Chapter 4

Speech recognition

This chapter considers phoneme recognition on the TIMIT database. A speech sentence is presented to the recogniser and the desired result is a sequence of phones. The preprocessing is essentially the same as that used for HMM segmentation, described in the previous chapter, and will not be repeated here. The baseline phoneme recogniser used in this dissertation, is discussed (Section 4.1). It is also shown how the segmentation information of the previous chapter can be incorporated into the recognition process (Section 4.2). Two methods are considered here. The first is to modify the HMM transition probabilities, as shown in Section 4.2.1, while the second is to make use of an adaptive word transition penalty term, as discussed in Section 4.2.2. Finally, the accuracy measure used to evaluate the recognition performance, is discussed in Section 4.3.

4.1 Recognition (baseline)

The baseline phoneme recogniser, used in this work, is based on the Cambridge University hidden Markov toolkit (HTK). In HTK, recognition of an unknown input utterance is performed through the use of a recognition network [3]. This network or lattice is constructed from a word-level network, also called a grammar, a dictionary (containing the phonetic transcriptions of each word), and a set of HMMs.

The recognition network, that is used to do phoneme recognition, consists of a set of nodes connected by arcs. The nodes in this network correspond to the HMM states, and the arcs connecting the nodes, correspond to the transitions between HMM states. When an utterance is to be recognised, a number of possible paths exist from the start node to the end node. The decoder finds the optimal path, using a modified form of the Viterbi algorithm, as described in Chapter 2, Section 2.2.2. Further details on the recognition process in HTK can be found in [3, 125].

In phoneme recognition experiments, each HMM models a different phoneme. In this dissertation, diagonal covariance HMMs are used. The optimal path found through the decoding process reveals the most likely underlying phoneme sequence of a given speech utterance. This is the desired result of the recognition process.

The TIMIT database contains a phonetic transcription for each speech waveform file. These transcription files make use of 61 phonemes. It is common practice to convert the phoneme set to the standard 39 phonemes, when phoneme recognition experiments are conducted. Table 4.1 shows the mapping of phonemes from the 61 TIMIT phones, to the standard 39 phonemes, typically used in measuring phonetic recognition performance. In this table, the glottal stop "q" is ignored.

In HTK we used the 39 phonemes as "words" and their associated transcriptions are just the phonemes themselves. In this dissertation, the terms phonemes and words are therefore used interchangeably, unless indicated otherwise.

The grammar, or word-level network, used for phoneme recognition, is a word loop grammar. Figure 4.1 shows a graphical depiction of a word loop grammar. Note that for phoneme recognition experiments, each phoneme is treated as a word. The figure shows how each "word" may follow any other "word". This grammar can be specified in the extended Backus-Naur form (EBNF) as shown in Figure 4.2.

Table 4.1: Mapping between the 61 TIMIT phones (2nd, 4th, 6th and 8th columns) and the standard 39 phones (in boldface) typically used in measuring phonetic recognition performance. The glottal stop "q" is ignored. Arpabet symbols are used for the phones.

aa	aa ao	**ae**	ae	**ah**	ah ax ax-h	**aw**	aw
ay	ay	**b**	b	**ch**	ch	**d**	d
dx	dx	**dh**	dh	**eh**	eh	**er**	er axr
ey	ey	**f**	f	**g**	g	**hh**	hh hv
ih	ih ix	**iy**	iy	**jh**	jh	**k**	k
l	l el	**m**	m em	**n**	n en nx	**ng**	ng eng
ow	ow	**oy**	oy	**p**	p	**r**	r
s	s	**sh**	sh zh	**t**	t	**th**	th
uh	uh	**uw**	uw ux	**v**	v	**w**	w
y	y	**z**	z	**cl**	bcl pcl dcl tcl gcl kcl epi pau h#		

In order to improve phoneme recognition performance, a language model must be used, as discussed in Chapter 2, Section 2.2.3. The use of a language model in a word loop grammar, with bigram probabilities, takes into account that not all "words" (phonemes) follow each other with equal probability. In HTK, a language model probability (calculated from the text data) is thus added to tokens emitted from word-end nodes, before they enter the following word. In addition, a word transition penalty, as discussed in Chapter 2, Section 2.2.4, is added.

