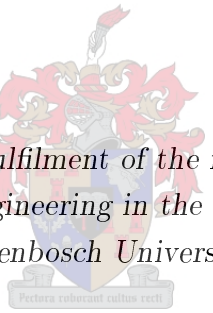
THE AUTOMATIC AND UNCONSTRAINED SEGMENTATION OF SPEECH INTO SUBWORD UNITS

by

Van Zyl van Vuuren



Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering in the Faculty of Engineering at Stellenbosch University

Department of Electrical and Electronic Engineering,
Stellenbosch University.

Supervisor: Prof. T.R. Niesler

April 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2014 .....

Abstract

We develop and evaluate several algorithms that segment a speech signal into subword units without using phone or orthographic transcripts. These segmentation algorithms rely on a scoring function, termed the local score, that is applied at the feature level and indicates where the characteristics of the audio signal change. The predominant approach in the literature to segmentation is to apply a threshold to the local score, and local maxima (peaks) that are above the threshold result in the hypothesis of a segment boundary. Scoring mechanisms of a select number of such algorithms are investigated, and it is found that these local scores frequently exhibit clusters of peaks near phoneme transitions that cause spurious segment boundaries. As a consequence, very short segments are sometimes postulated by the algorithms. To counteract this, ad-hoc remedies are proposed in the literature. We propose a dynamic programming (DP) framework for speech segmentation that employs a probabilistic segment length model in conjunction with the local scores. DP offers an elegant way to deal with peak clusters by choosing only the most probable segment length and local score combinations as boundary positions. It is shown to offer a clear performance improvement over selected methods from the literature serving as benchmarks.

Multilayer perceptrons (MLPs) can be trained to generate local scores by using groups of feature vectors centred around phoneme boundaries and midway between phoneme boundaries in suitable training data. The MLPs are trained to produce a high output value at a boundary, and a low value at continuity. It was found that the more accurate local scores generated by the MLP, which rarely exhibit clusters of peaks, made the additional application of DP less effective than before. However, a hybrid approach in which DP is used only to resolve smaller, more ambiguous peaks in the local score was found to offer a substantial improvement on all prior methods.

Finally, restricted Boltzmann machines (RBMs) were applied as features detectors. This provided a means of building multi-layer networks that are capable of detecting highly abstract features. It is found that when local score are estimated by such deep networks, additional performance gains are achieved.

Opsomming

Ons ontwikkel en evalueer verskeie algoritmes wat ‘n spraaksein in sub-woord eenhede segmenteer sonder om gebruik te maak van ortografiese of fonetiese transkripsies. Dié algoritmes maak gebruik van ‘n funksie, genaamd die lokale tellingsfunksie, wat ‘n waarde produseer omtrent die lokale verandering in ‘n spraaksein. In die literatuur is daar gevind dat die hoofbenadering tot segmentasie gebaseer is op ‘n grenswaarde, waarbo alle lokale maksima (pieke) in die lokale telling lei tot ‘n skeiding tussen segmente. ‘n Selektiewe groep segmentasie algoritmes is ondersoek en dit is gevind dat lokale tellings geneig is om groeperings van pieke te hê naby aan die skeidings tussen foneme. As gevolg hiervan, word baie kort segmente geselekteer deur die algoritmes. Om dit teen te werk, word ad-hoc metodes voorgestel in die literatuur. Ons stel ‘n alternatiewe metode voor wat gebaseer is op dinamiese programmering (DP), wat ‘n statistiese verspreiding van lengtes van segmente inkorporeer by segmentasie. DP bied ‘n elegante manier om groeperings van pieke te hanteer, deurdat net kombinasies van hoë lokale tellings en segmentwaarskynlikheid, met betrekking tot die lengte van die segment, tot ‘n skeiding lei. Daar word gewys dat DP ‘n duidelike verbetering in segmentasie akkuraatheid toon bo ‘n paar gekose algoritmes uit die literatuur.

Meervoudige lae perseptrone (MLPe) kan opgelei word om ‘n lokale telling te genereer deur gebruik te maak van groepe eienskapsvektore gesentreerd rondom en tussen foneem skeidings in geskikte opleidingsdata. Die MLPe word opgelei om ‘n groot waarde te genereer as ‘n foneem skeiding voorkom en ‘n klein waarde andersins. Dit is gevind dat die meer akkurate lokale tellings wat deur die MLPe gegenereer word minder groeperings van pieke het, wat dan die addisionele toepassing van die DP minder effektief maak. ‘n Hibriede toepassing, waar DP net tussen kleiner en minder duidelike pieke in die lokale telling kies, lei egter tot ‘n groot verbetering bo-op alle vorige metodes.

As ‘n finale stap het ons beperkte Boltzmann masjiene (BBMe) gebruik om patrone in data te identifiseer. Sodoende, verskaf BBMe ‘n manier om meervoudige lae netwerke op te bou waar die boonste lae baie komplekse patrone in die data identifiseer. Die toepassing van dié dieper netwerke tot die generasie van ‘n lokale telling het tot verdere verbeteringe in segmentasie-akkuraatheid gelei.

Acknowledgements

I would like to express my sincere gratitude to the following people who have contributed to making this work possible:

- My supervisor, Prof. Thomas Niesler, for all his support and encouragement, which enabled me to do my best in this work.

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are those of the author and are not necessarily to be attributed to the NRF.

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

F_j	The jth frame of the speech signal
f	Frame or frames just to the left of F_j
g	Frame or frames just to the right of F_j
S_j	State of a Markov chain corresponding to the jth frame
a_{ij}	Transition probability from state S_i to state S_j
b_j	Emission probability at state S_j
$P(A)$	Probability of event A
B_R	Reference boundary
B_H	Hypothesised boundary
DP cost	The time alignment between two boundary sequences
INS	Percentage insertions with respect to the reference boundaries
DEL	Percentage deletions with respect to the reference boundaries
ERR	Average INS and DEL
C	Cosine distance
E	Euclidean distance
NCB	Normalised city block distance
ϵ	Element of, or learning rate
$\mathbf{h}$	Hidden nodes of RBM
$\mathbf{v}$	Visible nodes of RBM
w_{ij}	Weight associated with visible node i and hidden node j

q	Clique
Q_q	Maximal clique
G	Undirected graph
$\mathbf{X}$	Set of random variables
$p(\mathbf{X})$	Joint probability distribution of $\mathbf{X}$
$\phi_q()$	Potential function
Z	Partition function
$E(.)$	Energy function
$\langle\rangle$	Expectation
Δ	Derivative or rate of change
σ_i	Standard deviation of the ith visible node of a GBRBM
$\mathcal{N}(.\|\mu, \sigma^2)$	A Gaussian distribution with mean μ and variance σ^2
$min(.,.,.)$	Function that returns the minimum of three values

Acronyms and Abbreviations

ASR	Automatic Speech Recognition
TTS	Text-To-Speech
HMM	Hidden Markov Model
DTW	Dynamic Time Warping
ANN	Artificial Neural Network
NN	Neural Network
MLP	Multilayer Perceptrons
DP	Dynamic Programming
RBM	Restricted Boltzmann Machine
LS	Local Score
FFT	Fast Fourier Transform
MMC	Maximum Margin Clustering
RBF	Radial Basis Function
MFCC	Mel-Frequency Cepstral Coefficient
HR	Hit Rate
OS	Over-Segmentation
TP	Target Point
AI	Artificial Intelligence
DBN	Deep Belief Network
MRF	Markov Random Field
CD	Contrastive Divergence
MCMC	Markov Chain Monte Carlo
GBRBM	Gaussian-Bernoulli RBM

Chapter 1

Introduction

This thesis develops and evaluates several algorithms that segment a speech signal into subword units without any additional prior knowledge of the signal. The following sections describe the motivation and the extent of this work.

1.1 Motivation for research

Automatic speech recognition (ASR) systems are being used more and more frequently in real-world applications. Although the accuracies of ASR systems have improved, they are only accessible to the people who can speak the language and correctly pronounce the dialect on which the system is trained. This research forms part of a project that aims to significantly reduce the time it takes to develop pronunciation dictionaries for ASR systems, and thereby make speech technologies more widely available for a greater diversity of languages.

The task of segmenting a speech signal into phonemes or phoneme-like units, as shown in Figure 1.1, plays an important role in the speech processing field. Although accurate manual segmentation can be achieved by trained phoneticians, the task is extremely time consuming, expensive and subjective. In view of these obstacles, automatic segmentation algorithms are frequently used to find coarse preliminary phonetic boundaries that can subsequently be refined by phoneticians. This approach expedites the development of a pronunciation dictionary and can also be used to obtain bootstrapping acoustic training data, thereby reducing the time it takes to develop a high quality ASR system. In an under-resourced setting, in which very little transcribed phonetic material is available, this can be extremely beneficial. It would be even more beneficial if the development of pronunciation dictionaries could be

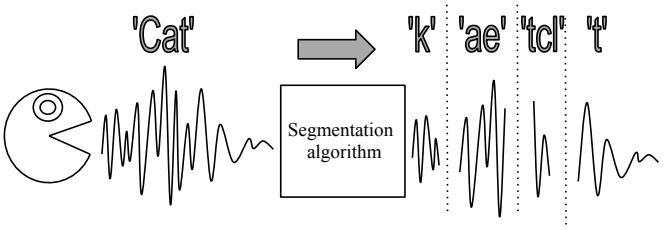


Figure 1.1: The segmentation of the word 'cat' into its phones.

fully automated. The main motivation behind the research presented in this thesis is to develop a high quality segmentation algorithm which can serve not only to facilitate the process of creating a pronunciation dictionary, but also as the first step in a bottom-up process to automatically construct a pronunciation dictionary from the audio and the orthographic transcripts [1].

Reliable automatic segmentation algorithms are also useful in technologies outside ASR, such as the study of pronunciation variation, the development of coherent large-scale dictionaries, text-to-speech (TTS) applications [4], and many others [5; 6; 7].

1.2 Scope

A distinction can be made between segmentation approaches that require phone or orthographic transcripts, and those that do not. These two approaches are often referred to as constrained and unconstrained, respectively [3]. There is also a distinction between those algorithms that segment speech into syllables, and those that segment into phoneme-like units.

Constrained approaches usually perform a forced alignment between phoneme-based hidden Markov models (HMMs) and a phonetic transcription [3; 4; 8], or align phoneme templates to a signal using dynamic time warping (DTW) [3; 4; 9]. Unconstrained approaches, on the other hand, typically rely on a scoring function that is applied at the feature level and can be used to infer segment boundaries. Because these scores are calculated from features grouped locally in time, we will refer to them as 'local scores'. Popular local scores are vector distance functions which respond to the dynamics of the features and from which a peak-picking algorithm finds viable local maxima at which to hypothesise segment boundaries [6; 10; 11; 12; 13]. Rule-based approaches that use language-specific knowledge to calculate a local score independent of the phone string [7; 14], and HMM phone loop segmentation also fall into the unconstrained class [3].

Artificial neural networks (ANNs) have been applied to both constrained and unconstrained segmentation. The constrained approaches are mostly based on hybrid HMM/ANN algorithms in which multilayer perceptrons (MLPs) act either as phone probability estimators [15; 16], or are used to detect phoneme transitions in order to refine the boundaries produced by a HMM alignment [17; 18]. For unconstrained segmentation, ANNs have recently been shown to be highly effective at producing very accurate local scores [3; 19].

In this thesis the focus will be on the unconstrained segmentation of speech into phoneme-like units in accordance with the ultimate goal of forming part of a system that can automatically generate a pronunciation dictionary. Furthermore, the focus will mainly be on unconstrained algorithms that employ local scores. After the scoring mechanisms of a number of algorithms satisfying this criteria have been investigated and some of the drawbacks associated with the established approaches have been discussed, a dynamic programming (DP) framework for speech segmentation will be introduced and shown to improve performance. This DP framework employs a probabilistic segment length model in conjunction with the local scores to make more insightful segmentations. Evaluations of this DP approach have been presented as papers at the PRASA 2012 [20] and INTERSPEECH 2013 [21] conferences, where the former focused on vector distance functions and the latter on ANNs. These papers are included in Appendices B and C, respectively.

It has recently been shown that restricted Boltzmann machines (RBMs) can be used as feature detectors. This provides a powerful means of building multi-layer networks that are capable of detecting abstract features at the higher layers. Networks that are produced in this manner can be used for classification by adding a layer of neurons, corresponding to the labels to the network, and training the network to recognise the labels through conventional backwards propagation of the error derivative. Promising results for image and phone recognition have been achieved by using neural networks (NNs) that are pre-trained by RBMs. In an attempt to increase the accuracy of the local scores produced by MLPs, the effects of pre-training with RBMs will be investigated.

1.3 Overview

Chapter 2 will study some noteworthy unconstrained speech segmentation algorithms described in the literature. The discussion will proceed by considering various local score functions, and studying how these local scores are used in segment boundary detection. This will lead to an investigation on the variety of algorithms that detect segment boundaries at points of maximum local acoustic change, algorithms that search through an utterance to find the sequence of segments with the least acoustic variation within segments, and finally, algorithms that use ANNs to discriminate between features that provide evidence of a segment boundary and those that do not. Chapter 3 follows with an introduction to the TIMIT corpus used for training and testing. Chapter 4 gives an in-depth explanation of the proposed DP-based algorithm. Here we consider the mechanisms behind the algorithm, as well as how to determine the segment length distribution from TIMIT. Chapter 5 describes several ways in which the accuracy of the automatically produced segmentations can be assessed. The experimental setup used for blind speech segmentation is discussed in Chapter 6, followed by the corresponding experimental results in Chapter 7. In Chapter 8 we consider the experimental results produced by the MLP-based segmentation algorithms. We discuss deep neural networks in Chapter 9. This starts with a background discussion and introduction and is followed by a more detailed explanation of RBMs, and concludes with a discussion of deep belief networks. The experimental evaluation of segmentation algorithms based on deep NNs that were pre-trained by RBMs are considered in Chapter 10. Finally, Chapter 11 concludes the thesis.

Chapter 2

Background on segmentation algorithms

Many segmentation algorithms are based on the assumption that there are regions in speech, termed speech segments, where the acoustic features remain relatively constant, and that there are clear transitions (boundaries) between these regions. To detect these transitions, the algorithms employ some estimate of the local acoustic change in the signal. 'Local' in this context refers to temporal acoustic changes taking place at a specific time independent of any previous or future acoustic changes within the signal. A function that quantifies these local acoustic changes will be referred to as the **local score** function in the remainder of this thesis. The local score function is central to many segmentation methods. Therefore, different types of local score functions and their applications in the recent literature will first be reviewed.

2.1 Algorithms employing blind local scores

The most common approach used in unconstrained speech segmentation is to hypothesise segment boundaries at the times at which local acoustic change is at a maximum. Segmentation algorithms following this approach mainly employ vector distance functions, applied to consecutive feature vectors, as local scores. Maximal acoustic changes will then correspond to the peaks, or valleys of the local scores, depending on the distance function. Local scores of this type are referred to as 'blind', because they are ignorant of the characteristics of the signal that is being segmented, including the language [10]. An advantage of this approach is that no language bias will exist, as would be present when using trained models. This should in principle make these algorithms applicable to different languages without an appreciable reduction in performance.

As a result of small acoustic changes taking place throughout an utterance, blind local scores will sometimes contain many small peaks, making them vulnerable to over-segmentation. Over-segmentation is the term used to describe the hypothesis of excessively many segment boundaries. One way to reduce such over-segmentation is to ignore local maxima that are smaller than a chosen threshold. The value of the threshold should be chosen so that large acoustic changes associated with true boundaries are detected, while spurious changes are ignored. This simple approach has been found to be prevalent in blind segmentation algorithms.

A selection of blind segmentation algorithms are reviewed in the following. They were specifically chosen to illustrate a diversity of these local score functions, of which a selection will later be compared in our own experiments. The local score will henceforth be denoted by LS in equations.

2.1.1 Algorithms that use a threshold

2.1.1.1 Räsänen et al. [10]

The local score function used in this algorithm is the cross correlation (also called the normalised dot product) between two FFT magnitude vectors. This is shown in Equation 2.1.1, where f and g represent the FFT magnitude vectors for the frames to the left and to the right, respectively, of the investigated frame, F_j:

$$LS(F_j) = \frac{f.g}{\parallel f \parallel \parallel g \parallel}. \tag{2.1.1}$$

Feature vectors that are similar will give a score close to 1, and dissimilar vectors will give a score closer to 0. The algorithm applies a non-linear filter to the cross-correlation sequence in order to quantify the degree of uniformity in the region preceding and following the point of interest. In a similar way, the dissimilarity between these regions is also determined. The difference between the dissimilarity and uniformity values leads to a signal of which the valleys correspond to probable segment boundaries. However, this signal is very noisy, and there are many small valleys. The number of these smaller valleys is reduced by application of a 'min-max' filter, which searches a fixed region (n_{mm}) around the point of interest to find the local maximum and minimum values. The difference between this maximum and minimum serves as the output of the filter at the position of the minimum. This filter is applied throughout the signal in non-overlapping regions. The filter output is a signal of which the peaks represent possible boundaries. Given that the 'minmax' filter region is usually very small and applied in non-overlapping intervals, many closely spaced peaks may remain. Temporal peak masking is therefore applied in a subsequent step. Two peaks falling within a determined interval (t_d) of each other and which are above a chosen threshold (p_{min}) are identified, and the highest peak retained. The location of the highest peak is also shifted a small distance toward the eliminated smaller peak in proportion to their amplitudes. Finally the algorithm takes regions of silence into account by removing boundaries at frames when the average energy content from 8ms before to 30ms after the frame in question fall below a certain multiple of the minimum energy for the signal.

2.1.1.2 Ten Bosch et al.[6]

This work uses the angle between the feature vectors just before and just after the point of interest to quantify the degree of local change. This is given by Equation 2.1.2, where f and g are the averages of the two feature vectors before and after the frame of interest F_j, respectively:

$$LS(F_j) = \arccos \frac{f.g}{(\parallel f \parallel \parallel g \parallel)^{\frac{1}{2}}}. \tag{2.1.2}$$

The technique uses 12 MFCC coefficients and log energy together with their first and second derivatives, resulting in 39-dimensional feature vectors. Furthermore, the local score is scaled by the log frame

energy to attenuate points of low energy. All local maxima above a threshold (δ) are hypothesised as boundaries. This approach was also considered in [1], where it was found that it was prone to over-segmentation caused by a noisy local score. To reduce the number of closely spaced peaks, the author proposed smoothing the local score by taking the average in a moving Hanning window.

2.1.1.3 Estevan et al.[12]

This algorithm employs maximum margin clustering (MMC) to detect points of change in a feature vector consisting of 12 MFCC coefficients, log energy and their first and second derivatives. A sliding window, N frames wide and centered about the frame of interest, sweeps through the signal. MMC (using an RBF kernel) is applied to the frames within this window. The width of the RBF kernel, W, is estimated from a development set. The MMC results in a cluster label for each frame within the window, and changes in these labels indicate possible segment boundaries. It was found that the best way to detect these changes is by using the Euclidean distance, as given by Equation 2.1.3, between the cluster labels and the cluster means. Let f be the cluster label of each frame within the sliding window, and g be the mean of the cluster throughout the signal. Peaks in the Euclidean distance above a threshold will then indicate the segment boundaries.

$$LS(F_j) = [\sum_{l=1}^{T}(f_l - g_l)^2]^{\frac{1}{2}} \tag{2.1.3}$$

2.1.1.4 Sarkar et al.[11]

The method proposed by these authors differs from the previous three by operating in the time domain rather than the frequency domain. The local score function used in this case is the average level crossing rate. The level crossing rate is closely related to the zero crossing rate, but with multiple additional levels other than $y = 0$, and among which the average crossing rate is taken. The levels can be distributed uniformly or non-uniformly. For this choice of local score, a boundary corresponds to a valley rather than a peak. As for some of the preceding algorithms, a threshold is required to suppress shallow valleys which would otherwise lead to over-segmentation.

2.1.2 Other approaches

Apart from the algorithms that have been described so far, the use of a threshold as a means of reducing over-segmentation appears to be persistent in literature [13; 22]. Many threshold-based blind segmentation approaches have been formulated, each one introducing some novelty in their methodology. In the end, however, they all rely on a threshold to reduce over-segmentation. Therefore, to avoid redundancy, we will conclude the discussion on threshold-based blind segmentation algorithms.

2.1.3 Algorithms that seek out uniform acoustic segments

Another approach to blind speech segmentation is to use a local score to increase the acoustic uniformity within segments. The uniformity of a segment is increased by minimising the acoustic distortion within

the segment. The work done by Sharma et al. [5] calculates the acoustic variation of a signal by using the Euclidean distance applied to MFCC features as the local score. The distortion within a segment ($D_{i,n}$) is then calculated by Equation 2.1.5, which uses the local score at frame j, given by Equation 2.1.3, and the mean of the local score from frame i to n, given by Equation 2.1.4. The segment stretching from frame i to frame n is denoted by $S_{i,n}$.

$$M_{i,n} = \frac{1}{n-i+1}\sum_{j=i}^{n} LS_j \tag{2.1.4}$$

$$D_{i,n} = \sum_{j=i}^{n}(LS_j - M_{i,n})^2 \tag{2.1.5}$$

The overall distortion of the speech signal is a cumulative sum of the distortions of all the segments. The overall distortion is minimised by applying a level-based DP algorithm to search for the optimal segmentation, assuming that the number of levels (segments) in the signal is known.

2.2 Algorithms that detect patterns of change

An inherent limitation of blind segmentation is that the vector distance functions have no regard for the pattern in which the features change over time. It is possible that some phoneme boundaries are characterised by a pattern in the features, and cannot be found by simply looking at the points of maximal acoustic change. Some phoneme boundaries may therefore go undetected. One solution to this problem is to train an MLP to estimate a local score from a group of consecutive frames. In this way the pattern of change between the frames will be taken into account, and if a pattern is recognised by the MLP, a high local score will result.

2.2.1 Using MLPs to compute local scores

An MLP can be employed to compute a local score on the basis of a group of consecutive feature vectors. In recent work this was achieved by training two output neurons, one outputs a high value when the evidence in the input feature vectors supports the presence of a boundary, and the other when the evidence supports the absence of a boundary [3]. The training data consists of feature vector groups located around phoneme boundaries in TIMIT and feature vector groups midway between two boundaries. The local score is obtained by taking the difference between the two outputs. Segmentation approaches that rely on the detection of peaks in the local scores, such as those in Section 2.1, may now be employed to find possible segment boundaries.

2.2.1.1 Keri et al. [3]

The authors of [3] proposed the use of an MLP with 30 hyperbolic tangent neurons in the hidden layer, and two hyperbolic tangent neurons in the output layer. The output neurons are trained to output 1 and -1 respectively for a boundary, and -1 and 1 otherwise. The difference between the two will then give a local score lying between 2 and -2. A frame length of 10 ms with a frame skip of 5 ms

between frames was used to calculate the feature vectors. Groups of 11 consecutive feature vectors centred about the point of interest were used with 12 MFCCs and log energy as features. The network was trained by backpropagation, and functions by detecting regions in time where the local score is larger than a threshold throughout the region. The authors proposed the use of 0 as the threshold, but different values can also be effective as shown in [19]. A segment boundary is then hypothesised at the frame at which the local score is at a maximum within the region as demonstrated by

$$[\hat{B_R}] = \underset{t\epsilon\{S_R...E_R\}}{\text{argmax}} \{LS(i_t)\}, \tag{2.2.1}$$

where B_R is the boundary frame in the region, S_R and E_R are the start and end of the region respectively, LS is the local score, and i_t are the frames between S_R and E_R [3].

2.3 The drawbacks of using a threshold

In the previous sections the concept of a local score was introduced. It was shown that segment boundaries are hypothesised at the local maxima of these local scores. Although there are different ways to formulate a local score, the predominant technique used to reduce over-segmentation is to impose a threshold to the local score. All local maxima that are smaller than this threshold are then ignored. In this section we will consider some of the drawbacks of this approach.

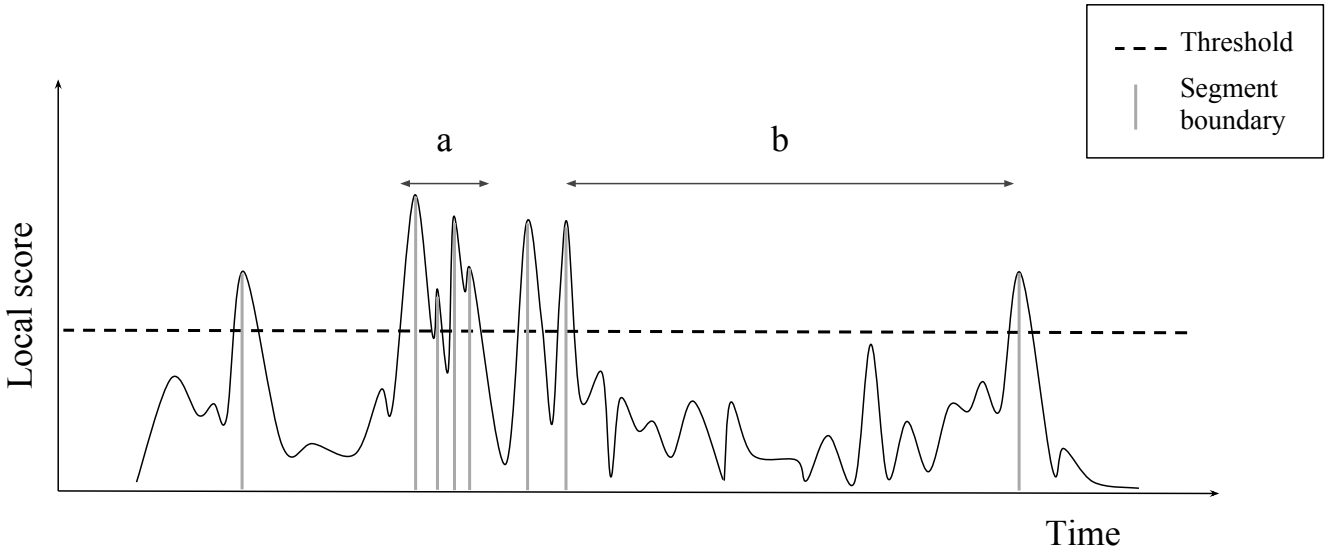


Figure 2.1: An illustration of segmenting an utterance based on imposing a threshold to the local score.

Figure 2.1 illustrates the segmentation of a typical utterance. The local score of the utterance, the segment boundaries, as well as the threshold are shown. Two major drawbacks of using a threshold are depicted at intervals 'a' and 'b'. Interval 'a' illustrates that a threshold is still vulnerable to over-segmentation when there are clusters of very closely spaced local maxima above the threshold. During informal testing, such clusters were found to be frequent and typically only one or two peaks would correspond to true phoneme boundaries while the rest were insertions. Remedies include applying a

min-max filter to the local score, peak masking, or smoothing with a Hanning window. These ad-hoc measures come with the cost of additional parameters that require optimisation [1; 10].

Another drawback of using a threshold to reduce over-segmentation is that long time intervals may occur where no local maxima are above the threshold and therefore no boundaries are hypothesised, for example interval 'b' in Figure 2.1. In speech, such very long phonemes are unlikely, and hence this situation probably indicates that one or more boundaries have been missed.

2.4 Summary and conclusion

This chapter has introduced the concept of a local score: a function that indicates the presence of change at a certain point within the speech signal. Blind local scores apply vector distance functions to the feature vectors, and are ignorant of the characteristics of the signal that is being segmented, including the language. It was also described how MLPs can be used to compute local scores on the basis of a group of consecutive speech frames, and how these local scores are able to detect patterns of change between frames.

At the points in time when the acoustic change is at a maximum, phoneme boundaries are likely to be present. Following this reasoning, segment boundaries are hypothesised at the peaks of the local score. In addition to the larger peaks, the local score will also contain many smaller peaks due to small acoustic changes. Most of these smaller peaks do not correspond to phoneme boundaries and can lead to over-segmentation, the hypothesis of more boundaries than are truly present. Segment boundaries at these locations can be avoided by ignoring all peaks falling below a chosen threshold. Such thresholds were found to be the predominant technique used by the segmentation algorithms in the literature to reduce over-segmentation, even though they have several drawbacks. The most serious drawback is that clusters of peaks above the threshold continue to cause over-segmentation. To address this, and the other drawbacks, a DP-based algorithm that detects segment boundaries using not only the local score, but also the segment lengths, is proposed in Chapter 4.

Chapter 3

The TIMIT speech corpus

It is common to evaluate automatically produced segmentations by measuring how similar they are to hand-crafted phonetic time-alignments. Unfortunately very little speech material with accompanying manual phonetic time alignments is available. The TIMIT speech database is one of the few that contains this information, and this is the reason why it is so widely used in literature to evaluate speech segmentations [3; 6; 10; 11; 12; 13; 22]. TIMIT also specifies an exclusive training set on which the MLPs and RBMs, which are used in this thesis, can be trained. The DP-based segmentation algorithm, that will be introduced in the next chapter, also uses this set to estimate the probability distributions that it employs.

3.1 Content

The TIMIT corpus was recorded at Texas instruments (TI) at a sample rate of 16kHz, transcribed at the Massachusette Institute of Technology (MIT), and is maintained by the American National Institute of Standards and Technology (NIST) [1]. It contains speech from 630 speakers, 438 male speakers and 192 female speakers [2], representing the 8 major dialect divisions of American English as illustrated in Table 3.1. Army brat refers to speakers who moved around during their childhood and who are not from a fixed area.

Each of the 630 speakers spoke 10 phonetically-rich sentences, which are divided into three categories:

- Dialect sentences (SA). This category consists of two sentences, namely: “She had your dark suit in greasy wash water all year." and ”Don’t ask me to carry an oily rag like that." The sentences are spoken by all 630 speakers and were designed to make the differences between dialects clear.
- Phonetically-compact sentences (SX). These are short sentences that were designed by hand to contain a large variety of phonetic material. Each of the 450 SX sentences was spoken by 7 different speakers.
- Phonetically-diverse sentences (SI). Sentences in the SI category were selected from existing text sources to provide rich phonetic coverage and to exploit the differences between dialects. Each of the 1890 SI sentences was spoken by only a single speaker.

Table 3.2 shows how these sentences are distributed within the corpus.

Table 3.1: Number of speakers in each dialect region represented in TIMIT [1].

Regions	Male	Female	Total
New England	31 (63%)	18 (37%)	49 (8%)
Northern	71 (70%)	31 (30%)	102 (16%)
North Midland	79 (77%)	23 (23%)	102 (16%)
South Midland	69 (69%)	31 (31%)	100 (16%)
Southern	62 (63%)	36 (37%)	98 (16%)
New York City	30 (65%)	16 (35%)	46 (7%)
Western	74 (74%)	26 (26%)	100 (16%)
Army brat	22 (67%)	11 (33%)	33 (5%)
Total	438 (70%)	192 (30%)	630 (100%)

Table 3.2: The distribution of the three types of sentences in TIMIT [2].

Sentence type	Sentences	Speakers	Total	Sentences/Speaker
Dialect (SA)	2	630	1260	2
Compact (SX)	450	7	3150	5
Diverse (SI)	1890	1	1890	3
Total	2342		6300	10

3.2 The training, development, and core-test sets

In speech applications it is typical to exclude the SA sentences because their high repetition rate biases the models and the results [1; 2; 3]. This convention will be followed in this thesis. The corpus was divided into training, development, and core-test sets that can be used for training, development, and final independent testing respectively. There is no speaker overlap between any of these three sets. The training set will be used to train the MLPs and RBMs, and to make probability distribution estimates for the DP algorithm (see next chapter). The development set, consisting of 50 speakers drawn from the full 168 speaker test set, is used to optimise the parameters of the algorithms, and the core-test set is used exclusively for final testing. The number of speakers, utterances, and hours of speech contained in these sets are shown in Table 3.3. For a more detailed description of these sets, refer to [2].

Table 3.3: Number of speakers, utterances, and hours of speech in the training, development, and core-test sets [2].

Set	Speakers	Utterances	Hours
Train	462	3696	3.14
Development	50	400	0.34
Core Test	24	192	0.16

The TIMIT sets define 61 different phones, and the phonetic annotations indicate the boundary positions of these. In this thesis these boundaries will be used without any modification.

3.3 Conclusion

The TIMIT corpus was chosen for experimentation, because it is the only speech corpus we could find that contains manual phonetic time-alignments. The structure of the TIMIT corpus has briefly been discussed, and its division into three exclusive sets that can be used for training, development, and testing respectively was described.

Chapter 4

A DP-based segmentation algorithm

Apart from the widespread threshold-based segmentation algorithms found in the literature, an interesting algorithm [23] was discovered that incorporates segment lengths by using dynamic programming. The use of segment lengths in addition to the local score, improves segment boundary decisions. For example, very short and very long segments will inherently be penalised, leading to a possible solution to some of the drawbacks associated with the threshold-based approaches illustrated in Section 2.3.

Although this DP algorithm introduced some novel ideas, it was not executed optimally and the experimental evaluation was limited. In this chapter we will build on the work done in [23] by incorporating the DP approach into the segmentation process by means of a Markov chain.

4.1 DP-based segmentation cast as a Markov chain

Consider a signal consisting of N+1 frames. Now let the time of occurrence of each frame correspond to a state in a Markov chain as shown in Figure 4.1, where M is the maximum allowed number of frames per segment and S_0 is the state corresponding to the time of occurrence of the first frame in the signal. The vertical dashed arrows between S_1 and S_1, and between S_{N-1} and S_{N-1} indicate an expansion of the same state. Each state in the model can be expanded in this way.

When a state is visited, a segment boundary is considered to occur at the corresponding speech frame. Transition and emission probabilities are calculated according to Equations 4.1.1 and 4.1.2 respectively, where SL refers to the segment length, LS to the local score, and SB to the occurrence of a segment boundary.

$$a_{i,j} = P(SL(S_j, S_i)) \tag{4.1.1}$$

$$b_j = P(SB|LS(S_j)) \tag{4.1.2}$$

The segment length in Equation 4.1.1 is equal to the frame skip between two consecutive frames multiplied by the number of states separating the currently visited state and its parent state, as shown in Equation 4.1.3, where S_j is the current state, and S_i is the parent state.

$$SL(S_j, S_i) = (j - i) \times frame\_skip \tag{4.1.3}$$

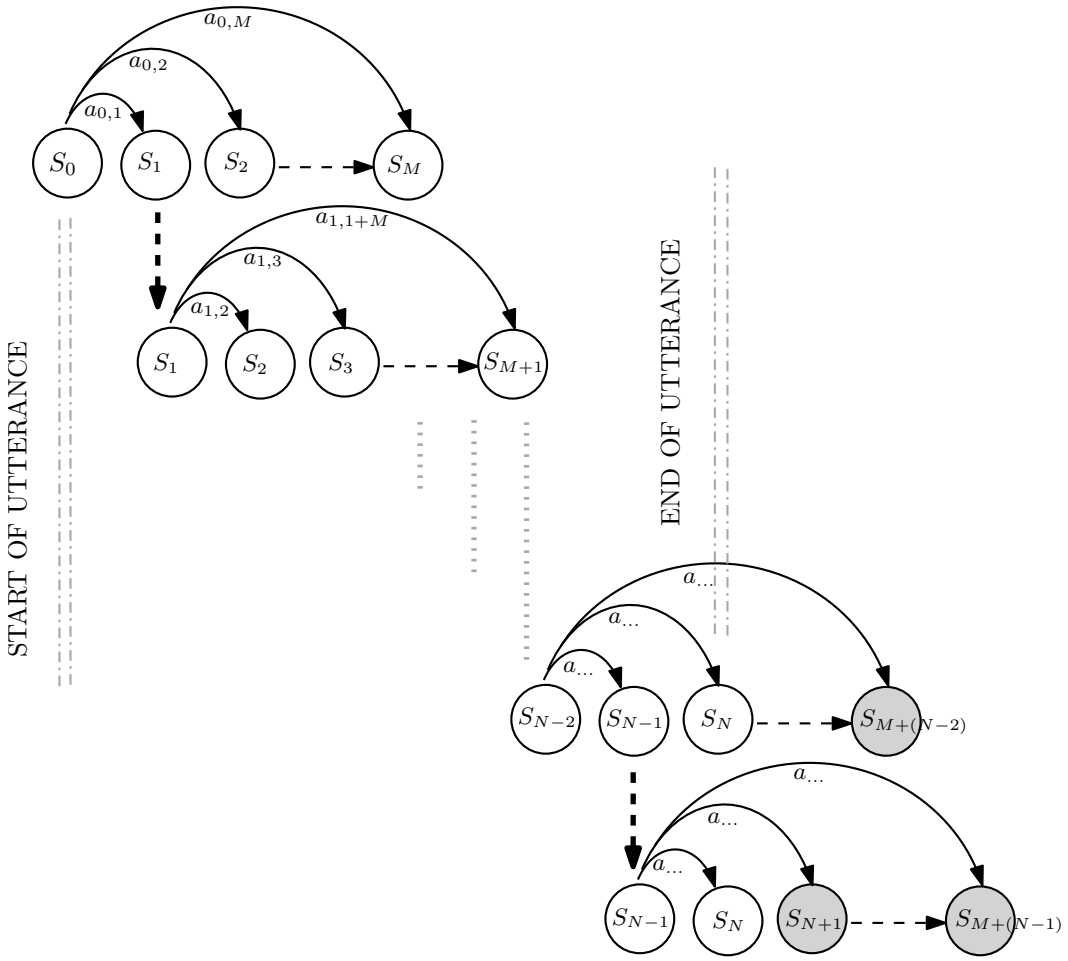


Figure 4.1: DP-based segmentation cast as a Markov chain.

