Chapter 3

Speech segmentation

Speech segmentation, as done in this dissertation, refers to the process of determining the boundaries between phonemes in the speech signal. No higher-level lexical information is used to accomplish this. This chapter presents the approach followed to achieve unconstrained (implicit) segmentation of speech into phonemes. Section 3.1 gives the features used, and Section 3.2 discusses the segmentation techniques, with Section 3.2.1 using hidden Markov models, and Section 3.2.2 recurrent neural networks. Postprocessing is discussed in Section 3.3. Finally, Section 3.4 explains the accuracy measure used to evaluate segmentation performance.

3.1 Preprocessing

Preprocessing, in the context of speech segmentation, refers to the feature extraction process, as discussed in detail in Chapter 2, Section 2.1. The aim of the preprocessing stage is to transform the raw speech samples into a feature space that is better suited for the discrimination between a phoneme boundary and no boundary.

To obtain the speech features, the raw speech samples (digitised) are first pre-emphasised

(using a pre-emphasis coefficient of 0.97), blocked into overlapping frames of 25.6 ms width and 10 ms frame shift, signal bias removed, windowed using a Hamming window, spectral analysis done on each frame, the energy and static mel frequency cepstrum coefficients (MFCCs) calculated (using 26 triangular filters), liftered (using a liftering coefficient of 22) and normalised, and finally the delta MFCCs (using two MFCCs left and right in a regression formula) and energy calculated. The features used can be characterised as a 26-dimensional vector containing

- 1 energy feature,
- 12 liftered mel frequency cepstrum coefficients (MFCCs),
- 1 delta energy feature, and
- 12 delta MFCCs.

For segmentation using hidden Markov models, normalisation includes energy normalisation, as well as cepstral mean normalisation. In addition to these, simple linear re-scaling is also used when the recurrent neural networks are used. The main purpose of the simple linear re-scaling is to prevent saturation of the neuron transfer functions. The mean and variance used in the re-scaling process, are estimated from the entire set of training speech utterances, so that all the utterances may be normalised by the same values.

3.2 Segmentation

The goal of speech segmentation in this dissertation, as stated earlier, is to determine the locations of the phoneme boundaries in the speech signal, without using higher-level lexical knowledge (the underlying phoneme sequence).

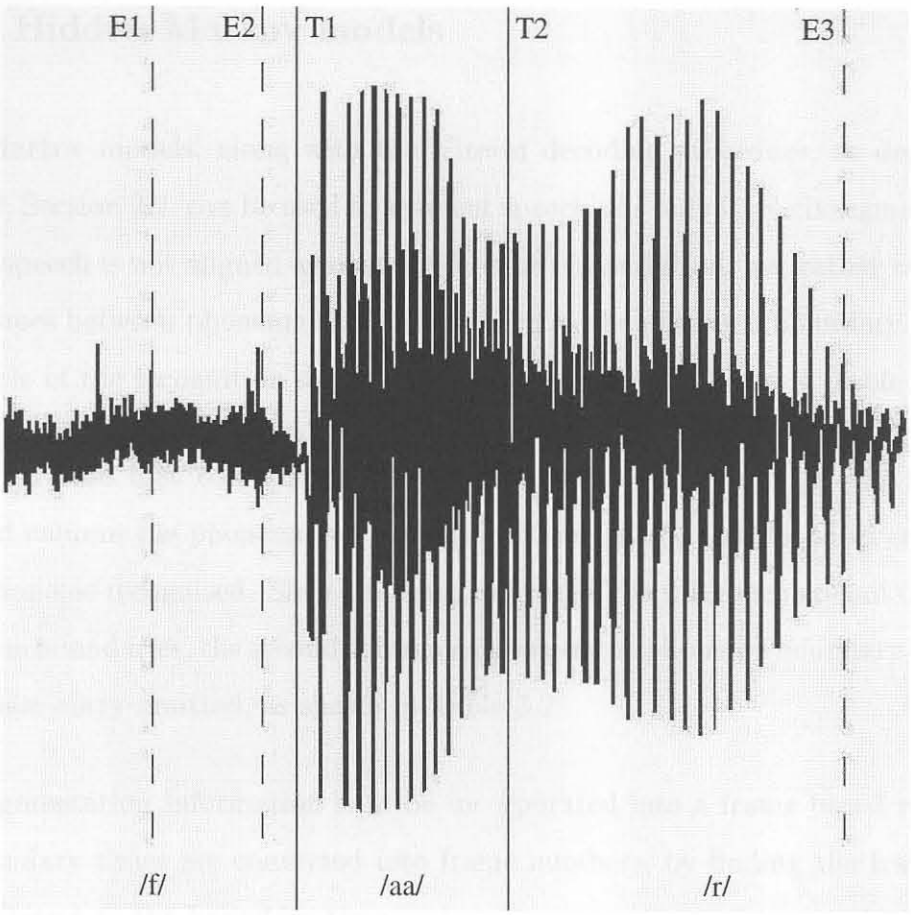

Figure 3.1: Example of the segmentation of the word "four".

