Recurrent neural network-enhanced HMM speech recognition systems

by

Jan Willem Frederik Thirion

Submitted in partial fulfillment of the requirements for the degree

Master of Engineering (Electronics)

in the

Faculty of Engineering

UNIVERSITY OF PRETORIA

October 2001

To Carina, who makes every moment in life special.

Abstract

Keywords: speech segmentation, speech recognition, recurrent neural networks, extended recurrent neural networks, bi-directional recurrent neural networks, hidden Markov models

Speech recognition is fast becoming an attractive form of communication between humans and machines, due to the recent advances made in the field. The technology is, however, far from being acceptable when used in an unlimited form. Many attempts have been made to overcome some of the many difficulties faced in the automatic recognition of human speech by machine.

This dissertation attempts to accomplish two ambitious goals. The first is the reliable, automatic segmentation of speech signals, in a speaker independent manner, where no higher level lexical knowledge is used in the process. The system is limited to segmentation in an off-line manner. The second is to improve the phoneme recognition accuracy of a state-of-the-art hidden Markov model (HMM) based recognition system, using the segmentation information. A new technique of incorporating segmentation information into the Viterbi decoding process is presented, and it is shown that this technique outperforms other attempts to include the segmentation probabilities.

The segmentation system consists of a bi-directional recurrent neural network (BRNN), also called an extended recurrent neural network (ERNN). In contrast to conventional recurrent neural networks, that only use speech vectors from the past, present and possibly a fixed window of the future, BRNNs use all of the past, present and future speech vectors. The problem of the choice of the fixed size window is thus eliminated, at the cost of difficult real-time implementation. The BRNN can be trained using a modified form of backpropagation through time (BPTT). The input to the BRNN is the entire speech sentence and the two BRNN outputs are the probability that a boundary between phonemes occurs in that frame of speech, as well as the probability that no boundary occurs. The segmentation system segments the speech signal into phonemes, using only the speech signal, and no higher level lexical knowledge such as the sequence of phonemes.

The recognition system can incorporate the segmentation probabilities in one of two ways. The first is to modify the HMM transition probabilities by combining the HMM transition probabilities and the BRNN outputs. The second method, developed in this dissertation, involves the use of an adaptive word (phoneme) transition penalty. Previously, only a fixed transition penalty was used between words (phonemes). By making the transition penalty adaptive (based on segmentation information), the phoneme recognition performance can be significantly improved. It is also shown that the adaptive word transition penalty outperforms the HMM transition probability modification technique, used by others.

All of the experiments used in this dissertation are conducted on the TIMIT database, in order to provide a convenient way to compare the results to that of others in the field. The hidden Markov toolkit (HTK) from Cambridge University is used for all phoneme recognition experiments.

Uittreksel

Sleutelwoorde: spraaksegmentering, spraakherkenning, terugvoer neurale netwerke, uitgebreide terugvoer neurale netwerke, bidireksionele terugvoer neurale netwerke, verskuilde Markov modelle

Spraakherkenning is tans een van die mees populêre intervlakke tussen mens en masjien. Talle probleme word egter nog steeds met hierdie tegnologie ondervind, veral wanneer dit in 'n onbeperkte vorm gebruik word.

Hierdie verhandeling pak twee ambisieuse doelwitte aan. Die eerste is die betroubare, outomatiese segmentering van spraakseine. Die stelsel word beperk tot 'n aflyn taak. Die tweede doelwit is om die foneemherkenning akkuraatheid van 'n hoë verrigting verskuilde Markov model (VMM) gebaseerde herkenning stelsel te verbeter, deur van die segmentering inligting gebruik te maak. Hier word 'n nuwe tegniek, om die segmentering inligting in die Viterbi dekodering proses in te sluit, voorgestel en daar word gewys dat hierdie tegniek beter resultate lewer as ander metodes wat tans gebruik word.

Die segmenteringstelsel bestaan uit 'n bidireksionele terugvoer neurale netwerk (BTNN), ook 'n uitgebreide terugvoer neurale netwerk (UTNN) genoem. Konvensionele terugvoer neurale netwerke gebruik slegs spraakvektore van die verlede, hede en moontlik vaste venster van die toekoms. In kontras hiermee, gebruik BTNN'e al die inligting van die verlede, hede en die toekoms. Die probleem van die keuse van 'n vaste grootte venster word dus vermy ten koste van 'n moeilike intydse implementasie. Die BTNN kan afgerig word met 'n aangepaste weergawe van terugvoering deur tyd (TVDT). Die inset na die

BTNN is die volledige sin met spraak en die twee BTNN uitsette is die waarskynlikheid dat 'n grens tussen foneme in daardie raam van spraak voorkom, as ook die waarskynlikheid dat geen grens voorkom nie. Die segmenteringstelsel segmenteer die spraak in foneme deur slegs van die spraaksein gebruik te maak. Geen hoër vlak leksiese kennis, soos bv. die volgorde van foneme, word gebruik nie.

Die herkenningstelsel kan die segmentering waarskynlikhede in een van twee maniere insluit. Die eerste is om die VMM oorgang waarskynlikhede aan te pas deur die VMM oorgang waarskynlikhede en BTNN uitsette te kombineer. Die tweede metode, ontwikkel in hierdie verhandeling, maak van aanpasbare woord (foneem) oorgang penalisasie gebruik. Voorheen was slegs van vaste oorgang penalisasie tussen woorde (foneme) gebruik gemaak. Deur die oorgang penalisasie term aanpasbaar te maak (gebaseer op segmentering inligting), kan die foneemherkenning akkuraatheid aansienlik verhoog word. Daar word ook gewys dat aanpasbare woord oorgang penalisasie die VMM oorgang waarskynlikheid aanpassing tegniek, soos deur ander gebruik, oortref.

Al die eksperimente wat in hierdie verhandeling gedoen word, word op die TIMIT databasis gedoen, om sodoende 'n gerieflike manier daar te stel vir die vergelyking van resultate met dié van ander in die veld. Die stel van verskuilde Markov model programme (HTK) van Cambridge Universiteit word vir alle foneemherkenning eksperimente gebruik.

Acknowledgements

I would like to thank Professor E.C. Botha for allowing me the opportunity to study under her. She provided me with the academic guidance that made the work given here possible. It was an honour to learn from such a great intellectual.

Special thanks goes to Darryl Purnell for all the advice and discussions we had throughout the completion of my studies. His invaluable comments guided me towards achieving something that I am proud of. I also thank Willie Smit, with whom I had many discussions about concepts and ideas.

I am grateful to my family and friends, for their love, encouragement, and patience. I have great appreciation for my parents who coped with me, and always encouraged and supported me in my studies.

Finally, I would like to give my deepest thanks to my Saviour, Jesus Christ, for giving me the talents and gifts to be able to think.

Contents

List of Abbreviations

ADC	Analogue to digital converter
AM	Auditory modeling
AR	Autoregressive
ASP	Auditory speech processing
ASR	Automatic speech recognition
BCD	Bayesian changepoint detector
BRNN	Bi-directional recurrent neural network
CRNN	Chaotic recurrent neural network
DC	Direct current
DP	Dynamic programming
DTW	Dynamic time warping
ERNN	Extended recurrent neural network
FBDYN	Forward-backward auditory dynamic cepstra
FFT	Fast Fourier transform
HMM	Hidden Markov model
HTK	Hidden Markov toolkit
KL	Kullback Leibler
LP	Linear prediction
LPC	Linear predictive coefficient
MAP	Maximum a posteriori
MFC	Mel frequency cepstrum
MFCC	Mel frequency cepstrum coefficient
MIT	Massachusetts Institute of Technology
MLP	Multilayer perceptron
NIST	National Institute of Standards and Technology
NN	Neural network
OGI	Oregon Graduate Institute
PDF	Probability density function

PSM	Polynomial segment model
PSS	Pseudosyllables
RASTA	Relative spectral
RMLP	Recurrent multilayer perceptron
RMS	Root mean square
RNN	Recurrent neural network
S/M-R	Synchrony/mean-rate
SCVQ	Sequence-constrained vector quantisation
SDU	Semantic dialogue unit
SLAM	Segmentation and labeling automatic module
SRN	Simple recurrent network
SVF	Spectral variation function
TI	Texas Instruments
VCV	Vowel-consonant-vowel, or vocal-consonant-vocal
VQ	Vector quantisation

Chapter 1

Introduction

The process of communication between humans is mainly based on the ability to recognise and understand the speech signals transferred between them. Speech forms an integral part of the way humans interact with each other, as it is a highly effective and efficient way of exchanging information.

The automatic recognition of human speech by machine is regarded as a particularly difficult problem [1]. After decades of research, the goal of recognition of fluent, spontaneous speech, and the comprehension of its meaning, from any speaker in any environment, is far from being realised. The main success of speech recognition is due to the realisation that machines are not yet able to reach the performance of humans, and consequently, applying the technology only in a constrained way. This includes the use of only one speaker instead of many speakers (speaker dependent vs. speaker independent systems), the use of a small instead of a large vocabulary, and a well structured dialogue between man and machine, instead of an entirely open conversation. By realising the limitations of the current technology, and applying it only in an application specific way, speech recognition systems can be built that have acceptable performance.

Most speech recognition systems use some form of parametric model. The parameters of these models are usually estimated from a database of training data during a training phase. After training the recogniser, speech can be recognised, using the trained models. The recognition phase thus refers to the process of recognising which models correspond to different parts of the speech signal. Usually the models of the recogniser correspond to some subword unit, such as a phoneme, since the models for these subword units can be reliably estimated with a limited amount of data.

Speech is usually recorded and stored on a sentence by sentence basis. The purpose of speech segmentation is to determine the boundaries of the recognition units, so that each model of the recogniser can be trained with the correct segment of speech corresponding to the model. Speech segmentation also finds use in segmental speech recognition systems, where speech is first segmented, after which the segments are classified. Segmentation can thus be used only during the training phase, or during both the training and recognition phase. Other uses of speech segmentation include the determination of sentence boundaries, for the automatic creation of speech sentences (e.g. for the creation of speech databases) from continuous speech, such as broadcast audio, determination of word boundaries, improving recognition performance, etc.

The task of speech segmentation is also of critical importance for speech synthesis. Most successful speech synthesis systems today typically employ the use of segment concatenation of speech units (i.e. phonemes, diphones, syllables, etc.) from input training corpora of between 1 to 10 hours of speech. More natural speech synthesis is possible if effective segmentation can be performed to extract reliable synthesis units [2].

Speech segmentation algorithms can be broadly classified as belonging to one of two categories, namely those that make use of the underlying sequence of recognition units (i.e. forced alignment), and those that do not. In the first case only the boundaries need to be determined for a fixed number of specified units. In the latter case, the number of recognition units, as well as where the boundaries occur between them in time, are unknown.

This dissertation presents a recurrent neural network segmentation system, capable of segmenting speech into phonemes, in a speaker independent manner. The system does not make use of the underlying sequence of phonemes, as such a sequence is not always available or reliable (as in conversational speech). This is especially true for the phoneme recognition experiments conducted here, as the sequence of phonemes is unknown prior to the recognition process. It will also be shown how the locations of the phoneme boundaries are incorporated into the speech recogniser in a novel way, in order to improve the recognition performance of the baseline system.

1.1 Problem statement

A need exists for the reliable, automatic determination of speech subword unit boundaries. The incorporation of the information from the boundary locations into a baseline recogniser also needs to be investigated, as this could potentially improve recognition performance. The research given here thus aims to meet the following objectives, namely to

- provide a general system capable of segmenting pre-recorded speech signals in a speaker-independent manner, where the desired result is the location of phoneme boundaries (at the frame level),
- establish a baseline phoneme recognition system against which the methods developed here can be tested,
- incorporate the information of phoneme boundary locations into a phoneme recogniser, in order to improve the phoneme recognition performance, and
- develop a new technique of incorporating segmentation information in a phoneme recogniser.

In order to make the research viable, a number of assumptions must be made, including

- the use of an available, general purpose speech recognition system from Cambridge University, called the hidden Markov toolkit (HTK) [3], which will be used for all recognition experiments,
- evaluation of the methods only on American English (the TIMIT database), which reflects read speech and not spontaneous speech, and
- processing of speech only in an off-line manner (i.e. where all of the speech is available at all times).

1.2 Summary of related work

This section presents a concise literature survey of concepts related to those used in this dissertation. For a complete review of speech recognition theory, the book by Rabiner and Juang [1] is recommended.

The comparison of different speech segmentation and recognition algorithms is a difficult task. Researchers tend to use different databases and performance measures. Only results on the TIMIT database are thus given here, unless other results can provide further insight into a particular algorithm or technique.

The TIMIT database [4] was designed to provide speech data for the acquisition of acoustic-phonetic knowledge. It is also used for the development and evaluation of automatic speech recognition systems. The speech was recorded at Texas Instruments (TI), transcribed at the Massachusetts Institute of Technology (MIT), and maintained, verified, and prepared for CD-ROM production by the US National Institute of Standards and Technology (NIST). It is currently available from the Linguistic Data Consortium (LDC). This database contains a total of 6300 spoken sentences, where 630 speakers each spoke a total of 10 sentences. The 10 sentences are made up of 2 dialect sentences (SA), 5 phonetically compact sentences (SX) and 3 phonetically diverse sentences (SI). For all our experiments, the SA sentences were ignored. This resulted in

3696 training files and 1344 test files (from which a subset of 192 sentences makes up the core test set).

The concepts presented in this dissertation make use of three parts, namely speech signal processing, segmentation, and recognition. Each of these components is discussed in the following sections in more detail.

1.2.1 Speech signal processing

Speech signal processing is usually the first step in segmenting or recognising speech. The main aim of this stage is to provide features which are better suited to the segmentation or recognition process than the raw data. The total number of these features is typically much smaller than the total number of raw speech samples.

Many different signal processing methods exist. General speech signal processing methods are discussed in [5]. Mel frequency cepstrum coefficients (MFCCs) are discussed in [6], as well as generalised MFCCs. Linear prediction coefficients (LPC) are discussed in [7], while auditory nerve representation is discussed in [8], and the bandpass liftering of speech in [9]. Vector quantisation (VQ), usually used in discrete HMM systems, is discussed in [10] and [11]. An algorithm to estimate the fundamental frequency of speech is given in [12]. The following paragraphs give the specific features used in the segmentation methods of the next section.

Vorstermans *et al.* [13] used an auditory model that incorporates an auditory filter bank, a bank of hair-cell models that emphasised the transitions at the phonetic boundaries, and a bank of envelope detectors that measured the envelopes of the hair-cell outputs in the different channels of the model. An acoustic vector was constructed every 10 ms, that contained an auditory spectrum (20 channels), difference spectrum, voicing evidence, a fundamental frequency (if enough voicing evidence was present), and energy from an energy function sampled at multiples of 2 ms (obtained by accumulating the hair-cell output envelopes across the different channels).

Pauws *et al.* [14] used five measurements that took into account the fact that unvoiced sounds are generally characterised by an energy concentration in the relatively high frequency region, while voiced sounds have an energy concentration in the lower frequencies. Speech and silence were distinguished through the use of an energy level measurement. The five measurements thus included the short-time energy, normalised such that silence was characterised by a value close to 1, normalised low-frequency energy in the range of 50 to 1200 Hz, normalised high-frequency energy in the range 2000 to 4000 Hz, the zero crossing rate, and the first linear predictive coefficient of a first-order LPC model. The 16 filterbank values, their first and second derivatives, and energy were also used. Pre-emphasis was used, with a coefficient of 0.95, as well as a Hamming window of 20 ms length with frame shifts of 2.5 ms, 5 ms, and 10 ms.

Bonafonte *et al.* [15] calculated an acoustic vector every 10 ms, by analysing speech frames of 20 ms using a Hamming window. For each frame, 20 mel-scaled filters were transformed to 12 MFCCs and a measure of the power, as well as their first and second derivatives were calculated. Only the 12 MFCCs were used to refine the boundary positions.

Olsen [16] used 31 triangular filters spaced linearly along the logarithmic mel scale, where each filter overlapped 50% with its two neighbours. Normalised log energy was also used. All the feature vectors were obtained from a 25.6 ms Hamming window with a 10 ms frame period. A total of 15 MFCCs, normalised log energy, and their first and second order derivatives, were used.

Lee [17] used 14 MFCCs with log energy, computed at 5 ms intervals. Pellom and Hansen [2, 18] parameterised the speech waveform every 5 ms, by a vector consisting of 12 MFCCs and normalised log-frame energy, as well as their first derivatives.

Jeong and Jeong [19] used a 256 point rectangular window, spaced at half a window size, and the LPC Burg algorithm for cepstrum analysis. The acoustic vector was then constructed from 16 whitened LPC cepstrum coefficients, obtained by using the whitening method. Policker and Geva [20] also used 12 LPC coefficients.

Cosi [21, 22, 23] allowed the use of many different kinds of features in an interactive segmentation and labelling automatic module (SLAM). The joint synchrony / mean-rate (S/M-R) model of auditory speech processing (ASP), fast Fourier transform (FFT) cepstrum, LPC based spectrograms, energy, pitch, and zero crossing, are some of these. The use of auditory model (AM) techniques was strongly supported.

Smith [24, 25] used an auditory front end, which used a Gammatone filter bank that bandpassed the signal into a number of channels. These were then rectified to model the effect of a set of inner hair cells.

Chang *et al.* [26] calculated a feature vector every 10 ms after several stages of processing. The first step was to compute a power spectrum every 10 ms over a 25 ms window. The power spectrum was then partitioned into quarter-octave channels between 0.3 and 4 kHz and logarithmically compressed in order to preserve the general shape of the spectrum distributed across frequency and time.

Regel [27] used normalised energy (frequency range 100 to 900 Hz), logarithm of the normalised energy, normalised autocorrelation coefficient at unit delay, first linear prediction (LP) coefficient, logarithm of the normalised LP error, and normalised amplitude, frequency, and bandwidth of the absolute maximum in the spectrum, to classify speech into one of a few broad categories ("silence", "voiceless", "voiced fricative", and "voiced non-fricative"). For classification of frames into the phone components, knowledge of the first classification stage allowed special features to be used for each of the broad categories. For the category "voiced non-fricative", normalised energy, logarithm of the normalised energy (frequency range 640 to 2800 Hz), normalised autocorrelation coefficient at unit delay, and normalised amplitude, frequency, and bandwidth of the lowest three formants were used. For voiceless sounds, the logarithm of the normalised LP error, normalised energy, logarithm of the normalised energy in five non-overlapping frequency ranges, normalised autocorrelation coefficient at unit delay, and normalised amplitude, frequency, and bandwidth of the absolute maximum in the spectrum were used. For the voiced fricative sounds, the same features were used as for voiceless sounds, except for normalised amplitude, frequency, and bandwidth of the absolute maximum in the spectrum.

Fukada *et al.* [28] calculated an acoustic vector every 10 ms. The acoustic vector included the 12 MFCCs, power, and the first derivatives of these. A window of 25.6 ms was used.

In conclusion, it can be seen that many different speech features have been used in the past. MFCCs and energy have however proven to be the choice in recent years, not only for segmentation but also the recognition of speech. In the work presented here, the same MFCC and energy features will thus be used for segmentation and recognition tasks.

1.2.2 Speech segmentation

Linguistically constrained (explicit) segmentation

When the underlying sequence of phones is known, the segmentation algorithm only has to calculate the location in time of the boundaries between the phones (i.e. referred to as "forced alignment"). These methods perform reasonably well, as the higher level of lexical information (phoneme sequence) is used in the segmentation process. It is important to note that in most cases text information is provided, so reliable word to phoneme sequence look-up is necessary. This also assumes that speech production of the word set occurs without alternative pronounciations, otherwise the segmenter must consider alternative pronounciations during processing. The following is a short summary of some of these methods.

Vorstermans *et al.* [13] developed a system for the automatic segmentation and labelling of speech. The system first did initial segmentation by identifying major changes (landmarks) in the acoustic signal obtained from an auditory model. This was achieved by a landmark identification, generation and elimination stage. Up to 4 consecutive initial segments were then merged to construct a set of candidate phonetic segments. A multilayer perceptron (MLP) was subsequently used in a phonetic segmentation stage to compute the probability of a boundary being a phonetic boundary, given the evidence of a preceding phonetic boundary and acoustic evidence. Phonetic classification was then performed through the use of another MLP that classified the acoustic vector into one of 5 broad phonetic classes, where the MLP's outputs were also interpreted as probabilities. A Viterbi search procedure aligned the speech with a state-transition model, derived from the transcription of the utterance, by taking the outputs of the two MLPs in the phonetic segmentation and classification stages into consideration. The result of the Viterbi search was a set of boundaries and labels, that maximised the combined likelihood of the phonetic boundaries and phone sequence, given the provided transcription and acoustic observations. The system was easily adaptable from one language to another, without the requirement of extensive linguistic knowledge or large (manually segmented and labeled) training databases of that language. A correct boundary placement of 76% (within 20 ms of the desired boundary location) was obtained on the core test set of the TIMIT database (5 SX sentences per speaker), with 48 phone labels. The speech was aligned against the manual transcriptions (only using the labels, not the manually provided boundary locations) of the TIMIT database, after adapting the baseline Flemish system, using 100 sentences of the TIMIT training set. A further gain of 5% on the overall system performance was achieved by also using the manually provided boundary locations, with about 200 training sentences, but it was not stated what the gain in segmentation accuracy was.

Pauws *et al.* [14] made use of the time alignment of the speech waveform against a sequence of HMMs, where each HMM represented a phoneme-like unit in the phonetic transcription of the utterance. Initialisation of the HMMs was performed by a 3-stage hierarchical procedure. The first stage involved the segmentation into broad phonetic classes (voiced, unvoiced and silence), on the basis of the phonetic transcription alone. This provided robust anchor points for the second stage, namely sequence-constrained vector quantisation (SCVQ), where the broad phonetic class regions were further de-composed into their constituent phoneme-like units. Finally Baum-Welch estimation was used to fine-tune the HMMs. Segmentation was then performed by the Viterbi alignment of the utterances with the HMMs. An accuracy of 89.51% was obtained for a 20 ms tolerance, and 95.37% for a 30 ms tolerance, on a database consisting of 827 isolated words of the Dutch language. Learning was performed on the database to be segmented.

Bonafonte *et al.* [15] used hidden Markov models and the Viterbi algorithm to obtain an initial segmentation of the speech. A corrective procedure was then applied, which considered the segments of the segmented speech as homogeneous regions. A model was estimated for each segment of the utterance and Gaussian probability density functions (PDFs) were used to model the feature vector. Hypotheses for moving the boundary one frame to the left or to the right were then analysed. Boundaries were iteratively moved until no further changes occur. The result was that the boundary positions were refined and segmentation error was significantly decreased. They obtained an accuracy of 64.4% with a 12 ms window, and 81.3% with a 20 ms window, on the TIMIT database.

Olsen [16] also used hidden Markov models and Viterbi decoding of the speech utterance to segment the utterance, as well as Lee [17]. They did not report any specific segmentation results, and thus none of their results are given here.

Pellom and Hansen [2, 18] used dynamic programming (DP) to investigate the effect of different signal processing methods on the segmentation accuracy. Here the effect of noise on the segmentation performance was also investigated. They achieved segmentation accuracies of 47.9%, 69.9%, 85.9%, 95.9% and 98.4% for tolerances of less than 5 ms, 10 ms, 20 ms, 40 ms, and 60 ms, respectively, on the TIMIT database.

Jeong and Jeong [19] used a higher order Markov process, and the mean field solution to the segmentation problem, in a closed loop system consisting of combined bottom-up (segmentation, recognition and labelling) and top-down (labelling, speech generation and segmentation) processing. A recursive procedure provided an estimation of the segmentation and phone label. Their system transformed the incoming continuous signal into one of the 61 phone classes at the rate of 73.7% when TIMIT was used.

Cosi [21, 22, 23] developed an interactive segmentation and labelling automatic module (SLAM). A multi-level segmentation theory was used. Speech was considered as a temporal sequence of quasi-stationary acoustic segments, where similarity of points in a segment is greater than for those between different segments. The segmentation problem was thus reduced to a clustering problem, where a decision was taken based on the similarity between the signal immediately preceding and following it. Initial "seed regions", which constitute the basis for the "hierarchical structuring", were created by a recursive technique that used a Euclidean similarity measure. Adjacent regions were then merged and a dendrogram was constructed. Pattern recognition techniques found the optimal segmentation path given the dendrogram structure and the target phonemic transcription.

Linguistically unconstrained (implicit) segmentation

When the underlying sequence of phonemes is unknown, the segmentation algorithm must not only estimate the location in time of the boundaries between phonemes, but also the number of boundaries (or alternatively the number of phones). These methods generally perform worse than those of the previous section, but are more versatile. The following is a short summary of some of these methods.

Smith [24, 25] used a general sound segmentation system to segment speech into phonemes. The sound signal was bandpass-filtered into a number of channels, rectified to model the effect of a set of inner hair cells, and filtered using an onset/offset filter. This made the transformed representation sensitive to energy rises and falls. The next step was to divide the onset/offset representation into two positive-going signals, an onset signal and offset signal. Both of these signals were then logarithmically compressed to increase the dynamical range of the system. These signals were sharpened with an integrate-and-fire neural network, where the data was integrated across frequency bands and across time. The effect of this was to produce sharp onset firing responses across adjacent channels in response to a sudden increase in energy in some channels, thus grouping onsets both tonotopically and temporally. The outputs of the neural network sharpening stage were the onset and offset maps. The onsets were used for segmentation as the offsets tended to be more gradual and the continuous signal was divided at each onset. A minimum segment length of 25 ms was used, and the sharpness of the segmentation was varied by setting the minimum number of onset (offset) spikes which had to occur in the 10 ms window before that onset or offset line was taken to signal a segment start (end). Two male and two female sentences from each of the 8 dialect regions of TIMIT were segmented using this method. An average of 59% of the phoneme boundaries were correctly found (the estimated boundary and true boundary of a phoneme was within 15 ms of each other).

Chang *et al.* [26] used an array of independent, temporal flow neural networks that classified each frame into one of five articulatory-based phonetic-feature classes, namely place, manner of articulation, voicing, lip-rounding, and front-back articulation (for vocalic segments). They used a separate class for silence. These phonetic-feature labels were then combined and used as the input to an MLP network that gave a preliminary phonetic label to a frame. The last stage was a Viterbi-like decoding process that produced a sequence of phonetic-segment labels along with the times of the boundaries between them. They achieved 38.4%, 76.0% and 83.7% hits, and 58.5%, 20.9% and 13.2% false alarms, for a frame tolerance of 10 ms, 20 ms and 30 ms, respectively on the Oregon Graduate Institute (OGI) Numbers95 corpus.

Regel [27] used two classification stages in an acoustic-phonetic transcription system. In the first stage a decision was made in favour of one of four categories, namely "silence", "voiceless", "voiced fricative", and "voiced non-fricative". A Bayes classifier was used for this purpose. The second stage consisted of the classification of the frames into phone components, using the results of the first step. In this stage only special features were used for each class. A Bayes classifier was also used for this purpose. The resultant probabilities of the two stages were then multiplied together to obtain an estimate of the *a posteriori* probability of the phone component. Adjacent frames were then combined in a two-stage process. In the first step, similar frames were lined up with simultaneous smoothing. In the second step, essential segments were extracted and attempts made to fill the gap between two essential segments, using a similarity measurement. The results given are not directly comparable to the work presented here, and are thus not given.

Policker and Geva [20] regarded the speech signal as a non-stationary time series. They developed a model and a set of algorithms to estimate the parameters of the non-stationary time series. Fuzzy clustering methods were used to estimate the continuous drift in the time series distribution and to interpret the resulting temporal membership matrix as weights in a time varying, mixture probability distribution function. A decision rule was imposed on the distribution temporal change, where a limiting procedure of any cluster crossing the 0.5 probability level, was used. This resulted in segmentation of the speech signal into phonemes. No specific results were given.