Hence the transition probability is dependent only on the elapsed time between states. The emission probability at state Sj, as shown in Equation 4.1.2, is dependent on the local score $LS(Sj)$. To calculate the emission probability, Bayes rule is applied as shown in Equation 4.1.4, where $!SB$ refers to the absence of a segment boundary.

$$P(SB|LS(S_j)) = \frac{P(LS(S_j)|SB)P(SB)}{P(LS(S_j)|SB)P(SB) + P(LS(S_j)|!SB)P(!SB)} \tag{4.1.4}$$

The prior probability of a segment boundary was estimated by dividing the number phoneme boundaries in the TIMIT annotations by the number of frames, as shown in Equation 4.1.5.

$$P(SB) = \frac{number\ of\ phoneme\ boundaries\ in\ TIMIT}{number\ of\ frames\ in\ TIMIT} \tag{4.1.5}$$

4.2 Segment length probability distribution

The probability that a segment has a specific length, Equation 4.1.1, is determined from a probability distribution. A Poisson distribution was proposed in the original algorithm [23] for which the value of lambda can be adjusted to find an optimal segmentation. Figure 4.2 illustrates an example of

the Poisson distribution with lambda equal to 50ms. For illustrative purposes, the distribution is normalised with respect to its maximum probability.

In this thesis, the time-aligned phonetic boundaries given in TIMIT were used to estimate a more accurate segment length probability distribution. A histogram estimation was used for this, and the resulting distribution is illustrated in Figure 4.2. Clearly, the two distributions are quite different.

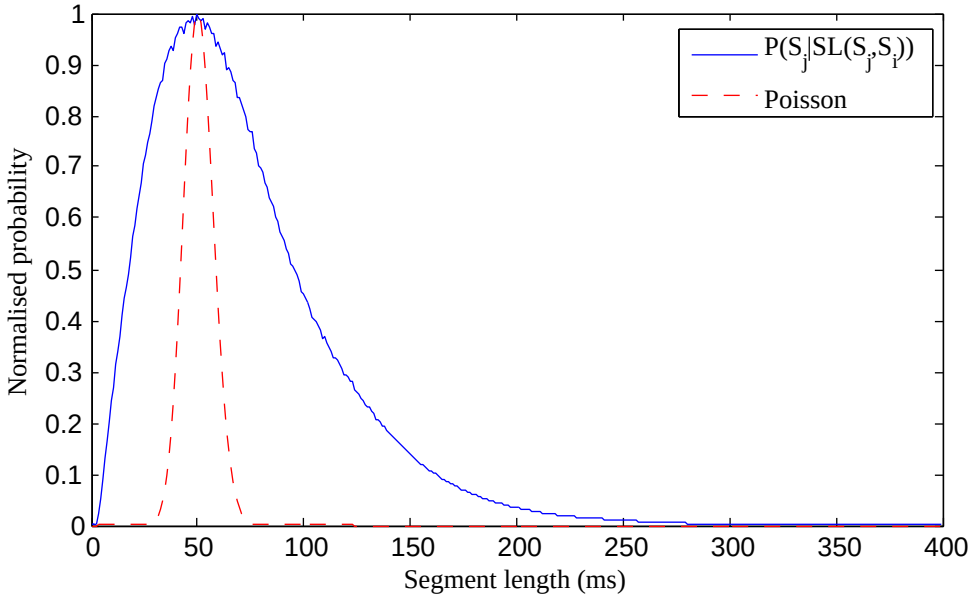


Figure 4.2: Poisson distribution with $\lambda = 50$ms, and the probability distribution of phoneme lengths estimated from TIMIT training set.

The histogram estimated from TIMIT is non zero over an interval stretching between the minimum and maximum segment lengths found in the corpus. However, during dynamic programming, segment lengths up to the length of the utterance must be considered. To allow this, a linear tail stretching to the utterance length is appended to the estimated distribution.

4.3 Local score probability distributions

The emission probability (Equation 4.1.4) requires probabilities of local scores to be estimated for the case when a segment boundary is present at a frame, and for the case when a segment boundary is absent at a frame. These probabilities are determined from two probability distributions, one for each of the two cases. The research described in [23] proposed a manner in which these probability distributions can be estimated for each utterance. A probability distribution of the local score given a segment boundary is estimated by applying a vector distance function to frames that have a large fixed time interval between them (200ms is proposed in [23]). These frames are likely to have different spectral components, and this dissimilarity will be indicative of a boundary. A similar approach is applied to determine the probability of a local score given that a segment boundary is absent. This is achieved by considering frames that are separated by a very small time interval (20ms is proposed in [23]), which means they are likely to have similar spectral components.

In this thesis, the TIMIT phoneme boundaries were used instead to estimate more accurate distributions. To gain some insight into the behaviour of the local scores near phoneme boundaries, the local scores in the close vicinity of the TIMIT phoneme boundaries were calculated and used to estimate a local score probability distribution given that a segment boundary is present. A similar distribution was determined for the local scores far from boundaries, i.e. a local score probability distribution given that a boundary is absent. Figure 4.3 shows these estimations for the normalised city block distance (Section 6.2.2) applied to the FFT as the local score.

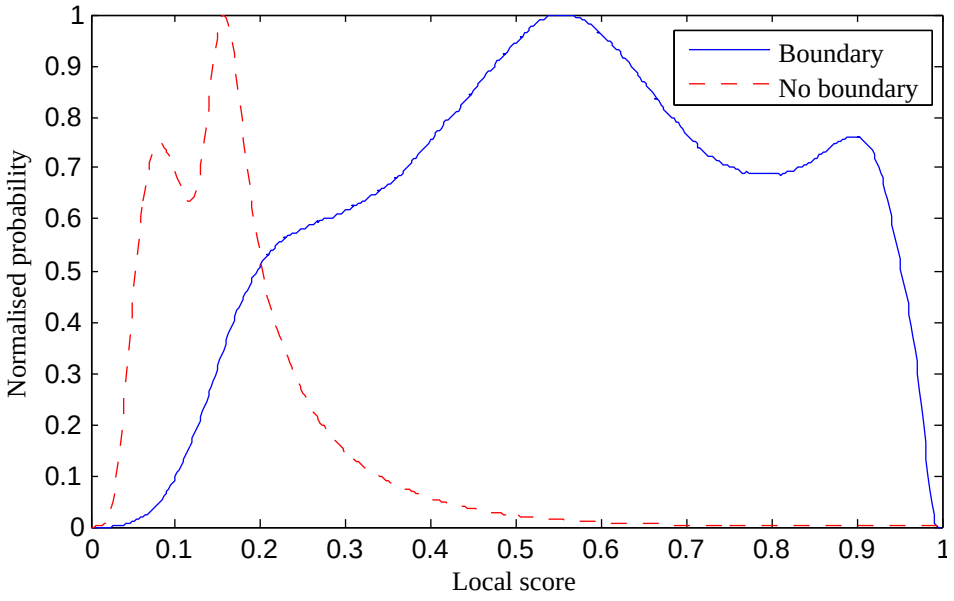


Figure 4.3: Probability distributions of the local score when a segment boundary is present and when it is absent. These were estimated from the local scores at frames close to the TIMIT phoneme boundaries and at frames far from the boundaries.

The distributions of the local scores and the phoneme lengths can now be used to determine the probability of a boundary to occur at a specific frame in a speech signal.

4.4 The optimal path

To find the globally optimal path from the beginning to the end of the utterance (from state S_0 to state S_N) all possible transitions shown in Figure 4.1 must be considered. This can be accomplished by using dynamic programming.

This procedure can be illustrated by considering a matrix containing the probabilities of visiting a state from a specific parent state as shown in Figure 4.4. Each row represents a different parent state. For example, in row 1, S_0 is the parent, while in row 2 it is S_1. The value shown in each cell is the probability of a transition from the parent to the current state (indicated by the column number), multiplied by the emission probability at the current state and by the maximum path probability from S_0 to the parent state. The maximum path probability from S_0 to the parent is equal to the maximum value in column i ($max(C[i])$). Hence the matrix values are calculated on a row by row basis.

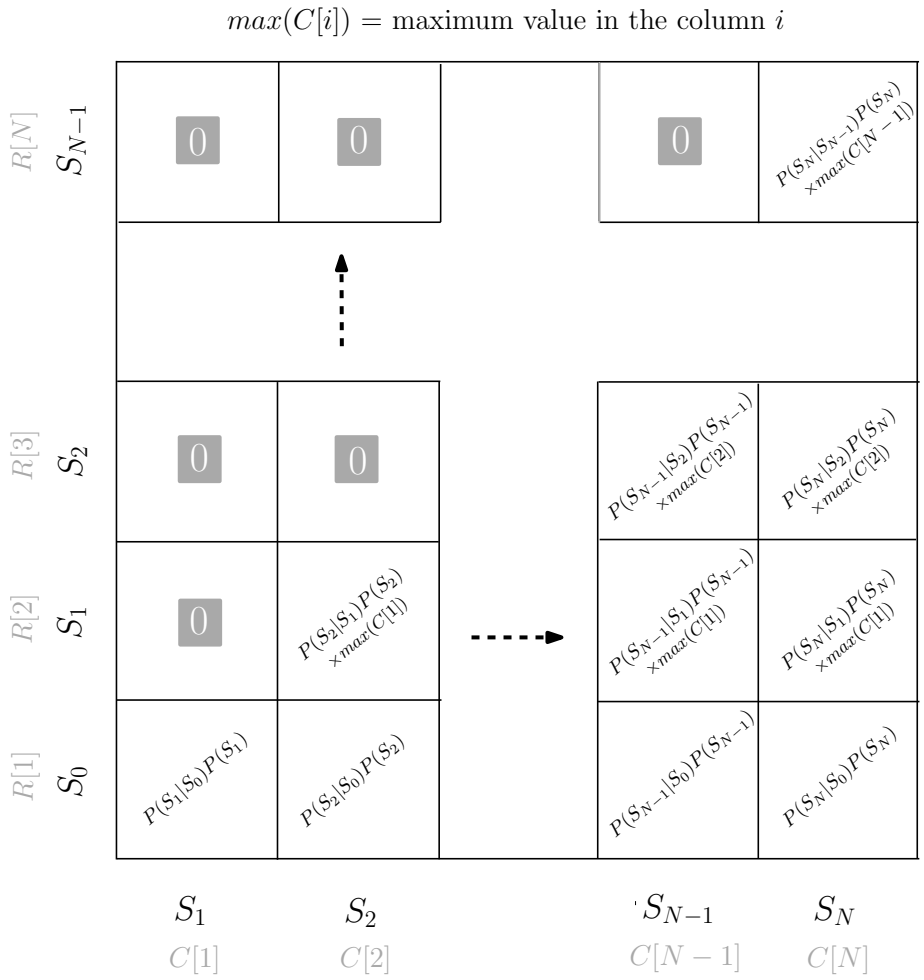


Figure 4.4: Optimal path matrix.

Finally, the optimal path can be obtained by tracing back from the last column of the matrix to state S_0. Two steps should be repeated until state S_0 is reached, these two steps are shown in Figure 4.5.

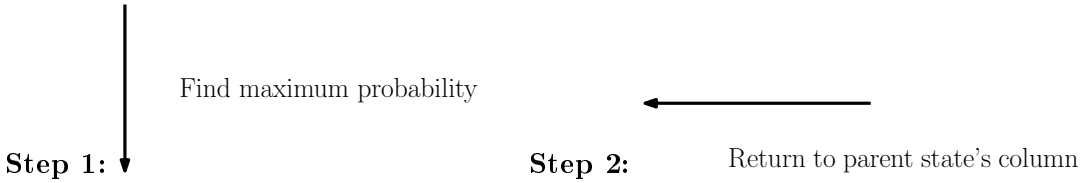


Figure 4.5: Re-tracing the optimal path.

The parent states that were visited along the backward trace will identify the optimal path, and therefore the optimal segmentation. It is important to note that S_0 and S_N are always included in the path, and therefore the algorithm assumes that segment boundaries are always present at the start and the end of the speech signal. This means that any initial and final silence must be removed before applying the algorithm.

4.4.1 Normalising path length

During the search for the optimal path, many probabilities are multiplied together for any given path. Shorter paths (which contain fewer multiplications and thus longer segments) may be preferred, even when these have low associated emission and transition probabilities. We compensate for this effect by modifying the emission and transition probabilities as shown in Equations 4.4.1 and 4.4.2.

$$a_{i,j} = P(S_j|SL(S_j, S_i))^{SL(S_j,S_i)} \tag{4.4.1}$$

$$b_j = P(SB|LS(S_j))^{SL(S_j,S_i)} \tag{4.4.2}$$

These modifications normalise the path probability and remove the bias towards segmentations containing fewer segment boundaries. The necessity of this normalisation is illustrated in Figure 4.6.

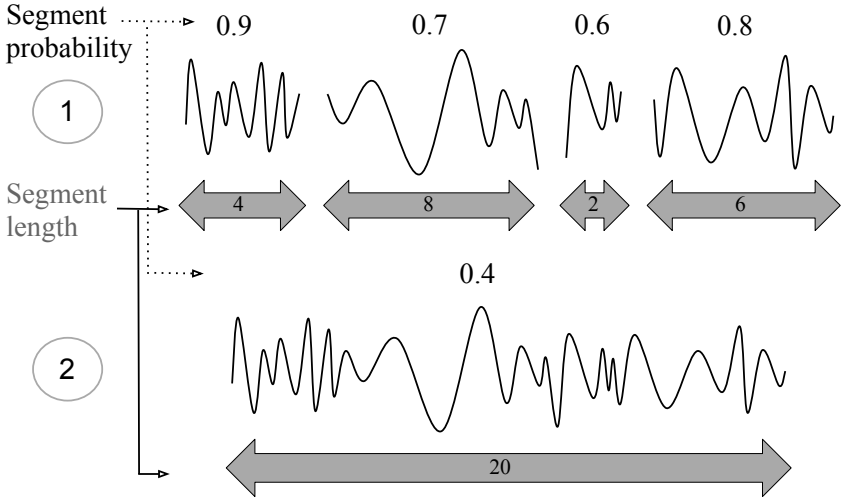


Figure 4.6: Comparison of two segmentations to demonstrate the necessity of path normalisation.

The path probability of segmentation 1 is equal to $0.9 \times 0.7 \times 0.6 \times 0.8 = 0.3$, which is smaller than the path probability of segmentation 2, which is 0.4, even though all the segments in segmentation 1 have substantially higher probabilities than the segment in segmentation 2. This shows that the algorithm gives preference to the shortest path. With path normalisation, the path probability of segmentation 1 becomes $0.9^4 \times 0.7^8 \times 0.6^2 \times 0.8^6 = 3.5 \times 10^{-3}$, which is bigger than the path probability of segmentation 2, which is $0.4^{20} = 1.1 \times 10^{-8}$, thereby showing that path normalisation is required to give the segmentation with the most probable segments the larger path probability.

4.5 Summary and conclusion

In this chapter we introduced an alternative segmentation algorithm to the typical threshold-driven approach. This alternative approach incorporates a segment length probability distribution into the segmentation process by dynamic programming. This was achieved by casting the segmentation problem as a Markov chain, where transition and emission probabilities correspond to the segment lengths and local scores respectively. By incorporating segment lengths in addition to the local score we

hope that the segmentation algorithm will be free from the drawbacks associated with the threshold approach.

Chapter 5

Assessing segmentation accuracy

In order to assess the quality of automatically-generated segmentations, we will determine how closely they correspond to the TIMIT phonetic segmentations. Three different approaches will be used, each comparing the hypothesised and phoneme boundary sequences in a different way. These three measures are the dynamic programming (DP) alignment, average error, and R-value.

5.1 Comparing segmentations by dynamic programming (DP)

In this approach, the best alignment between two sequences of boundary times will be determined by DP, which will also calculate a path cost. The alignment procedure is carried out using a matrix of partial path costs. In this matrix, the number of rows is equal to the number of boundaries in the hypothesised sequence, and the number of columns is equal to the number of boundaries in the reference sequence as shown in Figure 5.1.

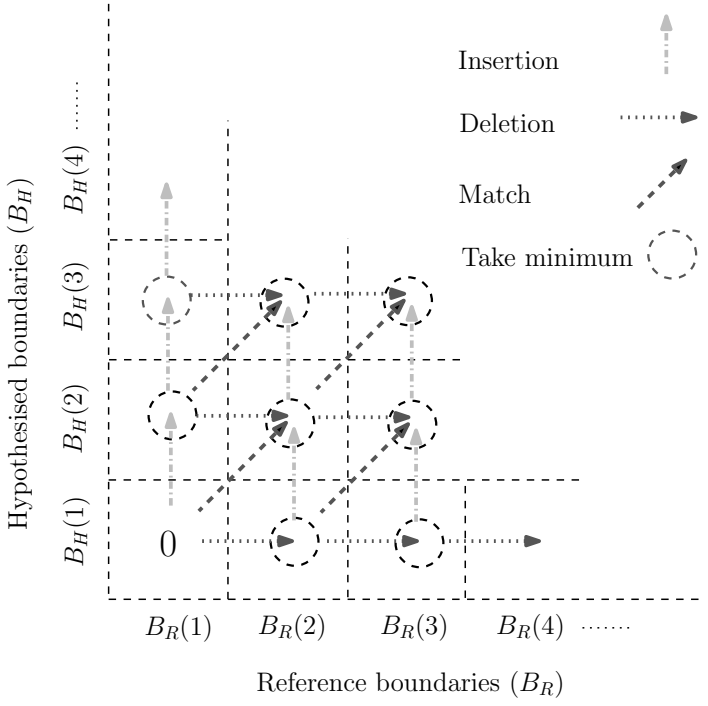


Figure 5.1: A matrix for determining the alignment between two sequences of segment boundary times.

The first boundary in both sequences must coincide, and this corresponds to the bottom left cell of the matrix. Three alternative scenarios are then considered: (i) a hypothesised boundary $B_H(i)$ is paired with (matches) a boundary $B_R(j)$ in the reference segmentation, (ii) a hypothesised boundary $B_H(i)$ is not paired with any boundary $B_R(j)$ in the reference segmentation (insertion) or (iii) there is no hypothesised boundary that can be paired with a boundary $B_R(j)$ in the reference segmentation (deletion).

All possible paths from the bottom left cell to top right cell in the matrix shown in Figure 5.1 are computed recursively by dynamic programming. Starting from the bottom left of this matrix, each path can be extended upwards, to the right, or diagonally up and to the right, indicating an insertion, a deletion or a match between boundaries respectively. Each of these possibilities has a specific associated cost, as described in Equations 5.1.1–5.1.5.

Insertion cost:

If $B_R(j) \leq B_H(i)$ **and** $B_H(i) < B_R(j+1)$

$$ins(i,j) = min(B_H(i) - B_R(j), B_R(j+1) - B_H(i)) \tag{5.1.1}$$

else

$$ins(i,j) = abs(B_H(i) - B_R(j)) \tag{5.1.2}$$

Deletion cost:

If $B_H(i) \leq B_R(j)$ **and** $B_R(j) < B_H(i+1)$

$$del(i,j) = min(B_R(j) - B_H(i), B_H(i+1) - B_R(j)) \tag{5.1.3}$$

else

$$del(i,j) = abs(B_R(j) - B_H(i)) \tag{5.1.4}$$

Match cost:

$$match(i,j) = abs(B_H(i) - B_R(j)) \tag{5.1.5}$$

When a reference boundary falls between two hypothesised boundaries, or vice versa, the cost is calculated by considering the closest of the two boundaries, as indicated by Equations 5.1.1 and 5.1.3. Otherwise the path cost is equal to the absolute time difference between the two boundaries (Equations 5.1.2, 5.1.4, and 5.1.5). When paths meet, only the path with the lowest cost survives. This is shown by Equation 5.1.6, where PC denotes 'Path Cost' and $min(.,.,.,.)$ is a function that returns the minimum of three values.

$$\begin{aligned} PC(i,j) = min\big((&PC(i-1,j-1) + match(i,j)), \\ &(PC(i-1,j) + ins(i,j)), \\ &(PC(i,j-1) + del(i,j))\big) \end{aligned} \tag{5.1.6}$$

In its described form, the algorithm may produce counterintuitive alignments, because a match carries the same weight as an insertion or a deletion. Consider the boundary sequences shown in Figure 5.2. If $B_H(2)$ is matched with $B_R(2)$, and $B_H(3)$ is considered an insertion; or if $B_H(3)$ is matched with $B_R(2)$, and $B_H(2)$ is considered an insertion; the path cost will be identical. A similar scenario exists for $B_R(3)$ and $B_R(4)$, either one of which can be considered a deletion while the other is matched with $B_H(4)$. To avoid this, a match must be given a slightly larger weight in the computation of the DP alignment cost, e.g. a scaling factor of 1.01. In Figure 5.2, this results in matches between $B_H(3)$ and $B_R(2)$, and $B_H(4)$ and $B_R(4)$, while $B_H(2)$ is an insertion and $B_R(3)$ is a deletion.

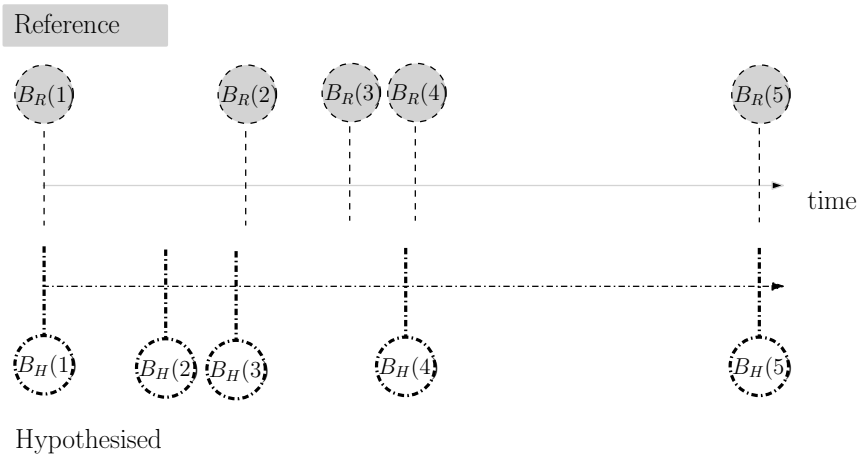


Figure 5.2: An example of two boundary sequences.

Equation 5.1.6 is applied iteratively until all paths have reached the top right cell, which will then contain the final alignment cost between the two sequences. This cost reflects the difference between the hypothesised and reference sequences since it is the cumulative cost of every match, insertion, and deletion in the alignment. Furthermore, the cost has dimensions of time. By dividing it by the number of reference boundaries, the cost in seconds per reference boundary can be obtained. This is the average time difference between a paired hypothesised- and reference boundary. In addition, the number of insertions, deletions, and matches can be obtained by tracing back along the optimal path.

5.2 Fixed margin method

It is standard practice in related research to consider a hypothesised and a reference segmentation boundary to be a match (or hit) whenever they occur within 20 ms of one another [3; 4; 8; 10; 11; 12; 13; 16; 18]. Only one hypothesised boundary can be matched to a reference boundary, and when a hypothesised boundary falls between two reference boundaries that are within 40 ms of each other, it must be closer to one of them than the midpoint between them to be a match [24]. All non-matching boundaries are then regarded as either insertions or deletions. In order to make our results comparable to those of others, this scoring framework has been employed. An error measure termed the **average error** (ERR) is calculated. This figure is the average percentage insertions (INS) and deletions (DEL) taken with respect to the number of reference boundaries in the utterance. Note that these insertions and deletions are the ones that will be shown in the experimental results, and not those of the DP

approach. This is because the path costs of the insertions and deletions corresponding to the dynamic program are already a part of the total path cost.

5.3 R-value

The R-value is a scalar value between 0 and 1 that was proposed for use in segmentation algorithms by Räsänen et al [24]. The hit rate (HR) is defined as the number of hits (N_{hit}) divided by the number of reference boundaries in an utterance (N_{ref}), and the over-segmentation (OS) as the number of hypothesised boundaries (N_f) divided by the number of reference boundaries minus 1, as shown by Equations 5.3.1 and 5.3.2, respectively. A hit occurs when a segment boundary is placed within a 20 ms fixed margin of the true boundary.

$$HR = \frac{N_{hit}}{N_{ref}} \times 100 \tag{5.3.1}$$

$$OS = (\frac{N_f}{N_{ref}} - 1) \times 100 \tag{5.3.2}$$

The R-value is defined as the average of two distances defined on a plane whose axes are HR and OS respectively, as shown in Figure 5.3. For every utterance, a target point (TP) is defined at which the HR is 100% and the OS is 0%. The hypothesised segmentation is indicated by the point X. Now let r_1 be the distance from TP to the hypothesised segmentation result, and r_2 be the distance from X to a point perpendicular to the 0% insertions line. The 0% insertions line is defined as the over-segmentation when there are 0% insertions. The r_2 distance serves to penalise insertions more strongly. When there are 0% insertions the number of hypothesised boundaries N_f are equal to the number of hits N_{hit}, and the over-segmentation OS is equal to the hit-rate HR minus 100 according to Equation 5.3.2. The r_1 and r_2 distances are determined by using Equations 5.3.3 and 5.3.4 respectively, from which the R-value as calculated by Equation 5.3.5.

$$r_1 = \sqrt{(100 - HR)^2 + OS^2} \tag{5.3.3}$$

$$r_2 = \frac{-OS + HR - 100}{\sqrt{2}} \tag{5.3.4}$$

$$R = 1 - \frac{abs(r_1) + abs(r_2)}{200} \tag{5.3.5}$$

The longer either one of the distances are, the smaller the R-value will become. Hence, the larger the R-value, the better the segmentation performance. The HR and OS can also be interpreted in terms of the percentage insertions (INS) and deletions (DEL) as illustrated by Equations 5.3.6 and 5.3.7, where N_{del} and N_{ins} are the number of deletions and insertions respectively. The R-value can then be defined in terms of INS and DEL as shown by Equation 5.3.8.

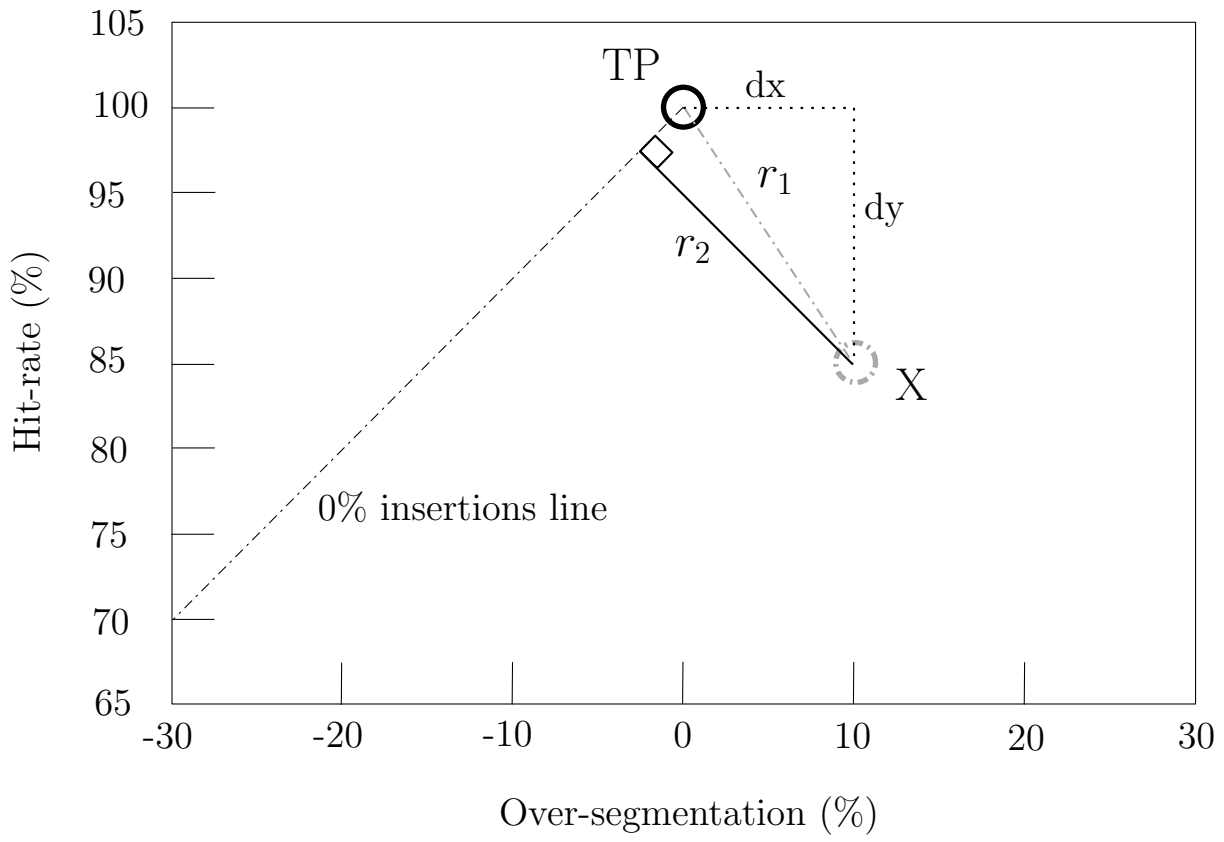


Figure 5.3: Calculation of the R-value from the over-segmentation (OS) and the hit rate (HR) of the current result (X). TP indicates the target point corresponding to OS=0 and HR=100.

$$\begin{aligned} HR &= \frac{N_{ref} - N_{del}}{N_{ref}} \times 100 \\ &= 100 - DEL \end{aligned} \tag{5.3.6}$$

$$\begin{aligned} OS &= (\frac{N_{hit} + N_{ins}}{N_{ref}} - 1) \times 100 \\ &= HR + INS - 100 \\ &= INS - DEL \end{aligned} \tag{5.3.7}$$

$$R = 1 - \frac{abs(\sqrt{DEL^2 + (INS - DEL)^2}) + abs(-\frac{INS}{\sqrt{2}})}{200} \tag{5.3.8}$$

From Equation 5.3.8 it is clear that the larger the difference between the number of insertions and deletions, the smaller the R-value will be. The deletions are penalised slightly more strongly than the insertions, because the effect of INS is reduced by a factor of $\frac{1}{\sqrt{2}}$. Larger R-values therefore result when the insertions and deletions are few and insertions are approximately equal to deletions. Hence

a large R-value indicates not only that the segmentation accuracy is good, but also that the number of insertions is close to the number of deletions.

5.4 Summary and conclusion

We introduced three different measures that can be used to assess the accuracy of the automatically produced segmentations. Each method compares the automatically produced segmentations to the corresponding manually-placed phoneme boundaries in TIMIT. The DP-based approach finds the best time alignment between the hypothesised and phoneme boundary sequences. The fixed margin method defines fixed 20 ms regions around the reference boundaries, and considers a hypothesised boundary to be a match when it falls within this region. The R-value is calculated from the number of insertions and deletions determined by the fixed margin method.

A disadvantage of the fixed margin method is that all insertions and deletions are considered equal regardless of their positions. For example, a succession of deletions is not explicitly penalised in the fixed margin method. Comparing segmentations by DP penalises insertions and deletions relative to their closest paired boundary, and a succession of deletions will therefore result in a large cost because the closest paired boundary will be far away. We therefore use the DP evaluation mechanism in combination with the fixed margin method and R-value to obtain a better impression of how closely two boundary sequences are aligned.

Chapter 6

Experimental setup for blind local scores

6.1 Experimental data

Our experimental evaluations are based on the TIMIT database. The use of an explicit development set avoids biased results which would be obtained if the performance of the algorithm was measured on the same data used to optimise its hyperparameters. In the literature dealing with automatic segmentation, the separation of development and testing data was found to be uncommon. Leading and trailing silences were removed to account for the assumption that each utterance begins and ends with a segment boundary.

6.2 Experimental setup

In the following we evaluate the performance of different feature vector and vector distance function combinations as local scores when embedded into the DP-framework described in Chapter 4.

6.2.1 Feature vectors

We have chosen four feature vector configurations popular in the literature on automatic speech segmentation for comparative experimentation.

1. FFT: Unprocessed FFT magnitudes
2. MFCC: 12 MFCCs and log energy
3. MFCC+Δ+$\Delta\Delta$: MFCC with appended first and second derivatives
4. FBANK: 16 filter-bank coefficients

The number of FFT magnitudes is half the number of samples in a frame, and the centre frequencies of the filter banks were calculated according to the Mel-scale. By considering the local scores separately for the MFCCs, for the delta and for the acceleration features, it was found that a peak for the MFCCs or the acceleration components always coincides with a valley for the delta component, and vice versa.

To account for this, the overall local score was calculated by averaging the local scores calculated for MFCCs and acceleration components, and the negative of the local score for the deltas.

6.2.2 The vector distance functions

In addition to the vector distance functions that were mentioned in Section 2.1, we also investigate the normalised city block distance (also called the taxicab distance):

$$LS(F_j) = \frac{\sum_{l=1}^{T} |f_l - g_l|}{\sum_{l=1}^{T} |f_l| + \sum_{l=1}^{T} |g_l|}. \tag{6.2.1}$$

The vector distance functions that were considered in our experiments were therefore:

1. The cosine distance (C), as given by Equation 2.1.1,
2. The Euclidean distance (E), as given by Equation 2.1.3, and
3. The normalised city block distance (NCB), as given by Equation 6.2.1.

In our experiments, f and g were taken to be the frame to the left and to the right of the inspected frame, respectively. Depending on the local score, boundaries are expected to occur at either peaks or valleys (local maxima or minima) of the local score. Equation 4.1.4 is therefore only calculated at frames which coincide with local maxima or minima of the local score and a probability of 0 is assigned to all other frames.

6.2.3 Weighting of transition and emission probabilities

As it stands, the DP segmentation algorithm will give equal weight to the transition and emission probabilities, which are due to the segment length and local score, respectively. However, it may be beneficial to shift the balance more strongly towards one or the other. By multiplying the log values of the emission and transition probabilities by positive constants that sum to one, this shift in balance can be achieved, and will allow deletions to be traded for insertions and vice versa. Optimal performance on the development set was achieved by assigning a heavier weight to the emission probability (0.6–0.7) than to the transition probability (0.3–0.4). This gives a stronger preference to higher emission probabilities and leads to a reduction in insertions.

6.3 Summary and conclusion

The experimental setup for blind segmentation was discussed, with a focus on the feature vector parameterisations and vector distance functions that will be evaluated in the experiments. In the next chapter, the results produced by these experiments are described.

Chapter 7

Experimental results for blind local scores

7.1 Smoothing window size

Before calculating the local score, each resulting MFCC, FFT, and FBANK vector was smoothed by taking the average within a window centered on the feature vector in question. Subsequently, using the DP framework described in Chapter 4, the average DP cost (Section 5.1), the average error (Section 5.2), and the average R-value (Section 5.3) were calculated on the development set for different combinations of frame lengths, frame skips, and smoothing window lengths. Frame lengths of 10 ms, 15 ms, 20 ms, and 25 ms; frame skips of 5 ms, 10 ms, and 15 ms; and smooth window lengths of up to 70 ms long were investigated.

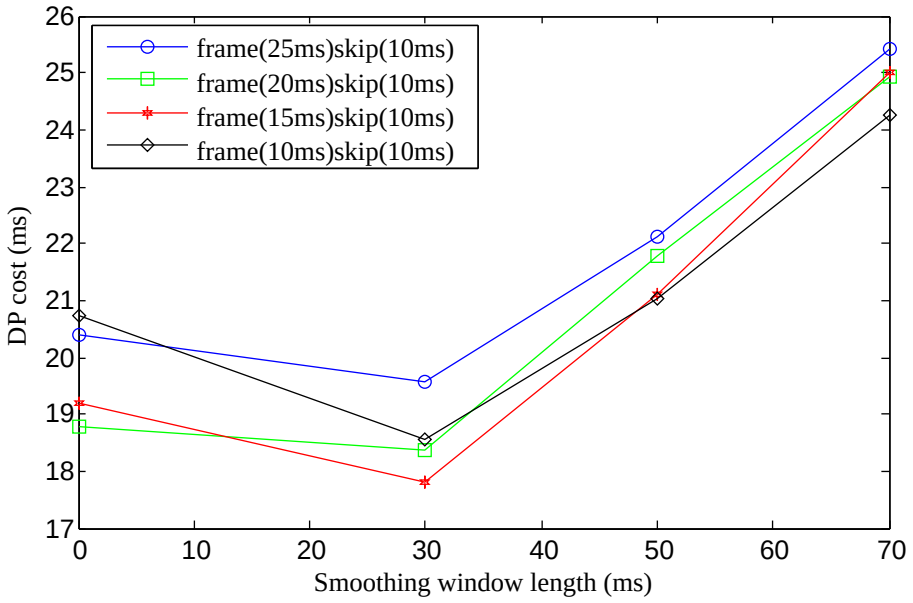


Figure 7.1: DP cost (Section 5.1) for frames of different lengths, a frame skip of 10 ms, and for different smoothing window lengths on the development set for the normalised city block distance applied to the FFT.