Figure 3.1 shows an example of a speech signal (the word "four"), with the true boundaries indicated by T1 and T2, and the estimated boundaries indicated by E1, E2 and E3. It can be seen that there is an error (or distance) between the true and estimated boundaries. We want the distance between the estimated and true (or desired) phoneme boundaries to be as small as possible. In addition, the number of boundaries falsely detected (or insertions), must be minimised. The boundary, E1, is an example of an inserted boundary, and if the distance between T2 and E3 is larger than the maximum allowed distance, E3 can also be considered as an insertion. The number of correct boundaries not detected (or deletions), must also be kept to a minimum. If the distance between T2 and E3 is larger than the window, there is boundary deletion.

3.2.1 Hidden Markov models

Hidden Markov models, along with the Viterbi decoding procedure, as described in Chapter 2, Section 2.2, can be used to segment speech. As only implicit segmentation is used, the speech is not aligned against a reference transcription, but rather recognised, and the times between phoneme transitions taken as the phoneme boundary locations. An example of the recognition result of a speech sentence is shown in Table 3.1.

In Table 3.1, the first column indicates the phoneme start time (in 100 ns units), the second column the phoneme end time (in 100 ns units), and the third column the specific phoneme recognised. Since the start and end of the utterance are not considered as phoneme boundaries, the second column can be used as phoneme boundary locations, with the last entry omitted, as shown in Table 3.2.

As the segmentation information is to be incorporated into a frame based recogniser, these boundary times are converted into frame numbers, by finding the frame whose centre is closest to the boundary time.

A sequence of 1's (representing a boundary), and 0's (representing no boundary) can then be constructed by taking a zero vector of length equal to the utterance length (in total number of frames). The 0's at the indices corresponding to the frame numbers of the boundaries are then replaced by a 1. Such a sequence might have the following form

0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 ... 0 0 1 0 0.

This sequence is in a convenient form for further processing or to evaluate the accuracy of the segmentation system. It directly indicates whether a frame contains a boundary or not (the centre of the frame).

Table 3.1: Example of a recognition result of a speech sentence.

0	1275000	sil
1798125	2271250	aa
2271250	2558125	r
2558125	2742500	dx
2742500	3195000	ih
3195000	4256250	f
4256250	4863125	ih
4863125	6109375	sh
6109375	6996875	l
6996875	7400000	ih
7400000	7847500	n
7847500	8083750	sil
8083750	8783750	t
8783750	9503750	eh
9503750	10208750	l
10208750	10600000	ih
10600000	10962500	sil
10962500	11513750	jh
11513750	12041875	ih
12041875	12756250	n
12756250	13970000	s

Table 3.2: Boundary locations for the given example, in 100 ns units.

1275000,	2271250,	2558125,	2742500	3195000,
4256250,	4863125,	6109375,	6996875,	7400000,
7847500,	8083750,	8783750,	9503750,	10208750,
10600000,	10962500,	11513750,	12041875,	12756250

3.2.2 Recurrent neural networks

A bi-directional recurrent neural network is used in the work presented here, for the task of segmenting the speech. It takes an entire speech utterance as input, and produces two output sequences, one estimating the probability of a boundary, and the other the probability of no boundary, for each input vector in the utterance.

In order to train the neural network, corresponding output target values must be given for each input vector. The first step in this process is to obtain the true boundary locations, in a process similar to that described in the previous section on segmentation by HMMs. The difference is that the transcription label files, provided with the TIMIT database, are used, instead of the recognised results. This leads to a sequence of 1's (representing a boundary) and 0's (representing no boundary), for each utterance on which the neural network should be trained. The second step is then to assign a value of 0.5 as target to the left and right of each boundary frame, only if a neighbouring frame is not a boundary. The value of 0.5 represents the uncertainty present in whether the boundary should be in one of the neighbouring frames. The resulting sequences of target values, one for each utterance, have an almost triangular wave-like nature.

Figure 3.2 shows an example of a sequence of target values. Also shown in the figure is an example of the output of the neural network, estimating the probability of a boundary. The neural network outputs approach that of the target values.

