On the automatic segmentation of transcribed words

Institute for Perception Research
P.O. Box 513 - 5600 MB Eindhoven

SP/sp 94/02
14.12.1994

Rapport no. 1023

On the Automatic Segmentation of Transcribed Words

Steffen Pauws, Yves Kamp and
Lei Willems

On the Automatic Segmentation of Transcribed Words

Steffen Pauws Yves Kamp Lei Willems

Contents

1 Overview **3**

1.1 Introduction . . . 3

1.2 Goal of the project . . . 3

1.3 About this document . . . 4

2 Summary **6**

3 Statement of the problem **8**

3.1 Segmentation for speech synthesis . . . 8

3.2 General formulation of the segmentation problem . . . 10

3.3 The speech inventory . . . 11

3.4 Performance assessment procedure . . . 12

3.5 Overview of Automatic Segmentation Approaches . . . 15

4 A Hierarchical Approach **19**

4.1 Introduction . . . 19

4.2 Broad Phonetic Classes . . . 22

4.2.1 Outline of the method . . . 22

4.2.2 Results . . . 27

4.2.3 Limiting Cases . . . 29

4.2.4 Discussion . . . 29

4.3 Sequence Constrained Vector Quantization . . . 31

4.3.1 Outline of method . . . 31

4.3.2 Results . . . 35

4.3.3 Contribution to hierarchical approach . . . 37

4.3.4 Limiting cases . . . 38

4.3.5 Variants . . . 40

4.3.6 Discussion . . . 44

4.4 Hidden Markov Models . . . 45

4.4.1 Outline of the method . . . 45

4.4.2 Results . . . 51

4.4.3 Variants . . . 55

4.4.4 Limiting cases . . . 56

4.4.5 Discussion . . . 57

5 **Discussion** 59
5.1 Performance evaluation 60
5.2 More broad phonetic classes 60
5.3 Finding centroids 60
5.4 Frame acquisition via glottal closure instants 60
5.5 Hidden Markov Model Structure 60
5.6 Starting form an orthographic transcription 61
5.7 Linear Discriminant Analysis 62
5.8 User interface 62

6 **Software Tools** 64
6.1 File formats 66
6.1.1 Label files 66
6.1.2 A priori knowledge files 67
6.1.3 Sampled speech data files 68
6.2 Assess 69
6.2.1 Function 69
6.2.2 Use 69
6.2.3 Implementation 71
6.3 BroPhon 72
6.3.1 Function 72
6.3.2 Use 72
6.3.3 Implementation 73
6.4 FrontEnd 75
6.4.1 Function 75
6.4.2 Use 75
6.4.3 Implementation 76
6.5 GenMarkov 77
6.5.1 Use 77
6.5.2 Use 77
6.5.3 Implementation 77
6.6 htk2ipo 78
6.6.1 Function 78
6.6.2 Use 78
6.6.3 Implementation 78
6.7 ipo2htk 79
6.7.1 Use 79
6.7.2 Implementation 79
6.8 Implicit 80
6.8.1 Function 80
6.8.2 Use 80
6.8.3 Implementation 81
6.9 Pitch 82
6.9.1 Function 82
6.9.2 Use 82
6.9.3 Implementation 83
6.10 SegKMeans 84

6.10.1 Function 84
6.10.2 Use 84
6.10.3 Implementation 86
6.11 SeqQuant 87
6.11.1 Function 87
6.11.2 Use 87
6.11.3 Implementation 89
6.12 Viterbi 92
6.12.1 Function 92
6.12.2 Use 92
6.12.3 Network specification 94
6.12.4 Bigram specification 97
6.12.5 Implementation 98
6.13 Small Tutorial 103

Abstract

Automatic speech segmentation according to a phonetic transcription is now commonly performed by the application of Hidden Markov Models (HMM). The segmentation results from a time-alignment of the utterance against a sequence of HMMs corresponding to the transcription. In order to allow HMMs to capture the intrinsic speech variability, they have to be tuned in a supervised way on a training set. Most approaches that try to fulfil these HMM training requirements resort to a manually segmented speech inventory. As we wished to develop a fully automatic segmentation technique, we avoid this prerequisite for manually segmented speech material and propose instead a hierarchical approach for acquiring a reliable training set. The preparation of training material in the first stage is accomplished by attributing local acoustic-phonetic properties to the utterance resulting into a broad phonetic class segmentation. Subsequently, a fairly accurate phonemic segmentation is obtained by segmenting each broad phonetic class segment into its constituting phonemes. The resulting segmented speech material serves for the HMM training procedure. The final segmentation is obtained by time-alignment of the utterances against the HMMs. The approach does not need external references nor prototypes of typical phonemes nor a collection of pre-segmented speech material.

By considering 50 phoneme-like HMMs, experiments on a Dutch speech database containing 827 isolated word utterances of a single male speaker have demonstrated that the difference between a manual segmentation and automatic segmentation is smaller than 20 ms for more than 88 % of the segment boundaries.

Preface

Finding time-aligned transcriptions of continuous speech in an automatic way has proven to be difficult. This fact is rather disappointing taking into account the clear need for large corpora of accurately transcribed speech.

Nowadays, the de facto Hidden Markov (HMM) modelling technique in automatic speech recognition (ASR) has brought adequate solutions for the translation of parametrized speech to linguistic units. If the sequence of these units is already known at recognition time, the application of HMMs is reduced from a recognition process to a segmentation process. The power of these statistical models now stimulates renewed interest in the segmentation problem. The incorporation of these relatively innovative ASR-instruments constitutes our main present effort toward the segmentation problem.

In fact, the presence of inspiring ASR techniques that may lead to big improvements in segmentation accuracy was not our primary motivation in tackling the segmentation problem. Actually, the quest of accurately transcribed speech material in all areas of speech research initiated a project that aims at research and development of automatic speech segmentation techniques. The acquired techniques can serve as tools for a wide range of disciplines within the speech research such as speech synthesis, speech perception, prosody research and speech recognition.

Diphone speech synthesis Incorporating a new voice or language in a speech synthesis system based on the concatenation of pre-recorded speech fragments such as diphones is a rather tedious and cumbersome procedure. Diphones have to be excised from carrier utterances and in order to prepare a new diphone database much recording, labelling and (manual) segmentation is required. The availability of a rapid, consistent and reliable technique that is free of human intervention for acquiring these segments from a new recording would alleviate problems involved in this preparation. This argument is the main motivation behind our project.

Prosody Accurately transcribed databases are necessary to obtain better speech production and perception models. Especially, the research in durational aspects of speech can profit from a method of fast detection and retrieval of speech segments that include specific phonetic contexts.

Automatic Speech Recognition Although speech recognition heavily relies on statistical models, these models do not usually come for free. A large amount of segmented speech data is needed in order to, at least, seed (bootstrap) these models.

Unfortunately, setting up large phonetically labeled speech databases for supporting speech research is a laborious task.

1. The manual time alignment procedure is time-consuming. Even positioning of the phoneme labels in a single three-syllable word takes several minutes on the average. It requires carefull listening along with waveform and spectogram interpretation. In addition, the repetitive work is very tedious, introducing human errors caused by loss of concentration.

2. Motivated expert acoustical phoneticians are required to skillfully perform the segmentation task. In fact, the mere prerequisite of reliable labelling of speech, which is, in first instance, performed on the basis of auditory impression, is a subjective and time-consuming task.

3. Each person has a different (and evolving) set of (subjective) criteria regarding the positioning of the segment boundaries. The variability across different experts and even within the same expert is not negligible, implying inconsistent results.

To summarize, many disciplines within the speech research can benefit from an automatic approach for acquiring an accurate time-alignment of a phonetic transcription with an utterance. This report presents a new hierarchical procedure for segmentation which is fully automatic in that it does not need any manually segmented learning data.

Chapter 1

Overview

1.1 Introduction

This report describes the results of a project aiming at research and development of automatic speech segmentation techniques and tools. It brings a solution to the segmentation problem that we have defined as finding the boundaries between the constituting phonemes of an utterance, relying only on its phonetic transcription.

It is suggested that the availability of such tools improves the overall quality of speech synthesis based essentially on the concatenation of well defined pre-recorded speech units known as diphones. Diphones are short speech segments which contain mainly the transition between two adjacent phonemes but also the last part of the left and the first part of the right phoneme. To date, they are extracted from short utterances via a tedious manual segmentation process. In this respect, a tool that automatically extracts the speech fragment is far more practical than manual segmentation process, since it reduces the amount of work and provides more consistent results. The reduction of work leads to an easier adaptation to a new speaker or language of a speech synthesis system that must cover a particular application. Consistent segmentation results may also produce smoother synthesized speech.

Besides speech synthesis, a wide range of disciplines within speech research involving speech perception, recognition, and production can benefit from an automatic segmentation tool. For instance, large speech database can automatically be provided with segmentation information that can serve as a facility for supporting the study of the temporal aspects of speech.

1.2 Goal of the project

This project tries to develop automatic segmentation tools for the excision of diphones from isolated utterances of words that are recorded for this purpose. However, it is mainly focused on the segmentation of phoneme-like units by time-aligning a phonetic transcription with the speech waveform. The reason is that knowledge of the phoneme-like unit positions in the waveform is easy to translate to diphone positions by e.g. applying *ad hoc* rules. In particular, the following topics have been addressed in the field of speech segmentation:

1. Improvement of existing speech segmentation tools at the Institute for Perception Research (IPO) in providing higher accuracy and in requiring less human intervention and

a priori knowledge. It is felt that higher accuracy can be reached by exploiting local (acoustic-phonetic) signal properties without using external references. Less human intervention can be reached by implementing a *fully automatic* procedure in which a minimum amount of *a priori* knowledge is used.

2. Incorporation of innovative speech recognition techniques such as Hidden Markov Modelling (HMM) technology. HMMs are statistical models that try to model the intrinsic variability of speech. Because of their statistical nature, their parameters have to estimated by a training process on a large collection of speech data. The application of HMMs for segmenting speech is now commonly used and a promising approach in terms of performance.

3. Incorporation of Vector Quantization techniques that try to compute jointly a set of segmentation points and a set of centroids characterizing corresponding segments. An attractive idea is to use adapting centroids by calculating them on-the-fly from the utterance at hand by means of some standard distortion measure while looking for an optimal segmentation.

1.3 About this document

This document describes the aspects of the project concerning the research and development of automatic segmentation techniques. In order to be accessible for all the readers we constructed this report by moving from general to particular. Chapter 1 and 2 are meant for an executive and managerial audience. Chapter 3, 4, and 5 give extra information needed to fully understand the proposed technique and the conducted experiments. The last chapter gives an overview of the software tools written in the course of the project.

Overview defines the purpose of the project by stating the problem and the need for automatic segmentation facilities.

Summary presents the main result of the project. Also, recommendatations for future research and conclusions are given.

Statement of the problem describes in full detail the need for speech segmention and the assessment method. Successively, we describe the segmentation requirements for a diphone database preparation, the segmentation problem in a formal and unified framework, the speech inventory at hand, and the performance assessment procedure. As concluding section, we present a literature survey about state-of-the-art segmentation techniques and experiments.

A Hierarchical Approach describes in full detail our proposed method for automatic segmentation. The hierarchical approach consists in a set of building blocks, each documented in a distinct section. In general, each section proceeds along the same lines:

- An outline of the method for each building block is given. Theoretical aspects needed to comprehend the method are intertwined with the description.
- The results achieved with that building block are presented along with its contribution to the hierarchical approach.

- Limiting cases are given to put the results in a proper perspective.
- Some implementation variants of the method are discussed and compared with our method.
- A discussion gives some reflections about the implementation, results, and direct recommendations.

Discussion presents in full detail the main results of the experiments, the implications for the project and recommendations for further research.

Software Tools presents the suite of software tools written in the course of the project. It is described for the benefit of users and future programmers.

Appendix A has a comprehensive list of mismatches between labelling and realization that are encountered during an examination of the speech inventory.

Chapter 2

Summary

A mix of conventional pattern recognition techniques and rather innovative speech recognition techniques is proposed in order to tackle the automatic segmentation problem. It consists in a hierarchical combination of these techniques. Software tools are written that implement this approach. The performance in segmentation accuracy is investigated on a speech inventory that was already available for the purpose of speech synthesis in Dutch. It consists of 827 isolated word utterances from a single male speaker. The whole inventory was manually segmented by considering 50 phoneme-like units.

We have strived to keep the required *a priori* knowledge for the segmentation process to a minimum by exploiting local acoustic-phonetic signal properties and by capitalizing on the following minimum knowledge: a phonetic transcription of the utterance, global duration constraints of phonemes, and the annotation of four acoustic-phonetic properties of phonemes, i.e. voicedness, unvoicedness, burstness, and silence. The system does not need external references nor prototypes of a typical phoneme nor a collection of pre-segmented speech material.

If we consider an interval of *20 ms* around a manually positioned segment boundary to be correct, the proposed hierarchical approach results into *88 %* of the phoneme boundaries positioned correctly in the speech inventory as compared with the manual segmentation. By considering an interval of *25 ms*, the accuracy is even *93 %* correct.

In order to put the methodology and results in a proper perspective we have assessed the present state-of-the-art segmentation by a literature survey. This survey demonstrates that our approach is rather unique in its accuracy performance and in the fact that it does not need pre-segmented speech material.

The hierarchical approach consists of

1. A conventional pattern recognition tool that attributes local acoustic-phonetic properties of the speech waveform. The output is a so-called broad phonetic class segmentation of the input utterances that provides reliable anchor points for subsequent detailed analysis.

2. A vector quantization-oriented tool that segments the waveform in phoneme-like units by considering only locally available spectral information and by capitalizing on a small set of required *a priori* knowledge. The resulting phoneme-like segmentation is guided by the broad phonetic class segmentation of the previous step.

3. A training tool that bootstraps phoneme-like Hidden Markov Models (HMM) using the

phoneme-like segmentation obtained in the previous stage. The training procedure is carried out by means of a Segmental K-Means algorithm.

4. A training tool that fine-tunes the initialized HMMs by subjecting them to a last training process and using only the transcription. This training tool iteratively re-estimates the parameters of the HMMs by means of a Baum-Welch reestimation. By this technique, HMMs will recover from possible time-alignment mismatches of the transcription introduced in earlier stages of the approach.

5. A speech recognition tool that is restricted to time-align the utterance with the HMMs. The transcription of the utterance determines the sequence of HMMs. In other words, the recognition process carried out by a conventional Viterbi decoding is reduced to a segmentation process.

Still, some major recommendations for future research can be made on the basis of the results. Research can be focused on an elaboration of the automatic segmentation techniques in several ways.

- Evaluating the segmentation performance on other speech inventories (Cross-validation)
- Starting the segmentation process from an orthographic transcription of the utterance.
- Exploiting Hidden Markov Models with distinct topology and structure for each phonetic class or having distinct models for different phonetic contexts.
- Refining frame acquisition via a pitch synchronous way.

Also, research can be directed to touch other topics that improve segmentation accuracy. Designing a user interface that nicely interacts with the tedious segmentation process, the preparation of speech fragments for speech synthesis, and the algorithms. Automatic segmentation is still affected by incorrect boundary placements. Much attention has to be paid how to repair these errors by means of audio and visual feedback facilities. Also, the preparation and 'polishment' of speech fragments that must be incorporated into a speech synthesis system will benefit from a professional work bench.

Chapter 3

Statement of the problem

3.1 Segmentation for speech synthesis

The quality, in terms of intelligibility and naturalness, of speech synthesis based on the concatenation of pre-recorded speech fragments involves many critical aspects. Beside the adequate application of knowledge concerning speech production and perception at a supra-segmental level (prosody), the purely acoustical characteristics of the speech fragments are predominant. In this respect, the segmental acoustic properties of the speech fragments are dependent on

1. the segmentation accuracy of the fragment.
2. the quality and sort of speech material from which the fragments are excised. This also includes speaker characteristics (timbre, dialect), recording protocol and conditions.
3. the coherence or proper acoustic matching of adjacent concatenated fragments in the synthesis process.

In the sequel, we will focus on the segmentation accuracy and ignore other quality influencing aspects.

One can choose from a range of fragments that could be used as building blocks for synthesis : word groups (phrases), isolated words, syllables and subsyllable units such as triphones or diphones. From an engineering point of view and without making any claims about the ultimate synthesis quality, the use of large units implies a loss of flexibility during the construction of a speech synthesizer. Large units do not efficiently support a large vocabulary and do not allow for a convenient extension possibility of the vocabulary without a repeated acquisition of new speech material. In contrast, small fragments such as diphones are prepared once and for all (for a given speaker) independently from the vocabulary or the application to be covered.

Diphones are defined as the speech segments spanned by the central (stable) part of a phoneme and the central part of the following phoneme. In other words, it spans the transitional speech portion (transient) between two phonemes and so captures the complex transition between speech sounds in precompiled chunks. Diphone boundaries are defined by the time marks in the carrier waveform from which the diphone will be excised. The marks indicate the diphone onset somewhere at the central part of the first constituting phoneme and the diphone end at the central part of the second constituting phoneme.

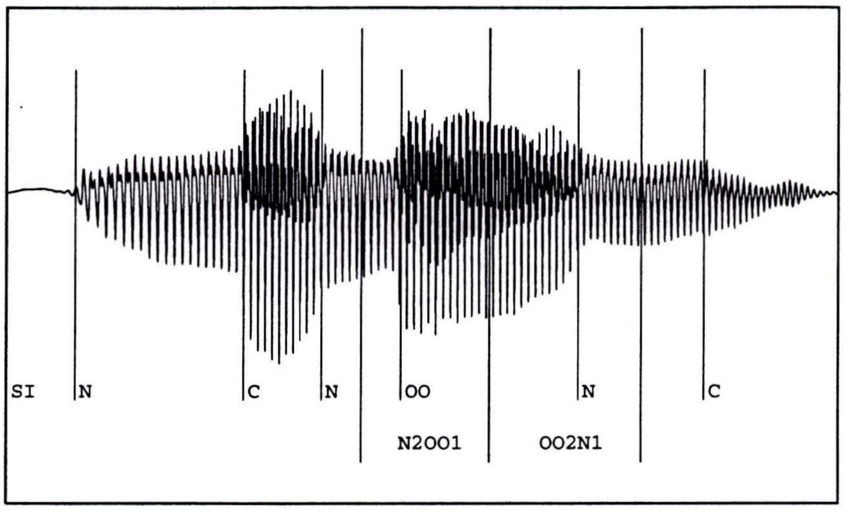

Figure 3.1: *Nonsense carrier word 'nenoone' for the excision of the diphones* N2OO1 *and* OO2N1.

In order to prepare a new diphone inventory, a speaker is instructed to read aloud a list of words out of which the diphones will be excised. In general, vowel-consonant (V-C) and consonant-vowel (C-V) diphones are embedded in an isolated three-syllable nonsense carrier word composed of the diphone of interest positioned in a stressed syllable and surrounded by some neutral phonetic context. Diphones involving schwa are positioned in unstressed syllables. In cases when a so-called silence is one of the demi-phonemes, the diphone is a word initial or final. It is felt that the use of isolated nonsense words remedies the problem of the diphones being polluted by their prosodic context of a carrier. As shown in Figure 3.1, the nonsense word 'nenoone' carries the C-V diphone N2OO1 and the V-C diphone OO2N1. On the other hand, vowel-vowel (V-V) and consonant-consonant (C-C) diphones are excised from lexical words in order to avoid unnatural articulation effects. Syllable boundaries within a diphone could also be included by a proper selection of the carrier. There are almost as many recordings of carrier words needed as there are different diphones required in the synthesis system.

The preparation of diphones for a synthesis system requires much labelling and segmentation work. First of all, diphone boundaries are required to excise the unit from the carrier word. The diphone boundaries can be found by different strategies. In principle, one could *directly* find them by exploiting some notion of stationary characteristics at the center of a phoneme. However, one usually starts by computing the sequence of phoneme boundaries by time-aligning a phonetic transcription with the speech waveform. Giving this time-alignment, it is easy to excise the desired diphone by applying explicit *ad hoc* rules or by using some spectral resemblance measure [Boëffard 92, Hemert 85, Hemert 87].

In addition, the phoneme boundary (between the two demi-phonemes) of the extracted diphone must also be specified to serve as anchor point for prosodic postprocessing in a synthesis system. Finally, in case of synthesis systems based on waveform manipulation techniques (PSOLA), the prosodic post-processing also necessitates a precisely specified placement of pitch instants.

3.2 General formulation of the segmentation problem

Several techniques for speech segmentation are already available and are described in a review paper[Vidal 90]. We propose here our own hierarchical combination of conventional and more innovative pattern recognition techniques to obtain boundaries at the phoneme level.

Obviously, the segmentation process needs some *a priori* knowledge in order to perform its task. In our method we have strived to keep the amount of a priori knowledge to a strict minimum: it consists in the phonetic transcription, some global duration statistics of realizations of phonemes and phonetic characteristics such as burstness, voicedness or unvoicedness of particular sounds. Other particularly interesting phonetic properties such as spectral differences between sounds will be *implicitly* acquired in a statistical framework by training.

The utterance is supplied by a sampled speech waveform which, in its raw form, is not directly suitable for the application of segmentation algorithms. An adequate description is a sequence of acoustic vectors where each vector characterizes the speech signal over a small time frame (typically 20 ms) at a low rate (typically 10 ms). This frame-by-frame analysis results in a so-called discrete observation sequence.

Observation sequence $\mathbf{o}(t) = (o_1(t), \cdots, o_p(t))'$ is the observed acoustic vector at frame t with $1 \leq t \leq T$; T is the number of speech frames; p is the number of vector elements. $\mathbf{O}_i^j = \mathbf{o}(i), \cdots, \mathbf{o}(j)$ is a *partial* sequence of $(j - i + 1)$ acoustic vectors extending from frame i to frame j. $\mathbf{O}_1^T = \mathbf{o}(1), \cdots, \mathbf{o}(T)$ is the *complete* observation sequence that generally represents a whole realization of an utterance.

There are numerous possibilities to characterize a speech frame spanning a short interval of about 20 ms. For example, one can describe the energy distributions over a pre-determined set of frequency bands (filterbank analysis). Considering only instantaneous values related to a single frame reduces too much the amount of information provided to the segmentation algorithm. Temporal changes in the speech spectrum which span several frames also contribute very useful cues. One can incorporate these changes into the vectors by estimating time-derivatives from instantaneous values of vectors that are some frames apart (see Section 4.4.1).

This preprocessing stage implies that the boundaries can only be indicated as frame numbers in the input observation sequence and the accuracy is thus inherently limited by the frame rate.

We denote the sequence of boundaries by the set of integer $\mathcal{B} = \{b_0, b_1, \cdots, b_L\}$ in which L is the pre-determined number of segments (labels). As already said, the segmentation is dictated by a phonetic transcription (sequence of phonetic labels). A segment denoted as label l starts at frame $b_{l-1} + 1$ in the observation sequence and ends at frame b_l. In order to increase readability, we denote each label by an integer. In reality, it corresponds to a segment label from a finite set.

Some constraints have to be satisfied in order to obtain a valid segmentation result.

1. The beginning and ending points of the sampled speech data file are given and fixed, i.e. $b_0 = 1$ and $b_L = T$.

2. The sequence of boundaries $\mathcal{B} = \{b_0, b_1, \cdots, b_L\}$ is strictly increasing, i.e. $b_{l-1} < b_l, l = 1, 2, \cdots, L$.

3. Obviously, the number of phonetic labels never exceeds the number of observation vectors, i.e. $L \leq T$.

This formulation of the segmentation problem is purposely stated in general terms to provide a common framework for each stage of our hierarchical approach. The precise setting for each stage will specified later.

3.3 The speech inventory

For the purpose of speech synthesis in Dutch, a speech inventory was available consisting of 827 carrier words for a pre-selected number of diphones. Among these, 522 words are three-syllable nonsense words and 305 words can be considered as real lexical words. All utterances were recorded from a single male speaker. Initially, they were digitized by means of an in-house VAX sampled data format with a 8kHz sampling frequency and 12-bit precision. For this project, the inventory was converted to the Apple Interface File Format (AIFF) with a 8kHz sampling frequency and dynamically scaled to 16-bit precision. All speech material is transcribed and segmented by hand. The database transcription and labelling convention follows the SAM-Phonetic Alphabet definition [ESPRIT 92], a system for intra-language phonemic representation. The tables hereunder provide a brief outline of the phonemes with an example word and its SAM-PA notation together with the more computer (keyboard) oriented IPO notation we have used as the phoneme label names thoughout the project.

Besides, diphtongs are divided into a segment corresponding to the relatively low vowel and a segment corresponding to the fast movement up to the relatively high vowel. The unvoiced plosives are split into a closure and a burst. Voiced plosives are divided into a blählaut and a burst. Taking this into account, the total number of distinct segments is equal to 50. The number of phoneme occurrences is equal to 7239. The number of hand-positioned phoneme boundaries in the database is equal to 6412 (without counting the beginning and ending of the sampled speech data files).

'Checked' vowels			Monophtongs			'Potential' Diphtongs		
SAM-PA	IPO	example	SAM-PA	IPO	example	SAM-PA	IPO	example
A	A	m**A**t	i	II	l**IE**p	e:	EE	l**EE**s
@	C	d**E**	y	Y	f**UU**t	2:	OE	k**EU**s
Y	CC	p**U**t	u	U	r**OE**t	o:	OO	r**OO**d
E	E	l**E**s	a:	AA	m**AA**t			
I	I	p**I**t						
O	O	r**O**t						

'Essential' diphtongs			Loanword vowels		
SAM-PA	IPO	example	SAM-PA	IPO	example
Ei	EI1 EI2	r**EI**s	E:	EH	mayon**AI**se
9y	UI1 UI2	m**UI**s	9:	UH	fr**EU**le
Au	AU1 AU2	k**OU**d	O:	OH	z**O**ne

Unvoiced fricatives			Voiced fricatives			Silence
SAM-PA	IPO	example	SAM-PA	IPO	example	
f	F	**F**ok	v	V	**V**eer	SI(#)
s	S	**S**ok	z	Z	**Z**eer	
S	SJ	**SJ**aak	Z	ZJ	**J**ournaal	
x	X	**G**ok				

Unvoiced plosives			Voiced plosives			Semi vowels		
SAM-PA	IPO	example	SAM-PA	IPO	example	SAM-PA	IPO	example
k	K1 K2	**K**as	b	B1 B2	**B**as	h	H	**H**ang
p	P1 P2	**P**as	d	D1 D2	**D**as	j	J	**J**an
t	T1 T2	**T**as	g	G1 G2	**G**oal	w	W	**W**ang

Nasals			Liquids		
SAM-PA	IPO	example	SAM-PA	IPO	example
m	M	**M**eer	l	L	**L**ang
n	N	**N**eer	r	R	**R**ang
N	Q	ba**NG**			

The segmentation and labelling was carried out by means of the in-house graphical speech processing system GIPOS (Graphical Interactive Processing Of Speech) that has sophisticated audio and visual feedback facilities. The native segmentator was a non-expert phonetician but was supervised by a second more phonetically experienced person. The labelling was given beforehand by a third phonetically experienced person. The tedious work was executed in ± 50 hours of effective worktime. The segmentation procedure was conducted along the following guidelines that closely resemble proposed segmentation rules [Cosi 91]:

- Boundaries were placed at significant events (e.g. boundaries in a voiced region were positioned near the first zero crossing of a pitch period).
- Unambiguous boundaries of labels were placed by means of visual waveform and audio inspection. In some ambiguous cases, an audio 'window' was placed at one side of a hypothesized boundary in order to extract the segment from its phonetic environment. The window was iteratively extended towards the hypothesized boundary until the next or previous segment was clearly perceived. If this point clearly divided both contiguous segments on auditory grounds, it was considered as a boundary.
- Spectral information was only utilized in cases of uncertainty (e.g. liquid-vowel transitions, transition mark within diphtongs).
- In cases of persistent uncertainty, a second opinion from the supervisor ultimately forced the decision.
- Special attention has been paid to check whether the phonetic transciption neatly corresponded to the acoustic realization. Many phonological processes such as assimilation were encountered. They are enumerated in Appendix A.

Some duration statistics can be drawn for this manual segmentation and are shown in Table 3.1. Although expected, it is apparent that the loanword vowels EH, UH, and OH are exotic. There is an excessive number of the vowel schwa C due to its constant appearence in the neutral phonetic contexts of the nonsense words. Moreover, durational aspects of the labels show rather irregular patterns. This is expressed by the large standard deviations and the big differences between the minimum and maximum durations.

3.4 Performance assessment procedure

The quantitative evaluation of an automatic segmentation procedure is difficult. A reasonable approach is to point out the differences with a manually segmented reference set. The

Label	No. Occ.	Ave. Dur. ± Std. Dev.	Min. Dur.	Max. Dur.
I	44	86.0 ± 20.3	33.0	139.2
E	79	88.9 ± 18.8	58.4	145.2
A	125	90.3 ± 22.6	12.1	142.6
O	73	96.4 ± 24.1	53.0	180.0
CC	99	89.3 ± 22.0	45.5	176.6
C	1117	95.5 ± 23.7	21.1	188.9
II	96	113.2 ± 28.3	28.0	183.8
Y	41	118.2 ± 38.2	58.6	230.0
U	71	99.9 ± 27.6	40.0	190.0
AA	134	168.8 ± 34.9	96.5	261.0
EE	116	143.9 ± 39.1	74.0	247.2
OE	33	175.4 ± 29.3	113.4	239.2
OO	77	160.0 ± 44.5	76.1	259.2
EI1	60	101.8 ± 18.3	58.1	146.4
EI2	60	85.3 ± 26.8	26.8	150.5
UI1	70	100.8 ± 22.5	64.5	181.0
UI2	70	70.5 ± 27.5	26.2	150.0
AU1	34	121.7 ± 23.0	78.6	173.4
AU2	34	83.1 ± 26.8	30.0	134.8
EH	14	173.8 ± 23.5	143.4	221.4
UH	3	218.9 ± 23.2	195.1	241.5
OH	6	178.4 ± 15.4	163.4	201.0
P1	114	60.6 ± 22.7	12.4	135.0
P2	114	26.6 ± 17.5	10.0	112.0
B1	94	61.0 ± 22.1	20.0	165.1
B2	94	17.1 ± 5.3	7.5	36.6
T1	287	56.1 ± 20.9	12.9	145.2
T2	289	37.8 ± 24.9	14.8	155.0
D1	159	61.2 ± 33.0	20.0	231.1
D2	159	19.9 ± 6.0	9.1	66.0
K1	163	55.0 ± 18.1	18.5	122.2
K2	163	39.7 ± 17.0	15.1	145.8
G1	74	50.6 ± 20.4	14.1	147.2
G2	74	24.7 ± 7.8	11.2	43.5
F	133	85.2 ± 31.7	34.5	257.4
V	83	73.8 ± 25.0	32.5	164.6
S	196	130.1 ± 62.5	39.1	306.6
Z	75	93.3 ± 26.5	56.6	187.0
X	123	99.4 ± 30.1	35.2	226.5
H	93	60.0 ± 18.3	14.6	108.1
ZJ	65	108.2 ± 32.1	53.9	213.0
SJ	94	115.4 ± 36.0	52.1	293.1
M	124	77.7 ± 27.5	21.5	184.6
N	179	88.6 ± 35.2	31.2	207.2
Q	88	118.2 ± 49.1	40.4	380.1
L	200	82.6 ± 31.4	20.6	175.1
R	171	74.4 ± 30.9	14.2	189.9
W	128	71.5 ± 26.4	18.5	164.0
J	93	86.9 ± 24.2	45.4	135.6
SI	1154	59.9 ± 23.6	3.9	233.9

Table 3.1: *Duration statistics (ms) of manually segmented labels in the speech inventory.*

segmentation discrepancies can be objectively expressed by considering an interval round the reference boundary as correct, the so-called correct margin. By counting the automatically obtained boundaries that fall within that margin, one acquires the segmentation accuracy in terms of absolute values and percentages. By extending the margin typically by multiples of 10 ms, one obtains cumulative statistics that reveal the persistent discrepancies of troublesome boundaries. The statistics can be displayed in tables or graphs (e.g. bar charts).