Figures 7.1, 7.2, and 7.3 respectively show the results when using a frame skip of 10 ms for the DP cost, average error, and R-value for the case in which the normalised city block distance is applied to the FFT. Note that only the results obtained by using the optimal frame skip, i.e. 10 ms in this

case, are illustrated in the figures to save space. Also notice that the optimal performance for all three performance measures is at the same configuration. The optimal configurations for the Euclidean distance applied to the MFCC, the cosine distance applied to the MFCC+Δ+ΔΔ, and normalised city block distance applied to the FBANK are illustrated in Figures 7.4, 7.5, and 7.6 respectively.

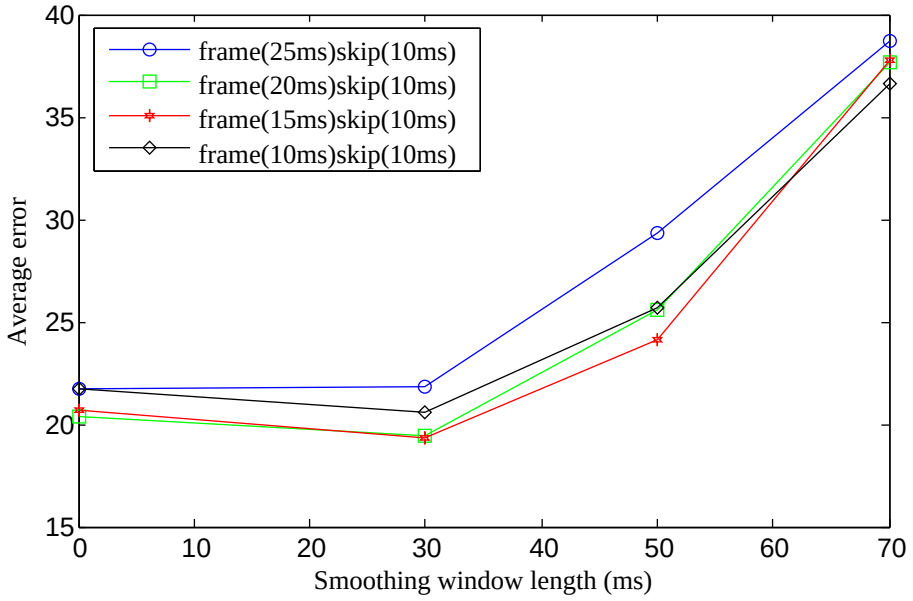


Figure 7.2: Average error (Section 5.2) for frames of different lengths, a frame skip of 10 ms, and for different smoothing window lengths on the development set for the normalised city block distance applied to the FFT.

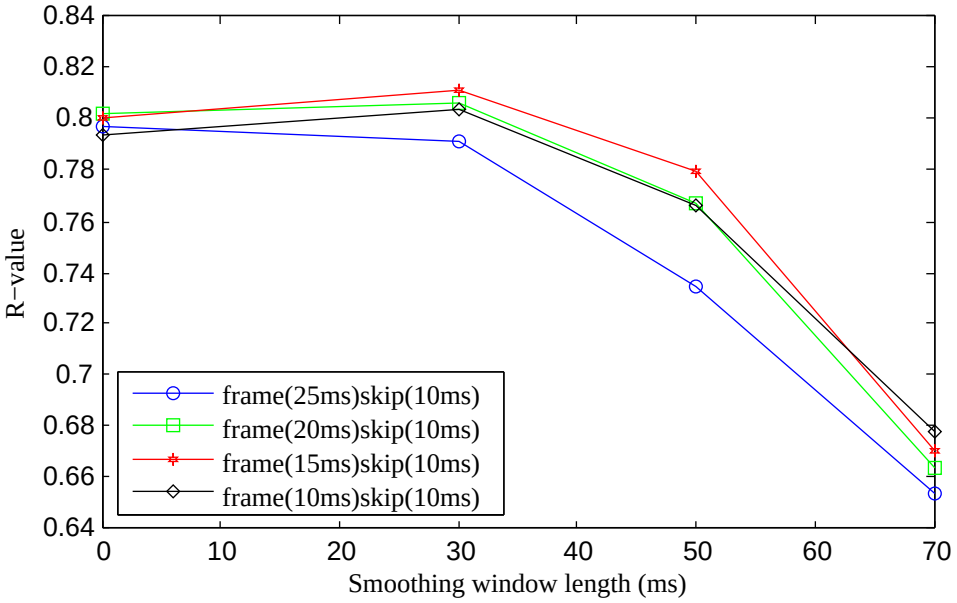


Figure 7.3: R-value (Section 5.3) for frames of different lengths, a frame skip of 10 ms, and for different smoothing window lengths on the development set for the normalised city block distance applied to the FFT.

The results revealed that the optimal configurations are similar for the FFT, MFCC, and FBANK parameterisations, namely a frame skip of 10 ms smoothed over 30 ms with a frame size of either 15 ms or 20 ms. A longer smoothing window (around 45 ms) is required by the MFCC+Δ+ΔΔ

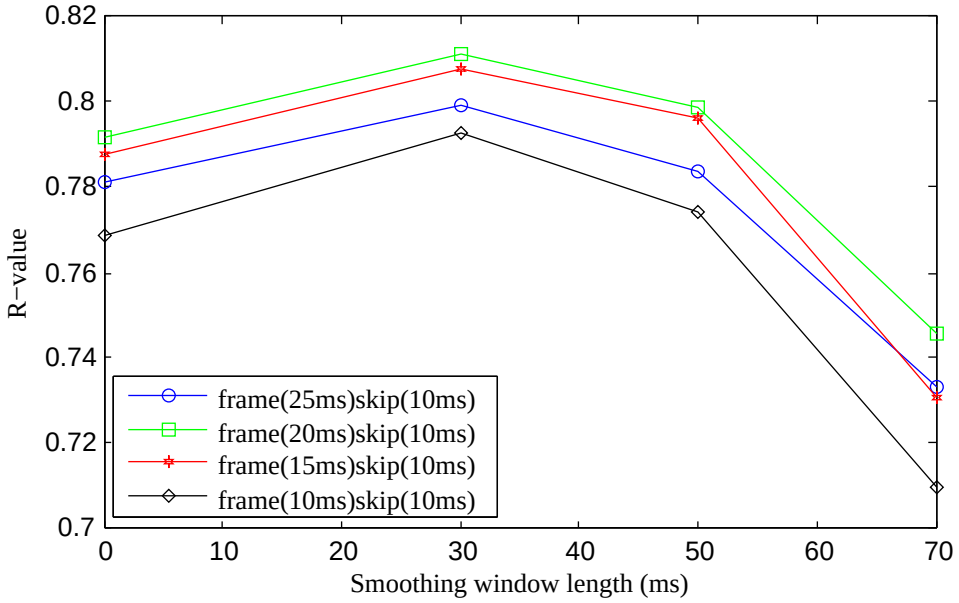


Figure 7.4: R-value (Section 5.3) for frames of different lengths, a frame skip of 10 ms, and for different smoothing window lengths on the development set for the Euclidean distance applied to the MFCC.

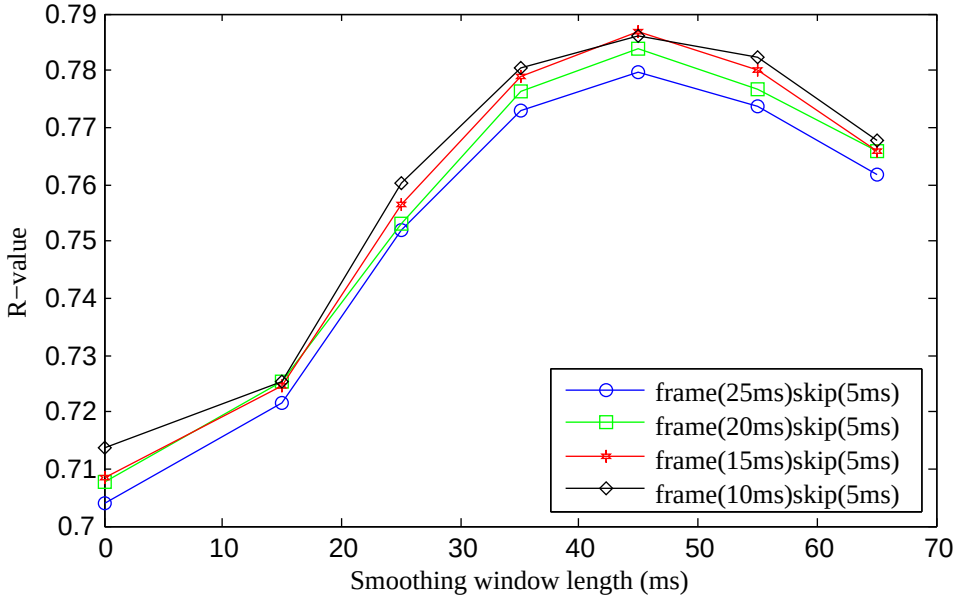


Figure 7.5: R-value (Section 5.3) for frames of different lengths, a frame skip of 5 ms, and for different smoothing window lengths on the development set for the cosine distance applied to the MFCC+Δ+$\Delta\Delta$.

parameters with a small frame skip and frame length of 5 ms and 10 ms respectively. Interestingly, this smoothing window spans exactly the frames used to estimate the derivative. It is also clear that smaller frame skips and frames perform better when determining the derivative.

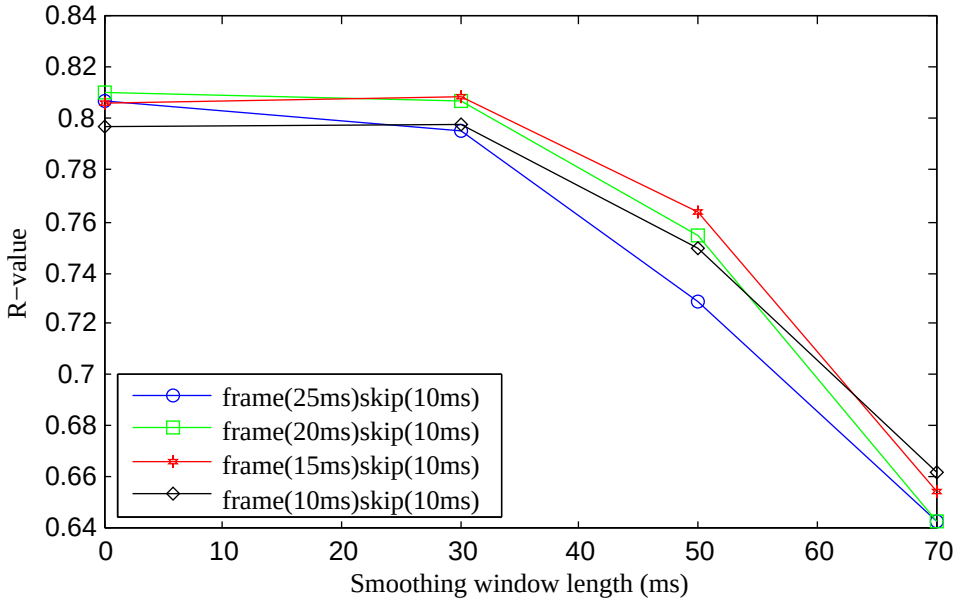


Figure 7.6: R-value (Section 5.3) for frames of different lengths, a frame skip of 10 ms, and for different smoothing window lengths on the development set for the normalised city block distance applied to the FBANK.

7.2 Choice of feature vector and vector distance function

The performance of the DP segmentation algorithm when using the four different feature parameterisations and the three different vector distance functions was compared experimentally, and the results are shown in Table 7.1. For each configuration, the length of the frame, the frame skip, and the smoothing window length as well as the probability weights were optimised on the development set, and segmentation accuracies measured on the core-test set. The average DP path cost, in milliseconds per reference boundary, the average of the fixed margin average error, and the average R-values are shown.

The normalised city block distance delivers the best overall performance with regard to different feature parameterisations, making it the most versatile vector distance function. When applied to the MFCC and the MFCC+Δ+$\Delta\Delta$ parameterisation, the Euclidean distance also achieved very good performance. The cosine distance has average performance for all features.

The configurations that stand out from the rest are the normalised city block distance applied to the FFT, and the Euclidean distance applied to the MFCC and MFCC+Δ+$\Delta\Delta$. Both the FFT and FBANK experience more frequent changes between frames than the MFCC. This is likely due to the encoding of the MFCC that causes it to capture only the more prominent changes between frames. Because of this, the Euclidean and cosine distances both suffer from over-segmentation when applied to the FFT and FBANK parameterisations, while the normalised city block distance performs well. Conversely, the normalised city block distance performs worse than the Euclidean distance when applied to the MFCC. This shows that the normalised city block distance is less sensitive to changes than the Euclidean distance. The frequently changing FFT and the less sensitive normalised city block distance, and the smoother MFCC and the sensitive Euclidean distance are therefore very effective combinations.

The addition of the derivatives for the MFCC results in performance close to the MFCC, but does not

Table 7.1: Development- and test-set performance of the DP segmentation algorithm for four choices of feature vector and for the normalised city block (NCB), Euclidean (E) and cosine (C) local score (LS) formulations.

Configuration		DP Cost (ms)		ERR(%)		R-value	
LS	**Feature**	**Dev**	**Test**	**Dev**	**Test**	**Dev**	**Test**
NCB	FFT	17.81	17.90	19.32	18.83	0.811	0.809
	MFCC	18.30	18.08	21.54	21.55	0.797	0.794
	MFCC+Δ+ΔΔ	19.56	19.04	22.42	22.22	0.790	0.786
	FBANK	17.26	17.07	20.35	20.41	0.810	0.806
C	FFT	23.73	24.07	26.87	27.00	0.749	0.745
	MFCC	19.01	19.00	22.99	23.35	0.787	0.782
	MFCC+Δ+ΔΔ	20.31	19.48	22.76	22.23	0.787	0.785
	FBANK	20.70	21.18	24.68	25.49	0.774	0.762
E	FFT	24.58	24.08	30.58	30.38	0.732	0.729
	MFCC	16.66	16.82	20.02	20.17	0.811	0.803
	MFCC+Δ+ΔΔ	17.41	16.93	20.16	19.87	0.808	0.807
	FBANK	28.02	27.82	39.79	40.36	0.656	0.647

lead to any significant improvements. This is to be expected when considering that the vector distance function already detects the changes between frames. If you consider that the vector distance function resembles taking the first derivatives, then applying a vector distance function to the first and second derivatives will lead to something closely resembling the second and third derivatives, respectively. By taking into account that phoneme boundaries are at the points in time where the local change in the speech signal is at a maximum, it is unlikely that the addition of these derivatives will add much value to the feature vectors. The experiments indicate that the addition of the derivatives is unnecessary, and does not lead to consistently better performance.

When comparing performance on the development and on the core-test sets, it is evident that the same patterns emerge from both. In the experiments that follow, the best overall performing configuration of each vector distance function and feature vector will be used. These are the normalised city block distance for the FFT, the cosine distance for the MFCC, the Euclidean distance for MFCC, and the normalised city block distance for the FBANK. These will henceforth be referred to as configuration C1, C2, C3, and C4 respectively.

Further insight into what configurations perform well can be obtained by inspecting the local score distributions. Figures 4.3, 7.7, 7.8, and 7.9 show the probability distributions of the local score when a segment boundary is present and when a boundary is absent for configurations C1, C2, C3, and C4 respectively. The configurations that perform well, i.e. C1, C3 and C4, all appear to follow a similar trend in terms of the shapes of their distributions when comparing them to the distribution of the worse performing C2. It is clear that the further the main lobes of the 'boundary' and the 'no boundary' distributions are from each other, the better the performance of the configuration. This trend was observed for all configurations, not just C1–4. This makes sense when considering that the classification algorithm needs to make a decision between a boundary and an absence of a boundary at

a frame based partly on the evidence that the local score provides, and that the greater the dichotomy between the 'boundary' and 'no boundary' distributions are, the better the algorithm can discriminate between them.

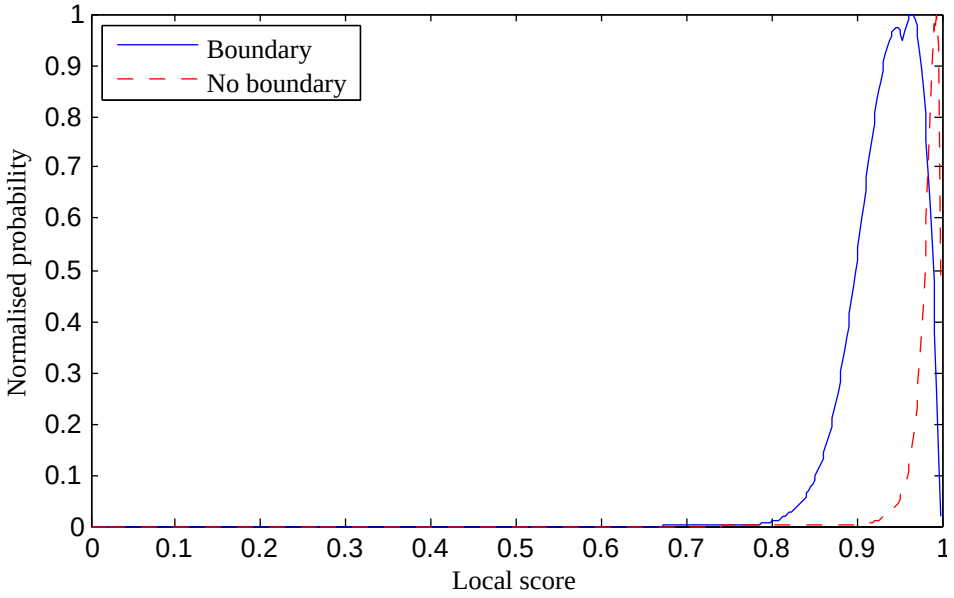


Figure 7.7: Probability distributions of the local score when a segment boundary is present and when it is absent for the cosine distance applied to the MFCC.

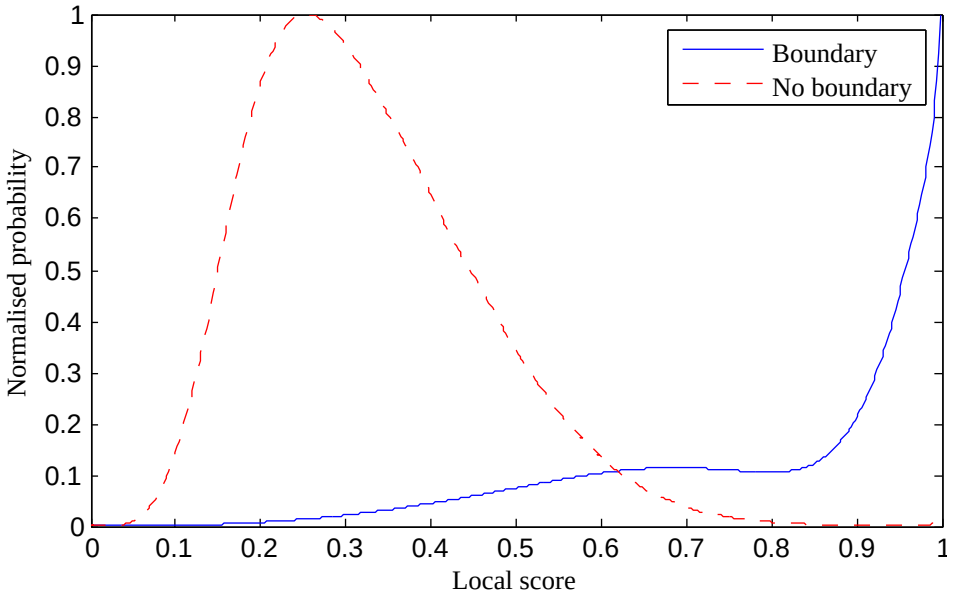


Figure 7.8: Probability distributions of the local score when a segment boundary is present and when it is absent for the Euclidean distance applied to the MFCC.

7.3 Silence removal

Many TIMIT sentences contain regions of silence in which temporal changes nevertheless occur. In order to avoid the hypotheses of segment boundaries in these regions, all boundaries were removed

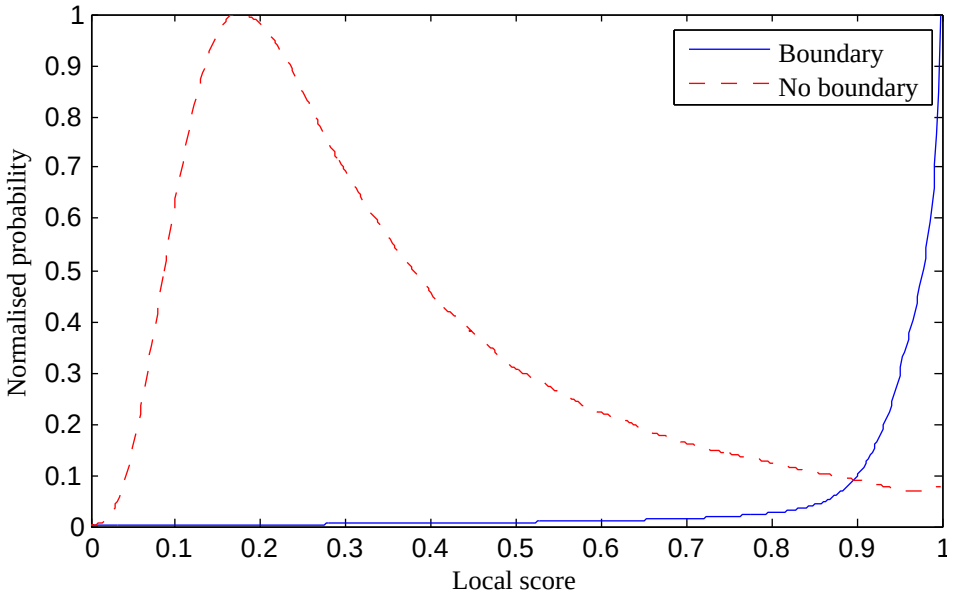


Figure 7.9: Probability distributions of the local score when a segment boundary is present and when it is absent for the normalised city block distance applied to the FBANK.

at frames when the ratio of the average energy content from 30ms before to 30ms after the frame in question, to the mean energy of the signal fall below a certain threshold. Different threshold values were investigated, and a typical result is shown in Figure 7.10. A threshold between 0.1–0.3% (i.e. a value of 0.001–0.003) delivered optimal performance.

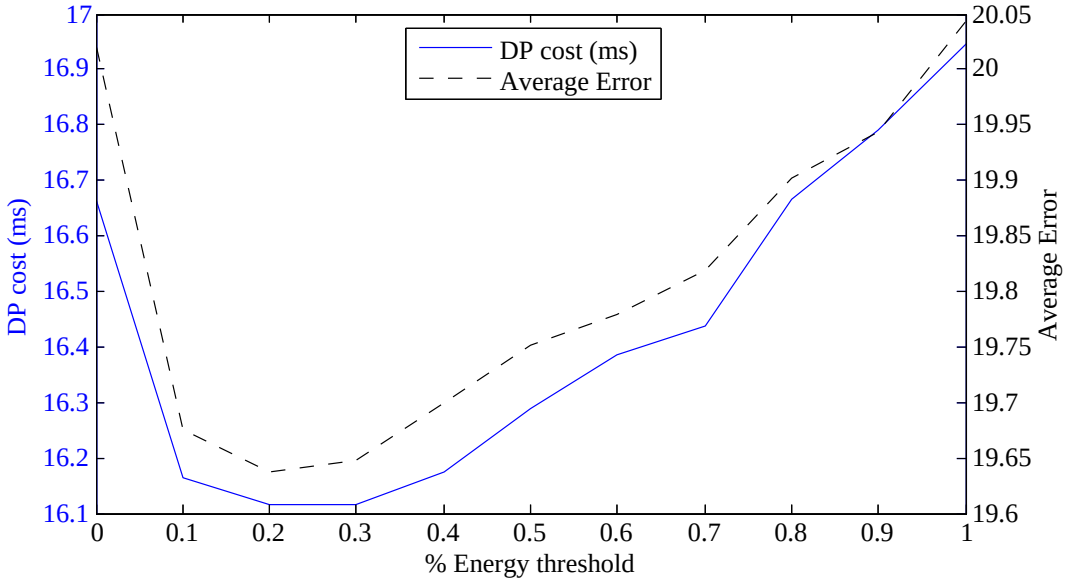


Figure 7.10: DP cost (Section 5.1) and average error (Section 5.2) against % energy threshold on the development set for configuration C3.

7.4 Comparison with other segmentation algorithms

In this section we will benchmark the performance of configurations C1–C4 against two recent approaches to blind speech segmentation found in the literature [6; 10]. The method proposed by Räsä-

nen et al. as described in Section 2.1.1.1 is claimed to achieve the same or better performance than many earlier approaches of which the results are given in [10], and will serve as the main point of reference with regard to the performance of blind segmentation algorithms in general. The method proposed by ten Bosch et al. as described in Section 2.1.1.2 was used in earlier research closely related this work [1], and was included to allow comparison with the DP approach.

Table 7.2 presents the DP cost in milliseconds per reference boundary, the percentage insertions and deletions with respect to the number of reference boundaries, the average error, and the R-value for the optimised cases on the development set.

Table 7.2: Performance comparisons on the development set after silence removal.

Method	DP Cost (ms)	INS(%)	DEL(%)	ERR(%)	R-value
DP (C1)	17.22	13.34	24.64	18.99	0.813
DP (C2)	17.94	17.39	26.98	21.19	0.793
DP (C3)	16.11	14.97	24.30	19.64	0.813
DP (C4)	16.84	16.53	23.62	20.08	0.812
Räsänen	18.91	17.92	26.99	22.46	0.787
ten Bosch	24.22	25.15	26.31	25.73	0.755

When applying these parameter values to the core-test set, the results shown in Table 7.3 are obtained.

Table 7.3: Performance comparisons on the core-test set after silence removal.

Method	DP Cost (ms)	INS(%)	DEL(%)	ERR(%)	R-value
DP (C1)	17.90	12.51	25.16	18.83	0.809
DP (C2)	18.20	17.34	27.97	22.65	0.786
DP (C3)	16.43	14.54	25.17	19.85	0.803
DP (C4)	16.78	16.41	23.99	20.20	0.807
Räsänen	19.40	17.18	28.19	22.68	0.778
ten Bosch	24.72	23.77	27.54	25.65	0.751

For illustrative purposes, the segmentations produced by the three algorithms for the same sentence, dr6-fbch0-sa1, are shown in Figures 7.11, 7.12, and 7.13, respectively, where configuration C4 was used for the DP algorithm. Each figure shows the first two seconds of the sentence as well as the TIMIT phoneme boundaries. The dashed vertical lines show the hypothesised boundaries, and the solid vertical lines show the reference boundaries.

The vertical axis in Figure 7.11 for the DP algorithm shows the emission probabilities. Unlike the other two approaches, there is no threshold under which boundaries are ignored. Thus, even when the local score results in a low emission probability, a boundary can be hypothesised if the transition probability is high. This is clear, for example, at the boundaries that are hypothesised at the 'ae', the second 's', and the 'r' phonemes. The converse may also be true, i.e. even when the emission probability is high,

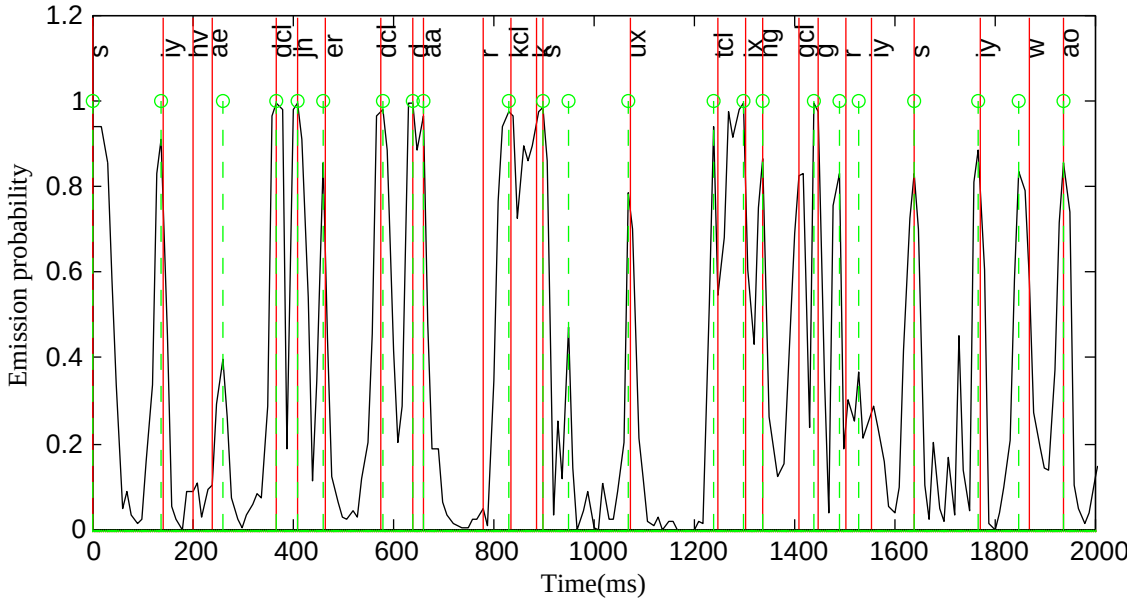


Figure 7.11: Segmentation results for the DP algorithm with C4 on dr6-fbch0-sa1.

a segment boundary may be suppressed by a low transition probability, as illustrated at the transition between 'ng' and 'gcl'.

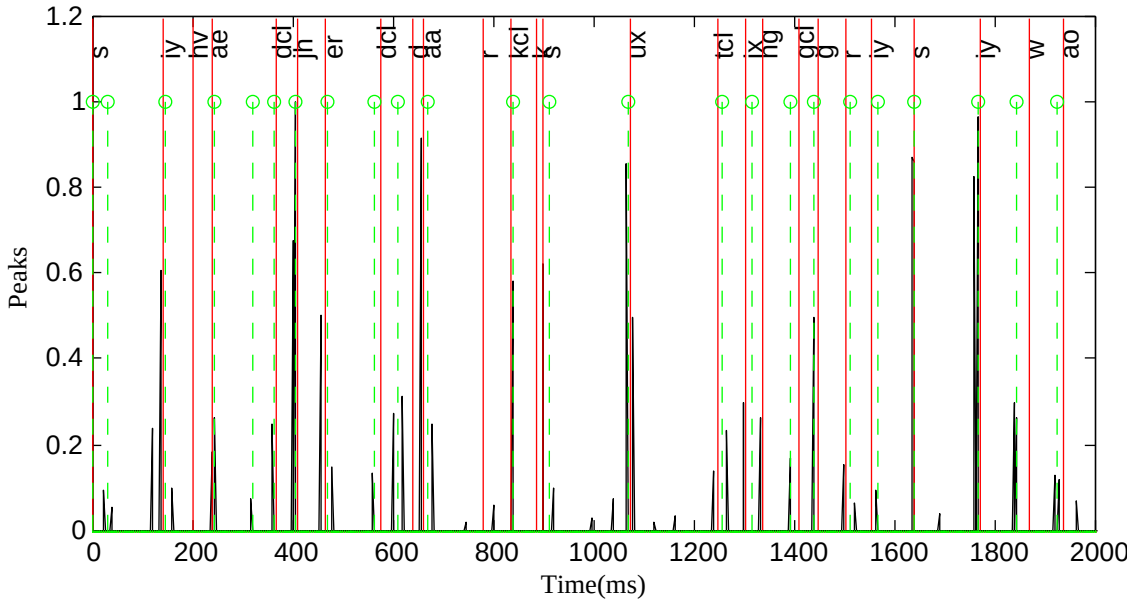


Figure 7.12: Segmentation results for the Räsänen algorithm on dr6-fbch0-sa1.

Figure 7.12 shows the output of the 'minmax' filter described in Section 2.1 of the Räsänen algorithm. Notice that all peaks falling within 32 ms of each other have been combined by temporal peak masking, and that the threshold in this case is 0.07, below which all peaks are ignored. These parameter values were determined to be optimal for the development set.

As proposed in [6], the local score of ten Bosch's algorithm has been multiplied by the log energy in an attempt to reduce the insertion of boundaries in regions of silence, which are characterised by very low energy. The resulting score was then smoothed by a Hanning window. Unfortunately multiplying

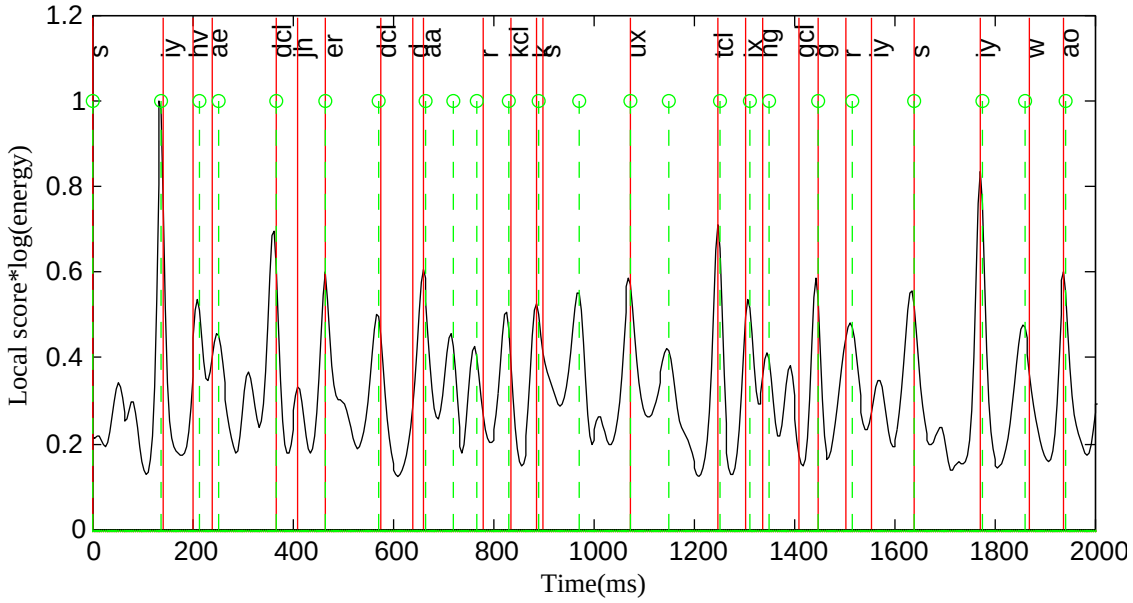


Figure 7.13: Segmentation results for the ten Bosch algorithm on dr6-fbch0-sa1.

by the log energy also causes many peaks that would otherwise have a low score to have a high score, especially those situated in the middle of vowels and other high energy regions. This led to the unwanted insertions at 'aa', the second 's' and 'ux' in Figure 7.13. The result is that the algorithm suffers from over-segmentation, which is clear when looking at the higher percentage insertions in Table 7.3. From the development set it was found that the optimal threshold for the ten Bosch algorithm is 0.4.

7.5 Conclusion

A DP-based segmentation algorithm was evaluated experimentally as an alternative to established threshold driven algorithms used in blind speech segmentation. Since it is common for local scores to show clusters of peaks as illustrated in Section 2.3, algorithms employ ad-hoc remedies to counteract this. For example, even after the min-max filter was applied to the local score in Figure 7.12, clusters of peaks are still present. They are then reduced further by using peak masking to combine peaks that are close to each other. The smoothing applied to the local score of Figure 7.13 is also an attempt to solve this problem, by smoothing small peaks into one. The DP algorithm's output in Figure 7.11 has many small peaks present that do not result in insertions due to the fact that only peaks corresponding to high emission and transition probabilities are selected. This is especially clear at the third 's' in Figure 7.11, where not one of the peaks results in an insertion. This demonstrates the DP algorithm's ability to address the problem of peak clusters without ad-hoc remedies.

By comparing the performances of the DP-based algorithm with the methods serving as benchmarks, it is clear that with the right feature vector and vector distance function combinations (C1, C3, and C4) the DP algorithm leads to much better performance on all accuracy measures. The DP algorithm is therefore a viable and more effective alternative to existing threshold driven approaches to blind speech segmentation.

Chapter 8

Experimental results for MLP-based local scores

In Chapter 7 the experimental results obtained by using blind segmentation were discussed. The local scores pertaining to blind segmentation algorithms are determined by applying vector distance functions to feature vectors, and do not employ trained models. This makes them independent of a specific language or corpus, and widely applicable to different speech data. In this chapter we investigate how a multi-layer perceptron (MLP) can be employed to generate a local score [3]. MLPs require training prior to segmentation, and do therefore not fall into the blind segmentation class. The training procedure was detailed in Section 2.2.1. The first step was to extract feature vectors (MFCC) centred on the phoneme boundaries in TIMIT, and feature vectors centred between two phoneme boundaries. An MLP with two output neurons was trained to let each neuron produce a high value depending on the training example, the first when training on an example of a phoneme boundary, and the second when training on an example of the absence of a boundary. The difference between the first and the second output neurons then serve as the local score.