Andre-Obrecht [29] (also in Basseville and Nikiforov [30]) used a statistical approach. [1] The signal was modeled by an autoregressive (AR) statistical model. Test statistics were then used to sequentially detect changes in the parameters of the model. Three different segmentation algorithms were presented, differing in the assumption of the excitation of the model (glottal impulses), and the choice of the test statistics (generalised likelihood, or statistics of cumulative sum type). The segmentation was performed on a sample-by-sample basis, and not a frame-by-frame basis, allowing more accurate location of boundaries, and the possibility of using shorter segments. Their results are not directly comparable to the work presented here, and are thus not given.

A Bayesian autoregressive changepoint detector (BCD) was used by Èmejla and Sovka [31]. A three-step algorithm was used to segment the speech. In the first step, a segmenta-

[1]Examples of some of these methods can also be found on the Internet at http://www.cnmat.berkeley.edu/~tristan/Thesis/timedomain.html

tion point was assigned to the centre of sound units composed of vowels and semivowels. This was done because the BCD was highly sensitive to spectral changes and was rather used to refine the positions gained from the first step. By using segmentation points inside stationary parts of the signal, the autoregressive order to the left and right sides of the data segment could be estimated with higher accuracy. The second step used a BCD between each pair of given segmentation points in the stationary parts of the speech. The final step used a BCD with data between the stationary segmentation point and the segmentation point gained in the second iteration. This method was suitable for vocal-consonant-vocal (VCV) structured utterances. No comparable results to the work presented here were given.

Petek *et al.* [32] investigated the robust automatic segmentation of spontaneous speech. They used the spectral variation function (SVF), which was defined as a correlation measure between successive windows of acoustic observation vectors, to segment the speech. They compared mel-frequency cepstra (MFC), relative spectral processing (RASTA), and forward-backward auditory masking dynamic cepstra (FBDYN) based SVF algorithms. The FBDYN-SVF method resulted in smoothing of the cepstra by the forward and backward masking lifter, giving an improvement over the other two methods. Their numerical results are not comparable to the work presented here, and are thus not given.

Fukada *et al.* [28] used a bi-directional recurrent neural network (BRNN) to segment speech. The system was trained to segment the speech signal into phonemes, using a target value of 1 for a frame in which a boundary occurred, and a target value of 0.5 for the frames to the left and right of the boundary frame. Frames in which no boundary occurred were given the target value of 0. The neural network estimated the probability of a boundary, given the acoustic vector. By using thresholds or a segment lattice, the segmentation points in the speech signal could be found. They found that the BRNN segmented the TIMIT database with 8.33%, 76.01%, and 79.61% accuracy for frame margins of 0, 1, and 2, respectively. They also found that normal MLP neural networks performed worse. An MLP with 1 context frame segmented the TIMIT database with

1.46%, 61.90%, and 68.64% accuracy, an MLP with 3 context frames 6.12%, 64.16%, and 70.94%, and an MLP with 5 context frames 6.20%, 64.69%, and 71.64% for frame margins of 0, 1, and 2, respectively.

Related segmentation tasks

This section briefly highlights research done in related segmentation tasks. These tasks perform segmentation at a level higher than the phoneme level, or in a limited way.

At the subword level, a number of other segmentation attempts are worth mentioning. Chan and Ng [33] discussed the separation of fricatives from aspirated plosives by means of temporal spectral variation. Ryeu and Chung [34] used chaotic recurrent neural networks (CRNN) to classify and segment Korean monosyllables. De Mori and Laface [35] used fuzzy algorithms to segment speech into vowel-consonant-vowel (VCV) pseudosyllables (PSS). In Shyu *et al.* [36], an automatic co-articulation segmentation algorithm was developed, that took co-articulation into account. Co-articulation was also taken into account by Yu and Oh [37], where a neural network (NN) was used to segment speech into non-uniform units. Co-articulation information and neural networks were also used by Hosom and Cole [38], where the neural networks segmented speech into diphones. Steady-state zones of all phones carrying a diphone boundary were specified by a centroid vector, and together with an objective distance measure, hypothetical boundary cost functions were used to extract diphone elements in Kaeslin [39]. Temporal flow neural networks were used by Shastri *et al.* [40] for finding the temporal boundaries of syllabic units. Hsieh *et al.* [41] also segmented speech into syllables, using a hybrid neuro-fuzzy network. Cook and Robinson [42] used an MLP to determine the onset of syllables.

The segmentation of speech into voiced, unvoiced, silence, and/or mixed regions of speech can also be done. Examples include [43, 44, 45, 46, 47, 48, 49, 50, 51].

Segmentation can also be performed at the word level. Zelinski and Class [52] segmented an utterance into single words, using statistical principles. Christiansen *et al.* [53, 54] used a simple recurrent network (SRN) for this purpose.

At the sentence level, Siegler *et al.* [55] used the Kullback Leibler (KL) distance metric to segment broadcast news audio into discrete utterances. Prosody-based automatic segmentation of speech into sentences and topics was investigated by Shriberg *et al.* [56]. Semantic dialogue unit (SDU) segmentation, which correspond roughly to speech act (utterance) segmentation, was provided by a multi-level segmentation algorithm in Lavie *et al.* [57]. Speech act detection was also performed by Ries [58], using HMM and neural network based methods. Swerts and Ostendorf [59] investigated prosodic and lexical indications of discourse structure and utterance purpose. Tzanetakis and Cook [60, 61] developed a framework for audio analysis based on classification and temporal segmentation.

Segmentation at even higher level is also possible. A syntactic-prosodic labelling scheme was developed in Batliner *et al.* [62] that could segment speech into sentences or phrases. Renals *et al.* [63] developed a system that could segment speech into stories, for the indexing and retrieval of broadcast news. Speaker-based segmentation system for audio data indexing was performed by Delacourt and Wellekens [64]. Energy-based speech endpoint detectors were compared in Bush *et al.* [65].

1.2.3 Speech recognition

This section does not focus on background of hidden Markov models, neural networks, fuzzy logic, or other methods as general techniques. Instead, a very brief literature survey of the application of these methods to speech recognition, is presented.

For a detailed review of hidden Markov models, see [1, 66, 67, 68]. For neural networks, [69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80] can be recommended. Fuzzy logic was also used frequently and background on the underlying theory and techniques can be found in [81] and [82]. In addition to fuzzy logic, [83] also explains neuro-fuzzy concepts.

Artificial intelligence concepts are explained in [84]. Numerical methods used can be found in [85, 86, 87]. Statistical theory can be found in [88] and [89], while general pattern recognition concepts can be found in [90].

Hidden Markov models have proven to be very well suited to recognising human speech. Some of the main advantages of HMMs are that they integrate well into systems incorporating both task syntax and semantics. Examples of HMM-based speech recognition include [1, 66, 67, 91, 92, 93, 94, 95, 96]. Related to hidden Markov models is the concept of dynamic programming, such as [97] and [98]. Dynamic time warping (DTW), as well as a probabilistic matching algorithm was used by [99]. Similar concepts, some involving template matching, were also used by [100, 101, 102]. Fuzzy logic concepts were used by [103] and [104]. In using hidden Markov models, a number of assumptions are made. These include the assumptions that the speech signal can be well characterised as a parametric random process, that the parameters can be estimated in a precise, well-defined manner, and the fact that a first order Markov chain is usually used ([105] showed how higher-order HMMs can be used efficiently). Conventional HMM systems make use of an independence assumption of the observation.

Neural networks have also found their way into the area of speech recognition. This is partially due to the thin biological connection that exists between neural networks and the human brain, and the fact that neural networks operate well as pattern classifiers and can estimate probabilities conveniently. Time delay neural networks were used in [106] and [107]. Neural-fuzzy concepts were combined with an HMM-based automatic speech recognition system in [108]. Spiking neural networks were used in [109] and [110]. Simple recurrent neural networks, also called Elman neural networks, were used in [111] and [112] to discover symantic/semantic features of words, and in [113] was used for speech recognition, where the neural networks were trained with the leap-frog algorithm. Recurrent neural networks were used in [114, 115, 116, 117]. In [118] various different neural networks are discussed for use in the context of speech recognition. An advantage of neural networks for speech applications is that they are general and do not impose a rigid structure into the recognition process. Some weaknesses include the increased size of training material over traditional HMMs, and their inability to allow for efficient adaption to changing noise conditions versus methods seen in HMMs. See [119] and [120] for more detailed information on the use of neural networks in speech recognition.

It is often the case that the best systems are hybrid ones. In speech recognition, hybrid systems perform particularly well. In [120] it is discussed how neural networks might be incorporated into HMM systems. Usually neural networks are used to estimate the observation density in the HMM states, or the *a posteriori* probability of a certain phone, given acoustic evidence. In [121] it was shown how the *a posteriori* probability of a complete utterance could be estimated, as an alternative approach to the regular split into acoustic model and language model likelihood. A bi-directional recurrent neural network estimated the occurring probability terms. In [42] syllable boundary information was included to improve the recognition process. Their method made use of two models for each phone, one model when the phone occurs at a syllable onset, and one when it does not. In [28] the transition probabilities of the HMMs were modified by a BRNN output. They also showed how the neural network could be used with a polynomial segment model (PSM) based recogniser. PSM based recognition systems do not rely on the observation independence assumption of conventional HMM systems. The neural network was used to segment the utterance into segments, where the boundary locations were determined by a segment lattice as a postprocessor.

1.3 Approach and research hypotheses

The work presented here addresses two types of problems, namely speech recognition and speech segmentation. It is also shown how information from the independent segmentation stage can be used to improve speech recognition.

Figure 1.1 gives an overview of the approach followed here. It consists of two independent components, namely a speech recognition component, and a speech segmentation

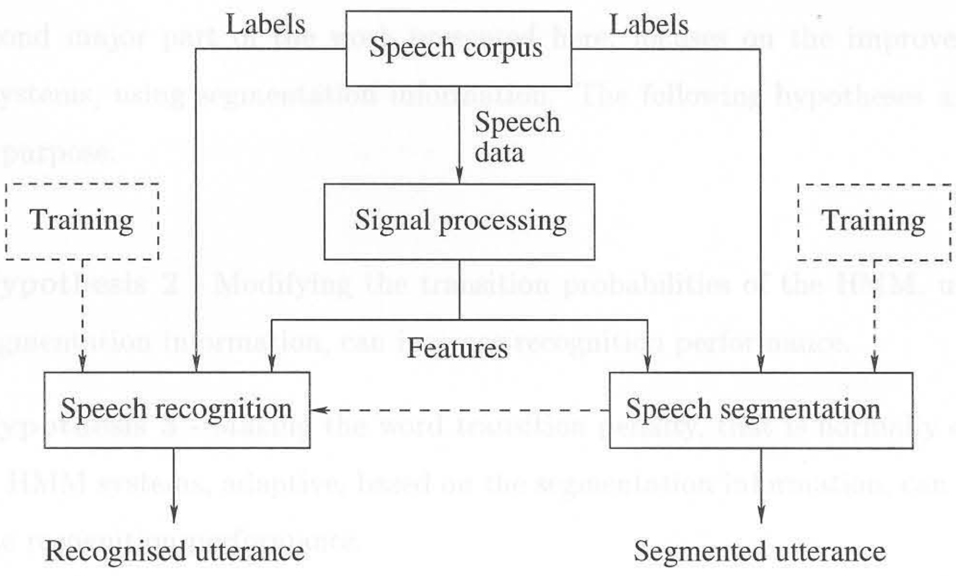

Figure 1.1: Overview of the approach taken.

component. A speech corpus contains label files and speech data files from a large number of speakers. These are used to make the system speaker independent. The signal processing module extracts features from the speech data that are useful for the recognition and segmentation stages. For simplicity, the recognition and segmentation stages presented here use the same set of features. Initially the recognition and segmentation components are trained independently using the corpus, and evaluated separately. Segmentation information is then used to improve the recognition performance of the entire system. American English (the TIMIT database) is used as the speech database.

As mentioned, one of the problems investigated here, is the problem of speech segmentation. It is known that hidden Markov model systems perform segmentation automatically as part of the Viterbi decoding process. Neural networks can also be used for the segmentation process. This leads to the following hypothesis:

- **Hypothesis 1** - Recurrent neural networks will perform the segmentation task (into phonemes) better than HMMs.

The second major part of the work presented here, focuses on the improvement of HMM systems, using segmentation information. The following hypotheses are made for this purpose:

- **Hypothesis 2** - Modifying the transition probabilities of the HMM, using the segmentation information, can increase recognition performance.

- **Hypothesis 3** - Making the word transition penalty, that is normally constant in HMM systems, adaptive, based on the segmentation information, can increase the recognition performance.

1.4 Contributions of this study

The work presented here offers a number of contributions. Not only is a new technique presented, but both old and new techniques are evaluated using American English (TIMIT) speech. Before this dissertation, the use of segmentation information in the recognition process of a standard speech recognition system, was not commonly used. The contributions of this dissertation include

- a high performance speech segmentation system, involving the use of a recurrent neural network, capable of segmenting speech into phonemes, without the use of higher-level lexical knowledge,

- a new technique of incorporating segmentation probabilities into the speech recognition system, in order to improve phoneme recognition performance,and

- evaluation of the methods, of incorporating segmentation information into HTK, on the TIMIT database, is presented in order to see how well they perform in a state-of-the-art speech recognition system.

It is shown that the new technique developed here outperforms some techniques used by others. It is also shown that the technique is fairly efficient and can be easily incorporated into a standard recogniser.

1.5 Dissertation outline

Chapter 2 presents the background theory relevant to the work presented here, in speech signal processing (2.1), hidden Markov models (2.2), and recurrent neural networks (2.3).

Chapter 3 provides detail about the segmentation process. Aspects discussed include preprocessing (3.1), segmentation (3.2), using hidden Markov models (3.2.1) and recurrent neural networks (3.2.2), postprocessing (3.3), and the accuracy measure used (3.4).

Chapter 4 deals with speech recognition. Baseline recognition (4.1), recognition using segmentation information (4.2), and the accuracy measure used (4.3), are discussed.

Chapter 5 gives the experimental results. In this chapter the concepts of Chapters 3 and 4 are evaluated, using the defined accuracy measures. Speech segmentation (5.1) methods include the use of hidden Markov models (5.1.1), as well as recurrent neural networks (5.1.2). Recognition experiments include the baseline system (5.2) and recognition using the segmentation information (5.3).

Chapter 6 gives a summary and some conclusions are made. A summary of results (6.1), statistical significance testing (6.2), conclusions (6.3) and shortcomings and future work are discussed (6.4).

Finally, Appendix A gives the details on training recurrent neural networks. The back-propagation through time (BPTT) technique is discussed in the context of recurrent neural networks, and it is shown how bi-directional recurrent NNs (BRNN) are trained.

i16011028
b15413391

Chapter 2

Theory

The work presented here makes use of techniques from various disciplines. The aim of this chapter is to provide the necessary background theory to understand the work presented later. The experienced speech researcher would be familiar with these concepts and therefore could move to Chapter 3 where the problem of speech segmentation is addressed. In Section 2.1 the signal processing techniques, used to extract features from the speech signal are discussed. Hidden Markov models are used for the phoneme recognition experiments and the relevant theory is discussed in Section 2.2. Finally recurrent neural networks are discussed in Section 2.3, as they are used to segment the speech signal into phonemes. This section should be read in conjunction with Appendix A where details on training of the recurrent neural networks are presented.

2.1 Speech signal processing

The human vocal tract mechanism is unable to instantly produce different sounds in speech. Due to the transition-like nature between sounds, there is a significant amount of correlation between segments of speech. By using parametric representations of the speech, rather than the speech signal itself, a more compact, reliable, and robust speech recognition system can be built. The use of parametric representations also reduces the amount of computation needed for both training and decoding. The following front-end is typically used in speech recognition systems.

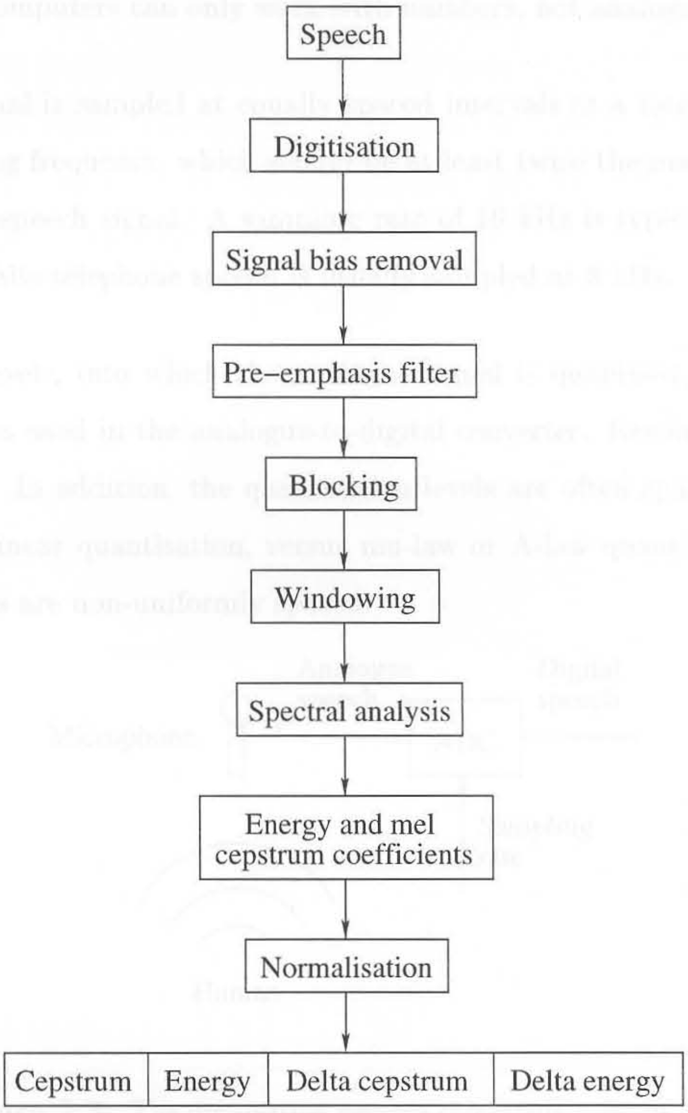

Figure 2.1: Signal processing front-end used.

The following sections describe each of these signal processing steps, in terms of the work presented here, in more detail.

2.1.1 Digitisation

Speech digitisation is the process of converting analogue signals to digital values (numbers), as digital computers can only work with numbers, not analogue values.

The analogue signal is sampled at equally spaced intervals at a rate equal to what is called the sampling frequency, which should be at least twice the maximum frequency of interest in the speech signal. A sampling rate of 16 kHz is typically used for high quality speech, while telephone speech is usually sampled at 8 kHz.

The number of levels, into which the analogue signal is quantised, is dependent on the number of bits used in the analogue-to-digital converter. Resolutions of 12 to 16 bits are common. In addition, the quantisation levels are often spaced equally. This is referred to as linear quantisation, versus mu-law or A-law quantisation, where the quantisation levels are non-uniformly spaced.

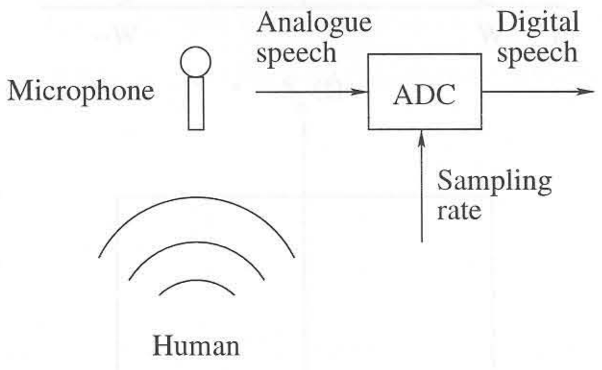

Figure 2.2: The digitisation process commonly used in ASR.

As can be seen from Figure 2.2, a human thus speaks a sentence to be recognised by the system. The speech signals are transmitted as vibrations of air to a microphone, which then converts the air pressure variations into electrical signals. The result is an analogue signal representing the spoken utterance. An analogue anti-aliasing filter (not shown) prevents aliasing effects, but this is not discussed here, and assumed to be part of the analogue-to-digital conversion (ADC) process. An ADC converts the analogue signal to a sequence of numbers (digital), to be processed by a computer.

2.1.2 Pre-emphasis filter

One of the problems that a speech recognition system must face, is the fact that a message (sequence of spoken sounds) will have frequency components that are high at low frequencies, but small amplitude at high frequencies [122]. This is also sometimes referred to as the lip effect. This also occurs because the speech signal is typically represented as a volume velocity [68], but microphones typically measure sound pressure. This difference results in a 6 dB spectral roll-off.

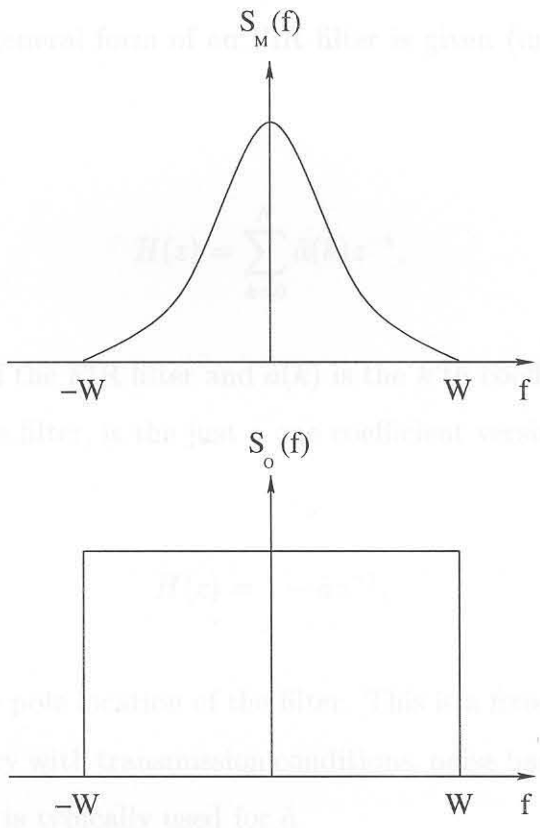

Figure 2.3: Purpose of the pre-emphasis filter.

The top part of Figure 2.3 above illustrates this concept. Here it can be seen that the power spectral density of the message usually falls off appreciably at higher frequencies. These high frequency components (that may carry vital information for accurate speech recognition) will thus be of a much lower magnitude than those at low frequencies. As such, the speech recognition process will not yield the best possible results. It is thus needed to equalise the frequency band over which important signal frequency components in the message exist, to obtain a spectrally flat frequency spectrum, as shown in the bottom part of Figure 2.3. This procedure is called pre-emphasis. Pre-emphasis reduces the effects of glottal pulses and radiation impedance, and enhances the spectral properties of the vocal tract [123, 124]. This makes the signal less susceptible to finite precision effects later in the signal processing front-end [1].

In order to perform equalisation, or pre-emphasis, a high-pass filter is usually used. Since the digitised speech samples are processed sequentially, an FIR filter is a natural implementation. The general form of an FIR filter is given (in the z-transform form) as

$$H(z) = \sum_{k=0}^{N} \tilde{a}(k) z^{-k}, \tag{2.1}$$

where N is the order of the FIR filter and $\tilde{a}(k)$ is the k'th coefficient, and $\tilde{a}(0) = 1$. A high-pass, pre-emphasis filter, is the just a one coefficient version of this, or

$$H(z) = 1 - \tilde{a}z^{-1}, \tag{2.2}$$

where $\tilde{a}$ determines the pole location of the filter. This is a fixed form of pre-emphasis, as $\tilde{a}$ does not slowly vary with transmission conditions, noise backgrounds, etc. A value of between 0.9 and 1.0 is typically used for $\tilde{a}$.

The easiest way to implement the filter of Equation (2.2), is to convert the z-transform equation into a difference equation (in the time domain). It is known that (from Equation (2.2))

$$H(z) = 1 - \tilde{a}z^{-1},$$

but

$$H(z) = \frac{\widetilde{S}(z)}{S(z)},$$

and thus it is possible to write

$$\widetilde{S}(z) = S(z) \cdot (1 - \tilde{a}z^{-1}),$$

with the difference equation which follows as

$$\tilde{s}(n) = s(n) - \tilde{a}s(n-1), \tag{2.3}$$

where $\tilde{s}(n)$ is the output of the pre-emphasis filter, $s(n)$ is the input, and $s(n-1)$ is the previous (one step delayed) input to the pre-emphasis filter.

Although pre-emphasis, as discussed here (using a high-pass filter), is a necessary step and greatly improves speech recognition performance, it has a major disadvantage. While the filter spectrally flattens the range of frequencies of interest, in which useful information lies, it also raises the spectral energy of high frequencies outside the signal bandwidth [6]. These very high frequencies are often associated with noise, and pre-emphasis thus also amplifies noise in the signal. Despite this disadvantage, it remains to be popularly used.

2.1.3 Blocking

After the signal has been pre-emphasised, the next step is to chop the signal into smaller pieces. These smaller pieces are called frames. Usually the frames are chosen as overlapping segments of speech, in order to reduce the noise in spectral estimates (increase correlation between adjacent frames). Figure 2.4 illustrates this process.

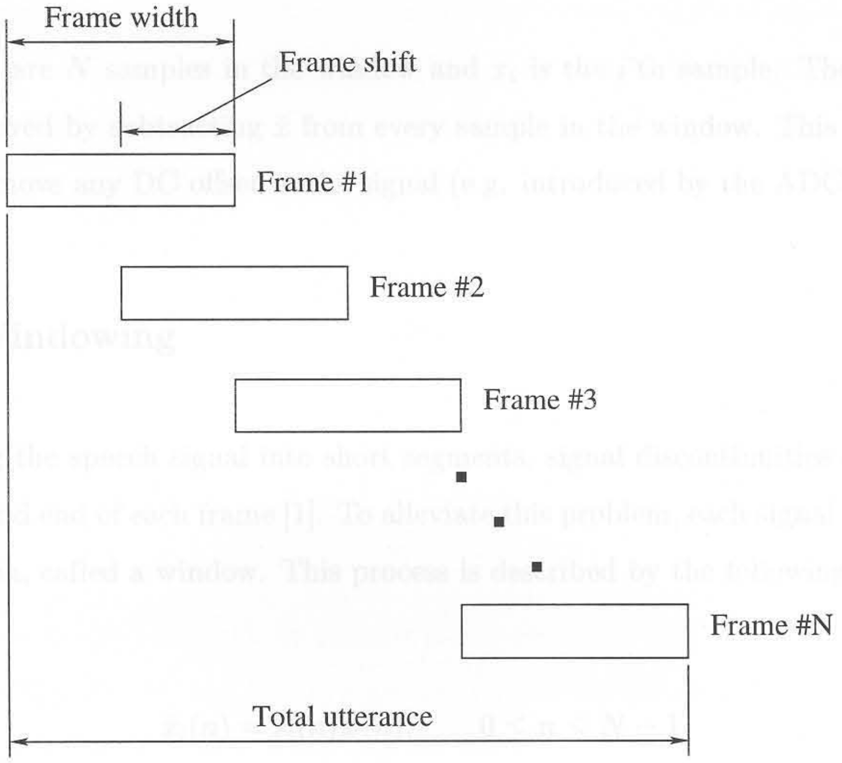

Figure 2.4: Blocking of the speech signal into overlapping frames.

The frame width is usually chosen to be between 20 and 30 ms with a frame shift (or period) of 5 to 10 ms. The bigger the window width, the better the spectral resolution, but the lower the time resolution. A low frame shift results in good temporal resolution, but is computationally expensive.

2.1.4 Signal bias removal

Signal bias removal refers to the process whereby the mean value (here called the signal bias), is removed from the signal. To estimate the mean, the average value of the signal is calculated over the current window through the following equation

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{2.4}$$

where there are N samples in the window and x_i is the i'th sample. The signal bias is then removed by subtracting $\bar{x}$ from every sample in the window. This technique is useful to remove any DC offset in the signal (e.g. introduced by the ADC).