4.2 Recognition using segmentation information

This section shows how to incorporate the segmentation information of Chapter 3, in order to potentially improve phoneme recognition performance. The basic idea behind the use of segmentation information, is that segmentation information can give a fairly accurate indication of when HMM model transitions must take place, thereby

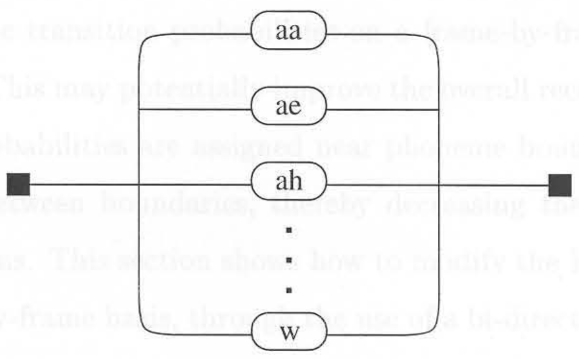

Figure 4.1: Loop grammar used for phoneme recognition.

```
$phn =  aa |  ae |  ah |  aw |  ay |  b |  ch |  cl |  d |  dx |
        dh |  eh |  er |  ey |  f |   g |  hh |  ih |  iy |  jh |
        k |   l |   m |   n |   ng |  ow |  oy |  p |   r |   s |
        sh |  t |   th |  uh |  uw |  v |   w |   y |   z;
(<$phn>)
```

Figure 4.2: Grammar used to construct a recognition network.

decreasing the number of insertions and deletions in the recognition process. Two methods are considered. The first involves the modification of the HMM transition probabilities, whereas the second method makes use of an adaptive word transition penalty. These techniques are discussed in the following sections.

4.2.1 HMM transition probability modification

During the recognition process, the transition from one HMM state to another, is dictated by the transition probabilities of the HMM. If a transition probability is low, the probability of taking that particular transition is low, and *vice versa*. In standard HMM recognition systems, these transition probabilities are determined during the training process, and remain fixed throughout the recognition of the utterance. By modifying the transition probabilities, based on segmentation information, an attempt is made to modify the transition probabilities on a frame-by-frame basis, during the recognition process. This may potentially improve the overall recognition performance, as high transition probabilities are assigned near phoneme boundaries, and low transition probabilities between boundaries, thereby decreasing the number of phoneme insertions and deletions. This section shows how to modify the HMM transition probabilities on a frame-by-frame basis, through the use of a bi-directional recurrent neural network (BRNN).

Consider the use of a neural network to estimate the *a posteriori* probability, $P(B|\boldsymbol{x})$, which is the probability of a phoneme boundary, given the input $\boldsymbol{x}$, and $P(B^{'}|\boldsymbol{x})$ which is the probability of no phoneme boundary, given the input data. The BRNN estimates $P(B|\boldsymbol{x})$ and $P(B^{'}|\boldsymbol{x})$ for the entire sequence of input vectors, thus producing two output sequences containing these probabilities. Details on the use of bi-directional recurrent neural networks can be found in Chapter 2 and Appendix A.

Two different ways of modifying the transition probabilities, based on segmentation information, are considered in this dissertation. The first involves a linear combination of the segmentation probability and the HMM transition probability, whereas the second makes use of a non-linear combination. These two techniques are discussed in the next two sections.

Linear combination

Let A be the event that an HMM transition takes place from state i to j, and B is the event that a phoneme boundary occurs and $B^{'}$ the event that no phoneme boundary occurs. In the following equations, substitute either B for C or $B^{'}$ for C, depending on whether an inter or an intra HMM transition is considered. The conditional probability that a transition from state i to j occurs, given C, can then be written as

$$P(A/C) = \frac{P(A \cap C)}{P(C)}, \tag{4.1}$$

where $P(A \cap C)$ is the joint probability of events A and C, and $P(C)$ is the *a priori* probability of event C. This can be rewritten as

$$P(A \cap C) = P(A/C)P(C). \tag{4.2}$$

A and C can be considered as independent random variables, and the conditional probability $P(A/C)$ can be written as

$$P(A/C) = P(A), \tag{4.3}$$

where $P(A)$ is the *a priori* probability of the event A. The joint probability of events A and C can then be written as

$$P(A \cap C) = P(A)P(C), \tag{4.4}$$

but since $P(A) = a_{ij}$ is just the transition probability from state i to state j, and $P(C)$ is one of the two neural network outputs, Equation (4.4) can be written as