The following experimental results compare two segmentation approaches that employ an MLP: the MLP-based segmentation algorithm given by Keri et al. [3], as described in Section 2.2, and the DP-based algorithm (Chapter 4) in which the local score generated by the same MLP is embedded.

In the experiments done by Keri et al. [3], linearly weighted mean smoothing was applied to the local score. During our initial experiments it became apparent that smoothing with a Hamming window gave better results. Hence we have computed the average value of the local score using a five-frame long Hamming window. It was also found that better performance can be achieved when only local score values larger than or equal to zero are used in the DP algorithm. This makes sense when one considers that the DP algorithm attempts to find the most probable segment boundaries from all possible boundary locations, and that the MLP was trained to produce a local score larger than 0 when there is a possibility that a segment boundary is present. It would therefore only be detrimental to the performance of the DP algorithm if one allows the DP algorithm to choose segment boundaries located at local score values smaller than 0. Consequently, the local score was restricted to the range 0–1 for the results produced by the DP algorithm.

Table 8.1 compares the performance of the MLP-based segmentation algorithm given by Keri et al. [3]

and the DP-based algorithm when embedded with the local score generated by the MLP. Note that Keri et al. [3] repeated results using different accuracy measures to those used in this thesis, and that the numbers are therefore not directly comparable.

Table 8.1: Comparison of core-test set performance of the MLP-based segmentation algorithm given by Keri et al. [3], and the DP-based algorithm when embedded with the local score generated by the MLP.

Method	DP Cost (ms)	INS(%)	DEL(%)	ERR(%)	R-value
Keri et al.	14.50	12.74	17.00	14.87	0.857
DP	13.57	12.46	17.96	15.21	0.852

The probability weights, as described in Section 6.2.3, were adjusted so that deletion and insertion errors for the different methods are as close as possible to each other in order to facilitate comparison. Table 8.1 shows that the average error for the DP-based segmentation is slightly higher than for the Keri et al. approach. On the other hand, the DP cost was considerably smaller for the DP-based segmentation. This indicates that the DP leads to segment boundaries that are on average more closely aligned to the TIMIT boundaries, but with slightly fewer boundary pairs within 20ms of one another. The R-values are close to each other as expected, because the insertions and deletions were kept close to each other. The slightly larger R-value seen for the Keri et al. algorithm is mainly due to the smaller number of deletions.

During informal evaluation we observed typical errors made by both the two algorithms. We illustrate some of these errors by means of the segmentations produced by both algorithms for the same sentence, dr6-fbch0-sa1, in Figures 8.1 (Keri et al.) and 8.2 (DP). Each figure shows the first two seconds of the utterance. The dashed vertical lines show the hypothesised boundaries, and the solid vertical lines show the TIMIT reference phone boundaries.

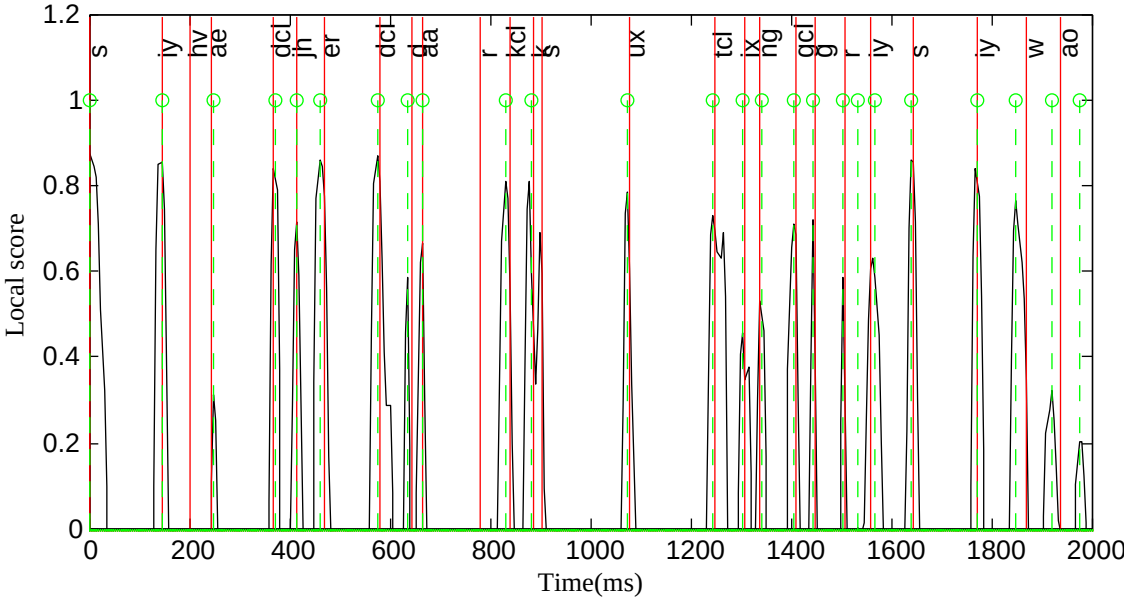


Figure 8.1: Segmentation results for the Keri et al. algorithm for sentence dr6-fbch0-sa1.

For the Keri et al. algorithm, insertions were found to be frequent at boundary regions where local scores had very small amplitudes. The boundary in the middle of the second 'r' in Figure 8.1 illustrates this type of insertion. This algorithm also misses segment boundaries when more than one large peak occurs in a boundary region. The deletion of the boundary at the start of the second 's' illustrates this.

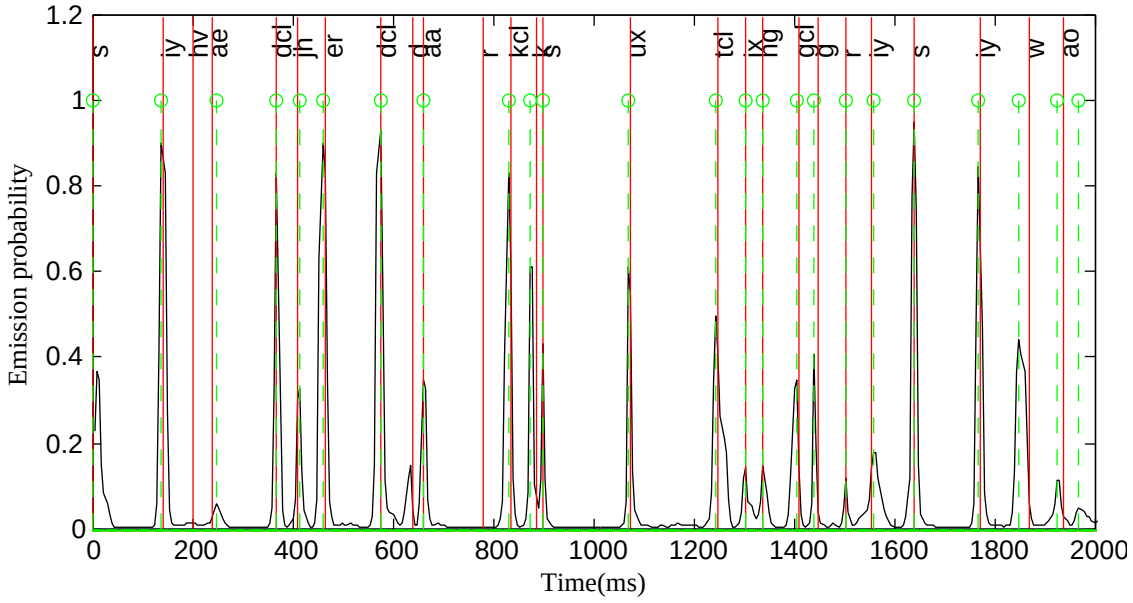


Figure 8.2: Segmentation results for the DP algorithm embedded with the MLP's local score for sentence dr6-fbch0-sa1.

The DP implementation, on the other hand, frequently skips plausible boundaries when they closely precede a boundary with higher probability, leading to a deletion. The deletion near 'd' in Figure 8.2 is an example of such an error.

8.1 Emission probability threshold

The local score probability distributions on the core-test set for the presence and absence of a segment boundary are shown in Figure 8.3. The distributions are shown on the core-test set to give an impression of the general behaviour of the local score free from the bias of the training set. From the experimental results on blind local scores, presented in Section 7.2, it was found that the better the lobes of the distribution representing the presence of a boundary and the distribution representing the absence of a boundary are separated, the better the performance of the local score will be. The main lobes in Figure 8.3 are located at the opposite sides on the local score axis, meaning that the local score in this case can quite accurately distinguish between the presence and the absence of a boundary.

It is apparent that local scores generated by the MLP are very accurate. The tendency of plausible boundaries to be skipped by the DP implementation is therefore detrimental to its performance. This shortcoming is addressed by creating a hybrid approach between the two algorithms. This hybrid approach takes all the boundaries proposed by the Keri et al. algorithm that have an emission probability above a specific threshold, and fixes these before DP is performed. The DP algorithm then

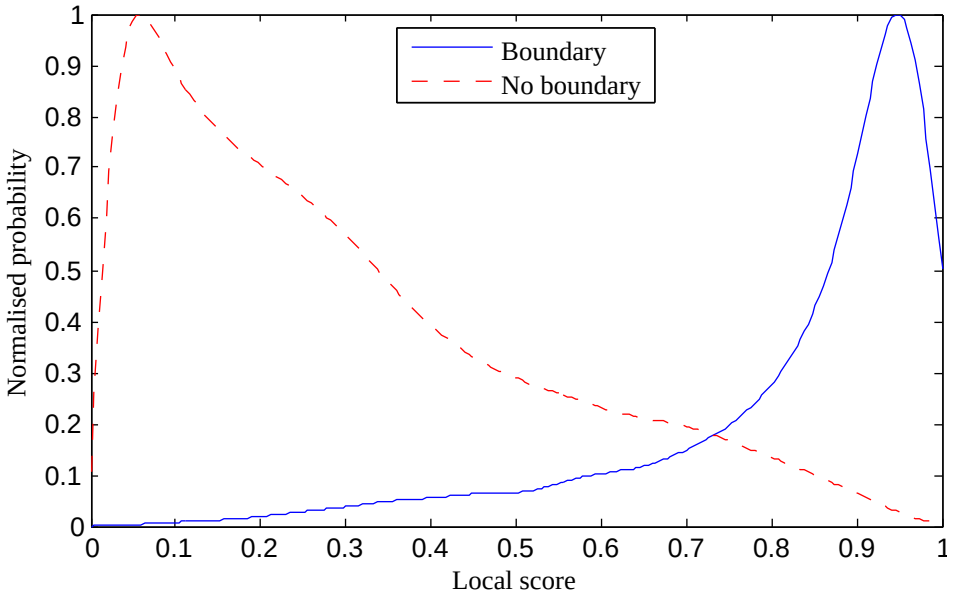


Figure 8.3: Probability distributions of the local score when a segment boundary is present and when it is absent for the MLP-based local score.

has the task of choosing between boundaries hypothesised by peaks with lower emission probabilities as well as between other higher probability peaks that were skipped by the Keri et al. segmenter. Figures 8.4 and 8.5 show the effect of this threshold on the DP cost and the average error respectively.

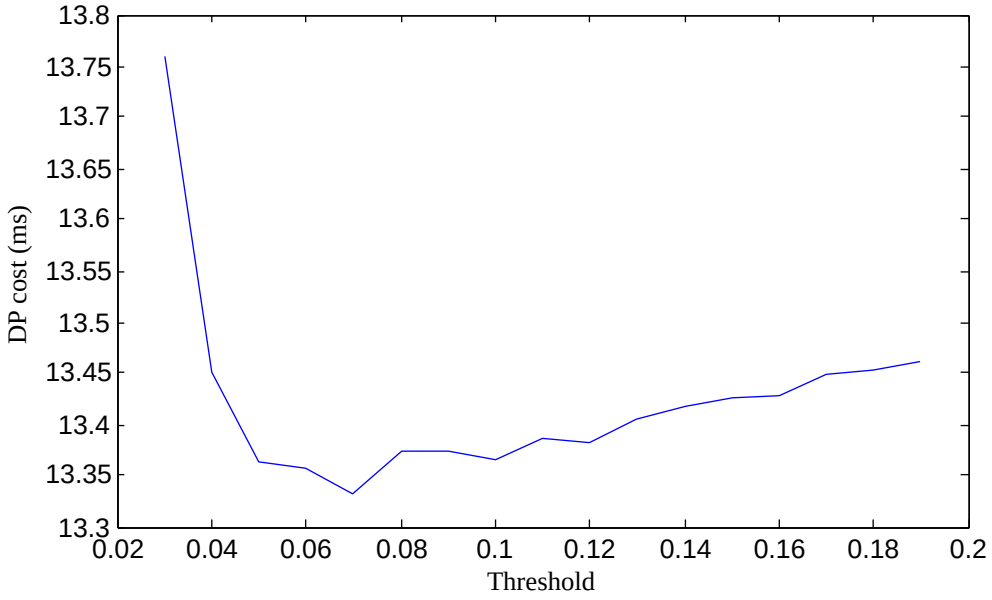


Figure 8.4: DP cost (as described in Section 5.1) as a function of the emission probability threshold for the combined method, measured on the development set.

Figures 8.4 and 8.5 show that segmentation accuracy is improved by including the new threshold. For the optimal threshold, Table 8.2 gives the segmentation performance achieved by the combined method on the core-test set. There is an improvement in terms of both DP cost and average error, indicating that the hypothesised boundaries are better aligned with the TIMIT boundaries and that

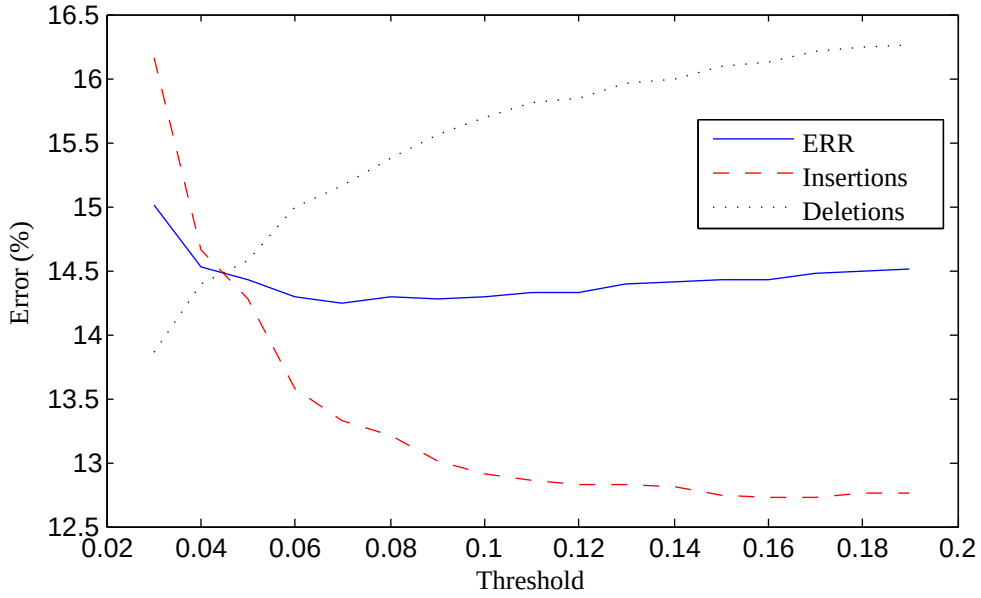


Figure 8.5: Average error (as described in Section 5.2) as a function of the emission probability threshold for the combined method, measured on the development set.

more boundary pairs are within 20ms of one another. The R-value also improves due to a substantial decrease in deletions.

Table 8.2: Comparison of core-test set performance of the MLP-based segmentation algorithm given by Keri et al. [3], the DP-based algorithm when embedded with the local score generated by the MLP, and when the approaches are combined.

Method	DP Cost (ms)	INS(%)	DEL(%)	ERR(%)	R-value
Keri et al.	14.50	12.74	17.00	14.87	0.857
DP	13.57	12.46	17.96	15.21	0.852
Combined	13.20	13.06	15.72	14.39	0.863

The segmentation achieved for sentence dr6-fbch0-sa1 by the combined method is shown in Figure 8.6. Both the insertion at the second 'r' and the deletion near the 'd' have been avoided. While this is merely a specific example, similar improvements were observed informally for many other utterances.

8.2 Conclusion

The DP segmentation approach using local scores generated by an MLP, having a single hidden layer with 30 neurons, was investigated. A better alignment to the reference boundaries was achieved by the DP algorithm than the benchmarked MLP-based threshold approach, although with fewer boundaries within the 20ms margin. It is possible to combine the two methods, however, to avoid some of the simple mistakes made by both algorithms, and this led to the best performance across all accuracy measures.

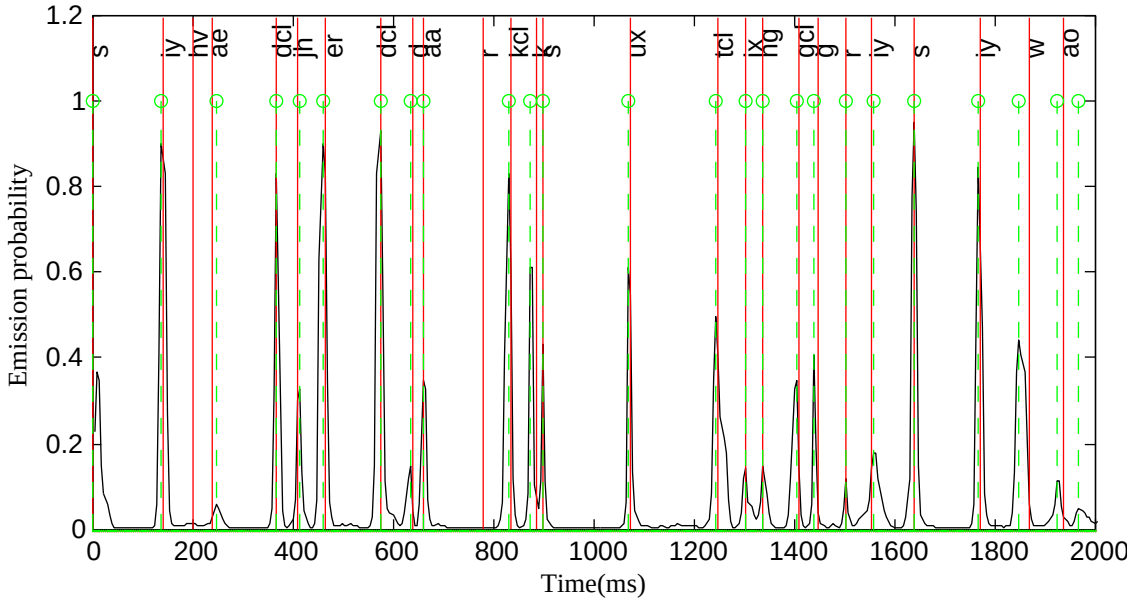


Figure 8.6: Segmentation results for the combined method for sentence dr6-fbch0-sa1.

The DP algorithm did not lead to the same increase in performance relative to the benchmarked method as it did when using blind local scores. We believe that this is due to the more accurate local scores generated by the MLP. It seems that the MLP-based local score rarely exhibits boundary clusters such as were seen for blind local scores. This limits the usefulness of incorporating DP. The DP algorithm attempts to find the optimal segmentation from a number of possible boundary locations with large local scores, but if these locations are already limited to a small number, the DP will be forced to choose most of them as segments. This limits its ability to improve on the conventional threshold-based segmentations.

Chapter 9

Deep neural networks in speech segmentation

9.1 Introduction

In a recent study the potential of deep architectures in machine learning was investigated [25]. The author claimed that the best way for an artificial intelligence (AI) to distinguish between a number of similar inputs is to learn different levels of abstractions (i.e. features) from the raw input data, starting with the basic features at the lower layers, like the edges in an image, to more and more abstract concepts per layer, like the shapes of a face, until a final high-level abstraction can be produced that is used to classify the input.

According to this point of view, the lower layers of networks with deep architectures should be trained to provide an accurate and broad representation of the input data from which the higher layers can detect abstract features [26]. For the case of MLPs, backpropagation is used to train the networks. The partial derivatives of the error function are propagated from the output to the input by using the chain rule for derivatives. These partial derivatives allow the parameters of the network to be trained by gradient descent. Backpropagation is ineffective for deep networks because both the lower layer features and higher layer features are being learned simultaneously. However it is not possible to learn higher layer features if the lower layer features do not yet exist. Backpropagation also functions in such way that the higher layers' parameters are trained more quickly than the lower layers' parameters because the partial derivative of the error function becomes more distorted as the length of the chain of derivatives grows. As a result, MLPs learn most of the patterns used for recognition in the two top-most layers, while the lower layers lack structure [26]. Because the lower layers did not learn basic low-level features, the higher levels will be unable to make effective generalisations regarding the data, and overfitting is likely [25; 26].

Restricted Boltzmann machines (RBMs) have recently become popular as a powerful way to extract features from raw input data. An RBM is a bipartite graph that uses hidden units to detect dependencies (also termed abstractions or features) in the training data through unsupervised training. The training process estimates the joint probability distribution of the training data and hidden variables that will serve as part of a generative model. Feature detection can be interpreted as a process in which

each hidden unit acts as a filter, amplifying certain parts of the input and attenuating others. Several studies have considered the application of RBMs as image filters for object recognition [27; 28; 29; 30]. RBM filters have also been used to learn a better feature representations for speech recognition [31].

RBMs have also proved to be useful building blocks in the creation of deep architectures. In a bottom up process, RBMs are stacked in layers, where each RBM detects dependencies between the features that were detected by the RBM below it. This process is called greedy layer-wise training [32]. In this way low level features are detected by the lower layers, while increasingly abstract features are learned by the higher layers. This approach was first used in the creation of deep belief networks (DBNs) [33], but has been applied to other deep architectures with the later discovery that such unsupervised pre-training of a network greatly facilitates supervised training [32]. The work done in [33], for example, proposed a supervised wake-sleep algorithm that can be applied to a DBN for handwritten digit classification of the MNIST database. Stacked RBMs also pose a solution to the problems associated with backpropagation. By first generating a structured network through greedy layer-wise pre-training and then adding a final layer whose output corresponds to the labels of the classification problem, supervised backpropagation can be used to fine-tune the parameters. This results in more accurate classification because a hierarchy of low to high level features already exists. This approach was also tested on the MNIST handwritten digit database where it was clearly shown that pre-training gives rise to better performing networks than the standard feed-forward networks initialised with random parameters [26]. High accuracy phone classification has also been achieved on the TIMIT corpus by using pre-trained deep neural networks as acoustic models for the HMMs [34]. In addition, auto-encoders with deep architectures can be trained by stacking RBMs [35]. An auto-encoder is a network that is symmetrical about a small middle layer, i.e. the sub-network on each side of the middle layer is equal but in opposite directions. The network is then trained to reconstruct the input at the output layer by using the reconstruction error during backpropagation. In this configuration the middle layer serves as an encoding of the input, while the top half of the network is used for reconstruction.

An in depth investigation intended to establish exactly how this unsupervised pre-training benefits deep architectures was carried out in [26]. Networks with deep architectures have highly non-convex objective functions with many possible local minima in the parameter space. Initialising the network with stacked RBMs restricts the parameters to local maxima in the cost function that lead to better generalisation. The conclusion is that unsupervised pre-training serves as a regulariser [26], an opinion that is in line with the ideas presented by [25].

The work done in [34] showed that very accurate speech recognition on the TIMIT corpus can be obtained when using pre-trained deep neural networks as acoustic models. This led us to wonder if a more accurate speech segmenter on TIMIT can be achieved by following a similar procedure to train the phoneme-boundary models that were discussed in Section 2.2.1. Our work will therefore build on the concepts and procedures discussed in [34]. This work will begin in Chapter 10, but to understand the work presented there it is first necessary to look at a theoretical overview of RBMs in the rest of this chapter.

9.2 Binary Restricted Boltzman Machines

The layers making up a deep belief network are obtained by stacking restricted Boltzmann machines in a process that is termed greedy layer-wise training. Before dealing with deep belief networks, it is therefore necessary to understand the workings of RBM's. An RBM is an energy-based stochastic generative model that can learn a probability distribution from observations (samples) by training the parameters of an undirected bipartite graph, also known as a Markov random field (MRF) [36]. The visible nodes $\mathbf{v} = (v_1, ..., v_m)$ and hidden nodes (latent variables) $\mathbf{h} = (h_1, ..., h_n)$ are illustrated in Figure 9.1, where each node represents a discrete random variable that, for the sake of this explanation, can take on values from a finite set of values, e.g. 1 and 0 for binary variables (continuous variables are discussed in Section 9.5).

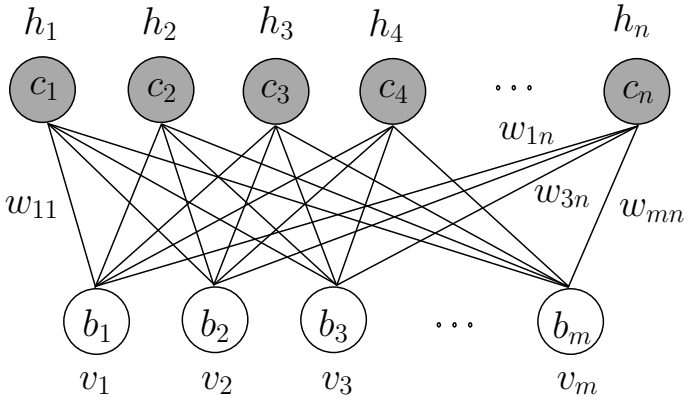


Figure 9.1: Restricted Boltzmann Machine

Henceforth, when referring to the probabilities of nodes having specific values, we will in fact be referring to the probabilities of the random variables associated with those nodes to be equal to those values. Note that there are no visible-to-visible or hidden-to-hidden connections. Each node has an associated real-valued bias indicated by b_i and c_j for the ith visible and jth hidden node respectively. Each undirected connection (edge) between a visible node $v_i \, \epsilon \, \{1, ..., m\}$ and hidden node $h_j \ \epsilon \{1, ..., n\}$ has an associated real-valued weight w_{ij}. These parameters form an integral part of the probability distribution that the graph represents, and it is by adjusting these parameters that the distribution can be made to become more representative of the training data. During parameter training, the visible nodes will correspond to the observations, i.e. training data, and the hidden nodes will detect dependencies (also known as non-linear features [36]) between visible nodes.

A graph can be said to be made up of smaller groups of nodes called cliques, where a clique q in a undirected graph G is defined as a subset of nodes in which every two nodes are connected by an edge. Maximal cliques are cliques to which no node can be added such that the subset remains a clique. Simple illustrations of cliques are shown in Figures 9.2(a) and 9.2(b). Figure 9.2(a) is a graph of three nodes, where the cliques within it are labelled. The cliques numbered 1 to 3 consist simply of two nodes connected by an edge. This structure is the basic building block of all graphs. In addition, all three nodes can also be interconnected, leading to clique number 4. Figure 9.2(b) shows a graph that is made up of four 3-node cliques. Each one of the four 3-node cliques is a maximal clique in this case, because it is not possible to include another node to any one of them for which the resulting subset of nodes will be interconnected.

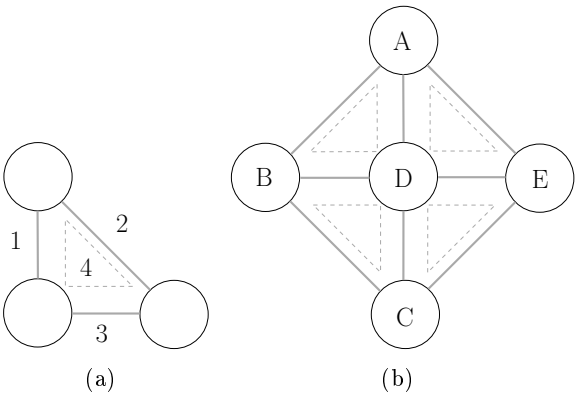


Figure 9.2: (a) A simple illustration of cliques. (b) An illustration of maximal cliques.

In an RBM, every hidden and visible node pair is a maximal clique. The Hammersley-Clifford theorem states that if a multivariate joint probability distribution $p(\mathbf{X})$ can be factorised into a product of strictly positive components on the maximal cliques Q_G of the undirected graph G, the distribution will satisfy the conditional dependencies and independencies given by the graph [36; 37; 38]. These conditional dependencies and independencies are defined according to the Markov property. For example, in Figure 9.2(b) the conditional dependencies are as follows,

$$\begin{aligned} p(A|S) &= p(A|B, D, E) \\ p(B|S) &= p(B|A, D, C) \\ p(C|S) &= p(C|B, D, E) \\ p(E|S) &= p(E|A, D, C) \\ p(D|S) &= p(D|A, B, C, E) \end{aligned}$$

where S in $p(Y|S)$ is the subset of all nodes other than Y. From these it is also clear that A is conditionally independent of C, and B is conditionally independent of E. The general form of such a joint probability distribution is called the **Gibbs distribution** and is shown in Equation 9.2.1, where $\phi_q(\mathbf{X}_q)$ is a strictly positive function called a potential function, and Z is a normalising factor known as the partition function [39]. The variable $\mathbf{X}$ is a vector of values corresponding to the nodes of the graph, and $\mathbf{X}_q$ is a subset these values corresponding to the nodes that are part of clique q. The joint probability $p(\mathbf{X})$ is then the probability that the nodes of the graph will be equal to the values $\mathbf{X}$.

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{q \epsilon Q_G} \phi_q(\mathbf{X}_q) \tag{9.2.1}$$

For an RBM, a joint probability distribution of this form is the **Boltzmann distribution** [38], also known as the **Gibbs measure** [40], and is given by Equation 9.2.2,

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \tag{9.2.2}$$

where Z is the partition function given by Equation 9.2.3. The relationship between Equation 9.2.1 and Equation 9.2.2 becomes clearer when it is known that the exponential function can be factorised

over the maximal cliques (an explanation follows shortly). Note also that instead of using a single vector $\mathbf{X}$ to specify the values of all the nodes in the graph, separate vectors are used to specify the values of the hidden $\mathbf{h}$ and visible $\mathbf{v}$ nodes respectively in correspondence with the layout of the RBM.

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{9.2.3}$$

Associated with each configuration of values for the visible $\mathbf{v}$ and hidden $\mathbf{h}$ nodes of the RBM is a scalar value $E(\mathbf{v},\mathbf{h})$ called the energy as given by Equation 9.2.4. This terminology is borrowed from analogous physical systems [40]. Hence when we refer to a configuration, we will refer to the configuration of hidden and visible node values, and not the model's parameters (weights and biases).

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} v_i h_j - \sum_{i=1}^{m} b_i v_i - \sum_{j=1}^{n} c_j h_j \tag{9.2.4}$$

An energy function of this form is suitable because when it is substituted into the Boltzmann distribution, the exponential function will factorise over the maximal cliques of the RBM as shown in Equation 9.2.5.

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{-\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} v_i h_j - \sum_{i=1}^{m} b_i v_i - \sum_{j=1}^{n} c_j h_j} = \frac{1}{Z} \prod_{i=1}^{m} e^{-b_i v_i} \prod_{j=1}^{n} e^{-c_j h_j} \prod_{i=1}^{m}\prod_{j=1}^{n} e^{-w_{ij} v_i h_j} \tag{9.2.5}$$

By further taking into account the fact that an exponential function is always positive, it is ensured that the joint probability distribution given by Equation 9.2.2 will satisfy the Hammersley-Clifford theorem. Training the model involves the adjustment of the parameters (weights and biases) in a way that lowers the energy for situations in which the visible nodes correspond to a training vector, while increasing the energy for other vectors [41]. Desired configurations will then be situated at valleys of the energy function. The mechanisms behind this procedure will be discussed later on. As mentioned earlier in this discussion, each node v_i and h_j represents a discrete random variable that can take on values from a finite set of values. For the partition function to serve as the normalising factor, all the possible permutations of visible and hidden node values ($\mathbf{v}$,$\mathbf{h}$) that can result from this set of values must be included in the sum ($\sum_{\mathbf{v},\mathbf{h}}$).

Because there are no connections between hidden nodes and between visible nodes, the conditional probabilities associated with individual hidden nodes are independent of one another, as are the conditional probabilities associated with the individual visible nodes. This is expressed in Equations 9.2.6 and 9.2.7.

$$p(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{n} p(h_j|\mathbf{v}) \tag{9.2.6}$$

$$p(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{m} p(v_i|\mathbf{h}) \tag{9.2.7}$$

Binary RBMs have binary stochastic nodes for which the visible and hidden nodes have associated binary values, i.e. $(\mathbf{v},\mathbf{h}) \epsilon \{0,1\}^{m+n}$. Furthermore, the probability of a 1 being associated with a visible or hidden node, given the hidden or visible vectors respectively, is shown to be a sigmoid function (sigmoid(.)) in Appendix A [27; 36]. These conditional probabilities are illustrated in Equations 9.2.8 and 9.2.9 respectively.

$$p(v_i = 1|\mathbf{h}) = \frac{1}{1 + e^{-\left(\sum_{j=1}^{n} w_{ij}h_j + b_i\right)}} = \text{sigmoid}\left(\sum_{j=1}^{n} w_{ij}h_j + b_i\right) \tag{9.2.8}$$

$$p(h_j = 1|\mathbf{v}) = \frac{1}{1 + e^{-\left(\sum_{i=1}^{m} w_{ij}v_i + c_j\right)}} = \text{sigmoid}\left(\sum_{i=1}^{m} w_{ij}v_i + c_j\right) \tag{9.2.9}$$

The marginal distribution of the visible nodes is given by Equation 9.2.10.

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{9.2.10}$$

To increase the probability of an observed or training vector the weights are adjusted by gradient ascent in order to maximise the log-likelihood of the training sample. This is derived in Equation 9.2.11 [25; 36].

$$\begin{aligned}
\frac{\partial \ln p(\mathbf{v})}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}}\left(\ln \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\right) - \frac{\partial}{\partial w_{ij}}\left(\ln \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\right) \\
&= -\frac{1}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \sum_{\mathbf{h}}\left(e^{-E(\mathbf{v},\mathbf{h})}\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}}\right) + \frac{1}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \sum_{\mathbf{v},\mathbf{h}}\left(e^{-E(\mathbf{v},\mathbf{h})}\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}}\right) \\
&= -\sum_{\mathbf{h}}\left(p(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}}\right) + \sum_{\mathbf{v},\mathbf{h}}\left(p(\mathbf{v},\mathbf{h})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}}\right)
\end{aligned} \tag{9.2.11}$$

The third step in Equation 9.2.11 applies Bayes' theorem as shown in Equation 9.2.12.

$$p(\mathbf{h}|\mathbf{v}) = \frac{p(\mathbf{v},\mathbf{h})}{p(\mathbf{v})} = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \tag{9.2.12}$$

The gradient can therefore be expressed as the difference between two expectations: the expectation of the partial derivative of the energy function with respect to w_{ij} under the conditional probability distribution of the hidden nodes given the observed vector and the expectation of the partial derivative of the energy function under the joint probability distribution. Stated in another way, the first term is

the expectation under all possible configurations of hidden node values when it is given that the visible node values are equal to the training vector, and the second term is the expectation under all possible configurations of visible and hidden node values. These expectations are commonly represented by the symbol $\langle\rangle$ as illustrated in Equation 9.2.13 [26; 34], where $v_i h_j$ is equal to $\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}}$.

$$\frac{\partial \ \ln \ p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{Data} - \langle v_i h_j \rangle_{Model} \tag{9.2.13}$$

In this discussion w_{ij} is used to refer to all weights $i \ \epsilon \ \{1, ..., m\}$, $j \ \epsilon \ \{1, ..., n\}$, and it should be taken into account that all weights are updated simultaneously. By maximising the log-likelihood, the energy function's shape is modified. The first term in Equation 9.2.11 descends the energy surface with respect to w_{ij}. The weight w_{ij} is updated with the expected descent of the energy slope under all possible configurations of hidden node values when it is given that the visible node values are equal to the training vector. Accordingly, the average probability of the training vectors will increase, which is clear when decreasing energies are substituted into Equations 9.2.10. Note that the exponential is taken to the power of minus the energy to avoid confusion.

The second term in Equation 9.2.11 ascends the energy surface with respect to w_{ij} leading to an increase in energy. In this case the weight w_{ij} is updated with the expected ascension of the energy slope under all possible configurations of visible and hidden node values. Configurations with low energies are likely to have steep slopes with regard to the weights, and will therefore experience the biggest increase in energy. As a result, the configurations that have low energies will give smaller contributions to the partition function, and other configurations' probabilities will increase, while theirs decrease. Through repeated updates, the energy function's shape will acquire a form that only has low energies at the desired training configurations.