1. By considering all phoneme-to-phoneme boundaries, a quick global overview is obtained that may express the overall accuracy of the segmentation procedure.
2. By grouping all phonemes in a smaller set of phonetic categories (e.g. vowels, fricatives), a clear abstracted overview is given that may give better evidence to possible common tendencies.
3. By considering each particular phoneme-to-phoneme boundary separately, a detailed inspection of boundary placement discrepancies is given that may help to refine and fine-tune the segmentation procedure.

In the sequel, we have standardized a correct margin of 20 ms round the reference boundary as reliability criterium. All boundaries associated with the 50 phoneme-like units are taken into account. Obviously, the first and last boundary associated with the beginning and ending of the utterance are left out.

The deviation of 20 ms is justified by several arguments about discrepancies across manual segmentations.

1. Several investigations involving the reliability across interindividual manual segmentations revealed that still only a percentage of 88-90 % was considered similar within a correct margin of 20 ms by comparing four manual segmentations in all possible combinations. This experiment was conducted with an isolated word segmentation task performed by four experts [Cosi 91].
2. Two manual segmentations of a French diphone inventory performed by one and the same expert show that a large number of boundaries would disagree if the correct margin was reduced below 20 ms.
   [Boëffard 92] In other words, it is not reasonable to expect that an automatic segmentation technique performs more accurately than the discrepancies involved with two segmentations performed by the same human expert.
3. Many researchers to date express their results in terms of this margin.
   (e.g. [Alphen 92, Angelini 93, Brugnara 92, Farhat 93])

When the ultimate goal (of segmentation) is speech synthesis, the overall quality of the automatic segmentation procedure can only be assessed by performing subjective listeners' tests for comparing the automatic and the manual way of preparing diphone inventories. It must be emphasized that from this point of view only the segmentation results around the acoustic unit of interest of the carrier word are important for the overall synthesis quality. The proper segmentation of the bearer phonemes constituting the phonetically neutral context of the carrier words do not (directly) contribute to the overall synthesis quality. It can indirectly affects the segmentation process because misalignments of one label may affect the positioning of neighbouring labels. Although we are developing an automatic segmentation strategy that will serve as a diphone inventory preparator, we take all boundaries into account.

3.5 Overview of Automatic Segmentation Approaches

A complete review of the state-of-the-art clearly falls beyond the scope of this report. On the other hand, some reference is needed to put our methodology as well as our results in a proper perspective. Therefore, we give hereunder a selection of recent representative papers on the segmentation by means of Hidden Markov Models (HMM) together with short comments on the method and the results. It is difficult (not to say impossible) to compare different systems by the lack of a formally unified assessment methodology that prescribes data base requirements, the set of objective assessment criteria, and the testbed. In addition, it is often difficult to retrieve the essential points of a segmentation method and the exact experimental conditions from short publications in conference proceedings. Nevertheless, we specify the segmentation accuracies as they were found in the literature in perspective with the training and test set. The main factors that may influence the segmentation accuracy and which highly differ from application to application are language, speaker, training and test database, recording conditions (including sample frequency and precision), and the set of speech units to be segmented. Some main conclusions can be drawn from this literature study:

1. The Hidden Markov Models are now commonly used for automatic segmentation troughout the speech community.
2. It is illustrative that an approach by exploiting the simplest from of HMMs, namely the context independent phoneme-like models, results into the highest segmentation accuracy compared with more elaborate models in which the left and/or right phonetic contexts of a phoneme are incorporated.
3. Most approaches heavily rely on a big amount of *a priori* knowledge by resorting to a collection of manually segmented data. In that sense, our approach clearly stands out from the list and has a rather unique position, since our method is fully automatic.

AT & T Bell - New Jersey, U.S.A. (Ljolje et al.) [Ljolje 91]

A set of HMMs was constructed for 47 phonemes with as wide triphone coverage as possible with the available training data. The training set consisted of two third of 5084 manually segmented sentences from the TIMIT-DARPA speech database [Lamel 87] recorded from a big collection of speakers. The test set was the remaining one third.

Two third of the manually segmented data was used to train some models which were applied for automatically segmenting all speech material. As a result the manually segmented as well as the automatic segmented data can be pairwise combined for training 4 different ultimate sets of HMMs. Results with the four training strategies were all quite comparable and resulted into a segmentation accuracy of *80 %* correct within a margin of *17 ms.*

CSTR - Edinburgh, U.K. (Taylor et al.) [Taylor 91]

Context-independent phoneme-like HMMs were trained on a training set of 400 manually segmented British English isolated words with at least 10 phoneme occurrences from one speaker. Another 400 British English nonsense words recorded from the same speaker were used as test set. The segmentation accuracy was *95 %* correct within a margin of *30 ms* and *98 %* correct within *40 ms.*

CSTR - Edinburgh, U.K. (Schmidt et al.) [Schmidt 91]

Context-independent phoneme-like HMMs were trained on a training set of 200 English sentences covering almost all permissible demi-syllables in English. The test set which was completely different from the training set contained 260 English sentences with 5780 phonemes. The segmentation accuracy was *50 %* correct within a margin of *12 ms*, *78 %* correct within *20 ms*, and *95 %* correct within *40 ms*.

CNET France Télécom - Lannion, France (Boëffard et al.) [Boëffard 92, Boëffard 93]

The HMMs were divided into two classes: models that represent so-called useful phonemes capturing the diphone sought, and models that represent the neutral phonetic context of the nonsense word (logatome). The training procedure consisted of three stages in which the models were enhanced in topology and structure at each stage. They were trained on a whole diphone inventory (± 1400 words) without any manually segmented material. Segmentation accuracy by using two French, one German, and one Spanish diphone database were nearly similar. The segmentation accuracy was in the range of *85-90 %* correct within a margin of *30 ms* for the four databases.

IFA - Amsterdam, the Netherlands (van Alphen) [Alphen 92]

The whole experiment was conducted by using a speech inventory of three repetitions of 100 Dutch sentences and 10 extra utterances in order to evaluate the segmentation accuracy. All utterances were spoken by a male speaker. The total number of 310 utterances were divided into an initialization set, a training set, and a test set.

The set of 39 context-independent phoneme-like discrete HMMs were bootstrapped on an initialization set of 34 Dutch sentences spoken 3 times. These (3×34) 102 utterances were manually segmented. A subset of 74 utterances was used for creating multiple codebooks.

The HMMs were subsequently trained on a collection of 100 sentences spoken 3 times. The initialization set was also included in this process but no manual segmentation information was utilized.

For testing purposes 10 extra sentences were recorded (these sentences were also present in the initialization set). The segmentation accuracy was *78.8 %* correct within a margin of *20 ms*.

IRST - Povo di Trento, Italy (Brugnara et al.) [Brugnara 92]

A set of 48 context-independent phoneme-like HMMs was applied to a subset of the TIMIT-DARPA speech database[Lamel 87]. The training set was constructed by selecting 64 speakers, each uttering 8 sentences ($64 \times 8 = 512$). The test set consisted of 24 speakers, not included in the training set, each of them uttering 8 sentences ($24 \times 8 = 192$). A distinct feature of this investigation is that the authors also examined the situation where only the orthographic transcription (but not the phonetic transcription) of utterances was available. This reduces even further the amount of a priori knowledge since the same orthographic transcription may result in quite different phonetic realizations. In such a way, the problem amounts to finding both segmentation and phonetic labelling of the utterance. In addition, three distinct training strategies were examined:

1. providing labelled and manually segmented speech material

2. providing labelled speech material
3. providing no information whatsoever

This gave rise to 6 different configurations of the system by considering all possible combinations of training and segmentation strategies.

In the case where the labelled and the manually segmented material was provided during training and where the segmentation was dictated by the labelling, the accuracy was *86.9 %* correct within a margin of *20 ms*.

A drastic performance reduction was observed when only the labelling of the speech material was provided during the training phase (*75.6 %* correct within *20 ms*).

When both labelling and segmentation had to be performed automatically, a slight error increase was observed in the two configurations mentioned above. The use of no labelling or segmentation information during both training and segmentation did not further influence substantially the performance.

It was concluded that exploiting manually segmented data was crucial for the segmentation accuracy. The training set was varied in its size (64, 128, 256, and 512 sentences) by still covering all 64 speaker in order to investigate the minimum *a priori* training requirements. No significant changes were observed when reducing the training size from 512 to 256. It was found that 128 sentences were a good compromise between training size and performance. Further reducing the size to 64 seemed to be inefficient.

IRST - Povo di Trento, Italy (Angelini et al.) [Angelini 93]

The context-independent phoneme-like HMMs were trained on 256 manually segmented sentences of TIMIT-DARPA speech database [Lamel 87]. The sentences were recorded from distinct speakers. The test set consisted of 192 sentences recorded from 24 speakers and was not included in the training set. The segmentation accuracy was *88.3 %* correct within a margin of *20 ms*. An extension of the size of the training set did not contribute to a higher segmentation accuracy. Moreover, a performance of *88.7 %* correct within a margin of *20 ms*, given a training set of 512 manually segmented sentences, was considered to be a limiting treshold in accuracy.

A set of 66 context-independent phoneme-like HMMs were trained on an Italian database consisting of 152 manually segmented phonetically rich sentences recorded from several speakers. The test set consisted of 48 sentences. The segmentation accuracy was *91.1 %* correct within a margin of *20 ms*.

IRIT - Toulouse, France (Farhat et al.) [Farhat 93]

The HMMs were divided in ten distinct phonetic classes, each having a different topology. The training set consisted of 23 manually segmented French sentences spoken by one speaker and manually segmented digits pronounced by ten other speakers. The test set consisted of the same 23 sentences, but spoken by another speaker. Different training strategies were carried out for obtaining context-independent, left-context dependent, right-context dependent, or triphone models. However, the highest segmentation accuracy was obtained by the simple context-independent HMMs and resulted into *75 %* correct within a magin of *20 ms*.

AT & T Bell - New Jersey, U.S.A. (Ljolje et al.) [Ljolje 93]

This research was conducted using different training strategies in order to obtain context-dependent as well as context-independent models. A set of 1158 manually segmented sentences, spoken by a single male speaker, was provided in order to train some models which are applied for automatically segmenting 50 entirely different phonetically balanced sentences from the same speaker and the 1158 sentences themselves. As a result the manually segmented as well as the automatic segmented data can be pairwise combined for training 4 different ultimate sets of HMMs. The test set consisted of the above mentioned 50 sentences. However, the highest segmentation accuracy was obtained by using the simple context-independent models and training the models with only the manually segmented data. The segmentation accuracy was more than *80 %* correct within a margin of *11.5 ms.*

Chapter 4

A Hierarchical Approach

4.1 Introduction

Our main interest and effort is to incorporate de facto speech recognition techniques into the segmentation process. The application of Hidden Markov Models (HMM) is a promising approach in terms of performance in which the segmentation is merely a by-product of the modelling of each phoneme-like unit by an HMM and of Viterbi decoding. Unfortunately, an accurate time-alignment of the utterance against a sequence of HMMs now shifts the problem to finding a set of reliable HMMs. Usually, HMMs are obtained by a maximum likelihood training procedure on a collection of utterances but a severe problem in this setting is the need for a good initialization (bootstrapping) of the models before they are further tuned on the training data.

Many researchers propose a solution in which they resort to a subset of pre-segmented speech from the same recording for HMM initialization (e.g. [Angelini 93, Ljolje 91, Ljolje 93]). Another way around is to initialize HMMs by using some pre-segmented material from an entirely different collection[Taylor 91]. However, this still requires the tedious hand-segmentation work that we want to get rid of. Also, the question arises about the amount of manually segmented material required for a new database to be effective in obtaining high segmentation accuracy. At the other extreme, HMMs can be initialized by a so-called 'linear segmentation' of the recorded corpus where the frames of the utterance are uniformly attributed to the states of the HMM model. This, however, yields poor quality models.

For obvious reasons we did not want to use any of these methods and we felt that the difficult initialization problem could be circumvented by a hierarchical approach. As can be seen in Figure 4.1, each stage in this approach needs a transcribed speech database (in sampled data or parametrized form). The hierarchy is basically structured by firstly segmenting the utterance in three broad phonetic classes (voiced, unvoiced, and silence) relying exclusively on the phonetic transcription and traditional statistical pattern recognition techniques. Its outcome is a broad phonetic class segmentation that provides robust anchor points for more detailed analysis and serves as a preamble for a next stage consisting of a Sequence Constrained Vector Quantization (SCVQ). This stage tries to segment each broad phonetic class region into its constituting phoneme-like units delivering a 'crude' phoneme-like segmentation. In other words, it tries to cluster the observation sequence into a pre-selected number of non-overlapping phoneme-like segments by minimizing a spectral distortion measure. The constraints to be satisfied are that the segments are contiguous in time, that the number

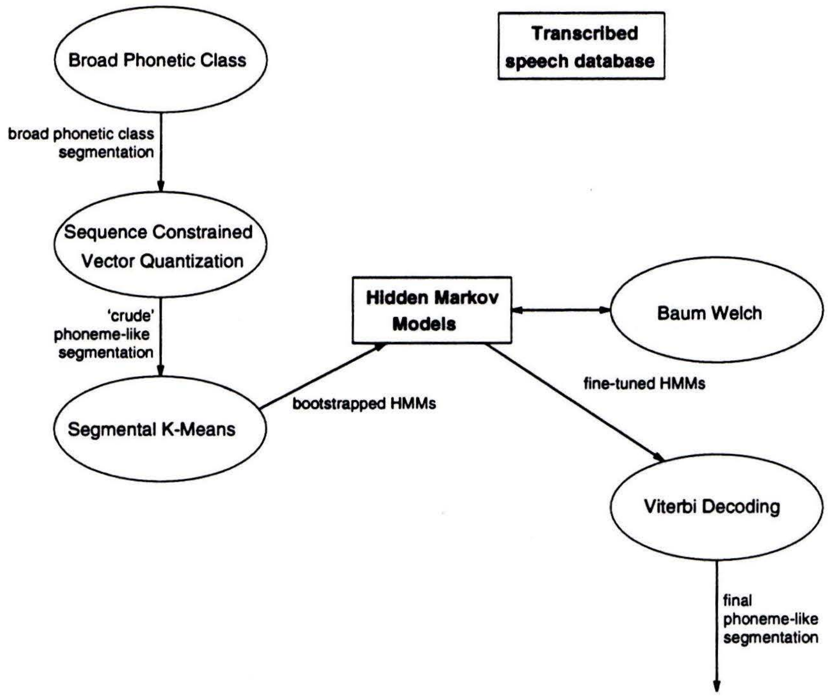
Broad Phonetic Class
Transcribed speech database
broad phonetic class segmentation
Sequence Constrained Vector Quantization
'crude' phoneme-like segmentation
Hidden Markov Models
Baum Welch
fine-tuned HMMs
bootstrapped HMMs
Segmental K-Means
Viterbi Decoding
final phoneme-like segmentation

Figure 4.1: *Flow diagram of the Hierarchical Approach. All processes (ellipses) need as input a transcribed speech data base.*

of segments is dictated by the phonetic transcription and that the segment locations should be fully compatible with the broad phonetic class segmentation obtained in the previous stage. Subsequently, these rudimentary segments obtained by this SCVQ stage are used as bootstrapping data for the HMMs. This supervised HMM initialization is conducted by the Segmental K-Means algorithm. Finally, fine-tuning of the HMMs is done by a Baum-Welch training by exploiting the whole speech inventory without using segment information. In order to obtain the final segmentation result, the Viterbi decoding algorithm is applied on the same speech inventory by using the transcription and the fully trained HMMs. It will be shown that this hierarchical approach will lead to an incremental refinement of the boundaries in which each stage has an effective contribution to the whole.

The *a priori* knowledge that must be 'fed' to the system is kept to a minimum; the whole process only requires the phonetic transcription, a mapping from which the broad phonetic class of each phoneme-like unit can be derived, and some very general duration statistics for each phoneme (see Table 3.1).

Beside the training requirements as mentioned above, other aspects that influence the accuracy must not be underestimated. In order to ensure a proper characterization of phonological processes within and between phonemes, they have to be adequately covered by the training material and occur in distinct phonetic contexts. In our application, it is hardly possible to address this prerequisite due to the structure of the speech inventory. The diphone carrier words are not phonetically rich and balanced. Rather, the acoustic units of interest are embedded in phonetically neutral contexts that have only a limited amount of phonemes (the neutral schwa vowel is predominant) resulting in an unbalanced phoneme-phoneme transition distribution. It may bias the HMM training and subsequently deteriorate the accuracy performance of the segmentation.

We have tried to convey a coherent picture of the distinct stages involved in the hierarchical approach. Each stage comes up with specific pecularities, possibilities and parameter settings leading off to side-ways that may not be interesting in a first instance. Thus, we have followed a main thread throughout this chapter by devoting a separate section for each stage within the hierarchical approach. First in each section, a detailed outline of the method of that stage is given intertwined with theory to comprehend the material. Second, the achieved experiment results are given along with their contribution to the hierarchical approach. Third, limiting cases in the sense of no-knowledge and full-knowledge instances of the technique are given to put the results in a proper perspective as well as implementation variants or small extensions of the technique. These sections called 'Variants' and 'Limiting Cases' can be skipped in a first reading.

4.2 Broad Phonetic Classes

In order to incorporate acoustic-phonetic knowledge into the hierarchical approach, a broad phonetic class (BPC) segmentation is used as preamble for subsequent processing. The objective is to provide reliable anchor points for a more detailed analysis. A similar approach is followed by some researchers who want to provide a robust description of significant acoustic-phonetic events of the utterance before it will be subjected to a next stage [Leung 84]. In an ideal case, these broad phonetic class segments have only to be divided in segments corresponding to their constituting phoneme-like units by a next stage.

4.2.1 Outline of the method

A straightforward and reasonable approach is to use a representation of the utterance into three broad phonetic classes,

1. *silence* (`SIL`), where no speech waveform is present,
2. *unvoiced* (`UNV`), where the speech waveform is aperiodic or random in nature,
3. *voiced* (`VOI`), where the speech waveform is quasi-periodic.

Our method is to exploit conventional statistical pattern recognition techniques which receive as input an observation sequence representing speech signal measurements. These measurements must provide a basis for distinguishing the three broad phonetic classes. A wide variety of measurements are candidates for this role. However, distributions of measurements show substantial overlap in their voiced and unvoiced regions. In a practical setting, the use of one measurement alone does not always come up with an unambiguous decision. The simultaneous utilization of several distinct measurements may alleviate this sensitivity for overlap.

We propose five distinct measurements that are simple to compute and roughly characterize the proposed classes. They all try to give a fairly good indication of the location of the dominant portion of the spectral energy of the speech sound. First of all, we take into account the fact that unvoiced sounds are generally characterized by an energy concentration in the relatively high frequency region, and voiced sounds in the relatively lower frequencies. Thus, these measurements can be effective in giving cues for discriminating voiced and unvoiced regions of speech. In addition, an energy level measurement is primarily used to distinguish speech from silence.

The observation vector consists of the following five measurements:

1. Normalized short-time energy E_N.
2. Normalized low-frequency energy E_{low} in the range 50-1200 Hz.
3. Normalized high-frequency energy E_{high} in the range 2000-4000 Hz.
4. Zero crossing rate Z_N.
5. First LPC coefficient a_1.

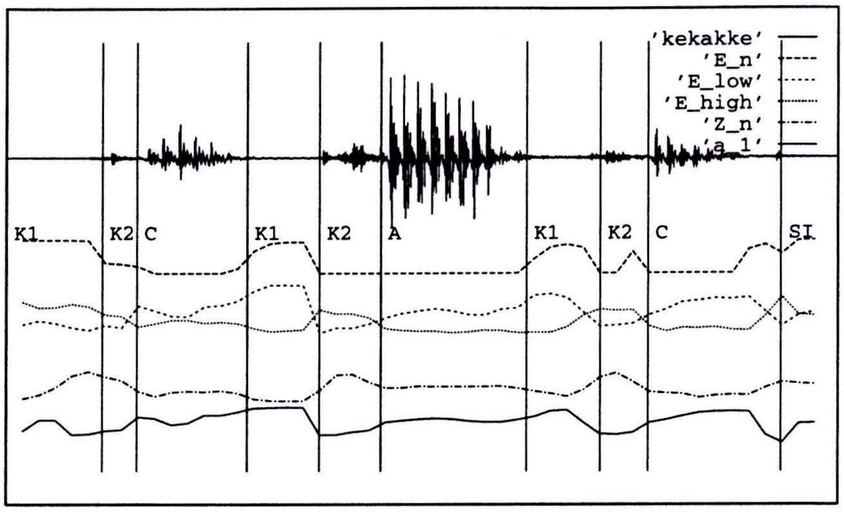

Figure 4.2: *Five measurements for the nonsense word 'kekakke'.*

In order to avoid undesirable weighting of the measurements due to differences in dynamic range, all measurements are normalized to fall within the interval $[0, 1]$.

We assume a sampled speech waveform $x(k)$, for $k = 0, \cdots, N-1$ that is pre-emphasized by the filter $1 - 0.95z^{-1}$, blocked and Hamming-windowed into frames of 20 ms, with a frame shift of 10 ms.

Normalized short-time energy

The short-time energy E_N can be estimated by

$$E_N = \sum_{k=0}^{N-1} x(k)^2 \tag{4.1}$$

The primary motivation for incorporating this measurement is to distinguish speech from silence. E_N is normalized by the maximum short-time energy $E_{\max}$ found in the utterance. Furthermore, E_N is scaled by a factor e (experimentally fixed at 500.0) in order to circumvent the problem in discriminating weak fricatives or plosive bursts from silence. Furthermore, it is flipped in magnitude to express the presence of silence by a value near 1, i.e.

$$E'_N = 1 - e\frac{E_N}{E_{\max}} \tag{4.2}$$

Finally, negative values of E'_N are clipped to zero.

Normalized low and high frequency energies

A straightforward method of performing broad-band filtering is to combine the output of Discrete Fourier Transform (DFT) channels that lie within the required frequency bands. First, an N-point FFT of the speech sequence $x(k)$ is taken and the squared modulus is calculated of each FFT channel (spectral power domain filtering). By considering cut-off frequencies of 50 Hz and 1200 Hz for the low frequency energy E_{low} and 2000

Hz and 4000 Hz for the high frequency energy E_{high}, the channels that fall within one of these frequency bands are added. Subsequently, both energy contours are normalized by the total energy that is found in the two frequency bands.

Zero crossing rate

A zero crossing occurs if successive samples have opposite algebraic signs. An appropriate definition for the zero crossing rate Z_N is,

$$Z_N = \frac{1}{2N} \sum_{k=1}^{N-1} |sgn[x(k)] - sgn[x(k-1)]| \tag{4.3}$$

in which $sgn[.]$ is the sign function.

Obviously, all that is required is to check samples in pairs for sign changes and compute the average over N samples present in the frame.

Since high frequencies imply high zero crossing rates, and low frequencies imply low zero crossing rates, there is a strong correlation between zero-crossing rate and spectral energy distribution. One can roughly state that a high zero crossing rate is due to unvoiced speech, while if the zero crossing rate is low, voiced speech is at stake.

First LPC coefficient

The first LPC coefficient of a first order LPC model is defined as the ratio of the first and zero-th autocorrelation lag, i.e.

$$a_1 = \frac{r(1)}{r(0)} \tag{4.4}$$

where $r(0)$ and $r(1)$ can be calculated by their unbiased estimations from the speech samples. An a_1 coefficient lying near 1 indicates a major energy concentration in the low frequencies and a value near -1 indicates a major concentration in the high frequencies.

We have linearly scaled the magnitude of the a_1 coefficient into an interval between 0 and 1, by defining

$$a_1' = (1 + a_1)/2 \tag{4.5}$$

Let us now describe in more detail the specific task of segmenting an utterance into a predetermined sequence of broad phonetic classes. We assume that each phoneme falls in only one of the three broad phonetic classes. This is accomplished by defining a function that uniquely maps each phonetic label onto a broad phonetic class. Thus, denoting by *Lab* the set of phonetic label names, and by *Class* the set of broad phonetic class names, we introduce the auxiliary function $M : Lab \longrightarrow Class$. By that means, we acquire a unique transcription into L' broad phonetic classes (SIL,UNV, and VOI) from a transcription in L phonetic labels where $L' \leq L$.

Now, the problem is to find L' consecutive broad phonetic class segments in an observation sequence $\mathbf{O}_1^T = \mathbf{o}(1), \cdots, \mathbf{o}(T)$ as formally defined in Section 3.2.

We can represent each broad phonetic class by a prototypical vector or centroid. So, let $\mathcal{C} = \{\hat{\mathbf{c}}_{\mathtt{SIL}}, \hat{\mathbf{c}}_{\mathtt{UNV}}, \hat{\mathbf{c}}_{\mathtt{VOI}}\}$ be a set of three (initially undetermined) centroids of these classes. We denote $d(\mathbf{o}(t), \hat{\mathbf{c}})$ as a distance measure between the t-th observation vector and a centroid

$\hat{\mathbf{c}} \in \mathcal{C}$. The l-th intra-segment distance $d_l(i,j)$ is simply the summed distances between the vectors spanning the segment and the centroid $\hat{\mathbf{c}} \in \mathcal{C}$,

$$d_l(i,j) = \sum_{t=i}^{j} d(\mathbf{o}(t), \hat{\mathbf{c}}_k) \quad \text{,with } k = F(l) \tag{4.6}$$

where $F : \mathbb{N} \longrightarrow \mathit{Class}$ represents an auxiliary function that directly maps an integer l (denoting a segment number) onto its broad class name for the utterance under consideration.

As the centroids are initially unknown, we both have to find a segmentation and a set of centroids such that the total distortion is minimal, i.e.

$$\sum_{l=1}^{L'} d_l(b_{l-1}+1, b_l) = \sum_{l=1}^{L'} \sum_{t=b_{l-1}+1}^{b_l} d(\mathbf{o}(t), \hat{\mathbf{c}}_k) \quad \text{,with } k = F(l) \tag{4.7}$$

The Euclidean distance between the observation vector $\mathbf{o}(t)$ and the centroid $\hat{\mathbf{c}}_k$ is used, thus

$$d^2(\mathbf{o}(t), \hat{\mathbf{c}}_k) = (\mathbf{o}(t) - \hat{\mathbf{c}}_k)'(\mathbf{o}(t) - \hat{\mathbf{c}}_k) \tag{4.8}$$

where $'$ denotes transpose. The centroid $\hat{\mathbf{c}}_k$ belonging to a broad phonetic class $k \in \mathit{Class}$ is simply the arithmetic mean of all observation vectors assigned to that class by a preceding iteration of the segmentation algorithm,

$$\hat{\mathbf{c}}_k = \frac{1}{T_k} \sum_{t=1}^{T_k} \mathbf{o}(t) \tag{4.9}$$

in which T_k is the number of vectors assigned to class k.

It is worth noting that the implementation of the minimization is accomplished in a dynamic programming framework that is quite similar to Viterbi training used in speech recognition [Rabiner 93] and close to the level-building approach described in Section 4.3.1. As can be seen in Equation 4.13, minimal and maximal durations allowed for a segment can be imposed to the level-building algorithm. We have capitalized on that feature by putting the duration parameters for each broad phonetic segment equal to the sum of the duration parameters of its constituting phoneme-like units. The required duration statistics for each phoneme-like unit are acquired from the speech inventory at hand. We used the minimal and maximal duration found in the inventory and they are shown in the columns with headings Min Dur and Max Dur in Table 3.1.

The broad phonetic class segmentation consists in an iterative procedure where each iteration has two steps. The first step seeks an optimal set of boundaries given a set of centroids. The second step updates the centroid set by using the new acquired boundaries. Both steps come up with a minimal total distance that is exploited to guarantee convergence.

Thus, each iteration i involves the updating of the boundaries using centroid set $\mathcal{C}_i$ resulting into boundary sequence $\mathcal{B}_i$ and a total distance D_i. Subsequently, the centroid set is updated using boundary sequence $\mathcal{B}_i$ resulting into $\mathcal{C}_{i+1}$ and total distance D'_{i+1}. This procedure is repeated until some convergence criterion involving D_i and D'_{i+1} is met.

The first iteration starts with an initial set of centroids $\mathcal{C}_0$ that represent some ideal broad phonetic classes. This is a reasonable thing to do, recalling the imprecise spectral energy distribution characterization of each measurement with respect to a broad phonetic class.

In more detail we describe hereunder the proposed algorithm

Outline of Algorithm

One is given an observation sequence of length T,

$$\mathbf{O}_{1,T} = \mathbf{o}(1), \cdots, \mathbf{o}(T)$$

where

$$\mathbf{o}(t) = (E'_N, E_{low}, E_{high}, Z_N, a'_1)'$$

the set of broad phonetic labels *Class* = $\{\texttt{SIL}, \texttt{UNV}, \texttt{VOI}\}$, a transcription function $F : \mathbb{N} \longrightarrow$ *Class*, and a convergence threshold ε.

Initialization:

Set $i = 0$.

Start with a initial centroid set $\mathcal{C}_0 = \{\hat{\mathbf{c}}_{\texttt{SIL}}, \hat{\mathbf{c}}_{\texttt{UNV}}, \hat{\mathbf{c}}_{\texttt{VOI}}\}$ where each centroid is a binary vector:

$$\hat{\mathbf{c}}_{\texttt{SIL}} = (1,0,0,1,1)' \quad \hat{\mathbf{c}}_{\texttt{UNV}} = (0,0,1,1,0)' \quad \hat{\mathbf{c}}_{\texttt{VOI}} = (0,1,0,0,1)'$$

(Justification for this initial centroid set: An ideal prototypical centroid for the class **SIL** is characterized by a low normalized short-time energy implying that E'_N lies near 1, low normalized energies in both frequency bands implying that both E_{low} and E_{high} lie near 0, a high zero rate crossing implying that Z_N lies near 1, and a high first LPC coefficient implying that a'_1 lies near 1. The same sort of arguments hold for the classes **UNV** and **VOI**.)

Step 1. Boundary updating:

Update the boundary sequence $\mathcal{B}_i = \{b_0, \cdots, b_{L'}\}$ by minimizing the total Euclidean distance,

$$D_i = \min_{\mathcal{B}_i} \sum_{l=1}^{L'} \sum_{t=b_{l-1}+1}^{b_l} (\mathbf{o}(t) - \hat{\mathbf{c}}_k)'(\mathbf{o}(t) - \hat{\mathbf{c}}_k) \quad \text{, with } k = F(l)$$

This time-alignment of the observation sequence against the centroids is achieved by an implementation of the level-building dynamic programming algorithm (see Section 4.3.1). Duration constraints of the segments are imposed to the algorithm.