2.1.5 Windowing

By blocking the speech signal into short segments, signal discontinuities occur at the beginning and end of each frame [1]. To alleviate this problem, each signal is multiplied by a function, called a window. This process is described by the following equation:

$$\tilde{x}_l(n) = \tilde{s}_l(n)w(n), \qquad 0 \le n \le N-1, \tag{2.5}$$

where $\tilde{x}_l(n)$ is the resultant, windowed signal, $\tilde{s}_l(n)$ is the original signal, $w(n)$ is the window with which the frame is multiplied, and N is the number of samples in the frame. The window with which the frame is multiplied, is typically chosen to be the Hamming window, defined by the following equation [1]:

$$w(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{N-1}\right), \tag{2.6}$$

for $0 \le n < N$ and $w(n) \equiv 0$ elsewhere. Here $w(n)$ is called the window, and N is the number of samples in the frame with which it should be multiplied.

The purpose of the window is to give more emphasis to samples in the centre of the window. Combined, the overlapping frame analysis and windowing, ensures that smoothly varying parametric estimates can be obtained.

2.1.6 Spectral analysis

For the spectral analysis of the speech signal (the blocked frames of speech multiplied by the window), a Fast Fourier Transform (FFT) of the signal is first computed (called a short-time FFT, because the FFT is taken only over short time segments). This procedure is based on the two equations [86]

$$W \equiv e^{2\pi i/N}, and \tag{2.7}$$

$$Y_n = \sum_{k=0}^{N-1} W^{nk}\tilde{x}_k, \tag{2.8}$$

where W is a complex number, as defined in Equation (2.7), N is the size of the FFT (usually a multiple of 2), and the Y_n's are the FFT components of the signal, with the original signal values denoted by the $\tilde{x}$'s.

After the FFT of the signal is taken, the next step is to calculate the squared magnitude of the spectrum. The squared magnitude spectrum is just the squared, absolute value of the FFT components (complex numbers), and is also called the power spectrum of the signal (or the energy spectrum if the square root is taken). This can be illustrated with the equation

$$Z_k = |Y(k)|^2, \qquad 0 \le k < K, \tag{2.9}$$

where K can be equal to N, the size of the FFT. Because only half of the (symmetric) spectrum is considered, K is taken to be equal to $N/2$. The result of spectral analysis is thus the squared magnitude spectrum, or power spectrum of the signal, as defined in Equation (2.9).

2.1.7 Energy and mel cepstrum coefficients

Energy and mel-scale frequency coefficients are the features most often used in speech recognition systems today. These coefficients provide a parameterised form of the speech that significantly reduces the data rate of the speech input, while still maintaining virtually all of the speech information [6].

To compute the cepstra, a bank of filters is constructed. The number of filters is usually chosen larger than the number of cepstra needed. These filters are equally spaced along the mel-scale frequency axis, but logarithmically spaced on the acoustic frequency axis. This is motivated by the fact that human perception of the frequency content of sounds, either for pure tones or for speech signals, does not follow a linear scale [1]. For each pure tone with an actual frequency, f, measured in Hz, a subjective pitch is perceived by humans, measured on a scale called the "mel" scale. The conversion between acoustic frequency to mel frequency, can be done with the equation [6]

$$melfrequency = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right), \tag{2.10}$$

where f is the acoustic frequency, in Hz, to be converted to a frequency in mels. Conversely, the mel frequency can be converted into an acoustic frequency through the equation (inverse of Equation (2.10))

$$f = 700 \cdot \left(10^{x/2595} - 1\right), \tag{2.11}$$

where x is the mel frequency to be converted into the acoustic frequency, f. The next step in the computation of the cepstra, is to construct a bank of filters. These filters are triangular filters, spaced equally along a mel-scale frequency axis, but logarithmically along the acoustic frequency axis.

Figure 2.5 shows the filter allocation in the frequency domain (acoustic frequency).

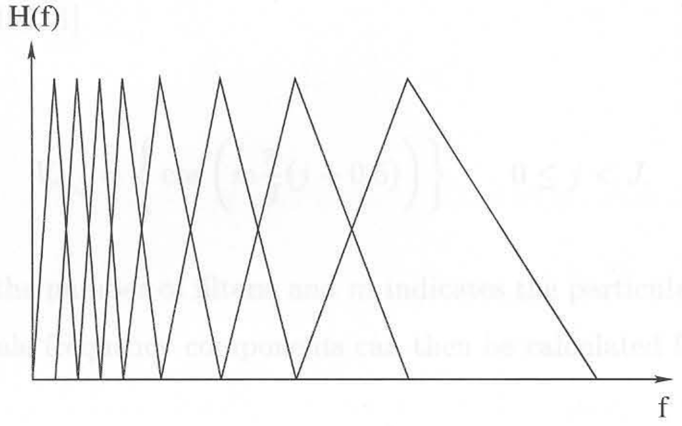

Figure 2.5: Filter allocation in the frequency domain.

The logarithmic nature of the filter spacing, described by Equation (2.10), is clearly seen in the figure. In this figure, each filter has a minimum and maximum frequency that correspond to the centre frequencies of the filters to the left and right, respectively. This results in filters having different bandwidths. In the logarithmic domain, however, these filters have equal bandwidth.

After the mel filters have been constructed, the energy output of each filter must be calculated. The energy output of each of the filters can be calculated as [6]

$$E_j = \sum_{k=0}^{K-1} \phi_j(k) Z_k, \qquad 0 \leq j < J, \tag{2.12}$$

where E_j is the energy from the j'th filter, $\phi_j(k)$ is the value of filter j's transfer function at k, K is equal to half of the size of the FFT (e.g. 256 for a 512-point FFT), and J is the number of filters. The filters must obey the constraint [6]

$$\sum_{k=0}^{K-1} \phi_j(k) = 1 \qquad \forall j. \tag{2.13}$$

To calculate the mel-scale frequency coefficients, a set of weighting factors is first computed. These weighting factors can be computed through an inverse discrete cosine transform, as [6, 102, 1]

$$V_{m,j} = \left\{ \cos\left(m\frac{\pi}{J}(j+0.5) \right) \right\} \qquad 0 \leq j < J, \tag{2.14}$$

where J is again the number of filters, and m indicates the particular cepstrum coefficient. The mel-scale frequency components can then be calculated from

$$c_m = \sum_{j=0}^{J-1} \beta_j V_{m,j} \log_{10}(E_j), \tag{2.15}$$

where E_j is the energy output of the j'th filter, as defined in Equation (2.12), $V_{m,j}$ is the weighing factor defined in Equation (2.14), and β is an amplification factor, which accommodates the dynamic range of the coefficients, c_m. The cepstra are normalised by the number of filters as

$$cn_m = c_m\sqrt{\frac{2}{J}}, \tag{2.16}$$

where cn_m is the final m'th mel cepstrum coefficient. The coefficients are sometimes liftered so that the higher and lower order cepstra have similar values. The cepstra are thus re-scaled to have similar magnitudes, according to

$$cn_m^{'} = \left(1 + \frac{L}{2}\sin\frac{\pi m}{L}\right) cn_m, \tag{2.17}$$

where L is the liftering coefficient and cn_m is defined in Equation (2.16).

To obtain the power, component 0 of the cepstrum vector is sometimes taken, which is just the average value of the spectrum, or root mean square (RMS) value of the signal [5]. The logarithm of the true energy, however, normally replaces component 0 of the cepstrum (since it is unreliable), and can be calculated as

$$E = \log_{10}\left(\sum_{n=0}^{N-1} \tilde{x}_n^2\right), \tag{2.18}$$

where $\tilde{x}_n$ is the windowed signal, and N is the number of samples in the window.

The above calculations will thus compute the mel frequency cepstrum coefficients as well as an energy measure. These provide a good parametric representation of the speech signal as well as good discrimination between the different speech sounds.

2.1.8 Normalisation

Normalisation of the speech features is usually done to reduce variability of the features with environment changes, microphone mismatch, etc. Many different algorithms exist, but only cepstral mean normalisation, energy normalisation, and a simple linear re-scaling technique will be discussed here.

Cepstral mean normalisation

After the cepstra have been computed, the next step is their normalisation. Both the power and the mel frequency coefficients can be normalised by the sentence-based mean. The first step is thus to compute the sentence-based mean, as

$$\mu(k) = \frac{1}{T}\sum_{t} x_t(k), \tag{2.19}$$

where T is the number of frames of the input utterance, k is the index of the cepstral coefficient, and $x_t(k)$ is the k'th cepstral coefficient. Next, the cepstra are normalised by the sentence-based mean, by simply subtracting the mean, as follows:

$$\tilde{x}_t(k) = x_t(k) - \mu(k), \tag{2.20}$$

where $\tilde{x}_t(k)$ is the k'th normalised cepstrum coefficient. This technique is useful to compensate for long-term spectral effects such as those caused by different microphones and audio channels.

Energy normalisation

The log energy is normalised to have values between $-E_{min}$ to 1.0 by subtracting the maximum value of the energy in the utterance from every energy value, and adding 1.0. This technique will cause silence to be indicated by a value of 1.0 and speech samples less than 1.0, and is in a convenient form for other processing tasks, such as silence detection.

Simple linear re-scaling

With this technique, each of the speech vector's components is normalised to have zero mean and unit variance. It is a simplified form of the more general whitening transform [69]. In this dissertation, it was used before speech vectors were applied to the input of the neural network that segmented the speech, in order to avoid any saturation problems of the neural network transfer functions. The mean and variance were calculated over all of the speech in the corpus that was used for training purposes, as the mean and variance can vary somewhat from speech utterance to speech utterance, which would degrade the neural network performance. It is, however, not uncommon to calculate the mean and variance only on a sentence by sentence basis. The mean of one speech vector feature is calculated as

$$\mu(k) = \frac{1}{N}\sum_n \frac{1}{T_n}\sum_t x_t^n(k), \tag{2.21}$$

where T_n is the number of frames of the n'th training utterance, k is the index of the feature in the speech vector, $x_t^n(k)$ is the k'th speech feature, and N is the number of training speech vectors. The variance of one of the speech features is calculated as

$$\sigma^2(k) = \frac{1}{N}\sum_n \frac{1}{T_n}\sum_t (x_t^n(k) - \mu(k))^2, \tag{2.22}$$

where $\sigma(k)$ is the standard deviation (square root of the variance). The speech vector components are then normalised by each of the feature means and standard deviations independently, as

$$xn_t^n(k) = \frac{x_t^n(k) - \mu(k)}{\sigma(k)}. \tag{2.23}$$

2.1.9 Delta energy and cepstrum coefficients

By adding time derivatives to the basic static features, the performance of speech recognition systems can be greatly improved. These dynamic features capture the dynamic changes that occur in the speech. To calculate the first derivative of the static features, also called the delta feature, a regression formula [3] such as

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2}, \tag{2.24}$$

is normally used, where d_t is the delta coefficient at time t, computed in terms of the corresponding static coefficients $c_{t-\theta}$ to $c_{t+\theta}$, and θ is the regression window. Equation (2.24) is used to calculate both the delta energy and delta cepstrum coefficients.

2.2 Hidden Markov models

Hidden Markov models (HMMs) are probably the most popular technology of choice for large vocabulary speech recognition systems. This is primarily due to the efficiency with which HMMs model the variation in the statistical properties of speech, both in the time and frequency domains [120]. A number of assumptions are, however, made when using HMMs [120, 1], but these will not be discussed here.

In the following sections, HMMs are discussed only briefly, in the context of how they are used in this study. In particular, training of HMMs is not discussed, as this dissertation only modifies the decoding process. The training of the HMMs is thus fairly standard and details can be found in other studies [1, 66, 67, 120, 3]. The basic concepts of HMMs are discussed in the next section, followed by a section on the Viterbi decoding process (used to recognise the unknown speech). Language models and the use of a word transition penalty are also discussed.

2.2.1 Basic elements and problems of an HMM

Hidden Markov models involve two underlying stochastic processes. One of these is hidden and can only be indirectly observed through the other stochastic process. This is the reason why HMMs are called "hidden" Markov models. The two stochastic processes occur concurrently within an HMM. When HMMs are considered to be used as a parametric model for a process, a number of concepts are of importance. Some of these concepts are discussed briefly in this section.

An important choice to be made when using HMMs is that of the structure of the HMM. An HMM consists of N states, often represented by circles in a pictorial representation. The states are connected by arcs, with arrows indicating the direction of an allowed transition between the states.

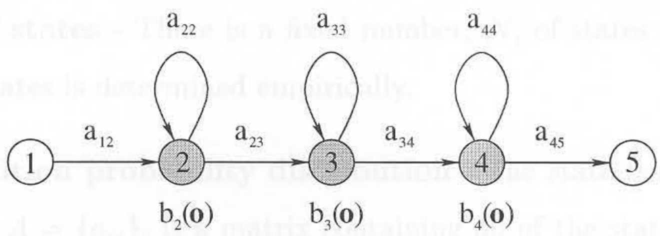

Figure 2.6: Structure of a basic HMM.

Figure 2.6 shows the structure of a basic HMM. In general, the theory allows for arbitrary connections between the states of the HMM. Left-to-right HMMs are, however, more commonly employed, especially in speech recognition, due to the temporal nature of the speech. In HMMs of this kind, as shown in Figure 2.6, there are fewer parameters than in an ergodic HMM. This limits the modeling power of the HMM, but the HMM parameters can be estimated more reliably with limited training data.

At the first time step, an initial state $q_1 = i$ is chosen, according to an initial state distribution π. At the following time instants, a stochastic process decides what the next valid states should be, based only on the current state. This process results in a first order HMM. This stochastic process, described by the transition probabilities, a_{ij}, thus determines what the next state j should be, given that the current state is state i. As each state is entered at time t (either the same or a different state), an observation $\boldsymbol{o}_t$ is generated according to a second stochastic process, described by an observation density $b_j(\boldsymbol{o}_t)$. The sequence of states $\boldsymbol{q}$ can not be observed directly, but only through the observations generated. This is why HMMs are "hidden".

All of the states in an HMM need not be emitting. In Figure 2.6, states 1 and 5 are non-emitting states. These states are used as a convenient way to connect multiple HMMs together. The shaded states (states 2 to 4) are emitting states.

An HMM is thus defined as a parametric model, consisting of the following components [1]:

- **Number of states** - There is a fixed number, N, of states in the model. The number of states is determined empirically.

- **State transition probability distribution** - The state transition probability distribution, $A = \{a_{ij}\}$, is a matrix containing all of the state transition probabilities. Each entry in the state transition matrix is defined as

$$a_{ij} = P[q_{t+1} = j | q_t = i], \qquad 1 \leq i, j \leq N, \tag{2.25}$$

where q_t is the state at time t, and q_{t+1} is the state at time $t+1$. Also, since a_{ij} are probabilities, they follow the usual stochastic constraints, e.g. $a_{ij} \in [0, 1]$.

- **Observation probability distribution** - The observation probability distribution, $B = \{b_j(\boldsymbol{o}_t)\}$ is the vector containing all of the state observation output probabilities. In this dissertation, these probabilities are modeled by Gaussian mixtures, resulting in continuous observation densities of the form

$$b_j(\boldsymbol{o}_t) = \sum_{m=1}^{M_s} c_{jm} \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}), \tag{2.26}$$

where M_s is the number of mixture components, and c_{jm} is the weight of the m'th component. In (2.26), $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate Gaussian with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, defined as

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}-\boldsymbol{\mu})}, \tag{2.27}$$

where n is the dimensionality of $\boldsymbol{o}$.

- **Initial state distribution** - The initial state distribution, $\pi = \{\pi_i\}$, is defined as the following

$$\pi_i = P[q_1 = i] \qquad 1 \leq i \leq N, \tag{2.28}$$

with q_1 being the initial state.

The complete parameter set, required to fully specify the HMM, can be compactly given as

$$\lambda = (A, B, \pi), \tag{2.29}$$

with A, B, and π as defined earlier.

To use HMMs, three basic problems must be solved. HMMs are popular, partly due to the efficiency in which they can solve these three problems:

- **Problem 1** - The first includes the need for an efficient way to calculate $P(\boldsymbol{O}|\lambda)$, the probability of the observation sequence ($(\boldsymbol{o}_1\boldsymbol{o}_2 \ldots \boldsymbol{o}_T)$, with T the maximum length of the sequence), given the model, and is referred to as an evaluation problem. The forward (or backward) procedure allows this problem to be solved in a computationally efficient means. This is not discussed any further here.

- **Problem 2** - The second involves the problem of finding the most likely state sequence $\boldsymbol{q}$, given the model λ and an observation sequence $\boldsymbol{O} = (\boldsymbol{o}_1\boldsymbol{o}_2 \ldots \boldsymbol{o}_T)$. This is referred to as the decoding process, used to uncover the "hidden" part of the HMM. In speech recognition, this is used to recognise the unknown speech (e.g. the most likely phone sequence). The Viterbi algorithm, based on dynamic programming (DP) concepts, is used for this purpose. It is discussed in more detail in the next section since the modification of this algorithm is part of our work.

- **Problem 3** - The third problem is concerned with the process of obtaining the HMM parameter set, $\lambda = (A, B, \pi)$ such that $P(\boldsymbol{O}|\lambda)$ is maximised. This is referred to as the training problem. The Baum-Welch algorithm is used for this purpose, but it is not discussed any further here.

In speech recognition applications of HMMs, the first stochastic process deals with the temporal sequence in which the HMM states occur and models the temporal structure of the speech signal. This is related to the transition probabilities. The second stochastic process models the locally stationary character of the speech signal and it is described by the conditional output probability density function. The solution to problem 3 is used to train HMM models for each of the phonemes that occur in the language of interest. A network of HMMs is then constructed, called a lattice or network, based on the task grammar, by joining HMMs together. The solution to problem 2, namely the Viterbi algorithm, is then used to determine the optimal sequence of phonemes (in a maximum likelihood sense in this dissertation). The sequence of phonemes is the recognition result from the unknown speech.

2.2.2 The Viterbi decoding process

As mentioned in the previous section, the most likely state sequence, given the model and the observation sequence, is of great importance in continuous speech recognition. The Viterbi algorithm, based on dynamic programming concepts, attempts to uncover the "hidden" part of an HMM. This algorithm attempts to find the single best path (state sequence) with the highest probability. Alternatively, it can be stated that the probability of the observation sequence $\boldsymbol{O} = (\boldsymbol{o}_1\boldsymbol{o}_2 \dots \boldsymbol{o}_T)$ and state sequence $\boldsymbol{q}$, given the model λ, or $P(\boldsymbol{q}, \boldsymbol{O}|\lambda)$, is maximised.

Figure 2.7 illustrates the Viterbi decoding process, as it is commonly used for continuous speech recognition. The vertical dimension represents the HMM states, while the horizontal dimension represents the frames of speech (time). Shown in the figure are two HMMs, namely HMM A and B, each having 5 states, with no skip transitions. In speech recognition systems, many more HMMs are connected together in a network determined by the task grammar. The result can be seen as a single, large HMM, having N states in total. It is also shown that state 5 of HMM A (labeled A5), and state 1 of HMM B (labeled B1) are regarded as being the same state (they are connected through the use of non-emitting states). The thicker lines in the figure represent the

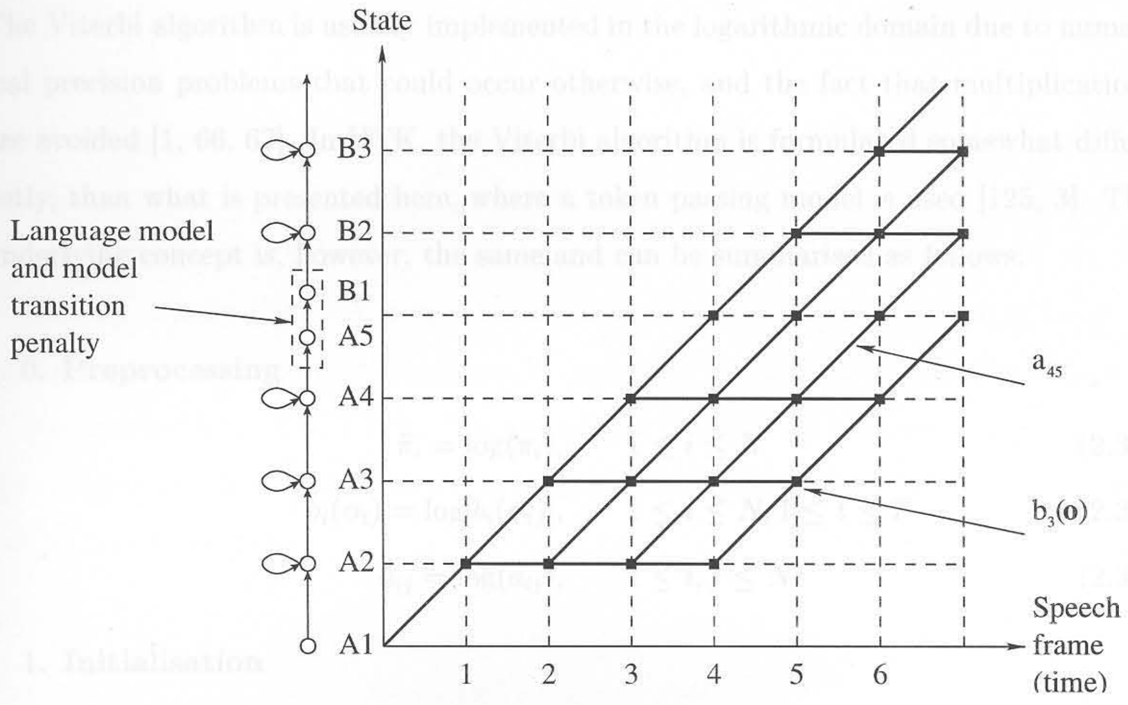

Figure 2.7: The Viterbi decoding process.

allowed paths (sequence of states), the dots represent the log state output probability of observing that frame at the specific time step, and the lines between the dots can be regarded as the log state transition probabilities.

In Figure 2.7, the log probability of a path is calculated by summing the log output probabilities and log transition probabilities along that path. At each time step, the log probability of the previous path that had the highest log probability of all candidate paths, are used to calculate the log probability of a set of new candidate paths. The one with the highest log probability is then chosen for use in the next time step. After each process of selecting the best path, the state sequence followed by that path is recorded. After the best path is calculated at the last time step, the optimal state sequence can be uncovered by backtracking the path followed. From the sequence of states, the times at which the state and model transitions occurred, can be derived. This can be used to align a phonetic transcription against a speech signal, or to recognise the speech and simultaneously provide the times at which model (phoneme) transitions occurred, thereby also segmenting the speech into phonemes.

The Viterbi algorithm is usually implemented in the logarithmic domain due to numerical precision problems that could occur otherwise, and the fact that multiplications are avoided [1, 66, 67]. In HTK, the Viterbi algorithm is formulated somewhat differently, than what is presented here, where a token passing model is used [125, 3]. The underlying concept is, however, the same and can be summarised as follows:

0. **Preprocessing**

$$\tilde{\pi}_i = \log(\pi_i), \qquad 1 \leq i \leq N \tag{2.30}$$

$$\tilde{b}_i(\boldsymbol{o}_t) = \log[b_i(\boldsymbol{o}_t)], \qquad 1 \leq i \leq N, 1 \leq t \leq T \tag{2.31}$$

$$\tilde{a}_{ij} = \log(a_{ij}), \qquad 1 \leq i, j \leq N \tag{2.32}$$

1. **Initialisation**

$$\tilde{\delta}_1(i) = \log(\delta_1(i)) = \tilde{\pi}_i + \tilde{b}_i(\boldsymbol{o}_1), \qquad 1 \leq i \leq N \tag{2.33}$$

$$\psi_1(i) = 0, \qquad 1 \leq i \leq N \tag{2.34}$$

2. **Recursion**

$$\tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}] + \tilde{b}_j(\boldsymbol{o}_t) \tag{2.35}$$

$$\psi_t(j) = \arg\max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}], \qquad 1 \leq j \leq N, 2 \leq t \leq T \tag{2.36}$$

3. **Termination**

$$\tilde{P}^* = \max_{1 \leq i \leq N} [\tilde{\delta}_T(i)] \tag{2.37}$$

$$q_T^* = \arg\max_{1 \leq i \leq N} [\tilde{\delta}_T(i)] \tag{2.38}$$

4. **Backtracking**

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1 \tag{2.39}$$

In the above equations $\delta_t(i)$ is the highest probability along a single path, at time t, that accounts for the first t observations and ends in state i, and Equation (2.39) gives the optimal state sequence.

2.2.3 Use of a language model

In this dissertation, a word loop grammar is used, which indicates that any word may follow any other word. In order to improve the recognition performance, a stochastic language model can be used. This takes into account that all word sequences are not necessarily equally likely. A simple probabilistic model of speech production relies on the fact that a specified word sequence, W, produces an acoustic observation sequence, Y, with probability $P(W, Y)$ [1]. The recognition or decoding process attempts to find the string of words that maximises the maximum *a posteriori* (MAP) probability. This can be represented as

$$\hat{W} \ni P(\hat{W}|Y) = \max_{W} P(W|Y). \tag{2.40}$$

Using Bayes' Rule, $P(W|Y)$ can be written as

$$P(W|Y) = \frac{P(Y|W)P(W)}{P(Y)}, \tag{2.41}$$

and consequently, Equation (2.40) can be written as

$$\hat{W} = \arg\max_{W} P(Y|W)P(W), \tag{2.42}$$

which is called the MAP decoding rule. In Equation (2.42), $P(Y|W)$ is the acoustic model (a probability of a sequence of acoustic observations, given a word string, estimated by the Markov models). $P(W)$ is called the language model. It gives the probability of observing a particular word sequence, W. As shown in Equation (2.42), the language model has a significant effect on the recognition accuracy.

To estimate $P(W)$, a text corpus is used that contains many word sequences. $P(W)$ can then be estimated by

$$P_N(W) = \prod_{i=1}^{Q} P(w_i|w_{i-1}, w_{i-2}, \ldots, w_{i-N+1}), \tag{2.43}$$

for an N-gram language model, where w_i is the i'th word and Q is the total number of words. For a bigram language model ($N = 2$), $P(w_i|w_{i-1})$ can be estimated by building a table of bigram counts, and then output a back-off bigram by using [3]

$$P(w_i|w_{i-1}) = \begin{cases} (F(w_i, w_{i-1}) - D)/F(w_{i-1}) & \text{if } F(w_i, w_{i-1}) > t \\ b(w_i)P(w_{i-1}) & \text{otherwise} \end{cases}, \tag{2.44}$$

where $F(w_i, w_{i-1})$ is the number of times word w_i follows word w_{i-1}, and $F(w_{i-1})$ is the number of times that word w_{i-1} appears. In Equation (2.44), which is based on a process called discounting [3], D is a discount constant, t is a bigram count threshold, and $b(w_i)$ is the back-off weight for word w_i, which ensures that all of the bigram probabilities for a given history sum to one.

The language model probability is added to each word-end transition (in the logarithmic domain), or multiplied with the acoustic model probability, according to Equation (2.42). The language model probability is an *a priori* probability, computed off-line. It is often scaled by a language model scale factor s. From Equation (2.42), recognition is thus based on maximising

$$\hat{W} = \arg\max_{W} P(Y|W)P(W)^s, \tag{2.45}$$

which can be conveniently incorporated in the Viterbi algorithm. Equation (2.35) can thus be modified for word-end transitions as

$$\tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1 \le i \le N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + s\tilde{P}(W)] + \tilde{b}_j(\boldsymbol{o}_t), \tag{2.46}$$

where $\tilde{P}(W) = \log(P(W))$ is the log language model probability, and s is language model probability scale factor. If the transition from state i to j is not between word-ends, Equation (2.35) is used.

2.2.4 Use of a word transition penalty

In addition to language models, a fixed word transition penalty is often used. The effect of the word transition penalty is to modify the language model log probability according to the equation

$$\tilde{P}(W)' = s\tilde{P}(W) + p, \tag{2.47}$$

where p is the word transition penalty when it is negative, and word confidence when it is positive (in the logarithmic domain). Equation (2.46) of the previous section can thus be further modified (for word-end transitions only), as

$$\tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1\leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + s\tilde{P}(W) + p] + \tilde{b}_j(\boldsymbol{o}_t). \tag{2.48}$$

In general, when unknown speech is given as input to the recogniser, the number of insertions tend to increase and the number of deletions decrease, as p is increased (more positive and less negative) and s decreased. Alternatively, the number of insertions tend to decrease and the number of deletions increase, as p decreases (more negative and less positive) and s increases. This can be explained due to the fact that the token log-likelihood is decreased when p is a negative number, penalising transitions between words (phonemes), resulting in a decrease in the number of insertions and an increase in the number of deletions. The optimal values for s and p are usually found by maximising the recognition accuracy on a development set, for both of these parameters.