By taking advantage of the independence of the hidden nodes and the fact that the partial derivative with respect to w_{ij} is zero for all weights w_{lk} with $l \neq i$ and $k \neq j$, Equation 9.2.11 can be further simplified to the form shown in Equation 9.2.14 for the case of binary RBMs [36].

$$\begin{aligned} \frac{\partial \ \ln \ p(\mathbf{v})}{\partial w_{ij}} &= \sum_{\mathbf{h}} (p(\mathbf{h}|\mathbf{v}) h_j v_i) - \sum_{\mathbf{v}} \left(p(\mathbf{v}) \sum_{\mathbf{h}} (p(\mathbf{h}|\mathbf{v}) h_j v_i) \right) \\ &= p(h_j = 1|\mathbf{v}) v_i - \sum_{\mathbf{v}} (p(\mathbf{v}) p(h_j = 1|\mathbf{v}) v_i) \end{aligned} \tag{9.2.14}$$

This simplification is achieved by splitting the expectation under the hidden nodes into two sums by factorising over the hidden nodes as shown in Equation 9.2.15.

$$
\begin{aligned}
\sum_{\mathbf{h}}(p(\mathbf{h}|\mathbf{v})h_j) &= \sum_{\mathbf{h}}(\prod_{r=1}^{n} p(h_r|\mathbf{v})h_j) \\
&= \sum_{h_j}\sum_{\mathbf{h_k}}(p(h_j|\mathbf{v})p(\mathbf{h_k}|\mathbf{v})h_j) \\
&= \sum_{h_j}(p(h_j|\mathbf{v})h_j \underbrace{\sum_{\mathbf{h_k}}(p(\mathbf{h_k}|\mathbf{v}))}_{=1}) \\
&= p(h_j = 0|\mathbf{v}) \times 0 + p(h_j = 1|\mathbf{v}) \times 1 \\
&= p(h_j = 1|\mathbf{v})
\end{aligned}
\tag{9.2.15}
$$

The first sum is the expectation of the value of the hidden node j, h_j, and second sum is the sum over the probability distribution of $\mathbf{h_k}$ given $\mathbf{v}$ which is equal to one. Because a binary node can only take on 0 or 1, the expectation of h_j is the probability of h_j having a value of one. This is reflected in the last two steps of Equation 9.2.15.

9.3 Contrastive divergence

Due to the fact that it is exponentially expensive to calculate the second term in Equation 9.2.14 in order to obtain the expectation under the joint probability distribution, an approximate training procedure known as contrastive divergence (CD) is used instead of the exact maximum likelihood learning [25; 27; 34; 36]. CD employs Gibbs sampling, a Markov chain Monte Carlo (MCMC) technique, to sample from the model's distribution. These samples are then used to estimate the expectation under the joint probability distribution. The Gibbs sampling procedure is illustrated in Figure 9.3, where the nodes at the bottom are the visible nodes, and the nodes at the top are the hidden nodes of the RBM.

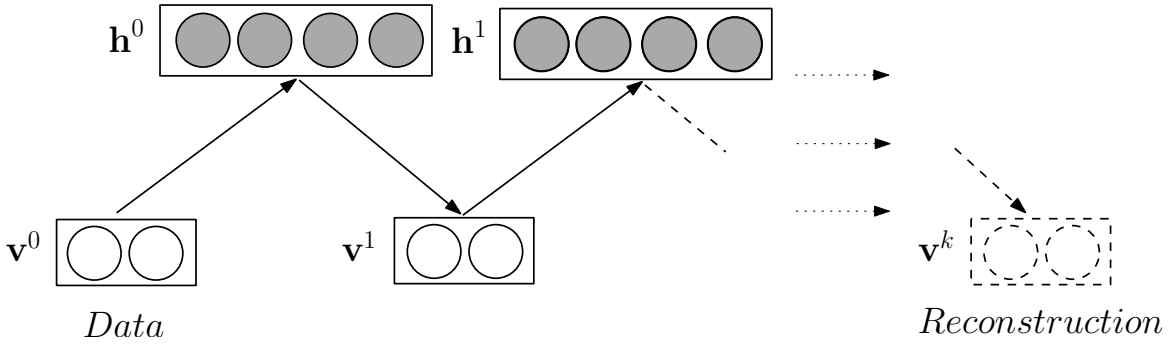


Figure 9.3: Block Gibbs sampling

The Gibbs chain starts at the visible nodes $\mathbf{v}^0$, which correspond to a training vector, from which the hidden nodes $\mathbf{h}^0$ can be sampled using the conditional dependencies given by Equations 9.2.6 and 9.2.9. This is accomplished by comparing the output of the sigmoid function to a random number with uniform

distribution between 0 and 1. If the output is larger than this random value, the node turns on (1), and if not, it turns off (0). The chain is extended by sampling the visible nodes $\mathbf{v}^1$ from the hidden node values $\mathbf{h}^0$ using Equations 9.2.7 and 9.2.8. This in turn allows the hidden nodes to be sampled again to obtain $\mathbf{h}^1$, and so on. For normal Gibbs sampling, usually only one node (random variable) is sampled per step in the Gibbs chain [38]. However, due to the independence characteristics of RBMs it is possible to sample all the hidden nodes or visible nodes together in one sampling step, a procedure which is referred to as block Gibbs sampling.

To obtain accurate samples from the model, the sampling procedure should continue until it converges to a stationary distribution. A drawback of Gibbs sampling is that it is not easy to determine when samples are being drawn from the stationary distribution. To solve this problem convergence diagnostics have been created to give an indication of when convergence has been reached, e.g. the Gibbs stopper, but these are not always accurate and can easily result in stopping the sampling process too early [42; 43]. However, to avoid sampling until the model converges, it has been shown that the second term in Equation 9.2.14 can be approximated by the expectation under the conditional distribution of the hidden nodes given that the visible nodes are equal to the k'th sample in the Gibbs chain $\mathbf{v}^k$, and that this still results in model training [36]. This procedure is called k-step contrastive divergence, and resulting weight update is given by Equation 9.3.1, where Equation 9.2.15 was used to simplify from the second to the last step, and ϵ is the learning rate. It is common to refer to $\mathbf{v}^k$ as the reconstruction of the training vector (data), and this convention was followed in this work. In literature it is also common to use 1-step CD [26; 34], and this approach was followed in our experiments.

$$\begin{aligned} \Delta w_{ij} &\propto \epsilon(\langle v_i h_j \rangle_{Data} - \langle v_i h_j \rangle_{Recon}) \qquad 0 < \epsilon < 1 \\ &= \epsilon(\sum_{\mathbf{h}} (p(\mathbf{h}|\mathbf{v}^0) h_j v_i^0) - \sum_{\mathbf{h}} (p(\mathbf{h}|\mathbf{v}^k) h_j v_i^k)) \\ &= \epsilon(p(h_j = 1|\mathbf{v}^0) v_i^0 - p(h_j = 1|\mathbf{v}^k) v_i^k) \end{aligned} \tag{9.3.1}$$

Using similar derivations to Equation 9.2.11, but with the partial derivative taken with respect to the biases instead of the weights, the resulting update rules for the biases given by Equations 9.3.2 and 9.3.3 can be obtained.

$$\begin{aligned} \Delta b_i &\propto \epsilon(\langle v_i \rangle_{Data} - \langle v_i \rangle_{Recon}) \\ &= \epsilon(v_i^0 - v_i^k) \end{aligned} \tag{9.3.2}$$

$$\begin{aligned} \Delta c_j &\propto \epsilon(\langle h_j \rangle_{Data} - \langle h_j \rangle_{Recon}) \\ &= \epsilon(p(h_j = 1|\mathbf{v}^0) - p(h_j = 1|\mathbf{v}^k)) \end{aligned} \tag{9.3.3}$$

9.4 Bottlenecks, filters and feature detection

During contrastive divergence training, the difference between the training vectors and their reconstructions after k Gibbs samples has, to some degree, been reduced. During this process the hidden nodes are trained to detect suitable features from the training vectors so that these can be accurately reconstructed. A feature in this regard is a specific pattern in a training vector or groups of vectors that allows the RBM to distinguish between this vector or group of vectors in the training set. Ideally the RBM should learn a wide range of exclusive features so that it can accurately reconstruct as many of the different input vectors as possible. Binary hidden nodes are especially suitable for this task, because they can only be on (1) or off (0) during Gibbs sampling. This leads to a bottleneck when only a few hidden nodes are on for a given input vector. Each hidden node is thus forced to detect the most representative features of the input vector in order for the whole input vector to be reconstructed. The bottleneck therefore serves as a regulariser.

The feature detection done by the hidden nodes can be viewed as a filtering process. This is apparent when considering each hidden node as a filter for which the weights connecting it to the visible nodes will either amplify or attenuate the values of the visible nodes. Each hidden node will filter the visible nodes differently, which corresponds to the different detected features.

9.5 Gaussian-Bernoulli RBMs

To accommodate the fact the the input to the final network will be real-valued and not binary, the visible nodes of the RBM at the bottom of the DBN's stack of RBMs will be modelled as Gaussian variables instead of binary variables. Note that only the visible nodes of the RBM at the bottom of the stack will be Gaussian, while its hidden nodes, as well as the visible and hidden nodes of all the other RBMs at the higher layers, will have binary nodes. This is done because binary hidden nodes are better feature detectors, due to the bottleneck effect described previously. An RBM with Gaussian visible nodes and binary hidden nodes is referred to as a Gaussian-Bernoulli RBM (GBRBM), and has the energy function shown in Equation 9.5.1 [27; 34], where σ_i is the standard deviation of the ith visible Gaussian variable.

$$E(\mathbf{v},\mathbf{h}) = \sum_{i=1}^{m} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j=1}^{n} c_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{v_i}{\sigma_i} h_j w_{ij} \tag{9.5.1}$$

The conditional probability of a visible vector given the hidden vector can be derived [27] and is given in Equation 9.5.2.

$$\begin{aligned} p(\mathbf{v}|\mathbf{h}) &= \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\int_{\mathbf{v}} e^{-E(\mathbf{v},\mathbf{h})} d\mathbf{v}} \\ &= \prod_{i=1}^{m} \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{1}{2\sigma_i^2}(v_i - b_i - \sigma_i \sum_{j=1}^{n} h_j w_{ij})^2} \end{aligned} \tag{9.5.2}$$

This is an m-dimensional Gaussian distribution with diagonal covariance as illustrated by Equation 9.5.3,

$$\begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_m^2 \end{bmatrix} \tag{9.5.3}$$

with the mean for the visible node v_i given by Equation 9.5.4.

$$\mu_i = b_i + \sigma_i \sum_{j=1}^{n} h_j w_{ij} \qquad 0 < i \leq m \tag{9.5.4}$$

The probability that a visible node is associated with some real value v, given a hidden vector $\mathbf{h}$, is therefore simply given by Equation 9.5.5,

$$p(v_i = v|\mathbf{h}) = \mathcal{N}\left(v \middle| b_i + \sigma_i \sum_j h_j w_{ij}, \sigma_i^2\right) \tag{9.5.5}$$

where $\mathcal{N}(.|\mu, \sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2 [44]. The probability of a one being associated with a hidden binary node, given a visible vector, remains a sigmoid function, but now the visible node's value is scaled by its variance as shown in Equation 9.5.6 [27].

$$p(h_j = 1|\mathbf{v}) = \frac{1}{1 + e^{-\left(\sum_{i=1}^{m} w_{ij} \frac{v_i}{\sigma_i} + c_j\right)}} = sigmoid\left(\sum_{i=1}^{m} w_{ij} \frac{v_i}{\sigma_i} + c_j\right) \tag{9.5.6}$$

The update rules for a GBRBM are the same as those of a binary RBM, but include a $\frac{1}{\sigma}$ scaling factor, and can also be estimated by contrastive divergence as shown in Equations 9.5.7–9.5.9.

$$\Delta w_{ij} \propto \epsilon \frac{1}{\sigma_i} (\langle v_i h_j \rangle_{Data} - \langle v_i h_j \rangle_{Recon}) \tag{9.5.7}$$

$$\Delta b_i \propto \epsilon \frac{1}{\sigma_i^2} (\langle v_i \rangle_{Data} - \langle v_i \rangle_{Recon}) \tag{9.5.8}$$

$$\Delta c_j \propto \epsilon (\langle h_j \rangle_{Data} - \langle h_j \rangle_{Recon}) \tag{9.5.9}$$

In addition to these parameter updates, the variance σ_i for each visible node must also be updated by using contrastive divergence.

$$\Delta \sigma_i \propto \epsilon \frac{1}{\sigma_i^3} \left(\left\langle (v_i - b_i)^2 - \sigma_i \sum_j v_i h_j w_{ij} \right\rangle_{Data} - \left\langle (v_i - b_i)^2 - \sigma_i \sum_j v_i h_j w_{ij} \right\rangle_{Recon} \right) \tag{9.5.10}$$

9.6 Practical considerations

When training GBRBMs it is necessary to maintain a positive value for the standard deviation. Failing to do so can lead to infinite energy, when $\sigma = 0$, or an ill-defined conditional distribution for the visible nodes, when $\sigma < 0$ [44]. The standard deviation also directly influences the learning of the other parameters, and must therefore be restricted to small changes during updates, because changes that are too rapid can easily cause the other parameters to diverge.

To circumvent the issues of learning the variance, some authors [34] propose the use of a fixed variance, commonly $\sigma_i = 1$. Another simplification is to use the means (μ_i) of the visible nodes, determined by Equation 9.5.4, as the samples instead of sampling from the Gaussian distribution [34] during Gibbs sampling. This is done because the standard deviations are not updated and therefore samples from the Gaussian distribution will be either dominated by noise or are only slightly affected by the standard deviation [44]. These restrictions were applied in the experiments that follow, and therefore all the features in the training data were preprocessed to have zero mean and unit variance before training the GBRBM.

When specifically using RBMs as pre-training for a deep neural network it is better to use sampled values of the hidden nodes during 1-step CD than their probabilities. The samples are taken in the same way as was done for Gibbs sampling. When RBMs are used as feature detectors, it is more important to ensure the bottleneck effect for regularisation than to learn an accurate probability distribution from the training data. By sampling the hidden nodes instead of using the probabilities, this bottleneck effect is further emphasised because only the training parameters that correspond to the hidden nodes that are on are affected during each CD update.

9.7 Deep Belief Networks

To construct a DBN, RBMs are stacked in layers as shown in Figure 9.4. A DBN generates data from the top down, using Gibbs sampling to sample the top RBM's joint probability distribution, and then sampling each layer below in turn until the generated data is produced at the visible nodes of the bottom most RBM. It can be shown that each layer we add to the DBN improves a variational lower bound on the log probability of the training data [33; 34]. The first RBM is trained to generate the input data. After this, the probability of activation for the hidden nodes, generated from the training data, are used as the training data for next RBM. This procedure is called greedy learning [33; 34]. As the RBMs are stacked, more abstract features are being detected by the RBMs at the top. It is from these high level abstractions that a DBN generates the raw data.

9.8 Pre-training in speech segmentation

To convert a top-down generative DBN into a discriminative feed-forward network used for classification, a layer of neurons corresponding to the labels of the classification problem is added to the top of the DBN. The whole network can then be trained as any feed-forward neural network. Previously it has been established that when feed-forward neural networks with many hidden layers are pre-trained

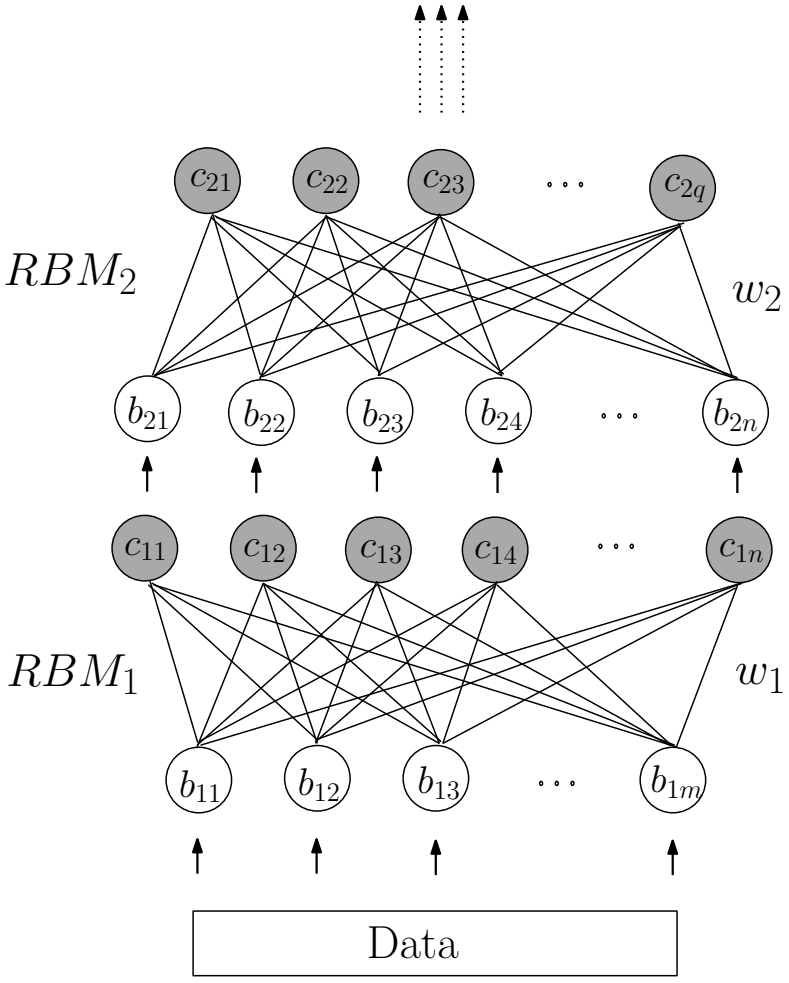


Figure 9.4: Stacking restricted Boltzman machines to create a DBN.

in this way, they will have stronger classification capabilities. This was ascribed to the view that generative pre-training establishes a network with a structure that is already capable of detecting abstract features. The supervised training of the classifier will then be facilitated by this well defined structure and only fine tuning of the parameters is necessary for accurate classification. Neural networks have already been proposed in Section 2.2.1 as a means of generating local scores, but the networks that were used were very simple, consisting of just 30 hidden neurons and a single hidden layer. It may therefore be possible that more accurate local scores can be estimated by a more complex network. The following chapter will investigate this by applying deeper and more complex networks to the speech segmentation task. We will conduct experiments both with and without unsupervised pre-training of the networks in order to establish the benefits of pre-training.

9.9 Summary and conclusion

RBMs are graphical models that can learn the probability distribution of the data on which they are trained. It was shown that by updating a scalar energy function, which forms an integral part of the probability distribution, the model can be trained to increase the probability of a training vector. This is achieved by maximising the log-likelihood of the training vectors by appropriate adjustment of the parameters of the graphical model. However, due to the complexity of the log-likelihood calculation,

it is more practical to employ a procedure known as contrastive divergence. Contrastive divergence uses Gibbs sampling, a Markov chain Monte Carlo technique, to sample from the model's distribution. When the Gibbs chain is initialised on the training vectors, it was shown that the difference between the samples taken at the beginning of the chain and those taken after k steps can be used as an approximate of maximum likelihood learning.

During Gibbs sampling the visible nodes' values will be reconstructed from the hidden nodes, and, after a few epochs of training, the hidden nodes start to detect tell-tale patterns in the training vectors from which they can make more accurate reconstructions. An RBM can therefore be used as a feature detector, and this characteristic will be used to construct deep neural networks with a well defined structure.

DBNs can be constructed by stacking RBMs on top of each other. The bottom-most RBM will be trained to generate the input data, and the RBMs above it will be trained to generate the probability activations of the hidden nodes of the RBM directly below it. The same procedure is used to pre-train feed-forward neural networks. By adding a final layer of neurons, corresponding to the labels of the classification problem, to the top of the network, backpropagation can be used to fine tune the parameters for classification purposes. The use of this pre-training procedure to generate a more accurate local score for speech segmentation is investigated in the next chapter.

Chapter 10

Experimental results using deep neural networks

This chapter presents an experimental evaluation of the application of deep neural networks to the estimation of the local score used by a speech segmentation system. The segmentations produced by networks with different number of layers and hidden neurons, with and without unsupervised generative pre-training will be compared.

10.1 Network architectures

Since a sigmoid function is used to sample values from an RBM's hidden nodes, as shown in Equation 9.2.9, the neural networks that are investigated in this chapter have logistic sigmoid neurons, and not hyperbolic tangent neurons as discussed in Section 2.2.1.1. Furthermore, we employ a single binary output neuron, instead of two output neurons as proposed in Section 2.2.1, because the classification is of two mutually exclusive labels and can therefore accurately be achieved by a single sigmoid neuron. This simplification makes the network simpler and more intuitive. Using the same training data that was used in Section 2.2.1, and using the same frame length and frame skip when extracting features, the output neuron will be trained with a value of 1 for segment boundaries and 0 for non-boundaries. This leads to a local score between 0 and 1, which is scaled linearly to lie between -0.5 and 0.5 to make it appropriate for the segmentation approach proposed by [3] and described in Section 2.2.1.1. Since we wish to focus on the accuracy of the local score that can be achieved with a deep neural network, we will not incorporate the segment lengths by dynamic programming as described in Chapter 4. We will consider networks with between 1 and 5 hidden layers, with each hidden layer having the same number of neurons (256, 512 or 1024). For each architecture, two networks are prepared: one with pre-training and the other without. The subsequent supervised training by backpropagation employs early stopping to find the best performing network on the development set, after which the network's performance is evaluated on the core-test set.

10.2 Early stop training

Early stopping is a technique used during supervised training of MLPs to avoid overfitting [26; 34]. After each training epoch the network's development set performance is compared to the previous epoch's performance. When the performance deteriorates, the learning rate is halved [34]. Training then continues from the previous epoch's weight values. In our application we found that the performance measure tends to alternate slightly between epochs, while still improving in the longer term. For this reason the learning rate is only halved when the performance drops consistently over 5 trial epochs.

Key to the effectiveness of early stopping is a performance measure that can accurately detect overfitting. It was observed during our experiment that, at the beginning of training, both the insertions and the deletions are many. As training progresses, insertions and deletions decrease until a point is reached at which the insertions begin to increase while the deletions continue to decrease. During informal testing, we observed that, when the insertions begin to strongly outnumber the deletions, the performance on the core-test set seriously degrades indicating that overfitting takes place. We therefore required a performance measure that would indicate when the number of insertions was close to the number of deletions. An illustration of the behaviour of insertions, deletions and the R-value (Section 5.3) during a typical early stop training session is shown in Figure 10.1. At epoch 7 the R-value has reached a maximum, and the number of insertions is close to the number of deletions, indicating that overfitting has probably been avoided.

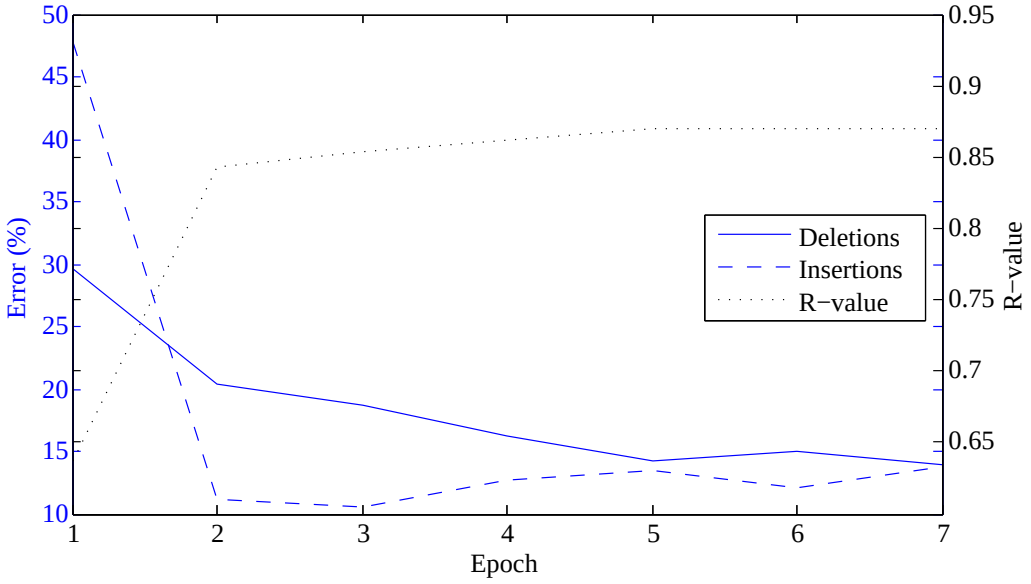


Figure 10.1: Behaviour of insertions and deletions during early stop training.

A large R-value indicates not only that the segmentation accuracy is good, but also that the number of insertions is close to the number of deletions. For this reason it has been found to be a more successful means of avoiding overfitting than the DP cost (Section 5.1) or average error (Section 5.2), which are both independent of the ratio of insertions to deletions.

10.3 Training parameters

The training parameters were chosen to be similar to those used to train acoustic models for speech recognition [34]. For unsupervised pre-training, each RBM was subjected to 50 training epochs, i.e. 50 epochs of unsupervised training per layer of the DBN, at a learning rate of 0.005 for the Gaussian Bernoulli RBM and 0.05 for the binary RBMs, and with a momentum of 0.9.

Early stop training starts with a learning rate of 0.1 and momentum of 0.9, and continues until the learning rate is smaller than 0.01, similar to the approach in [34]. Stochastic gradient descent and unsupervised pre-training used mini-batches consisting of 128 training samples [34].

10.4 Features

The two features that were investigated are listed below:

- MFCC: 12 MFCCs and log energy
- MFCC+Δ+$\Delta\Delta$: 12 MFCC and log energy with appended first and second derivatives

These features were extracted as described in Section 2.2.1. An input vector to the NN consists of 11 consecutive frames centred on the test frame [3]. Hence the NN input vectors have 143 and 429 components for the MFCC and MFCC+Δ+$\Delta\Delta$ features respectively. Frames are 10ms long and extracted every 5ms as proposed by Keri et al [3]. Different configurations of frame length and skip could not be investigated due to time constraints.

10.5 Software and hardware employed during model training

Training the many parameters contained in large neural networks is computationally very demanding and time consuming. To expedite the training of the parameters, parallel processing was employed. The CUDA library made available by the NVIDIA corporation is a very powerful tool that can be used to this end. We employed the GEFORCE GTX650Ti graphics card that contains 768 CUDA cores through the use of the GPU processing capabilities of MATLAB.

10.6 Experiments

10.6.1 Experiments using MFCC features

All the networks discussed in this section were trained to produce local scores, which were then incorporated into the approach described in Section 2.2.1.1 to segment audio. Whether pre-training was applied or not, every network architecture was subjected to three early stop training sessions, from which a mean performance was calculated. This average was taken to indicate typical performance.

Since training a network with many hidden layers and neurons is very computationally intensive and therefore slow, only three early stop sessions were feasible.

Performance in terms of the R-value using networks with 256 neurons per hidden layer are shown in Figure 10.2, while Figures 10.3 and 10.4 show performance in terms of the DP cost and average error respectively. To facilitate comparison between networks with and without pre-training, the core-test set R-values are shown for both cases on the same graph. From the graphs it is clear that a network with the best performing R-value is likely also to have both the best performing DP cost and average error. For example, the performance of the 4 layer network with pre-training performed the best over all performance measures. We conclude that the R-value is a viable indicator of the overall performance of a network.

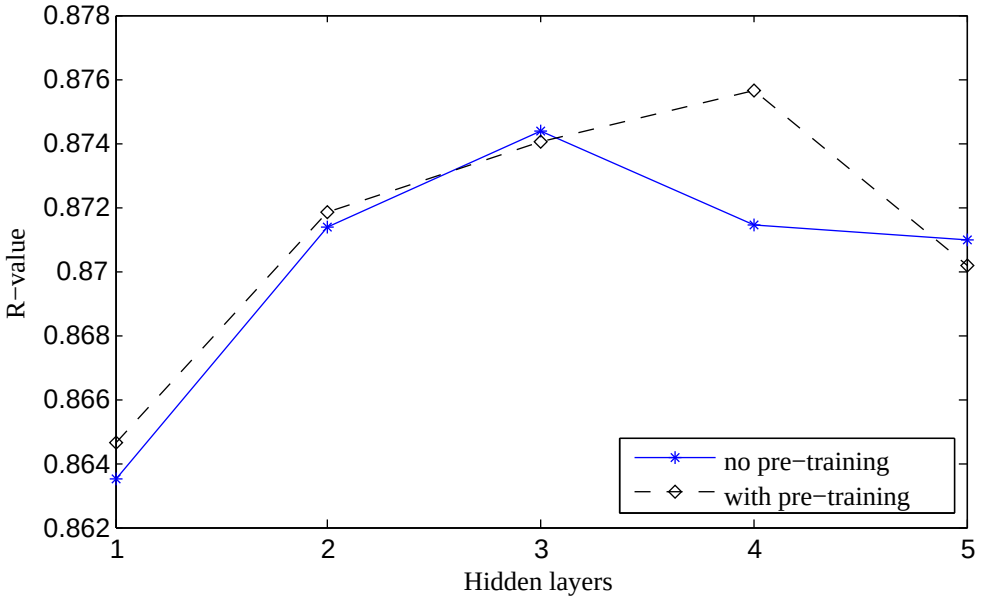


Figure 10.2: R-value performance for a deep NN with 256 hidden neurons per layer and using MFCC as input parameterisation.

Two trends regarding the performances of different network architectures are observable from Figures 10.2 to 10.4. By comparing the performance of the networks with a single hidden layer to the results shown in Table 8.1, which shows the performance of a network with a single hidden layer and 30 hidden neurons, it is apparent that networks with more neurons per hidden layer perform better. This corresponds to the intuition that the more hidden neurons there are, the greater the capacity of a network, and the more accurately it can classify the data. It should be noted that when training networks with many hidden layers and neurons, the use of early stop training is necessary to avoid overfitting. This trend was investigated further by considering the R-value performance of networks with 512 and 1024 neurons per hidden layer respectively (Figures 10.5 to 10.6). For networks with a single hidden layer, a small increase in performance results when the number of neurons per hidden layer is increased from 256 to 512, and from 512 to 1024. For more than one hidden layer the performance does not always improve when the number of hidden neurons is increased. In general, however, it was still likely that an improvement would be achieved. For example, the best performance in terms of the R-value of the networks with 256 and 1024 hidden neurons respectively, and without pre-training, is 0.874 and 0.876, both for 3 hidden layers.

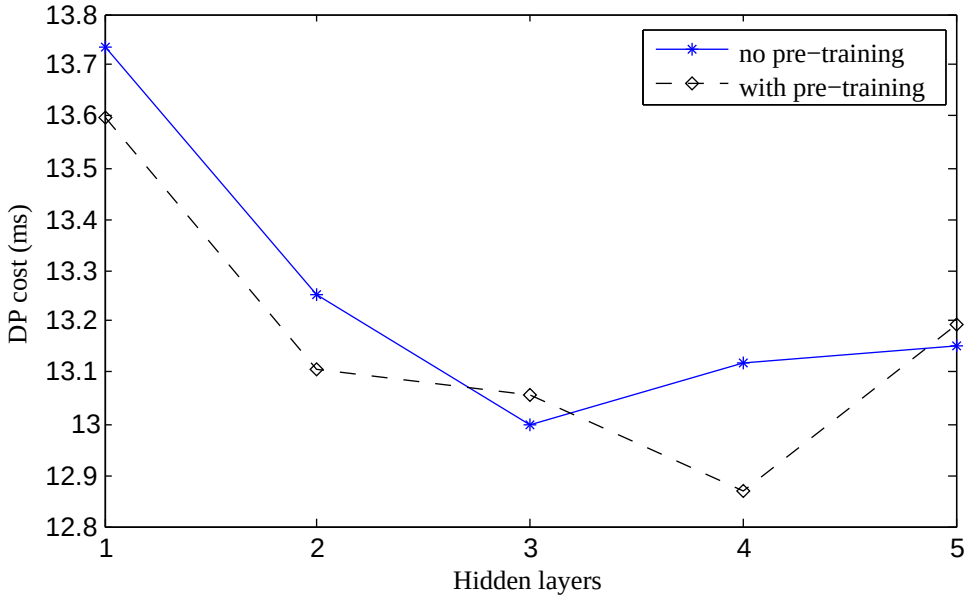


Figure 10.3: DP cost performance for 256 hidden neurons per layer using MFCC as input parameterisation.

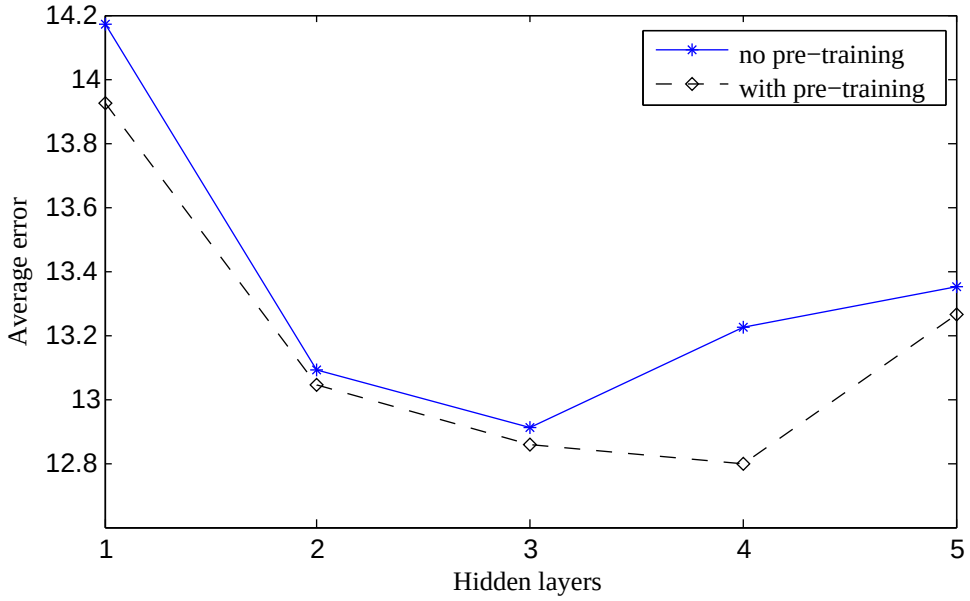


Figure 10.4: Average error performance for 256 hidden neurons per layer using MFCC as input parameterisation.

When moving from a single hidden layer to two hidden layers, a notable improvement in performance can be seen in the graphs. This is caused by the sudden increase in capacity and flexibility of having two fully connected hidden layers adjacent two each other which enable more accurate classifications and therefore also better segmentation performance [34]. On the other hand, although a better performing network is often achieved, having more than 2 hidden layers does not guarantee improved performance.

Applying pre-training to the networks using MFCC as input parameterisation did not reliably lead to an improvement in performance. For example, in Figure 10.2 it can be seen that pre-trained networks with 4 hidden layers and 256 hidden neurons did result in an improvement, but also that the performance of networks with and without pre-training was very similar for different numbers of layers. Furthermore, while the pre-trained networks with 512 neurons per hidden layer did perform better on the core-test

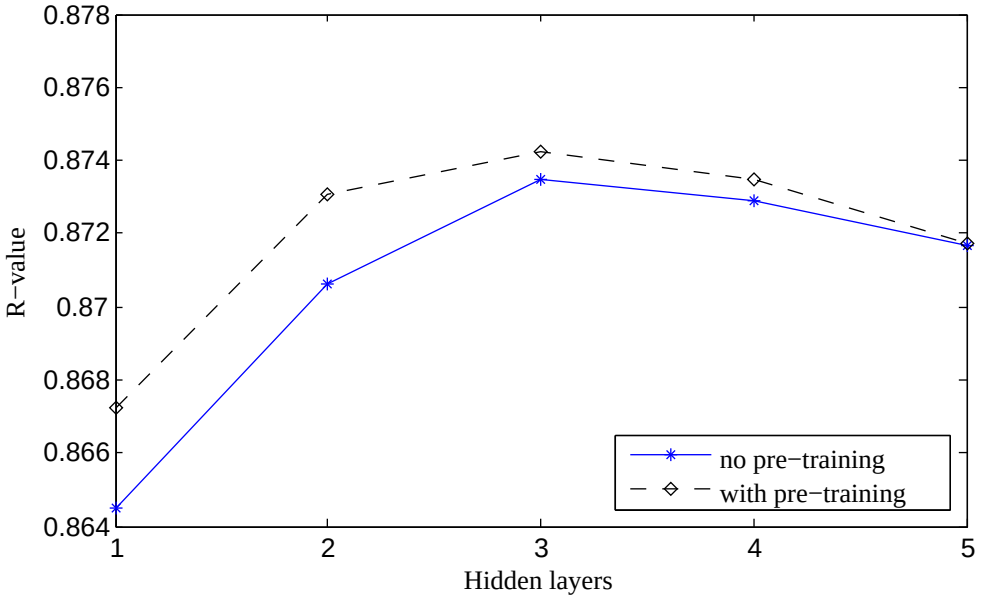


Figure 10.5: R-value performance for 512 hidden neurons per layer using MFCC+Δ+ΔΔ as input parameterisation.