Step 2. Centroids updating:

Update the centroid set $\mathcal{C}_{i+1}$ by first collecting all *partial* observation sequences that correspond to the same broad phonetic class label into three distinct sequences $\mathbf{O}^{(k)}_{1,T_k}$, where T_k denotes the number of vectors for each label $k \in$ *Class*, i.e.

$$\mathbf{O}^{(k)}_{1,T_k} = \{\mathbf{o}(t) \mid b_{l-1} + 1 \leq t \leq b_l \ \wedge \ k = F(l) \ \wedge \ 1 \leq l \leq L'\}$$

The optimal centroid $\hat{\mathbf{c}}_k \in \mathcal{C}_{i+1}$ for the Euclidian distance is the arithmetic mean of each sequence $\mathbf{O}^{(k)}_{1,T_k}$,

$$\hat{\mathbf{c}}_k = \frac{1}{T_k} \sum_{t=1}^{T_k} \mathbf{o}(t) \quad , \mathbf{o}(t) \in \mathbf{O}^{(k)}_{1,T_k}$$

Compute the distortion decrease

$$\delta = (D_i - D'_{i+1})/D'_{i+1}$$

where

$$D'_{i+1} = \sum_{l=1}^{L'} \sum_{t=b_{l-1}+1}^{b_l} d(\mathbf{o}(t), \hat{\mathbf{c}}_k) \quad \text{, with } k = F(l)$$

represents the new distortion one obtains by using the updated centroid set $\mathcal{C}_{i+1}$.

Step 3. Termination:

If the convergence criterion $\delta < \varepsilon$ is met, terminate with boundary sequence $\mathcal{B}_i$ and minimum total distance D_i, else set $i = i + 1$ and go to *Step 1.*

We have to guarantee that this iterative procedure converges to a local optimal solution by showing that the sequence of intermediate total distances is monotonically decreasing, namely $D_0 \geq D_1 \geq \cdots \geq D_i \geq D_{i+1} \geq \cdots$. This can be proved inductively along the following lines (see [Rabiner 93] for a completely similar argument concerning the convergence of Viterbi training):

1. As described in Step 2, summing up all intra-segment distances by using the updated centroid set $\mathcal{C}_{i+1}$ yields a new distance D'_{i+1} that is smaller than or equal to D_i obtained by the centroid set $\mathcal{C}_i$, i.e. $D_i \geq D'_{i+1}$. This can be explained by the fact that each centroid $\hat{\mathbf{c}} \in \mathcal{C}_{i+1}$ is chosen as the one that minimizes all intra-segment distances to that centroid. This is also true for the first iteration in which the first updated centroid set $\mathcal{C}_1$ is only a improved version of the initial centroid set $\mathcal{C}_0$ with respect to the segmentation $\mathcal{B}_0$.

2. As descibed in Step 1, calculating a set of boundaries $\mathcal{B}_i$ by using the centroid set $\mathcal{C}_i$ yields a minimal total distance D_i that is smaller than or equal to D'_i because $\mathcal{B}_i$ is obtained by means of a distance minimization scheme, i.e. $D'_i \geq D_i$.

Combining both results proves that the inequality $D_i \geq D_{i+1}$ holds for all i.

4.2.2 Results

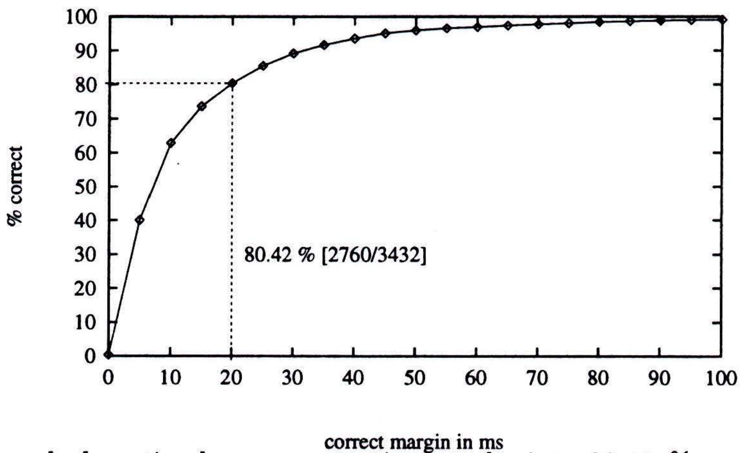

Figure 4.3: *The broad phonetic class segmentation results into 80.42 % correct within 20 ms [2760/3432]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurrences).*

The BPC segmentation is evaluated on the speech inventory by mapping each manual segmentation to its broad phonetic class representant. It resulted into 3432 manually positioned boundaries between broad phonetic class segments. The mapping itself is shown in Table 4.1. Obviously, the first and last boundary representing respectively the beginning and ending of

phoneme-like unit		broad phonetic class
P1 T1 K1 SI	⟶	SIL
P2 T2 K2 F S X SJ	⟶	UNV
I E A O CC C Y U AA EE OE OO II EH UH OH		
EI1 EI2 UI1 UI2 AU1 AU2	⟶	VOI
D1 D2 B1 B2 G1 G2 V Z H ZJ M N Q L R W J		

Table 4.1: *Mapping from phoneme-like unit to broad phonetic class.*

the sampled speech data file are not taken into account in the segmentation results (see also Section 3.2 and 3.4).

Each sampled speech waveform from the inventory was pre-emphasized by the filter $1 - 0.95z^{-1}$, blocked and Hamming-windowed into frames of 20 ms, with a frame shift of 2.5 ms. A frame shift of 2.5 ms resulted into higher segmentation accuracies than settings in which the frame shift was 10 ms. The phonetic transcription of the utterance, some global duration statistics for each phoneme-like unit as shown in the columns with heading Min Dur and Max Dur in Table 3.1, and the mapping to the broad phonetic classes were also provided to the algorithm.

	UNV	VOI	SIL		
UNV	0 / 0	743 / 914(81.29%)	55 / 98 (56.12%)	798 /1012	78.85 %
VOI	299 / 370 (80.81%)	0 / 0	774 /1066 (72.61%)	1073 /1436	74.72 %
SIL	562 / 618 (90.94%)	327 / 366(89.34%)	0 / 0	889 /984	90.35 %
	861 /988	1070 /1280	829 /1164	2760 /3432	
	87.15 %	83.59 %	71.22 %		80.42 %

Table 4.2: *Detailed overview of the boundary disagreements for each broad phonetic class by considering a correct margin of 20 ms. [p/q] means p correct out of q occurrences.*

As shown in Figure 4.3, the segmentation method demonstrates that the difference between the manual segmentation and the automatic segmentation is inferior to *20* ms for *80.42 %* of the broad phonetic class boundaries. A more detailed overview of the boundary discrepancies by considering a correct margin of *20 ms* is shown by Table 4.2 in which vertically the broad phonetic class to the left of the boundary is indicated, and horizontally the class to the right of the boundary. The score p/q in each entry of the matrix means p correct out of q occurrences. Also, the corresponding percentage is presented. In the bottom row and the right most columns, summations over respectively the column and the row is given. In the bottom-right entry, the total segmentation performance is given.

It is apparent in Table 4.2 that the VOI-SIL boundaries (with only 72.61 % correct and large number of occurrences) are a first candidate for improvement. These boundaries occur at word final positions, mostly at the ending of the neutral schwa (recall the phonetic structure of the nonsense words). Also it represents the closure onset of unvoiced plosives which are of special interest (T1, P1, and K1 segments). It is difficult to determine exactly the onset of these SIL regions by relying solely on our set of signal measurements.

The boundaries between voiced and unvoiced regions still need improvement. Our method does not come up with a clear discrimination between these regions in all cases. Many misalignments are observed in burst-vowel transitions in which the bursts are not well articulated.

On the other hand, the SIL-VOI score is rather high, delivering clear closure-burst event detections within unvoiced plosives (e.g. P1 P2 clusters). However, permanent attention has to be paid to distinguishing silence from weak unvoiced fricatives and plosives.

4.2.3 Limiting Cases

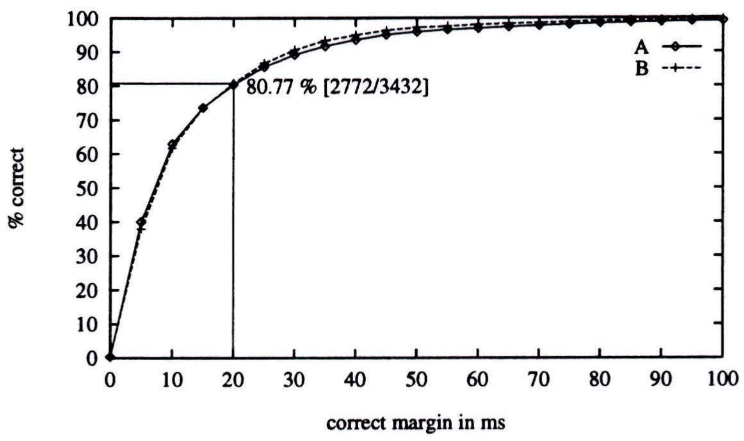

Figure 4.4: *Limiting case of broad phonetic class segmentation.* **A** *The iterative broad phonetic class segmentation method results into 80.42 % correct within 20 ms [2760/3432].* **B** *The broad phonetic class segmentation with externally provided centroids results into 80.77 % correct within 20 ms [2772/3432]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurrences).*

The final performance of our proposed method of segmenting broad phonetic classes must be investigated in a proper perspective. Therefore, we conducted the same experiment by providing the algorithm a set of external centroids. Instead of computing interatively a new centroid set, the set is known and determined by averaging the collection of signal measurements that fall within each manually positioned broad phonetic class segment in the speech inventory. This reduces our method into a template matching scheme. In this respect, template matching technique must deliver better results and thus is considered an upper bound of performance that can be achieved by our method. By considering a margin of *20 ms*, *80.77 %* of the automatic positioned broad phonetic class boundaries are correct. In Figure 4.4 the global segmentation performance of this template matching method is shown compared with our iterative method. No substantial performance difference between the two methods exists. As a matter of fact, the same observations as mentioned in Section 4.2.2 are made. Thus, the addition of more *a priori* knowledge in terms of externally provided centroids does not remove the lack of discriminative power of the signal measurements.

4.2.4 Discussion

Some aspects of our method have to be kept in mind. Some of them may be considered weaknesses and must be remedied in future research projects:

1. The existence of an acoustic-phonetic property at a particular position in the utterance is assumed *a priori*. A phonetic transcription of the utterance never assures that a particular phoneme is acoustically realized according to an assumed broad phonetic class. Especially, assimilation processes as they can occur in clusters of phonemes with voiced and unvoiced characteristics, such as /*sz*/ and /*fv*/ clusters, are not detected. No mechanism that takes account of these kind of phonological processes was utilized here.

2. The choice of signal measurements made here is probably not optimal and they are combined in a rather *ad hoc* manner. Other measurements and combinations may turn out to be better.

3. The particular choice for starting from an initial centroid set that reflects an ideal broad phonetic class representant and iteratively updating the centroids according to the information at hand is made on rather intuitive grounds. Another possibility was to start with nothing and compute the centroids on-the-fly during a single dynamic programming pass, such as in the method that will be explained in Section 4.3.1. However, we felt that this is inappropriate because we can make good guesses for an initial set recalling their discriminative power regarding the three classes. Besides, we want to represent all corresponding broad phonetic class segments by one and the same centroid because these classes are heavily linked with each other regarding their acoustic-phonetic properties. This so-called centroid-pooling proved to us to be very difficult to solve in a single dynamic programming pass. The need for centroids can also be solved by providing them externally such as the template matching method mentioned in Section 4.2.3, but this does not fit in the philosophy of our approach.

4. The normalization procedure of the measurements (by scaling them to an interval between 0 and 1) in order to compensate for dynamic range differences may be rather crude. A more sophisticated way is to normalize the measurements by their (estimated) variances. This results into the application of a weighted Euclidian distance (i.e. Mahalanobis distance). This distance metric has the advantage that it assigns a lower weight to measurements that show a high standard deviation, i.e. they are considered less reliable.

4.3 Sequence Constrained Vector Quantization

The segmentation into broad phonetic classes delivers anchor points for a more detailed processing. In principle, the segmentation at phoneme level of the complete utterance can now be broken down in the segmentation of each broad phonetic class segment into its constituting phoneme-like units. The benefit is that segmentation inaccuracies in one broad phonetic class segment do not propagate to the next one, thus improving accuracy as reflected in the results in Section 4.3.2. In practice, we shall take into account the fact that the anchor points are not totally exact and we shall allow small adjustments to this anchor points when this improves the accuracy of the Sequence Constrained Vector Quantization.

4.3.1 Outline of method

The problem being addressed now is the determination of L consecutive phoneme segments in an observation sequence $\mathbf{O}_1^T = \mathbf{o}(1), \cdots, \mathbf{o}(T)$ as formally defined in Section 3.2.

We shall follow a method originally proposed by Svendsen and Soong [Svendsen 87, Vidal 90] which aims at segmenting the utterance into consecutive quasi-stationary elements. Obviously, diphtongs and plosives have to be further divided in their constituents as explained in Section 3.3.

In order to quantify the concept of quasi-stationarity we need a distortion measure between a pair of observation vectors. This measure gives the possibility to cluster vectors or to associate an observation with a segment. By that means, each segment l can be represented by a generalized centroid that is based on the observation vectors spanning that segment. One can recognize the resemblance with ordinary Vector Quantization codebook design [Gray 84] in which T vectors are partitioned into L clusters and L codebook vectors are chosen as the centroid of each of the L clusters. In our case, the technique is further subjected to the constraint that all vectors in a cluster (segment) are contiguous in time and is therefore called Sequence Constrained Vector Quantization (SCVQ) in the sequel.

As already noted in Section 4.2.1, the method to identify the positions of the broad phonetic classes and this SCVQ approach are quite similar. Nevertheless, we have treated them separately because they show some specific differences.

An 'optimal' segmentation and a set of L centroids can be found by minimizing a total distortion which we now define. We denote the distortion measure between the t-th observation vector $\mathbf{o}(t)$ and a centroid $\hat{\mathbf{c}}_l$ of the l-th segment as $d(\mathbf{o}(t), \hat{\mathbf{c}}_l)$. The l-th *intra-segment* distortion $d_l(i,j)$ is simply the sum of distortion terms between the vectors $\mathbf{O}_i^j = \mathbf{o}(i), \cdots, \mathbf{o}(j)$ spanning the segment l and the centroid $\hat{\mathbf{c}}_l$,

$$d_l(i,j) = \sum_{t=i}^{j} d(\mathbf{o}(t), \hat{\mathbf{c}}_l) \tag{4.10}$$

Now the problem can be formulated as finding the boundaries $\{b_0, b_1, \cdots, b_L\}$ by minimizing the total distortion measure,

$$\sum_{l=1}^{L} d_l(b_{l-1}+1, b_l) = \sum_{l=1}^{L} \sum_{t=b_{l-1}+1}^{b_l} d(\mathbf{o}(t), \hat{\mathbf{c}}_l) \tag{4.11}$$

Still, attention has to be paid to the determination of the centroid of a segment. One possibility is to define the centroid as the vector that minimizes the distortion in a particular fixed segment spanned by the observation sequence $\mathbf{O}_{b_{l-1}+1}^{b_l} = \mathbf{o}(b_{l-1}+1), \ldots, \mathbf{o}(b_l)$, i.e.

$$\hat{\mathbf{c}}_l \triangleq \underset{\mathbf{c}_l}{\arg\min} \frac{1}{b_l - b_{l-1}} \sum_{t=b_{l-1}+1}^{b_l} d(\mathbf{o}(t), \mathbf{c}_l) \tag{4.12}$$

This has the obvious advantage that no additional knowledge is required and the centroids are acquired from the observation at hand. It must be emphasized that this centroid computation is on-the-fly and must re-computed each time the segment is stretched (or shrinked) during the minimization of Equation 4.11. So, many candidate centroids for each segment are to be considered.

Another possibility could be to assume that the centroids are provided externally by single reference frame templates or prototypes via a hand-labelled speech database. This however would require *a priori* knowledge and therefore does not fit into our approach.

The actual centroid computation procedure as defined by Equation 4.12 is highly dependent on the choice of the distortion measure. This discussion will be postponed until the description of the particular distortion measure that is incorporated in the SCVQ framework is given.

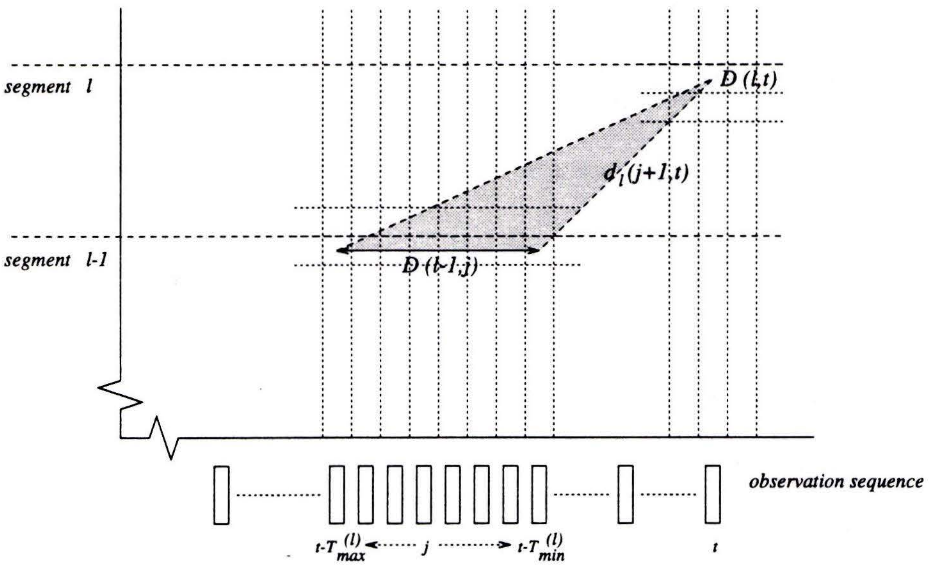

Figure 4.5: *An operational interpretation of level-building according to Equation 4.13.*

The SCVQ approach can be efficiently implemented by the application of level-building dynamic programming [Rabiner 93]. As with any technique based on dynamic programming, this one tries to find a time-aligned path between the observation sequence and a set of centroids. This path indicates the best match between groups of consecutive vectors (segments) and each centroid.

In addition to this dynamic programming framework, the level-building approach performs all computations at each segment[1] on all candidate centroids before it proceeds to the next segment. By that means, we can compare partial accumulated distortions at each segment

[1]Level-building denotes segments as 'levels' because in an operational graphical representation of the process the segments appear as horizontal lines.

and retain only the minimum at each path ending frame. The computation of the next segment simply picks up this minimum as well as its path ending frame and starts from there. It exactly meets our needs of finding consecutive segments in such a way that the sum of all intra-segment distortions is kept to a minimum.

In this respect, we define the minimum accumulated distortion $D(l,t)$ over a *partial* observation $\mathbf{O}_1^t = \mathbf{o}(1), \cdots, \mathbf{o}(t)$ that is decomposed into l segments (levels). This minimal distortion corresponds to a time-alignment path that intersects with a range of candidate ending frames that marks the transition from segment $l-1$ to l. This range of candidate ending frames represents the path expansion constraints and are specified by the interval $[t - T_{\max}^{(l)}, t - T_{\min}^{(l)}]$. As explicitly shown by the superscripts, each path constraint can be handled separately for each particular segment under consideration. In other words, $T_{\min}^{(l)}$ and $T_{\max}^{(l)}$ may respectively represent the global minimum and maximum duration allowed for a phoneme-like unit derived from statistics of (manually segmented) speech data as given in Table 3.1.

As is shown in Figure 4.5, the accumulated distortion $D(l,t)$ is recursively defined as the minimum distortion $D(l-1,j)$ of the *partial* observation $\mathbf{O}_1^j = \mathbf{o}(1), \cdots, \mathbf{o}(j)$ divided into $l-1$ segments plus the intra-segment distortion $d_l(j+1,t)$ from Equation 4.10. That is,

$$D(l,t) = \min_{t-T_{\max}^{(l)} \leq j \leq t-T_{\min}^{(l)}} [D(l-1,j) + d_l(j+1,t)] \tag{4.13}$$

By satisfying the boundary condition $D(0,0) = 0$, one can interpret $D(L,T)$ as the minimal total distortion in which the *complete* observation sequence $\mathbf{O}_1^T = \mathbf{o}(1), \cdots \mathbf{o}(T)$ is decomposed into L consecutive segments. By retaining all candidate ending frames corresponding to minimal partial time-alignment paths, one can easily find the boundaries $\mathcal{B} = \{b_0, b_1, \cdots, b_L\}$ by backtracking the optimal path ending at frame T.

It is perhaps worthwhile to emphasize two major differences between SCVQ and the segmentation in broad phonetic classes described in Section 4.2.1. There, all segments carrying the same broad phonetic class label are represented by the same centroid. This is not the case with SCVQ where segments with the same phonetic label will generally have different centroids according to Equation 4.12. In addition, the determination of centroids and segment boundaries are integrated in a single minimization step of the total distortion defined by Equation 4.11, whereas in Section 4.2.1 computation of boundaries and centroids update is performed in distinct successive steps.

So far, nothing is said about the integration of the broad phonetic class segmentation as provided by BPC into this SCVQ stage. Obviously, each broad phonetic class segment spans acoustic realizations of some phoneme-like units. By seeking locally in a specific broad phonetic class segment for boundaries of its constituting phoneme-like units, one can expect that segmentation inaccuracies at one place do not propagate to other positions. However in practice, the segmentation into broad phonetic classes is not totally exact. Together with the fact that a simple linear normalization is needed to scale the duration constraints for each phoneme-like unit, a small compliance factor has to be maintained for these anchor points to improve accuracy. This is done as follows.

Assume a broad phonetic class segments can be broken down into L constituting phoneme-like units. We are given the actual duration T_s of the broad phonetic class segment. Also, we know some estimated average duration $\hat{\mu}^{(l)}$ of each phoneme-like unit. In particular, we have used the average duration statistics as shown in Table 3.1 in the column with heading

Ave Dur $\pm$ Std Dev. An approximation for the minimal and maximal allowed duration of a phoneme-like unit as denoted by $T_{\min}^{(l)}$ and $T_{\max}^{(l)}$ in Equation 4.13 can be made by a straightforward normalization procedure. An appropriate minimum and maximum allowed duration for phoneme label l is given by

$$T_{\min}^{(l)} = \hat{\mu}^{(l)} \frac{T_s}{\sum_{l=1}^{L} \hat{\mu}^{(l)}} - \gamma \tag{4.14}$$

$$T_{\max}^{(l)} = \hat{\mu}^{(l)} \frac{T_s}{\sum_{l=1}^{L} \hat{\mu}^{(l)}} + \gamma \tag{4.15}$$

where γ is a compliance factor. The highest accuracy was obtained by fixing γ at 20 ms.

A subtle, but important addition in terms of accuracy performance is to follow a slightly different procedure for burst-like units, such as P2 and B2. In preliminary experiment, it was observed that just these bursts highly contributed to the overall segmentation inaccuracy due to a too loose specification of the allowed durations. Thus, no compliance factor to the maximum allowed duration $T_{\max}^{(l)}$ for the bursts is added.

Itakura Distortion Measure

In automatic speech-recognition applications, studies to date have not indicated conclusively that a particular distortion measure leads to a higher recognition accuracy. The choice for a specific local distortion measure is often a function of the specific application, speech parametrization, and environment. Our application deals with speech recorded from a single speaker that is noise-free, in which co-articulation effects are kept to a minimum. In this respect, we have used a distortion measure that is very suitable for speech recognition in such an environment, i.e. the Itakura distortion. The rationale behind this distortion is that the set of all possible speech segments derived from the same speech sound are similar in that the underlying true Linear Predictive Coding (LPC) coefficient vectors $\hat{\mathbf{a}}$ are identical. The difference with the calculated LPC coefficients of a speech segment are primarily due to the inaccuracy of the linear prediction speech model. In fact, the Itakura distance is a proposed simplification of the Gaussian log-probability that the calculated coefficients vector $\mathbf{a}$ are from a speech segment with true coefficient vector $\hat{\mathbf{a}}$ [Itakura 75]. Whenever we consider the centroid $\hat{\mathbf{a}}_l$ as the true LPC model of segment l, we can state this measure as our *intra-segment* distortion (see also Equation 4.10),

$$d_l(i,j) = \sum_{t=i}^{j} \log\left(\frac{\hat{\mathbf{a}}_l' \mathbf{R}_t \hat{\mathbf{a}}_l}{\sigma_t^2}\right) \tag{4.16}$$

where $\hat{\mathbf{a}}_l = (1, a_1, \cdots, a_p)'$ is the p-th order LPC centroid vector of segment l ($p = 12$), σ_t^2 is the residual energy in frame t, and $\mathbf{R}_t$ is the augmented $(p+1) \times (p+1)$ autocorrelation matrix of frame t

$$\mathbf{R}_t = \begin{bmatrix} \hat{r}(0) & \hat{r}(1) & \cdots & \hat{r}(p) \\ \hat{r}(1) & \hat{r}(2) & \cdots & \hat{r}(p-1) \\ \vdots & \vdots & & \vdots \\ \hat{r}(p) & \hat{r}(p-1) & \cdots & \hat{r}(0) \end{bmatrix} \tag{4.17}$$

The Itakura distortion is a logarithm of the ratio of prediction residuals resulting from inverse filtering the waveform of frame t by the LPC centroid model $\hat{\mathbf{a}}_l$ and by the LPC model of that frame. If the frame model is close to the process generated by the LPC centroid model $\hat{\mathbf{a}}_l$ the distortion will be close to zero; otherwise, it will be significantly large.

An alternative efficient computation of the Itakura distortion that requires a factor of $(p+2)$ less operations than the direct way is by evaluating the residual energy term by [Rabiner 93],

$$\hat{\mathbf{a}}_l'\mathbf{R}_t\hat{\mathbf{a}}_l = r_a(0)\hat{r}(0) + 2\sum_{n=1}^{p} r_a(n)\hat{r}(n) \tag{4.18}$$

where

$$r_a(n) \triangleq \sum_{i=0}^{p-n} a_i a_{i+n} \qquad ,\text{for } n = 0, 1, \cdots, p \tag{4.19}$$

which is the autocorrelation of the predictor coefficients.

As defined in Equation 4.12, the centroid is that vector that minimizes the intra-segment distortion with respect to the observation sequence in that segment. By substituting the Itakura distortion in Equation 4.12, we have to solve

$$\hat{\mathbf{a}}_l \triangleq \underset{\mathbf{a}_l}{\arg\min} \; \frac{1}{b_l - b_{l-1}} \sum_{t=b_{l-1}+1}^{b_l} \log\left(\frac{\mathbf{a}_l'\mathbf{R}_t\mathbf{a}_l}{\sigma_t^2}\right) \tag{4.20}$$

However, this can not be solved in a straightforward way due to the involved minimization procedure of a geometric mean. A reasonable approximation is to ignore the logarithm in Equation 4.20 which reduces the problem into a minimization of an arithmetic mean that can be considered an upperbound of the geometric mean [Rabiner 93]. Now, the centroid becomes the LPC solution of the set of normal equations defined by the average residual-normalized autocorrelations

$$\frac{1}{b_l - b_{l-1}} \sum_{t=b_{l-1}+1}^{b_l} \frac{\mathbf{R}_t}{\sigma_t^2} \tag{4.21}$$

4.3.2 Results

We evaluated the segmentation results by comparing them with their corresponding manually positioned boundaries. A total of 6412 boundaries are considered without the boundaries that specify the beginning and ending of the sampled speech data file (see also Sections 3.2 and 3.4).

Each sampled speech waveform was pre-emphasized by the filter $1 - 0.95z^{-1}$, blocked and Hamming-windowed into non-overlapping frames of 10 ms, with a frame shift of 10 ms. We used an order of 12 ($p = 12$) for the LPC analysis. Non-overlapping frames guarantee the correct calculation of the centroid minimizing the Itakura distortion within a segment. This calculation requires the estimation of an average (residual-normalized) autocorrelation sequence as shown in Equation 4.21, thus overlapping frames may bias this estimation (or at least hamper an efficient computation). We chose a rather low time-frequency resolution by considering a frame width of only 10 ms because this resulted into higher performances than with frame widths of 20 ms.

With respect to the provision of duration information to the algorithm, we followed the scenario that moves from no duration constraints to specific duration constraints that can be derived from a preceding broad phonetic class segmentation.

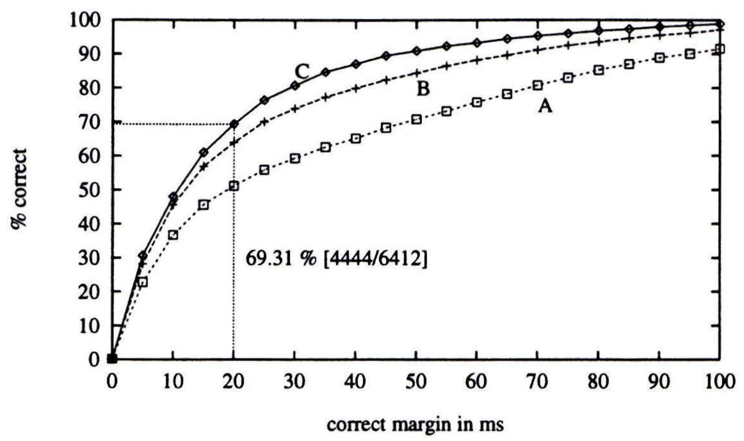

Figure 4.6: *Segmentation results for three configurations of the SCVQ imposed to different duration constraints.* **A** *No duration constraints results into 51.19 % correct within 20 ms [3282/6412]* **B** *Global duration statistics as given in Table 3.1 results into 63.91 % correct within 20 ms [4098/6412].* **C** *Duration constraints as provided by a preceding broad phonetic class segmentation results into 69.31 % correct within 20 ms [4444/6412]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurrences)*

A Segmenting the inventory with no duration statistics, i.e. $T_{\text{min}}^{(l)} = 1$ and $T_{\text{max}}^{(l)} = T + 1$ for all l, where T is the number of speech frames in the observation sequence at hand.

B Segmenting the inventory with global duration statistics for each phoneme-like unit l, i.e. $T_{\text{min}}^{(l)} = \hat{\mu}^{(l)} - \hat{s}^{(l)}$ and $T_{\text{max}}^{(l)} = \hat{\mu}^{(l)} + \hat{s}^{(l)}$ for all l, where $\hat{\mu}^{(l)}$ and $\hat{s}^{(l)}$ are respectively the number of frames corresponding to the average duration and its standard deviation as shown in Table 3.1 in the column with heading Ave Dur ± Std Dev.

C Segmenting the inventory by using the anchor points as provided by the broad phonetic class segmentation. This provides more reliable duration constraints specified for the utterance at hand as described in Section 4.3.1.