2.3 Recurrent neural networks

For a number of years, neural networks have been a popular choice to solve a variety of problems. Some of these include classification, regression, probability estimation, time series prediction, process modeling and process control.

The most common neural network architecture in use today, is probably the multilayer perceptron [118]. This neural network is regarded as a "static" neural network in that it does a static mapping of inputs to outputs. Recurrent neural networks (RNNs) are regarded as "dynamic" neural networks, since all of the past information is used, in addition to the current input, to calculate the output. Future input information may also be useful in calculating the output of the neural network. Conventional recurrent neural networks use future input information by delaying the output by a number of time steps. Alternatively stated, a "window" of future input information is applied to the neural network's inputs. The window size must be set empirically. Bi-directional recurrent neural networks avoid the choice of window size, in that they use the entire sequence of inputs to calculate the outputs.

This section discusses the use of recurrent neural networks (conventional and bi-directional) as pattern classifiers. In Section 2.3.1 conventional recurrent neural networks are discussed, and bi-directional recurrent neural networks in Section 2.3.2. Appendix A discusses the training of these networks.

2.3.1 Conventional recurrent neural networks

RNNs are generally regarded as being more powerful than multilayer perceptrons. These neural networks contain one or more feedback connections. Although RNNs are not limited to a specific architecture, only recurrent multilayer perceptrons (RMLPs) [70] are discussed here. RNNs map their input space to the output space, by responding temporally to an externally applied input signal. The network acquires state representations and effectively uses all of the past input information to calculate the output. For this reason, they are often considered "dynamically driven recurrent networks".

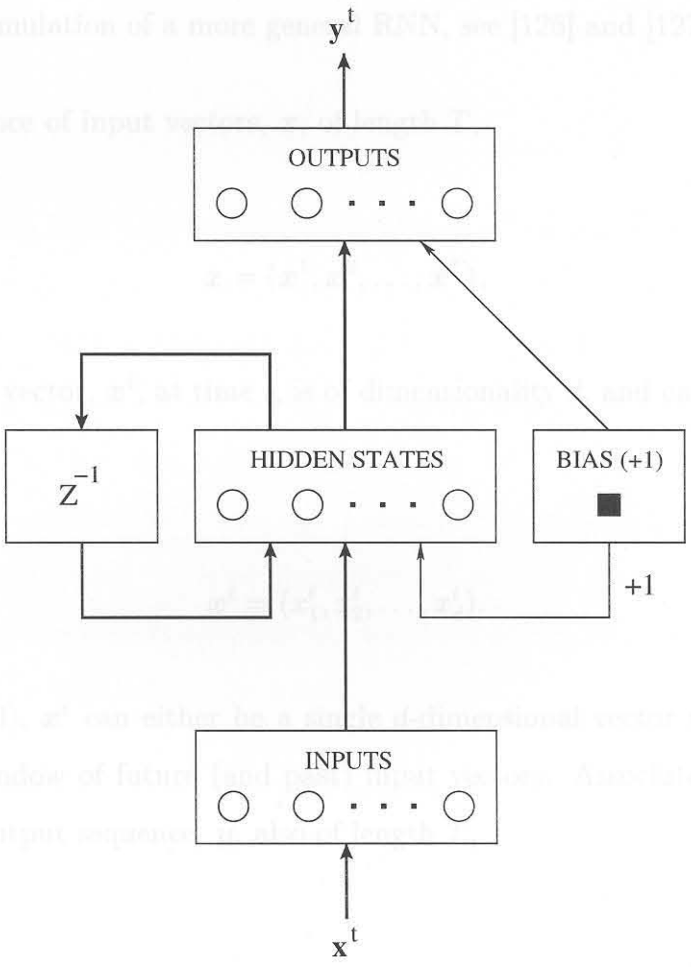

Figure 2.8: A conventional recurrent neural network (RNN).

Figure 2.8 shows the structure of a simple RMLP having two layers of weights. The output of the hidden layer nodes are delayed by one time step and applied to the input of these hidden layer neurons. In this way, past information continues to circulate through the network. The output of the neural network is calculated by using the outputs of the hidden layer neurons. Also shown in the figure is an input whose value is fixed at +1. This bias input is connected to both the hidden neurons and output neurons and allows more freedom to the formation of the decision boundaries by the network. The inputs shown in the figure are not regarded as being neurons. They are simply a convenient way to connect the external inputs to the hidden neurons.

The forward propagation of input vectors to output vectors for the two layer RNN, given in Figure 2.8, is presented below. This is probably the most common form of RNN. For the formulation of a more general RNN, see [126] and [127].

Consider a sequence of input vectors, $\boldsymbol{x}$, of length T,

$$\boldsymbol{x} = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^T), \tag{2.49}$$

where each input vector, $\boldsymbol{x}^t$, at time t, is of dimensionality d, and can be conveniently written as

$$\boldsymbol{x}^t = (x_1^t, x_2^t, \ldots, x_d^t). \tag{2.50}$$

In Equation (2.50), $\boldsymbol{x}^t$ can either be a single d-dimensional vector at time t, or may also include a window of future (and past) input vectors. Associated with the input sequence, is an output sequence, $\boldsymbol{y}$, also of length T,

$$\boldsymbol{y} = (\boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^T), \tag{2.51}$$

where each output vector, $\boldsymbol{y}^t$, at time t is of dimensionality c, and can be written as

$$\boldsymbol{y}^t = (y_1^t, y_2^t, \ldots, y_c^t). \tag{2.52}$$

A sequence of hidden nodes outputs, $\boldsymbol{o}$, of length T, is also formed,

$$\boldsymbol{o} = (\boldsymbol{o}^1, \boldsymbol{o}^2, \ldots, \boldsymbol{o}^T), \tag{2.53}$$

where each hidden layer vector, $\boldsymbol{o}^t$, at time t is of dimensionality m, and can be written as

$$\boldsymbol{o}^t = (o_1^t, o_2^t, \ldots, o_m^t). \tag{2.54}$$

The input to hidden layer weight matrix, $\boldsymbol{W}^{IH} = \{w_{ji}^{IH}\}$ contains the matrix of input to hidden layer weights. It is of dimension m x d. The bias unit to hidden layer weight matrix is denoted as $\boldsymbol{W}^H = \{w_j^H\}$ (of dimension m x 1), while the bias unit to output layer weight matrix is given by $\boldsymbol{W}^O = \{w_k^O\}$ (of dimension c x 1). The feedback matrix of hidden layer to hidden layer weights, is given by $\boldsymbol{W}^{HH} = \{w_{jk}^{HH}\}$. The hidden to output layer weight matrix, $\boldsymbol{W}^{HO} = \{w_{kj}^{HO}\}$ contains the weights from the hidden layer nodes to the output nodes. It is of dimension c x m. The first index on the weight subscript indicates the neuron to which the weight is going, while the rightmost index indicates the neuron from which the weight originated.

The output of a neuron is calculated as a function of the weighted sum of its inputs. For the hidden neurons this is

$$\begin{aligned} o_j^t = f_j(a_j^t) &= f_j\left(\sum_{i=1}^{d} w_{ji}^{IH} x_i^t + \sum_{k=1}^{m} w_{jk}^{HH} o_k^{t-1} + w_j^H\right), \\ & j = 1, 2, \ldots, m; t = 1, 2, \ldots, T, \end{aligned} \tag{2.55}$$

where o_k^{t-1} is the one step delayed hidden neuron output, taken as 0 at $t = 1$. In Equation (2.55), a_j^t is the weighted sum of inputs to the j'th hidden neuron. Hidden neuron j's transfer function is denoted as f_j, and it is often taken as the hyperbolic tangent function for reasons of faster training convergence,

$$f_j(a_j) \equiv \tanh(a_j) = \frac{e^{a_j} - e^{-a_j}}{e^{a_j} + e^{-a_j}}, \tag{2.56}$$

where e is the exponential function. The outputs of the output layer neurons can be calculated as

$$y_k^t = f_k(a_k^t) = f_k\left(\sum_{j=1}^{m} w_{kj}^{HO} o_j^t + w_k^O\right), \qquad k = 1, 2, \ldots, c; t = 1, 2, \ldots, T, \tag{2.57}$$

where f_k is the transfer function of the output neurons. The softmax activation function [69] is often used for the output neurons,

$$f_k(a_k) \equiv \frac{e^{a_k}}{\sum_{l=1}^{c} e^{a_l}}, \tag{2.58}$$

where the sum is taken over all c outputs. This activation function will ensure that the outputs satisfy the usual stochastic constraints ($y_k \geq 0$, $\sum y_k = 1$) so that they may be interpreted as probabilities.

Forward propagation procedure

The forward propagation of the inputs to the outputs, as used in the simulation of the network once it has been trained, can be summarised as follows:

1. Set activations of all neurons to zero, as well as the vector representing the 1 step delayed version of the hidden layer neuron outputs.
2. Apply an external input vector, $\boldsymbol{x}^t$, at the inputs of the network.
3. Calculate the outputs of hidden layer neurons, using Equations (2.55) and (2.56).
4. Calculate the outputs of output layer neurons, using Equations (2.57) and (2.58).
5. Form the output vector, $\boldsymbol{y}^t$, by taking the outputs of the output neurons.
6. Repeat steps 2 to 5 for all t.

2.3.2 Bi-directional recurrent neural networks

Context information often plays an important role in calculating the output of a neural network. Performance can be significantly increased when either or both past and future vectors are applied to the input of the neural network, in addition to the current input vector. The designer of the neural network is then faced with the problem of choosing the size of the window of future and past input vectors. Bi-directional recurrent neural networks [116, 117, 121] avoid the problem of choosing the window size, by using the entire sequence of input vectors. It thus makes use of all of the available past and future information at any time instant. The primary disadvantage is, however, that the architecture is not suitable for most on-line applications.

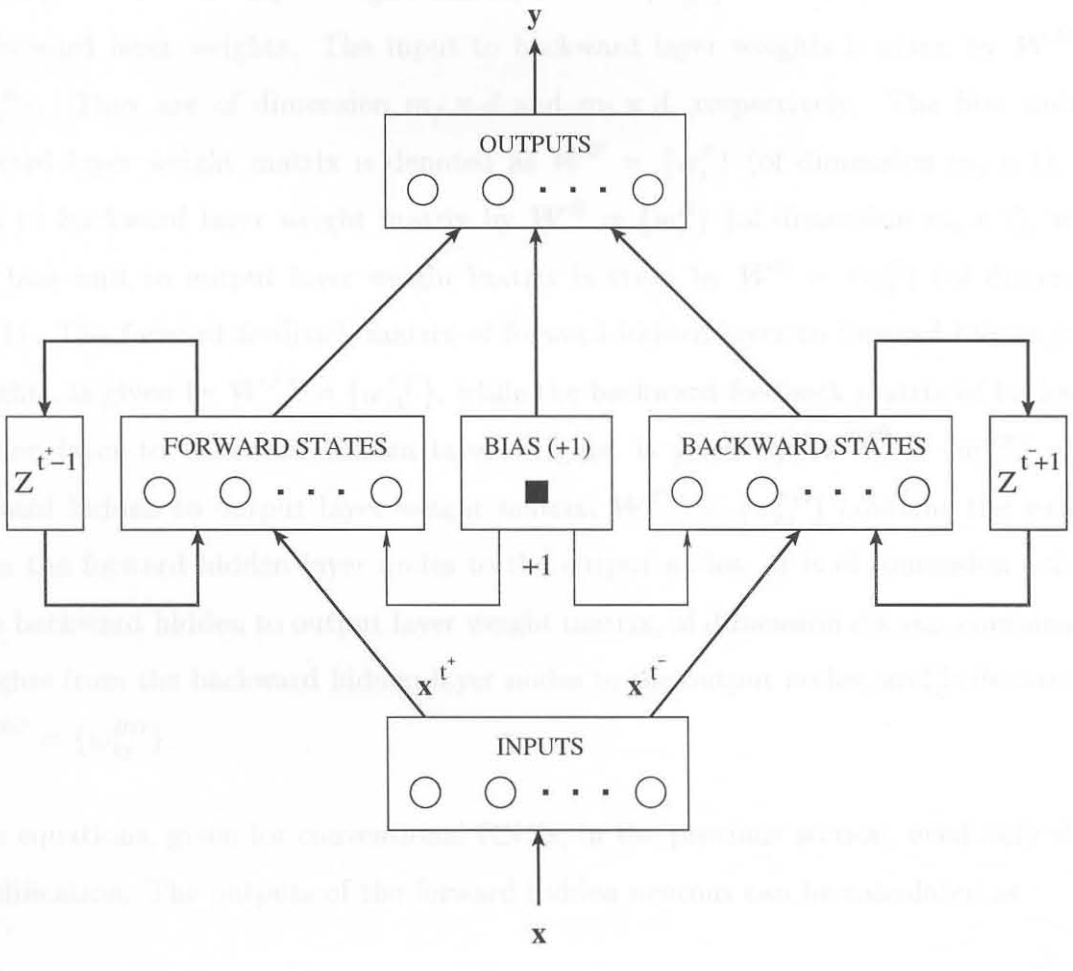

Figure 2.9: A bi-directional recurrent neural network (BRNN).

Figure 2.9 gives the structure of a bi-directional recurrent neural network. It consists of two underlying conventional recurrent neural networks, one operating in the forward time direction ($t^+ = 1$ to T) and one in the negative time direction ($t^- = T$ to 1). The hidden states of both these forward and backward RNNs are used to calculate the outputs of the neural network. The delay operators, z^{t^+-1} and z^{t^-+1}, are also shown for the forward and backward states, respectively. The former delays the forward state (neurons) outputs by 1 time step, while the latter advances the backward state (neurons) outputs by 1 time step. In the figure, $\boldsymbol{x}^{t^+}$ denotes the positive time input sequence, starting from $t = 1$, proceeding to $t = T$, while $\boldsymbol{x}^{t^-}$ denotes the negative time input sequence, starting from $t = T$, proceeding to $t = 1$.

The input to forward layer weight matrix, $\boldsymbol{W}^{IF} = \{w_{ji}^{IF}\}$ contains the matrix of input to forward layer weights. The input to backward layer weights is given by $\boldsymbol{W}^{IB} = \{w_{ji}^{IB}\}$. They are of dimension m_f x d and m_b x d, respectively. The bias unit to forward layer weight matrix is denoted as $\boldsymbol{W}^{F} = \{w_{j}^{F}\}$ (of dimension m_f x 1), the bias to backward layer weight matrix by $\boldsymbol{W}^{B} = \{w_{j}^{B}\}$ (of dimension m_b x 1), while the bias unit to output layer weight matrix is given by $\boldsymbol{W}^{O} = \{w_{k}^{O}\}$ (of dimension c x 1). The forward feedback matrix of forward hidden layer to forward hidden layer weights, is given by $\boldsymbol{W}^{FF} = \{w_{jk}^{FF}\}$, while the backward feedback matrix of backward hidden layer to backward hidden layer weights, is given by $\boldsymbol{W}^{BB} = \{w_{jk}^{BB}\}$. The forward hidden to output layer weight matrix, $\boldsymbol{W}^{FO} = \{w_{kj}^{FO}\}$ contains the weights from the forward hidden layer nodes to the output nodes. It is of dimension c x m_f. The backward hidden to output layer weight matrix, of dimension c x m_b, contains the weights from the backward hidden layer nodes to the output nodes, and is denoted by $\boldsymbol{W}^{BO} = \{w_{kj}^{BO}\}$.

The equations, given for conventional RNNs, in the previous section, need only slight modification. The outputs of the forward hidden neurons can be calculated as

$$o_j^{f,t^+} = f_j^f(a_j^{f,t^+}) \;=\; f_j^f\left(\sum_{i=1}^{d} w_{ji}^{IF} x_i^{t^+} + \sum_{k=1}^{m_f} w_{jk}^{FF} o_k^{f,t^+-1} + w_j^F\right),$$
$$j = 1,2,\ldots,m_f; t^+ = 1,2,\ldots,T. \quad (2.59)$$

The outputs of the backward neurons are calculated as

$$o_j^{b,t^-} = f_j^b(a_j^{b,t^-}) \;=\; f_j^b\left(\sum_{i=1}^{d} w_{ji}^{IB} x_i^{t^-} + \sum_{k=1}^{m_b} w_{jk}^{BB} o_k^{b,t^-+1} + w_j^B\right),$$
$$j = 1,2,\ldots,m_b; t^- = T, T-1,\ldots,1. \quad (2.60)$$

The outputs of the neural network are obtained only after calculating the forward and backward neurons' outputs for the entire input sequence, $\boldsymbol{x}$, as

$$y_k^t = f_k(a_k^t) \;=\; f_k\left(\sum_{j=1}^{m_f} w_{kj}^{FO} o_j^{f,t} + \sum_{i=1}^{m_b} w_{kj}^{BO} o_j^{b,t} + w_k^O\right),$$
$$k = 1,2,\ldots,c; t = 1,2,\ldots,T, \quad (2.61)$$

where $o_i^{f,t}$ is the output of the i'th forward neuron at time t, $o_i^{b,t}$ is the output of the i'th backward neuron at time t, m_f is the number of forward neurons, m_b is the number of backward neurons, and f_j^f, f_j^b and f_k are the neuron transfer functions for the forward hidden, backward hidden, and output layers, respectively. In the equations above, o_j^{f,t^+-1} denotes the 1 step delayed forward hidden node output, and o_j^{b,t^-+1} is the 1 step advanced backward hidden node output. The j'th forward hidden node output at time t and t^+ is given by $o_j^{f,t}$ and o_j^{f,t^+}, respectively. The j'th backward hidden node output at time t and t^- is given by $o_j^{b,t}$ and o_j^{b,t^-}, respectively.

Forward propagation procedure

The forward propagation of the inputs to the outputs, as used in the simulation of the network once it has been trained, can be summarised as follows:

1. Set activations of all neurons to zero, as well as the vectors representing the 1 step delayed, and 1 step advanced version of the forward and backward neurons' outputs.
2. Apply an external input sequence, $\boldsymbol{x}$, at the inputs of the network.
3. Calculate the outputs of the forward hidden layer neurons, using Equations (2.59) and (2.56) for $t^+ = 1$ to T (the entire sequence).
4. Calculate the outputs of the backward hidden layer neurons, using Equations (2.60) and (2.56) for $t^- = T$ to 1 (the entire sequence).
5. Calculate the outputs of the output layer neurons, using Equations (2.61) and (2.58) (the entire sequence of outputs).
6. Form the output sequence, $\boldsymbol{y}$, by taking the sequence of network outputs.

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).

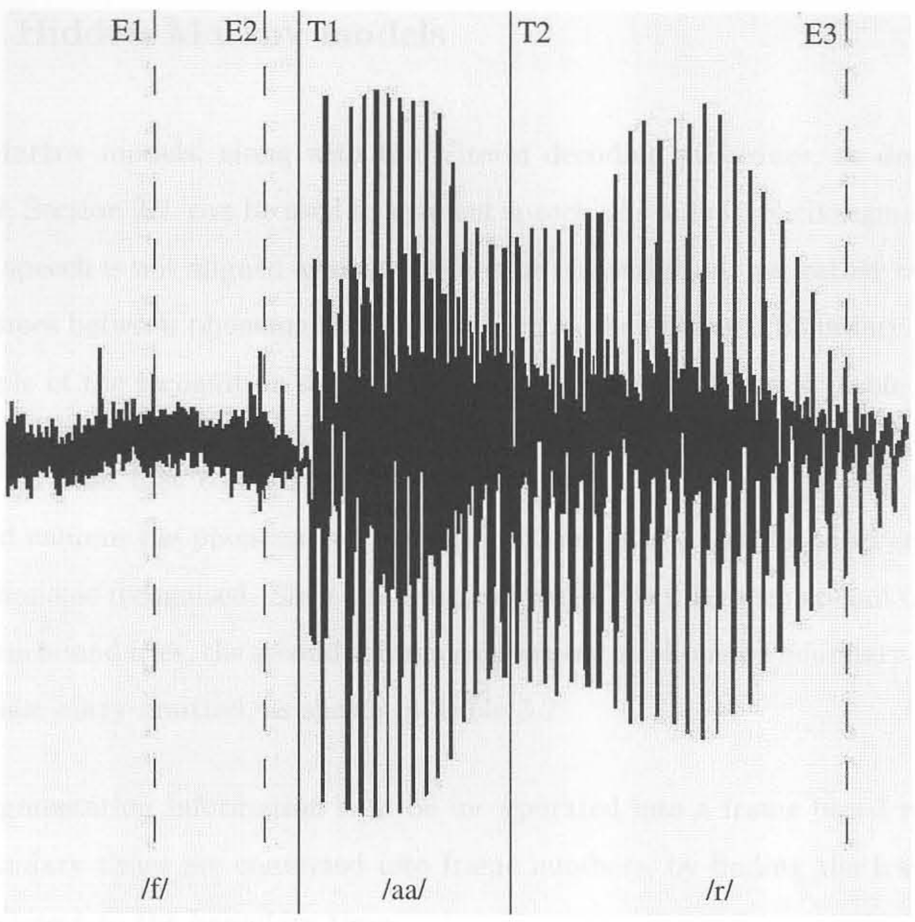

Figure 3.1: Example of the segmentation of the word "four".

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

$$0\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,1 \ldots 0\,0\,1\,0\,0.$$

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

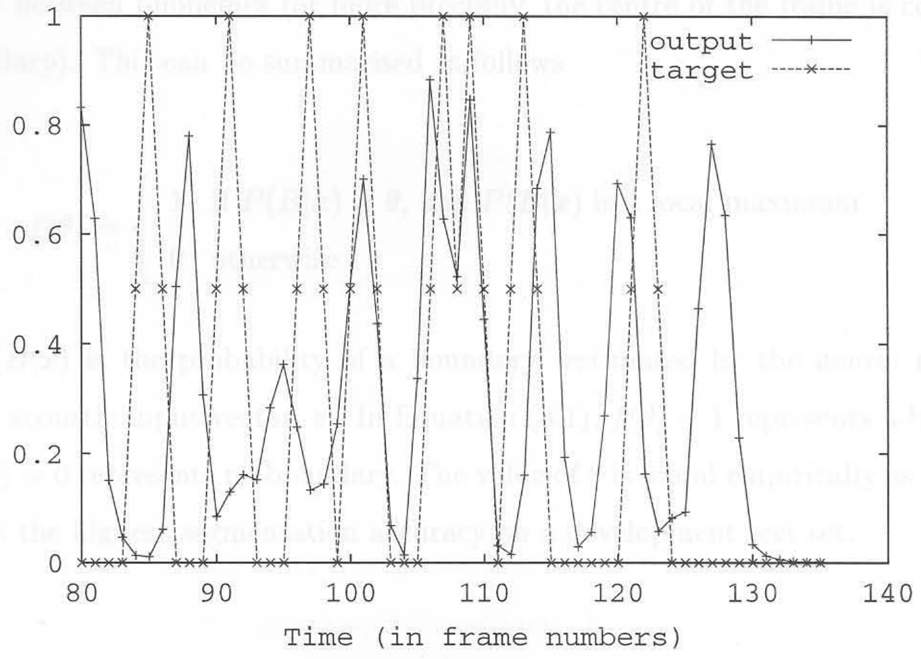

Figure 3.2: Example of neural network target and output values.

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

$$d_{ji} = |p^t_j - p^e_i|, \qquad j = 0, 1, \ldots, N_t - 1, i = 0, 1, \ldots, N_e - 1.$$

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	$\leq$ 5 ms
1	$\leq$ 10 ms
2	$\leq$ 15 ms
3	$\leq$ 20 ms
4	$\leq$ 25 ms
5	$\leq$ 30 ms
6	$\leq$ 35 ms
7	$\leq$ 40 ms
8	$\leq$ 45 ms
9	$\leq$ 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.

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

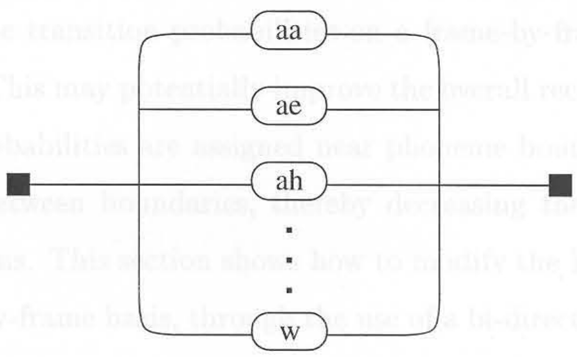

Figure 4.1: Loop grammar used for phoneme recognition.

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

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

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

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

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.

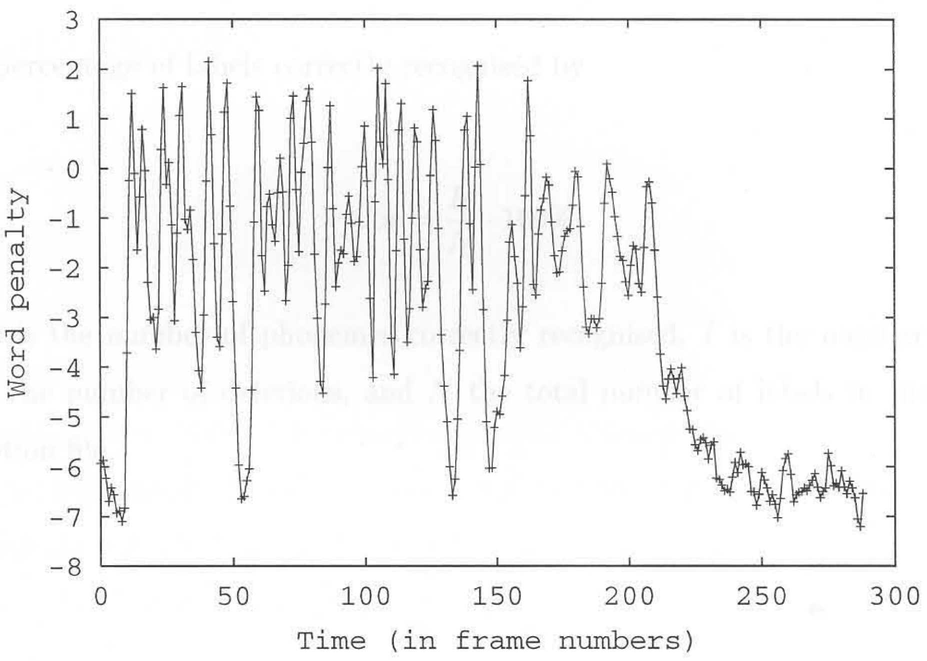

Figure 4.3: Example of an adaptive word transition penalty term with $w_b = 0$.

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.

Chapter 5

Experiments

This chapter reports the results obtained on phoneme segmentation and recognition experiments. Section 5.1 describes the segmentation of speech into phonemes, where Section 5.1.1 gives results for HMM based segmentation, and Section 5.1.2 for BRNN based segmentation. Section 5.2 discusses results obtained for the baseline phoneme recognition experiments. Finally, Section 5.3 gives results on the recognition of speech using segmentation information. Please note that in this chapter, the word "optimised" is used to refer to empirical optimisation, and is thus not strictly correct in a mathematical sense.

5.1 Experiment 1: Speech segmentation

The segmentation of speech into phonemes, as discussed in detail in Chapter 3, can be done in two ways. The first is the use of hidden Markov models, where the Viterbi algorithm is used to find the most likely phone sequence, given the speech utterance. Since the times between the model transitions are known, these boundary locations can be used as the segmentation of the speech signal. The second segmentation method, is to use a bi-directional recurrent neural network, that estimates the probability of a boundary and no boundary, given the speech vectors. A postprocessor then determines the actual boundary locations, through the use of a threshold function. Both of these techniques will be considered here.