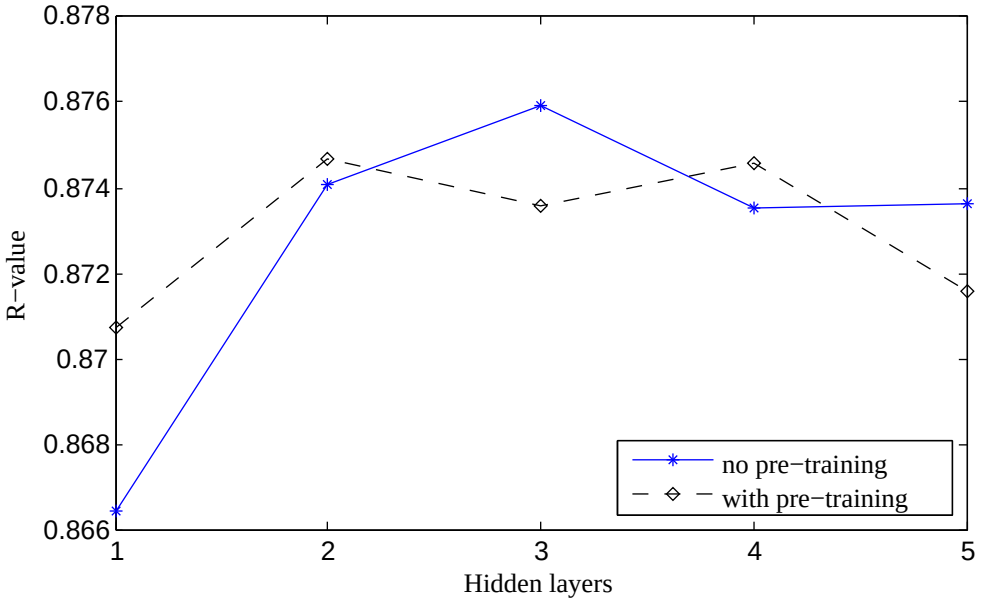


Figure 10.6: R-value performance for 1024 hidden neurons per layer using MFCC+Δ+ΔΔ as input parameterisation.

set, it was found to perform worse on the development set. Similarly, for the networks with 1024 neurons per hidden layer shown in Figure 10.6, performance was usually worse when pre-training was applied. The reason for this will be discussed in Section 10.6.3. It is not possible, however, to conclude from these results that pre-training reliably leads to improved performance.

10.6.2 Experiments using MFCC+Δ+ΔΔ features

Performance in terms of the R-value obtained when training the networks on the MFCC+Δ+ΔΔ features are illustrated in Figures 10.7 to 10.9, where 256, 512, and 1024 hidden neurons are used per hidden layer, respectively. The same trends seen for the MFCC features regarding the network

architectures are apparent. Improvements are seen when increasing the number of hidden layers from 1 to 2, and also when increasing the number of hidden neurons per layer.

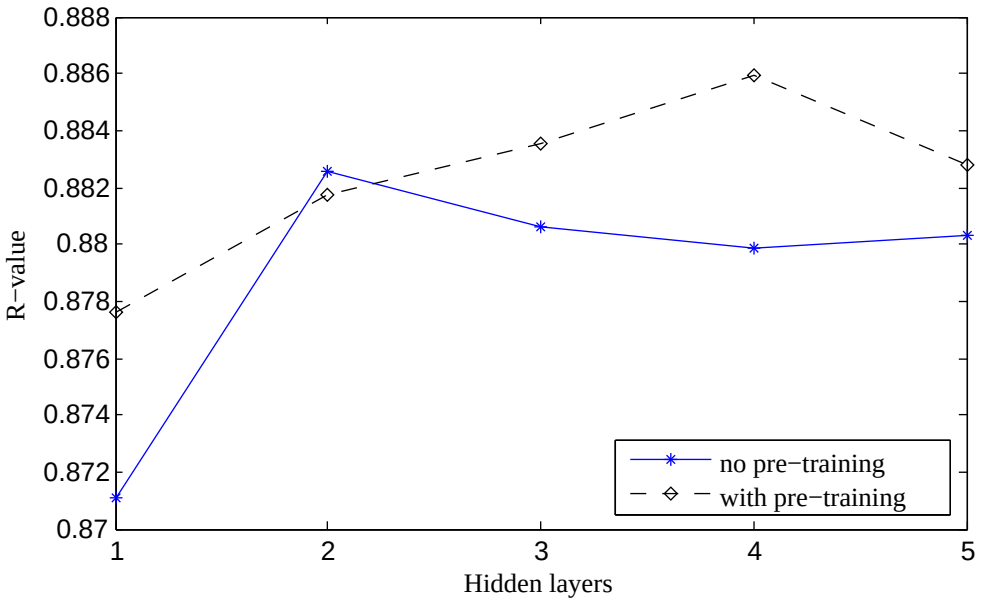


Figure 10.7: R-value performance for 256 hidden neurons per layer using MFCC+Δ+ΔΔ as input parameterisation.

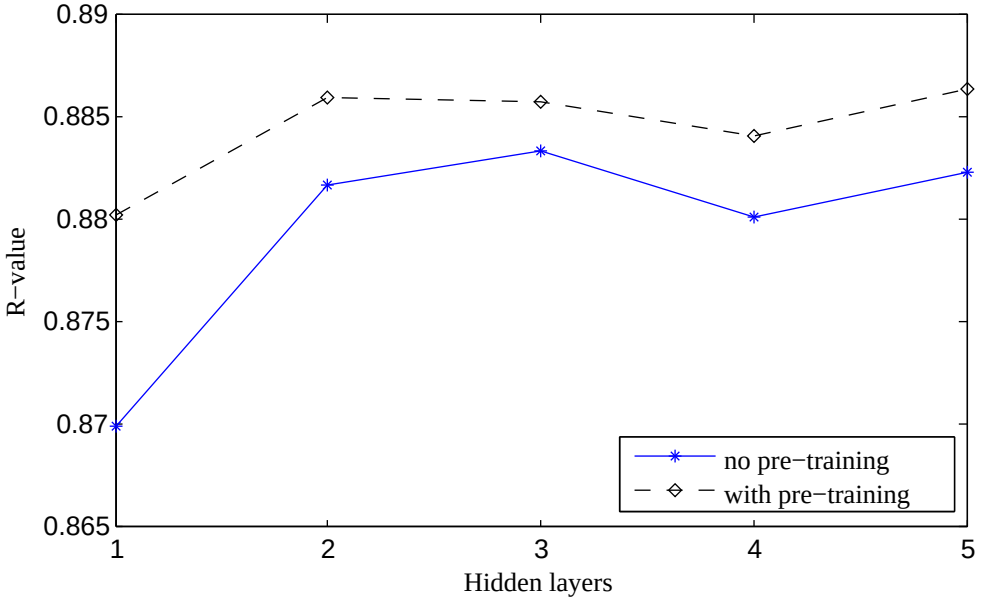


Figure 10.8: R-value performance for 512 hidden neurons per layer using MFCC+Δ+ΔΔ as input parameterisation.

The addition of the derivatives to the MFCC was found to substantially improve the segmentation accuracy achieved by the networks. This is apparent when comparing the best R-value achieved when training on the MFCC features to that achieved when training on the MFCC+Δ+ΔΔ features. For example the best R-value for the networks trained using the MFCC features is 0.876 (Figure 10.6) and was achieved using 3 hidden layers and 1024 hidden neurons per layer. For the MFCC+Δ+ΔΔ features, the best performance is 0.888, and was obtained by the pre-trained network with 5 hidden

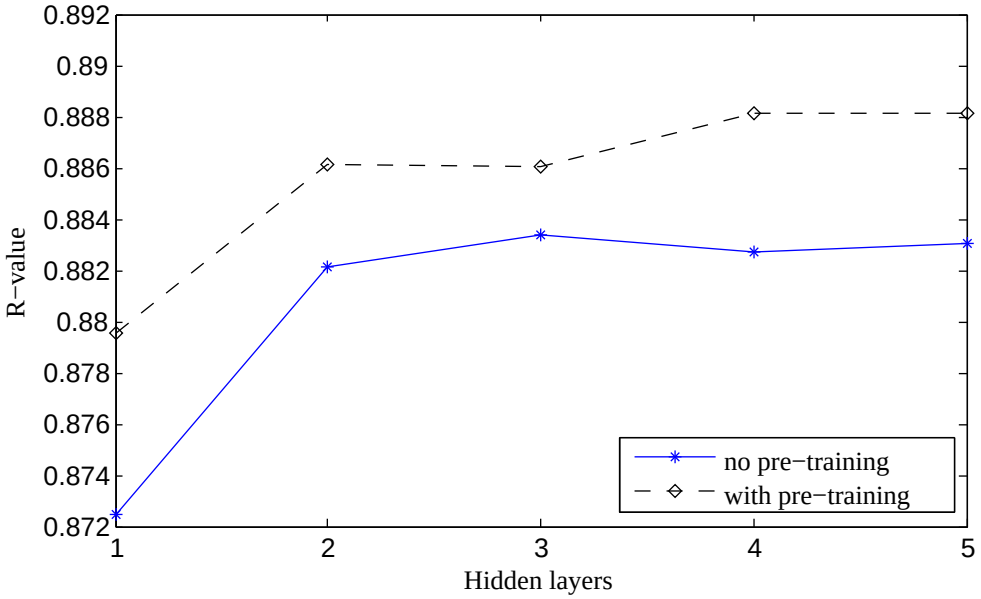


Figure 10.9: R-value performance for 1024 hidden neurons per layer using MFCC+Δ+ΔΔ as input parameterisation.

layers and 1024 hidden neurons (Figure 10.9). Contrary to what was seen for the MFCC parameterisation, it is very likely that pre-training leads to an improvement when using the MFCC+Δ+ΔΔ. This is true for all three accuracy measures (R-value, DP cost and average error) as illustrated by Figures 10.9, 10.10 and 10.11.

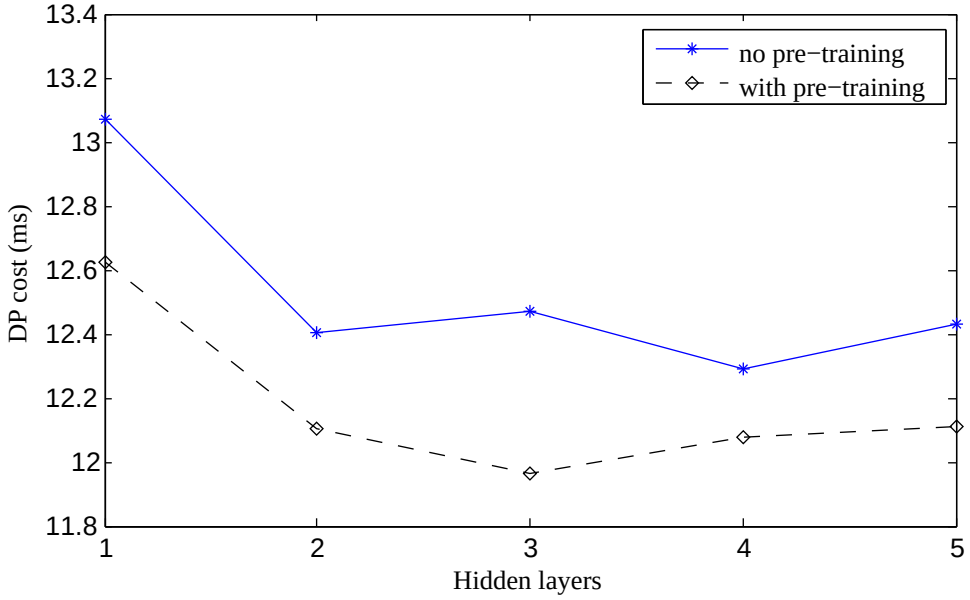


Figure 10.10: DP cost performance for 1024 hidden neurons per layer using MFCC+Δ+ΔΔ as input parameterisation.

10.6.3 Conclusions regarding pre-training

Pre-training the networks led to contradictory behaviour for the two different feature types. Applying pre-training to networks that used the MFCC features did not give conclusive evidence of better per-

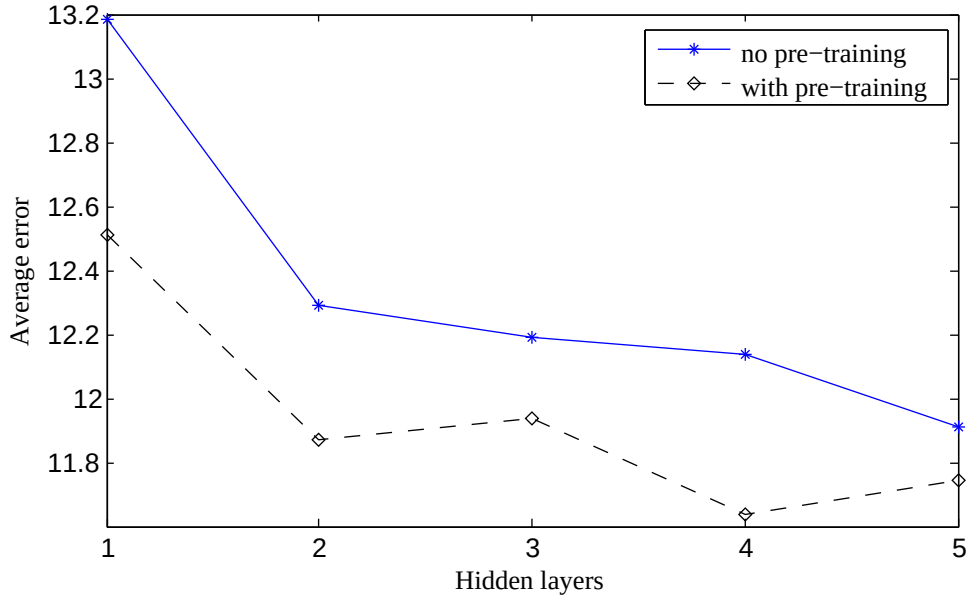


Figure 10.11: Average error performance for 1024 hidden neurons per layer using MFCC+Δ+ΔΔ as input parameterisation.

formance. On the contrary, deteriorated performance was sometimes seen. However, the application of pre-training to networks using the MFCC+Δ+ΔΔ features did generally result in better performance.

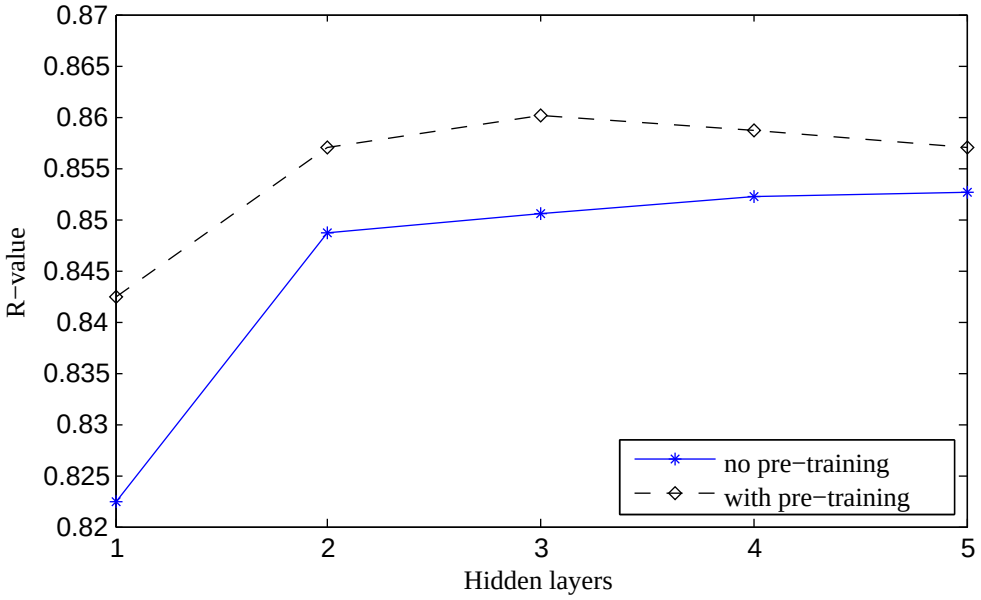


Figure 10.12: R-value performance for 512 hidden neurons per layer using 38 MFCCs and log energy as input parameterisation.

It was suspected that the size of the input vector may be responsible for the different effect of pre-training. To investigate this hypothesis, a new feature vector using 38 MFCCs and log energy (instead of 12 MFCCs and log energy) was introduced. The core-test set performance obtained when using this larger MFCC vector with 512 and 1024 hidden neurons per hidden layer is shown in Figures 10.12 and 10.13 respectively. In both cases pre-training resulted in an improvement, although overall performance was worse than when using the 12 MFCCs and log energy. It appears therefore that pre-training is more beneficial when higher dimensional feature vectors are used. We believe that

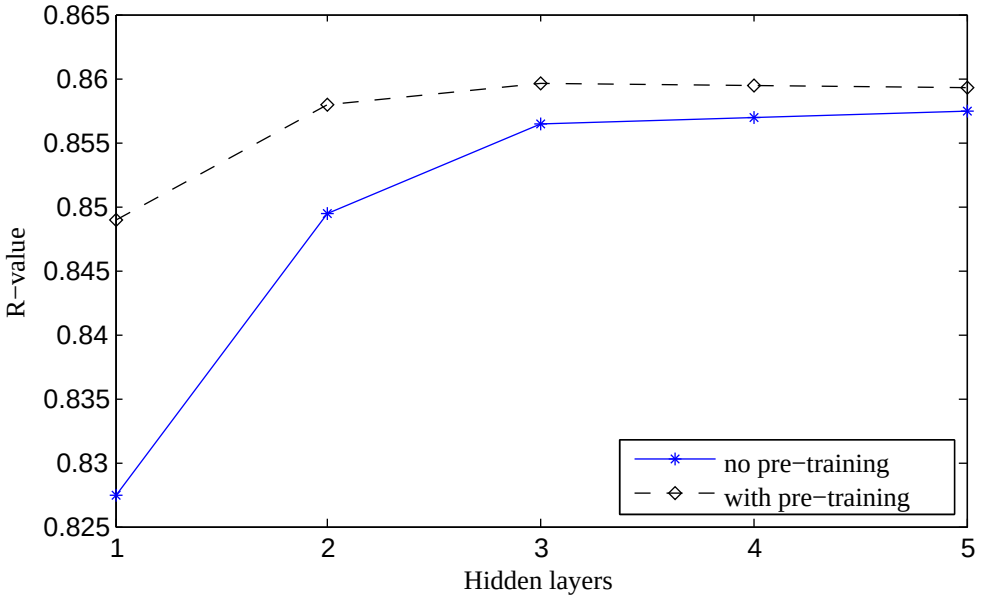


Figure 10.13: R-value performance for 1024 hidden neurons per layer using 38 MFCCs and log energy as input parameterisation.

this can be explained by considering the feature-extraction mechanism used by an RBM. During pre-training, the RBMs will try to extract features according to patterns observed in the training vectors. If the input vectors have a low dimensionality the number of patterns that can be detected by the RBMs might be more limited than for a richer feature set. If an RBM has many hidden nodes and the input vector has low dimensionality, it is very likely that some of the detected features are spurious and do not generalise, i.e. overfitting takes place. Such a network may produce spurious classification decisions in the form of inaccurate local scores.

This agrees with what is seen in Figures 10.2 to 10.6. When the number of hidden neurons per hidden layer is relatively small (256), the performance obtained after pre-training is better or at least very close to the performance obtained without pre-training. But when the number of hidden neurons per hidden layer is very large (1024), the pre-trained networks are more likely to perform poorly relative to the randomly initialised networks.

The use of pre-training on the networks that were trained on the MFCC+Δ+$\Delta\Delta$ parameterisation did result in a notable improvement in performance. For illustrative purposes, the core-test set performance of the networks with best performance on the development set, with and without pre-training are shown in Table 10.1. The performance of the network that was investigated in Chapter 8, which had a single hidden layer and 30 hidden neurons, is included to be compared with the deeper networks.

The segmentation of the sentence dr6-fbch0-sa1 produced by the network without pre-training is illustrated in Figure 10.14, and the segmentation produced by the pre-trained network on the same sentence is illustrated in Figure 10.15. Both produce a more accurate segmentation than the network with a single hidden layer and 30 hidden neurons as illustrated in Figure 8.1. The larger networks correctly detect the boundary at the end of the 'hv', while the simple single layer network deleted this boundary. They also do not insert the extra boundary at the second 'r'. On the other hand, the boundary at the beginning of the first 'r' phoneme was skipped by all three. Furthermore, the pre-trained network was able to detect the boundary at the end of 'kcl', while the network without pre-training was not.

Table 10.1: Segmentation performance of deep networks, with and without pre-training. The pre-trained network consisted of 5 hidden layers with 1024 neurons per layer, while the network without pre-training consisted of 3 hidden layers with 1024 neurons per layer. The performance of a network with a single hidden layer and 30 hidden neurons, which was investigated in Chapter 8, is also included.

Method	DP Cost (ms)	Ins(%)	Del(%)	ERR(%)	R-value
Single hidden layer network	14.50	12.74	17.00	14.87	0.8570
Deep network without pre-training	12.50	11.17	12.73	11.95	0.8865
Deep network with pre-training	11.98	10.14	12.66	11.40	0.8911

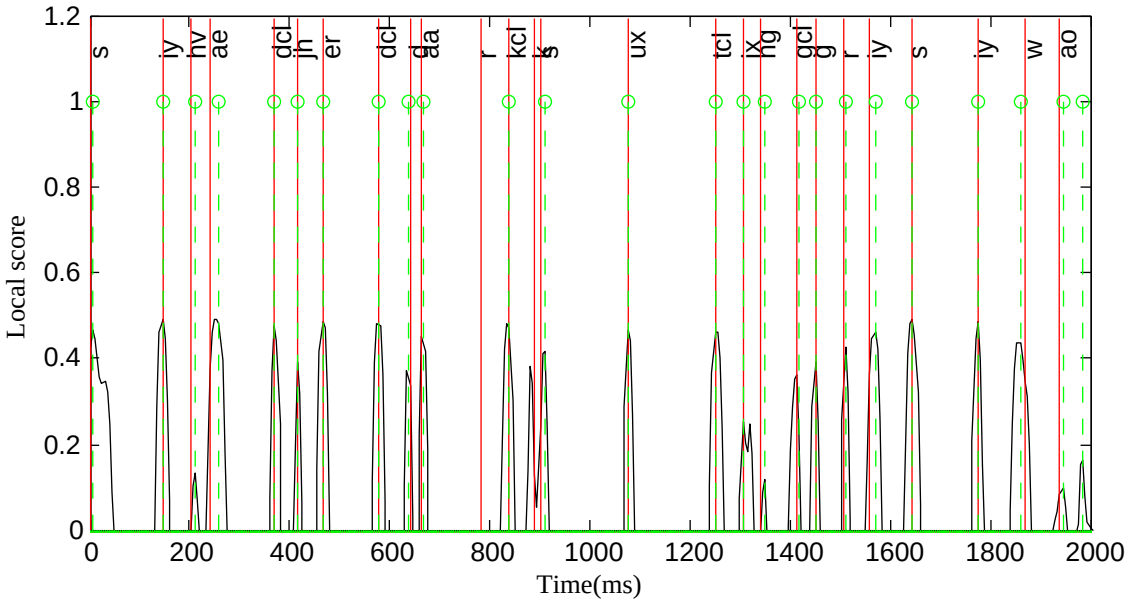


Figure 10.14: Segmentation results for the randomly initialised network, for sentence dr6-fbch0-sa1.

10.6.4 Deep networks in DP-based segmentation

In Chapter 8 we found that the DP-based segmentation algorithm, described in Chapter 4, becomes less effective when a very accurate local score is used. This will be investigated further by using the DP algorithm with the local scores produced by the best performing deep network. The results obtained on the core-test set when using the DP approach, the threshold-based approach, and the combination of the DP and threshold approaches, as described in Section 8.1, are shown in Table 10.2.

Table 10.2: Segmentation performance using the local score of the best performing deep network when employed by the DP approach, the threshold approach, and a combination of the DP and threshold approaches.

Method	DP Cost (ms)	Ins(%)	Del(%)	ERR(%)	R-value
DP approach with deep network	12.17	13.12	14.05	13.59	0.8684
Threshold approach with deep network	11.98	10.14	12.66	11.40	0.8911
Combined method with deep network	11.72	13.96	10.69	12.32	0.8782

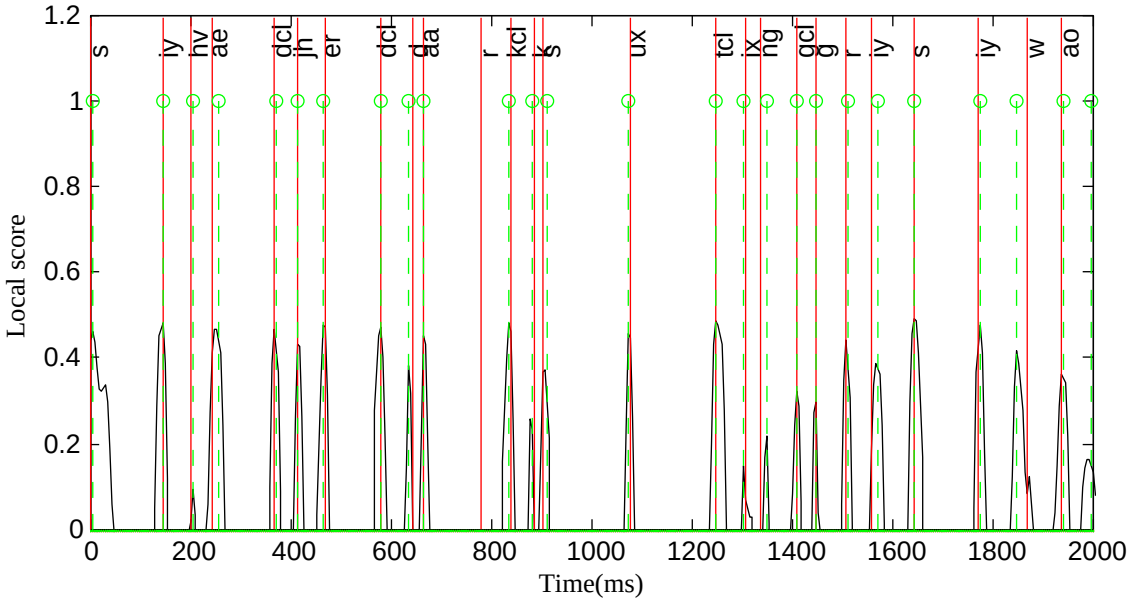


Figure 10.15: Segmentation results for the pre-trained network, for sentence dr6-fbch0-sa1.

The DP approach performed poorly across all performance measures in comparison to the threshold-based segmentation approach. This shows that the more certain you are of boundary positions, the less effective the segment length measurement, which the DP incorporates, becomes. When considering the combined DP and threshold approach, a substantial decrease in deletions and a slightly smaller DP cost is observed, but at the expense of a large increase in the number of insertions in comparison to the threshold approach. Therefore, the combination of the threshold and DP approaches also did not yield a convincing improvement.

10.6.5 Summary and conclusion

The application of deep neural networks to the estimation of the local score used by a speech segmentation system was evaluated. It was found that the accuracy of the local scores increases as the neural networks grow in size, i.e. more neurons per layer or more layers. Additionally, the influence of generative pre-training by RBMs on the local scores of the neural networks was investigated. Two different behaviours were observed for two different feature vectors. When 12 MFCCs and log energy were used to train the networks, the results were inconclusive as to whether pre-trained networks lead to a more accurate local score. On the other hand, when the first and second derivatives were appended, the pre-trained networks convincingly outperformed the randomly initialised networks. An investigation found that input vectors with high dimensionality produce more accurate local scores in networks that were pre-trained than in those that were not. This led us to suspect that overfitting may take place for lower- dimensional input vectors during pre-training, because their limited content causes the RBMs to detect spurious features. This demonstrates that the more information you supply at the input of the networks, the more beneficial generative pre-training is. Finally, the best performing pre-trained and randomly initialised networks were compared, and a substantial improvement was observed for the pre-trained network using 12 MFCCs, log energy and delta features as input parameterisation.

Subsequently, the local score of the best performing pre-trained network was incorporated into the DP segmentation approach proposed in Chapter 4, and was found to perform poorly. This supports the

findings of Chapter 8, which found that the performance of the DP approach deteriorates in comparison to the use of a simple threshold as the local score becomes more accurate.

Chapter 11

Summary and conclusions

In this thesis we have considered the automatic and unconstrained segmentation of speech into phoneme-like units. While the main motivation behind this research is to facilitate the process of creating pronunciation dictionaries for under-resourced languages, such segmentations are useful for a variety of other purposes.

The thesis begins with a background study on the different unconstrained segmentation algorithms found in the literature, which found that most algorithms employ some variation of a local score. A local score is a value that indicates where the characteristics of the audio signal change, and therefore gives an indication of when a segment boundary is likely to be present. There are many ways to calculate a local score, and we investigated the two major approaches: the first employs vector distance functions to detect changes in the feature vectors, and the second trains an MLP to estimate a local score. The training data consists of groups of feature vectors centred around phoneme boundaries and feature groups midway between phoneme boundaries in TIMIT, serving as examples of where segment boundaries are present and where they are not, respectively.

The predominant segmentation approach used in the literature to unconstrained segmentation is to apply a threshold to a local score. Local maxima that lie above this threshold lead to the hypothesis of a segment boundary. Closer investigation of this threshold driven approach led to the insight that very short segments (due to clusters of local maxima) and very long segments are postulated, although they are unlikely. We therefore proposed a generalisation of the segmentation algorithm that is based on dynamic programming and which incorporates a statistical segment length distribution in addition to the local score.

Blind segmentation algorithms are sensitive to clusters of local maxima close to phoneme boundaries that result from non-uniform transitions between phonemes. To counteract this, ad-hoc remedies are employed in the literature, at the cost of additional parameters. The DP algorithm inherently solves this problem by only selecting segments with both high local scores and an appropriate length. By comparing the performance of the DP algorithm with selected blind segmentation algorithms from the literature serving as benchmarks, it was demonstrated that, with the right feature vector and vector distance function combinations, the DP algorithm offers a clear performance improvement.

The DP framework was also used in combination with local scores generated by a simple MLP with a single hidden layer and 30 hidden neurons. In this configuration it did not lead to the same substantial

improvement in performance over the benchmark approach. This is due to the more accurate local scores generated by the MLP, which rarely exhibited clusters of peaks. This lack of boundary clusters limits the ability of DP to improve on the simpler threshold-based segmentations. It is possible, however, to combine the threshold and DP approaches in a way that uses the threshold to detect large peaks and DP to choose between smaller and more ambiguous peaks. This configuration led to the best performance across all accuracy measures.

As a final step, the application of deep neural networks to the estimation of a local score was investigated, with and without generative pre-training. The NNs consisted of between 1 and 5 hidden layers, and were made up of RBMs when pre-trained, with a Gaussian Bernoulli RBM at the bottom layer to accommodate the real-valued inputs. The experiments revealed that the accuracy of the local scores increases as the neural networks grow in size. When 12 MFCCs and log energy were used to train the networks, pre-trained networks did not consistently lead to better performance. On the other hand, when the first and second derivatives were appended to the MFCCs, pre-trained networks convincingly outperformed the randomly initialised networks. It was found that this was due to the higher dimensionality of the second type of input vectors. For smaller input dimensionalities, overfitting occurs during pre-training. Finally, the best performing pre-trained and randomly initialised networks were compared, and a substantial improvement was observed for the pre-trained network using MFCCs, log energy and first and second derivatives as features.

The incorporation of the local score estimated by the best-performing deep neural network into the DP algorithm was also investigated, but found to be unsuccessful. This shows that the DP approach is made obsolete by the higher quality local scores estimated by the deep network.

11.1 Software contributions

MATLAB was used for the experimental evaluations presented in this thesis. The software used for extracting the training data from TIMIT, estimating the distributions for the DP-based segmentation algorithm, the DP and other blind segmentation algorithms, the accuracy measures, and all the experimental evaluations were developed entirely by the author. The software used to train the MLPs and RBMs is based on a deep learning toolbox [45], which was modified to serve our needs. These modifications include: adapting the code to use the MATLAB CUDA libraries and early stop training, and including training for Gaussian-Bernoulli RBMs.

11.2 Research contributions

- The typical threshold-based blind segmentation algorithms that are used in the literature were improved upon by incorporating DP as a means to take segment lengths into account. In order to achieve this we re-formulated and significantly extended an older DP algorithm found in the literature, for example by refining the probability distributions employed and by introducing the concept of path normalisation.

- For NN-based segmentation, where NNs are trained to distinguish between segment boundaries and continuity, we introduced deep neural networks. We established that deeper neural networks improved on the performance of single layer networks used in the literature.
- We were also able to show that generatively pre-trained deep neural networks further improved performance when the networks contained a very large number of input parameters. In this way we obtained the best performance for NN-based segmentation reported anywhere to date.
- Finally, we found that the DP algorithm is effective when the local scores exhibit multiple peaks, and the true segment boundaries are consequently unclear. However, when the local score is better estimated, such as by means of a deep neural network, the additional incorporation of DP becomes unnecessary. This finding will be useful in future implementations since it means that a simpler segmentation algorithm can be used.

11.3 Further work

It will be interesting to establish how the investigated and proposed algorithms perform on different corpora and languages. This will determine whether the blind algorithms can sustain their performance across different languages, and will also show how the MLPs, trained on the TIMIT corpus, perform on other corpora. Such work will be difficult due to the extreme rarity of corpora with manually-produced time-aligned phonetic transcription.

A related point to take into consideration is that the segment length distribution was trained on the TIMIT phonemes, and are not necessarily representative of the phonemes seen in other languages. It would therefore be interesting to determine these distributions for other languages to see how universal the distribution of phoneme lengths is.

List of References

[1] Goussard, G. and Niesler, T.: Automatic discovery of subword units and pronunciations for automatic speech recognition using TIMIT. In: *Proceedings of Pattern Recognition Association of South Africa, PRASA*. 2010.

[2] Halberstadt, A.K.: *Heterogeneous Acoustic Measurements and Multiple Classifiers for Speech Recognition*. Ph.D. thesis, Massachusetts Institute of Technology, MIT, 1998.

[3] Keri, V. and Prahallad, K.: A comparative study of constrained and unconstrained approaches for segmentation of speech signal. In: *Proceedings of International Conference on Spoken Language Processing, ICSLP*, pp. 2238–2241. 2010.

[4] Adell, J., Bonafonte, A., Gomez, J. and Castro, M.: Comparative study of automatic phone segmentation methods for TTS. In: *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 1, pp. 309 – 312. 2005. ISSN 1520-6149.

[5] Sharma, M. and Mammone, R.: 'Blind' speech segmentation: automatic segmentation of speech without linguistic knowledge. In: *Proceedings of the Fourth International Conference on Spoken Language Processing, ICSLP*, vol. 2, pp. 1237 –1240. oct 1996.

[6] ten Bosch, L. and Cranen, B.: A computational model for unsupervised word discovery. In: *Order: A Journal On The Theory Of Ordered Sets And Its Applications*, pp. 1 – 4. 2007.

[7] Wang, D., Lu, L. and Zhang, H.-J.: Speech segmentation without speech recognition. In: *Proceedings of International Conference on Multimedia and Expo, ICME*, vol. 1, pp. 405 – 408. july 2003.

[8] Hoffmann, S. and Pfister, B.: Fully automatic segmentation for prosodic speech corpora. In: *Proceedings of the International Conference on Spoken Language Processing, ICSLP*, pp. 1389–1392. 2010.

[9] Malfrère, F. and Dutoit, T.: High-quality speech synthesis for phonetic speech segmentation. In: *Proceedings of Fifth European Conference on Speech Communication and Technology, EUROSPEECH, Rhodes, Greece, September 22-25*. 1997.

[10] Okko Räsänen, Laine, U.K. and Altosaar, T.: Blind segmentation of speech using non-linear filtering methods. In: *Ipsic I. (Ed.): Speech Technologies*, pp. 105 –124. InTech Publishing, 2011.

[11] Sarkar, A. and Sreenivas, T.: Automatic speech segmentation using average level crossing rate information. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 1, pp. 397 – 400. 2005. ISSN 1520-6149.

[12] Estevan, Y.P., Wan, V. and Scharenborg, O.: Finding Maximum Margin Segments in Speech. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pp. 937 –940. 2007.

[13] Aversano, G., Esposito, A. and Marinaro, M.: A new text-independent method for phoneme segmentation. In: *Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems, MWSCAS*, vol. 2, pp. 516 –519. 2001.

[14] Hatazaki, K., Komori, Y., Kawabata, T. and Shikano, K.: Phoneme segmentation using spectrogram reading knowledge. In: *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 1, pp. 393 –396. may 1989. ISSN 1520-6149.

[15] Finster, H.: Automatic speech segmentation using neural network and phonetic transcription. In: *Proceedings of International Joint Conference on Neural Networks, IJCNN*, vol. 4, pp. 734 –736. June 1992.

[16] Malfrère, F., Deroo, O. and Dutoit, T.: Phonetic alignment : Speech synthesis based vs. hybrid HMM/ANN. In: *Proceedings of International Conference on Spoken Language Processing, ICSLP*, pp. 1571–1574. Sydney, Australia, 1998.