As shown in Figure 4.6, the configuration **C** with the broad phonetic class segmentation as preamble resulted into the highest accuracy (*69.31 %* correct within a margin of *20 ms* [4444/6412]). Configuration **B** resulted into *63.91 %* correct within a margin of *20 ms* [4098/6412], and configuration **A** into *51.19 %* correct within a margin of *20 ms* [3282/6412]. It is clear that a substantial improvement is accomplished by providing duration information for each phoneme-like unit that is made as specific as possible for the utterance at hand.

We have clustered all labels into 6 categories, as shown in Table 4.3, in order to look more closely at particular transitions. The two segments of the diphtongs and unvoiced plosives are 'glued' together again to form a single label. This means that we disregard boundaries between these distinct segments in the results and statistics. A detailed overview of the boundary discrepancies by considering a correct margin of *20 ms* is shown by Table 4.4 in which vertically the category to the left of the boundary is indicated, and horizontally the category to the right of the boundary. The score p/q in each entry of the matrix means p correct out of q occurrences. In the bottom row and the most right columns, summations over respectively the column and the row is given. In the bottom-right entry, the total

name	category	phoneme-like units
Vow	vowels	`I E A O CC C II Y U AA EE OE OO EH UH OH AU1 AU2 EI1 EI2 UI1 UI2`
Plo	plosives	`B D G P1 P2 K1 K2 T1 T2`
Fri	fricatives	`Z ZJ V S SJ F X`
Liq	liquids, semi-vowels	`L R H W J`
Nas	nasals	`N M Q`
`SI`	silence	`SI`

Table 4.3: *Categories for evaluation purposes.*

segmentation performance is given. This percentage is slighty different (smaller) compared with the percentage as shown in Figure 4.6 due to the disregarding of boundaries within plosives and diphtongs.

The method is far from perfect and some severe problems can be noticed from Table 4.4 without picking out all of them.

- The vowel-vowel transitions (43.75 % correct [35/80]). Many transitions are realized with a untranscribed glottal stop.
- The beginning of concluding silences (50.89 % correct [372/731]). Because of their relatively large number and their inaccuracy, these transitions highly bias the overall performance, although we are not interested in concluding silences in the first place.
- The transitions involving liquids and semi-vowels (60.88 % correct [372/611] and 61.70 % correct [414/671]).

	Vow	**Plo**	**Fri**	**Liq**	**Nas**	SI		
Vow	35 / 80	336 / 503	330 / 426	228 / 394	186 / 261	335 / 606	1450 /2270	63.88 %
Plos	567 / 740	10 / 15	20 / 27	22 / 42	7 / 8	5 / 42	631 / 874	72.20 %
Fri	498 / 595	41 / 49	14 / 25	20 / 33	9 / 14	4 / 13	586 / 729	80.38 %
Liq	336 / 530	20 / 37	32 / 47	8 / 14	5 / 12	13 / 31	414 / 671	61.70 %
Nas	202 / 277	16 / 33	18 / 23	6 / 14	0 / 4	15 / 39	257 / 390	65.90 %
SI	48 / 54	60 / 79	73 / 101	88 / 114	66 / 75	0 / 0	335 / 423	79.20 %
	1686 /2276	483 / 716	487 / 649	372 / 611	273 / 374	372 / 731	3673 /5357	
	74.08 %	67.46 %	75.04 %	60.88 %	72.99 %	50.89 %		68.56 %

Table 4.4: *Detailed overview of the correct boundary placements for each category by considering a correct margin of 20 ms as achieved by the combined SCVQ and BPC stages within the hierarchical approach. [p/q] means p correct out of q occurrences.*

4.3.3 Contribution to hierarchical approach

Although it is already lightly touched upon in Section 4.3.2 and Figure 4.6, we are interested in the contribution to the hierarchical approach when the Sequence Constrained Vector Quantization is preceeded by a broad phonetic class segmentation. The primary motivation

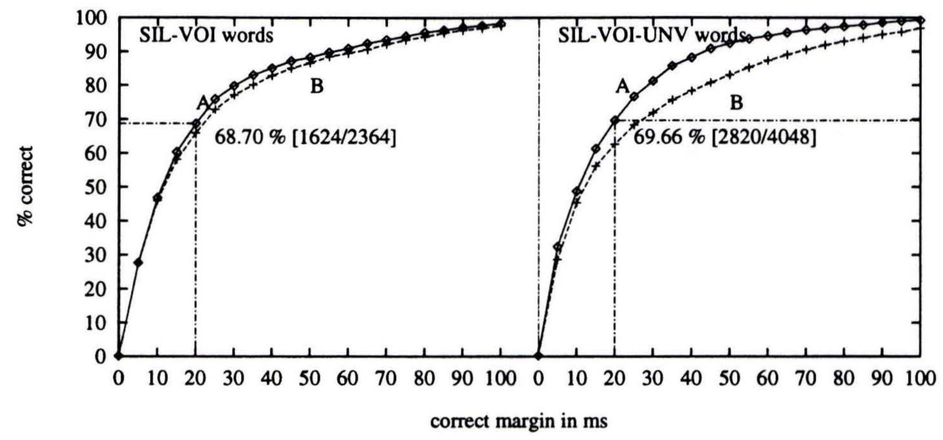

Figure 4.7: *Improvement contribution of the broad phonetic class segmentation with respect to the Sequence Constrained Vector Quantization and considering separately words with* `SIL` *and* `VOI` *regions and words with* `SIL`*,* `VOI`*, and* `UNV` *regions.* **A** *Duration constraints as provided by a preceding broad phonetic class segmentation.* **B** *Global duration statistics per phoneme-like unit as given in Table 3.1. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurrences)*

for using the broad phonetic class segmentation as a preamble for the SCVQ segmentation is the provision of reliable anchor points. In particular, utterances with both voiced and unvoiced regions may profit from this. In order to evaluate this in more detail we have divided the segmentation results into a set containing words with only voiced and silence regions (`SIL-VOI` words) and a set containing words with silence, voiced and unvoiced regions (`SIL-VOI-UNV` words). By evaluating the performance for each set separately, we can measure the contribution of the broad phonetic class segmentation in more detail. In Figure 4.7, a SCVQ application preceeded with a broad phonetic class segmentation is compared with a SCVQ application imposed with global duration statistics for each phoneme-like unit. As might expected, words consisting of a single voiced region possibly enclosed by silence (e.g. `SI J C J OO J C SI`) only profit marginally by a preceding broad phonetic class segmentation. On the other hand, words with alternating voiced and unvoiced characterisitics show a substantial performance improvement due to the provision of anchor points in the utterance.

4.3.4 Limiting cases

In our hierarchical approach, Sequence Constrained Vector Quantization tries jointly to find a set of centroids and a set of boundaries that is optimal with respect to the Itakura distortion of the observation sequence at hand. A limiting case is to relax one of these objectives by providing the algorithm a set of 'exact' centroids. This centroid provision simplifies the algorithm to a template matching scheme in an ideal setting. The centroids are *directly* calculated from the manually positioned segments in the speech inventory by using Equation 4.21. Two ways in calculating the 'exact' centroids from this manual segmentation are investigated.

A. A centroid is calculated for each segment by Equation 4.21. Thus, each realization of a label in the transcription has attached to it an *exact and unique* centroid during the dynamic programming pass.

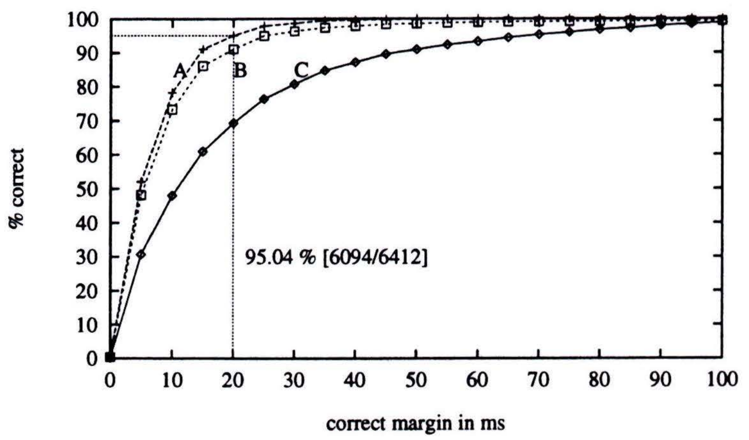

Figure 4.8: *Segmentation result for the limiting case of SCVQ by providing it 'exact' centroids.* **A** *SCVQ provided with 'exact' centroids calculated from the manually position segments results into 95.04 % correct within 20 ms [6094/6412].* **B** *SCVQ provided with centroids calculated from all collected realizations form manually positioned segments results into 91.06 % correct within 20 ms. [5839/6412]* **C** *SCVQ within the hierarchical approach results into 69.31 % correct within 20 ms [4444/6412]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurrences)*

B. A centroid is calculated for each collection of segments corresponding to the same label by Equation 4.21. The realizations for each label are pooled. Thus, each corresponding label in the transcription is represented by the *same* centroid during the dynamic programming pass.

We imposed no durational constraints to the algorithm, i.e. $T_{\text{min}}^{(l)} = 1$ and $T_{\text{max}}^{(l)} = T$ for all l, where T is the number of speech frames in the observation sequence at hand. As shown in Figure 4.8, we found that *95.04 %* of the boundaries [6094/6412] were positioned in an interval of *20 ms* around their manually positioned correspondent for case **A**. By pooling firstly the realizations, and successively calculating centroids from these, the accuracy is still *91.06 %* correct. These pooled centroids do not exactly correspond one-to-one with the segments causing more confusion during the segmentation process. However, both cases show a rather great performance difference with respect to the accuracy of SCVQ within the hierarchical approach. We conclude that the disposal of reliable centroids is crucial for the performance of the algorithm, leaving room for further studies that aim at finding optimal centroids by clustering strategies. A similar conclusion was drawn by Svendsen and Soong [Svendsen 87]. In addition, we were convinced by the fact that looking at local signal properties contribute to high accuracy suggesting that a concluding SCVQ stage at the end of a HMM Viterbi segmentation may give our final result. In other words, this SCVQ stage *directly* calculates a set of centroids by considering the segments (or some central part of them) provided by the HMMs and subsequently carries out a template match alignment. Unfortunately, preliminary experiments were not in favour of this idea; although it mend some problems involving liquids-schwa transitions introduced by the HMMs, it lowered the *20 ms* accuracy achieved by HMMs by $\pm$ *4%*.

A preceding step of a broad phonetic class segmentation provides durational constraints that nicely correspond with the desired segments of the observation sequence at hand. As

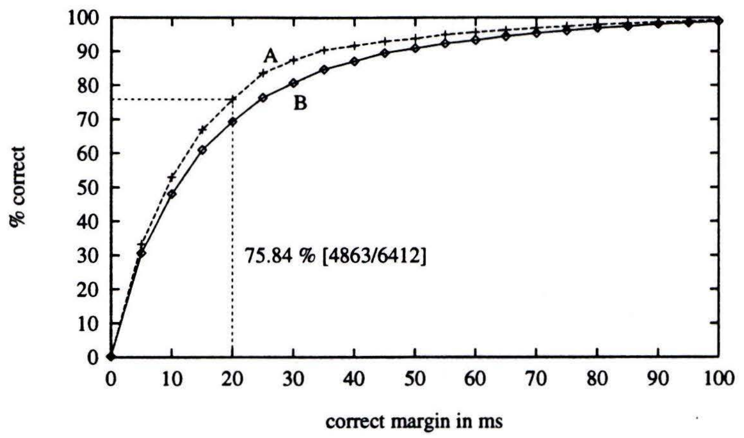

Figure 4.9: *Segmentation result for the limiting case of SCVQ by providing it an 'exact' broad phonetic class segmentation.* **A** *SCVQ provided with the manually positioned 'exact' broad phonetic class segmentation results into 75.84 % correct within 20 ms [4863/6412].* **B** *SCVQ within the hierarchical approach results into 69.31 % correct within 20 ms [4444/6412]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurrences)*

shown in Figure 4.6, this boost the segmentation accuracy of the SCVQ. A limiting case in this respect is to pass on an exact broad phonetic class segmentation for each observation sequence to the SCVQ algorithm. This can be established by means of the manually positioned boundaries in the speech inventory. For each utterance we conducted a SCVQ in this ideal setting, even for the 155 utterances for which a one-to-one correspondence exists between the phonetic transcription and the broad phonetic class transcription (e.g. `SI F C F AA F C SI`). The centroids were calculated *on-the-fly* using Equation 4.21. As shown in Figure 4.9, we found that *75.84 %* of the boundaries [4863/6412] were positioned in an interval of *20 ms* around their manually positioned location. Obviously, the improvement compared to the SCVQ within the hierarchical approach can solely be attributed to utterances with both voiced and unvoiced regions (the so-called `SIL-VOI-UNV` words as in Section 4.3.3), emphasizing the need for a reliable broad phonetic class segmentation preamble.

4.3.5 Variants

We have surveyed some implementation or application variants with respect to the SCVQ stage. Although they are not included in the hierarchical approach at this moment because they did not fulfil the performance expectations, they can be investigated in the near future again by implementing some small modifications. Skipping this section on first reading does not blur the coherent picture of the hierarchical approach.

Cepstral distance

Another distortion measure that has been extensively investigated in the framework of SCVQ, but delivered only inferior performance, is the cepstral distance. Essentially, the cepstrum coefficients correspond to a frequency smoothed representation of the log magnitude spectrum. We derived the first $(p+1)$ cepstrum coefficients making up the observation vector $\mathbf{c}(t) = (c_o, c_1, \cdots, c_p)'$ at frame t from the LPC coefficients by the recursion [Rabiner 93]

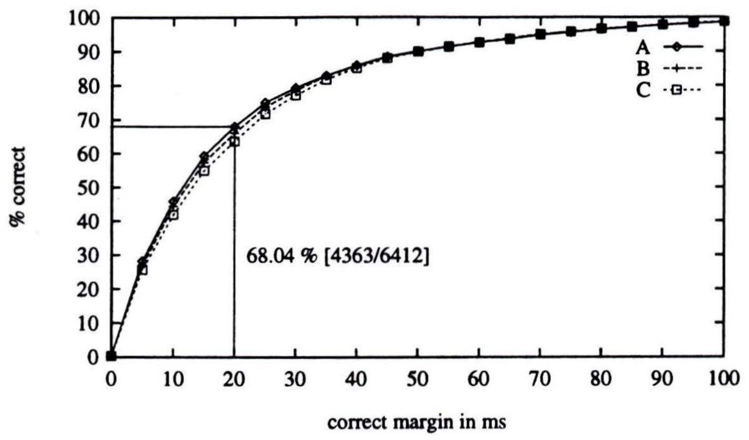

Figure 4.10: *Segmentation results for cepstral distance oriented SCVQ imposed to duration constraints as provided by a preceding broad phonetic class segmentation.* **A** *SCVQ with a cepstral distance results into 68.04 % correct within 20 ms [4363/6412].* **B** *SCVQ with a weighted cepstral distance results into 66.13 % correct within 20 ms [4240/6412].* **C** *SCVQ with a cepstral distance augmented with delta features results into 63.72 % correct within 20 ms [4086/6412]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurrences).*

$$\begin{aligned} c_0 &= \ln \sigma^2 \\ c_n &= -a_n + \tfrac{1}{n}\textstyle\sum_{k=1}^{n-1}(n-k)a_k c_{n-k} \quad ,n = 1,\cdots,p \end{aligned}$$

where σ^2 is the residual energy and $(a_0, a_1, \cdots, a_p)'$ is the LPC coefficient vector where $a_0 = 1$.

A truncated version of the weighted cepstral distance is used as distortion measure and can be incorporated in the *intra-segment* distortion of Equation 4.10 by considering a centroid $\hat{\mathbf{c}}_l = (\hat{c}_0, \hat{c}_1, \cdots, \hat{c}_p)'$,

$$d_l(i,j) = \sum_{t=i}^{j}\sum_{n=1}^{p}[w(n)c_n(t) - w(n)\hat{c}_n(t)]^2 + \alpha^2(c_0(t) - \hat{c}_0(t))^2 \tag{4.22}$$

in which $w(n)$ is a liftering function and α a energy scaling factor. A proper handling of the energy term c_0 in Equation 4.22 was needed to incorporate the important phonetic information-bearing of the energy contour for sound identification. Therefore, we separated the energy term from the rest of the cepstrum that defines the general LPC spectral shape. In order to compensate for the differences in dynamic range of the terms involved, we experimentally fixed α at 0.1.

It was shown that the variability of higher cepstral coefficients are more influenced by the inherent artifacts of the LPC analysis than that of lower cepstral coefficients. The variability of low cepstral coefficients is primarily due to transmission and speaker characteristics [Rabiner 93]. Thus, in order to de-emphasize these variabilities in the calculation of the cepstral distance, a cepstral weighting or liftering procedure $w(n)$ was suggested as a raised sine function [Rabiner 93]

$$w(n) = \begin{cases} 1 + \frac{W}{2} sin(\frac{n\pi}{W}) & \text{for } n = 1, 2, \cdots, W \\ 0 & \text{otherwise,} \end{cases} \tag{4.23}$$

where W = 10 is typical for 4 kHz bandwidth speech. However, we found that the use of the liftering procedure deteriorated the segmentation accuracy. Therefore, we excluded the cepstral weighting and put simply $w(n) = 1$, for $n = 1, \cdots, p$ and $w(n) = 0$ otherwise.

The truncated cepstral distances are Euclidian distances and thus, the calculation of the centroid is straightforward. The cepstral centroid $\hat{\mathbf{c}}_l = (\hat{c}_0, \hat{c}_1, \cdots, \hat{c}_p)'$ that minimizes the *intra-segment* cepstral distance is simply the arithmetic mean of all cepstral vectors within that segment, thus

$$\hat{c}_n = \frac{1}{b_{l+1} - b_l} \sum_{t=b_l+1}^{b_{l+1}} c_n(t) \quad , n = 0, 1, \cdots, p \tag{4.24}$$

where $c_n(t)$ denotes the n-th cepstral coefficient at frame t.

Considering instantaneous cepstral coefficients related to a single frame provides a locally correct representation of the spectrum. Cepstral time derivatives also contribute very useful cues and are found to improve the performance of a speech-recognition system [Rabiner 93]. A reasonable estimation of these time derivatives is to approximate it by a polynomial fit, as described in Section 4.4.1. Based on these computations, the observation vector in our SCVQ consists of $p+1$ instantaneous cepstral coefficients including the scaled energy term αc_0 augmented with $p+1$ cepstral time-derivatives. The use of Equation 4.22 as distance measure for this augmented cepstral observation vector is still legitimate [Rabiner 93]. However, we found that the incorporation of delta features into the SCVQ approach rather confused the segmentation.

After all, the usage of a cepstral distance rather than the Itakura distance does not deteriorate the segmentation accuracy that much. We feel that many segment modelling inaccuracies are caused by the inexactness of the LPC analysis. Investigations regarding isolated word recognition within a time warping scheme [Davis 80] revealed that FFT-based acoustic analysis methods better represent relevant short-term speech spectrum aspects than LPC-based implying higher recognition performance.

Frame acquisition in a pitch synchronous way

It is felt that the availability of frames that are acquired period by period over an interval that is centred at a glottal closure instant contribute to a higher performance. It is known that LPC based signal analysis methods suffer from a frame acquisition strategy in which each successive frame is shifted by a fixed amount. This equidistant spacing of analysis frames does not take local signal characteristics into consideration: influences of the decaying impulse response function originating from the previous pitch period are not properly taken into account, source excitation components (glottal closures) are at different unpredictable positions in the speech segments. In other words, these equidistant speech segments are considered poor candidate analysis frames for the estimation of LPC parameters. By taking instants of the glottal closure, simply called the epochs, as the centre of each frame, these influences are equally distributed over all analysis frames.

For the detection of the instants, we follow a method originally proposed by Ma [Ma 94]. These instants are efficiently computed by calculating a so-called ‘running’ Frobenius norm

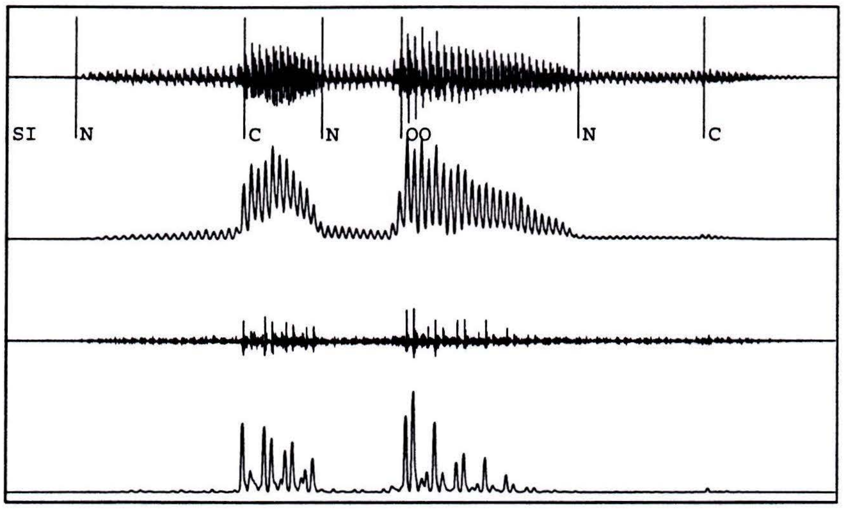

Figure 4.11: *Glottal closure instant detection by method proposed by Ma* [Ma 94]. *From top to bottom, a pre-emphasized utterance of the nonsense word 'nenoone', the 'running' Frobenius norm estimated from this pre-emphasized utterance, its LPC-residual signal, and the 'running' Frobenius norm of this residual are successively shown. Local maxima of the Frobenius norm of the pre-emphasized speech correlate with glottal closure instants.*

of signal matrices

$$\mathbf{X} = \begin{bmatrix} x(t-p) & x(t-p+1) & \cdots & x(t) \\ x(t-p+1) & x(t-p+2) & \cdots & x(t+1) \\ \vdots & \vdots & & \vdots \\ x(t-p+l-1) & x(t-p+l) & \cdots & x(t+l-1) \end{bmatrix} \tag{4.25}$$

The Frobenius norm of a matrix $\mathbf{A} = \{a_{ij} : 1 \leq i \leq m, 1 \leq j \leq n\}$ is defined as the square root of the summation of all squared entries of matrix $\mathbf{A}$, thus

$$\| \mathbf{A} \|_F^2 = (\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2) \tag{4.26}$$

By normalizing the Frobenius norm value by the factor $p \cdot l$, one acquires an energy measure. Maxima of the numerical value of the Frobenius norm nicely correlate with the epochs [Ma 94]. The location of the maxima in the Frobenius norm of the signal segment extending from $t = 1$ to $t = p + l$ is put at time point $t = p + 1$ because this maximum appears when the excitation point $x(p + 1)$ enters the first column of the signal matrix.

It seemed preferable to apply this method on pre-emphasized speech or on the LPC-residual signal. The latter was obtained by inverse filtering the sampled speech waveform by its estimated sequence of 12-order LPC filters. However, the use of pre-emphasized speech surpassed the use of the residue in performance. The LPC-residual gives rise to significant pulses other than those on the excitation moments and loss of epochs due to the inexactness of the LPC analysis. This is illustrated by Figure 4.11 in which the local maximum of the 'running' Frobenius norm estimated from the pre-emphasized utterance nicely corresponds to epochs, whereas the one derived from the LPC-residual shows, beside spurious local maxima, a considerable lack of local maxima in voiced regions.

However, the integration of frame acquisition via a pitch synchronous way in the SCVQ stage was never completely evaluated. Although we strongly believe in its strength for robust LPC analysis, the automatic epoch detection still has to contend with some difficulties that rather aggravate the segmentation process. Due to the inexactness of the method, an irregular pattern of pitch markers having misses and misplacements of epochs is provided. This has dramatic consequences for short segments, implying misalignment and propagation of errors in other positions. Regarding duration constraints for segments, a more elaborated approach has to be followed to map time points to specific observation vectors. A simple scaling involving an average pitch period is not sufficient due to the inexactness of the method.

4.3.6 Discussion

A minor modification to the proposed method may be by realizing that the Itakura distortion between two speech segments only models their general spectral shape without compensating for an overall spectral level (or gain). The role of this gain term is not explicitly modelled. However, a energy contour of the utterance may contain important information about the phonetic identity of the sounds. A flexible and proper way of incorparating this energy information over time, either the absolute power or minimum residual energy term (gain), has to be added to the Itakura distortion, as it could increase the segmentation accuracy.

As demonstrated in Section 4.3.4, the availability of reliable duration knowledge and/or centroids is a crucial factor in the performance of the SCVQ method. Further research must be focused on providing these durations; this can only be resolved by an improved and extended broad phonetic class segmentation preamble. An extra modification to the approach may be to extend it with an adequate solution to the centroid problem. A proposal is to apply an extra template matching scheme by means of centroids that are extracted from the SCVQ segmentation.

4.4 Hidden Markov Models

It is common practice nowadays to use Hidden Markov Models (HMM) for supervised segmentation of speech. This is because HMMs are able to capture the intrinsic variability of speech regarding both speaking rate and spectral characteristics of the speech sounds. The aim of using HMMs in our approach is to achieve even higher segmentation accuracy than can be obtained through SCVQ by capitalizing on this property.

In the present context of segmenting utterances according to a given phonemic transcription, we have defined a set of 50 phoneme-like units and used a separate HMM for each unit. Segmentation results then from time-alignment of the utterance against a sequence of HMMs corresponding to the transcription.

However, the HMM contain a large number of parameters that have to be tuned in order for the models to work well. This tuning is performed by a supervised training on a collection of speech segments, called the training set. Most often (e.g. [Alphen 92, Brugnara 92, Angelini 93, Farhat 93]), the training set is obtained by a manual segmentation performed on a subset of the same database or on a separate database. This however does not fit in the fully automatic procedure we want to develop. Therefore, initialization of the HMM will be performed on the phoneme-like segments provided by the previous stage of our hierarchical approach, namely the SCVQ. Although the segmentation accuracy of SCVQ is not perfect, it is still fairly high and allows to obtain high quality HMMs in a fully automatic way.

4.4.1 Outline of the method

A HMM consists of a finite set of states which are connected by transitions. Each transition is characterized by a probability of using it. With each state an emission probability distribution is associated that takes care of the production of an acoustic vector when visiting a state. The underlying statistical law of emitting a vector is necessary because the speech waveform can not be considered as a sequence of stereotyped patterns due to its intrinsic variability. The temporal structure in speech, on the other hand, is accounted for by the transition probabilities. A possibly prolonged stay in a state produces another vector and thus consumes a next piece of time. A majority of self-loop transitions in a path through a model, corresponds to a slow pronunciation while many jump transitions characterize fast speech.

HMM A Hidden Markov Model λ is defined by a set of M states $\{s_i | i = 1, \cdots, M\}$ extended with an initial state s_0 and a final state s_{M+1} which are non-emitting. Each state s_i can be connected by a transition to s_j. The probability of actually making this transition is given by $P(s_j|s_i)$. The visiting of a state s_i with $1 \leq i \leq M$ is accompanied by the emission of an acoustic vector $\mathbf{o}(t)$ according to a probability distribution $P(\mathbf{o}(t)|s_i)$ and is often called *local contribution.* The specification of $P(\mathbf{o}(t)|s_i)$ implies that states are not associated with one typical observation vector but with all possible vectors. By that, the vectors in an observation sequence can not be mapped *one-to-one* on the states in a state sequence since each vector can be generated by any state. In other words, the state sequence is *hidden.*

State sequence The event of visiting state s_i at time t is denoted by s_i^t. A state sequence through a HMM model is denoted by $S_1^t = s_0^0, s_{i_1}^1, s_{i_2}^2, \cdots, s_{i_t}^t$.

$S_1^T = s_0^0, s_{i_1}^1, s_{i_2}^2, \cdots, s_{i_{T-1}}^{T-1}, s_{M+1}^T$ is a path through a HMM model beginning at the initial state s_0 and ending at the final state s_{M+1}. The length of the path necessarily

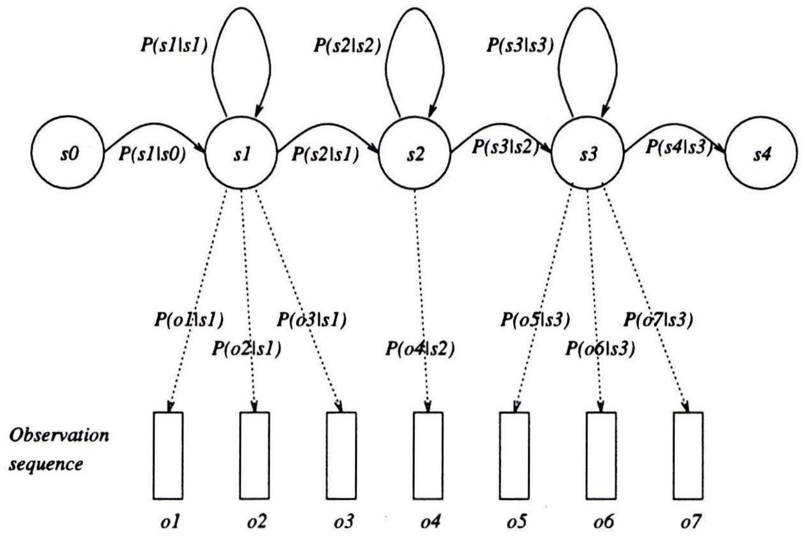

Figure 4.12: *A Hidden Markov as a generating device.*

equals the number of vectors emitted.

The particular form of the emission distribution $P(\mathbf{o}(t)|s_i)$ considered here is a multivariate multiple-mixture Gaussian distribution [Young 93]. Each observation vector $\mathbf{o}(t)$ is split into D independent data streams $\mathbf{o}(t)^{(d)}$ representing different information sources (e.g. instantaneous features and delta features). Each stream d is modelled by a separate Gaussian mixture distribution,

$$P(\ \mathbf{o}(t) \mid s_i\ ) = \prod_{d=1}^{D} \left[\sum_{m=1}^{M_d} w_{im}^{(d)} \mathcal{N}(\ \mathbf{o}(t)^{(d)}\ ;\ \mu_{im}^{(d)}, \mathbf{\Sigma}_{im}^{(d)}) \right] \tag{4.27}$$

where M_d is the number of mixtures in stream d, $w_{im}^{(d)}$ is the weight of the m'th mixture and $\mathcal{N}(.\ ;\mu, \Sigma)$ is a multi-variate Gaussian distribution with mean vector μ and *diagonal* covariance matrix $\mathbf{\Sigma}$. All experiments were conducted by using 4 mixtures ($M_d = 4$) for each datastream d.

Thus, two sets of parameters characterize the HMM: emission and transition probabilities. We shall assume that we have a separate HMM for each phoneme-like unit. From these building blocks we can derive a word HMM simply by concatenating the models of the constituting phoneme-like units as given by the transcription. Having defined the HMM as a stochastic speech production model, we can clarify how to use this model for aligning a known transcription with an utterance (segmentation), and how to estimate the optimal values of the HMM parameters (training).

All segments, except for the burst-like segments P2, T2, K2, B2, G2, and D2 are modelled by the topology as depicted in Figure 4.12. The states s_0 and s_4 are non-emitting. The first state represents (in an over-idealized way) the transition into the phoneme-like unit, the central state represents the stationary part in the phoneme-like unit, and the third state represents the transition out of the phoneme-like unit. It must be emphasized that this specific topology requires a minimum duration of a modelled phoneme-like unit of at least 3 frames (30 ms with a frame rate of 10 ms). All burst-like segments are modelled by a one-state model.