5.1.1 Hidden Markov models

For the segmentation of speech, using hidden Markov models, four factors that have an influence on the segmentation accuracy, are investigated. The first is the number of mixtures used in the output probability distribution of each HMM. The second is the window size, or allowed difference between the true and estimated phoneme boundary positions. The effects of both a language model and embedded re-estimation, are also investigated.

Number of HMM mixtures

Purpose: The purpose of this experiment is to determine the effect that the number of mixtures in the HMM output probability distribution, has on the segmentation accuracy of an HMM based segmentation system.

Experimental setup: Speech utterances are recognised using trained HMMs, and the recognition results converted into a set of phoneme boundary locations, as described in Chapter 3. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by segmenting the 1344 files of the TIMIT test set (the full test set). No language model is used, and embedded re-estimation is not performed. The window size is fixed at 3 (20 ms).

Results: Figure 5.1 shows the effect of the number of mixtures in the HMM output probability density has on the segmentation accuracy. The details of these results are given in Table 5.1. The table shows the average accuracy and percentage of boundaries correctly identified, the number of boundaries correctly matched (hits), insertions and deletions, over the full test set. It can be seen that the number of mixtures does not have a significant influence on the segmentation accuracy. The best segmentation accuracy, at a window size of 3, is obtained with 8 mixtures, and is equal to 71.23%, with a percentage correct of 87.07%. The average number of hits, insertions and deletions, per utterance (sentence) over the complete test data set, is equal to 31.40, 5.40, and 4.64, respectively. The average number of true boundaries per utterance is 36.05 and the average number of observations (frames) per utterance is 305.71. It should be noted that even though the best performance was obtained with 8 mixtures, 2 mixtures would probably be sufficient to model the acoustic space due to the fact that the TIMIT database consists of male and female speakers.

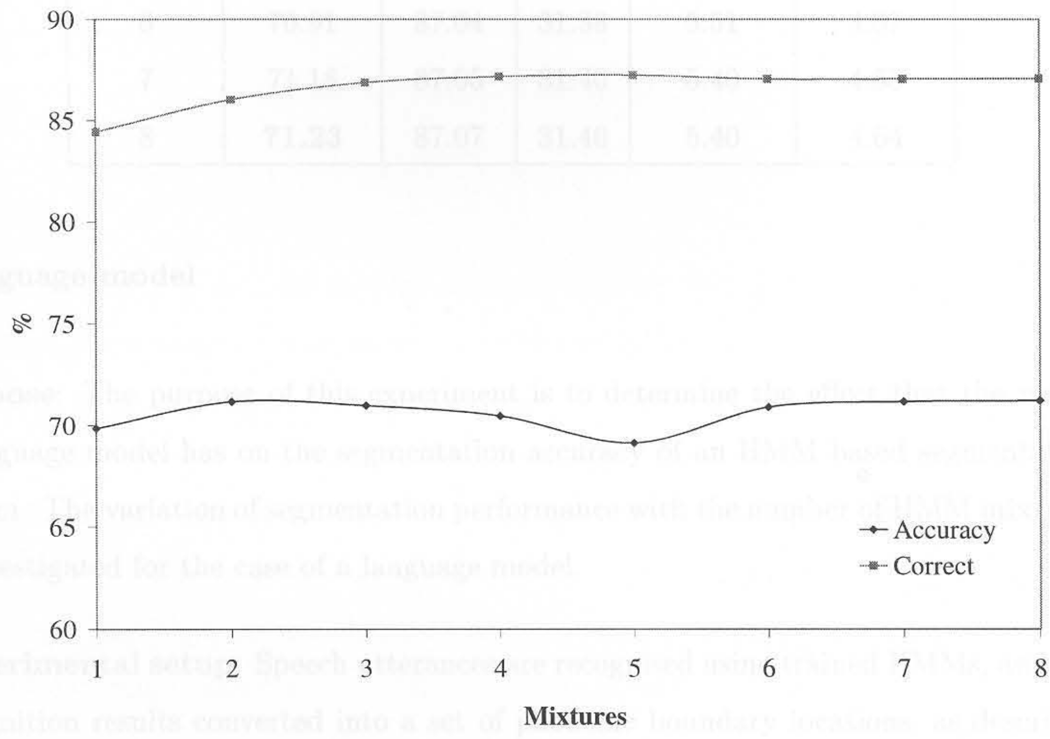

Figure 5.1: Effect of the number of mixtures in the HMMs, on the segmentation performance. No language model is used, and no embedded re-estimation performed.

Table 5.1: Numerical results of the segmentation performance obtained with the HMM based segmentation system, for different numbers of mixtures in the output probability densities. No language model or embedded re-estimation is performed.

Mixtures	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
1	69.86	84.44	30.40	**5.04**	5.64
2	71.18	86.03	30.97	5.12	5.07
3	70.98	86.87	31.28	5.45	4.77
4	70.47	87.17	31.41	5.71	4.64
5	69.13	**87.23**	**31.45**	6.20	**4.60**
6	70.91	87.04	31.38	5.51	4.67
7	71.18	87.05	31.40	5.40	4.65
8	**71.23**	87.07	31.40	5.40	4.64

Language model

Purpose: The purpose of this experiment is to determine the effect that the use of a language model has on the segmentation accuracy of an HMM based segmentation system. The variation of segmentation performance with the number of HMM mixtures is investigated for the case of a language model.

Experimental setup: Speech utterances are recognised using trained HMMs, and the recognition results converted into a set of phoneme boundary locations, as described in Chapter 3. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by segmenting the 1344 files of the TIMIT test set (the full test set). A bigram language model, calculated on the text data of the training set, is used and embedded re-estimation is not performed. The window size is fixed at 3 (20 ms).

Results: Table 5.2 shows the variation of the segmentation performance with the number of mixtures in the HMM output probability density, when a language model

Table 5.2: Summary of results showing the segmentation performance obtained with the HMM based segmentation system, for different numbers of mixtures in the output probability densities. A bigram language model is used and no embedded re-estimation is performed.

Mixtures	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
1	72.85	80.56	29.04	**2.63**	7.02
2	75.08	82.84	29.85	2.66	6.20
3	75.36	83.53	30.12	2.76	5.93
4	75.37	83.95	30.27	2.90	5.77
5	74.95	84.02	30.31	3.07	5.73
6	75.58	84.10	30.35	2.88	5.69
7	**75.69**	**84.11**	**30.35**	2.85	**5.69**
8	75.62	84.03	30.33	2.84	5.72

is used. The table shows the average accuracy and percentage of boundaries correctly identified, the number of boundaries correctly matched (hits), insertions and deletions, over the full test set. It can be seen that number of mixtures does not have a significant influence on the segmentation accuracy. The language model has the additional effect of smoothing the segmentation performance, as the number of insertions is decreased. The number of hits, however, decreases slightly, and the number of deletions slightly increases. This explains the slight decrease in the percentage correct, over the case when no language model is used. The best segmentation accuracy is obtained with 7 mixtures, and is equal to 75.69%, with a percentage correct of 84.11%. The average number of hits, insertions and deletions is equal to 30.35, 2.85, and 5.69, respectively. The average number of true boundaries is 36.05 and the average number of observations (frames) is 305.71. Here 3 mixtures would probably be sufficient for the segmentation task, even though the best performance was obtained with 7 mixtures.

Embedded re-estimation

Purpose: The purpose of this experiment is to determine the effect that the use of embedded re-estimation has on the segmentation accuracy of an HMM based segmentation system. The variation of segmentation performance with the number of HMM mixtures is investigated both when no language model is used, and when a language model is used.

Experimental setup: Speech utterances are recognised using trained HMMs, and the recognition results converted into a set of phoneme boundary locations, as described in Chapter 3. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by segmenting the 1344 files of the TIMIT test set (the full test set). When a language model is used, a bigram language model is chosen. Embedded re-estimation is performed. The window size is fixed at 3 (20 ms).

Results: Tables 5.3 and 5.4 show the variation of the segmentation performance with the number of mixtures in the HMM output probability distribution, when both a language model and no language model is used, and embedded re-estimation is performed. The tables shows the average accuracy and percentage of boundaries correctly identified, the number of boundaries correctly matched (hits), insertions and deletions, over the full test set. It can be seen that number of mixtures does not have a significant influence on the segmentation accuracy. The language model has the additional effect of smoothing the segmentation performance, as the number of insertions is decreased. The number of hits, however, decreases slightly, and the number of deletions slightly increases. This explains the slight decrease in the percentage correct, when no language model is used. When a language model is not used, the best segmentation accuracy is obtained with 7 mixtures, and is equal to 70.27%, with a percentage correct of 88.04%. The average number of hits, insertions and deletions is equal to 31.76, 6.07, and 4.29, respectively. When a language model is used, the best segmentation accuracy is obtained with 4 mixtures, and is equal to 75.47%, with a percentage correct of 85.28%. The average number of hits, insertions and deletions is equal to 30.77, 3.31, and 5.27,

Table 5.3: Numerical results of the segmentation performance obtained with the HMM based segmentation system, for different numbers of mixtures in the output probability densities. No language model is used and embedded re-estimation is performed.

Mixtures	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
1	69.00	85.83	30.90	**5.82**	5.15
2	68.28	87.18	31.42	6.49	4.63
3	69.47	87.90	31.68	6.34	4.37
4	69.52	88.29	31.83	6.42	4.22
5	68.14	**88.47**	**31.91**	6.97	**4.14**
6	69.73	88.03	31.76	6.25	4.29
7	**70.27**	88.04	31.76	6.07	4.29
8	69.70	87.43	31.53	6.07	4.51

respectively. The average number of true boundaries is 36.05 and the average number of observations (frames) is 305.71. Embedded re-estimation thus results in a slight decrease in the segmentation performance [71.23% vs 70.27% (no LM) and 75.69 vs 75.47 (with LM)].

Window size

Purpose: The purpose of this experiment is to determine the effect that the window size has on the segmentation accuracy of an HMM based segmentation system. The variation of segmentation performance with the window size is investigated both when no language model is used, and when a language model is used.

Experimental setup: Speech utterances are recognised using trained HMMs, and the recognition results converted into a set of phoneme boundary locations, as described in Chapter 3. The HMMs are trained on 3696 files of the TIMIT training set, and

Table 5.4: Numerical results of the segmentation performance obtained with the HMM based segmentation system, for different numbers of mixtures in the output probability densities. A bigram language model is used and embedded re-estimation is performed.

Mixtures	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
1	73.20	82.49	29.72	**3.18**	6.32
2	73.78	84.14	30.35	3.52	5.70
3	74.92	84.88	30.62	3.40	5.43
4	**75.47**	85.28	30.77	3.31	5.27
5	75.03	**85.34**	**30.80**	3.51	**5.24**
6	75.23	85.24	30.79	3.40	5.26
7	75.40	85.22	30.78	3.32	5.26
8	74.34	84.83	30.64	3.56	5.41

evaluated by segmenting the 1344 files of the TIMIT test set (the full test set). When a language model is used, a bigram language model is chosen. Embedded re-estimation is performed. The number of mixtures is fixed at 8.

Results: Figure 5.2 shows the variation of the segmentation performance with the window size, when no language model is used, and embedded re-estimation is performed. These results are given numerically in Table 5.5. Table 5.6 shows the results when a language model is used. The tables shows the average accuracy and percentage of boundaries correctly identified, the number of boundaries correctly matched (hits), insertions and deletions, over the full test set. It can be seen that the window size has a significant effect on the perceived segmentation performance. The use of a language model increases the segmentation performance, as the number of insertions is decreased. Many researchers mention segmentation accuracy at a window size of 20 ms, or a window size of 3, as used in this dissertation. When a language model is not used, the segmentation performance at a window size of 3 is equal to 69.70%, with a percentage correct of 87.43%. The average number of hits, insertions and deletions is equal to 31.53, 6.07, and 4.51, respectively. When a language model is used, the best segmentation accuracy at a window size of 3, is equal to 74.34%, with a percentage correct of 84.83%. The average number of hits, insertions and deletions is equal to 30.64, 3.56, and 5.41, respectively. The average number of true boundaries is 36.05 and the average number of observations (frames) is 305.71.

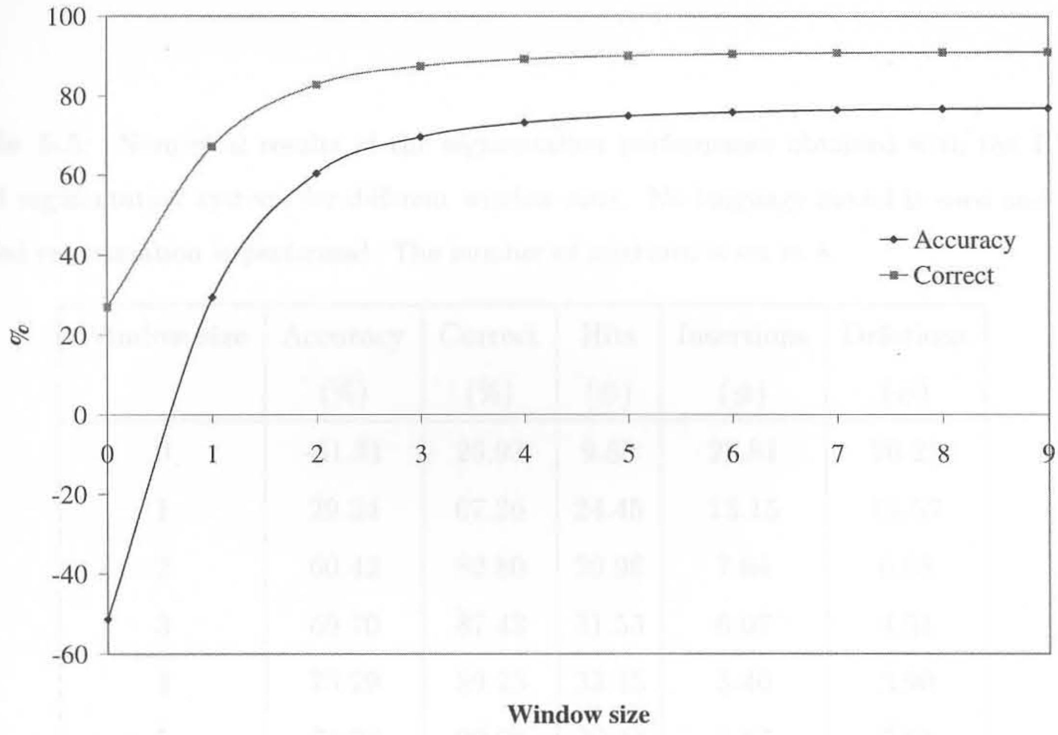

Figure 5.2: Effect of the window size (see Section 3.4 for a discussion of the window size and its mapping to time) on the segmentation performance. No language model is used, and embedded re-estimation performed. The number of mixtures is equal to 8.

The results of Section 5.1.1 (segmentation using HMMs) are summarised and compared in Chapter 6.

5.1.2 Recurrent neural network

For the segmentation of speech, using a recurrent neural network, or more precisely, a bi-directional recurrent neural network, three main parameters have an influence on the

Table 5.5: Numerical results of the segmentation performance obtained with the HMM based segmentation system, for different window sizes. No language model is used and embedded re-estimation is performed. The number of mixtures is set to 8.

Window size	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
0	-51.31	26.93	9.80	27.81	26.25
1	29.34	67.26	24.45	13.15	11.59
2	60.42	82.80	29.96	7.64	6.08
3	69.70	87.43	31.53	6.07	4.51
4	73.29	89.23	32.15	5.46	3.90
5	74.98	90.08	32.43	5.17	3.61
6	75.88	90.53	32.59	5.02	3.46
7	76.45	90.81	32.68	4.92	3.37
8	76.72	90.95	32.73	4.88	3.32
9	76.93	91.05	32.76	4.84	3.28

Table 5.6: Numerical results of the segmentation performance obtained with the HMM based segmentation system, for different window sizes. A bigram language model is used and embedded re-estimation is performed. The number of mixtures is set to 8.

Window size	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
0	-41.37	26.97	9.85	24.34	26.19
1	36.23	65.77	23.96	10.25	12.10
2	65.34	80.32	29.12	5.08	6.93
3	74.34	84.83	30.64	3.56	5.41
4	78.10	86.71	31.27	2.93	4.78
5	79.83	87.57	31.55	2.64	4.49
6	80.74	88.03	31.71	2.49	4.33
7	81.31	88.31	31.80	2.39	4.24
8	81.65	88.48	31.86	2.34	4.19
9	81.84	88.57	31.89	2.31	4.16

segmentation accuracy. The first is the number of hidden nodes in the network, that affects the network modeling capability. The second is the value of the threshold in the postprocessor, used to decide whether the probability of a boundary, as estimated by the neural network, is high enough to be regarded as a boundary or not. The final parameter that has a major influence on segmentation accuracy, is the window size, or allowed difference between the true and estimated phoneme boundary positions.

Number of hidden nodes

Purpose: The purpose of this experiment is to determine the effect that the number of hidden nodes in the bi-directional recurrent neural network has on the segmentation accuracy of a BRNN based segmentation system.

Experimental setup: The bi-directional recurrent neural network is trained using the 3696 TIMIT training files. The 1344 TIMIT test files (full test set) are then segmented. The neural network is trained for 250 iterations of backpropagation through time, with a learning rate of 0.001, with the weights in the network initialised uniformly between -0.15 and 0.15. For each number of hidden nodes, the network chosen is the one at the iteration which gives the best performance on the test set. The number of forward hidden nodes is equal to the number of hidden nodes, and the number backward hidden nodes as well. The total number of hidden nodes is thus equal to twice the number of hidden nodes, as mentioned here. In this experiment, the boundary threshold is fixed at 0.35 and the window size at 3.

Results: Figure 5.3 shows the effect that the number of hidden nodes has on the segmentation accuracy. These results are summarised numerically in Table 5.7. The table shows the average accuracy and percentage of boundaries correctly identified, the number of boundaries correctly matched (hits), insertions and deletions, over the full test set. It can be seen that number of hidden nodes has a significant effect on the segmentation performance. The best segmentation accuracy is obtained with 60 hidden nodes, and is equal to 80.12%, with a percentage correct of 86.20%. The average number of hits, insertions and deletions is equal to 31.14, 2.14, and 4.91, respectively. The average number of true boundaries is 36.05 and the average number of observations (frames) is 305.71.

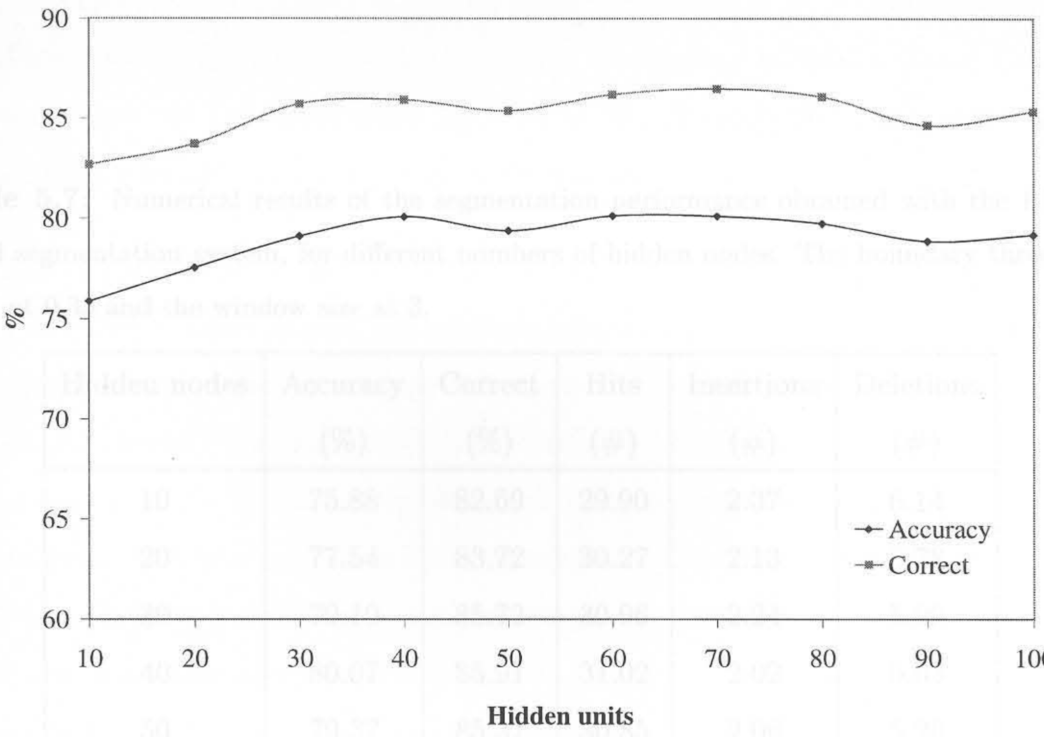

Figure 5.3: Effect of the number of neural network hidden nodes on the segmentation performance. The boundary threshold is set at 0.35 and the window size at 3.

Boundary threshold

Purpose: The purpose of this experiment is to determine the effect that the boundary threshold in the postprocessor of the neural network, has on the segmentation accuracy of a BRNN based segmentation system.

Experimental setup: The bi-directional recurrent neural network is trained using the 3696 TIMIT training files. The 1344 TIMIT test files (full test set) are then segmented. The neural network is trained for 250 iterations of backpropagation through time, with

Table 5.7: Numerical results of the segmentation performance obtained with the BRNN based segmentation system, for different numbers of hidden nodes. The boundary threshold is set at 0.35 and the window size at 3.

Hidden nodes	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
10	75.88	82.69	29.90	2.37	6.14
20	77.54	83.72	30.27	2.13	5.78
30	79.10	85.72	30.96	2.24	5.09
40	80.07	85.91	31.02	2.02	5.03
50	79.37	85.37	30.85	2.06	5.20
60	**80.12**	86.20	31.14	2.14	4.91
70	80.10	**86.48**	**31.22**	2.20	**4.82**
80	79.73	86.08	31.07	2.23	4.98
90	78.86	84.65	30.58	**1.99**	5.46
100	79.18	85.34	30.84	2.12	5.21

a learning rate of 0.001, with the weights in the network initialised uniformly between -0.15 and 0.15. For each number of hidden nodes, the network chosen is the one at the iteration (between 1 and 250) which gives the best performance on the test set. The number of forward hidden nodes (N_f) is equal to the number of backward hidden nodes (N_b). We call this the number of hidden nodes ($N_f = N_b = N_h$). The total number of hidden nodes (N_{htot}) is thus equal to twice the number of hidden nodes (N_h), as mentioned here. In this experiment $N_f = N_b = N_h$ is fixed at 50. The window size is set to 1 (the true and estimated boundaries may only differ by 1 frame).

Results: Figure 5.4 shows the effect that the boundary threshold has on the segmentation accuracy. These results are given numerically in Table 5.8. The table shows the average accuracy and percentage of boundaries correctly identified, the number of boundaries correctly matched (hits), insertions and deletions, over the full test set. It can be seen that the boundary threshold has a significant effect on the segmentation performance. The best segmentation accuracy is obtained with a boundary threshold of 0.35, and is equal to 68.41%, with a percentage correct of 79.89%. The average number of hits, insertions and deletions is equal to 28.95, 3.95, and 7.10, respectively. The average number of true boundaries is 36.05 and the average number of observations (frames) is 305.71.

Window size

Purpose: The purpose of this experiment is to determine the effect that the window size has on the segmentation accuracy of a BRNN based segmentation system.

Experimental setup: The bi-directional recurrent neural network is trained using the 3696 TIMIT training files. The 1344 TIMIT test files (full test set) are then segmented. The neural network is trained for 250 iterations of backpropagation through time, with a learning rate of 0.001, with the weights in the network initialised uniformly between -0.15 and 0.15. For each number of hidden nodes, the network chosen is the one at

Table 5.8: Numerical results of the segmentation performance obtained with the BRNN based segmentation system, for different boundary threshold values. The number of hidden units is set at 50 (for both the forward and backward hidden nodes) and the window size at 1.

Threshold	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
0.00	25.93	**86.53**	**31.28**	20.42	**4.77**
0.05	56.83	85.81	31.02	9.93	5.03
0.10	61.84	85.13	30.78	7.99	5.27
0.15	64.94	84.38	30.51	6.69	5.53
0.20	66.60	83.44	30.18	5.78	5.86
0.25	67.74	82.52	29.85	5.07	6.19
0.30	68.20	81.31	29.44	4.51	6.61
0.35	**68.42**	79.89	28.95	3.96	7.10
0.40	68.25	79.09	28.66	3.76	7.38
0.45	67.85	77.32	28.05	3.29	8.00
0.50	66.89	75.05	27.24	2.84	8.81
0.55	65.42	72.55	26.39	2.48	9.66
0.60	63.25	69.20	25.22	2.07	10.83
0.65	59.67	64.48	23.54	1.68	12.50
0.70	54.13	57.94	21.20	1.34	14.85
0.75	45.15	48.03	17.60	1.01	18.45
0.80	33.00	35.23	12.90	0.77	23.15
0.85	18.16	19.98	7.32	0.63	28.73
0.90	5.49	7.16	2.60	0.58	33.44
0.95	-0.46	0.77	0.27	0.42	35.77
1.00	-0.60	0.63	0.22	**0.42**	35.82

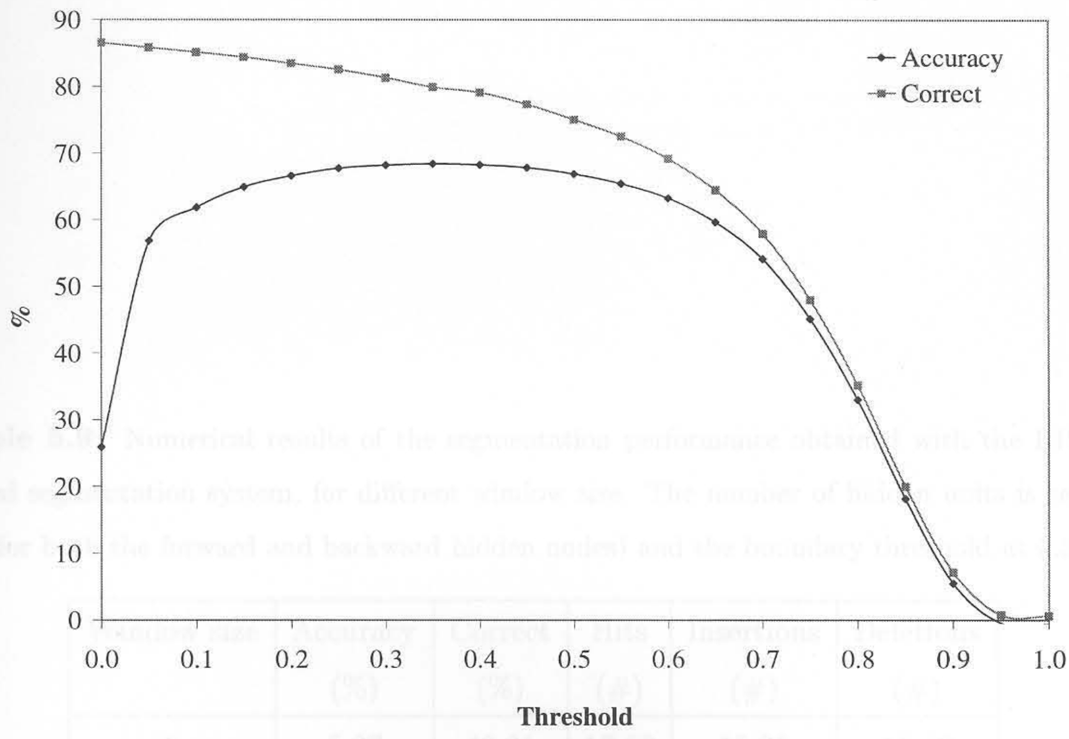

Figure 5.4: Effect of the threshold used at the neural network outputs to decide in favour of a boundary or not. The number of hidden nodes is 50 and the window size is 1.

the iteration (1 to 250) which gives the best performance on the test set. In this experiment, $N_f = N_b = N_h$ is fixed at 50. The boundary threshold is set at 0.35.