[17] Toledano, D.: Neural network boundary refining for automatic speech segmentation. In: *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 6, pp. 3438 –3441. 2000. ISSN 1520-6149.

[18] Lee, K.-S.: MLP-based phone boundary refining for a TTS database. *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 981–989, 2006.

[19] Suh, Y. and Lee, Y.: Phoneme segmentation of continuous speech using multi-layer perceptron. In: *Proceedings of Fourth International Conference on Spoken Language Processing, ICSLP 96*, vol. 3, pp. 1297 –1300. oct 1996.

[20] van Vuuren, V.Z., ten Bosch, L. and Niesler, T.: Automatic segmentation of TIMIT by dynamic programming. In: *Proceedings of the twenty-third annual conference of the Pattern Recognition Association of South Africa, PRASA, CSIR*. Pretoria, 2012.

[21] van Vuuren, V.Z., ten Bosch, L. and Niesler, T.: A dynamic programming framework for neural network-based automatic speech segmentation. In: *Proceedings of the fourteenth International Conference on Spoken Language Processing, ICSLP*. Lyon, France, 2013.

[22] Almpanidis, G. and Kotropoulos, C.: Phonemic segmentation using the generalised gamma distribution and small sample Bayesian information criterion. *Speech Communication*, vol. 50, no. 1, pp. 38–55, 2008.

[23] Cohen, J.R.: Segmenting speech using dynamic programming. *The Journal of the Acoustical Society of America*, vol. 69, p. 1430, 1981.

[24] Räsänen, O.J., Laine, U.K. and Altosaar, T.: An improved speech segmentation quality measure: the R-value. In: *Proceedings of the International Conference on Spoken Language Processing, ICSLP*. 2009.

[25] Bengio, Y.: Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[26] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P. and Bengio, S.: Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[27] Krizhevsky, A. and Hinton, G.: *Learning multiple layers of features from tiny images*. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[28] Kivinen, J.J. and Williams, C.: Multiple texture Boltzmann machines. In: *Proceedings of International Conference on Artificial Intelligence and Statistics*, pp. 638–646. 2012.

[29] Osindero, S. and Hinton, G.E.: Modeling image patches with a directed hierarchy of Markov random fields. In: *Advances in neural information processing systems*, pp. 1121–1128. 2007.

[30] Ranzato, M. and Hinton, G.E.: Modeling pixel means and covariances using factorized third-order Boltzmann machines. In: *Proceedings of Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 2551–2558. IEEE, 2010.

[31] Jaitly, N. and Hinton, G.: Learning a better representation of speech soundwaves using restricted Boltzmann machines. In: *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 5884–5887. IEEE, 2011.

[32] Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H.: Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[33] Hinton, G.E., Osindero, S. and Teh, Y.-W.: A fast learning algorithm for deep belief nets. *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[34] Mohamed, A.-r., Dahl, G.E. and Hinton, G.: Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[35] Hinton, G.E. and Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[36] Fischer, A. and Igel, C.: An introduction to restricted Boltzmann machines. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 14–36. Springer, 2012.

[37] Cheung, S.: Proof of Hammersley-Clifford theorem. Tech. Rep., 2008.

[38] Bishop, C.M. *et al.*: *Pattern recognition and machine learning*, vol. 1. Springer, 2006.

[39] Barber, D.: *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[40] Kindermann, R., Snell, J.L. *et al.*: *Markov random fields and their applications*, vol. 1. American Mathematical Society Providence, RI, 1980.

[41] Hinton, G.: A practical guide to training restricted Boltzmann machines. *Momentum*, vol. 9, p. 1, 2010.

[42] Cowles, M.K. and Carlin, B.P.: Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 883–904, 1996.

[43] Walsh, B.: Markov chain Monte Carlo and Gibbs sampling. 2004.

[44] Cho, K. *et al.*: *Improved learning algorithms for restricted Boltzmann machines*. Master's thesis, School of science, Aalto University, 2011.

[45] Palm, R.B.: *Prediction as a candidate for learning deep hierarchical models of data*. Master's thesis, Technical University of Denmark, DTU Informatics, 2012.
Available at: https://github.com/rasmusbergpalm/DeepLearnToolbox

Appendices

Appendix A

Derivation of RBM conditional probabilities

Let $\mathbf{v}_{-l}$ denote the state of all visible nodes except the lth one and define

$$\alpha_l(\mathbf{h}) := -\sum_{j=1}^{n} w_{lj}h_j - b_l \tag{A.0.1}$$

and

$$\beta(\mathbf{v}_{-l}, \mathbf{h}) := -\sum_{i=1, i\neq l}^{m}\sum_{j=1}^{n} w_{ij}v_i h_j - \sum_{i=1, i\neq l}^{m} b_i v_i - \sum_{j=1}^{n} c_j h_j \ . \tag{A.0.2}$$

Then $E(\mathbf{v}, \mathbf{h}) = \beta(\mathbf{v}_{-l}, \mathbf{h}) + v_l\alpha_l(\mathbf{h})$, where $v_l\alpha_l(\mathbf{h})$ contains all the terms involving v_l and we can write:

$$\begin{aligned}
p(v_l = 1|\mathbf{h}) = p(v_l = 1|\mathbf{v}_{-l}, \mathbf{h}) &= \frac{p(v_l = 1, \mathbf{v}_{-l}, \mathbf{h})}{p(\mathbf{v}_{-l}, \mathbf{h})} \\
&= \frac{e^{-E(v_l=1,\mathbf{v}_{-l},\mathbf{h})}}{e^{-E(v_l=1,\mathbf{v}_{-l},\mathbf{h})} + e^{-E(v_l=0,\mathbf{v}_{-l},\mathbf{h})}} = \frac{e^{-\beta(\mathbf{v}_{-l},\mathbf{h})-1.\alpha_l(\mathbf{h})}}{e^{-\beta(\mathbf{v}_{-l},\mathbf{h})-1.\alpha_l(\mathbf{h})} + e^{-\beta(\mathbf{v}_{-l},\mathbf{h})-0.\alpha_l(\mathbf{h})}} \\
&= \frac{e^{-\beta(\mathbf{v}_{-l},\mathbf{h})}e^{-\alpha_l(\mathbf{h})}}{e^{-\beta(\mathbf{v}_{-l},\mathbf{h})}e^{-\alpha_l(\mathbf{h})} + e^{-\beta(\mathbf{v}_{-l},\mathbf{h})}} = \frac{e^{-\beta(\mathbf{v}_{-l},\mathbf{h})}e^{-\alpha_l(\mathbf{h})}}{e^{-\beta(\mathbf{v}_{-l},\mathbf{h})}(e^{-\alpha_l(\mathbf{h})} + 1)} \\
&= \frac{e^{-\alpha_l(\mathbf{h})}}{e^{-\alpha_l(\mathbf{h})} + 1} = \frac{\frac{1}{e^{\alpha_l(\mathbf{h})}}}{\frac{1}{e^{\alpha_l(\mathbf{h})}} + 1} = \frac{1}{1 + e^{\alpha_l(\mathbf{h})}} \\
&= \text{sigmoid}(-\alpha_l(\mathbf{h})) = \text{sigmoid}\left(\sum_{j=1}^{n} w_{lj}h_j + b_l\right)
\end{aligned} \tag{A.0.3}$$

Appendix B

PRASA paper

Van Vuuren, V.Z., ten Bosch, L. and Niesler, T.: Automatic segmentation of TIMIT by dynamic programming. In: *Proceedings of the twenty-third annual conference of the Pattern Recognition Association of South Africa*, PRASA, CSIR. Pretoria, 2012.

Automatic segmentation of TIMIT by dynamic programming

Van Zyl van Vuuren*, Louis ten Bosch† and Thomas Niesler*

*Department of Electrical and Electronic Engineering
University of Stellenbosch, South Africa
Email: {15446204,trn}@sun.ac.za

†Department of Linguistics
Radboud University Nijmegen, The Netherlands
Email: l.tenbosch@let.ru.nl

***Abstract*—We propose an algorithm based on the principle of dynamic programming for the automatic segmentation of continuous speech into phoneme-like units. A measure of local dissimilarity among consecutive feature vectors is combined with a knowledge of the expected statistical distribution of the segment lengths within a dynamic programming framework to obtain an optimal placement of segment boundaries. We compare the performance of our algorithm with the performance of two recently-proposed alternatives by measuring how closely the hypothesised boundaries match the TIMIT phone boundaries. The results showed that we are able to improve on the performance of the two contrasting approaches. Furthermore, we show that a hybrid approach which combines aspects of all three algorithms leads to even better results.**

I. Introduction

The task of accurately segmenting a speech signal into phoneme-like units plays an important role in the speech processing field. Although accurate manual segmentation can be achieved by trained phoneticians, the task is tedious, expensive and intrinsically subjective. In HMM-based ASR systems, time-aligned phonetic transcriptions are often needed for the development of the pronunciation dictionary and acoustic models. This is not always feasible, and is a particular obstacle for the development of ASR systems for under-resourced languages, for which no, or very little, transcribed phonetic material is available. In these situations, automatic segmentation algorithms can accelerate the task of developing a pronunciation dictionary and obtaining suitable bootstrapping acoustic training data, thereby substantially reducing the time it would take to develop the ASR system. The availability of reliable automatic segmentation algorithms is also useful in technologies outside ASR, such as the study of pronunciation variation and the development of coherent large-scale dictionaries.

Several approaches to the automatic segmentation of speech have been proposed over the years. Some require prior training, relying for example on HMM forced alignments [1]. Others make use of previously stored speech segments for template matching by using the phonetic transcription [2]. A third and more prevalent class of algorithms rely solely on the acoustic information to detect transient events in the speech signal [3]–[7]. When considering a under-resourced setting in which speech corpora are unavailable or very small, model training may not be feasible. Under these circumstances this latter class of algorithms represents the most viable option.

In this paper we propose a new algorithm for the acoustic segmentation of speech based on the principle of dynamic programming (DP). DP-based segmentation has been proposed in [3], in which a distortion metric within segments is minimised by using prior knowledge of the number of phones in a sequence. The algorithm we propose requires no information regarding the number of phones and maximises the probability of a specific segment boundary sequence.

A well known class of speech segments are phonemes, the identification of which is the goal of most published segmentation algorithms. By using the annotated phoneme boundaries given in TIMIT, the acoustic characteristics in the vicinity of the phoneme boundaries as well as the lengths of the phonemes can be inspected. The proposed algorithm then uses this prior information to infer the probability of a boundary occurring at every specific point in time in a speech signal. Dynamic programming principles are then applied to detect the most probable sequence of boundary positions.

Section II gives a brief overview of the class of segmentation algorithms based on transient events in acoustic information, and includes a discussion on a few recent algorithms. Section III provides a detailed description of the proposed DP-based segmentation algorithm, and Section IV discusses the quality measures used to assess segmentations. The experimental setup is specified in Section V, and experimental results are given in Section VI. Finally concluding remarks are presented in Section VII.

II. Background

Many segmentation algorithms are based on the assumption that there are regions in speech, termed speech segments, where the acoustic features stay relatively constant, and that there are clear transitions between such regions. To detect these transitions, the algorithms employ some estimate of the local acoustic change in the signal. 'Local' in this context refers to temporal acoustic changes taking place at a specific time independent of any previous or future acoustic changes within the signal. A function that quantifies these local acoustic changes will be referred to as the **local score** function in the remainder of this text. The local score function is central to all acoustic segmentation methods, and therefore different types of local score functions and their application in the recent literature will briefly be reviewed.

A. Algorithms based on maximum local acoustic change

The most common approach used in speech segmentation is to hypothesise segment boundaries at the times at which local acoustic change is at a maximum. These local maxima are found by searching for the peaks in the local score. However, the local score may contain many small peaks, which are the result of small acoustic changes that do not necessarily indicate segment boundaries. These additional peaks can lead to *over-segmentation*, where more than one segment boundary is hypothesised while only one is truly present. Over-segmentation can be reduced by including a threshold below which peaks are ignored. A selection of segmentation algorithms falling into this category are reviewed in the following. They were specifically chosen to illustrate a diversity of local score functions, of which a

selection will later be compared experimentally. The local score will henceforth be denoted as LS in equations.

1) Räsänen et al. [4]: The local score function used in this algorithm is the cross correlation between two FFT magnitude vectors. This is shown in Equation 1, where f and g represent the FFT magnitude vectors for the frames to the left and to the right respectively of the investigated frame, F_j.

$$LS(F_j) = \frac{f.g}{\parallel f \parallel \parallel g \parallel} \tag{1}$$

Feature vectors that are similar will give a score close to 1, and dissimilar vectors will give a score closer to 0. The algorithm applies a non-linear filter to the cross-correlation sequence in order to quantify the degree of uniformity in the region preceding and following the point of interest. In a similar way, the dissimilarity between these regions is also determined. The difference between the dissimilarity and uniformity values leads to a signal of which the valleys corresponds to probable segment boundaries. However, this signal is very noisy, and there are many small valleys. The number of these smaller valleys is reduced by application of a 'minmax' filter, which searches a fixed region (n_{mm}) around the point of interest to find the local maximum and minimum values. The difference between this maximum and minimum serves as the output of the filter at the position of the minimum. This filter is applied throughout the signal in non-overlapping regions. The filter output is a signal of which the peaks represents possible boundaries. Given that the 'minmax' filter region is usually very small and applied in non-overlapping intervals, many closely spaced peaks may still remain. Temporal peak masking is therefore applied in a subsequent step. Two peaks falling within a determined interval (t_d) of each other and which are above a chosen threshold (p_{min}) are identified, and the highest peak retained. The location of the highest peak is also shifted a small distance toward the eliminated smaller peak in proportion to their amplitudes.

2) Ten Bosch et al. [5]: This work uses the angle between the smoothed feature vectors just before and just after the point of interest to quantify the degree of local change. This is given by Equation 2, where f and g are the averages of the two feature vectors before and after the frame of interest F_j respectively.

$$LS(F_j) = arccos\frac{f.g}{(\parallel f \parallel \parallel g \parallel)^{\frac{1}{2}}} \tag{2}$$

12 MFCC and log energy together with their first and second derivatives are used as a 39-dimensional feature vector. All local maxima above a threshold (δ) are hypothesised as boundaries.

3) Estevan et al. [7]: This algorithm employs maximum margin clustering to detect points of change in a feature vector consisting of 12 MFCC coefficients, log energy and their first and second derivatives. A sliding window, N frames wide and centered about the frame of interest, sweeps through the signal. MMC clustering (using a RBF kernel) is applied to the frames within this window. The width of the RBF kernel, W, is estimated from a development set. The MMC clustering results in a cluster label for each frame within the window, and changes in these labels indicate possible segment boundaries. It was found that the best way to detect these changes is by using the Euclidean distance, as given by Equation 3, between the cluster labels and the cluster means. Let f be the cluster label of each frame within the sliding window, and g be the mean of the cluster throughout the signal. Peaks in the Euclidean distance will then indicate the segment boundaries.

$$LS(F_j) = [\sum_{l=1}^{T}(f_l - g_l)^2]^{\frac{1}{2}} \tag{3}$$

4) Sarkar et al. [6]: This method differs from the previous three by operating in the time domain rather than the frequency domain. The local score function used in this case is the average level crossing rate. The level crossing rate is closely related to the zero crossing rate, but with multiple additional levels other than $y = 0$, and among which the average crossing rate is taken. The levels can be distributed uniformly or non-uniformly. For this choice of local score, a boundary corresponds to a valley rather than a peak. As for some of the preceding algorithms, a threshold is required to prune out shallow valleys which lead to over-segmentation.

B. Algorithms based on minimising a distortion metric

Another approach to speech segmentation is to increase the uniformity within segments, i.e. to minimise some distortion metric within segments. In the work by Sharma et al. [3], the local score is the Euclidean distance applied to MFCC features. The distortion within a segment is calculated by Equation 5. This calculation employs the local score at frame j, given by Equation 3, and the mean of the local score from frame i to n, given by Equation 4. The segment stretching from frame i to frame n is denoted by $S_{i,n}$.

$$M_{i,n} = \frac{1}{n-i+1}\sum_{j=i}^{n} LS_j \tag{4}$$

$$distortion\_metric(S_{i,n}) = \sum_{j=i}^{n}(LS_j - M_{i,n})^2 \tag{5}$$

The overall distortion of the speech signal is a cumulative sum of the distortions of all the segments. The overall distortion can be minimised by applying a level-based DP algorithm to search for the optimal segmentation, assuming that the number of levels (segments) in the signal is known.

C. A proposed local score

For our formulation of the segmentation problem it is convenient if the local score lies between the values of 0 and 1. We propose the use of a normalised city block distance as shown in Equation 6,

$$LS(F_j) = \frac{\sum_{l=1}^{T}|f_l - g_l|}{\sum_{l=1}^{T}|f_l| + \sum_{l=1}^{T}|g_l|} \tag{6}$$

where f and g are the feature vectors before and after the frame of interest Fj. This proposed formulation of the local score will be compared with other candidates in the experimental evaluation. Note that parameterisations for f and g are not specified, allowing different feature vectors to be used during experimentation.

III. A DP-BASED SEGMENTATION ALGORITHM

Most segmentation algorithms based on maximum local-acoustic changes are prone to over-segmentation because they hypothesise more than one segment boundary at a point of acoustic change. This occurs due to the presence of multiple local maxima in the local score. To counteract this, the algorithms include various types of thresholds to eliminate such very short segments. Several examples

of such measures were given in Section II. These remedies are ad-hoc, however, and introduce additional parameters into the algorithm that require optimisation.

The algorithm we propose includes an explicit probabilistic model for the length of a segment. Segments that are either very short or very long are penalised by their associated low probability. The probability distribution of phoneme lengths for TIMIT can be estimated from the phonetic annotations, as illustrated in Figure 1. For illustrative purposes, the distribution is normalised with respect to its maximum probability.

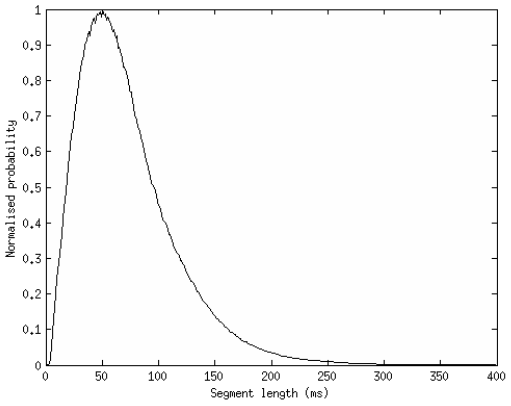


Fig. 1. Probability distribution of phoneme lengths in the TIMIT training set [8].

A. Segment probability

To gain some insight into the behaviour of local scores near segment boundaries, the local score in the close vicinity of phoneme boundaries, as given by the TIMIT annotations, is calculated and used to estimate a local score probability distribution given a boundary. A similar distribution is determined for the local score values taken far from boundaries, i.e. a local score probability distribution given that there is no boundary. Figure 2 shows these distributions, each normalised with respect to its maximum probability, for the local score calculated with Equation 6 when using FFT magnitudes as the feature vector. The distributions of the local score and the phoneme length can now be used to determine the probability of a boundary occurring at a specific frame in a speech signal.

Consider a signal consisting of N+1 frames. Now let the time of occurrence of each frame correspond to a state of a HMM as shown in Figure 3, where M is the maximum allowed number of frames per segment and S_0 is the time of occurrence of the first frame of the signal. The vertical dashed arrows between S_1 and S_1, S_2 and S_2, and between S_{N-1} and S_{N-1} indicate an expansion of the same HMM state.

When a state is visited by a path through the Markov model shown in Figure 3, a segment boundary is considered to occur at the corresponding speech frame. The transition and emission probabilities are calculated according to Equations 7 and 8 respectively, where SL refers to the segment length, LS to the local score, and SB to the occurrence of a segment boundary.

$$a_{i,j} = P(S_j|SL(S_j, S_i)) \tag{7}$$

$$b_j = P(SB|LS(S_j)) \tag{8}$$

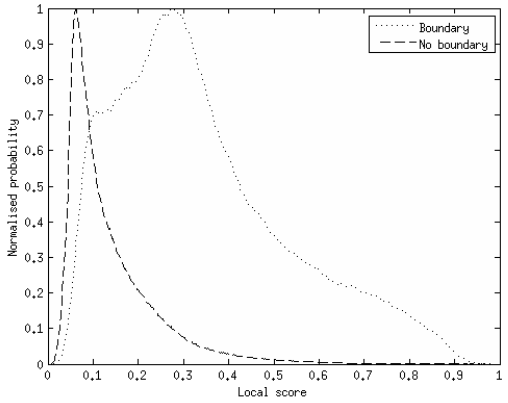


Fig. 2. Probability distribution estimates of local score values at, and away, from phoneme boundaries for Equation 6 applied to the FFT magnitudes.

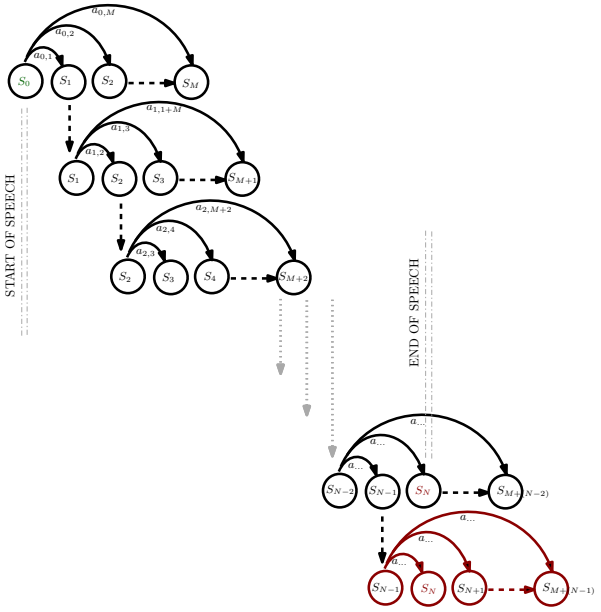


Fig. 3. DP-based segmentation cast as a HMM.

The segment length in Equation 7 is equal to the time step between two consecutive frames multiplied by the number of states separating the currently visited state and its parent state, as shown in Equation 9, where S_j is the current state, and S_i is the parent state.

$$SL(S_j, S_i) = (j - i) * step \tag{9}$$

Hence the transition probability is dependent only on the elapsed time between states. The emission probability at state Sj, as shown in Equation 8, is dependent on the local score $LS(Sj)$. To calculate the emission probability, Bayes rule is applied as shown in Equation 10, where $!SB$ refers to the absence of a segment boundary.

$$P(SB|LS(S_j)) = \frac{P(LS(S_j)|SB)P(SB)}{P(LS(S_j)|SB)P(SB) + P(LS(S_j)|!SB)P(!SB)} \tag{10}$$

The prior probability of a segment boundary can be estimated by dividing the number phoneme boundaries in the TIMIT annotations by the number of frames, as shown in Equation 11.

$$P(SB) = \frac{number\ of\ phoneme\ boundaries\ in\ TIMIT}{number\ of\ frames\ in\ TIMIT} \quad (11)$$

The probability that a boundary occurs at a particular frame can now be calculated by using Equations 9 and 10 in conjunction with estimates of the various probability distributions.

B. Optimal path

To find the globally optimal path from S_0 to S_N, all possible transitions shown in Figure 3 must be considered. This can be accomplished by using a DP algorithm. The states that were visited along the optimal path will identify the optimal segmentation. It is important to note that S_0 and S_N are always included in the path, and therefore the algorithm assumes that segment boundaries are always present at the start and the end of the speech signal. This means that any initial and final silence must be removed before applying the algorithm.

C. Normalising for path length

During the Viterbi decoding, many probabilities are multiplied together for any given path. When determining the optimal path, shorter paths (which contain fewer multiplications and thus longer segments) may be preferred, even when these have low associated emission and transition probabilities. We compensate for this effect by modifying the emission and transition probabilities as shown in Equations 12 and 13.

$$a_{i,j} = P(S_j|SL(S_j, S_i))^{SL(S_j,S_i)} \quad (12)$$

$$b_j = P(SB|LS(S_j))^{SL(S_j,S_i)} \quad (13)$$

These modifications normalise the path probability and remove the bias towards segmentations containing fewer segment boundaries.

IV. Assessing segmentation accuracy

In order to assess the quality of automatic-generated segmentations, we will determine how closely they correspond to the TIMIT phonetic segmentations. This provides a useful measure of segmentation accuracy. However it is dependent on the segmentation conventions used in TIMIT. For example, even though it is common practice for the /p/ to be segmented as a single phone in human annotations, the silence (closure) associated with the stop is considered a separate acoustic event in TIMIT. We found that the automatic segmentation algorithms could detect these closures quite accurately, and therefore decided to adhere to the original 61 TIMIT phone definitions without modification.

A. Comparing segmentations by DP

Comparing two sequences of segment boundary times can again be achieved by DP. We will proceed by first determining the best alignment between two sequences of boundary times. Then we will use this alignment to calculate a path cost. The alignment procedure uses a matrix of path costs as shown in Figure 4.

The first boundary in both sequences must coincide, and this corresponds to the bottom left cell of the matrix. Three alternative scenarios are then considered: (i) a hypothesised boundary $P_H(i)$ is paired (matches) a boundary $P_R(j)$ in the reference segmentation, (ii) a hypothesised boundary $P_H(i)$ is not paired with any boundary in the reference transcription (insertion) or (iii) there is no hypothesised boundary that can be paired with a boundary $P_R(j)$ in the reference transcription (deletion).

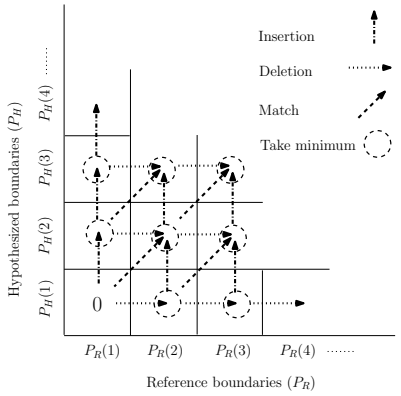


Fig. 4. Alignment matrix for segmentation scoring.

All possible paths from the bottom left to top right in the matrix shown in Figure 4 are computed recursively by dynamic programming. Starting from the bottom left of this matrix, each path can be extended upwards, to the right, or diagonally up and to the right, indicating an insertion, a deletion or a match between boundaries respectively. Each of these possibilities has a specific associated cost. When a reference boundary falls between two hypothesised boundaries, or vice versa, the cost is calculated by considering the distance to the nearest of the two boundaries. When paths meet, only the path with the lowest cost survives.

This procedure is applied iteratively, until all paths have reached the top right cell, which will then contain the final alignment cost between the two sequences. This cost reflects the difference between the hypothesised and reference sequences since it is the cumulative cost of every match, insertion, and deletion in the alignment. Furthermore, the cost has dimensions of time. By dividing it by the number of reference boundaries, the cost in seconds per reference boundary can be obtained. This is the average time difference between a hypothesised- and reference boundary and it will be used as a figure of merit in our later experiments. In addition, the number of insertions, deletions, and matches can be obtained by tracing back along the optimal path.

B. Fixed margin method

It appears to be standard practice in related research to consider a hypothesised and a reference segmentation boundary to be a match whenever they occur within 20ms of one another [4]. All non-matching boundaries are then either insertions or deletions. In order to make our results more directly compatible with those of others, this scoring framework has also been employed. An error measure termed the **average error** is defined, which is the average of the percentage insertions and deletions taken with respect to the number of reference boundaries in a speech signal. Furthermore, this interpretation of insertions, deletions and average error will be used.

V. Experimental setup

A. Data

Our experimental evaluations are based on the TIMIT database. The development set specified in [8] was used to optimise all

parameters, and the core test set defined in [8] was used exclusively for final testing. There is no speaker overlap between these two sets. The use of an explicit development set avoids biased results which would be obtained if the performance of the algorithm was measured on the same data used to optimise its hyperparameters. In the literature dealing with automatic segmentation, the separation of development and testing data was found not to be common. Leading and trailing silences were removed to account for the assumption that each utterance begins and ends with a segment boundary.

B. Feature vectors

We have chosen three feature vector configurations popular in literature on automatic speech segmentation for comparative experimentation.

1) FFT: Unprocessed 128-point FFT magnitudes
2) MFCC: 12 MFCCs and log energy
3) MFCC+Δ+$\Delta\Delta$: MFCC with appended first and second derivatives

By considering the local scores separately for the MFCCs, for the delta and for the acceleration features, it was found that a peak for the MFCCs or the acceleration components always coincides with a valley for the delta component, and vice versa. To account for this, the overall local score was calculated by averaging the local scores calculated for MFCCs and acceleration components, and the negative of the local score for the deltas.

C. The local score

Three local scores were investigated:

1) The cosine distance (C) shown in Equation 1,
2) The Euclidean distance (E) shown in Equation 3, and
3) The normalised city block distance (NCB) shown in Equation 6.

In our experiments, f and g were taken to be the averages of two frames to the left and two to the right of the inspected frame respectively. Depending on the local score, boundaries are expected to occur at either peaks or valleys (local maxima or minima) of the local score. Equation 10 is therefore only calculated at frames which coincide with local maxima or minima of the local score and a probability of 0 is assigned to all other frames.

D. The probability weights

As it stands, the DP segmentation algorithm will give equal weight to the transition and emission probabilities, due to the segment length and local score respectively. However, it may be beneficial to shift the balance more strongly towards one or the other. By multiplying the log values of the emission and transition probabilities by positive constants that sum to one, this shift in balance can be achieved, and will allow deletions to be traded for insertions and vice versa. Optimal performance on the development set was achieved by assigning a heavier weight to the emission probability (0.6–0.7) than to the transition probability (0.3–0.4). This gives a stronger preference to higher emission probabilities and leads to a reduction in insertions.

VI. Experimental results

A. Smoothing window size

In the following experiments a frame size of 16ms and a frame shift of 4ms were used. Before calculating the local score, each resulting MFCC and FFT value were smoothed by taking the average within a window centered on the feature vector in question. Subsequently, the average DP cost (Section IV-A) and the average error (Section IV-B) was calculated on the development set for different smoothing window sizes applied to different local score and feature vector combinations. Figures 5, 6, and 7 respectively show these results for the cases in which the cosine distance is applied to the FFT, the normalised city block distance is applied to MFCCs, and the Euclidean distance is applied to MFCC with first and second derivatives. For each configuration, all other parameters were optimised on the development set.

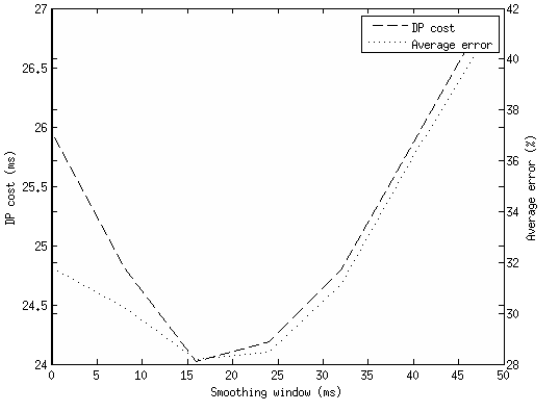


Fig. 5. DP cost (Section IV-A) and average error (Section IV-B) for different smoothing window sizes on the development set for the cosine distance applied to the FFT.

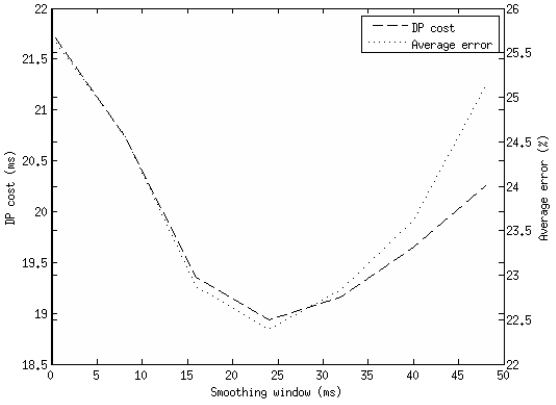


Fig. 6. DP cost (Section IV-A) and average error (Section IV-B) for different smoothing window sizes on the development set for the normalised city block distance applied to the MFCCs.

The results show that the optimal smoothing window sizes are similar for the FFT and MFCC parameterisations (16–24ms). A longer window (around 40ms) is required by the MFCC+Δ+$\Delta\Delta$ parameters, however. We believe that the introduction of first and second differentials introduces additional local maxima into the local score, which can lead to an increase in insertions. By lengthening the smoothing window, this is compensated for.

B. Choice of feature vector and local score

The performance of the DP segmentation algorithm when using the three different feature parameterisations and the three different local score formulations was compared experimentally, and results are shown in Table I. For each configuration, the length of the smoothing

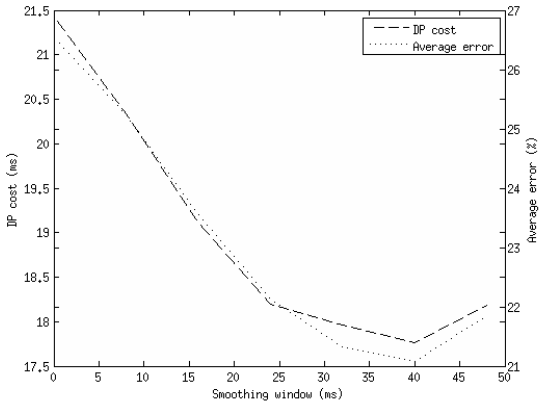


Fig. 7. DP cost (Section IV-A) and average error (Section IV-B) for different smoothing window sizes on the development set for the Euclidean distance applied to the MFCC with their first and second derivatives.

window as well as the probability weights are optimised on the development set, and segmentation accuracies determined on the test set. Both the DP path cost, in milliseconds per reference boundary, and the fixed margin average percentage error are shown.

TABLE I
DEVELOPMENT- AND TEST-SET PERFORMANCE OF THE DP SEGMENTATION ALGORITHM FOR THREE CHOICES OF FEATURE VECTOR AND FOR THE NORMALISED CITY BLOCK (NCB), EUCLIDEAN (E) AND COSINE (C) LOCAL SCORE (LS) FORMULATIONS.

Configuration		**DP Cost (ms)**		**%ERR**	
Feature	**LS**	**Dev**	**Test**	**Dev**	**Test**
FFT	NCB	18.62	18.42	20.21	19.80
MFCC	NCB	18.94	18.82	22.40	22.80
MFCC+Δ+ΔΔ	NCB	19.28	18.83	21.81	22.07
FFT	C	24.02	24.13	28.20	27.62
MFCC	C	19.01	18.98	22.53	22.85
MFCC+Δ+ΔΔ	C	18.94	18.72	21.56	21.72
FFT	E	28.06	27.93	33.27	33.13
MFCC	E	18.49	18.22	22.67	22.85
MFCC+Δ+ΔΔ	E	17.77	17.58	21.08	21.40

The normalised city block distance delivers the best overall performance. When applied to the MFCC+Δ+ΔΔ parameterisation, the Euclidean distance achieved similar performance. A configuration that stands out from the rest is the normalised city block distance applied to the FFT, which greatly outperforms all other combinations with the FFT feature vector. The FFT in general is the feature which is most sensitive to the remaining parameters, and was seen to be prone to over-segmentation. The FFT also has a higher dimensionality than the other parameterisation. It appears from the results that the normalised city block distance is most robust to this variation in dimensionality. Thus, the the normalised city block distance with a weighting leaning towards the emission probability (0.7) to reduce insertions gives very promising results. Among the feature parametrisations, the MFCC and the MFCC+Δ+ΔΔ are most competitive.

When comparing performance on the development and on the test sets, it is evident that the same patterns emerge from both. In the experiments that follow, each local score's best overall performing configuration will be used. These are the normalised city block distance for the FFT, the cosine distance for the MFCC+Δ+Δ, and the Euclidean distance for MFCC+Δ+Δ. These will henceforth be referred to as configuration C1, C2, and C3 respectively.

C. Silence removal

Many TIMIT sentences contain regions of silence in which temporal changes nevertheless occur. In order to avoid the hypotheses of segment boundaries in these regions, all boundaries were removed at frames when the ratio of the average energy content from 30ms before to 30ms after the frame in question, to the mean energy of the signal fall below a certain threshold. Different threshold values were investigated, and a typical result is shown in Figure 8. A threshold of 0.2% (i.e. a value of 0.002) delivered optimal performances for all configurations.

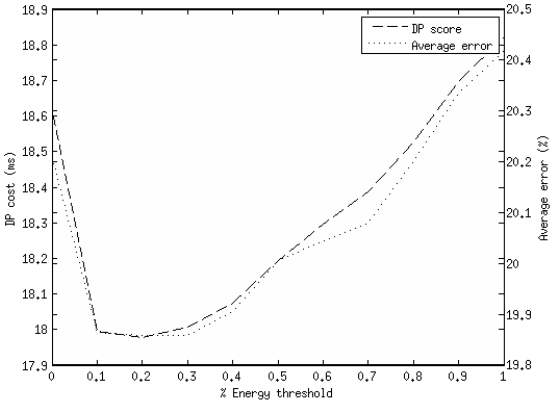


Fig. 8. DP cost (Section IV-A) and average error (Section IV-B) against % energy threshold on the development set for configuration C1.

