idea is that the spectral properties of the units are distinct for males and females. The problem is that by separating male from female talkers, the amount of training data for each separate gender set is reduced. Hence the reliability of the estimates of both sets of models is reduced even further. Experience again shows small, consistent gains in recognition performance using gender-specific models; hence this method is worth considering for practical implementations.

Finally it has been proposed that a combination of word models and subword unit models might give the best performance for specific tasks. The idea is that for words that do occur often in the training set (e.g., function words), creation of whole-word models provides the highest recognition performance. For all other words in the lexicon, some type of subword units is required. Hence a combination of word and subword units would probably lead to the best implementation for many applications. This idea has yet to be evaluated in a practical application.

8.9.10 Acoustic Segment Units

In this chapter we have shown that large vocabulary continuous speech recognition systems use a combination of ideas from phonetics and acoustics to define subword units and to create a "consistent" framework for training the units and implementing the overall recognition structure. The resulting system is neither phonetically nor acoustically consistent, but is instead a hybrid of the two methodologies. This is why the resulting subword units are called phonelike units (PLUs) rather than phones or allophones.

In an attempt to create a consistent acoustic framework (devoid, in theory, of the phonetic basis), it is possible to define a set of acoustic segment units (ASUs) that can be trained from continuous (unlabeled) speech, and which form a basis for representing any spoken input. In concept, all one need do is to have a procedure that automatically segments fluent speech into unlabeled sections [9, 23] (based on a maximum likelihood procedure using some type of spectral similarity measure), and then cluster the resulting segments to create a codebook of ASUs.

The problem now becomes one of creating an acoustic lexicon that represents words in the recognition vocabulary in terms of the appropriate sequence of ASUs. For systems in which every vocabulary word is seen in the training set, techniques for creating the acoustic lexicon exist and appear to work well [9, 24]. However, for large vocabulary systems the problem of automatically creating the acoustic lexicon remains a major obstacle to the practical use of ASUs.

8.10 CREATION OF VOCABULARY-INDEPENDENT UNITS

A major limitation in the training procedures discussed in this chapter is that the resulting subword unit models are not truly vocabulary independent. This is because the unit models are generally trained from tokens that occur in only a small subset of the possible contexts, and this subset is from the same words as used in the recognition tests. As such, the resulting units are word/vocabulary dependent and do not perform well for tasks in which different vocabularies and task syntaxes are used.

To alleviate this problem of vocabulary dependence, the "ideal" training procedure would be to use a training set that is completely independent of the test material, both in vocabulary and in syntax. If a sufficiently large training set is available, the units models will eventually converge so that the resulting recognition performance is virtually independent of the vocabulary and task.

To evaluate this idea, two experiments were run at CMU [25]. Using a vocabulary-independent training set of 15,000 sentences, subword unit models were created from subsets of 5000 (VI–5000), 10,000 (VI–10000), and 15,000 (VI–15000) sentences and tested against two tasks, namely a 122-word office correspondence task, and the 911 RM task. The results of these two experiments are given in Tables 8.7 and 8.8. For comparison, in Table 8.7, results based on training models from 1000 sentences from the office correspondence task (VD–1000) are also given. For 5 times the size training set the error rate from VI–5000 models is more than twice that of the VD–1000 models. Even with 15 times as much training data, the error rate is still somewhat larger for the VI–15000 model than for the VD–1000 model.

An even worse performance is seen for the RM task in which a VI–15000 training set led to almost twice the error rate of the RM–4200 (sentence) training set. (A final test was run at CMU in which new test sentences were recorded at CMU under the same recording conditions as those of the VI–15000 set, and the resulting recognition performance of both the VI–15000 set and the RM–4200 set were comparable. Thus, some of the large differences in performance result from differences in recording conditions.)

The results of these tests show that robust techniques for creating truly vocabulary-independent units are yet to be devised. Until such methods are available, truly continuous speech recognition for unlimited vocabularies and tasks will be out of range. The interim solution is to use VI models and bootstrap them to VD models for specific applications. Experimental evidence exists that such procedures are viable for many applications.