$$P(A \cap C) = a_{ij}^{'} = a_{ij}s_{ij}. \tag{4.5}$$

In Equation (4.5), $a_{ij}^{'}$ is the modified transition probability, and s_{ij} is one of the two neural network outputs, defined as

$$s_{ij} = \begin{cases} P(B|\boldsymbol{x}) & \text{if transition from state } i \text{ to } j \text{ is an inter HMM transition} \\ P(B^{'}|\boldsymbol{x}) & \text{if transition from state } i \text{ to } j \text{ is an intra HMM transition} \end{cases}, \tag{4.6}$$

where $P(B|\boldsymbol{x})$ and $P(B^{'}|\boldsymbol{x})$ are the probabilities estimated by the two neural network outputs. At every time instant, the transition probabilities are thus modified according to Equations (4.5) and (4.6). Alternatively, the Viterbi algorithm, given by Equation (2.35) in Chapter 2, can be reformulated as

$$\tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + \tilde{s}_{ij}] + \tilde{b}_j(\boldsymbol{o}_t), \tag{4.7}$$

where $\tilde{s}_{ij} = \log(s_{ij})$.

Non-linear combination

The previous section showed how the segmentation information (probability of a boundary and no boundary) can be linearly combined with the HMM transition probabilities (in the logarithmic domain). It is, however, interesting to consider using only the maximum of the segmentation probability and HMM transition probability, thus in effect trusting the "source" of the probability to be used as HMM transition probability, as the one with the highest probability.

The modified HMM transition probability can thus simply be given as the maximum of the segmentation probability and the HMM transition probability, or

$$a'_{ij} = \max(a_{ij}, s_{ij}). \tag{4.8}$$

where s_{ij} is defined in Equation (4.6). The standard Viterbi algorithm, given by Equation (2.35), needs no modification if a'_{ij} is used instead of a_{ij}.

4.2.2 HMM word transition penalty modification

In HTK, a word transition penalty is added when a transition from one word to another occurs, as explained in Chapter 2, Section 2.2.4. This word transition penalty is fixed throughout the recognition process, and must be set empirically. This section shows how the word transition penalty can be made adaptive on a frame-by-frame basis, using the segmentation information. It is intended that such an adaptive word transition penalty (also a phoneme penalty, as each phoneme is considered as a word in phoneme recognition experiments) will result in a reduction in both deletions and insertions. When the word transition penalty term is negative and becomes more negative, insertions tend to increase, but deletions decrease, and *vice versa*. By using the segmentation information on a frame-by-frame basis, it is believed that the word transition penalty term approaches a more optimal value for each frame, instead of a fixed word transition penalty term over all observations. Note that the term "word transition penalty" is used as HTK provides a fixed word transition penalty, that can be conveniently modified as shown below. Due to the fact that phonemes are equal to words in the phoneme recognition experiments, this is of little consequence. In a true word recognition task, the Viterbi decoding engine must be modified so that our word transition penalty becomes a phone transition penalty instead.

The adaptive word transition penalty can be viewed as consisting of three terms, or

$$w = a \cdot w_c + b \cdot w_p + c \cdot w_b \tag{4.9}$$

where w is the adaptive word transition penalty, w_c is a varying confidence term (used to encourage word transitions), w_p is a varying penalty term (used to discourage word transitions), and w_b is the standard HTK fixed penalty term, here regarded as a bias or offset term. The segmentation information can be conveniently incorporated into w_c and w_p. Let

$$w_c = -\log\{P(B^{'}|\boldsymbol{x})\}, \tag{4.10}$$

and

$$w_p = +\log\{P(B|\boldsymbol{x})\}, \tag{4.11}$$

where $P(B|\boldsymbol{x})$ and $P(B^{'}|\boldsymbol{x})$ are the two outputs of a bi-directional recurrent neural network, estimating the probability of a boundary given the input data, and the probability of no boundary given the input data, respectively. In Equation (4.9), w_b is set empirically, or can be set to zero. Equation (4.9) can thus be rewritten as

$$w = -a \cdot \log\{P(B^{'}|\boldsymbol{x})\} + b \cdot \log\{P(B|\boldsymbol{x})\} + c \cdot w_b. \tag{4.12}$$