D. Comparison with other segmentation algorithms

In the previous sections, an optimal configuration for the DP segmentation algorithm proposed in this paper is determined by experimentation. In this section we will benchmark the performance of this optimal configuration against two recent approaches to speech segmentation found in literature [4] [5]. Both approaches belong to the class of segmentation algorithms that rely on transient events in the acoustical information, as described in Section II-A. The method described in [4] claimed to achieve the same or better performance than many earlier approaches, while [5] is an algorithm with which the authors have had good prior experience.

Each method compensates for silences in its own way. The algorithm given in [5] scales the local score by the log frame energy to attenuate points of low energy, while the algorithm in [4] uses a similar approach to that proposed in this paper, but uses the average energy measured over the interval from -8ms to +30ms about the point of interest, and a threshold which is a multiple of the minimum energy for the signal. In the evaluation presented in the following, the parameters of each method were optimised on the development set.

Table II presents the DP cost in milliseconds per reference boundary, the percentage insertions and deletions with respect to the number of reference boundaries, and the average error for the optimised cases on the development set. The values shown for configurations C1, C2 and C3 are those achieved after silence removal.

When applying these parameter values to the core test set, the results shown in Table III are obtained.

TABLE II
PERFORMANCE COMPARISONS ON THE DEVELOPMENT SET AFTER SILENCE REMOVAL.

Method	DP Cost (ms)	% Ins	% Del	%ERR
DP (C1)	17.98	15.56	24.16	19.86
DP (C2)	18.12	15.28	26.97	21.13
DP (C3)	17.04	18.15	23.05	20.60
Räsänen	18.91	17.92	26.99	22.46
ten Bosch	25.07	26.19	27.37	26.78

TABLE III
PERFORMANCE COMPARISONS ON THE CORE TEST SET AFTER SILENCE REMOVAL.

Method	DP Cost (ms)	% Ins	% Del	%ERR
DP (C1)	17.92	14.49	24.53	19.51
DP (C2)	18.23	14.80	28.04	21.42
DP (C3)	17.13	17.14	24.93	21.03
Räsänen	19.40	17.18	28.19	22.68
ten Bosch	25.17	25.36	28.28	26.82

For illustrative purposes, the segmentations produced by the three algorithms for the same sentence, dr6-fbch0-sa1, are shown in Figures 9, 10, and 11, where configuration C3 was used for the DP algorithm. Each figure shows the first two seconds of the sentence as well as the TIMIT phone boundaries. The dashed vertical lines show the hypothesised boundaries, and the solid vertical lines show the reference boundaries.

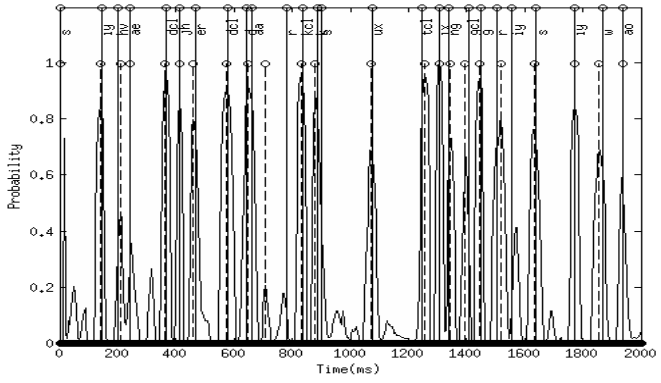


Fig. 9. Segmentation results for the DP algorithm on dr6-fbch0-sa1.

The vertical axis in Figure 9 for the DP algorithm shows the emission probabilities. Unlike the other two approaches, there is no threshold under which boundaries are ignored. Thus, even when the local score results in a low emission probability, a boundary can be hypothesised if the transition probability is high. This is clear, for example, at the boundary that is hypothesised at the 'aa' phoneme. The converse may also be true, i.e. even when the emission probability is high, a segment boundary may be suppressed by a low transition probability, as illustrated at the second 'iy'.

Figure 10 shows the output of the 'minmax' filter described in Section II-A of the Räsänen algorithm. Notice that all peaks falling within 32ms of each other have been combined by temporal peak masking, and that the threshold in this case is 0.07, below which all peaks are ignored. These parameter values were determined to be optimal for the development set.

The local score of ten Bosch's algorithm has been multiplied by the log energy to reduce the insertion of boundaries in regions of

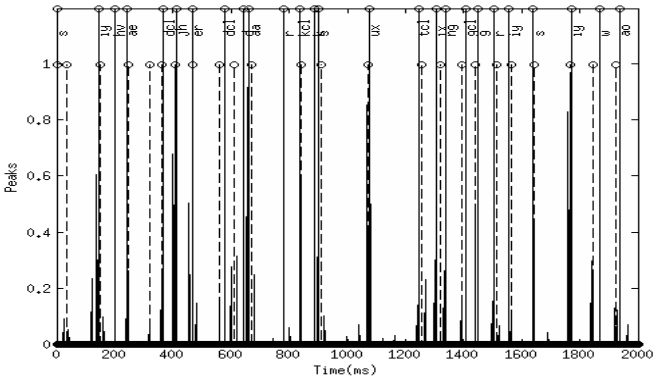


Fig. 10. Segmentation results for the Räsänen algorithm on dr6-fbch0-sa1.

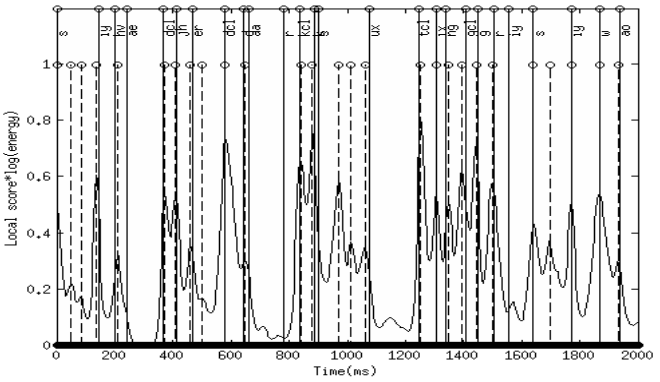


Fig. 11. Segmentation results for the ten Bosch algorithm on dr6-fbch0-sa1.

silence, which are characterised by very low energy. Unfortunately this also results in the introduction of unwanted small peaks when the log energy increases while the local score decreases, or when the energy decreases while the local score increases. Because the segment boundaries usually coincide with the peaks of the local score, these newly added peaks lead to insertions as shown, for example, in Figure 11 at 'er' and at each 's'. This leads to over segmentation, which is clear when looking at the higher percentage insertions in Table III. From the development set it was found that the optimal threshold for the ten Bosch algorithm is 0.13.

E. Combined methods

By inspection of the segmentation results produced by the DP algorithm, it was found that there regularly are small emission probability peaks present between the boundaries of very long segments. When these peaks coincide with high probability segment lengths, as determined by the segment length distribution, boundaries are hypothesised at these locations, resulting in unwanted insertions. With some experimentation it was found that better results can be obtained by applying a threshold to the emission probability (Equation 8) before searching for the optimal path by DP. All probabilities above the threshold are unchanged, and the probabilities below the threshold are reduced to 0. A variety of threshold values were investigated on the development set for each of the chosen three DP configurations, with all other parameters fixed at their previously found optimal values. Figures 12 and 13 show the resulting effect on the DP cost and

on the average error for configuration C1. By inspecting the average error graph, there is a point at which the reduction in insertions is greater than the rise in deletions. However, the average error can only be reduced to a certain point, after which the hypothesised and reference boundaries rapidly become misaligned. This is indicated at the point of DP cost increase. The DP cost is therefore the best way to determine the optimal threshold.

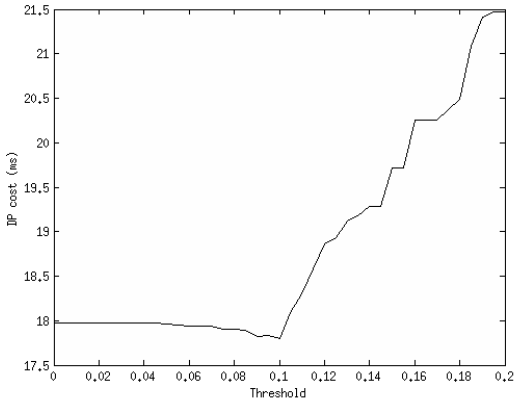


Fig. 12. DP cost (Section IV-A) against emission probability threshold on the development set for configuration C1.

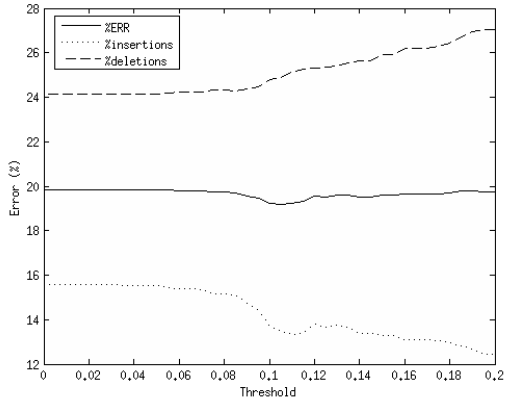


Fig. 13. Average error (Section IV-B) against emission probability threshold on the development set for configuration C1.

It was found that thresholds of 0.1, 0.5, and 0.1 lead to optimal performance on the development set for configurations C1, C2 and C3 respectively. When applied to the core test set, this leads to the results in Table IV.

TABLE IV
METHOD COMPARISONS, AFTER EMISSION PROBABILITY THRESHOLD WERE APPLIED, ON THE CORE TEST SET.

Method	DP Cost	% Ins	% Del	%ERR
DP (C1)	17.68	12.83	24.96	18.89
DP (C2)	18.08	13.91	28.44	21.17
DP (C3)	16.99	16.65	25.02	20.83
Räsänen	19.40	17.18	28.19	22.68
ten Bosch	25.17	25.36	28.28	26.82

By comparing the results in Tables III and IV, improvements in performance for all three configurations are seen. Two key values that stand out from Table IV are the small DP cost obtained by configuration C3, and the small average error obtained by configuration C1. The overall best, and most consistent configuration thus far, is configuration C1, which has the normalised city block distance and the FFT.

VII. SUMMARY AND CONCLUSION

We have proposed an algorithm based on the principle of dynamic programming for the automatic segmentation of continuous speech into phoneme-like units. A measure of the local dissimilarity between feature vectors is combined with a statistical description of the expected segment lengths within the dynamic programming framework in order to determine the optimal locations of segment boundaries within the speech utterance. We find that this approach leads to performance improvements relative to two alternative methods drawn from the literature. Analysis of the strengths of the individual techniques revealed that further improvements can be obtained by a hybrid approach employing aspects of each. We conclude that the use of dynamic programming as a basis for speech segmentation is a successful approach. In future work we plan to analyse the occurrence of insertion and deletion errors more carefully with respect to the type of phoneme within which they occur, as well as the role of context in the placement of segment boundaries. The effectiveness of our DP-based segmentation will also be tested on other languages using the distributions created from TIMIT to see how universal the segment boundary behaviour is. Furthermore, we will investigate the sensitivity of the segmentation algorithms to parameter changes, and the effect of increased parameters.

REFERENCES

[1] Y. jun Kim and A. Conkie, "Automatic segmentation combining an hmm-based approach and spectral boundary correction," in *Proceedings of the International Conference on Spoken Language Processing, ICSLP*, 2002, pp. 145–148.

[2] T. Svendsen and F. Soong, "On the automatic segmentation of speech signals," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 12, apr 1987, pp. 77 – 80.

[3] M. Sharma and R. Mammone, "'blind' speech segmentation: automatic segmentation of speech without linguistic knowledge," in *Proceedings of the Fourth International Conference on Spoken Language Processing, ICSLP*, vol. 2, oct 1996, pp. 1237 –1240.

[4] Okko Räsänen, U. K. Laine, and T. Altosaar, "Blind segmentation of speech using non-linear filtering methods," in *Ipsic I. (Ed.): Speech Technologies*. InTech Publishing, 2011, pp. 105 –124.

[5] L. ten Bosch and B. Cranen, "A computational model for unsupervised word discovery," in *Order A Journal On The Theory Of Ordered Sets And Its Applications*, 2007, pp. 1 – 4.

[6] A. Sarkar and T. Sreenivas, "Automatic speech segmentation using average level crossing rate information," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 1, 2005, pp. 397 – 400.

[7] Y. P. Estevan, V. Wan, and O. Scharenborg, "Finding Maximum Margin Segments in Speech," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 2007, pp. 937 –940.

[8] A. K. Halberstadt, "Heterogeneous Acoustic Measurements and Multiple Classifiers for Speech Recognition ," Ph.D. dissertation, Massachusetts Institute of Technology, MIT, 1998.

Appendix C

INTERSPEECH paper

Van Vuuren, V.Z., ten Bosch, L. and Niesler, T.: A dynamic programming framework for neural network-based automatic speech segmentation. In: *Proceedings of the fourteenth International Conference on Spoken Language Processing*, ICSLP. Lyon, France, 2013.

A dynamic programming framework for neural network-based automatic speech segmentation

[1]*Van Zyl van Vuuren,* [2]*Louis ten Bosch,* [1]*Thomas Niesler*

[1]Department of Electrical and Electronic Engineering, University of Stellenbosch, South Africa
[2] Department of Linguistics, Radboud University, Nijmegen, The Netherlands
{vzvv,trn}@sun.ac.za, l.tenbosch@let.ru.nl

Abstract

Neural networks have recently been shown to be a very effective approach to the unconstrained segmentation of speech into phoneme-like units. The neural network is trained to indicate when a short local sequence of feature vectors is associated with a segment boundary, and when it is not. Although this approach delivers state-of-the-art performance, it is prone to over-segmentation at ambiguous segment boundaries. To address this, we propose the incorporation of the neural network segmenter into a dynamic programming (DP) framework. We evaluate the DP-based approach on the TIMIT corpus, and show that it leads to improved performance.
Index Terms: unconstrained automatic speech segmentation, dynamic programming, multilayer perceptrons

1. Introduction

The task of accurately segmenting a speech signal into phoneme-like units plays an important role in the speech processing field. Although accurate manual segmentation can be achieved by trained phoneticians, the task is tedious, expensive and subjective. In an under-resourced setting, in which very little transcribed phonetic material is available, automatic segmentation algorithms can accelerate the task of developing a pronunciation dictionary and obtaining suitable bootstrapping acoustic training data, thereby substantially reducing the time it would take to develop an automatic speech recognition (ASR) system. The availability of reliable automatic segmentation algorithms is also useful in technologies outside ASR, such as the study of pronunciation variation, the development of coherent large-scale dictionaries, text-to-speech (TTS) applications [1], and many others [2, 3, 4].

A distinction can be made between segmentation approaches that require phone or orthographic transcripts, and those that do not. These two approaches are often referred to as constrained and unconstrained respectively [5]. There is also a distinction between algorithms that segment speech into syllables and into phoneme-like units.

Constrained approaches usually perform a forced alignment between phoneme based hidden Markov models (HMMs) and a phonetic transcription [1, 5, 6], or align phoneme templates to a signal using dynamic time warping (DTW) [1, 5, 7]. Unconstrained approaches typically rely on a scoring function that is applied at the feature level and indicates possible segment boundaries. Because these scores are calculated from a local group of features, we will refer to them as 'local scores'. Popular local scores are vector distance functions which respond to the dynamics of the features and from which a peak-picking algorithm finds viable local maxima at which to hypothesise segment boundaries [8, 9, 10, 11, 3]. Rule based approaches that use language-specific knowledge to calculate a local score independent of the phone string [4, 12], and HMM phone loop segmentation also fall into the unconstrained class [5].

Artificial neural networks (ANNs) have been applied to both constrained and unconstrained segmentation approaches. The constrained approaches are mostly based on hybrid HMM/ANN algorithms in which multilayer perceptrons (MLPs) act either as phone probability estimators [13, 14], or are used to detect phoneme transitions in order to refine the boundaries produced by an HMM alignment [15, 16]. For unconstrained segmentation, ANNs have recently been shown to be highly effective [5, 17]. We propose an improved unconstrained ANN segmentation algorithm by introducing a dynamic programming (DP) framework which employs a probabilistic segment length model in conjunction with the ANN scores to hypothesise segment boundaries. The TIMIT corpus will be used for the training and testing of the proposed algorithm.

Section 2 gives a brief overview of the ANN-based segmentation algorithm used as the baseline, Section 3 describes the DP algorithm, Section 4 discusses the evaluation methods used, Section 5 gives an overview of the experimental setup, Section 6 contains the results, and conclusions are given in Section 7.

2. Segmentation using neural networks

An MLP can be employed to compute a local score on the basis of a group of consecutive feature vectors. In recent work this was achieved by training two output neurons, one outputs a high value when the evidence in the input feature vectors supports the presence of a boundary, and the other when the evidence supports the absence of a boundary [5]. The training data consists of feature vector groups located around phoneme boundaries and feature vector groups midway between two boundaries. The local score is obtained by taking the difference between the two outputs. Approaches that rely on the detection of local maxima in such local scores, such as those proposed in [8, 9, 10, 11, 3], may now be employed to find possible segment boundaries.

As proposed in [5], an MLP with 30 hyperbolic tangent neurons in the hidden layer, and two hyperbolic tangent neurons in the output layer was employed for segmentation in our experiments. A window size of 10ms with a step of 5ms was used to calculate the feature vectors. Groups of 11 consecutive feature vectors centred about the point of interest were used with 12 MFCCs and log energy as features. The network was trained by back-propagation, and functions by detecting regions in time where the local score is larger than zero throughout the region. A segment boundary is then hypothesised at the frame

at which the local score is at a maximum within the region as demonstrated by Equation 1,

$$[\hat{B_R}] = \underset{t\epsilon\{S_R \ldots E_R\}}{\operatorname{argmax}} \{LS(i_t)\} \quad (1)$$

where B_R is the boundary frame in the region, S_R and E_R are the start and end of the region respectively, LS is the local score, and i_t are the frames between S_R and E_R [5].

3. DP-based segmentation

We propose to embed the neural-network based detection mechanism described in the previous section in a dynamic programming (DP) framework by including an explicit probabilistic model for the length of a segment. In this way segments that are either very short or very long are penalised by their associated low probability. The probability distribution of phoneme lengths for TIMIT is illustrated in Figure 1. For illustrative purposes, the distribution has been normalised with respect to its maximum.

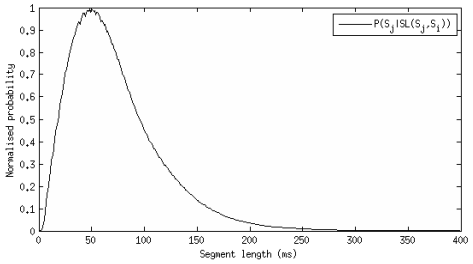


Figure 1: Probability distribution of phoneme lengths in the TIMIT training set [18], as used by Equation 2.

The DP-based segmentation algorithm will in general need to compute probabilities for segments as long as the utterance itself. This is achieved by attaching a linear decaying tail stretching from the maximum segment length for which a probability estimate is available to the length of the utterance at hand.

3.1. Local score probability distributions

To gain some insight into the behaviour of the local score near segment boundaries, its probability distribution in boundary regions can be estimated. A similar distribution can be determined for regions that are far from these boundaries. Figure 2 shows these distributions for the local score when calculated with the MLP proposed in Section 2. The distributions were estimated from the TIMIT core test set, and are normalised with respect to their maxima for clarity. The distributions shown in Figures 1 and 2 are used to determine the probability that a boundary occurs at a specific frame in a speech signal.

3.2. Dynamic programming

Consider an utterance consisting of N+1 frames. Let the time of occurrence of each frame correspond to a state of an HMM as shown in Figure 3, where M is the maximum allowed number of frames per segment and S_0 is the time of occurrence of the first frame of the signal. The vertical dashed arrows between S_1 and S_1, and between S_{N-1} and S_{N-1} indicate an expansion of the same HMM state.

When a state is visited by a path through the Markov model, a segment boundary is considered to occur at the corresponding speech frame. The transition and emission probabilities are calculated according to Equations 2 and 3 respectively, where SL refers to the segment length, LS to the local score, and SB to the occurrence of a segment boundary.

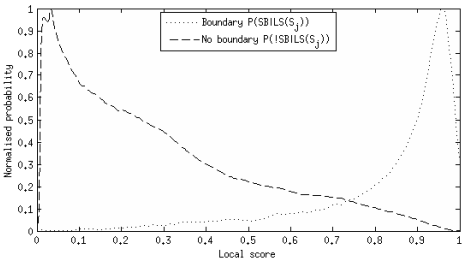


Figure 2: Estimated probability distribution of the MLP local score values at, and away from, phoneme boundaries (Eq. 3).

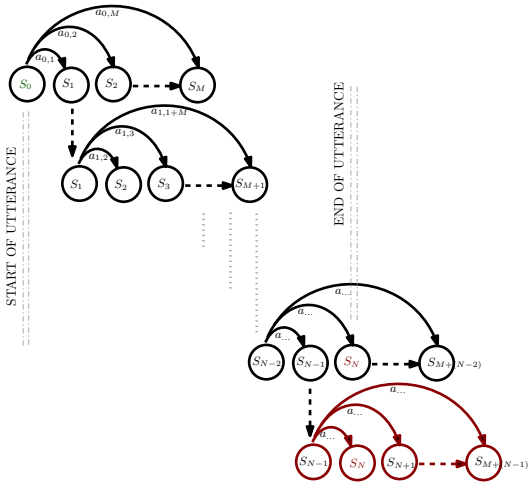


Figure 3: DP-based segmentation formulated as an HMM.

$$a_{i,j} = P(S_j|SL(S_j, S_i)) \quad (2)$$

$$b_j = P(SB|LS(S_j)) \quad (3)$$

The segment length is given by Equation 4, where $step$ is the frame step in seconds.

$$SL(S_j, S_i) = (j - i) * step \quad (4)$$

Hence the transition probability is dependent only on the elapsed time between states, while the emission probability at state Sj is dependent on the local score $LS(Sj)$. The emission probability can be calculated by the application of Bayes rule as shown in Equation 5, where $!SB$ refers to the absence of a segment boundary.

$$P(SB|LS(S_j)) = \frac{P(LS(S_j)|SB)P(SB)}{P(LS(S_j)|SB)P(SB) + P(LS(S_j)|!SB)P(!SB)} \quad (5)$$

The prior probability of the occurrence of a segment boundary can be estimated from the TIMIT corpus, as shown in Equation 6.

$$P(SB) = \frac{number\ of\ phoneme\ boundaries\ in\ TIMIT}{number\ of\ frames\ in\ TIMIT} \quad (6)$$

The probability that a boundary occurs at a particular frame can now be calculated by using Equations 4 and 5 in conjunction with estimates of the probability distributions.

3.3. Optimal path

The globally optimal path from S_0 to S_N in Figure 3 can be determined using the Viterbi algorithm. The states that are visited by the optimal path identify the optimal segmentation. States S_0 and S_N are always included in the path, and therefore the algorithm assumes that segment boundaries are always present at the start and the end of the speech signal. This means that any initial and final silence must be removed before applying the algorithm.

3.4. Normalising for path length

During the Viterbi decoding, many probabilities are multiplied together for any given path. Shorter paths (which contain fewer multiplications and thus longer segments) may therefore be preferred, even when these have low associated emission and transition probabilities. We compensate for this by modifying the emission and transition probabilities as shown in Equations 7 and 8.

$$a_{i,j} = P(S_j|SL(S_j, S_i))^{SL(S_j,S_i)} \quad (7)$$

$$b_j = P(SB|LS(S_j))^{SL(S_j,S_i)} \quad (8)$$

This modification normalises segment probabilities with respect to their lengths.

4. Assessing segmentation accuracy

In order to assess the quality of automatic-generated segmentations, we will determine how closely they correspond to the TIMIT phonetic segmentations.

4.1. Comparing segmentations by fixed margins

It appears to be standard practice in related research to consider a hypothesised and a reference segmentation boundary to be a match whenever they occur within 20ms of one another [1, 5, 6, 8, 9, 10, 11, 14, 16]. All non-matching boundaries are then regarded as either insertions or deletions. In order to make our results comparable to those of others, this scoring framework has been employed. An error measure termed the **average error** (ERR) is calculated. This figure is the average percentage insertions (INS) and deletions (DEL) taken with respect to the number of reference boundaries in the utterance.

4.2. Comparing segmentations by DP

Two sequences of segment boundary times can also be compared using DP. We proceed by first determining the best alignment between the two sequences of boundary times. The total absolute time difference between the boundaries paired in this alignment is taken as the cost of the path. By dividing the path cost by the number of reference boundaries, the cost in seconds per reference boundary can be obtained. This will be referred to as "DP Cost" and used as a figure of merit in our experiments.

A disadvantage of the fixed margin method is that all insertions and deletions are considered equal regardless of their positions. For example, a succession of deletions is not explicitly penalised in the fixed margin method. Comparing segmentations by DP penalises insertions and deletions relative to their closest paired boundary, and a succession of deletions will therefore result in a large cost because the closest paired boundary will be far away. We will use the DP evaluation mechanism in addition to the fixed margin method to obtain a better impression of how closely two boundary sequences are aligned.

5. Experimental setup

5.1. Data

Our experimental evaluations are based on the TIMIT database [19]. The development set specified in [18] was used to optimise all parameters, and the core test set was reserved exclusively for final testing. There is no speaker overlap between any of the sets. Leading and trailing silences were removed to ensure that each utterance begins and ends with a segment boundary.

5.2. Probability weights

As it stands, the DP segmentation algorithm will give equal weight to the transition and emission probabilities, due to the segment length and local score respectively. However, it may be beneficial to shift the balance more strongly towards one or the other. By multiplying the log values of the emission and transition probabilities by positive constants that sum to one, this shift in balance can be achieved, and will allow deletions to be traded for insertions and vice versa. In all our experiments, this value was optimised on the development set.

6. Results

Table 1 compares the performance of the NN based segmentation algorithm proposed in [5], the same algorithm when embedded in the DP framework we propose in Section 3, and a combined approach that will be discussed in Section 6.1. Note that the results presented by [5] were calculated by using different assessment measurements to those used in our paper, and are therefore not directly comparable.

Method	DP Cost (ms)	%INS	%DEL	%ERR
NN	14.50	12.74	17.00	14.87
DP	13.57	12.46	17.96	15.21
Combined	13.20	13.06	15.72	14.39

Table 1: Comparison of core test set performance of NN based segmentation in isolation, when embedded in a DP framework, and when NN and DP approaches are combined.

In our experiments, we attempted to keep the deletion and insertion errors of the DP segmentation close to those obtained in [5] to facilitate comparison. Table 1 shows that an increase in the average error is observed for the DP segmentation method. On the other hand, a considerable decrease in the DP cost was achieved. This indicates that the DP leads to segment boundaries that are on average more closely aligned to the TIMIT boundaries, but with slightly fewer boundary pairs within 20ms of one another.

During informal evaluation we observed typical errors made by both the NN and DP algorithms. We illustrate some of these errors by means of the segmentations produced by both algorithms for the same sentence, dr6-fbch0-sa1, in Figures 4 (NN) and 5 (DP). Each figure shows the first two seconds of the utterance. The dashed vertical lines show the hypothesised boundaries, and the solid vertical lines show the TIMIT reference phone boundaries.

For the NN algorithm, insertions were found to be frequent at boundary regions where local scores had very small amplitudes. The boundary in the middle of the second 'r' in Figure 4 illustrates this type of insertion. The NN method also has a tendency to miss segment boundaries when more than one large peak occurs in the boundary region. The deletion of the boundary at the start of the second 's' illustrates this tendency.

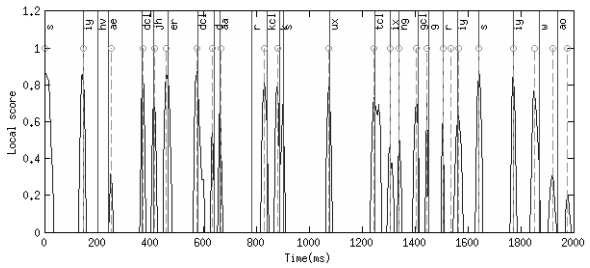


Figure 4: Segmentation results for the NN algorithm for sentence dr6-fbch0-sa1.

The DP implementation, on the other hand, frequently skips a plausible boundary because it closely precedes a boundary with relatively high probability, leading to a deletion. The deletion near 'd' in Figure 5 is an example of such an error.

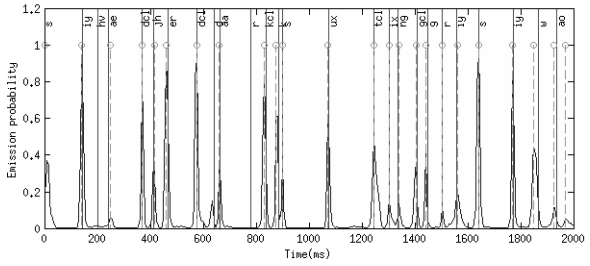


Figure 5: Segmentation results for the DP algorithm for sentence dr6-fbch0-sa1.

6.1. Emission probability threshold

The local scores generated by the MLP appear to be very good indicators of the presence of boundaries. The tendency of plausible boundaries to be skipped by the DP implementation is therefore detrimental to its performance when using the MLP-based local score. We attempted to address this shortcoming by proposing a hybrid approach between the two algorithms. This hybrid approach takes all the boundaries proposed by the NN segmentation approach that have an emission probability above a specific threshold, and fixes these before DP is performed. The DP then has the task of choosing between boundaries hypothesised by peaks with lower emission probabilities as well as between other higher probability peaks that were skipped by the NN segmenter. Figures 6 and 7 show the effect of this threshold on the DP cost and the average error respectively.

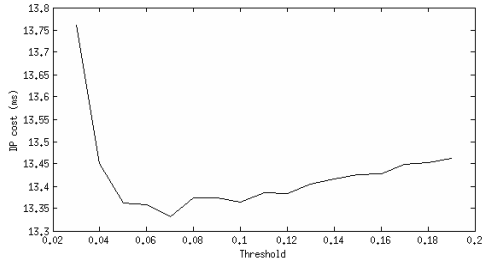


Figure 6: DP cost (as described in Section 4.2) as a function of the emission probability threshold for the combined method, measured on the development set.

Figures 6 and 7 show that segmentation accuracy is improved by including the new threshold. For the optimal threshold, Table 1 gives the segmentation performance achieved by the combined method on the core test set. There is a substantial improvement in terms of both DP cost and average error, indicating that the hypothesised boundaries are better aligned with the TIMIT boundaries and that more boundary pairs are within 20ms of one another.

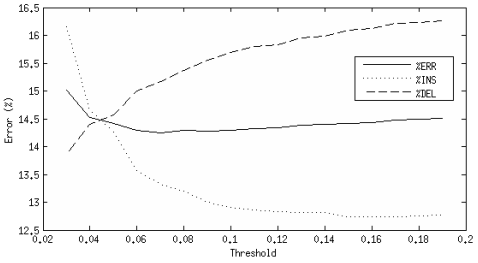


Figure 7: Average error (as described in Section 4.1) as a function of the emission probability threshold for the combined method, measured on the development set.

The segmentation of sentence dr6-fbch0-sa1 achieved by the combined method is shown in Figure 8. Both the insertion at the second 'r' and the deletion near the 'd' have been avoided. While this is merely a specific example, similar improvements were observed informally for many other utterances.

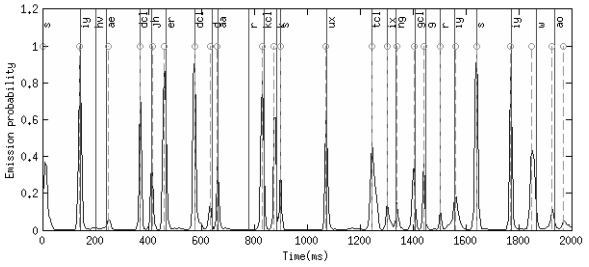


Figure 8: Segmentation results for the combined method for sentence dr6-fbch0-sa1.

7. Summary and conclusion

We propose an algorithm based on the principle of dynamic programming for the unconstrained automatic segmentation of continuous speech into phoneme-like units. A measure of segment boundary probability is computed by an MLP from a number of consecutive feature vectors. This is combined with a knowledge of the statistical distribution of the segment lengths within a dynamic programming framework to obtain an optimal placement of segment boundaries. We compare the performance of our algorithm with the performance of a recently proposed alternative, which applies MLPs, but not within a DP framework. For experimental evaluation, we measure how closely the hypothesised boundaries match the TIMIT phone boundaries. It was found that an improved alignment between the generated and TIMIT boundaries was achieved when employing the DP-based framework, and that a hybrid approach which combines aspects of both algorithms leads to even better results. We conclude that the incorporation of dynamic programming into speech segmentation algorithms is successful.

8. Acknowledgements

We gratefully acknowledge the financial assistance of the National Research Foundation (NRF) towards this research. Opinions expressed and conclusions arrived at are those of the authors and are not necessarily to be attributed to the NRF.

9. References

[1] J. Adell, A. Bonafonte, J. Gomez, and M. Castro, "Comparative study of automatic phone segmentation methods for TTS," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 1, 2005, pp. 309 – 312.

[2] M. Sharma and R. Mammone, "'Blind' speech segmentation: Automatic segmentation of speech without linguistic knowledge," in *Proceedings of the Fourth International Conference on Spoken Language Processing, ICSLP*, vol. 2, oct 1996, pp. 1237 –1240.

[3] L. ten Bosch and B. Cranen, "A computational model for unsupervised word discovery," in *Order: A Journal On The Theory Of Ordered Sets And Its Applications*, 2007, pp. 1 – 4.

[4] D. Wang, L. Lu, and H.-J. Zhang, "Speech segmentation without speech recognition," in *Proceedings of International Conference on Multimedia and Expo, ICME*, vol. 1, july 2003, pp. 405 – 408.

[5] V. Keri and K. Prahallad, "A comparative study of constrained and unconstrained approaches for segmentation of speech signal." in *Proceedings of International Conference on Spoken Language Processing, ICSLP*, 2010, pp. 2238–2241.

[6] S. Hoffmann and B. Pfister, "Fully automatic segmentation for prosodic speech corpora," in *Proceedings of the International Conference on Spoken Language Processing, ICSLP*, 2010, pp. 1389–1392.

[7] F. Malfrre and T. Dutoit, "High-quality speech synthesis for phonetic speech segmentation," in *Fifth European Conference on Speech Communication and Technology, EUROSPEECH, Rhodes, Greece, September 22-25*, 1997.

[8] Okko Räsänen, U. K. Laine, and T. Altosaar, "Blind segmentation of speech using non-linear filtering methods," in *Ipsic I. (Ed.): Speech Technologies*. InTech Publishing, 2011, pp. 105 –124.

[9] A. Sarkar and T. Sreenivas, "Automatic speech segmentation using average level crossing rate information," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 1, 2005, pp. 397 – 400.

[10] Y. P. Estevan, V. Wan, and O. Scharenborg, "Finding maximum margin segments in speech," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 2007, pp. 937 –940.

[11] G. Aversano, A. Esposito, and M. Marinaro, "A new text-independent method for phoneme segmentation," in *Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems, MWSCAS*, vol. 2, 2001, pp. 516 –519.

[12] K. Hatazaki, Y. Komori, T. Kawabata, and K. Shikano, "Phoneme segmentation using spectrogram reading knowledge," in *International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 1, may 1989, pp. 393 –396.

[13] H. Finster, "Automatic speech segmentation using neural network and phonetic transcription," in *International Joint Conference on Neural Networks, IJCNN*, vol. 4, jun 1992, pp. 734 –736.

[14] F. Malfrere, O. Deroo, and T. Dutoit, "Phonetic alignement : Speech synthesis based vs. hybrid HMM/ANN," in *Proceedings of International Conference on Spoken Language Processing, ICSLP*, Sidney, Australia, 1998, pp. 1571–1574.

[15] D. Toledano, "Neural network boundary refining for automatic speech segmentation," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 6, 2000, pp. 3438 –3441.

[16] K.-S. Lee, "MLP-based phone boundary refining for a TTS database," *IEEE Transactions on Audio, Speech amp; Language Processing*, vol. 14, no. 3, pp. 981–989, 2006.

[17] Y. Suh and Y. Lee, "Phoneme segmentation of continuous speech using multi-layer perceptron," in *Proceedings of Fourth International Conference on Spoken Language, ICSLP 96*, vol. 3, oct 1996, pp. 1297 –1300.

[18] A. K. Halberstadt, "Heterogeneous acoustic measurements and multiple classifiers for speech recognition," Ph.D. dissertation, Massachusetts Institute of Technology, MIT, 1998.

[19] L. F. Lamel, R. H. Kassel, and S. Seneff, "Speech database development: Design and analysis of the acoustic-phonetic corpus," in *Speech Input/Output Assessment and Speech Databases, SIOA*, vol. 2, 1989, pp. 1297 –1300.