TABLE 8.7. Recognition performance on 122-word, office correspondence task with both VI and VD models (after Hon & Lee [25]).

Training Set	Word Coverage (%)	Triphone Coverage (%)	Word Error Rate (%)
VI–5000	44.3	63.7	23.9
VI–10000	63.9	95.3	15.2
VI–15000	70.5	99.2	13.3
VD–1000	100	100	11.4

TABLE 8.8. Recognition performance on 991-word, RM task, with both VI and VD models (after Hon & Lee [25]).

Training Set	Word Coverage (%)	Triphone Coverage (%)	Word Error Rate (%)
VI–15000	57.0	90	15.4
RM–4200	100	100	8.3

8.11 SEMANTIC POSTPROCESSOR FOR RECOGNITION

The final stage of processing in most speech recognizers is a semantic processor whose job is to eliminate from consideration all semantically meaningless sentences. In a sense, the semantic processor exploits the fact that the syntax used in recognition has a great deal of overcoverage; that is, it allows meaningless sentences to be passed to the semantic analyzer. The semantic processor can use the actual perplexity of the task (generally much lower than the perplexity of the syntax) to convert the recognized output to a semantically valid string.

In theory, the semantic processor should be able to communicate back to the recognizer to request a new string whenever the resulting string is deemed invalid. In practice, one of two simple strategies can be used; either the recognizer can generate a list of the best N sentences ($N = 500 - 1000$) that the semantic processor can search until a valid one is found, or it can assume that the best (recognized) string is semantically "close" to the correct string and therefore process it appropriately to determine a valid approximation.

Rather than discussing the details of how such semantic processing is done in practice, Figure 8.23 shows plots of improvements in word and sentence accuracy for different sets of subword units due to the use of a simple semantic postprocessor for the RM task [26]. Improvements in word accuracy of up 10% and improvements in sentence accuracy of over 20% are achieved, even with simple processing.

8.12 SUMMARY

The framework of large vocabulary, continuous speech recognition is well established.

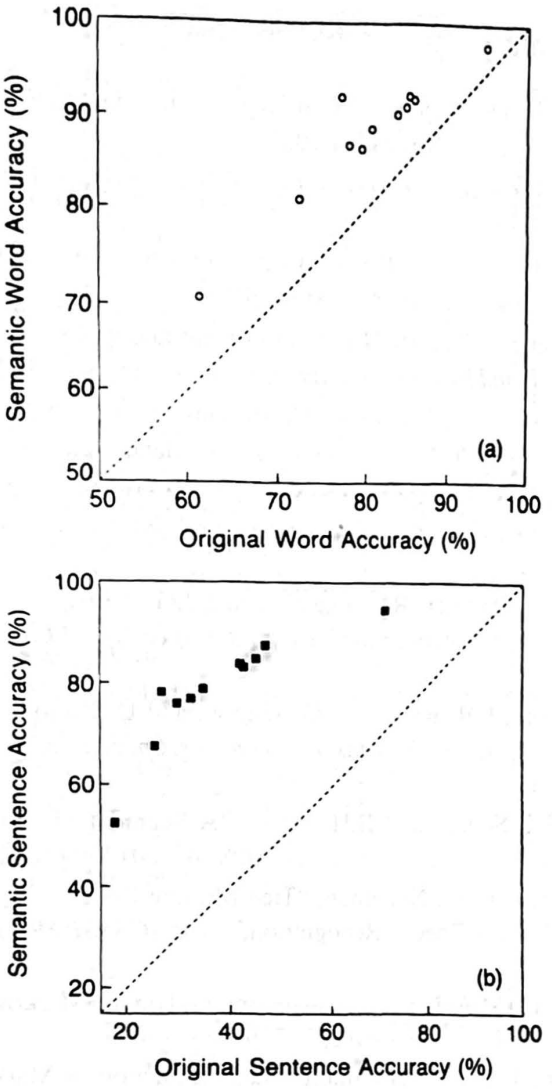


Figure 8.23 Word and sentence accuracy improvements in RM after semantic processing (after Pieraccini and Lee [26]).