Results: Figure 5.5 shows the effect that the window size has on the segmentation accuracy. These results are given numerically in Table 5.9. The table shows the average accuracy and percentage of boundaries correctly identified, the number of boundaries correctly matched (hits), insertions and deletions, over the full test set. It can be seen that the window size has a significant effect on the segmentation performance. The accuracy at a window size of 20 ms, or a window size of 3, as used in this dissertation, is equal to 79.37%, with a percentage correct of 85.37%. The average number of hits, insertions and deletions is equal to 30.85, 2.06, and 5.20, respectively. The average number of true boundaries is 36.05 and the average number of observations (frames) is 305.71.

Table 5.9: Numerical results of the segmentation performance obtained with the BRNN based segmentation system, for different window size. The number of hidden units is set at 50 (for both the forward and backward hidden nodes) and the boundary threshold at 0.35.

Window size	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
0	5.27	48.31	17.56	15.35	18.49
1	68.42	79.89	28.95	3.96	7.10
2	76.66	84.01	30.39	2.52	5.66
3	79.37	85.37	30.85	2.06	5.20
4	80.36	85.86	31.02	1.89	5.03
5	80.86	86.11	31.10	1.81	4.94
6	81.14	86.25	31.15	1.76	4.90
7	81.34	86.35	31.18	1.72	4.86
8	81.47	86.41	31.20	1.70	4.84
9	81.56	86.46	31.22	1.70	4.83

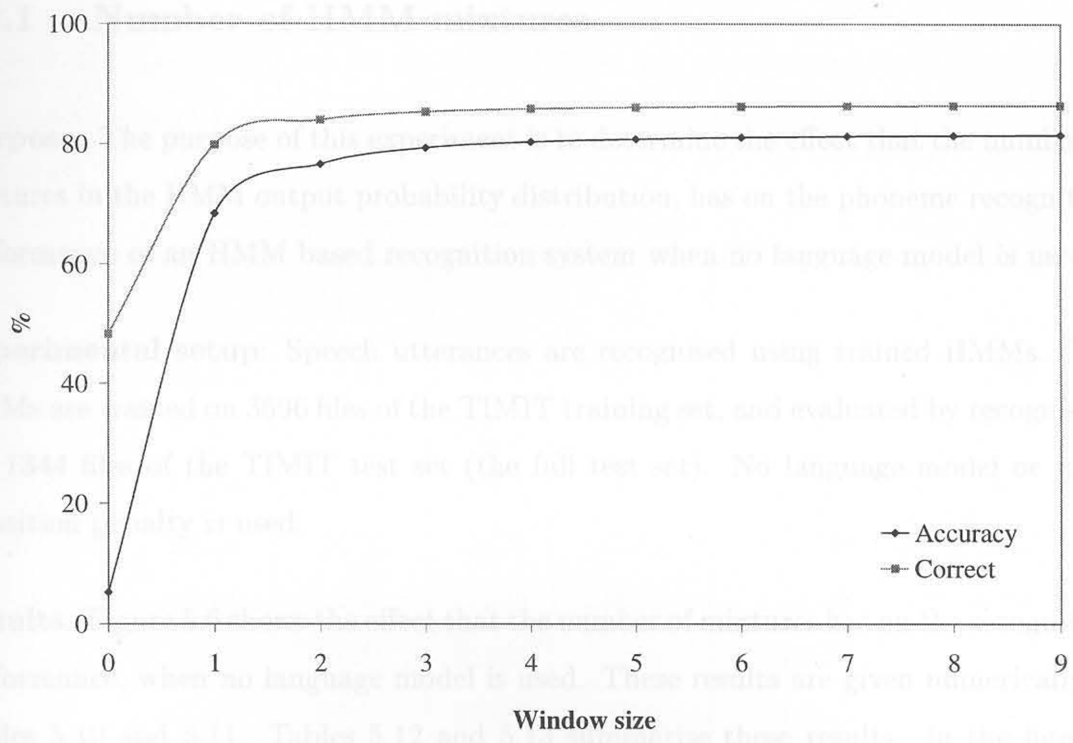

Figure 5.5: Effect of the window size on the segmentation performance of the BRNN. The number of hidden units is 50 and the boundary threshold is set at 0.35.

The results of Section 5.1.2 (segmentation using HMMs) are summarised and compared in Chapter 6. In conclusion it can be seen that the BRNN segments speech better than the HMM-based approach for TIMIT data not containing any noise.

5.2 Experiment 2: Speech recognition (baseline)

This section reports on the phoneme recognition performance of the baseline phoneme recogniser, based on HTK, as discussed in Chapter 4. Four different experiments are conducted. The first is to evaluate the effect of the number of mixtures in the HMM output probability distribution. The second and third experiments investigate the effect of a language model and word transition penalty, respectively. Finally, the combined use of both a language model and word transition penalty is investigated.

5.2.1 Number of HMM mixtures

Purpose: The purpose of this experiment is to determine the effect that the number of mixtures in the HMM output probability distribution, has on the phoneme recognition performance of an HMM based recognition system when no language model is used.

Experimental setup: Speech utterances are recognised using trained HMMs. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by recognising the 1344 files of the TIMIT test set (the full test set). No language model or word transition penalty is used.

Results: Figure 5.6 shows the effect that the number of mixtures has on the recognition performance, when no language model is used. These results are given numerically in Tables 5.10 and 5.11. Tables 5.12 and 5.13 summarise these results. In the figures, "HERest" indicates the use of additional embedded re-estimation iterations (see [3] for further details). When a language model and embedded re-estimation are not used, the maximum percentage correct is 62.17% and accuracy 52.37%, with 31554 hits, 4915 deletions, 14285 substitutions, and 4975 insertions. When no language model is used, but embedded re-estimation is performed, the maximum percentage correct is 64.80% and accuracy 54.08%, with 32890 hits, 4308 deletions, 13556 substitutions, and 5440 insertions. There are a total of 50754 phonemes in the complete test set.

5.2.2 Language model

Purpose: The purpose of this experiment is to determine the effect that the number of mixtures in the HMM output probability distribution, has on the phoneme recognition performance of an HMM based recognition system when a language model is used.

Experimental setup: Speech utterances are recognised using trained HMMs. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by recognising

Table 5.10: Numerical results of the baseline HMM recognition performance variation with the number of mixtures, when no language model is used, and no embedded re-estimation is performed.

Mixtures	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
1	53.37	44.64	27087	6204	17463	**4428**
2	57.52	48.32	29192	5570	15992	4670
3	59.26	49.22	30079	5145	15530	5096
4	60.73	49.96	30823	4983	14948	5468
5	61.56	49.66	31244	**4852**	14658	6041
6	62.04	51.91	31487	4976	14291	5141
7	61.97	52.08	31452	4966	14336	5017
8	**62.17**	**52.37**	**31554**	4915	**14285**	4975

Table 5.11: Numerical results of the baseline HMM recognition performance variation with the number of mixtures, when no language model is used, and embedded re-estimation is performed.

Mixtures	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
1	55.87	45.54	28354	5304	17096	**5243**
2	60.12	48.02	30514	4602	15638	6142
3	62.41	50.52	31674	4339	14741	6032
4	62.98	50.80	31963	4181	14610	6179
5	63.43	49.76	32194	**4097**	14463	6937
6	63.97	52.45	32469	4176	14109	5846
7	64.06	52.94	32515	4218	14021	5644
8	**64.80**	**54.08**	**32890**	4308	**13556**	5440

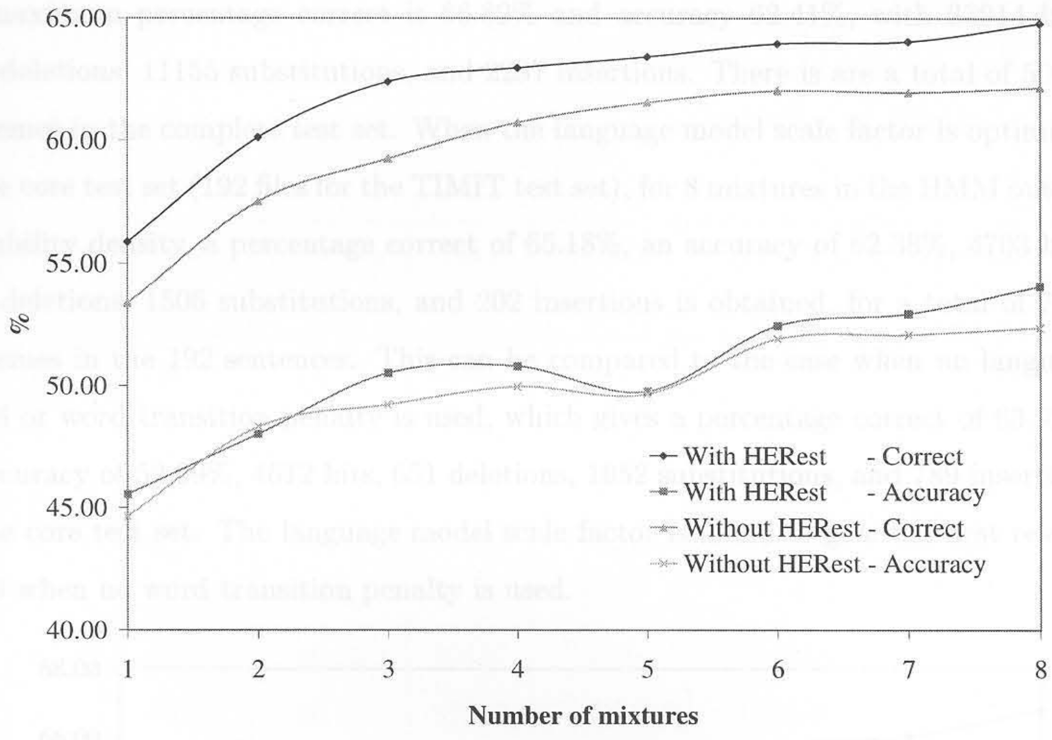

Figure 5.6: Effect of the number of mixtures on HMM phoneme recognition performance. No language model is used.

the 1344 files of the TIMIT test set (the full test set). A bigram language model is used. In the figures given in this experiment, the language model scale factor is set to 2.0. In order to determine the "optimal" value of the language model scale factor (s in Equation (2.45)), it is varied from 0.0 to 10.0 in steps of 1.0. The language model scale factor that results in the highest recognition accuracy is seen as the "optimal" value. A word transition penalty is not used.

Results: Figure 5.7 shows the effect that the number of mixtures has on the recognition performance, when a language model is used. These results are given numerically in Tables 5.12 and 5.13. In the figures, "HERest" indicates the use of additional embedded re-estimation iterations (see [3] for further details). When a language model is used, and embedded re-estimation is not used, the maximum percentage correct is 64.21% and accuracy 60.54%, with 32589 hits, 6695 deletions, 11470 substitutions, and 1861 insertions. When a language model is used, but embedded re-estimation is performed, the maximum percentage correct is 66.82% and accuracy 62.41%, with 33914 hits, 5685 deletions, 11155 substitutions, and 2237 insertions. There is are a total of 50754 phonemes in the complete test set. When the language model scale factor is optimised on the core test set (192 files for the TIMIT test set), for 8 mixtures in the HMM output probability density, a percentage correct of 65.18%, an accuracy of 62.38%, 4703 hits, 1006 deletions, 1506 substitutions, and 202 insertions is obtained, for a total of 7215 phonemes in the 192 sentences. This can be compared to the case when no language model or word transition penalty is used, which gives a percentage correct of 63.92%, an accuracy of 52.99%, 4612 hits, 651 deletions, 1952 substitutions, and 789 insertions on the core test set. The language model scale factor is found to give the best results at 4.0 when no word transition penalty is used.

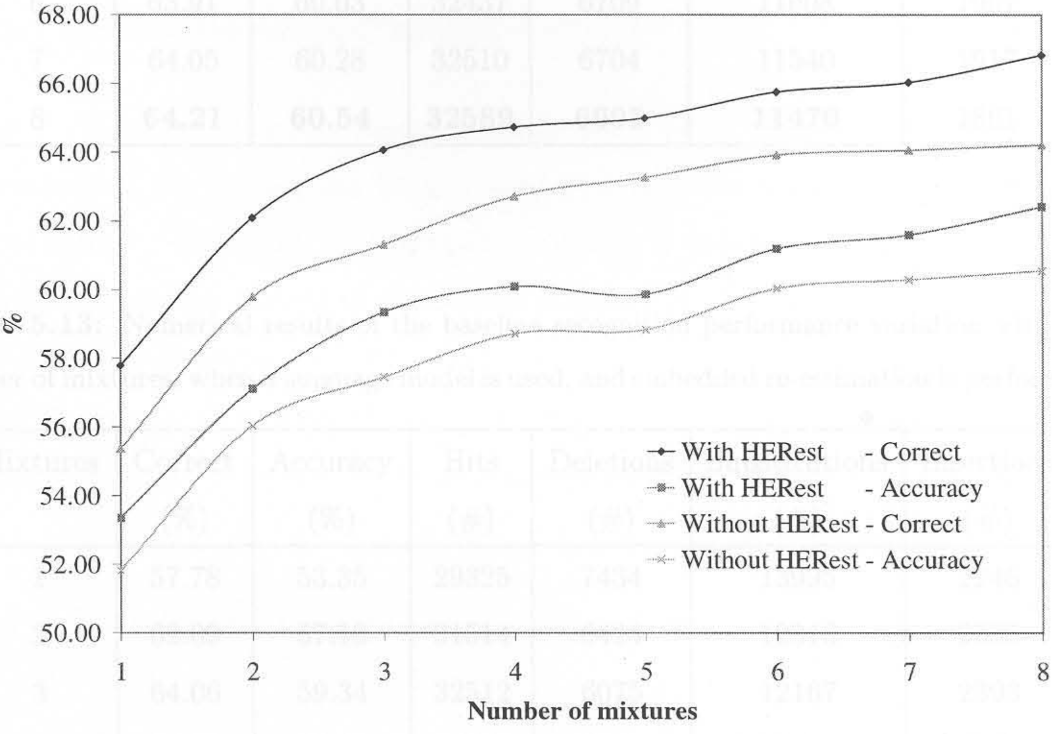

Figure 5.7: Effect of the number of mixtures on HMM phoneme recognition performance. A language model is used.

Table 5.12: Numerical results of the baseline recognition performance variation with the number of mixtures, when a language model is used, and no embedded re-estimation is performed.

Mixtures	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
1	55.37	51.83	28104	8655	13995	**1800**
2	59.78	56.04	30341	7615	12798	1897
3	61.31	57.44	31115	7185	12454	1962
4	62.71	58.71	31827	6861	12066	2031
5	63.26	58.83	32105	6790	11859	2246
6	63.91	60.03	32437	6709	11608	1967
7	64.05	60.28	32510	6704	11540	1917
8	**64.21**	**60.54**	**32589**	**6695**	**11470**	1861

Table 5.13: Numerical results of the baseline recognition performance variation with the number of mixtures, when a language model is used, and embedded re-estimation is performed.

Mixtures	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
1	57.78	53.35	29325	7434	13995	2246
2	62.09	57.10	31514	6424	12816	2535
3	64.06	59.34	32512	6075	12167	2393
4	64.73	60.09	32855	5949	11950	2355
5	64.99	59.86	32984	5905	11865	2605
6	65.75	61.19	33372	5783	11599	2318
7	66.03	61.59	33514	5821	11419	2254
8	**66.82**	**62.41**	**33914**	**5685**	**11155**	**2237**

5.2.3 Word transition penalty

Purpose: The purpose of this experiment is to determine the effect that a word transition penalty has on the phoneme recognition performance of an HMM based recognition system when a language model is not used.

Experimental setup: Speech utterances are recognised using trained HMMs. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by recognising the 192 files of the TIMIT test set (the core test set). A language model is not used. In order to determine the "optimal" value of the fixed word transition penalty (the bias term, w_b, in Equation (4.9)), experiments were performed with w_b varied from -10.0 to 20.0 in steps of 1.0. The word transition penalty that results in the highest recognition accuracy is seen as the "optimal" value.

Results: When no language model is used, but a word transition penalty is, the performance of the system can be increased over the case when no word transition penalty is used. When the word transition penalty is optimised on the TIMIT core test set, a percentage correct of 60.18%, an accuracy of 55.45%, 4342 hits, 1052 deletions, 1821 substitutions, and 341 insertions on the core test set (containing a total of 7215 phonemes). The word transition penalty is found to give the best recognition performance when set to -6.0, when no language model is used.

5.2.4 Combined language model and word transition penalty

Purpose: The purpose of this experiment is to determine the effect that the combined use of a language model and word transition penalty has on the phoneme recognition performance of an HMM based recognition system.

Experimental setup: Speech utterances are recognised using trained HMMs. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by recognising the 192 files of the TIMIT core test set. A bigram language model is used. In order to determine the "optimal" values of the language model scale factor (s in Equation (2.45)) and fixed word transition penalty (the bias term, w_b, in Equation (4.9)), they are varied from 0.0 to 10.0, and -10.0 to 20.0, respectively, in steps of 1.0. The values that result in the highest recognition accuracy is seen as the "optimal" values.

Results: When a language model is used, as well as a word transition penalty, the performance of the system can be increased significantly. When the language model scale factor and word transition penalty is jointly optimised on the TIMIT core test set, a percentage correct of 67.05%, an accuracy of 63.23%, 4838 hits, 820 deletions, 1557 substitutions, and 276 insertions on the core test set (containing a total of 7215 phonemes). The language model scale factor and word transition penalty is found to give the best recognition performance when both are set to 5.0. Figure 5.8 shows the general trend of the recognition performance, as the language model scale factor is varied, with the fixed word transition penalty (bias) fixed at 17.0. Figure 5.9 shows the general trend of the recognition performance, as the fixed (bias) word transition penalty is varied, with the language model scale factor fixed at 4.0.

5.3 Experiment 3: Speech recognition using segmentation information

This section reports results on the recognition of speech using segmentation information, as discussed in detail in Chapter 4. Results on the modification of the HMM transition probabilities (both linear and non-linear), as well as an adaptive word transition penalty, are given. The combined effect of transition and word transition penalty modification is also evaluated.

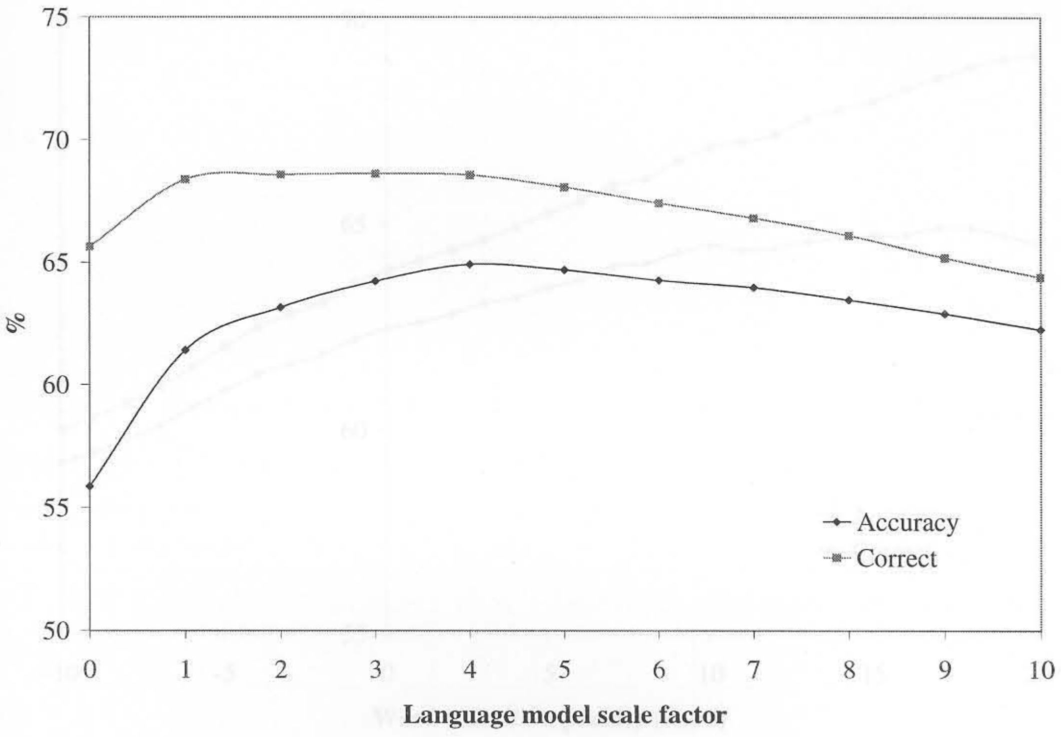

Figure 5.8: Effect of the language model scale factor on HMM recognition performance. A fixed (bias) word transition penalty of 17.0 is used.

5.3.1 HMM transition probability modification

Purpose: The purpose of this experiment is to determine the effect that the modification of HMM transition probabilities, from the segmentation information, has on the phoneme recognition performance of an HMM based recognition system.

Experimental setup: Speech utterances are recognised using trained HMMs. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by recognising the 192 files of the TIMIT test set (the core test set). When a language model is used, a bigram language model is chosen, and the word transition penalty used is just the fixed bias term, as described in Chapter 4. In order to determine the "optimal" values of the language model scale factor (s in Equation (2.45)) and fixed word transition penalty (the bias term, w_b, in Equation (4.9)), they are varied from 0.0 to 10.0, and -10.0 to

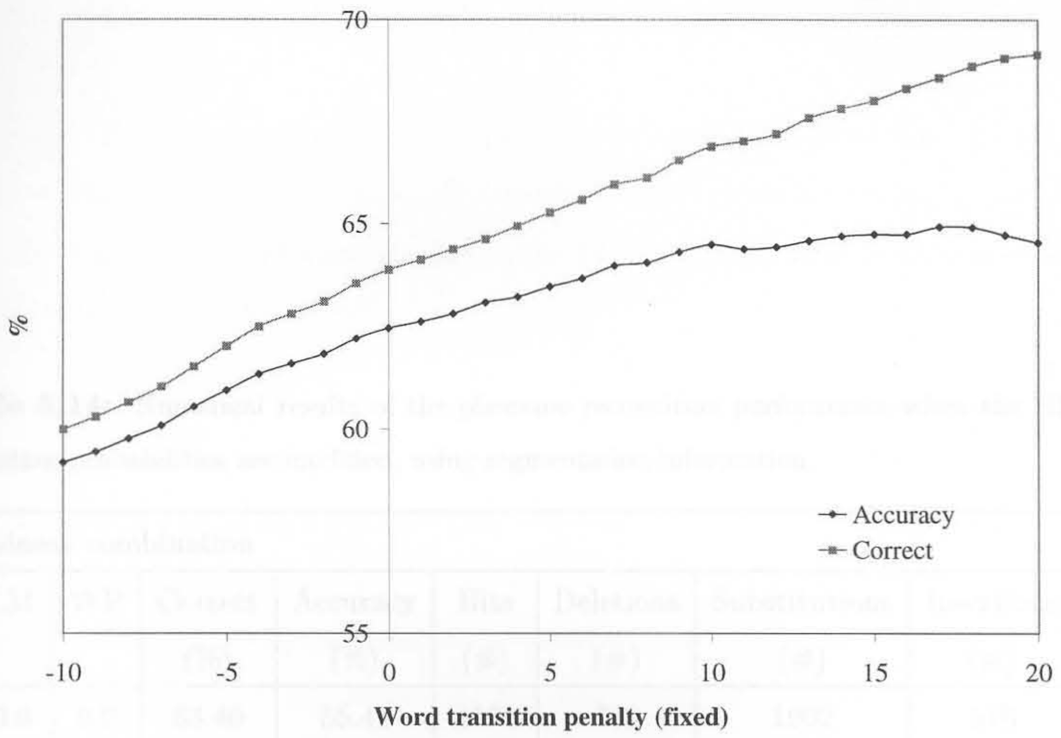

Figure 5.9: Effect of the fixed (bias) word transition penalty on HMM recognition performance. A language model scale factor of 4.0 is used.

20.0, respectively, in steps of 1.0. The values that result in the highest recognition accuracy is seen as the "optimal" values.

Results: The use of segmentation information, to modify the HMM transition probabilities, can increase the recognition performance of the system. Table 5.14 gives the results numerically. In this table, "LM" indicates the language model scale factor and "WP" indicates the fixed word transition penalty (bias). Results are given for both linear and non-linear combination of segmentation information with HMM transition probabilities.

Table 5.14: Numerical results of the phoneme recognition performance when the HMM transition probabilities are modified, using segmentation information.

Linear combination							
LM	WP	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
0.0	0.0	63.40	55.41	4574	739	1902	576
0.0	-4.0	61.00	56.34	4401	989	1825	336
3.0	0.0	65.29	62.54	4711	963	**1541**	**199**
5.0	8.0	**67.78**	**63.58**	**4890**	**730**	1595	303

Non-linear combination							
LM	WP	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
0.0	0.0	64.27	52.47	4637	**620**	1958	851
0.0	-5.0	61.29	55.56	4422	934	1859	413
4.0	0.0	65.36	62.37	4716	976	**1523**	**216**
5.0	5.0	**67.08**	**63.02**	**4840**	805	1570	293

5.3.2 HMM word transition penalty modification

Purpose: The purpose of this experiment is to determine the effect that the modification of HMM word transition penalty (used here as a phoneme transition penalty), from the segmentation information, has on the phoneme recognition performance of an HMM based recognition system.

Experimental setup: Speech utterances are recognised using trained HMMs. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by recognising the 192 files of the TIMIT test set (the core test set). When a language model is used, a bigram language model is chosen, and the word transition penalty used is the fixed bias term, plus two additional time-varying components, as described in Chapter 4. In order to determine the "optimal" values of the language model scale factor (s in Equation (2.45)) and fixed word transition penalty (the bias term, w_b, in Equation (4.9)), they are varied from 0.0 to 10.0, and -10.0 to 20.0, respectively, in steps of 1.0. The values that result in the highest recognition accuracy is seen as the "optimal" values. The scale factors (a and b) for the two time-varying components (w_p and w_c in Equation (4.9)) of the word transition penalty are set equal to each other and varied from 0.0 to 15.0 in steps of 1.0, in order to determine the "optimum" scale factor.

Results: The use of segmentation information, to modify the HMM word transition penalty term, can increase the recognition performance of the system. Table 5.15 gives the results numerically. In this table, "LM" indicates the language model scale factor, "WP" indicates the fixed word transition penalty (bias), and "AWP" indicates the scale factors ((a and b in Equation (4.9)) used for the two time-varying word transition penalty terms (w_c and w_p in Equation (4.9)). Figure 5.10 shows the effect that the adaptive word transition penalty terms have on the recognition performance, when both a language model and fixed word transition penalty term are used.

Table 5.15: Numerical results of the phoneme recognition performance when the word transition penalty is modified, using segmentation information.

LM	WP	AWP	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
0.0	0.0	9.0	61.98	59.40	4472	859	1884	186
0.0	1.0	9.0	62.33	59.60	4497	819	1899	197
3.0	0.0	3.0	65.07	63.01	4695	963	**1557**	**149**
4.0	17.0	10.0	**68.58**	**64.93**	**4948**	**610**	1657	263

5.3.3 Combined transition probability and word penalty modification

Purpose: The purpose of this experiment is to determine the effect that the combined modification of HMM transition probabilities and word transition penalty (used here as a phoneme transition penalty), from the segmentation information, has on the phoneme recognition performance of an HMM based recognition system.