After the neural network has been trained on a large number of training utterances, it can be used to estimate the probability of a phoneme boundary and probability of no phoneme boundary, for each utterance. It does this by taking as input, the entire speech utterance, and estimating these probabilities for each acoustic vector. The neural network is thus used as a classifier, where the input vectors are classified as either belonging to the class of "boundary" or "no boundary", and the outputs can be interpreted as probabilities, according to [69]. Two sequences, corresponding to these two probabilities, are thus obtained. The length of each sequence is equal to the

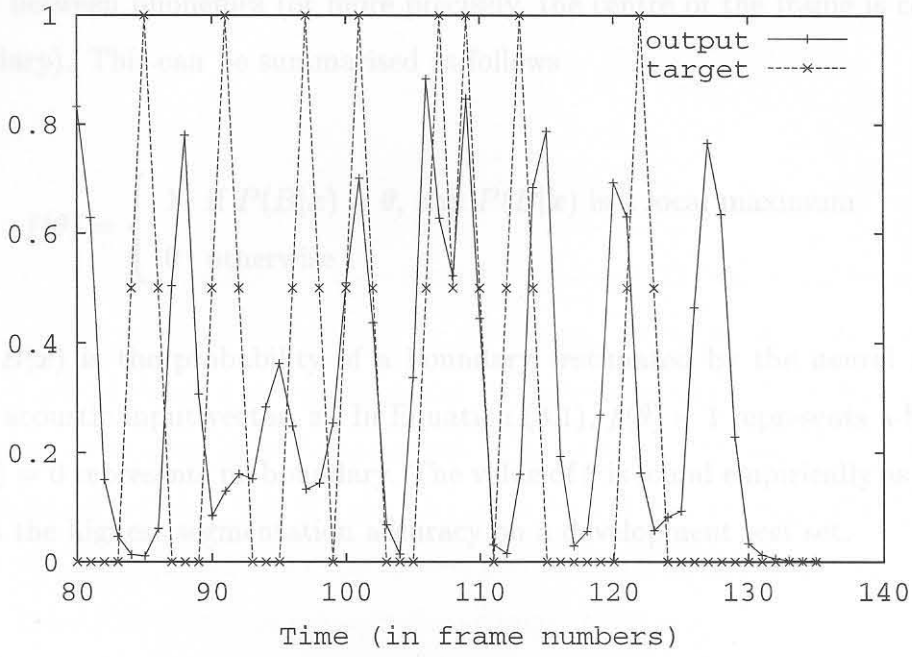

Figure 3.2: Example of neural network target and output values.

number of frames in the utterance. In order to determine the actual phoneme boundary locations, a subsequent postprocessing stage, discussed in the next section, is used.

3.3 Postprocessing

After the segmentation process, as described above, a set of exact boundary locations exist in the case of hidden Markov model based segmentation. No postprocessing is thus required, as the boundaries between phonemes are simply the time instances between phoneme HMM model transitions. In the case of the recurrent neural networks, however, only the probability of a boundary (and no boundary) is present. Some decision must be made as to when an output indicates a boundary or not.

In the work presented here, a simple threshold function is applied. If the output of the neural network, representing the probability of a boundary, is a local maximum and is above a certain threshold, θ, then that frame of speech is said to contain a

boundary between phonemes (or more precisely, the centre of the frame is considered the boundary). This can be summarised as follows

$$f(\theta) = \begin{cases} 1 & \text{if } P(B|\boldsymbol{x}) \geq \theta, \text{ and } P(B|\boldsymbol{x}) \text{ is a local maximum} \\ 0 & \text{otherwise} \end{cases}, \tag{3.1}$$

where $P(B|\boldsymbol{x})$ is the probability of a boundary (estimated by the neural network), given the acoustic input vector, $\boldsymbol{x}$. In Equation (3.1), $f(\theta) = 1$ represents a boundary, while $f(\theta) = 0$ represents no boundary. The value of θ is found empirically as the value that gives the highest segmentation accuracy on a development test set.

3.4 Accuracy measure

In order to measure how well a particular segmentation method performs, an accuracy measure is needed. The accuracy measure used in our work, attempts to evaluate a segmentation method objectively, penalising both insertions and deletions of boundaries. When $N_t = H + D$ (hits plus deletions), the accuracy of the segmentation system can be calculated as

$$Acc = \frac{N_t - D - I}{N_t} \cdot 100\%, \tag{3.2}$$

and the percentage of boundaries correctly identified, as

$$Cor = \frac{H}{N_t} \cdot 100\%, \tag{3.3}$$

where D is the number of deletions, I the number of insertions, H the number of boundaries correctly identified (or hits), and N_t the total number of target boundaries. The number of insertions and deletions are related to N_t, N_e and H through

$$D = N_t - H, \tag{3.4}$$

$$I = N_e - H., \tag{3.5}$$

where N_e is the total number of estimated boundaries. To use Equations (3.2) and (3.3), the number of hits, H, must first be determined. After H has been determined, the number of deletions and insertions can be calculated using Equations (3.4) and (3.5). In particular, care must be taken not to associate an estimated boundary with a target boundary to the left or right of target boundaries closest to the estimated boundary. If this is the case, the estimated boundary must rather be regarded as an insertion. Also, if the distance between the estimated and true boundary is larger than a certain threshold, the estimated boundary should also not be counted as a hit.