Techniques for training subword models have been developed and work well in practice. Recognition systems have been developed and these, too, work well in practice. Recognition systems have been implemented with upward of 1000–20,000 word vocabularies using upward of 1000–2000 subword units. Many unanswered questions remain. A key one is how to efficiently choose and design context-dependent, vocabulary-independent units from training sets of reasonable (but finite) size. Other issues concern effectiveness of different spectral representations, including codebooks and tied-mixture densities, efficiency of implementation of search strategies, and efficient implementations of task syntax. Finally, the issues involved with task semantics are yet to be fully understood or resolved. Large vocabulary recognition has come a long way, but a great deal remains to be done before such systems will be used for practical applications.

REFERENCES

[1] F. Jelinek, "The Development of an Experimental Discrete Dictation Recognizer," *Proc. IEEE*, 73: 1616–1624, November 1985.

[2] K.F. Lee, *Automatic Speech Recognition—The Development of the SPHINX System*, Kluwer Academic Publishers, Boston, 1989.

[3] D.B. Paul, "The Lincoln Robust Continuous Speech Recognizer," *Proc. ICASSP 89*, Glasgow, Scotland, pp. 449–452, May 1989.

[4] R. Schwartz et al., "The BBN BYBLOS Continuous Speech Recognition System," *Proc. DARPA Speech and Natural Language Workshop*, pp. 94–99, February 1989.

[5] M. Weintraub et al., "Linguistic Constraints in Hidden Markov Model Based Speech Recognition," *Proc. ICASSP 89*, Glasgow, Scotland, pp. 699–702, May 1989.

[6] V. Zue, J. Glass, M. Phillips, and S. Seneff, "The MIT Summit Speech Recognition System: A Progress Report," *Proc. DARPA Speech and Natural Language Workshop*, pp. 179–189, February 1989.

[7] C.H. Lee, L.R. Rabiner, R. Pieraccini, and J.G. Wilpon, "Acoustic Modeling for Large Vocabulary Speech Recognition," *Computer Speech and Language*, 4: 1237–165, January 1990.

[8] A.E. Rosenberg, L.R. Rabiner, J.G. Wilpon, and D. Kahn, "Demisyllable Based Isolated Word Recognition," *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, ASSP–31: 713–726, June 1983.

[9] C.H. Lee, F.K. Soong, and B.H. Juang, "A Segment Model Based Approach to Speech Recognition," *Proc. ICASSP 88*, New York, pp. 501–504, April 1988.

[10] J.R. Bellegarda and D. Nahamoo, "Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition," *Proc. ICASSP 89*, Glasgow, Scotland, pp. 13–16, May 1989.

[11] X.D. Huang and M.A. Jack, "Semi-Continuous Hidden Markov Models for Speech Signals," *Computer Speech and Language*, 3: 239–251, 1989.

[12] F. Jelinek and R.L. Mercer, "Interpolated Estimation of Markov Source Parameters From Sparse Data," *Pattern Recognition in Practice*, E.S. Gelsema and L.N. Kanal, Eds., North-Holland Pub. Co., Amsterdam, pp. 381–397, 1980.

[13] P.J. Price, W. Fischer, J. Bernstein, and D. Pallett, "A Database for Continuous Speech Recognition in a 1000-Word Domain," *Proc. ICASSP 88*, New York, pp. 651–654, April 1988.

[14] R. Schwartz et al., "Robust Smoothing Methods for Discrete Hidden Markov Models," *Proc. ICASSP 89*, Glasgow, Scotland, pp. 548–551, May 1989.

[15] E. Giachin, C.H. Lee, L.R. Rabiner, A.E. Rosenberg, and R. Pieraccini, "On the Use of Interword Context-Dependent Units for Word Juncture Modeling," *Computer Speech and Language*, 6: 197–213, 1992.

[16] C.H. Lee, E.P. Giachin, and A.E. Rosenberg, "Word Juncture Modeling Using Phonological Rules for HMM Based Continuous Speech Recognition," *CSELT Tech. Reports*, 18 (3): 189–194, June 1990.