Experimental setup: Speech utterances are recognised using trained HMMs. The HMMs are trained on 3696 files of the TIMIT training set, and evaluated by recognising the 192 files of the TIMIT core test set. When a language model is used, a bigram language model is chosen, and the word transition penalty used is just the fixed bias term, as described in Chapter 4. In order to determine the "optimal" values of the language model scale factor (s in Equation (2.45)) and fixed word transition penalty (the bias term, w_b, in Equation (4.9)), they are varied from 0.0 to 10.0, and -10.0 to 20.0, respectively, in steps of 1.0. The values that result in the highest recognition accuracy is seen as the "optimal" values. The scale factors (a and b) for the two time-varying components (w_p and w_c in Equation (4.9)) of the word transition penalty are set equal to each other and varied from 0.0 to 15.0 in steps of 1.0, in order to determine

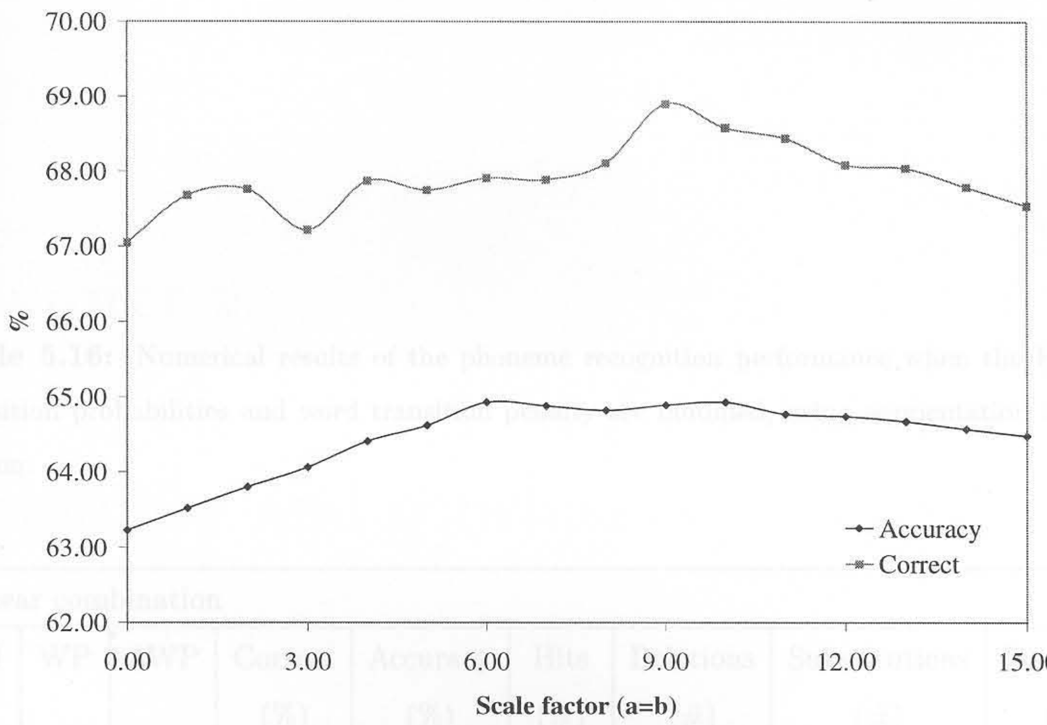

Figure 5.10: Effect of the adaptive word transition penalty scale factor on the phoneme recognition performance.

the "optimum" scale factor.

Results: The use of segmentation information, to modify the HMM transition probabilities, as well as the word transition penalty term, can increase the recognition performance of the system. Table 5.16 gives the results numerically. In this table, "LM" indicates the language model scale factor, "WP" indicates the fixed word transition penalty (bias), and "AWP" indicates the scale factors ((a and b in Equation (4.9)) used for the two time-varying word transition penalty terms (w_c and w_p in Equation (4.9)). Results are given for both linear and non-linear combination of segmentation information with HMM transition probabilities.

Table 5.16: Numerical results of the phoneme recognition performance when the HMM transition probabilities and word transition penalty are modified, using segmentation information.

Linear combination

LM	WP	AWP	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
0.0	0.0	8.0	61.66	59.22	4449	882	1884	**176**
0.0	3.0	9.0	62.56	59.58	4514	785	1916	215
3.0	0.0	1.0	65.03	62.73	4692	986	**1537**	166
4.0	17.0	8.0	**68.84**	**64.91**	**4967**	**597**	1651	284

Non-linear combination

LM	WP	AWP	Correct (%)	Accuracy (%)	Hits (#)	Deletions (#)	Substitutions (#)	Insertions (#)
0.0	0.0	9.0	61.91	59.21	4467	841	1907	195
0.0	5.0	11.0	62.80	59.33	4531	727	1957	250
3.0	0.0	3.0	65.35	63.10	4715	943	**1557**	**162**
4.0	15.0	9.0	**68.30**	**64.80**	**4928**	**616**	1671	253

Chapter 6

Summary and conclusion

This chapter summarises the results given in Chapter 5 and we draw some conclusions from it. In particular, the best results obtained with the various techniques tested, are repeated here, and the improvement over the baseline systems are given and compared with each other. The shortcomings and possible future work, related to that done in this dissertation, are also discussed here.

6.1 Summary of results

For the purposes of comparing the various techniques, we define the improvement as

$$Improvement = \frac{new - old}{old} \cdot 100\%, \tag{6.1}$$

where *new* indicates the improved system performance, and *old* indicates the original or baseline system. The improvement is multiplied by -1 when it is calculated for insertions, deletions and substitutions, as these are to be minimised, not maximised. Using this convention, any improvement with a negative value, indicates that performance

Table 6.1: Summary of results showing the segmentation performance obtained with the HMM based segmentation system, for different experimental conditions.

System no.	RE	LM	Mixs. (#)	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
1	No	0.0	8	71.23	87.07	31.40	5.40	4.64
2	No	2.0	7	**75.62**	84.03	30.33	**2.84**	5.72
3	Yes	0.0	7	70.27	**88.04**	**31.76**	6.07	**4.29**
4	Yes	2.0	4	75.47	85.28	30.77	3.31	5.27

degraded, rather than improved.

Table 6.1 summarises the segmentation performance obtained with the HMM based segmentation system, for different experimental conditions. In this table, "RE" indicates embedded re-estimation of the HMM models, and "LM" indicates the use of a language model with the corresponding values in the table indicating the language model scale factor (0.0 if no language model is used). From this table it can be seen that HMM segmentation performance is higher, if accuracy is the criterium, when no embedded re-estimation is used during the training process. This is primarily due to the fact that phoneme recognition performance is maximised and not segmentation performance. The use of a language model also improves segmentation performance significantly. This is due to the fact that fewer insertions occur. Again it should be noted that even though the best performances were obtained with 8, 7, and 4 mixtures, 2 to 3 mixtures would probably have been sufficient to model the acoustic space due to the fact that the TIMIT database only consists of two genders (male and female).

Table 6.2 shows the results obtained for BRNN based segmentation. Here the best performance is obtained with 60 forward hidden nodes, and 60 backward hidden nodes, indicated just by "hidden nodes" in the table. The threshold in the postprocessor, used to decide when the probability of a boundary is high enough to indicate the presence of a phoneme boundary, is found to be best at 0.35. This table shows that a segmentation

Table 6.2: Summary of results showing the segmentation performance obtained with the BRNN based segmentation system (with 60 forward and backward hidden nodes), for different experimental conditions. Also shown is the improvement over the HMM based segmentation systems.

Hidden nodes	Threshold	Accuracy (%)	Correct (%)	Hits (#)	Insertions (#)	Deletions (#)
60	0.35	80.12	86.20	31.14	2.14	4.91
Improvement over HMM based segmentation systems						
HMM system						
1		12.59	-1.00	-0.83	60.37	-5.82
2		6.06	**2.58**	**2.67**	24.65	**14.16**
3		**14.13**	-2.09	-1.95	**64.74**	-14.45
4		6.27	1.08	1.20	35.35	6.83

accuracy of 80.12% is obtained with the BRNN based segmentation system, in contrast to the 71.23%, 75.62%, 70.27%, and 75.47% accuracies in Table 6.1 of HMM based segmentation systems 1, 2, 3, and 4, respectively. The corresponding improvement in accuracy of the BRNN based segmentation system over the HMM based segmentation systems is also shown in Table 6.2 as 12.59%, 6.06%, 14.13% and 6.27%, respectively. The BRNN based segmentation system thus significantly outperforms all of the HMM based segmentation systems. This can be partially explained by the fact that the neural network is able to use all of the context information in the prediction of the phoneme boundaries, in effect learning a better "language model" than that of the HMM systems. The neural network is able to efficiently discriminate between a phoneme boundary and no phoneme boundary, while the HMM systems can only discriminate between two different phonemes.

Table 6.3 shows the baseline recognition performance of the HMM recognition system, for different experimental conditions. In this table, "WP" indicates the use of a fixed

Table 6.3: Summary of results showing the baseline recognition performance, for different experimental conditions.

System no.	RE	LM	WP	Cor (%)	Acc (%)	Hits (#)	Del (#)	Sub (#)	Ins (#)
1	No	0.0	0.0	61.76	52.40	4456	**713**	2046	675
2	No	0.0	-6.0	57.56	53.96	4153	1237	1825	260
3	No	4.0	0.0	62.62	60.58	4518	1178	**1519**	**147**
4	No	5.0	5.0	**64.24**	**61.44**	**4635**	986	1594	202
5	Yes	0.0	0.0	63.92	52.99	4612	**651**	1952	789
6	Yes	0.0	-6.0	60.18	55.45	4342	1052	1821	341
7	Yes	4.0	0.0	65.18	62.38	4703	1006	**1506**	**202**
8	Yes	5.0	5.0	**67.05**	**63.23**	**4838**	820	1557	276

word transition penalty (the constant bias term, w_b with $c = 1$, in Equation (4.12)). It can be seen that the use of embedded re-estimation during the training process improves recognition performance. When a language model and fixed word transition penalty are used separately, the performance can be increased. The language model significantly decreases the amount of substitutions and insertions, as it models the probability of one phoneme following another. The use of a word transition penalty has a similar effect, decreasing the likelihood of a transition from one phoneme to another, resulting in a reduction in the number of insertions. When both a language model and word transition penalty are used, the best performance is obtained. These two parameters must be jointly optimised on the test set.

Table 6.4 shows the phoneme recognition performance when segmentation information is included into the decoding process. In this table, "AWP" indicates the use of the two adaptive word transition penalty terms (the first two terms in Equation (4.12)), with the corresponding values in the table indicating the adaptive word transition penalty scale factor (same factor is used for both terms). In systems 1 to 4, the transition probabilities of the HMMs are modified from the segmentation information, using a linear combination of segmentation probabilities and HMM transition probabilities (indicated by "Trans. (Linear)"). In systems 5 to 8 a non-linear combination is used ("Trans. (Non-linear)"). Systems 9 to 12 indicate the use of only the adaptive word transition penalty ("AWP"). Finally, systems 13 to 20 are similar to systems 1 to 8, where the adaptive word transition penalty is used in addition to the HMM transition probability modification. Results are shown for the cases when no language model or fixed word transition penalty is used, when only a language model or fixed word transition penalty is used, and when both are used simultaneously. It can be seen that phoneme recognition performance improves in all the cases when either or both a language model and fixed word transition penalty are used. The linear combination of HMM transition probabilities and segmentation probabilities is slightly better than the non-linear combination. This is due to the fact that both the HMM transition probability and segmentation probabilities are used in linear combination, instead of only the maximum of the two for non-linear combination. Linear combination is thus less susceptible to "noisy" estimates of segmentation probability and in effect performs a smoothing function as a result of the averaging procedure. The use of only adaptive word transition terms is also superior to the other systems. When both the HMM transition probability modification and adaptive word transition penalty are used, the performance is still not as good as that obtained when only the adaptive word transition penalty is used. This may be attributed to the fact that the two techniques may oppose each other in a small way.

Table 6.5 gives the improvements obtained with the use of segmentation information, as shown in Table 6.4, over that of the baseline recognition systems 5 to 8, given in Table 6.3, for the different cases when no language model or fixed word transition penalty is used (combined systems 1, 5, 9, 13 and 17, over baseline system 1), when only a fixed word transition penalty is used (combined systems 2, 6, 10, 14, and 18, over baseline system 2), when only a language model is used (combined systems 3, 7, 11, 15, and 19, over baseline system 3), and when both a language model and fixed word transition penalty are used (combined systems 4, 8, 12, 16, and 20, over baseline

Table 6.4: Summary of results showing the recognition performance with segmentation information included, for different experimental conditions.

System no.	Type	LM	WP	AWP	Cor (%)	Acc (%)	Hits (#)	Del (#)	Sub (#)	Ins (#)
1	Trans.	0.0	0.0	0.0	63.40	55.41	4574	739	1902	576
2		0.0	-4.0	0.0	61.00	56.34	4401	989	1825	336
3	(Linear)	3.0	0.0	0.0	65.29	62.54	4711	963	**1541**	**199**
4		5.0	8.0	0.0	**67.78**	**63.58**	**4890**	**730**	1595	303
5	Trans.	0.0	0.0	0.0	64.27	52.47	4637	**620**	1958	851
6		0.0	-5.0	0.0	61.29	55.56	4422	934	1859	413
7	(Non-linear)	4.0	0.0	0.0	65.36	62.37	4716	976	**1523**	**216**
8		5.0	5.0	0.0	**67.08**	**63.02**	**4840**	805	1570	293
9	AWP	0.0	0.0	9.0	61.98	59.40	4472	859	1884	186
10		0.0	1.0	9.0	62.33	59.60	4497	819	1899	197
11		3.0	0.0	3.0	65.07	63.01	4695	963	**1557**	**149**
12		4.0	17.0	10.0	**68.58**	**64.93**	**4948**	**610**	1657	263
13	Trans. and	0.0	0.0	8.0	61.66	59.22	4449	882	1884	176
14	AWP	0.0	3.0	9.0	62.56	59.58	4514	785	1916	215
15		3.0	0.0	1.0	65.03	62.73	4692	986	**1537**	**166**
16	(Linear)	4.0	17.0	8.0	**68.84**	**64.91**	**4967**	**597**	1651	284
17	Trans. and	0.0	0.0	9.0	61.91	59.21	4467	841	1907	195
18	AWP	0.0	5.0	11.0	62.80	59.33	4531	727	1957	250
19		3.0	0.0	3.0	65.35	63.10	4715	943	**1557**	**162**
20	(Non-linear)	4.0	15.0	9.0	**68.30**	**64.80**	**4928**	**616**	1671	253

system 4). Here it can be seen that improvement over the best baseline recogniser (baseline system 4), from the use of linear HMM transition probability modification is 0.55%, non-linear HMM transition probability modification -0.33%, adaptive word transition penalty only 2.69%, adaptive word transition penalty and linear HMM transition probability modification 2.66%, adaptive word transition penalty and non-linear HMM transition probability modification 2.48%. It is clear that the technique developed in this dissertation (the adaptive word transition penalty), is superior to that proposed by others (the HMM transition probability modification). It is also interesting to note that improvements can be obtained when no language model or word transition penalty is used by the recognition system. Improvement is also significant when only a word transition penalty is used. A slight improvement is still obtained when only a language model is used, and no word transition penalty.

6.2 Statistical significance

In order to determine whether the improvement obtained is statistically significant, a test of significance, or one-tailed hypothesis test of the difference of proportions, must be performed [89]. Let the null hypothesis be denoted by H_0, representing the hypothesis that there is no statistical difference between the baseline system and the modified system (with the neural network). Let the alternative hypothesis, H_1 represent the hypothesis that there is a statistical significance. The null hypothesis at a certain level of significance, α, is then rejected if the z score lies outside a certain range, indicating that the improvement is statistically significant. Otherwise the null hypothesis is accepted, indicating that there is no statistical significance. Hypothesis H_0 is thus accepted if

$$Z = \frac{P_1 - P_2}{\sigma_{P_1 - P_2}} < \alpha, \tag{6.2}$$

where $\tilde{P} = \frac{n_1 P_1 + n_2 P_2}{n_1 + n_2}$ is used as an estimate of the population proportion p, $n_1 = n_2 =$

Table 6.5: Summary of results showing the improvement of the recognition performance with segmentation information included, for different experimental conditions. The improvement is shown only for embedded re-estimation cases.

Improvement over HMM baseline recognition systems							
System no.	Type	Cor (%)	Acc (%)	Hits (#)	Del (#)	Sub (#)	Ins (#)
1	Trans.	-0.81	**4.57**	-0.82	-13.52	**2.56**	**27.00**
2		**1.36**	1.61	**1.36**	5.99	-0.22	1.47
3	(Linear)	0.17	0.26	0.17	4.27	-2.32	1.49
4		1.09	0.55	1.07	**10.98**	-2.44	-9.78
5	Trans.	0.55	-0.98	0.54	4.76	**-0.31**	-7.86
6		**1.84**	**0.20**	**1.84**	**11.22**	-2.09	-21.11
7	(Non-linear)	0.28	-0.02	0.28	2.98	-1.13	-6.93
8		0.04	-0.33	0.04	1.83	-0.83	**-6.16**
9	AWP	-3.04	**12.10**	-3.04	-31.95	**3.48**	**76.43**
10		**3.57**	7.48	**3.57**	22.15	-4.28	42.23
11		-0.17	1.01	-0.17	4.27	-3.39	26.24
12		2.28	2.69	2.27	**25.61**	-6.42	4.71
13	Trans. and	-3.54	**11.76**	-3.53	-35.48	**3.48**	**77.69**
14	AWP	**3.95**	7.45	**3.96**	25.38	-5.22	36.95
15		-0.23	0.56	-0.23	1.99	-2.06	17.82
16	(Linear)	2.67	2.66	2.67	**27.20**	-6.04	-2.90
17	Trans. and	-3.14	**11.74**	-3.14	-29.19	**2.31**	**75.29**
18	AWP	**4.35**	7.00	**4.35**	**30.89**	-7.47	26.69
19		0.26	1.15	0.26	6.26	-3.39	19.80
20	(Non-linear)	1.86	2.48	1.86	24.88	-7.32	8.33

N, P_1 is the percentage accuracy of the modified system, divided by 100 (to get a proportion), and P_2 is the percentage accuracy of the baseline system, divided by 100. The standard deviation, $\sigma_{P_1-P_2}$ is given by

$$\sigma_{P_1-P_2} = \sqrt{p(1-p)\left(\frac{1}{n_1}+\frac{1}{n_1}\right)}. \tag{6.3}$$

The above equations can be rewritten in terms of the improvement in accuracy, as

$$Improvement < \frac{\alpha \cdot \sigma_{P_1-P_2}}{P_2} \cdot 100\%, \tag{6.4}$$

where $Improvement$ is defined in Equation (6.1). This equation indicates the maximum percentage improvement allowed in order to accept hypothesis H_0, and the difference between the results of the two experiments to be statistically insignificant.

6.3 Conclusion

From the Section 6.1 it can be seen that the most important segmentation results are that of the segmentation accuracy of HMM segmentation system number 2 in Table 6.1, as well as the RNN segmentation system in Table 6.2. The most important results of the phoneme recognition experiments, are that of phoneme recognition systems 4 and 12 in Table 6.4, where a neural network is incorporated into the recognition process. The improvement of the RNN based segmentation system over HMM based segmentation system number 2, is shown in Table 6.2. The phoneme recognition accuracy improvements of the modified phoneme recognition systems, over baseline system 8 in Table 6.3, are summarised in Table 6.5.

To determine whether the segmentation accuracy improvement is statistically significant, the accuracies (divided by 100) of the BRNN based segmentation system in

Table 6.6: Minimum percentage improvement required, at various levels of significance, so that the improvement in segmentation accuracy is statistically significant.

z	α	Improvement required (%)
1.28	0.1	0.45
1.645	0.05	0.58
1.96	0.025	0.69
2.33	0.01	0.82

Table 6.2 is used as P_1, and the accuracy (divided by 100) of baseline system 2 in Table 6.1 is used as P_2, and the minimum percentage improvement required to be statistically significant, at a specific significance level, can thus be calculated. Here $N = 48446$ is the total number of phoneme boundaries in the TIMIT full test set. Table 6.6 shows the improvement required to be statistically significant, for various levels of significance.

In the Section 6.1 it is shown that the best accuracy obtained with the baseline HMM segmentation system number 2 is 75.62%. The best segmentation performance obtained with the BRNN based segmentation system is 80.12%. When this improvement of 6.06% is compared with the values in Table 6.6, it is clear that the BRNN based segmentation system significantly outperforms the HMM based segmentation system.

When the accuracies (divided by 100) of phoneme recognition systems 4 and 12 in Table 6.4 are used as P_1, and the accuracy (divided by 100) of baseline system 8 in Table 6.3 is used as P_2, then the minimum percentage improvement required to be statistically significant, at a specific significance level, can be calculated. Here $N = 7215$ is the total number of phonemes in the TIMIT core test set. Table 6.7 shows approximate values for phoneme recognition systems 4 and 12, at various levels of significance.

In the Section 6.1 it is shown that the best accuracy obtained with the baseline system

Table 6.7: Minimum percentage improvement required, at various levels of significance, so that the improvement in phoneme recognition accuracy is statistically significant.

z	α	Improvement required (%)
1.28	0.1	1.62
1.645	0.05	2.08
1.96	0.025	2.48
2.33	0.01	2.94

is 63.23%. When the HMM transition probabilities are combined linearly with the neural network outputs, the best accuracy is 63.58%, with a corresponding improvement of 0.55%. When this improvement is compared with the values in Table 6.7, it is clear that this technique, used by others, does not give a statistically significant improvement, in the experiments we conducted. When an adaptive word penalty is used (the technique developed in this dissertation) the best accuracy is 64.93%, with a corresponding improvement over the baseline system of 2.69%. This improvement is statistically significant up to a significance level of 0.025 (97.5% confidence). The technique developed in our work thus not only outperforms that used by others, but also gives a statistically significant improvement in the phoneme recognition accuracy.

6.4 Shortcomings and future work

The work presented in this dissertation is partially limited due to the significant amount of computational power needed. In particular, the following future work needs to be carried out, that could not be done here due to the limited computational resources and time constraints, namely to

- find the best neural network architecture, where experiments are performed to determine the optimal number of forward and backward hidden nodes separately (in this dissertation, the forward and backward hidden nodes are set equal to each other, resulting in many "unused" weights),

- determine the best features for segmentation and recognition separately (in this dissertation the same features were used for segmentation and recognition, which may not be optimal),
- investigate the real-time implementation of such a system (in this dissertation, segmentation is performed off-line first, before recognition is performed, that uses the segmentation information),
- investigate the use of better segmentation postprocessors, e.g. based on segmentation lattices and dynamic programming, instead of just a threshold function, and
- explore performance of NN and HMM segmentation and recognition when the test corpus is clean (i.e., noise-free), but consists of non-read speach or speakers not included in the training sets.

Appendix A

Recurrent neural network training

This chapter discusses the training of recurrent neural networks, and more specifically, bi-directional recurrent neural networks. The training of conventional recurrent neural networks is not explicitly considered here, as only bi-directional recurrent neural networks were used in this dissertation. BRNNs can also simplify to conventional RNNs when the number of backward hidden neurons is set to zero. In Section A.1 a general overview of gradient descent training is given, while backpropagation through time is discussed in Section A.2. The specific equations used to train the BRNNs as used in this dissertation, are then given in Section A.3. Finally, Section A.3.6 gives the strategy used to make sure that the network has a good generalisation capability.

A.1 Gradient descent training

Gradient descent, also called steepest descent, is one of the most commonly used and simplest training algorithms for neural networks [69]. In gradient descent, the idea is to adjust the network weights in the direction of the greatest rate of decrease in error.

Initial values need to be chosen for the weights for gradient descent to function correctly. Often, and as used in this dissertation, the weights are initialised uniformly from $-\epsilon$ to ϵ, where ϵ is a value that determines the range of initial weight values.

The next step in gradient descent training involves the calculation of the gradient of the error, with respect to the weights in the network, ∇E, as

$$\nabla E = \frac{\partial E}{\partial w_{ji}}, \tag{A.1}$$

where w_{ji} is the weight to which the gradient is calculated.

The weights can then be updated by adding a fraction of the negative of the gradient to the weight, or

$$w_{ji}^{t} = w_{ji}^{t-1} - \eta \nabla E, \tag{A.2}$$

where ∇E is defined in Equation (A.1) and η is called the learning rate. Provided that η is sufficiently small, the value of the error E should steadily decrease, allowing the network to learn.

A.2 Backpropagation through time

In the previous section, it was shown that the gradient of the error with respect to the network weights, is needed for the network to be able to learn. Backpropagation provides a particularly effective way to calculate these gradients. Originally, backpropagation was developed for simple multilayer perceptrons [74, 128, 129]. Backpropagation through time is based on the ability to represent any recurrent neural network as an equivalent feedforward network, unfolded in time, with the weights shared at the different time steps [126, 127, 130, 131, 132, 133, 134, 135]. The backpropagation technique of multilayer perceptrons can then be applied to the unfolded network.

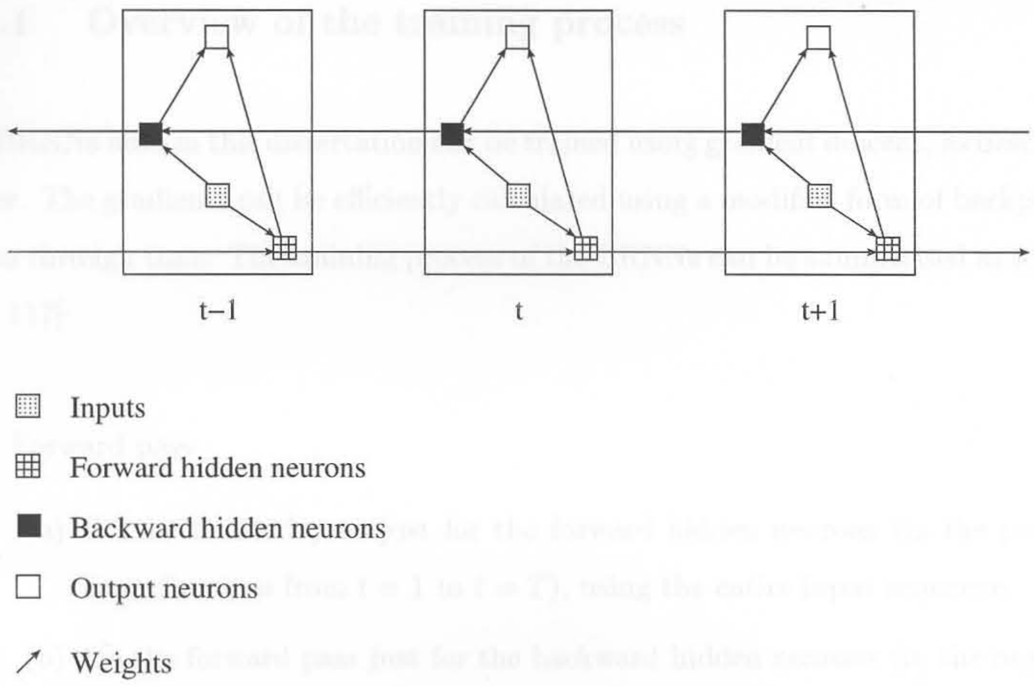

Figure A.1: Structure of the BRNN shown unfolded in time for three time steps.

Figure A.1 shows how a bi-directional recurrent neural network is unfolded for three time steps. In this figure, the arrows represent the groups of weights, and the squares the network nodes. Note that the outputs cannot be calculated until the entire sequence of forward and backward hidden node outputs have been calculated. This forward pass (consisting of three phases), and backpropagation, used to train the network, is discussed in the next section.

A.3 BRNN training equations

This section describes the training process used to train the bi-directional recurrent neural networks used in this dissertation. Since the training process is similar to that of conventional recurrent neural networks, and the fact that standard backpropagation through time needs only slight modification, the training equations will not be derived.