Now, we shall describe how we can use Viterbi decoding for time-aligning HMM word models with an utterance represented by an observation sequence. Next, we shall describe how the HMM parameters are tuned to their optimal values. This is achieved in two stages: a Segmental K-Means for HMM initialization followed by a Baum-Welch re-estimation that represent the core HMM training process. We conclude with a description of the front end of the HMM system that converts the sampled speech waveform into a discrete observation sequence.

Viterbi decoding as a segmentation algorithm

Giving an observation sequence $\mathbf{O}_1^T = \mathbf{o}(1), \cdots, \mathbf{o}(T)$ of a word, the *Viterbi decoding* uses a dynamic programming algorithm to find the most probable state sequence from the collection of all possible paths through the HMM word model λ [Rabiner 89, Rabiner 93]. Assume that we know the optimal *partial* path terminating in state s_j at frame $t-1$. Let $\delta_i(t)$ represent the probability along the optimal partial path that state s_i is reached at frame t and the partial observation sequence $\mathbf{O}_1^t$ is observed given a model λ. This probability $\delta_i(t)$ can be computed as a product of transition probablities between states and emission probabilities of acoustic vectors on the visited states. The following dynamic programming recursion lies at the basis of the Viterbi decoding and allows a recursive computation of $\delta_i(t)$ together with the determination of the optimal path.

$$\delta_i(t) = \max_{j=1,\cdots,M} \delta_j(t-1) P(\, s_i \mid s_j \,) P(\, \mathbf{o}(t) \mid s_i \,) \tag{4.28}$$

with the following boundary conditions

$$\begin{aligned} \delta_i(1) &= P(\, s_i \mid s_0 \,) P(\, \mathbf{o}(1) \mid s_i \,) && (4.29) \\ \delta_{M+1}(T) &= \hat{P}(\, \mathbf{O}_1^T \mid \lambda \,) && (4.30) \end{aligned}$$

where $\hat{P}(\mathbf{O}_1^T|\lambda)$ is the probability of the complete observation sequence along the optimal path in the word model λ.

If we keep track of the maximization decisions taken in Equation 4.28, we are able to find the state-time trajectory of the optimal path in the word model λ and, in particular, the time instants of the transitions between phoneme models, i.e. the segmentation points. This is essentially how we shall use the HMM for segmentation.

Segmental K-Means

The *Segmental K-Means* algorithm is a combination of the well-known K-means iterative procedure for clustering data[Wilpon 85] and Viterbi decoding. It is devised to provide reliable initial models for subsequent HMM training, in our case, by Baum-Welch re-estimation [Rabiner 93].

We assume we have for each phoneme-like unit a finite set $\mathcal{O}$ of observation sequences and a parameter specification for the corresponding phoneme-like HMM λ provided by a previous iteration of the algorithm.

Each observation sequence in $\mathcal{O}$ is segmented into the HMM states by application of Viterbi decoding as explained above. In the absence of an initial parameter estimate at the first iteration, the process is started by performing a linear segmentation of each observation sequence into the states. The state-alignment by Viterbi decoding assigns to each state s_j of the current model λ a particular segment of each observation sequence. Now, the K-Means algorithm [Wilpon 85] clusters the acoustic vectors belonging to state s_j into a set of M clusters (using a Euclidean distortion measure). In our case of multivariate multiple-mixture Gaussian densities, each cluster represents one of the mixtures. From this clustering an updated model $\hat{\lambda}$ can be derived by using sample estimations for the mixture weights, means and covariances. The transition probabilities $P(s_j|s_i)$ can be obtained by using the transitions statistics as provided by the Viterbi decoding alignment.

This process of model-reestimation is repeated until either convergence denoting statistical similarity between two successive model-estimations is met or a default number of iterations is reached. The convergence criterion is expressed by the reduction of the Viterbi aligment score between two successive models λ and $\hat{\lambda}$.

Baum-Welch reestimation

There is no known analytical procedure to come up with the optimal model parameters. Thus, we must resort to a method that adjusts the model parameters to satisfy a certain optimization criterion. We assume we have a finite (training) set $\mathcal{O}$ of observation sequences, and a provisional specification of a HMM λ modelling any type of speech unit. In our case, the parameter specification comes from the previous Segmental K-Means application. The HMM parameters are estimated by a hill-climbing procedure, known as *Baum-Welch reestimation*, which iteratively increases the likelihood $P(\mathcal{O}|\lambda)$ that the observation set $\mathcal{O}$ is generated by model λ [Rabiner 89, Rabiner 93].

Remark that this likelihood can be estimated by means of the training set $\mathcal{O}$ by considering

$$P(\,\mathcal{O}\mid\lambda\,) = \prod_{\mathbf{O}_1^T\in\mathcal{O}} P(\,\mathbf{O}_1^T\mid\lambda\,) \tag{4.31}$$

Let $\mathcal{S}$ be the set of all mutually exclusive paths of length T in the model λ. The probability of producing the sequence $\mathbf{O}_1^T = \mathbf{o}(1),\cdots,\mathbf{o}(T) \in \mathcal{O}$ given the model λ is the sum over all mutually exclusive paths $S_1^T = s_0^0, s_{i_1}^1, s_{i_2}^2, \cdots, s_{M+1}^T$ of length T in the model λ. Each individual probability of generating $\mathbf{O}_1^T$ along a path is the product of transition and emission probabilities associated with this path. More precisely,

$$\begin{aligned} P(\,\mathbf{O}_1^T\mid\lambda\,) &= \sum_{S_1^T\in\mathcal{S}} P(\,S_1^T\,)P(\,\mathbf{O}_1^T\mid S_1^T\,) &(4.32)\\ &= \sum_{S_1^T\in\mathcal{S}} \{\prod_{t=1}^{T+1} P(\,s_{i_t}\mid s_{i_{t-1}}\,)\prod_{m=1}^{M} P(\,\mathbf{o}(m)\mid s_{i_m}\,)\} &(4.33)\end{aligned}$$

in which $i_0 = 0$ and $i_{T+1} = M+1$.

The right hand side can be efficiently computed by a recurrence relation known as the *forward Baum-Welch recurrence* [Rabiner 89, Rabiner 93]. The whole procedure is complemented by an iterative parameter update until the likelihood converges to a critical point. However in practice, we conduct a pre-determined number of iterations of a single Baum-Welch reestimation procedure on the observation set $\mathcal{O}$.

HMM Front End

There is no consensus about the most adequate speech signal description for recognition or segmentation. Investigations showed substantial performance differences by using different parametric speech representations with regard to speech recognition[Davis 80]. It was concluded that Fourier-spectrum derived parameters preserve information that LPC derived parameters omit, especially for consonant spectra. Also, filters spaced at non-linear frequency distances (e.g. mel-frequency) allow better suppression of insignificant spectral variation in the higher frequency bands. Moreover, a cepstrum parameter set succeeds better than linear prediction coefficient or reflection coefficients. Another study [Alphen 92] showed that the segmentation with linear prediction coefficients as front end was less accurate than one with a Bark scaled filterbank analysis. Inspired by these arguments, we have decided to use a filterbank analysis with non-uniform spacing as described hereunder without really conducting comparative experiments with different front ends.

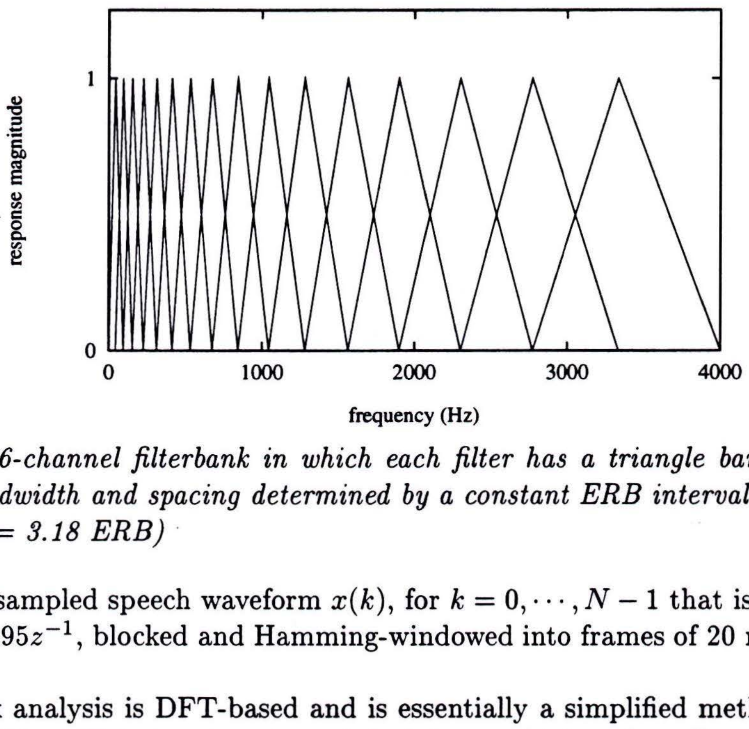

Figure 4.13: *A 16-channel filterbank in which each filter has a triangle bandpass frequency response with bandwidth and spacing determined by a constant ERB interval (spacing = 1.59 ERB, bandwidth = 3.18 ERB)*

We assume a sampled speech waveform $x(k)$, for $k = 0, \cdots, N-1$ that is pre-emphasized by the filter $1 - 0.95z^{-1}$, blocked and Hamming-windowed into frames of 20 ms, with a frame shift of 10 ms.

The filterbank analysis is DFT-based and is essentially a simplified method of designing bandpass filters. The DFT structure is exploited by combining the adjacent DFT-outputs according to a warping criterion to realize a 16-channel nonuniform filterbank analysis.

Each individual filter has a triangular bandpass frequency response and is applied to the power spectrum, as can be seen in Figure 4.13. The spacing as well as the bandwidth is determined by a constant Equivalent Rectangular Bandwidth (ERB) frequency interval in order to simulate the perceived spectrum[Glasberg 90]. The spacing is approximately 1.59 ERB and the width of the triangle is 3.18 ERB. The ERB frequency warping function is defined by

$$\mathrm{ERB}(f) = 21.4 \log_{10}(1 + \frac{f}{229}) \tag{4.34}$$

Dynamic features are incorporated by means of estimations of time-derivatives of the filterbank parameters, obtained by a polynomial approximation. This leads to smoother estimates than a direct difference method [Rabiner 93]. We assume a small trajectory of a vector element $\{o_n(\tau+t)\}_{t=-M}^{M}$. Its origin is shifted to τ, thus index t is with respect to τ and the time-derivative approximation is done for each τ. Also, all elements $o_n(\tau+t), n = 1, \cdots, p$ are individually taken into account. It is formulated by fitting a trajectory of a coefficient $\{o_n(\tau+t)\}_{t=-M}^{M}$ by a second-order polynomial $h_1 + h_2 t + h_3 t^2$ such that the fitting error

$$E = \sum_{t=-M}^{M} [o_n(\tau+t) - (h_1 + h_2 t + h_3 t^2)]^2 \tag{4.35}$$

is minimized. Expressions for h_1, h_2, and h_3 can be found by differentiating E with respect to h_1, h_2, and h_3, setting the result to zero, and solving the set of three simultaneous equations. Estimations of the first and second time-derivatives of $o_n(\tau+t)$ (i.e. delta and delta-square parameters) can then be obtained by differentiating the fitting polynomial, which gives

$$\frac{\delta o_n(\tau+t)}{\delta t} \simeq h_2 = \sum_{t=-M}^{M} \frac{t o_n(\tau+t)}{T_M} \tag{4.36}$$

and

$$\begin{aligned}\frac{\delta^2 o_n(\tau+t)}{\delta t^2} &\simeq 2h_3 \\ &= \frac{2\{T_M \left[\sum_{t=-M}^{M} o_n(\tau+t)\right] - (2M+1)\left[\sum_{t=-M}^{M} t^2 o_n(\tau+t)\right]\}}{T_M^2 - (2M+1)\left[\sum_{t=-M}^{M} t^4\right]}\end{aligned} \tag{4.37}$$

where

$$T_M = \sum_{t=-M}^{M} t^2 \tag{4.38}$$

The parameter M specifies the window width for the polynomial interpolation. We compute all time-derivatives fixing M at the number of frames corresponding to 20 ms. By considering a frame shift of 10 ms, M is fixed at 2. This means that 5 observation vectors are considerated in the computation. At the edges of the observation sequence, simple first order differences are used.

To summarize, the observation vector $\mathbf{o}(t)$ consists of a set of 16 filter channel outputs, augmented with an estimation of its first time-derivatives (delta features) and second time-derivatives (delta-square features). Morever, a short-time log energy value that is normalized by the maximum short-time log energy as found in the utterance is added to the vector, appended with estimations of its first and second time-derivatives. This results into a 51-dimensional observation vector with 4 data streams (16 instantaneous features, 16 delta features, 16 delta-square features, and 3 features associated with short-time log energy). Each data stream is modelled by Gaussian densities with 4 mixtures on each HMM state.

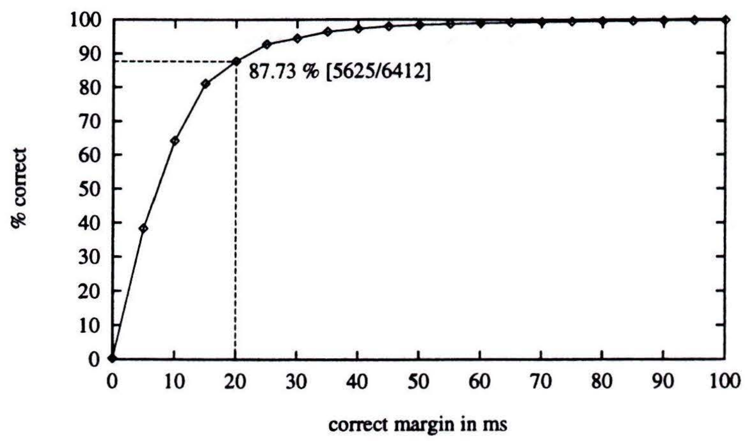

Figure 4.14: *Segmentation results of HMM segmentation by exploiting the hierarchical approach. It results into 87.73 % correct within 20 ms [5625/6412]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurences).*

4.4.2 Results

Each sampled speech waveform was pre-emphasized by the filter $1 - 0.95z^{-1}$, blocked and Hamming-windowed into non-overlapping frames of 20 ms, with a frame shift of 10 ms. Subsequently, it was converted to a discrete observation sequence by subjecting it to the filterbank analysis.

We have passed through the hierarchical procedure by conducting its first two stages (BPC and SCVQ) resulting into segmentation information that is correct for *69.31 %* considering a margin of *20 ms*. These provisional segments were provided for initializing the HMMs by means of a Segmental K-Means bootstrapping procedure on the whole speech inventory. As concluding stage, the HMMs were further trained by means of three Baum-Welch iterations by building a single composite HMM model for each observation sequence according to the transcription and training the model on the whole observation sequence.

In Figure 4.14, the final segmentation result by exploiting the hierarchical approach is shown. A segmentation accuracy of *87.73 %* correct is achieved within a margin of *20 ms*.

In order to investigate the contribution of Baum-Welch, several iterations of this algorithm were conducted (see Figure 4.15). We started with the provisional HMMs that one acquired by a Segmental K-Means initialization on the segment information provided by the first two stages of the hierarchical approach. After each Baum-Welch training iteration a segmentation task is conducted. Some observations can be drawn from this figure

- It is apparent that the bootstrapped HMMs (iteration 0) have already a high performance (86.82 %). We conclude that the provision of reliable segments for bootstrapping the HMMs is crucial for obtaining a high accuracy. Subsequent Baum-Welch fine-tuning of the HMMs, solely relying on the transcription and maximum-likelihood training, does not contribute that much.

- If we consider a correct margin larger than 20 ms, the segmentation performance tends to increase by the first few Baum-Welch iterations before it settles down. On the other hand, Baum Welch training rather deteriorates the performance by considering smaller segmentation discrepancies (correct margin of 20 ms).

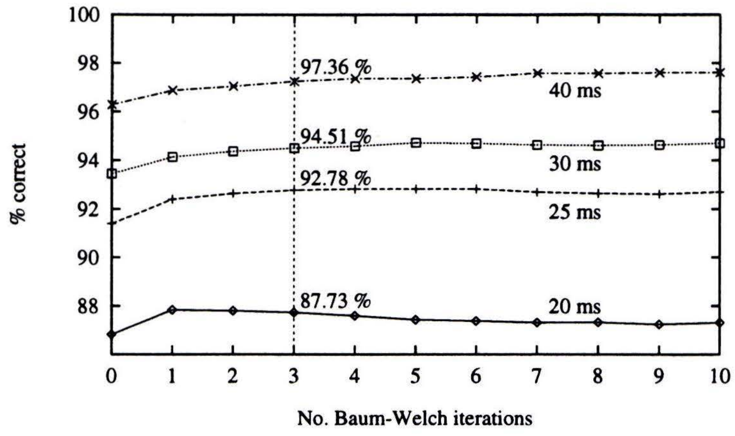

Figure 4.15: *Zero to ten iterations of the Baum-Welch training within the hierarchical approach. Four distinct correct margins are shown (20 ms, 25 ms, 30 ms, and 40 ms). Frame width = 20 ms, Frame shift = 10 ms.*

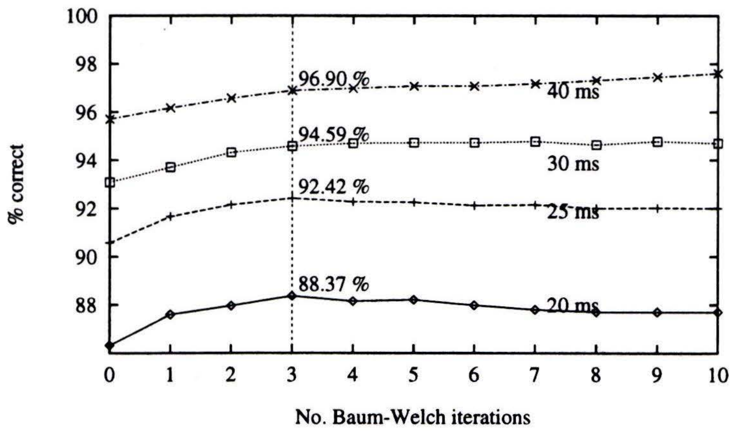

Figure 4.16: *Zero to ten iterations of the Baum-Welch training within the hierarchical approach. Four distinct correct margins are shown (20 ms, 25 ms, 30 ms, and 40 ms). Frame width = 20 ms, Frame shift = 5 ms.*

- A rather great performance difference exists when comparing a 20 ms with a 25 ms deviation. This may bring hope to improvement proposals in future studies to the hierarchical approach. Even a small accuracy improvement will already boost the global segmentation performance. A suggestion was made that a smaller frame shift during the filterbank analysis would bridge this performance gap. However, a frame shift of 5 ms did not bring us much further. As shown in Figure 4.16, it resulted in *88.37 %* correct within *20 ms* [5666/6412] at the expense of *much more* computer processing time.

We have adopted 3 Baum Welch-iterations as default. Although the performance decreases for margins of 20 ms by executing this number of Baum-Welch iterations in Figure 4.15, the performances increases for greater margins as shown in Figure 4.15 and 4.16.

The same clustering of labels into categories is conducted as described in Section 4.3.2, in order to look more closely at particular transitions. The set of categories is shown in Table 4.3. The two segments of the diphtongs and unvoiced plosives are 'glued' together again to form a single label. This means that we disregard the boundaries between these distinct segments in the results and statistics.

The differences between the automatically obtained boundaries and the manually positioned boundaries considering the categories are examined by calculating the mean difference and the standard deviation. In Table 4.5, these statistics are shown by considering the left and the right boundary of each category separately. It is apparent that liquids (semi-vowels) and nasals show a great variation in their boundary placement. Furthermore, vowel onsets are on the average later and their endings are on the average earlier compared to the manual segmentation. The table also shows that fricative and silence beginnings are on average earlier and their ending on the average later than the manual ones. Nasals are on the average stretched too long.

	Left boundary			*Right boundary*		
category	no	mean	std.dev.	no	mean	std.dev.
Vow	2276	4.33	15.28	2270	-3.48	16.57
Plo	716	-1.11	10.98	874	-0.05	12.86
Fri	649	-5.20	11.10	729	2.97	9.82
Liq	611	3.11	22.40	671	1.13	21.84
Nas	374	-0.19	14.86	390	5.36	23.23
SI	731	-13.02	15.47	423	2.25	9.24
	5357	-0.37	16.46	5357	-0.37	16.46

Table 4.5: *Number, mean, and standard deviation (in ms) of automatic obtained boundaries as compared with the manually positioned boundary classified by phonetic categories. Both the statistics of the left and right boundary of each category is shown.*

A detailed overview of the boundary discrepancies between categories by considering a correct margin of *20 ms* is shown by Table 4.6 in which vertically the category to the left of the boundary is indicated, and horizontally the category to the right of the boundary. The score p/q in each entry of the matrix means p correct out of q occurrences. In the bottom row and the right most columns, summations over respectively the column and the row is given.

In the bottom-right entry, the total segmentation performance is given. This percentage is slighty different (smaller) compared with the percentage as shown in Figure 4.14 due to the disregarding of boundaries within plosives and diphtongs.

	Vow	**Plo**	**Fri**	**Liq**	**Nas**	**SI**		
Vow	55 / 80	465 / 503	388 / 426	289 / 394	244 / 261	451 / 606	1892 /2270	83.35 %
Plo	733 / 740	13 / 15	25 / 27	37 / 42	8 / 8	15 / 42	831 / 874	95.08 %
Fri	572 / 595	48 / 49	15 / 25	31 / 33	13 / 14	10 / 13	689 / 729	94.51 %
Liq	447 / 530	31 / 37	32 / 47	10 / 14	11 / 12	8 / 31	539 / 671	80.33 %
Nas	232 / 277	18 / 33	14 / 23	5 / 14	1 / 4	17 / 39	287 / 390	73.59 %
SI	51 / 54	78 / 79	90 / 101	110 / 114	75 / 75	0 / 0	404 / 423	95.51 %
	2090 /2276	653 / 716	564 / 649	482 / 611	352 / 374	501 / 731	4642 /5357	
	91.83 %	91.20 %	86.90 %	78.89 %	94.12 %	68.54 %		86.65 %

Table 4.6: *Detailed overview of the correct boundary placements for each category by considering a correct margin of 20 ms as achieved by the hierarchical approach. [p/q] means p correct out of q occurrences.*

Although the hierarchical approach achieves good results for many transitions by considering a margin of 20 ms, some sources of concern can be summarized. It is apparent that most misaligments occur in well-defined pairs of categories. In general, this means that only these indicated transitions need further (manual) checking, substantially reducing the work.

- The vowel-vowel transitions (62.50 % correct [55/80]). Many of these transitions occur at word boundaries (i.e. `Z EI1 EI2 E R X E T1 T2`) and are realized with a glottal stop. Although this temporary amplitude lowering is clearly visible in the waveform, it is not transcribed as such (see also Appendix A).
- The beginning of concluding silences (68.43 % correct [501/731]).
- The transitions involving liquids (semi-vowels), especially the transitions from vowel (schwa) to liquid (73.35 % correct [289/394]).
- The transitions from vowel to voiced fricative (75.00 % correct [84/112]).
- The detection of vowel onsets within nasals (83.75 % correct [232/277]) and liquids (84.34 % correct [447/530]). This is mainly attributed to the schwa onsets within nasals (75.17 % correct [109/145]) and liquids (80.66 % correct[196/243]). Especially word initial nasals and liquids (i.e. `Q` and `J`) are extended too far within the following schwa. It is observed that these boundaries are troublesome. Their misplacements are persistent considering larger correct margins. Moreover, it is apparent that this particular kind of misalignments scarcely occur at the SCVQ stage and, thus, are introduced by the HMMs.
- The closure beginning of unvoiced plosives (88.76 % correct [387/436]). Although this can not be derived from Table 4.6, we did not want to deny the reader this information because of our special attention already paid to these closures by splitting the unvoiced plosives into two segments.

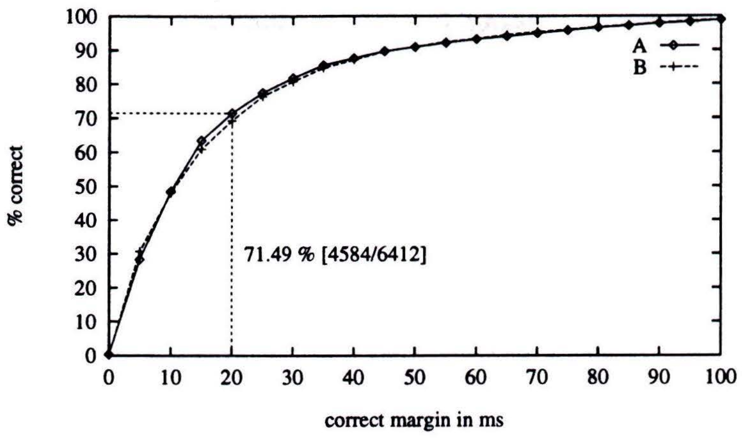

Figure 4.17: *Segmentation results by using a set of 'fresh' HMMs for each observation sequence. The HMMs were bootstrapped by using the segmentation as provided by the first two stages of the hierarchical approach.* **A.** *Fresh HMMs for each observation sequence resulted into 71.49 % correct within 20 ms [4584/6412].* **B.** *The bootstrapping segmentation was 69.31 % correct within 20 ms [4444/6412]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurences).*

4.4.3 Variants

For the purpose of segmentation, HMMs have to be made as specific as possible for the training speech material. In particular, local signal characteristics have to be captured and attributed to a specific model. This approach can be pushed to the extreme that reserves a set of 'fresh' HMMs for each *complete* observation sequence. Still, each 'fresh' HMM models all *partial* observation sequences of corresponding labels within a single utterance. However, this implies fewer observations for each model which means that simpler HMMs have to be used in order to allow reliable estimations of their parameters. In this experiment, each observation sequence was designated a new set of HMMs. The emission probabilities were modelled by single mixture Gaussian densities. The HMMs were bootstrapped by means of Segmental K-Means (that reduces to a simple Viterbi training) using the segment information as provided by the first two stages of the hierarchical approach. Three Baum-Welch iterations on the observation sequence rather deteriorated the segmentation accuracy, so these were not applied. Although we did not expect miracles from this approach considering the lack of sufficient and reliable training data for each HMM, we were at the end disappointed about the ultimate result. Only a small fraction of the boundaries were placed correctly compared with the inputted segmentation. The segmentation accuracy was *71.49 %* [4584/6412] correct within *20.0 ms*. In Figure 4.17, the segmentation accuracy considering distinct margins is shown along with the segmentation that served as bootstrap material.

Two extra options regarding this HMM variant were also investigated

Intermediate stage

A possibility is to consider the segmentation provided by fresh HMMs as an intermediate stage in the hierarchical approach. However, the usage of this segmentation as bootstrap material for initializing models that cover all observations did not bring any relief; the segmentation accuracy was *86.51 %* correct within *20 ms*.

Specializing boot-strapped models

Starting from 'fresh' models that have not yet undergone any training at all is too much, considering the lack of training material. Therefore, having reliable HMMs at our disposal (i.e. the HMMs that result from the hierarchical approach), some extra Baum-Welch iterations by using only the observation sequence at hand may specialize the models and adapt them just that little bit. In a preliminary experiment, we used a set of bootstrapped HMMs as provided by the Segmental K-Means stage as starting point. As mentioned in Section 4.4.2, these models already resulted into *86.19 %* [5533/6412] correct within *20.0 ms*. Three Baum-Welch iterations focused only on the observation sequence to be segmented and a successive Viterbi segmentation resulted into *87.31 %* [5598/6412] correct within *20.0 ms*.

4.4.4 Limiting cases

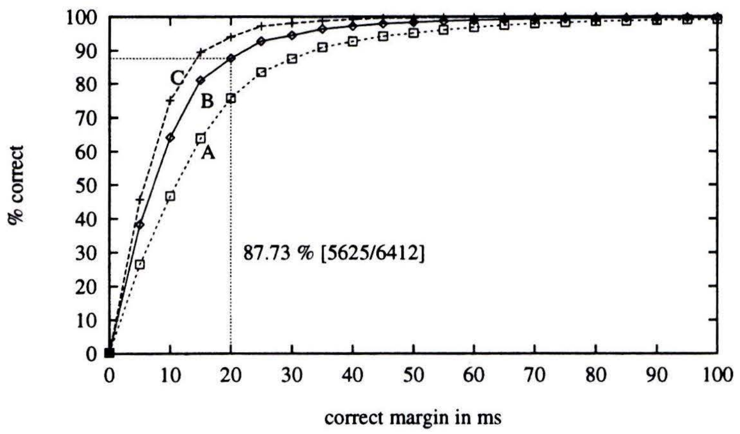

Figure 4.18: *Limiting cases in perspective with the hierarchical approach.* **A.** *HMMs initialized on linear segmentation of inventory results into 75.67 % correct within 20 ms [4852/6412].* **B.** *Hierarchical approach results into 87.73 % correct within 20 ms [5625/6412].* **C.** *HMMs initialized on the manually segmented inventory results into 94.03 % correct within 20 ms [6029/6412]. Horizontally, the correct margin is successively extended by 5 ms. Vertically, the percentage of boundaries that fall within a margin is specified ([p/q] means p correct out of q occurrences).*

Two straightforward limiting cases for the HMM application for segmentation can be made up. One is to start from the manual segmentation. We have used the manual segmentation for both the Segmental K-Means initialization and Baum-Welch training stage. These HMMs are subjected to the segmentation task. The resulted accuracy draws the borderline of how accurate one can get with these type of models. At the other extreme, we can follow the approach as advocated by many researchers by simply bootstrapping the HMMs by a linear segmentation, that uniformly attributes the observation vectors to the states of the HMM. A Segmental K-Means initialization procedure considering this linear segmentation delivers provisional HMMs of a poor quality. Subsequently, Baum-Welch iterations must correct the errors introduced in the initialization phase. The performance difference between this 'blind-folded' approach and our hierarchical approach expresses the contribution of the BPC and SCVQ stages. In Figure 4.18, the hierarchical approach is compared with its two limiting cases. All experiments were conducted by considering three Baum-Welch iterations in the training process.

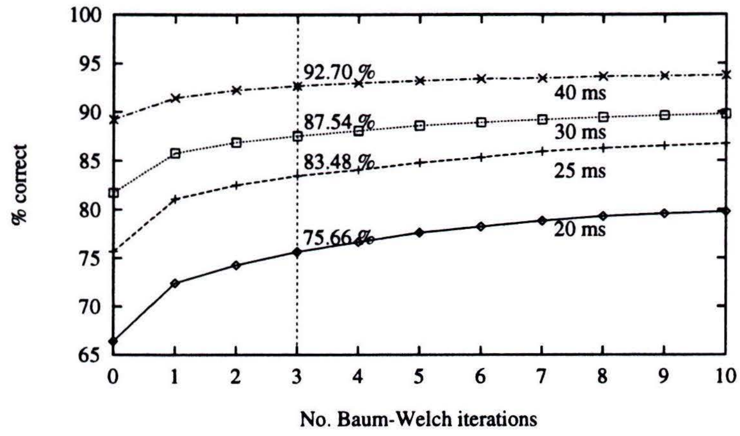

Figure 4.19: *Zero to ten iterations of the Baum-Welch training. The HMMs were initialized by means of a linear segmentation. Four distinct correct margins are shown (20 ms, 25 ms, 30 ms, and 40 ms). Frame width = 20 ms, Frame shift = 10 ms.*

For comparison with the experiment in Section 4.4.2, we have investigated the contribution of each Baum-Welch reestimation to the segmentation accuracy. We started with the provisional HMMs acquired by a Segmental K-Means initialization on the linear segmented data. The results are shown in Figure 4.19 and some remarkable observation can be drawn.