The process followed in this dissertation can be best explained by way of an example. Consider the following sequence of hypothetical target values

0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0

with estimated boundaries being given as

0 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0

From these sequences it can be seen that the number of target boundaries, N_t, is 4, while the number of estimated boundaries, N_e, is 5. The process of calculating the number of hits can then be summarised as follows:

1. Calculate two arrays, p^t and p^e, containing the frame numbers of the target and estimated boundaries, respectively. For the above example (using zero array indexing), that is

$$p^t = [3,7,10,13]$$

$$p^e = [2,4,6,7,10]$$

2. Calculate a distance matrix, $\boldsymbol{d} = \{d_{ji}\}$, that indicates the distances between all estimated and target boundaries, where

$$d_{ji} = |p_j^t - p_i^e|, \qquad j = 0, 1, \ldots, N_t - 1, i = 0, 1, \ldots, N_e - 1.$$

For the example, this matrix can be given as

$$\boldsymbol{d} = \begin{pmatrix} 1 & 1 & 3 & 4 & 7 \\ 5 & 3 & 1 & 0 & 3 \\ 8 & 6 & 4 & 3 & 0 \\ 11 & 9 & 7 & 6 & 3 \end{pmatrix}$$

where the horizontal dimension represents the estimated boundaries, and the vertical dimension the target boundaries.

3. Find the minimum of the distance matrix and save the result in a result matrix, $\boldsymbol{r}$, of the same dimension as the distance matrix (initially filled with infinite values (or at least larger than the total number of frames in the input sentence)) at the same position as where the minimum value in the distance matrix is located. The entire row and column of the distance matrix, in which the minimum value is located, is then replaced by infinite values (large numbers). For the example above, the minimum value in the matrix (minimum operation starting from top left proceeding to bottom right) is the 0 located at the second target boundary ($j = 1$), and fourth estimated boundary ($i = 3$). The result matrix can thus be given as

$$\boldsymbol{r} = \begin{pmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

and the distance matrix becomes

$$\boldsymbol{d} = \begin{pmatrix} 1 & 1 & 3 & \infty & 7 \\ \infty & \infty & \infty & \infty & \infty \\ 8 & 6 & 4 & \infty & 0 \\ 11 & 9 & 7 & \infty & 3 \end{pmatrix}$$

4. Repeat the previous step until distance matrix contains only infinite values (large numbers). The result matrix for the example given here, is then

$$\boldsymbol{r} = \begin{pmatrix} 1 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & 7 & \infty & \infty \end{pmatrix}$$

5. In order to make sure that the estimated boundaries are not associated with target boundaries to the left or right of the target boundaries nearest to the estimated boundary, the result matrix must be checked and erroneous hits replaced by infinite values (large numbers). The 7 in the result matrix given above is an example of such a boundary. When removed, the final result matrix can be given as

$$\boldsymbol{r} = \begin{pmatrix} 1 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

6. The next step used to compute the number of hits, is to compare the distances in the result matrix with a threshold. Each entry in the result matrix is retained only if it is less than or equal to the threshold. For a threshold of 1, the result matrix given above is the final answer. For a threshold of 0 (estimated and true boundaries must be in the same frames), the top left value is removed, giving the following results matrix

$$\boldsymbol{r} = \begin{pmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

7. The final step is to count the number of non-infinite numbers in the results matrix. This is equal to the number of boundaries correctly found, or the number of hits, H. For the results matrix given above, $H = 2$.

In the example given above, the number of deletions is $D = N_t - H = 4 - 2 = 2$, and the number of insertions $I = N_e - H = 5 - 2 = 3$. The percentage accuracy and correct can be calculated using Equations (3.2) and (3.3). For the example given above, the percentage accuracy is -25% and the percentage correct is 50%. The negative accuracy indicates that more incorrect boundaries have been predicted than correct boundaries (hits).

The threshold used in the process described above, also called the window, can be used to compute statistics about the number of boundaries correctly identified with a certain allowed tolerance. For a window size of 0, the estimated and target boundaries may not be in different frames, for a window size of 1, they may differ by 1 frame, etc. In this dissertation, a frame shift of 10 ms is used, and a window size of 0 indicates that the target and estimated boundaries must lie within 5 ms of each other (5 ms to the left and right of the centre of the frame, where the boundary is assumed to be located in the centre of a frame).

Table 3.3: Mapping between window size and tolerance.

Window size	Tolerance
0	≤ 5 ms
1	≤ 10 ms
2	≤ 15 ms
3	≤ 20 ms
4	≤ 25 ms
5	≤ 30 ms
6	≤ 35 ms
7	≤ 40 ms
8	≤ 45 ms
9	≤ 50 ms

Table 3.3 summarises the requirement of the distance between an estimated boundary and a true boundary, to be regarded as a hit.