A.3.1 Overview of the training process

The BRNNs used in this dissertation can be trained using gradient descent, as described earlier. The gradients can be efficiently calculated using a modified form of backpropagation through time. The training process of the BRNNs can be summarised as follows [116, 117]:

1. Forward pass
   (a) Do the forward pass just for the forward hidden neurons (in the positive time direction from $t = 1$ to $t = T$), using the entire input sequence.
   (b) Do the forward pass just for the backward hidden neurons (in the negative time direction from $t = T$ to $t = 1$), using the entire input sequence.
   (c) Do the forward pass for the output neurons, using the two sequences of forward and backward hidden neuron outputs.
2. Backward pass
   (a) Do the backward pass for the output neurons, using the two sequences of output neuron outputs and target values, to obtain a sequence of delta values for each output neuron.
   (b) Do the backward pass only for the forward hidden neurons (in the negative time direction $t = T$ to $t = 1$), using the sequence of delta values of the outputs, to obtain a sequence of delta values for each forward hidden neuron.
   (c) Do the backward pass only for the backward hidden neurons (in the positive time direction $t = 1$ to $t = T$), using the sequence of delta values of the outputs, to obtain a sequence of delta values for each backward hidden neuron.
3. Calculate the gradients
4. Update the weights

A.3.2 Forward pass

Following the notation given in Chapter 2, the outputs of the forward hidden neurons can be calculated as

$$\begin{aligned} o_j^{f,t^+} = f_j^f(a_j^{f,t^+}) &= f_j^f\bigg(\sum_{i=1}^{d} w_{ji}^{IF} x_i^{t^+} + \sum_{k=1}^{m_f} w_{jk}^{FF} o_k^{f,t^+-1} + w_j^F\bigg), \\ & j = 1,2,\ldots,m_f; t^+ = 1,2,\ldots,T. \end{aligned} \tag{A.3}$$

The outputs of the backward neurons are calculated as

$$\begin{aligned} o_j^{b,t^-} = f_j^b(a_j^{b,t^-}) &= f_j^b\bigg(\sum_{i=1}^{d} w_{ji}^{IB} x_i^{t^-} + \sum_{k=1}^{m_b} w_{jk}^{BB} o_k^{b,t^-+1} + w_j^B\bigg), \\ & j = 1,2,\ldots,m_b; t^- = T,T-1,\ldots,1. \end{aligned} \tag{A.4}$$

The outputs of the neural network are calculated only after calculating the forward and backward neurons' outputs for the entire input sequence, $\boldsymbol{x}$, as

$$\begin{aligned} y_k^t = f_k(a_k^t) &= f_k\bigg(\sum_{j=1}^{m_f} w_{kj}^{FO} o_j^{f,t} + \sum_{i=1}^{m_b} w_{kj}^{BO} o_j^{b,t} + w_k^O\bigg), \\ & k = 1,2,\ldots,c; t = 1,2,\ldots,T, \end{aligned} \tag{A.5}$$

where $o_i^{f,t}$ is the output of the i'th forward neuron at time t, $o_i^{b,t}$ is the output of the i'th backward neuron at time t, m_f is the number of forward neurons, m_b is the number of backward neurons, and f_j^f, f_j^b and f_k are the neuron transfer functions for the forward hidden, backward hidden, and output layers, respectively. In the equations above, o_j^{f,t^+-1} denotes the 1 step delayed forward hidden node output, and o_j^{b,t^-+1} is the 1 step advanced backward hidden node output. The j'th forward hidden node output at time t and t^+ is given by $o_j^{f,t}$ and o_j^{f,t^+}, respectively. The j'th backward hidden node output at time t and t^- is given by $o_j^{b,t}$ and o_j^{b,t^-}, respectively.

A.3.3 Backward pass

In order to calculate the errors, also called delta values, the errors at output of the neural network are first calculated. These errors are dependent on both the transfer functions of the neurons, as well as the performance function used. In this dissertation, the cross-entropy error function is used, which is given as

$$E = \sum_{t=1}^{T} \sum_{k=1}^{c} t_k^t ln\left(\frac{y_k^t}{t_k^t}\right), \tag{A.6}$$

where t_k^t is the target (or desired) value of the k'th output neuron at time t. The errors at the outputs can then be calculated as

$$\delta_k^t \equiv \frac{\partial E^t}{\partial a_k^t} = y_k^t - t_k^t, \qquad k = 1, 2, \ldots, c; t = 1, 2, \ldots, T, \tag{A.7}$$

where δ_k^t is the error at the k'th output at time t. The deltas at the backward neurons is calculated next, by using

$$\begin{aligned} \delta_j^{b,t^+} &= g_j^b(a_j^{b,t^+})\left(\sum_{k=1}^{c} w_{kj}^{BO}\delta_k^{b,t^+} + \sum_{k=1}^{m_b} w_{jk}^{BB}\delta_k^{b,t^+-1}\right), \\ & \quad j = 1, 2, \ldots, m_b; t^+ = 1, 2, \ldots, T, \end{aligned} \tag{A.8}$$

where g_j^b is the derivative of the j'th backward hidden neuron transfer function. For the hyperbolic tangent transfer function given in Chapter 2, Equation (2.56), g_j is given as

$$g_j = \frac{\partial f_j}{\partial a_j} = 1 - o_j^2, \tag{A.9}$$

where o_j is the output activation of the j'th forward or backward hidden neuron. The deltas at the forward neurons can be calculated as

$$\delta_j^{f,t^-} = g_j^f(a_j^{f,t^-})\Big(\sum_{k=1}^{c} w_{kj}^{FO}\delta_k^{f,t^-} + \sum_{k=1}^{m_f} w_{jk}^{FF}\delta_k^{f,t^-+1}\Big),$$
$$j = 1,2,\ldots,m_f; t^- = T, T-1,\ldots,1. \qquad \text{(A.10)}$$

A.3.4 Gradient calculation

After the forward and backward passes, sequences of neuron outputs and delta values are available for the forward hidden neurons, backward hidden neurons, and output neurons. These can then be used to calculate the gradient of the error, with respect to the network weights. The error gradients can be given as

$$\frac{\partial E}{\partial w_{jk}^{FF}} = \delta_j^{f,t} o_k^{f,t-1}, \qquad j = 1,2,\ldots,m_f, k = 1,2,\ldots,m_f, \qquad \text{(A.11)}$$
$$\frac{\partial E}{\partial w_{ji}^{IF}} = \delta_j^{f,t} x_i^t, \qquad i = 1,2,\ldots,d, j = 1,2,\ldots,m_f, \qquad \text{(A.12)}$$
$$\frac{\partial E}{\partial w_j^{F}} = \delta_j^{f,t}, \qquad j = 1,2,\ldots,m_f, \qquad \text{(A.13)}$$
$$\frac{\partial E}{\partial w_{jk}^{BB}} = \delta_j^{b,t} o_k^{b,t+1}, \qquad j = 1,2,\ldots,m_b, k = 1,2,\ldots,m_b, \qquad \text{(A.14)}$$
$$\frac{\partial E}{\partial w_{ji}^{IB}} = \delta_j^{b,t} x_i^t, \qquad i = 1,2,\ldots,d, j = 1,2,\ldots,m_b, \qquad \text{(A.15)}$$
$$\frac{\partial E}{\partial w_j^{B}} = \delta_j^{b,t}, \qquad j = 1,2,\ldots,m_b, \qquad \text{(A.16)}$$
$$\frac{\partial E}{\partial w_{kj}^{FO}} = \delta_k^{t} o_j^{f,t}, \qquad j = 1,2,\ldots,m_f, k = 1,2,\ldots,c, \qquad \text{(A.17)}$$
$$\frac{\partial E}{\partial w_{kj}^{BO}} = \delta_k^{t} o_j^{b,t}, \qquad j = 1,2,\ldots,m_b, k = 1,2,\ldots,c, \qquad \text{(A.18)}$$
$$\frac{\partial E}{\partial w_k^{O}} = \delta_k^{t}, \qquad k = 1,2,\ldots,c. \qquad \text{(A.19)}$$

A.3.5 Update of weights

The weights of the neural network can now be updated, using the gradients of the previous section. The weight update is through the use of gradient descent. Each weight w in the network, is updated using the gradient $\frac{\partial E}{\partial w}$ as follows

$$w^t = w^{t-1} + \eta \nabla E = w^{t-1} + \eta \frac{\partial E}{\partial w}, \tag{A.20}$$

where w is one of the weights w_{ij}^{IF}, w_{ij}^{IB}, w_{jk}^{FF}, w_{kj}^{FO}, w_{kj}^{BO}, w_j^F, w_j^B, or w_k^O, and $\frac{\partial E}{\partial w}$ is one of the gradients given in the previous section, associated with w. The learning rate is denoted by η and is empirically set.

A.3.6 Generalisation capability

In order to make sure that the neural network has good generalisation, training must be stopped before overtraining occurs. In this dissertation, a simple approach to this difficult problem was taken. The training procedure can be summarised as:

1. Train the neural network on one sequence of speech vectors.
2. Evaluate the network on a small, independent, development test set and compute an average performance measure (such as the cross-entropy error).
3. Repeat steps 1 and 2 for a large number of iterations, saving the neural networks weights, and performance measure, at each iteration.
4. The neural network with the best generalisation performance is regarded as the one with the best performance on the development test set.

This training procedure is a simplification of a technique called cross-validation. The network finally chosen, is then evaluated on the full test set.

Bibliography

[1] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[2] B.L. Pellom and J.H.L. Hansen, "Automatic segmentation of speech recorded in unknown noisy channel characteristics," *Speech Communication*, vol. 25, pp. 97–116, 1998.

[3] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK book (for HTK Version 3.0)*, Cambridge University, United Kingdom, 2000.

[4] "The TIMIT database," National Institute of Standards and Technology (NIST), USA, Oct. 1990.

[5] J.W. Picone, "Signal modeling techniques in speech recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1214–1247, Sept. 1993.

[6] R. Vergin, D. O'Shaughnessy, and A. Farhat, "Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 5, pp. 525–532, Sept. 1999.

[7] G.S. Kang, "Quality improvement of LPC-processed noisy speech by using spectral subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 6, pp. 939–942, June 1989.

[8] O. Ghitza, "Auditory nerve representation criteria for speech analysis/synthesis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 6, pp. 736–740, June 1987.

[9] B.W. Juang, L.R. Rabiner, and J.G. Wilpon, "On the use of bandpass liftering in speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 7, pp. 947–954, July 1987.

[10] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proceedings of the IEEE*, vol. 73, no. 11, pp. 1551–1588, Nov. 1985.

[11] Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. COM-28, no. 1, pp. 84–95, Jan. 1980.

[12] G.S. Jovanović, "A new algorithm for speech fundamental frequency estimation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 3, pp. 626–630, June 1986.

[13] A. Vorstermans, J.-P. Martens, and B. Van Coile, "Automatic segmentation and labelling of multi-lingual speech data," *Speech Communication*, vol. 19, pp. 271–293, 1996.

[14] S. Pauws, Y. Kamp, and L. Willems, "A hierarchical method of automatic speech segmentation for synthesis applications," *Speech Communication*, vol. 19, pp. 207–220, 1996.

[15] A. Bonafonte, A. Nogueiras, and A. Rodriguez-Garrido, "Explicit segmentation of speech using Guassian models," in *Proceedings of ICSLP 96, Philadelphia, PA*, Oct. 1996.

[16] J.Ø. Olsen, "A two-stage procedure for phone based speaker verification," *Pattern Recognition Letters*, vol. 18, pp. 889–897, 1997.

[17] S.C. Lee, "Probabilistic segmentation for segment-based speech recognition," M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, May 1998.

[18] B.L. Pellom and J.H.L. Hansen, "Automatic segmentation of speech recorded in unknown noisy channel characteristics," Tech. Rep. RSPL-98-9, Robust Speech Processing Laboratory, Department of Electrical Engineering, Duke University, Box 90291, Durham, North Carolina 27708-0291, USA, 1998.

[19] C.-G. Jeong and H. Jeong, "Automatic phone segmentation and labeling of continuous speech," *Speech Communication*, vol. 20, pp. 291–311, 1996.

[20] S. Policker and A.B. Geva, "Nonstationary time series analysis by temporal clustering," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 30, no. 2, pp. 339–343, Apr. 2000.

[21] P. Cosi, "SLAM: Segmentation and labelling automatic module," in *Proceedings of EUROSPEECH 93, Berlin, Germany*, 1993.

[22] P. Cosi, "SLAM: a pc-based multi-level segmentation tool," in *Speech Recognition and Coding. New Advances and Trends*, A.J. Rubio Ayuso and J.M. Lopez, Eds., pp. 124–127. Springer-Verlag, 1995, NATO ASI Series, Series F: Computer and Systems Sciences, N. 147.

[23] P. Cosi, "Auditory modeling and neural networks," in *A Course on Speech Processing, Recognition, and Artificial Neural Networks*. Springer-Verlag, 1998, Lecture Notes in Computer Science.

[24] L.S. Smith, "Using an onset-based representation for sound segmentation," in *Proceedings of NEURAP 95, Marseilles, France*, Dec. 1995.

[25] L.S. Smith, "Onset-based sound segmentation," in *Advances in Neural Information Processing Systems*, D.S. Touretzky, M.C. Mozer, and M.E. Haselmo, Eds., vol. 8. MIT press, 1996.

[26] S. Chang, L. Shastri, and S. Greenberg, "Automatic phonetic transcription of spontaneous speech (American English)," in *Proceedings of ICSLP 2000, Beijing, China*, Oct. 2000.

[27] P. Regel, "A module for acoustic-phonetic transcription of fluently spoken german speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, no. 3, pp. 440–450, June 1982.

[28] T. Fukada, S. Aveline, M. Schuster, and Y. Sagisaka, "Segment boundary estimation using recurrent neural networks," in *Proceedings of the EUROSPEECH 97, Rhodos, Greece*, 1997, pp. 2839–2842.

[29] R. Andre-Obrecht, "A new statistical approach for the automatic segmentation of continuous speech signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 1, pp. 29–40, Jan. 1988.

[30] M. Basseville and I.V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[31] R. Èmejla and P. Sovka, "Speech segmentation using Bayesian autoregressive changepoint detector," *Radioengineering*, vol. 7, no. 4, pp. 14–17, Dec. 1998.

[32] B. Petek, O. Andersen, and P. Dalsgaard, "On the robust automatic segmentation of spontaneous speech," in *Proceedings of ICSLP 96, Philadelphia, PA*, Oct. 1996, vol. 2, pp. 913–916.

[33] C. Chan and K.W. Ng, "Separation of fricatives from aspirated plosives by means of temporal spectral variation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, no. 4, pp. 1130–1137, Oct. 1985.

[34] J.K. Ryeu and H.S. Chung, "Chaotic recurrent neural networks and their application to speech recognition," *Neurocomputing*, vol. 13, pp. 281–294, 1996.

[35] R. De Mori and P. Laface, "Use of fuzzy algorithms for phonetic and phonemic labeling of continuous speech," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 2, pp. 136–148, Mar. 1980.

[36] R.-C. Shyu, J.-F. Wang, and J.-Y. Lee, "Improvement in connected Mandarin digit recognition by explicitly modeling coarticulatory information," *Journal of Information Science and Engineering*, vol. 16, pp. 649–660, 2000.

[37] H.-J. Yu and Y.-H. Oh, "A neural network for 500 word vocabulary word spotting using non-uniform units," *Neural Networks*, vol. 13, pp. 681–688, 2000.

[38] J.-P. Hosom and R.A. Cole, "A diphone-based digit recognition system using neural networks," in *Proceedings of ICASSP 97, Munich, Germany*, 1997.

[39] H. Kaeslin, "A systematic approach to the extraction of diphone elements from natural speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 2, pp. 264–271, Apr. 1986.

[40] L. Shastri, S. Chang, and S. Greenberg, "Syllable detection and segmentation using temporal flow neural networks," in *Proceedings of the Fourteenth International Congress of Phonetic Sciences, San Francisco*, Aug. 1999.

[41] C.-T. Hsieh, M.-C. Su, E. Lai, and C.-H. Hsu, "A segmentation method for continuous speech utilizing hybrid neuro-fuzzy network," *Journal of Information Science and Engineering*, vol. 15, pp. 615–628, 1999.

[42] G.D. Cook and A.J. Robinson, "The 1997 ABBOT system for the transcription of broadcast news," in *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, Virginia*, Feb. 1998.

[43] L.S. Smith, "A neurally motivated technique for voicing detection and F_0 estimation for speech," Tech. Rep. CCCN-22, Centre for Cognitive and Computational Neuroscience, University of Stirling, Stirling FK9 4LA, Scotland, July 1996.

[44] R.J. Di Francesco, "Real-time speech segmentation using pitch and convexity jump models: Application to variable rate speech coding," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 5, pp. 741–748, May 1990.

[45] L.J. Siegel and A.C. Bessey, "Voiced/unvoiced/mixed excitation classification of speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, no. 3, pp. 451–460, June 1982.

[46] A.K. Krishnamurthy, "Two-channel speech analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 4, pp. 730–743, Aug. 1986.

[47] C.K. Gan and R.W. Donaldson, "Adaptive silence deletion for speech storage and voice mail applications," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 6, pp. 924–927, June 1988.

[48] H. Kobatake, "Optimization of voiced/unvoiced decisions in nonstationary noise environments," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 1, pp. 9–18, Jan. 1987.

[49] G. Bruno, M.D. Di Benedetto, M.G. Di Benedetto, A. Gilio, and P. Mandarini, "A Bayesian-adaptive decision method for the V/UV/S classification of segments of a speech signal," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 4, pp. 556–559, Apr. 1987.

[50] M. Lahat, R.J. Niederjohn, and D.A. Krubsack, "A spectral autocorrelation method for measurement of the fundamental frequency of noise-corrupted speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 6, pp. 741–750, June 1987.

[51] D.G. Childers, M. Hahn, and J.N. Larar, "Silent and voiced/unvoiced/mixed excitation (four-way) classification of speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1771–1774, Nov. 1989.

[52] R. Zelinski and F. Class, "A segmentation algorithm for connected word recognition based on estimation principles," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-31, no. 4, pp. 818–827, Aug. 1983.

[53] M.H. Christiansen and J. Allen, "Coping with variation in speech segmentation," in *Proceedings of GALA 1997: Language Acquisition: Knowledge Representation and Processing*, A. Sorace, C. Heycock, and R. Shillcock, Eds., 1997, pp. 327–332.

[54] M.H. Christiansen, J. Allen, and M.S. Seidenberg, "Learning to segment speech using multiple cues: A connectionist model," *Language and Cognitive Processes*, vol. 13, no. 2, pp. 221–268, 1998.

[55] M.A. Siegler, U. Jain, B. Raj, and R.M. Stern, "Automatic segmentation, classification and clustering of broadcast news audio," in *Proceedings of the DARPA Speech Recognition workshop, Chantilly, VA*. Feb. 1997, pp. 97–99, Morgan Kaufmann.

[56] E. Shriberg, A. Stolcke, D. Hakkani-Tür, and G. Tür, "Prosody-based automatic segmentation of speech into sentences and topics," *Speech Communication*, vol. 32, pp. 127–154, 2000.

[57] A. Lavie, D. Gates, N. Coccaro, and L. Levin, "Input segmentation of spontaneous speech in janUS: a speech-to-speech translation system," in *Dialogue Processing in Spoken Language Systems*, E. Maier, M. Mast, and S. LuperFoy, Eds., pp. 86–99. Springer-Verlag, 1997.

[58] K. Ries, "HMM and neural network based speech act detection," in *Proceedings of ICASSP 99, Phoenix, Arizona, USA*, Mar. 1999.

[59] M. Swerts and M. Ostendorf, "Prosodic and lexical indications of discourse structure in human-machine interactions," *Speech Communication*, vol. 22, pp. 25–41, 1997.

[60] G. Tzanetakis and P. Cook, "A framework for audio analysis based on classification and temporal segmentation," in *Proceedings of EUROMICRO 99*, June 1999.

[61] G. Tzanetakis and P. Cook, "Multifeature audio segmentation for browsing and annotation," in *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, New York*, Oct. 1999.

[62] A. Batliner, R. Kompe, A. Kießling, M. Mast, H. Niemann, and E. Nöth, "M = syntax + prosody: A syntactic-prosodic labelling scheme for large spontaneous speech databases," *Speech Communication*, vol. 25, pp. 193–222, 1998.

[63] S. Renals, D. Abberley, D. Kirby, and T. Robinson, "Indexing and retrieval of broadcast news," *Speech Communication*, vol. 32, pp. 5–20, 2000.

[64] P. Delacourt and C.J. Wellekens, "DISTBIC: A speaker-based segmentation for audio data indexing," *Speech Communication*, vol. 32, pp. 111–126, 2000.

[65] K. Bush, A. Ganapathiraju, P. Kornman, J. Trimble III, and L. Webster, "A comparison of energy-based endpoint detectors for speech signal processing," in *Proceedings of the IEEE Southeastcon, Tampa, Florida, USA*, Apr. 1996.

[66] L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[67] L.R. Rabiner and B.H. Juang, "An introduction to hidden Markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 4–16, Jan. 1986.

[68] J. Deller, J.H.L. Hansen, and J. Proakis, *Discrete-Time Processing of Speech Signals*, IEEE Press, New York, 2000.

[69] C.M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

[70] S. Haykin, *Neural networks: a comprehensive foundation*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2nd edition, 1999.

[71] W. Maass and C.M. Bishop, Eds., *Pulsed Neural Networks*, The MIT Press, Cambridge, Massachussets, 1999.

[72] D. Morgan, *Neural Network based Speech Processing*, Kluwer Academic Publishers, Norwell, Massachussets, 1991.

[73] R.P. Lippmann, "An introduction to computing with neural nets," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 4, no. 2, pp. 4–22, Apr. 1987.

[74] B. Widrow and M.A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, Sept. 1990.

[75] R. Rojas, *Neural Networks: A Systematic Introduction*, Springer-Verlag, Berlin, Germany, 1996.

[76] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the theory of neural computation*, A lecture notes volume in the Santa Fe Institute studies in the sciences of complexity. Addison-Wesley Publishing Company, Redwood City, California, 1991.

[77] A. Carling, *Introducing Neural Networks*, Sigma Press, Wilmslow, United Kingdom, 1992.

[78] R.J. Mammone and Y.Y. Zeevi, *Neural networks: theory and applications*, Academic Press, Inc., San Diego, California, 1991.

[79] I. Pitas, Ed., *Parallel algorithms: for digital image processing, computer vision, and neural networks*, John Wiley and Sons, Inc., New York, 1993.

[80] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1990.

[81] L.A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[82] L.A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 1, pp. 28–44, Jan. 1973.

[83] C. Von Altrock, *Fuzzy logic and neurofuzzy applications explained*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1995.

[84] S.J. Russel and P. Norvig, *Artificial intelligence: a modern approach*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1995.

[85] J.H. Mathews, *Numerical methods for mathematics, science, and engineering*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1990.

[86] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C - The Art of Scientific Computing*, Cambridge University Press, New York, 2nd edition, 1992.

[87] W.T. Vetterling, S.A. Teukolsky, W.H. Press, and B.P. Flannery, *Numerical Recipes - Example Book (C)*, Cambridge University Press, New York, 2nd edition, 1992.

[88] I. Miller and M. Miller, *John E. Freund's Mathematical Statistics*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 6th edition, 1999.

[89] M.R. Spiegel, *Theory and problems of probability and statistics*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, SI (metric) edition, 1999.

[90] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[91] L.R. Rabiner and S.E. Levinson, "A speaker-independent, syntax-directed, connected word recognition system based on hidden Markov models and level building," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, no. 3, pp. 561–573, June 1985.

[92] F. Jelinek, "Continuous speech recognition by statistical methods," *Proceedings of the IEEE*, vol. 64, no. 4, pp. 532–556, Apr. 1976.

[93] B.H. Juang and L.R. Rabiner, "Mixture autoregressive hidden Markov models for speech signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, no. 6, pp. 1404–1413, Dec. 1985.

[94] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1641–1648, Nov. 1989.

[95] C.-H. Lee and L.R. Rabiner, "A frame-synchronous network search algorithm for connected word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1649–1658, Nov. 1989.

[96] L.R. Bahl, P.F. Brown, P.V. De Souza, R.L. Mercer, and M.A. Picheny, "A method for the construction of acoustic Markov models for words," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 4, pp. 443–452, Oct. 1993.

[97] H. Ney, "The use of a one-stage dynamic programming algorithm for connected word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 2, pp. 263–271, Apr. 1984.

[98] C.S. Myers and S.E. Levinson, "Speaker independent connected word recognition using a syntax-directed dynamic programming procedure," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, no. 4, pp. 561–565, Aug. 1982.

[99] F. Charot, P. Frison, and P. Quinton, "Systolic architectures for connected speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 4, pp. 765–779, Aug. 1986.

[100] M.A. Bush and G.E. Kopec, "Network-based connected digit recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 10, pp. 1401–1413, Oct. 1987.

[101] A.E. Rosenberg, L.R. Rabiner, J.G. Wilpon, and D. Kahn, "Demisyllable-based isolated word recognition system," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-31, no. 3, pp. 713–726, June 1983.

[102] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans-*

[111] J.L. Elman, "Generalization, simple recurrent networks, and the emergence of structure," in *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*, M.A. Gernsbacher and S. Derry, Eds. Lawrence Erlbaum Associates, Mahwah, NJ, 1998.

[112] J.L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[113] J.E.W. Holm and E.C. Botha, "Leap-frog is a robust algorithm for training neural networks," *Network: Computation in Neural Systems*, vol. 10, pp. 1–13, 1999.

[114] T. Burrows and M. Niranjan, "The use of recurrent neural networks for classification," in *IEEE Workshop on Neural Networks for Signal Processing IV*, Piscataway, New Jersey, USA, 1994, pp. 117–125.

[115] T. Robinson, "Several improvements to a recurrent error propagation network phone recognition system," Tech. Rep. CUED/F-INFENG/TR82, Cambridge University, United Kingdom, Sept. 1991.

[116] M. Schuster, "Learning out of time series with an extended recurrent neural network," in *Proceedings of the IEEE Neural Network Workshop for Signal Processing, Kyoto, Japan*, 1996, pp. 170–179.

[117] M. Schuster and K.K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[118] M. Schuster, "Neural networks for speech processing," in *Encyclopedia of Electrical and Electronic Engineering*. John Wiley and Sons, 1999.

[119] Y. Bengio, *Neural Networks for Speech and Sequence Recognition*, International Thomson Computer Press, London, 1996.

[120] H.A. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*, Kluwer Academic Publishers, Norwell, Massachussets, 1994.

[121] M. Schuster, "Acoustic model building based on non-uniform segments and bi-directional recurrent neural networks," in *Proceedings of ICASSP 97, Munich, Germany*, 1997, pp. 3249–3252.

[122] S. Haykin, *Communication Systems*, John Wiley and Sons, Inc., New York, third edition, 1994.

[123] F.H. Liu, *Environmental Adaption for Robust Speech Recognition*, Ph.D. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, June 1994.

[124] U. Jain, "Connected digit recognition over long distance telephone lines using the SPHINX-II system," M.S. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1996.

[125] S. Young and N. Russell, "Token passing: a simple conceptual model for connected speech recognition systems," Tech. Rep. 38, Cambridge University Engineering Department, United Kingdom, 1989.

[126] L.A. Feldkamp and G.V. Puskorius, "A signal processing framework based on dynamic neural networks with application to problems in adaption, filtering, and classification," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2259–2277, Nov. 1998.

[127] G.V. Puskorius, L.A. Feldkamp, L.I. Davis, and Jr., "Dynamic neural network methods applied to on-vehicle idle speed control," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1407–1420, Oct. 1996.

[128] E. Barnard, "Optimization for training neural nets," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 232–240, Mar. 1992.

[129] E. Barnard and E.C. Botha, "Back-propagation uses prior information efficiently," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 794–802, Sept. 1993.

[130] O. Nerrand, P. Roussel-Ragot, D. Urbani, L. Personnaz, and G. Dreyfus, "Training recurrent neural networks: why and how? an illustration in dynamicall

process modeling," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 178–183, Mar. 1994.

[131] O. Olurotimi, "Recurrent neural network training with feedforward complexity," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 185–196, Mar. 1994.

[132] B.A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1212–1228, Sept. 1995.

[133] P.J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[134] P.J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, pp. 339–356, 1988.

[135] R.J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Computation*, vol. 2, pp. 490–501, 1990.