- In contrary with Figures 4.15 and 4.16, the performance increases for each extra Baum-Welch iteration. Although the accuracy never exceeds 80 % correct considering a 20 ms margin, its recovery from the linear segmentation is quite remarkable.

- Provisional models at iteration 0 have a poor quality, delivering only *66.41 %* correct within *20 ms*. However, only a single Baum-Welch iteration already boosts the accuracy to *72.41 %*.

4.4.5 Discussion

Our method of training HMMs on the data and exploiting the resulting HMMs on the same data is comparable with one of the testing principles one encounters during the development of a speech recognizer. HMMs designated for speech recognition are subjected to a recognition task on the training data in order to investigate how well they reflect the characteristics of the training material. This so-called self-test may indicate the maximum feasible performance one can obtain with that set of HMMs. However, HMMs for the purpose of speech recognition are not allowed to specialize too much on the training material (i.e. become too training set specific). This may degrade the performance on the test set, not to mention the performance decrease one gets when the models are utilized in a 'real world' application. In contrast with that, our approach for segmentation has a rather opposite point of view: we are striving to capture as much training material characteristics as possible for achieving high performances because the segmentation is done on the same data. The idea of attributing local signal properties to specific models as described in Section 4.4.3 has to be further explored. We strongly recommend to make the HMMs as specific as possible to the observation sequence at hand.

It is observed that most alignment errors are associated with well-defined phoneme categories. As a result of this, it is not necessary to check all boundaries but restrict oneself to those category pairs containing liquids, semi-vowels, and schwas. In addition, one can improve specific inadequacies by modelling segments according to own insights and views (both phonological and pragmatic). For instance, different phoneme categories can be modelled by distinct topologies or structures. Corresponding segments in distinct phonetic contexts can have different models. One can also allow more syllable-oriented modelling avoiding the rather *ad hoc* definition of a phonemic segment.

Chapter 5

Discussion

The hierarchical approach for automatically segmenting speech is a clear stepping stone for further research. It demonstrates that a good bootstrapping procedure for HMMs is crucial for obtaining high segmentation accuracies. Moreover, we feel that small improvements at the first two stages of the hierarchical approach contribute to the overall performance of the automatic segmentation. Some of these feelings are reflected into the recommendations for further research.

At this moment, the hierarchical approach consists of three successive stages, each with its' own characteristics.

A Broad Phonetic Class Segmentation segments an observation sequence into three phonetic classes (unvoiced, voiced, silence) according to a phonetic transcription. The observation is a sequence of a five-dimensional vectors containing signal measurements that roughly characterize the three phonetic classes. Iteratively, the segment boundaries are found by minimizing a total Euclidean distance, and subsequently the centroids representing the phonetic classes are updated. The method results into *80.42 %* correct placements of *broad phonetic class* boundaries considering a margin of *20 ms.*

A Sequence Constrained Vector Quantization segments each broad phonetic class segment into its' constituting phone-like units. It clusters the observation sequence into a pre-selected number of non-overlapping phoneme-like segments by minimizing the Itakura distortion. The method results into *69.31 %* correct placement of *phoneme-like unit* segments considering a margin of *20 ms.*

A Hidden Markov Model System bootstrapped by segments as provided by the SCVQ method, serves as concluding stage and delivers the final phoneme-like unit segmentation. The hierarchical approach result into *87.73 %* correct placement of *phoneme-like unit* boundaries considering a margin of *20 ms.* Considering a margin of *25 ms* results into a segmentation accuracy of even *92.78 %.*

The quantitative performance metrics and overall design emphasize the rather unique position of the hierarchical approach concerning performance and required *a priori* knowledge.

5.1 Performance evaluation

A first recommendation is to verify whether possible idiosyncrasies of the speech inventory at hand play some rule. The hierarchical approach has therefore to be cross-validated by subjecting it to other speech inventories. As the inventory used in this project is a subset of a database with a much richer pre-determined set of diphones and phoneme-like units, a first strengthening of the experimental conditions is to conduct experiments on the whole database. A second step may consist in using an isolated word corpus recorded from another speaker or a collection of speakers. An ambitious, but concluding experiment is to apply the technique to a corpus with fluently spoken utterances. One can think about the PHONDAT [PHONDAT 92] corpus, a manually segmented database that contains continously spoken German sentences recorded from several speakers for the purpose of diphone preparation.

5.2 More broad phonetic classes

The provision of anchor points of broad phonetic classes can be extended by defining more classes such as nasals and liquids. As observed in the final segmention results, boundaries between category pairs containing liquids need some improvement. This may be remedied by marking these speech parts as liquids in an early stage in the hierarchical approach, during the broad phonetic class segmentation.

5.3 Finding centroids

The availability of reliable centroids in the Sequence Constrained Vector Quantization is crucial for the performance of this stage. Therefore, it must be further explored. A minor modification is to apply an extra template matching between the SCVQ stage and the HMM stage. It may extract centroids from the segmentation as resulting from the SCVQ stage. In addition, a more elaborated clustering technique (originated from the Vector Quantization field) for acquiring centroids must be investigated.

5.4 Frame acquisition via glottal closure instants

The speech analysis method is still block based by spacing analysis frames uniformly. This may result into a mixing of dynamic characteristics of speech in each frame and, hence, an unreliable signal analysis. Therefore, a frame positioning synchronized with glottal closure instants (epochs) provides means for analyzing speech segments corresponding to these excitation events.

5.5 Hidden Markov Model Structure

The set of HMMs can easily be adapted to new insights as provided on the detailed examination of the results. It is observed that most misalignment errors occur in well-defined phoneme category pairs. This provides a setting in which the HMM system can be incrementally improved by iteratively modifying small parts of it and evaluating the performance (in/de)crease. HMM modelling other speech segments such as glottal stops can also be provided. In addition, different phoneme categories can be modelled by distinct topologies or structures. Corresponding segments in distinct phonetic contexts can be modelled by different models. One can also allow more syllable-oriented modelling avoiding the rather *ad hoc* definition of a phoneme-like segment.

5.6 Starting form an orthographic transcription

One of the severe problems that has to be addressed in future studies is the possible mismatch one encounters between the prescribed transcription and the actual acoustic realization of the utterance. This phenomenon already shows up at the first stage of the hierarchical approach, i.e. the broad phonetic class segmentation. Beforehand, it is never assured that a speech segment possess the acoustic-phonetic properties that are associated with a label: all depends on the actual realization made by the speaker. There is a large variability in speech sound realizations due to human adjustments, most of them are formalized by phonological rules[Jongenbur 91]. Some of them can be considered compulsory such as the assimilation of voice in the cluster */sz/* in the word 'waszak'. It can be clearly seen in Figure 5.1 that the segment `Z` is realized with strong unvoiced characteristics under the influence of its phonetic context. Another example of an adjustment is the insertion of schwa `C` between the plosive `K1 K2` and the nasal `N` in the word 'knieen', as shown in Figure 5.2. Although this insertion is very common in Dutch, it was not transcribed as such in the first place. On the other hand, utterances in the inventory were encountered having mismatches that could not be simply clarified by application of a phonological rule. In some cases, it was felt that the speaker did not obey the prescribed transcription during the recording session.

In order to address these phenomena, acoustic-phonetic events have to be automatically detected by using only an orthographic transcription of the utterance. At this moment, we have circumvented it by examining the speech inventory and correcting mismatches as much as possible in the phonetic transcription (see Appendix A).

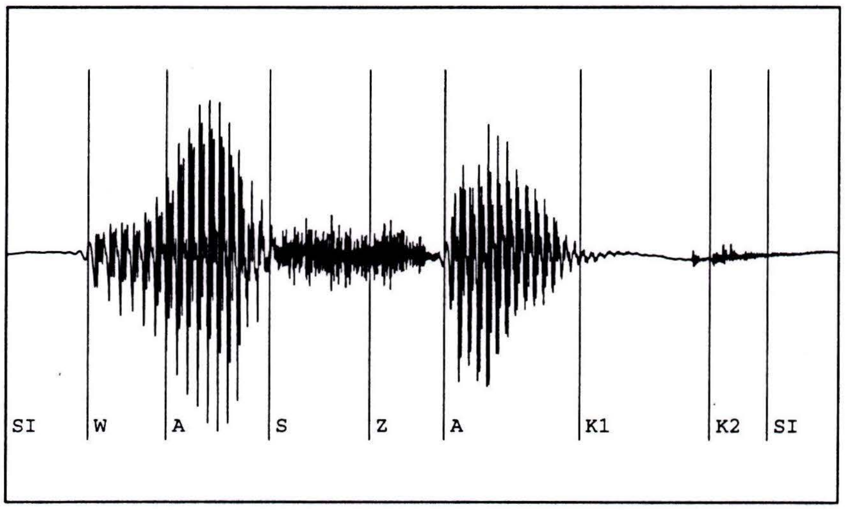

Figure 5.1: *Utterance of the word 'waszak' (soiled-linen bag) with assimilation of voice in its* `S Z` *cluster.*

Thus, it is a false assumption that each word can be phonemically represented by a unique

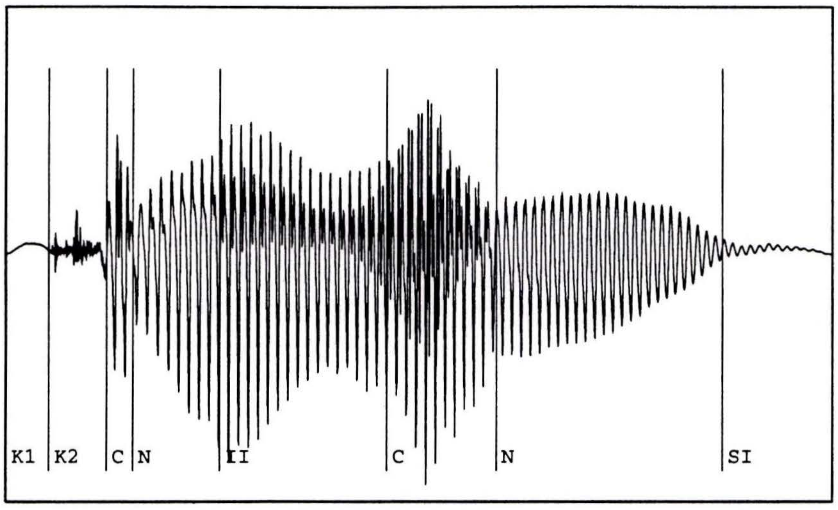

Figure 5.2: *Utterance of the word 'knieen' (knees) with a schwa* C *(svaribhakti vowel) inserted between the plosive* K1 K2 *and the nasal* N.

phoneme label string. Consequently, it would seem preferable to start from a given orthographic transcription of the utterance (and not from an unreliable phonetic transcription). In this case, the task of the HMM is to perform both segmentation and labelling. A plausible implementation of this idea is to represent all possible pronunciations of an utterance optionally by a finite state network of pronunciations (transcriptions). In addition, the mismatches between labels and realizations also occur during the training stages and will affect the quality of the HMMs. In this respect, an unsupervised training procedure (with no explicit label information) has to be designed. For work along this line see e.g. [Brugnara 92].

5.7 Linear Discriminant Analysis

It is known [Haeb 92] that the application of Linear Discriminant Analysis (LDA) results in improved performance for speech recognition. It must be investigated whether segmentation techniques can also benefit from this. LDA is a feature selection technique in statistical pattern recognition [Fukunaga 90]. It improves the class separability and reduces the dimensionality of the feature space by a linear transformation. The main question that arises (beside the implementation strategy concerning the computation method of the between-class scatter matrix) is how to define an appropriate class that will be subjected to LDA. Results of previous training and segmentation with class annotations referring to the class definition at hand are required for applying a LDA.

5.8 User interface

The hierarchical approach is implemented by a set of ANSI-C programs that can be invoked via a UNIX-shell script in an off-line, batch-oriented way. However, the interaction and parameter setting of these programs require expert knowledge. This may not be a good facility for the work involved with segmentation and the preparation of a diphone inventory. Because of the fact that automatic segmentation is still affected by incorrect boundary placements, designing a user interface that nicely interacts with the hierarchical approach would result into a truly professional work bench for speech researchers.

Chapter 6

Software Tools

This chapter represents an overview of the software programs that are written in the course of the project. As a matter of fact, they can be considered as the evaluation objects of the automatic segmentation techniques as described in this document. This software description is primarily meant for the benefit for future users or programmers and can be considered a supplement to this report.

The software tools are written in ANSI-C on top of a software library from a third party, namely the Hidden Markov Toolkit (HTK) [Young 93]. The HTK is a suite of libraries and programs for manipulating continuous density Hidden Markov Models (HMM). In order to be file-compatible with this toolkit we have conformed ourselves to the file formats and data structures as they are defined within HTK. Beside the creation of totally new programs, we have fixed some minor bugs within HTK that we encountered during extensive HTK usage and adapted some data structures and HTK tools to our specific needs. This has led to a revised version of the HTK originated from the 'last academic' version V 1.4A.

Unfortunately, the *HTK Toolkit Reference Manual* has some pitfalls regarding its information structure. It rather carries information around the document. We have tried to collect all required information, written them down in a condensed way while adopting the same format and style, and made references to the corresponding sections in the HTK manual. A little *a priori* knowledge on how speech recognition is implemented by HTK is recommended.

It must the emphasized that the overview as presented here is oriented towards speech segmentation. The HTK tools are developed for the purpose of speech recognition implying an enormous amount of reading material in this chapter. If the purpose is solely speech segmention, some sections can be skipped by first reading. We devote a section for each tool and try to explain its functionality, usage and how it is implemented. Especially, the concluding part dealing with the implementation is meant for programmers and it is recommended to carefully examine the source code simultaneously. We conclude this chapter with a small tutorial in order to illustrate the practical usage of the tools in the hierarchical approach.

Hereunder, a list of software tools is given in alphabetical order. These software tools merely represent the instruments needed to conduct the segmentation task as described in this document. They fall into three different categories: new tools that are developed by using the HTK library, tools that are extensions to existing tools within HTK, and HTK tools. The last category is not extensively documented in this report (see [Young 93]).

Assess

A tool for assessing the performance in segmentation accuracy of an automatically obtained segmentation. It takes pairs of labelled transcription files from which one may be a (manually segmented) reference. First, it performs a dynamic programming match between them in order to assure compatibility. Subsequently, it compares each boundary as specified in one file with its correspondent from the reference file and draws statistics from this. It outputs global cumulative statistics about the accuracy plus optionally information pinned to specific transitions. Also, LaTeX-compatible text files can be generated.

BroPhon

A tool that segments an input utterance into three broad phonetic classes (silence, unvoiced, voiced) according to a phonetic transcription, the specification which phoneme belongs to which broad phonetic class, and some global duration statistics allowed for each phoneme. It is a direct implementation of the method described in Section 4.2.1.

FrontEnd

A tool that encodes an input utterance into a discrete observation sequence that can be subjected to the HMM system. The parameterized form of the speech is a FFT-based filterbank in which the spacing and bandwidth of the triangular filters are determined by an Equivalent Rectangle Bandwidth warping function. Both the calculation of delta and delta-square features are supported. It is a direct implementation of the front end as described in Section 4.4.1.

GenMarkov

A tool that generates a HMM definition file according to syntax construction rules. The generated file serves as a topology and structure description of an HMM that can be provided to training tools.

htk2ipo ipo2htk

Two conversion tools that convert data formats as standardized within HTK to the IPO/OTS speech data formats and vice versa.

Implicit

A tool that tries to segment an input utterance into a sequence of consecutive spectral stable portions by applying a correlation technique between observation vectors. It can also be used to 'polish' a given segmentation in order to require more exact positions. It is a direct implementation of the 'implicit' method proposed by van Hemert [Hemert 87, Hemert 85].

Pitch

A tool that calculates the pitch markers indicating glottal closure (epoch) instants of an input utterance. It is a direct implementation of the method as described by Ma [Ma 94].

SegKMeans

A former HTK tool that is used to initialize HMM on a set of labelled speech segments by means of a Segmental K-Means algorithm. In first instance, it is meant to provide sufficiently good initial HMM estimates that can be further subjected to stronger training such as Baum-Welch re-estimations. In fact, this tool is a small extension to the **HInit** tool as present in the HTK (see also Section 4.4.1).

SeqQuant
A tool that segments an input utterance in contiguous non-overlapping segments by means of a so-called Sequence Constrained Vector Quantization. The process is controlled by a phonetic transcription. Extra *a priori* knowledge that is helpful in performing its task properly is by giving global duration statistics allowed for each phoneme. Moreover, a broad phonetic class segmentation can also be supplied to the process. It is a direct implementation of the method as described in Section 4.3.1.

HERest
A HTK tool that does a single Baum-Welch reestimation on a collection of labelled (parametrized) speech material. It takes a set of provisional HMMs and trains them simultaneously using a training set of observation sequences and their corresponding transcriptions. For each sequence, the transcription is used to build a large composite HMM model and Baum-Welch is used to train the model on the whole utterance. No segment information is used during the whole process (see also Section 4.4.1). This tool is not further documented in this chapter. A complete description of **HERest** can be found in the *HTK manual* (see pp. 65-71 [Young 93]).

Viterbi
A former HTK tool that performs continuous speech Viterbi decoding with finite state syntax constraints, beam search, and garbage collection. Throughout the project, we have used it solely as segmentation tool by restricting the search process to very simple finite state network, namely the transcription (see also Section 4.4.1). In fact, this tool is an extension to the **HVite** tool as present in the HTK. We have implemented a lattice N-best algorithm and bigram with word categories for the purpose of speech recognition [Pauws 93].

6.1 File formats

6.1.1 Label files

A HTK label file uses a text file consisting of one or more levels where each level is separated by a / character

    transcription = level { "/ " level }

Each level is a sequence of text lines, each defining a label

    level = label { label }

A label is specified in the format

    label = [start end] name [score]

where start denotes the start time of the labelled segment in 100ns units, end denotes the end time in 100ns units, name is the name of the segment and score is a floating point confidence score. All fields except name are optional. If start and end are both missing then the label file is treated as a simple symbolic transcription. The optional score would typically be a log probability generated by a recognition (segmentation) tool.

Actual label names can be any sequence of characters but must begin with a non-digit character other than the level separator /. However, the - and + characters are reserved for identifying left and right context, respectively, in a context-dependent label (triphones).

The following sample text represents a label file with segment information of the manual segmentation at the phoneme level of the word 'nenoone'. Observe that this label file has only one level.

```
0         682500    SI
682500    2343750   N
2343750   3116250   C
3116250   3901250   N
3901250   5657500   OO
5657500   6900000   N
6900000   8223750   C
```

All tools described in this chapter use this label file format. Also, **Pitch** writes its pitch-marker files in this format.

In addition, tools reading in label files have also access to label file formats from other vendors. These formats are solely for input.

TIMIT a text file format used in the prototype version of the US DARPA TIMIT database produced by Texas Instruments and MIT. TIMIT assumes that the corresponding audio data is sampled with 16 kHz.

SPED a non-text label file format.

SCRIBE the UK SCRIBE database label file format which is compatible with the European SAM standard.

6.1.2 A priori knowledge files

The tools **BroPhon** and **SeqQuant** both require some *a priori* knowledge that must be provided by a text file. The following text file format is used with one label and its data stored per line

```
name ("UNV"|"VOI"|"SIL") ["PLOS"] mindur maxdur
```

where name denotes the name of the label, UNV, VOI, and SIL denote the set of disjoint broad phonetic classes the given label belongs to, PLOS denotes an extra optional broad phonetic class, mindur denotes the minimum allowed duration of the given label in ms units, maxdur denotes the maximum allowed duration of the given label in ms units.

Actual label names can be any sequence of characters. However, the - and + characters are reserved for identifying left and right context, respectively, in a context-dependent label (triphones).

The following sample text represents a part of a file that can be provided to the two tools,

SI	SIL		36.1	83.5
P1	SIL		37.0	82.8
P2	UNV	PLOS	10.0	44.1
B1	VOI		38.9	83.1
B2	VOI	PLOS	11.8	22.4
F	UNV		51.1	120.3
V	VOI		52.6	100
S	UNV		69.8	199
Z	VOI		67.4	120.6
AA	VOI		133.9	203.7
EE	VOI		104.8	183
A	VOI		67.7	112.9
E	VOI		70.1	107.7

6.1.3 Sampled speech data files

Several file formats for sampled speech data are supported.

AIFF Apple Interchange File Format.

HTK HTK format.

SUNAU8 8 kHz 12 bit μ-law NeXt/SUN audio format.

TIMIT the format used in the prototype version of the US DARPA TIMIT database produced by Texas Instruments and MIT.

NIST the standard format being promoted by the National Institute of Standards and Technology. The DARPA Resource Management Database and the full version of the US DARPA TIMIT database are in this format.

SCRIBE the UK SCRIBE database format which is compatible with the European SAM standard.

SDES1 the *Sound Designer 1* format developed by Digidesign for use in multimedia applications.

In addition, unknown audio file formats can be read provided that the overall structure concerning header followed by data is known. Setting the following environment variables enables this feature.

HDSIZE	- number of bytes in header
SAMPSIZE	- number of bytes per sample
SAMPPERIOD	- sample period in units of 100ns
SAMPKIND	- sample kind

For instance, the following script will load an alien audio file with a header of 64 bytes, 16 bit precision, and 16 kHz sampling frequency.

```
setenv HDSIZE 64
setenv SAMPSIZE 2
setenv SAMPPERIOD 625
setenv SAMPKIND ALIEN
```

6.2 Assess

6.2.1 Function

This tool reads in a set of HTK compatible label files as outputted by the tools **BroPhon**, **SeqQuant**, or **Viterbi** and compares them with a dynamic programming algorithm with corresponding reference label files on possible substitutions, deletions, or insertions. This is done solely for verification on labelling compatibility. In other words, one must be assured that a pair of label files represent exactly the same transcription. Within the dynamic programming framework, boundaries between two labels are compared with their correspondents in the reference files and statistics are drawn. The basic output is the segmentation accuracy statistics of all label[1] boundaries for the whole file set:

```
--------------------- Segmentation Results ------------------------
CORRECT MARGIN: [ 0 ms -   0 ms ]  %Correct=0.27  [N=6412,C=17]
CORRECT MARGIN: [ 0 ms -  10 ms ]  %Correct=62.57  [N=6412,C=4012]
CORRECT MARGIN: [ 0 ms -  20 ms ]  %Correct=86.96  [N=6412,C=5576]
CORRECT MARGIN: [ 0 ms -  30 ms ]  %Correct=93.75  [N=6412,C=6011]
CORRECT MARGIN: [ 0 ms -  40 ms ]  %Correct=96.30  [N=6412,C=6175]
CORRECT MARGIN: [ 0 ms -  50 ms ]  %Correct=97.52  [N=6412,C=6253]
CORRECT MARGIN: [ 0 ms -  60 ms ]  %Correct=98.07  [N=6412,C=6288]
CORRECT MARGIN: [ 0 ms -  70 ms ]  %Correct=98.61  [N=6412,C=6323]
CORRECT MARGIN: [ 0 ms -  80 ms ]  %Correct=98.88  [N=6412,C=6340]
CORRECT MARGIN: [ 0 ms -  90 ms ]  %Correct=99.17  [N=6412,C=6359]
CORRECT MARGIN: [ 0 ms - 100 ms ]  %Correct=99.42  [N=6412,C=6375]
-------------------------------------------------------------------
```

Each line gives the percentage of boundaries that fall within a correct margin round their corresponding reference boundaries and that may be considered correct. `N` is the total number of boundaries in the defining label files, `C` is the number of boundaries that fall with the correct margin. Each line represents an extension of the correct margin by a multiple of 10 ms. The resulting cumulative statistics reveal the overall performance by considering looser inaccuracies.

Optional extra outputs from **Assess** are

- Inspection of a single label with a pre-determined correct margin (see `-i` option below).
- Creation of tables that depict each label-to-label boundary in particular (see `-t` option below).
- LaTeX-formatted text files (see `-x` switch below).

For comparison purposes, it is also possible to assign two labels to the same equivalence class (see `-e` option below). Also, label files containing (context-dependent) triphone labels of the HTK defined form `A-B+C` can be optionally stripped down to just the label name `B` via the `-s` switch.

6.2.2 Use

Assess is invoked via the command line (or from a UNIX shell script)

[1]The term 'label' refers to each possible modelling entity such as phoneme, syllable, word.

```
Assess [options] labelList labelFiles ...
```

This causes **Assess** to be applied to each `labelFile` in turn. The file `labelList` contains all label names per line for which statistical results information is required. For each `labelFile`, a reference file with the same name but with the extension `lab` (see also option `-X`) is read in and matched with it. The available options are

`-a` By default, the first and last boundaries representing respectively the beginning and ending of the sampled speech data file, in general, are left out of the statistics. Setting `-a` also takes these boundaries into account.

`-m` Lists the mean difference and standard deviation (in ms) between the test and reference boundaries. By default, both the left and the right boundary of each label is taken into account.

`-l` Take only left boundaries into account when switch `-m` is on.

`-r` Take only right boundaries into account when switch `-m` is on.

`-i t f` Lists all label files in which the boundaries associated with label `t` have a deviation of `f` ms equal to or greater than their corresponding references.

`-e s t` The label `t` is made equivalent to the label `s`. More precisely, `t` is assigned to an equivalence class of which `s` is the identifying member. The null label `???` is defined so that making any label equivalent to the null label means that it will be ignored in the comparison process. Note that the order of equivalence labels is important, to ensure that label `X` is ignored, the command line option `-e ??? X` would be used.

`-s` Causes all label names with the form `A-B+C` to be stripped to `B`.

`-g i` Set the number of margins around the reference boundary that must be investigated to `i`. These margins are extended by some multiple and thus, show cumulative statistics of boundary discrepancies (default 10)

`-d f` Set the multiple to `f` in ms that must be maintained by extending the correct margin (default 10.0 ms).

`-t f` Evaluates each label-to-label boundary separately by considering a margin of `f` ms. It gives a detailed overview by a table in which vertically the label to the left is indicated, and horizontally the label to the right of the boundary. In each entry of the table the score p/q is given meaning p correct out of q boundary occurences. In the bottom row and the most right columns, summations over respectively the column and the row are given. In the bottom-right entry, the total segmentation performance is given.

`-x` Print the statistics in LaTeX-format.

`-G fmt` Set the *label* file format to `fmt`. (Default `HTK`).

`-L dir` Set the directory to search for the reference label files to `dir`. If this option is not specified, **Assess** tries to find the files in the same directory as the label files to be evaluated.

`-X ext` Set the extension of the reference label file to `ext` (default: `lab`).

6.2.3 Implementation

The reference and test label files are read in and stored in the datastructures `ref` and `test`. Some possible modifications to the labels are applied according to the command line arguments. For instance, equivalent labels are stored in a linked list that is used to convert any equivalenced label in the transcription before they are further process (see `ConvertEq`).

First, the transcriptions are compared by means of dynamic programming. The principle data structure is the matrix `grid`. Each entry in the grid is a `Cell` containing integer fields for the total match score, and the number of insertions, deletions, and hits along the path leading to that grid cell. While tracing back the grid, hits between a reference label and a test label (notified by a diagonal move along the grid) are used for gathering the statistics regarding the alignments of boundaries associated with these labels. Test boundaries are compared considering a range of intervals around the reference boundary by the routine `CompareBound`. These results are stored in a linked list `scoreList` where each node in this list captures the results considering specific interval.

When required, some matrices (`occBounds`, `corrBounds`) are constructed considering each label-label transition separately. They respectively represent the number of occurrences of each label-label transition and the number of test boundaries that fall within a margin around the reference boundary (correct). These tables are constructed and printed out at the end of the DP match. (see routine `OutTable`).

In addition, all output routines have their LaTeX-version.

6.3 BroPhon

6.3.1 Function

This tools tries to segment a given sampled speech waveform into a set of non-overlapping broad phonetic class segments according to a transcription. The transcription is provided by a text file as described in Section 6.1.1. The segmentation is carried out by iteratively calculating optimal segment boundaries by means of a level-building dynamic programming, followed by an updating of the centroids representing the new acquired segments (see also option `-i` below). At this moment, the number of supported broad phonetic classes is rather small (`UNV`: unvoiced, `VOI`:voiced, `SIL`:silence).

Two extra information sources are needed for this tool to accomplish its task: a mapping from label name to broad phonetic class, and some global duration statistics of each label. This data must be provided by means of a text file as described in Section 6.1.2.

One can choose any combination of measurements to discriminate voiced from unvoiced regions from a pre-defined set: low-high frequency band pass filtered energy, zero crossing metrics, and first lpc coefficient (see `-m` option below).

Optional extra output is the provision for a gnuplot compatible files that depicts graphically the result of the segmentation (see `-G` option below).

6.3.2 Use

BroPhon is invoked via the command line (or from a UNIX-shell script)

```
BroPhon [options] featFile speechFile
```

This causes the given sampled speech data file **`speechFile`** to be subjected to a broad phonetic class segmentation. The file **`featFile`** contains information about the mapping from label to broad phonetic class and some global duration constraints allowed for each label (see Section 6.1.2). The output is written in the form of HTK label files (see Section 6.1.1) whose path name is determined from the input speech file name and the `-S` and `-E` option described below.

The detailed operation of **BroPhon** is controlled by the following command line options. Many settings have to be optimized by a trial-and-error procedure.

`-i i` Enable a method that iteratively determines a set of boundaries and a set of centroids that represent each broad phonetic class until some convergence criterium between two successive iterations is met. `i` states the maximum number of iterations. By default, a single iteration is conducted by using centroids that are acquired from a manually segmented database.

`-m flags` By default, all measurements that discriminate between the three broad phonetic classes are used. This option causes just the measurements indicated by the **`flags`** argument to be used. This argument is a string containing one or more of the letters `f` (low-pass/high-pass energy), `z` (zero crossings), or `r` (first coefficient of a first order LPC analysis). The presence of a letter enables the calculation of the corresponding measurement (default **`fzr`**).

`-v f` Set the onset frequency of the low-pass filter to `f`. (Default: 50.0 Hz).

-V f Set the cut-off frequency of the low-pass filter to f. (Default: 1200.0 Hz)

-u f Set the onset frequency of the high-pass filter to f. (Default: 2000.0 Hz)

-U f Set the cut-off frequency of the high-pass filter to f. (Default: 4000.0 Hz)

-s f Set the energy scaling factor to f in order to improve silence detection. (Default: 500.0)

-W Enable a weighted Euclidian distance metric (Mahalanobis distance) in the segmentation process. (Default: Euclidean distance)

-G Enable production of gnuplot-compatible printouts.

-f T Set the frame period to T ms (default value 10.0 ms).

-k f Set the pre-emphasis coefficient to f (default value 0.95).