Equation (4.12) is in logarithmic units. Alternatively, it can be rewritten in non-logarithmic units as

$$w^{'} = e^{w} = P(B|\boldsymbol{x})^{b} \cdot P(B^{'}|\boldsymbol{x})^{-a} \cdot {w_b}^{c}, \tag{4.13}$$

or

$$w^{'} = e^{w} = \frac{P(B|\boldsymbol{x})^{b}}{P(B^{'}|\boldsymbol{x})^{a}} \cdot {w_b}^{c}. \tag{4.14}$$

In this dissertation, the values of a and b are all fixed at the same value, and c is set to $+1$. The specific value of a is found empirically, as the value that maximises recognition performance on a development test set. Equation (4.14) thus simplifies to

$$w^{'} = e^{w} = \left[\frac{P(B|\boldsymbol{x})}{P(B^{'}|\boldsymbol{x})}\right]^{a} \cdot w_b. \tag{4.15}$$

At every time instant, the word transition penalty term is thus calculated, using Equation (4.15), and the logarithm of this value is added to tokens emitted from word-end nodes. Alternatively, the Viterbi algorithm, given by Equation (2.35) in Chapter 2, can be reformulated as

$$\tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1 \le i \le N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + w] + \tilde{b}_j(\boldsymbol{o}_t), \tag{4.16}$$

where w is only added to the terms in brackets if a transition between words (phonemes) occur, when it is to be considered in the Viterbi search.

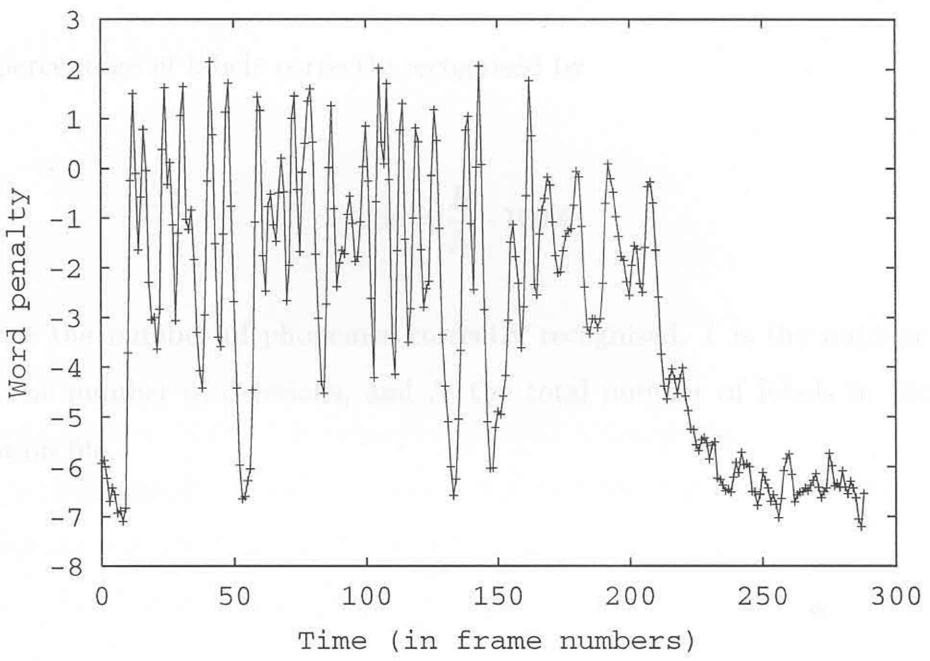

Figure 4.3: Example of an adaptive word transition penalty term with $w_b = 0$.

Figure 4.3 shows an example of the adaptive word transition penalty, w, with the bias term set to 0. In this figure, it can be seen that the term varies significantly, and it is intended that such variations will result in better recognition performance, than if only a fixed penalty term (w_b) is used. This is because the Viterbi search is modified to favour the more likely path (with the word penalty increasing or decreasing the path score at each time instant) based on the phoneme boundary probabilities.

4.3 Accuracy measure

In order to evaluate the recognition performance, HTK makes use of a dynamic programming based string alignment procedure. The recognition result is compared with a reference transcription, provided with the database. The accuracy can be defined as

$$Acc = \frac{H - I}{N} \cdot 100\%, \tag{4.17}$$

and the percentage of labels correctly recognised by

$$Cor = \frac{H}{N} \cdot 100\%, \tag{4.18}$$

where H is the number of phonemes correctly recognised, I is the number of insertions, D the number of deletions, and N the total number of labels in the defining transcription file.