[17] C.H. Lee, L.R. Rabiner, and R. Pieraccini, "Speaker Independent Continuous Speech Recognition Using Continuous Density Hidden Markov Models," Proc. NATO-ASI, Speech Recognition and Understanding: Recent Advances, Trends and Applications, P. Laface

and R. DeMori, Eds., Springer-Verlag, Cetraro, Italy, pp. 135–163, 1992.

[18] R. Pieraccini and A.E. Rosenberg, "Automatic Generation of Phonetic Units for Continuous Speech Recognition," *Proc. ICASSP 89*, Glasgow, UK, pp. 623–626, May 1989.

[19] R. Pieraccini, C.H. Lee, E. Giachin, and L.R. Rabiner, "An Efficient Structure for Continuous Speech Recognition," Proc. NATO-ASI, Speech Recognition and Understanding: Recent Advances, Trends and Applications, P. Laface and R. DeMori, Eds., Springer-Verlag, Cetraro, Italy, pp. 211–216, 1992.

[20] D. Paul, "A Speaker Stress Resistant Isolated Word Recognizer," *Proc. ICASSP 87*, Dallas, TX, pp. 713–716, April 1987.

[21] C.H. Lee, E. Giachin, L.R. Rabiner, R. Pieraccini, and A.E. Rosenberg, "Improved Acoustic Modeling for Large Vocabulary Continuous Speech Recognition," *Computer Speech and Language*, 6: 103–127, 1992.

[22] L. Deng et al., "Acoustic Recognition Component of an 86,000 Word Speech Recognizer," *Proc. ICASSP 90*, Albuquerque, NM, pp. 741–744, April 1990.

[23] T. Svendsen and F.K. Soong, "On the Automatic Segmentation of Speech Signals," *Proc. ICASSP 87*, Dallas, TX, pp. 77–80, April 1987.

[24] L.R. Bahl, P.F. Brown, P.V. de Souza, R.L. Mercer, and M.A. Picheny, "Automatic Construction of Acoustic Markov Models for Words," *Proc. Int. Symposium on Signal Processing Applications*, (Brisbane, Australia), 565–569, 1987.

[25] H.W. Hon and K.F. Lee, "On Vocabulary Independent Speech Modeling," *Proc. ICASSP 90*, Albuquerque, NM, pp. 725–728, April 1990.

[26] R. Pieraccini and C.H. Lee, "Factorization of Language Constraints in Speech Recognition," *Proc. 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, CA, June 1991.*

ELECTRICAL ENGINEERING
Signal Processing

LAWRENCE RABINER
BIING-HWANG JUANG

FUNDAMENTALS OF SPEECH RECOGNITION

For those involved in designing and implementing systems for human-machine interface via voice, authors Lawrence Rabiner and Biing-Hwang Juang offer a comprehensive examination of the principles and underlying theory of speech recognition.

Fundamentals of Speech Recognition:

- Begins with an overview of the entire field, including a discussion of the breadth and depth of the various disciplines that are required for a deep understanding of all aspects of speech recognition.
- Reviews the theory of acoustic-phonetics in which the authors try to characterize basic speech sounds according to both their linguistic properties and the associated measurements.
- Discusses the fundamental techniques used to provide the speech features used in all recognition systems, in particular the filter bank approach and the linear prediction method.
- Deals with the fundamental problems of defining speech feature vector patterns and comparing pairs of feature vector patterns both locally and globally so as to derive a measure of similarity between speech utterances.
- Examines the key issues of training a speech recognizer and adapting the recognizer parameters to different speakers, and speaking environments.
- Introduces a basic set of automatic statistical modeling techniques for characterizing speech.
- Extends the speech recognition problem from single word utterances to fluent speech.
- Considers the main factors that affect the performance of a speech recognizer in various deployment conditions and discusses means to enhance the system robustness.
- Concludes by discussing the basic principles that make some tasks successful while others fail.

Whether you're a practicing engineer, scientist, linguist, programmer, or are just interested in learning more about this fascinating field, this book provides a comprehensive yet accessible introduction to the fundamentals of speech recognition.

ISBN 0-13-015157-2

PRENTICE HALL
Upper Saddle River, NJ 07458