-w T Set the frame (window) duration to T ms (default value 20.0 ms).

-L dir Set the directory in which the label files are stored to dir. These label files represent the phonetic transcription along which the segmentation is conducted. (default: current directory).

-X ext Set the label input file name extension to ext (default lab).

-S dir Set the directory to store the output label files to dir (default: current directory).

-E ext Set the output label file name extension to ext (default bpc).

-T N Set the trace level to N (default: tracing disabled). One level of trace output is provided, consisting of the partial distances at each level and the trace back matrix (N=1).

-F fmt Set the audio file format to fmt for the input sampled speech data file (default AIFF). See Section 6.1.3.

6.3.3 Implementation

The tool **BroPhon** uses level-building for accomplishing its task. Level-building is a dynamic programming technique that tries to find a pre-determined number of non-overlapping segments by minimizing an overall distortion (see Section 4.3.1). Routine LevelBuild is a direct implementation of this algorithm.

The pre-determined sequence of broad phonetic class label sequence is obtained by translating the phonetic transcription into its corresponding broad phonetic class transcription. The internal represention of a transcription is the same as for other tools. See Section 6.11.3 for more details about this.

In C-like pseudo code, the implementation looks like

```
/* t : phonetic transcription */
/* b : broad phonetic class transcription */
b = MakePhonFeatList(t);      /* Init centroids, translate transcription */
PreProcess(&src,&tgt);        /* Convert waveform */
do {                          /* do until convergence criterium is met */
  it++;
```

```
    prevDist = LevelBuild(&tgt,b);  /* Update boundaries */
    UpdateCentroids(&tgt,b);        /* Update centroids with new bounds */
    newDist = CalcOverallDist(&tgt,b);  /* Calc. new total distance */
    score = (prevDist - newDist) / newDist;
  } while ((score >= epsilon) && (it < maxIter));
```

Each broad phonetic class is represented by a centroid that is calculated from the utterance at hand. The initial set of centroids represents some ideal broad phonetic classes (binary vectors).

The determination of the initial set of centroids, the incorporation of the duration constraints, and the mapping to broad phonetic class transcription are conducted by the routine **MakePhonFeatList**. The next step is converting the sampled speech waveform **src** into an observation sequence **tgt** according to the command line arguments by the routine **PreProcess**. A set of signal measurements are computed frame-by-frame composing the acoustic vector. This is implemented rather straightforwardly.

In principle, **BroPhon** consists in an iterative procedure where each iteration has two steps. The first step seeks an optimal set of boundaries given a set of centroids by means of level-building (see routine **LevelBuild**). The second step updates the centroid set by using the new acquired boundaries (see routine **UpdateCentroids**). Both steps come up with a minimal total distance that is exploited to guarantee convergence. The convergence criterium is implemented by seeing whether the relative distortion decrease **score** is smaller than a treshold **epsilon** or a pre-determined maximum number of iterations is reached. A detailed overview of the algorithm is given in Section 4.2.1.

In addition, a non-iterative procedure is implemented that conducts only a single level-building pass for acquiring boundaries by using some externally provided (but hard-coded) centroids. These centroids are obtained from a manually segmented speech database recorded from a single male speaker.

6.4 FrontEnd

6.4.1 Function

This tool converts a sampled speech waveform into a parametrized form, i.e. an observation sequence. The basic action is to divide the input file into a sequence of (possibly) overlapping frames. Each frame is converted by means of a filterbank analysis and outputted to the designated target file. The filterbank is FFT-based and essentially non-uniform in the sense that the triangular filters are spaced and have a bandwidth as specified by an Equivalent Rectangular Bandwidth warping function (ERB). Additional options exists for appending normalized log energy (see `-e` option below), delta and delta-square features (see `-d` and `-D` switches below), and a frame acquisition synchronized by pitch instants (see `-G` option below).

The output of the parameterized speech is always written in pre-defined HTK audio file format.

6.4.2 Use

FrontEnd is invoked via the command line (or from a UNIX-shell script)

```
FrontEnd [options] inFile outFile
```

Its effect is to read in the given `inFile`, parameterize it according to the supplied options, and finally output it to `outFile`. The parameterization is to output FFT-filterbank channels along a ERB-scale. The options are

`-G` Enable frame acquisition via a pitch synchronous way. (default: equidistant frame acquisition).

`-P f` Set average pitch period to `f` ms. This value is needed to guide the pitch marker detection (default value 8.0 ms). It is only effective when `-G` switch is set.

`-L N` Set length of data matrix to `N` for calculating 'running' Frobenius norm (default value 20). It is only effective when `-G` switch is set.

`-O N` Set order of data matrix (assumed LPC model) for calculating 'running' Frobenius norm (default value 10). It is only effective when `-G` switch is set.

`-d` Append delta features to observation vector. These delta features are estimations of the first time-derivatives of vectors spanning several frames.

`-D` Append delta square features to observatin vector. These delta square features are estimations of the second time-derivatives of vectors spanning several frames.

`-e` Append the normalized log energy to the observation vector.

`-f T` Set the frame period to `T` ms for equidistant frame acquisition (default value 10.0 ms).

`-k f` Set the pre-emphasis coefficient to `f` (default value 0.95).

`-w T` Set the frame (window) duration to `T` ms for equidistant frame acquisition (default value 20.0 ms).

-m Apply the filterbank analysis on the magnitude spectrum rather than the power spectrum (default power spectrum).

-h Apply a Hamming window.

-n N Set number of filterbank channels to be outputted to N. (default value 16).

-q N Set the interpolation window to estimate the delta and delta square features to the frame range ± N (default value 2)

-T N Several levels of trace output are provided. A trace value of 1 gives basic progress information in the form of a dot being output for each processed speech frame. A trace value of 3 gives maximum tracing.

-F fmt Specifies the audio file format **fmt** for the input sampled speech data file (default AIFF). See also Section 6.1.3.

6.4.3 Implementation

The filters are triangular and they are equally spaced out along the ERB-scale. This is implemented by computing a FFT of the sampled speech waveform. By default, the first half of the FFT components are used to filter in the power domain. An option exists to filter in the spectral magnitude domain. The gathering of FFT components that fall within a triangular filter is implemented as follows: First, all centre frequencies of the filters are determined by equally spacing along the warped frequency axis using the ERB-scale function (see also Equation 4.34),

$$ERB(k) = 21.4 \log_{10}(1 + (k-1)f_s/(N * 229))$$

in which k denotes the index of the FFT component, N is the number of samples in the analysis FFT window and f_s is the sample frequency. Each warped FFT component index is added to a lower bin representing the filter. Then the bins are weighted to form the filterbank channels. All channels along with the log short-time energy are stored in the acoustic vector. This analysis is done for all speech frames at once to construct an observation sequence with instantaneous features. The log short-time energy is normalized by the maximum value found in the observation sequence. The last step is the addition of delta and/or delta-square features. This is directly implemented using the Equations 4.36 and 4.37.

6.5 GenMarkov

6.5.1 Use

This tool generates a so-called prototype of an HMM. The output HMM definition file is generated according to the syntax construction rules (see pp. 21-28 [Young 93]). It is primarily meant to specify the topology and structure of an HMM in a simple way. Means of emission densities are set to zero, variances are set to a positive value such as 1.0, and the sum of all mixture weights within a data stream amounts to 1.0. The transition matrix of the prototype specifies both the allowed transitions and their initial probabilities.

Subsequently, the generated file can be used as input for training tools, such as **SegK-Means**. Only linear, left-to-right HMMs with no skips and a self-loop at each state can be specified. No parameter tying within in a model can be specified. For more elaborate HMMs, the generated file must be subjected to the HTK tool **HHEd** (see pp.72-81 [Young 93]).

6.5.2 Use

GenMarkov is invoked via the command line (or from a UNIX-shell script)

```
GenMarkov [options] outFile
```

Its effect is to generate an HMM definition written to `outFile` according to the given options. The options are

`-s i` Set the number of emitting states to i (Default 3).

`-d i j ...` Set number of datastreams at each state to `i`. Subsequently, each size of a datastream must be specified. This is done by the sequence of integers denoted by `j` .... For instance, the option `-d 4 16 16 16 3` sets the number of streams to 4, where the first stream is of size 16, the second stream of size 16, the third stream of size 16 and the last stream of size 3 (Default, one stream of size 16).

`-m i` Set the number of mixtures for each stream to `i` (Default 1).

`-p s` Set the preprocessing type to the string `s`. Several string expressions correspond to distinct front ends (default `FBANK`).

- `FBANK` ERB-frequency filter bank analysis.
- `MELSPEC` linear mel-frequency filter bank analysis.
- `LPCEPTRA` Linear predictive cepstral coefficients.
- `IREFC` Reflection coefficients in (only) 16 bit precision.
- `MFCC` Mel-frequency cepstral coefficients.
- `LPREFC` Reflection coefficients.

`-f` Enable a full covariance matrix for each mixture (default diagonal matrix).

6.5.3 Implementation

The implementation is straightforward, mainly consisting of output via `fprintf()` calls.

6.6 htk2ipo

6.6.1 Function

htk2ipo converts the file format concerning audio and label information as specified within HTK to the file format as standardized at IPO. In addition, some other audio (e.g. NeXt/Sun) and label formats (e.g. TIMIT) can be converted.

6.6.2 Use

htk2ipo is invoked via the command line (or via a UNIX shell script)

```
htk2ipo [options] speechFiles ...
```

causing the `speechFiles` to be converted according to the command line arguments. The options are

`-L dir` Set the input label file directory to `dir` (default current directory).

`-X ext` Set the input label file extension to `ext` (default `lab`).

`-O dir` Set the output file directory to `dir` (default current directory).

`-E ext` Set the output file extension to `ext` (default `ipo`).

`-F fmt` Specifies the audio file format `fmt` for the input sampled speech data file (default `AIFF`). See also Section 6.1.3.

`-G fmt` Set the *label* file format to `fmt` (default `HTK`).

6.6.3 Implementation

The support of file formats as standardized within IPO is implemented by using the IPO / OTS speech software library [Veenker 94].

6.7 ipo2htk

ipo2htk converts the file format concerning audio and label information as standardized at IPO to the file formats as specified within HTK. Besides the HTK audio format, AIFF audio file format can be outputted.

6.7.1 Use

ipo2htk is invoked via the command line (or via a UNIX shell script)

```
ipo2htk [options] speechFiles ...
```

causing the `speechFiles` to be converted according to the command line arguments. The options are

-L dir Set the output label file directory to **dir** (default current directory).

-X ext Set the output label file extension to **ext** (default **lab**).

-O dir Set the output audio file directory to **dir** (default current directory).

-E ext Set the output audio file extension to **ext** (default **ipo**).

-F fmt Specifies the audio file format **fmt** for the output sampled speech data file (default `AIFF`). See also Section 6.1.3.

6.7.2 Implementation

The support of file formats as standardized within IPO is implemented by using the IPO / OTS speech software library [Veenker 94].

6.8 Implicit

6.8.1 Function

This tool tries to find a set of non-overlapping locally stable spectral segments by means of a correlation metric between LPC spectrum observation vectors in a given sampled speech data file. This technique was originally proposed by van Hemert [Hemert 87, Hemert 85]. It does its work blind-folded with no usage of transcribed data. This may produce insertion or deletion of segments that do not correspond to an ideal transcription of the utterance. However, this technique can be exploited to 'polish' a given segmentation of the sampled speech data (see `-P` option below).

Optional extra output is the provision for a gnuplot compatible files that depict graphically the result of the segmentation (see `-G` option below).

6.8.2 Use

Implicit is invoked via the command line (or via a UNIX shell script)

```
Implicit [options] speechFile
```

This causes the speech file **`speechFile`** to be subjected to a conversion to a sequence of LPC spectra according to the supplied option parameters, subsequently a correlation metric is computed on this sequence. At the ends of proposed segments where the correlation is low with respect to the central part of that segment, it is likely that a segment transition takes place. The option are

- `-P` Adjust ('polish') the boundaries as given by the label files (see `-L` option) by this method. If this switch is not set, the default action is the acquisition of all segment transitions as calculated by this method regardless of a transcription.
- `-n fn` Specify the boundaries associated with the labels that must not be taken into account in the 'polishing' process by means of the text file `fn`. Each label is specified per line in the text file `fn`.
- `-q f` 'Polishing' may only occur if the input boundary lies in an interval of `f` ms around the boundary as proposed by this method (default 20.0 ms).
- `-f f` Set the frame period to `f` ms (default 10.0 ms).
- `-k f` Set the pre-emphasis coefficient to `f` (default 0.95).
- `-p N` Set the LPC-order to `N` (default 12).
- `-w f` Set the frame (window) duration to `f` ms (default 20.0 ms).
- `-G` Enable gnuplot-compatible printout of the result.
- `-L dir` Set the input label file directory to `dir` (default current directory).
- `-X ext` Set the input label file extension to `ext` (default `lab`).

-S dir Set the output ('polished') label file directory to **dir** (default current directory).

-E ext Set the output ('polished') label file extension to **ext** (default **imp**).

-F fmt Specifies the audio file format **fmt** for the input sampled speech data file (default **AIFF**). See also Section 6.1.3.

6.8.3 Implementation

It is a direct implementation of the proposed method of van Hemert [Hemert 85, Hemert 87].

The sampled speech waveform is converted frame-by-frame into an observation sequence of LPC spectra. These spectra are obtained by evaluating the expression using 128 frequency points

$$S(f) = \frac{\sigma^2 \Delta t}{|1 + \sum_{k=1}^{p} a_k e^{-j2\pi f k \Delta t}|^2}$$

where σ^2 is the residual energy, a_k, $k = 1, \cdots, p$ the LPC coefficients and Δt is the time interval between two adjacent samples. The LPC coefficients are calcuted by using the standard autocorrelation method (Levinson-Durbin recursion). The amplitude of the spectra are expressed in dB.

The determination of transition from one spectral state to another spectral state (boundary) closely resembles the article of van Hemert. 'Polishing' the given boundaries is implemented by comparing them with the spectral state transitions of this method (see routine **PolishSegments**). A boundary is equalized to a spectral state transition if it is the closest boundary to that transition and falls within a margin (of typically 20.0 ms) around it.

6.9 Pitch

6.9.1 Function

This tool tries to detect glottal closure instants (epochs) or pitch markers in a speech waveform by means of calculating a 'running' Frobenius norm over signal matrices obtained by advancing a rectangular window over the waveform one sample further successively. Local maxima of the numerical value of the Frobenius norm correlates with the glottal closure instants. This technique was originally proposed by Ma [Ma 94].

The pitch markers are written in the HTK label file format (see Section 6.1.1) in which the onset of each label (first column) specifies the actual calculated glottal closure instant in units of 100 ns.

Optional extra output is the provision for a gnuplot compatible files that depict graphically the waveform and its 'running' Frobenius norm (see `-G` option below).

6.9.2 Use

Pitch is invoked via the command line (or from a UNIX shell script)

```
Pitch [options] speechFile outFile
```

This causes the `speechFile` to be subjected to an epoch detection and the results to be written to `outFile`. The options are

`-k f` Set the pre-emphasis coefficient to `f` (default 0.95).

`-r` Apply the method to the LPC-residue. By default pre-emphasized speech is used, resulting into more reliable detections.

`-P f` Set the assumed average pitch period to `f` ms. This value is needed to find the position of the local maxima of the Frobenius norm. These maxima are assumed to lie in a region corresponding to this period. If no maxima are found (i.e. unvoiced speech) a pitch marker is positioned corresponding to this period. (default 8.0 ms)

`L N` Set the length of the signal matrix to `N` (default 20).

`-O N` Set the order of the signal matrix to `N` (default 10).

`-E fn` The 'running' Frobenius norm is written to file `fn`. Its format is HTK audio format.

`-F fmt` Specifies the audio file format `fmt` for the input sampled speech data file (default `AIFF`). See also Section 6.1.3.

`-G` Enable production of gnuplot-compatible printouts.

6.9.3 Implementation

An efficient computation is implemented for calculating a 'running' Frobenius norm. We present this implementation by means of an operational example. Consider $p = 3$ and $l = 6$ and the signal matrices at two successive time points $t = 4$ and $t = 5$.

$$\begin{bmatrix} x(1) & x(2) & x(3) & x(4) \\ x(2) & x(3) & x(4) & x(5) \\ x(3) & x(4) & x(5) & x(6) \\ x(4) & x(5) & x(6) & x(7) \\ x(5) & x(6) & x(7) & x(8) \\ x(6) & x(7) & x(8) & x(9) \end{bmatrix} \begin{bmatrix} x(2) & x(3) & x(4) & x(5) \\ x(3) & x(4) & x(5) & x(6) \\ x(4) & x(5) & x(6) & x(7) \\ x(5) & x(6) & x(7) & x(8) \\ x(6) & x(7) & x(8) & x(9) \\ x(7) & x(8) & x(9) & x(10) \end{bmatrix}$$

The Frobenius norm F for the signal matrix at time point $t = 4$ can be *directly* calculated by implementing Equation 4.26

$$F = x(1)^2 + 2x(2)^2 + 3x(3)^2 + 4x(4)^2 + 4x(5)^2 + 4x(6)^2 + 3x(7)^2 + 2x(8)^2 + x(9)$$

However, this requires many additions and multiplies for each time point. An efficient method is by considering the sum of the squared entries for each row r_i in the signal matrix. For the signal matrix at time point $t = 4$, we obtain

$$\begin{aligned} r_1 &= x(1)^2 + x(2)^2 + x(3)^2 + x(4)^2 \\ r_2 &= x(2)^2 + x(3)^2 + x(4)^2 + x(5)^2 \\ &\vdots \\ r_6 &= x(6)^2 + x(7)^2 + x(8)^2 + x(9)^2 \end{aligned}$$

Remark, the Frobenius norm can now be calculated by the sum of all r_i's. ($F = \sum r_i$). This calculation of all r_i's has only to be carried out for the first 'running' Frobenius norm calculation (in this case, for the signal matrix at time point $t = 4$).

Advancing the Frobenius norm calculation one time point ($t = 5$) further, only requires a 'scrolling' assignment for all r_i's, i.e.

$$r_{i-1} = r_i \quad , i = 1, \cdots, 5$$

and an updating of the last summation r_6 due to the departure of the sample $x(6)$ and the arrival of the sample $x(10)$ in the row of the updated signal matrix.

Having computed all numerical values of the 'running' Frobenius norm, it is now looking for local maxima. This is implemented by differencing the Frobenius trajectory and detecting the (largest) positive zero-crossings in some interval around an average pitch period.

6.10 SegKMeans

6.10.1 Function

SegKMeans is used to provide initial estimates for the means, variances, mixture weights, and transition probabilities of a single HMM using a set of observation sequences. It is a small extension to the tool **HInit** as present in the HTK toolkit (see page 82-85 of [Young 93]). Mainly, it is extended with a possibility to use pitch synchronously spaced training vectors and the updating of transition probabilities. It works by repeatedly using Viterbi alignment to segment the training observations and then recomputing the means and variances by pooling the vectors in each segment. In the multiple mixture case, the vector pools are clustered using a modified K-Means algorithm. Transition probabilities are estimated by counting the number of visits at the states during the Viterbi aligment. In the absence of an initial model, the process is started by performing a linear segmentation of each training observation.

It normally takes as input a prototype HMM definition which defines the required HMM topology, i.e. it has the form of the required HMM topology and structure except that means, variances and mixture weights are ignored. Means should be set to zero, variances should be set to a positive value such as 1.0, and the sum of all mixture weights within a data stream should amounts to 1.0. The transition matrix of the prototype specifies both the allowed transitions and their initial probabilities. Transitions which are assigned zero probability will remain zero and hence denote not-allowed transitions. Observation sequences can consist of continuously spoken training material. By giving each *partial* observation sequence a segment label corresponding to the HMM to be initialized, **SegKMeans** will simply cut it out of the training data automatically.

SegKMeans supports multiple streams, multiple mixture, parameter tying within a single linear model and full or diagonal covariance matrices.

6.10.2 Use

SegKMeans is invoked via the command line (or from a UNIX shell script)

```
SegKMeans [options] hmmFile trainFiles ...
```

This causes the parameters of the given `hmmFile` to be estimated repeatedly using the data in `trainFiles` until either a maximum iteration limit is reached or the estimation converges.

The detailed operation of **SegKMeans** is controlled by the following command line options

`-b` Take only one-third of the segment for bootstrapping the HMM, if possible.

`-c N` Set the maximum number of cluster iterations to `N` (default 16).

`-e f` Set the convergence factor to the real value `f`. The convergence factor is the relative change between successive values of the Viterbi alignment score (default value 0.0001)

`-i N` Set the maximum number of estimation cycles to `N` (default 20).

-l s Set the string **s** to be the name of the segment label. When this option is used, **SegKMeans** searches through all of the training files and cuts out all segments with the given label. When this option is not used, **SegKMeans** assumes that each training file is a single token.

-m N Set the minimum number of training examples, so that if fewer than **N** examples are supplied, an error is reported (default value 3).

-n Suppress the initial linear segmentation but use Viterbi alignment instead. This is useful for updating the parameters of an existing model.

-o fn Write the new HMM definition file to **fn**. By default, the original file will be overwritten.

-p Use vectors that are spaced pitch synchronously. It causes to load a pitch file, preferably generated by **Pitch**.

-u flags By default, all HMM parameters are updated, that is, means, variances, mixture weights, and transition probabilities. This option causes just the parameters indicated by the **flags** argument to be updated. This argument is a string containing one or more of the letters **m** (means), **v** (variance), **w** (mixture weight), and **t** (transitions). The presence of a letter enables the updating of the corresponding parameter set. (default **mvwt**).

-v f Set the minimum variance to the real value **f** (default value 0.01).

-M mf Force a macro file to be loaded regardless of whether or not the actual HMM definition contains any clauses for using a macro. Macros are required when using parameter tying.

-G fmt Set the *label* file format to **fmt** (default **HTK**).

-L dir Specify the directory to search for the label files. If this option is not used it is expected that the label files are in the same directory as the training files.

-P dir Specify the directory to search for the pitch files. If this option is not used it is expected that the pitch files are in the same directory as the training files.

-X ext Set the extension of the label files to **ext** (default **lab**).

-Y ext Set the extension of the pitch files to **ext** (default **pit**).

-T flags Set the trace level to **N**. Various levels of trace output are provided. The meaning of the **flags** argument is as follows, where an octal base is usd

0001	basic progress reporting
0002	information of training files
0004	display each Viterbi alignment
0010	trace the Viterbi decoding operationally
0020	display each updated HMM
0040	display cluster
0100	print out cost of each cluster at each iteration
0200	show cluster centre

6.10.3 Implementation

Details about the implementations can be found in the HTK manual (see page 84-86 in [Young 93]). We extend the source code with possibilities to update transition probabilities. Required statistics for this are obtained at each Viterbi decoding alignment. Also, the possibility to use training vectors that are spaced pitch synchronously is implemented.

6.11 SeqQuant

6.11.1 Function

This tool tries to segment a given sample speech data file into consecutive quasi-stationary elements according to a transcription subjected to the constraint that all vectors in a cluster (segment) are contiguous in time. This is accomplished by means of a level-building dynamic programming framework. The transcription must be provided by a file as described in Section 6.1.1. Although not strictly necessary but recommended for performance reasons, the accuracy can be improved by providing extra *a priori* knowledge to the process by means of a file as described in Section 6.1.2. Also the integration of broad phonetic class segmentation is made possible that further improves the accuracy (see `-B` and `-Y` options below).

Optional extra outputs from **SeqQuant** are

- Selection of two distortion measures: Itakura distortion and cepstral distance (see option `-c`, `-d`, and `-W` below).
- Creation of a simple linear segmentation (see `-l` switch below).
- Provision for gnuplot compatible files that depict graphically the result of the segmentation (see `-G` option below).
- Acquisition of frame by synchronizing it with pitch instants (see `-s` option below).

6.11.2 Use

SeqQuant is invoked via the command line (or from a UNIX-shell script)

```
SeqQuant [options] speechFile
```

This causes the given sampled speech data file **speechFile** to be subjected to a Sequence Constrained Vector Quantization along a given transcription. The output is written in the form of HTK label files (see Section 6.1.1) whose path name is determined from the input speech file name and the `-S` and `-E` options described below.

The detailed operation of **SeqQuant** is controlled by the following command line options. Many settings have to be optimized by a trial-and-error procedure.

`-a` Enable segmentation boundaries outputted at frame centres. Default boundaries are placed at frame onsets.

`-i` Pool the *partial* observation sequences of corresponding labels in the transcription during the dynamic programming process. During the process of level-building, all separate *partial* observation sequences that correspond to the *partial* minimal distortion are collected. From these collected sequence (representing several realization of the same label), a single centroid is computed. By default, each label occurrence in the transcription has attached its own private centroid.

`-l` Apply a simple linear segmentation.

`-D fn` Provide a file `fn` containing the mapping from label to broad phonetic class, some global duration constraints allowed for each label (see Section 6.1.2). The provision of this file overrules the options `-m` and `-M`.

`-m f` Set a minimum duration for each label to `f` ms (default value is the number of ms corresponding to a single frame).

`-M f` Set the maximum duration for each label to `f` ms (default value is the total speech file length).

`-b f` Set boundary compliance factor to `f` ms. (default value 20.0 ms). This factor is used for integrating the broad phonetic class segmentation and **SeqQuant** and leaves a small margin for the duration of each label.

`-s` Enable frame acquisition via a pitch synchronous way. (default: equidistant frame acquisition).

`-c` Enable truncated cepstral distance for segmenting the utterance (default: Itakura distortion).

`-d` Use delta features with the cepstral distance.

`-W d` Set the cepstral liftering coefficient to `N`. The default value of 0 disables cepstral liftering.

`-f T` Set the frame period to `f` ms in the case of equidistant frame acquisition (default value 10 ms).

`-k f` Set the pre-emphasis coefficient to `f` (default value 0.95).

`-w T` Set the frame (window) duration to `T` ms in the case of equidistant frame acquisition (default value 10.0 ms).

`-P` Extract centroids (prototypes) from the segmented utterance. The segment information must be given in the label file. A single centroid is computed from all observation sequence corresponding to a label. This option reduces the algorithm into a template matcher.

`-H` Extract centroids (prototypes) from the segmented utterance. The segment information must be given in the label file. Each label occurrence in the transcription has a *unique* centroid computed from the observation sequence. This option reduces the algorithm into a template matcher.

`-o` Extract only one-third of the data required for the centroid computation for the `-P` and `-H` options.

`-L dir` Specifies the directory in which the label files are stored. These label files represent the phonetic transcription along which the segmentation is conducted. (default: current directory).

`-X ext` Set the label input file name extension to `ext` (default `lab`).

`-S dir` Set the directory to store the output label files to `dir` (default: current directory).

`-E ext` Set the output label file name extension to `ext` (default `seg`).

`-B dir` Set the directory that contains the label files as provided by the broad phonetic class segmentation to `dir` and enable the usage of broad phonetic class segments.

-Y ext Set the label input file name extension of the broad phonetic class segmentation to `ext`. (default `bpc`).

-T N Set the trace level to `N` (default: tracing disabled). One level of trace output is provided, consisting of the partial distances at each level and the trace back matrix `N=1`.

-F fmt Set the audio file format to `fmt` for the input sampled speech data file (default `AIFF`). See also Section 6.1.3.

6.11.3 Implementation

The tool **SeqQuant** uses level-building for accomplishing its task. Level-building is a dynamic programming technique that tries to find a pre-determined number of non-overlapping segments by minimizing an overall distortion (see Section 4.3.1). Routine `LevelBuild` is a direct implementation of this algorithm.

Minimum and maximum duration allowed for each label can be adapted to the broad phonetic class segmentation as provided by **BroPhon**. For this, routine `AlignDurations` is a straightforward implemention of Equations 4.14 and 4.15.

For each label occurrence in the phonetic transcription a unique centroid is reserved. The actual computation for a given centroid is implemented by routine `CalcCentroid`. Each centroid is recomputed on-the-fly during the dynamic programming process each time a new intra-segment distortion has to be computed by invoking this routine (see routine `CalcCepDist` and `CalcItakura`).

In addition, centroids can be calculated once by using the time annotation in the label file. Therefore, routine `ConstUniqCentroids` calculates a centroid for each label occurrence separately by *cutting* the label out of the observation sequence as specified in the label file and invoking `CalcCentroid`. On the other hand, routine `ConstPooledCentroids` calculates a centroid for each corresponding label in the transcription. Thus, all *cut* observation sequences for corresponding labels are brought together for calculating a 'pooled' centroid by invoking `CalcCentroid`.

By default, 3 observation sequences are used for calculating the Itakura distortion: the sampled speech waveform is converted into a sequence of autocorrelation sequences (`auCorr`), a sequence of inverse LPC-filters (`acVec`) and a sequence of residual energies (`sigma`). The Itakura distortion is efficiently calculated using Equations 4.18 and 4.19. When using the cepstral distance, a LPC-derived cepstral vector sequence is calculated. When necessary, delta features are incorporated into this cepstral vector according to Equation 4.36.

Internal representation of a transcription

The internal representation of a transcription along which the levelbuilding is conducted as illustrated by Figure 6.1. It consists of several 'transcription' levels and each level is represented by a `LabList` record. Each of these holds a count of the number of labels in that level of the transcription, a pointer to an array of the labels themselves, and a pointer to the next level of the transcription. However, the facility of multiple levels in the transcription is not used. The whole transcription is represented by the type `Transcription` which is just a pointer to the first `LabList` record in the list.

Each array of `Label` records is indexed from 0 and each element holds the label identifier `labid`, the start boundary location, the end boundary location, an optional score, duration

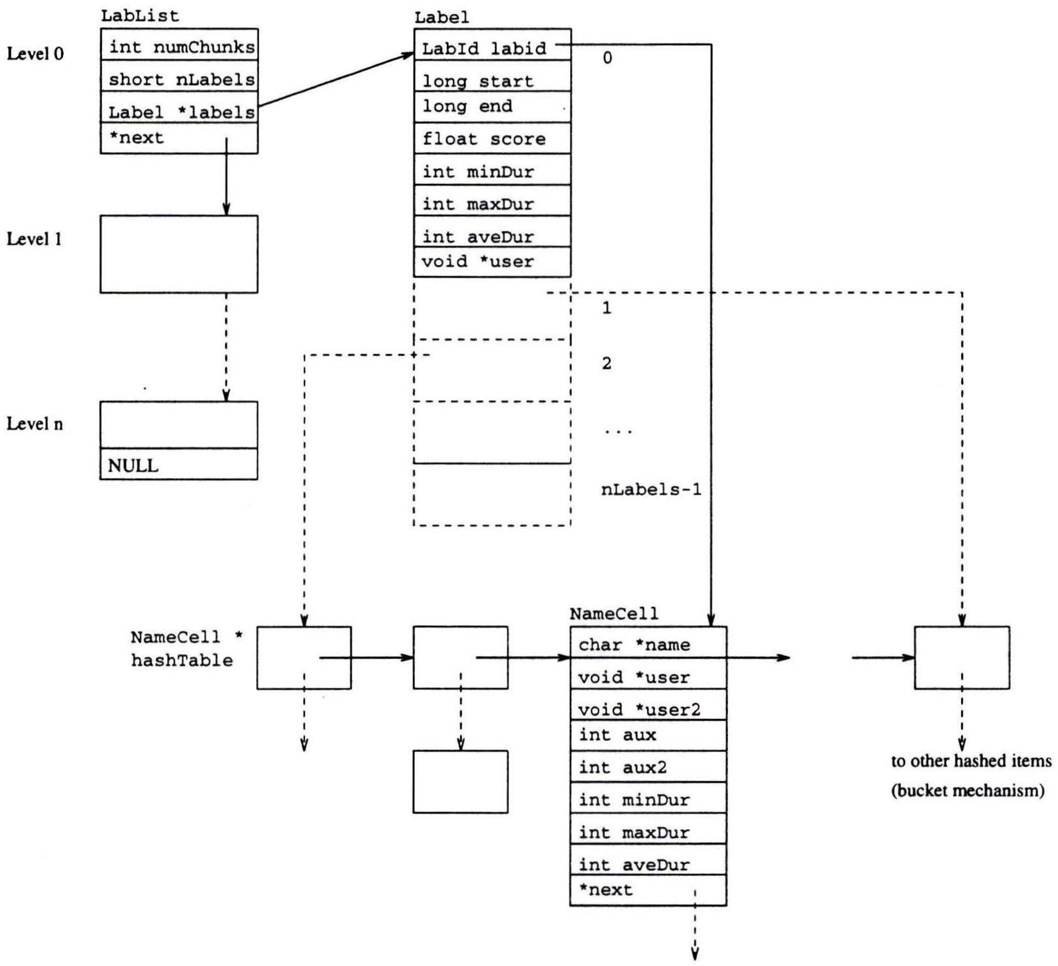

Figure 6.1: *The main datastructures of* **SeqQuant**

statistics (minimum, maximum and average), and an extra **user** pointer for specific usage. The variable **numChunks** is simply for internal memory management of the labels. The start and end locations are defined as absolute times in 100ns units. The **user** pointer is used for holding the centroid vector of that specific label (segment).

The label indentifier **labid** is a pointer to a hashed linked datastructure called **NameCell**. This hash linking is implemented by using the hash function as proposed by Kernighan and Ritchie [Kernigha 88]. Besides its efficiency, this hash linking mechanism also guarantees the maintainance of only one label reference throughout the program.

Nevertheless, more than one occurrence of a label in a transcription can be present; they simply refer to the same **NameCell** record as linked in the hash table. **NameCell** consists of a label name string, two specialized pointer for integration with HTK tools (not shown in Figure 6.1), two pointers **user** and **user2** for specific usage, two auxiliary flags **aux** and **aux2**, and duration statistics (minimum, maximum and average), and a pointer to the next item in the hash list. The **aux**-flags are used for holding the broad phonetic class types of labels. The **user**-pointer are used for holding intermediate or global (external reference) centroids of each label reference during the dynamic programming pass.

As can be seen, there is a similarity between the member variables of **Label** and **NameCell** regarding the centroids and duration statistics. This is done to emphasize the distinction between the duration statistics and centroids for the actual realization of a label (segment) at hand as it is the case for **Label**, and for a more global perspective as it is the case for **NameCell**.

6.12 Viterbi

6.12.1 Function

This tool is a general-purpose Viterbi recognizer with the possibility to generate N-best hypotheses. It is an extended version of the HTK Basic Tool **HVite** (Version V 1.4A) in which a lattice N-best algorithm and a bigram with word categories is implemented. It will match a finite state network of HMMs against one or more parametrized speech files and output one or more segmented transcriptions (output label files) in the format as described in Section 6.1.1.

Before actually invoking this tool, much prior work has to be done. All raw speech files have to be preprocessed in order to obtain parametrized speech files in the form of acoustic vectors (observation sequences). A particular front end has to be chosen for this reason by using the programs **HCode** from HTK[Young 93] or **FrontEnd**. Fully-trained HMMs have to acquired from a selected set of utterances, the training set. This computer time-consuming training process involves a proper design of the HMMs and proper usage of HTK utilities (see e.g. [Young 93],[Pauws 93]). Also for the purpose of speech recognition, a pronunciation dictionary and language model (possibly with bigram probabilities) must be specified by means of a textual specification of a finite state network.

When it comes to solely supervised speech segmentation, the finite state network is simply the sequence of HMMs corresponding to the transcription of the utterance. Also, the amount of preparation work is clearly lesser than it is documented here and Sections 6.12.3 and 6.12.4 can be skipped by a first reading.

6.12.2 Use

Viterbi is invoked via the command line (or from a UNIX-shell script)

```
    Viterbi [options] hmmList netFile speechFiles ...
```

This causes the given parametrized speech files to be matched against the finite state network of HMMs defined by `netFile`. This file contains the textual specification of the network (see Section 6.12.3). The recognizer output is written in the form of HTK label files (see Section 6.1.1) with the time alignment information whose path name is determined from the input speech file name and the `-L` and `-X` option described below. The list of speech files can be stored in a script file if required.

The detailed operation of `Viterbi` is controlled by the following command line options. Many settings have to be optimized by a trial-and-error procedure.

`-N N` Enables a lattice N-best algorithm that maintains the `N` hypotheses with the best scores. (default: simple Viterbi decoding)

`-b fn` Load bigram probabilities from the file `fn`. When the bigram consists of categories the `-w` option must be specified.

`-w` Enable a bigram with word categories.

`-W f` Set the fixed transition log probability *within* categories to `f`. Setting `f` to 0.0 means that word sequences within categories are considered to be a single entity (default 0.0).

-c f Set the tied-mixture pruning threshold to **f**. When all mixtures of all models are tied to create a full tied-mixture or semi-continuous system, the calculation of emission probabilities is treated as a special case. Only those mixture component probabilities which fall within **f** of the maximum mixture component probability are used in calculating the state emission probability (default 20.0).

-d dir This specifies the directory to search for the HMM definition files corresponding to the labels used in the recognition network. The HMM definition files must be specified by means of syntax constructions rules. (see Section 4.3 including 4.5 of [Young 93] pp.21-28). The tool **GenMarkov** generates correct HMM files that subsequently can be trained.

-l Set the transcription labels to start and end on frame centres. This is useful if **Viterbi** is used to provide segment boundaries for later training. By default, **Viterbi** places the start time of the label at the start of the first frame of the segment and the end time at the end of the last frame of the segment.

-n Disable triphone-stripping of bigram names. This is necessary if bigram names contain either + or - symbols but do not correspond to HTK context dependent names (triphones).

-p f Set the fixed transition log probability (word entrance penalty) to **f**. Typical value for word recognition is -15.0 (default 0.0).

-s f Sets the grammar scale factor to real **f**. This factor post-multiplies either bigram log probabilities (or the $-\log(N_{succ}(i))$ factors). Typical value for word recognition with a bigram language model is 10.0 (default 1.0).

-t f Enables beam searching with pruning threshold **f**. Each model is deactivated whose maximum log probability token falls more than **f** below the maximum for all models. Typical values for this threshold are 30 for phone recognition and 150 for word recognition. Setting **f** to 0.0 disables the beam search pruning mechanism (default 0.0).

-x ext Sets the extension to use for HMM definition files to **ext**.

-F fmt Sets the parametrized speech format to **fmt**. Several base kind of acoustic vector types (such as LPC, FILBANK) can be expressed with a number of possible variants indicated by attaching a qualifier to the name of the base kind at the code **fmt**. Four such qualifiers are provided for adding delta features (_D), adding delta-square features (_A), adding energy coefficients (_E), and suppressing absolute energy coefficients (_N). (see Section 2.5 pp.11 and Section 3.1 pp.13-14 of [Young 93]).

-L dir Specifies the directory to store the output label files (default: current directory). The conventions for naming these label files are as follows. The extension of the corresponding parametrized speech file (if any) is removed and the extension for output label files (default: **rec**) is appended. When the N-best -N option is set, more than one label file will be outputted and all label file extensions are preceded by a count representing the hypothesis ordering.

-X ext Sets the output label file name extension to **ext** (default **rec**).

-S fn Enables the script file fn. In this file all test speech files are enumerated.

-T N Sets the trace level to N (default: tracing disabled). Various levels of trace outputs are provided, each of which is enabled by a specific bit of the trace value set N. Many trace flags involves an inner look in the implementation issues of **Viterbi**. The meaning of these flags is as follows, where an octal base is used.

0001 enables basic progress reporting. This consists of a line being printed as each speech file is processed followed by a line giving the average log probability per frame and the maximum number of PLRs (phone link records) generated.

0002 dump the recognition network.

0004 trace garbage collection.

0010 show beam width.

0020 show beam width.

0040 show phone link records.

0100 show the calculation of the emission probabilities.

0200 show the recognition output.

0400 print the network memory statistics.

Example

In order to illustrate the rather elaborated way of recognizing speech using word categories we present the following command line that will invoke **Viterbi** in which all HMMs will be retrieved from the directory hmm. A list of all HMMs is given in the file PhonList. The recognition results are stored in the directory Results and displayed according to the -T option. The beam search pruning treshold is 150.0, 'optimal' for phoneme-based recognition. The fixed transition probabilities is -15.0, 'optimal' for continuous speech recognition applications. The recognition will be applied along the network as specified in the file Network (see Section 6.12.3). Also, a word-categorized bigram as specified in the file Bigram will be used (see Section 6.12.4). The network file as well as the bigram file must be compatible. The option -w indicates that the language model is built up by word categories. In order to balance the bigram probabilities and the HMM probablitities, a grammar scale factor fixed at 10.0 is used. All parametrized speech files in the directory AcVec will be tested. These files contain filterbank analysis vectors with added delta, delta square, and energy features.

```
Viterbi -b Bigram -w -s 10.0 -d hmm -L Results -t 150.0 -p -15.0
        -T 0200 -F FBANK_D_A_E PhonList Network AcVec/*.erb
```

A much more simpler usage of **Viterbi** is demonstrated in Section 6.13.

6.12.3 Network specification

The finite state network consists of nodes that are a slot for an HMM or represent so-called 'pseudo-models' necessary for guiding the recognition process. In particular, three types of nodes exist.

pseudo models Network nodes that have no container for holding an HMM instance, but represent specific network information necessary for guiding the recognition process (e.g. word and category boundaries). These nodes are denoted by the reserved words `WD_BEGIN`, `WD_END`, and `WD_CAT`.

word-internal nodes Network nodes that represent HMM instances and are delimited by `WD_BEGIN` and `WD_END` nodes.

word-external nodes Network nodes that are representants of HMM instances, but are no constituents of a composite word model. These nodes are useful when a whole (isolated) word recognition system must be built. Each word-external node keeps a huge HMM instance representing a whole word model.

Connections between several nodes can be specified by a regular language notation. The whole network can be specified in a bottom-up procedure by firstly specifying single nodes, collecting them into sub-networks, and joining the sub-networks in a single network.

The textual specification of a network is inputted using the following syntax construction rules in which characters delimited by double quotes are considered terminals (see also Section 6.6 of [Young 93]) Each node in the network is represented by a model name, often this will correspond to an HMM. Additionally, each node holds an external name which is used when printing out the identity of a node.

```
name   =  char { char }
model  =  name [ "%" ( "%" | name ) ]
```

Here char represents any character except one of the following meta characters `{ } [ ] < > | = $ ( ) ; / *`.

The first name in a model definition is the internal name. There are three reserved internal model names : `WD_BEGIN`, `WD_END`, and `WD_CAT`. They represent the 'pseudo models' in the network. `WD_BEGIN` and `WD_END` nodes are used to delimit word boundaries for word recognition systems based on sub-word units (e.g. phoneme-based systems). The `WD_CAT` node name is reserved for describing the entrance of a category of word sequences. The second optional name is the external name. If the external name is not given then it is set by default to be the same as the internal name. If a second `%` symbol is written then the external name is set to `NULL` and will not be outputted by **Viterbi** (e.g. for silence words).

Thus, `WD_BEGIN%word` represents a node (pseudo model) with a reserved internal name `WD_BEGIN` and external name `word`. `SI%%` represents a node (HMM) with the internal name `SI` and a `NULL` external name. `AA` represents a node (HMM) in which the internal an external name are considered as the same.

Network definitions may also contain variables

```
variable  =  "$"name
```

Variables are identified by a leading `$` character. They stand for sub-networks and must be defined by an expression before they appear in the right hand side of a rule. This means that recursion *cannot* occur.

```
subnet  =  variable "=" expr ";"
```

An expr consists of a set of alternative sequences representing parallel branches of the network.

```
expr      =  sequence { " | " sequence }
sequence  =  factor { factor }
```

Each sequence is composed of a sequence of factors where a factor is either a model name, a variable representing some sub-network or a *regular expression* contained within various sort of brackets.

```
factor  =  " (" expr " )"      |
           "{" expr "}"        |
           "<" expr ">"        |
           " [" expr "]"       |
           "<<" expr ">>"      |
           model               |
           variable
```

Ordinary parentheses ( ) denote simple factoring of expressions, curly braces { } denote zero or more repetitions of expressions (Kleene star operation) and angle brackets < > denote one or more repetitions of expressions (Kleene plus operation). Square brackets [ ] are used to enclose optional items. The double brackets << >> are a special feature included for building context dependent loops (triphones, see Section 6.6 of [Young 93] pp.48-52).

Finally, the complete finite network state network is defined by a list of sub-network definitions followed by a single expression within parentheses.

```
network  =  { subnet } " (" expr " )"
```

C style comments may be placed anywhere in the text of the network definition.

Network example

The most sophisticated network that can be specified is by formulating a regular language model consisting of word categories. Each category is described by a regular expression of words. Each word is formulated as a regular expression of HMM models (phonemes). This example defines a language in which each sentence is simply a sequence of verbs, determiners, nouns, and preposition delimited by silence (e.g the monkey takes the banana from the table). As it is shown for some transcriptions, words can be expressed by any regular expression of HMM models, representing distinct pronunciations of the same word. Also, word categories can be built up by more elaborated constructions representing whole phrases (e.g. compound nouns, parenthetical remarks, verb phrases). Other networks without word categories can be found in the *HTK manual* [Young 93].

```
$puts    = WD_BEGIN%puts    P ( AH | UW ) T S         WD_END%puts;
$takes   = WD_BEGIN%takes   T EY K S                  WD_END%takes;
$eats    = WD_BEGIN%eats    IY T S                    WD_END%eats;
$the     = WD_BEGIN%the     ( TH | DH ) [ AX ]        WD_END%the;
$a       = WD_BEGIN%a       AX                        WD_END%a;
$man     = WD_BEGIN%man     M AE N                    WD_END%man;
```

```
$monkey  = WD_BEGIN%monkey  M AO NX K IY                     WD_END%monkey;
$banana  = WD_BEGIN%banana  B AX N (AE | AX ) N ( AE | AX )  WD_END%banana;
$car     = WD_BEGIN%car     K AR                             WD_END%car;
$table   = WD_BEGIN%table   T EY B EL                        WD_END%table;
$key     = WD_BEGIN%key     K IY                             WD_END%key;
$from    = WD_BEGIN%from    F R AO M                         WD_END%from;
$on      = WD_BEGIN%on      AO N                             WD_END%on;
$sil     = WD_BEGIN%%       SI                               WD_END%%;

$VERB  = WD_CAT%VERB ( $puts | $takes | $eats );
$DET   = WD_CAT%DET  ( $the | $a );
$PREP  = WD_CAT%PREP ( $from | $on );
$NOUN  = WD_CAT%NOUN ( $man | $monkey | $banana | $car | $table | $key );
$SIL   = WD_CAT%SIL  ( $sil );

( $SIL < $VERB | $DET | $NOUN | $PREP > $SIL )
```

Now, **Viterbi** parses this network specification and builds a finite state recognition network from it. The recognition task is conducted along this network.

6.12.4 Bigram specification

Transitions between nodes in the network can be characterized by a bigram probability. When only word-external and/or composite word models (WD_END/WD_END pairs) are present in the network specification, bigrams can be specified between them. Whenever also categories are expressed in the network specification, only bigram probabilitites between categories (WD_CAT node) are allowed.

The bigram probabilities are specified via a text file using the following format that simply defines a matrix structure

$$\begin{array}{lllll} l_1 & p_{11} & p_{12} & \cdots & p_{1H} \\ l_2 & p_{21} & p_{22} & \cdots & p_{2H} \\ & \cdots & & & \\ l_H & p_{H2} & p_{H2} & \cdots & p_{HH} \end{array}$$

where l_h is the *bigram name* for the node. The bigram name for each composite word-model and word-external node is the external name, or if it is NULL the internal name is used. The quantity p_{ij} denotes the conditional *a priori* probability of model (node) j followed by model i. I(f context-dependent models are being used (e.g. triphones) then, by default, the contexts are ignored when forming the bigram names.

Bigram example

A bigram file that specifies the transition probabilities between 5 distinct word categories determiner, preposition, noun, verb, and silence is as follows

```
DET     0.05    0.05    0.80    0.05    0.05
PREP    0.80    0.05    0.05    0.05    0.05
NOUN    0.05    0.2     0.2     0.35    0.2
VERB    0.4     0.3     0.05    0.05    0.2
SIL     0.80    0.05    0.05    0.05    0.05
```

Obviously, the estimation of these probablities must be done by resorting to a training text. A first impulse is to obtain the bigram probablities by a simple relative frequency approach and setting a floor probabilities for those pairs that are absent in the training text. However, more elaborate approaches exist [Pauws 93].

6.12.5 Implementation

The main data structures used by **Viterbi** are the finite state network `theNet` with its nodes `Node`, the HMM definitions `HMMDefs` and the phone instances. Figure 6.2 shows a graphical representation of the way it is actually implemented in C.

Viterbi is implemented using the Token Passing Paradigm that is applied along a finite state network [Young 93]. The network `theNet` is represented by a set of nodes with explicit pointers to all successor and all predecessor nodes. This is implemented by the arrays `LinkSet` that join nodes with their successor and predecessors. All nodes in the network can be visited one-by-one (in a linear list way) by traversing the pointer structure `chain`. Each node contains a single unused pointer member called `user` which can be used to attach application specific data structures and members for the internal name (`modelName`) and external name (`extName`). The network contains a single entry node called `ENTRY` and a single exit node called `EXIT`.

Once the finite state network has been built, a `PhoneInstance` structure is attached to each network node via the `user` member. The `PhoneInstance` also holds the identifier `piType` for the node type (word-internal, word-external, `WD_BEGIN` etc.) and the bigram index `bigidx` which points to the row or column of the bigram matrix (if any) for the node. This bigram matrix is maintained globally. For all nodes, except `WD_BEGIN/WD_END` and `WD_CAT` nodes, the attached `PhoneInstance` points to the corresponding HMM definition. The actual HMM definitions are stored as an HMM list in `hlist`. This structure contains an array of `HMMEntry`'s where each entry holds a logical HMM name `lName`, a physical HMM name `pName` and a pointer `def` to the corresponding HMM definition (see Section 6.5 of [Young 93] pp.43-47). The `PhoneInstance` contains the array `*state` of storage cells indexed by HMM state, each of which holds a token. A token represents an alignment path by the `*link` member and its probability `logProb` between the HMM system and the unknown utterance up to the current time frame. Every state holds a single token since the best path to the current input frame could end in any HMM state. The basic token passing paradigm algorithm simply propagates tokens from each state to every connecting state updating the probability and history information at each step. The algorithm is as follows

```
InitModels;
for (t-1;t<=T;t++){
   Calculate_output_probs();
   StepModels():       /* ie propagate tokens within the models */
   CheckActive();      /* deactivate models with low prob tokens */
   NullEntryTokens(); /* put null token in all entry states */
   PropExitTokens();  /* copy best exit tokens into succ models */
   if (t<T) RecordDecisions();
   else                /* trace back to get optimal seq of models */
      OutputResults();
}
```

This algorithm repeatedly propagates tokens through the network of HMM instances.

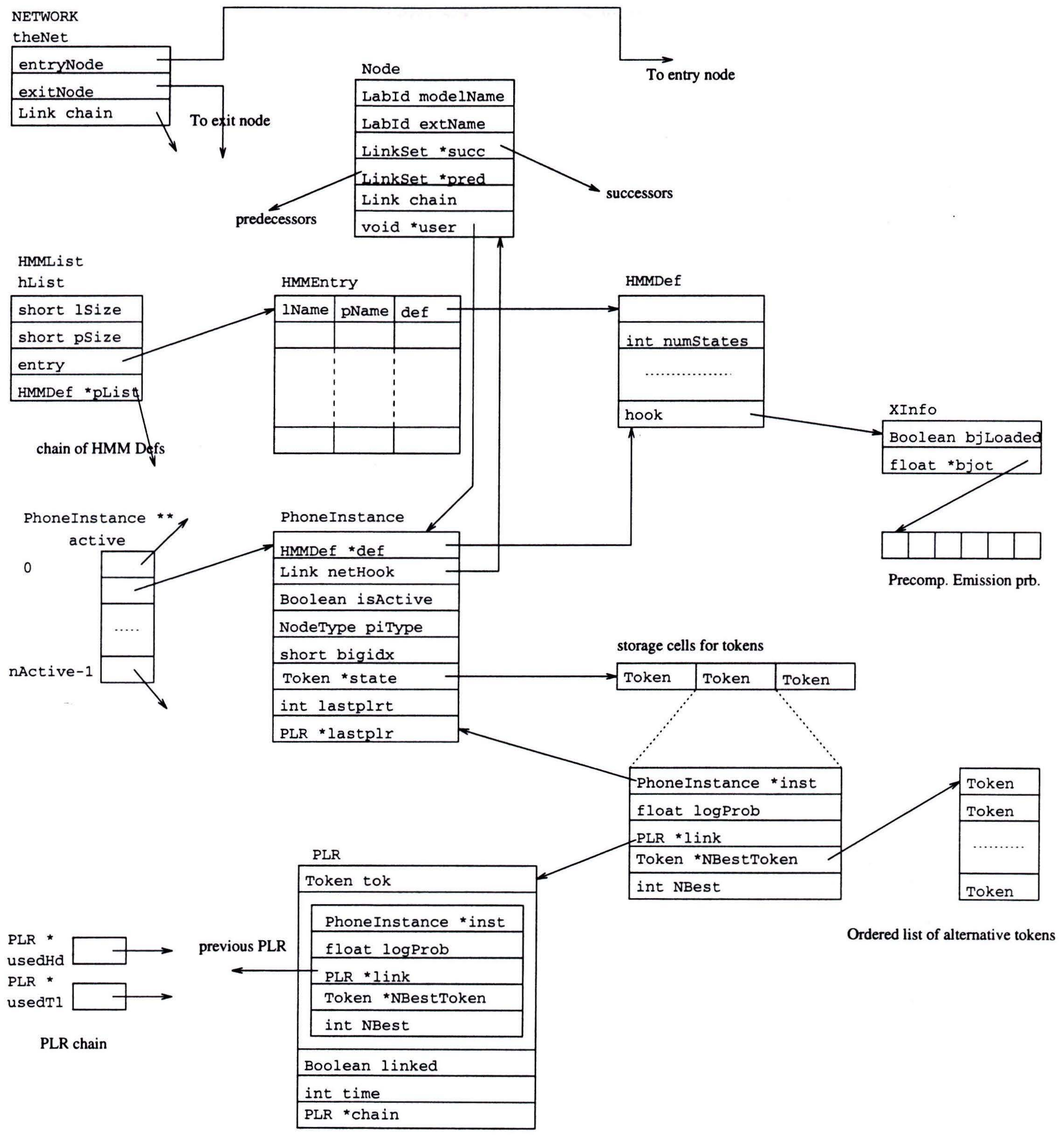

Figure 6.2: *The main datastructures of* **Viterbi**

First tokens are passed through each of the HMM instances (see `StepModels` in source code) and then between models according to the network links (see `PropagateExitTokens` in source code). Each *best* token propagated from a distinct `WD_END` or word-external node is recorded in the form of a *phone link record* (`PLR`) (see `RecordDecisions` in source code). For this reason, a `PLR` has a container `tok` for a token. When the whole utterance has been matched, the last emitted best token is traced back to give the best match sequence (see `OutputResults`).

When N-Best alternatives are generated, each token is accompanied by an ordered list of tokens `*NBestToken` denoting the alternative word predecessors of the current word. This list is constructed when entering a new word in `PropagateExitTokens` and propagated until a distinct `WD_END` is reached. Now not only the *best* token is recorded in the form of a *phone link record* (`PLR`), but also the tokens of the associated ordered token list (see `RecordDecisions` in source code). During backtracking, the N-Best paths are traced back from the last emitted best token(s) to give the lattice N-Best match sequences (see `OutputResults` in source code).

Calculation of emission probability

Each HMM definition has extra information associated with it, stored in an `XInfo` record attached to the `hook` field of the `HMMDef`. This `XInfo` record has an array called `bjot` for holding the emission probabilities for each state of the model for the current observation vector. This allows the emission probabilities to be calculated just once regardless of how many instances there are for each model.

Viterbi has two additional ways of computing emission probabilities for cases where the system is shared or fully-tied (semi-continuous). In order to reduce computation in HMM systems with parameter sharing, `PreComp` records are attached to shared `StateInfo` and shared `MixPDF` records (see Section 7.2 of [Young 93] pp.65.71). When a probability is computed for time frame t, then the probability and time are stored in the `PreComp` record. Whenever an emission probability was already calculated for the current time, then that pre-stored value is used instead. For fully tied-mixture (semi-continuous) systems, and mixture probabilities for each independent data stream are pre-computed and stored in the global matrix `tmProb`. These are then used for calculating emission probabilities. If the threshold is set by the `-c` option, then all mixture probabilities which fall below the threshold of the maximum are ignored.

Token propagation

Between models token propagation proceeds in a number of stages. Tokens are first propagated between word-internal nodes, and then from the `WD_END` nodes to all external nodes, `WD_BEGIN` or `WD_CAT` nodes (if any). When tokens are entering `WD_CAT` nodes, these tokens are immediately propagated to the subsequent `WD_BEGIN` nodes.

The (log) transition probabilities between any two connected word external models and/or composite word models is defined as

$$s \log[P(M_j|M_i)] + p$$

where M_i and M_j are the word-external or word composite models, s is a grammar scale factor and p a fixed transition score (word entrance penalty).

If a bigram is used then

$$P(M_j|M_i) = \text{bigram}[i, j]$$

When composite models are used, bigram indices are *always* kept at the attached `PhoneInstance` records of the `WD_BEGIN` and `WD_END` nodes or word-external nodes.

When no bigram is used then

$$P(M_j|M_i) = \frac{1}{N_{succ}(i)}$$

where $N_{succ}(i)$ is the number of successors to model M_i. For word-internal nodes the (log) transition probability is always $-\log[N_{succ}(i)]$.

If a bigram language model is not used, computational savings in token propagation can be made when there are shared `LinkSets` in the network. This is implemented by allocating space for a token for each `LinkSet` in the array `linkSetTokens` and performing a two stage propagation operation(see `PropExitTokensLS` in source code). If a bigram is used, the computational saving cannot be made and `PropExitTokensBG` is used.

When a N-Best ordered list is associated to a token, this list is left unchanged during word-internal token propagation. Only during propagation from all `WD_END` nodes to new `WD_BEGIN` (or `WD_END`) nodes the tokens are inserted in a new list. This insertion only occurs if the token belongs to the N-Best alternative tokens associated to the best token propagated from the new `WD_BEGIN` node. In the `LinkSet` case this list is created once for a `LinkSet` and propagated to the connected `WD_BEGIN` nodes (see `PropExitTokensLS`) whereas in the bigram case the list is constructed once for each `WD_BEGIN` node (see `PropExitTokensBG`).

Beam Search

For beam searching, **Viterbi** keeps a list of all active phone instances with HMMs attached in the global array `active`. Inter-model token propagation is then modified such that only the output tokens from active models whose scores lie within the beam, are passed to successor models. If the receiving model is inactive then it is activated. After each cycle, all active models are checked and any model containing no token within the beam is deactivated. Further, active lists for `WD_BEGIN` nodes (`beginActive`) and `WD_END` nodes (`endActive`) are maintained for composite word-model propagation and a further list of active `LinkSet` tokens for the two-stage token propagation in `LinkActive` (when no bigram is used).

Recording optimal path and boundaries

All PLRs generated to record phone boundaries are chained in a linked list with head and tail stored in `usedHs` and `usedTl`, respectively. The fields `lastplrt` and `lastplr` in each `PhoneInstance` are used to record the last PLR generated for that instance in order to allow `RecordDecisions` to avoid generating duplicate PLRs. PLRs are only generated for `WD_END` nodes and word-external nodes.

During generation of PLRs for N-Best alternatives, if a PLR is created for a token arriving at a node, then PLRs are also created for all alternative tokens in the ordered list associated with the best token (see `RecordDecision`). These PLRs are also inserted in the linked list (`usedHd` and `usedTl`).

Garbage Collection

Garbage collection is performed whenever the number of PLRs generated since the last garbage collection exceeds the margin `GCMARGINB`. The PLRs attached to all active tokens are traced back and marked as in use, the `linked` field of each PLR is used for this. Then all PLRs not marked are recovered by putting them back into the free list `freeHd/freeTl` (see `GarbageCollectPLR`).

When N-Best PLRs are generated, all alternative tokens also have to be traced back and marked because they could be needed later for generating an alternative match sequence. At this moment, avoiding the undesired removal of N-Best PLRs is implemented by turning off the garbage collection. A recursive removal of only the correct PLRs was implemented, but this version is not completely error free because of some unrevealed constraints in the PLR data structure.

Backtracking

When all tokens have reached the end of the network (`EXIT` node), a back tracking procedure is started. Parallel with the token passing from word hypothesis to word hypothesis, partial sentence hypotheses are created. As stated, when a token leaves the last word of a partial sentence, then the corresponding score, a description of the last word and a backpointer to the PLR of the previous word in the partial sentence is stored. This backpointer registers the continuation of the best partial sentence, observed when the token entered the last word. So by backtracking through the backpointers of all preceding PLRs of the best scoring token reaching the end of the network, the entire corresponding best scoring sentence can be constructed (see `OutputResults` in source code).

When the N-Best sentences have to be constructed, the backtracking has to use the back pointers generating the N-Best alternatives. Since for each PLR, the backpointers to the N-Best preceding PLRs (registering the continuation of the N-Best partial sentences observed when the best token entered the last word) and their score differences are known, the alternatives can be generated in an iterative way. When generating the best sentence, an ordered list of alternative PLR scores and back pointers is updated when better alternatives are observed along the alternatives of the PLRs in the best sentence. Once the best sentence is generated, the next candidate of this list will be the next best alternative and the corresponding PLR backpointer can be used to generate the next alternative. Again the ordered list is updated with observed better scoring alternatives of the PLRs along the next best sentence and corresponding PLR backpointers are stored. This process is repeated until the N-Best alternative sentences are generated.

6.13 Small Tutorial

In order to illustrate the practical use of the tools, this section is devoted to a shell script that realizes the hierarchical approach. The tools are embedded in a C-shell script cut into pieces for didactic reasons. It depicts the fully automatic procedure for segmentation. This script can easily be adopted and subsequently adapted for specific needs.

We assume that all preparation work regarding the transcription and labelling of speech has already been done. Therefore, all speech audio files are stored in a directory called `Speech` and the corresponding transcription files in directory `Label`. The demonstration of the hierarchical approach is performed in a sequence of 7 steps and a concluding performance assessment step.

1. **BroPhon** is invoked for obtaining the segmentation in broad phonetic classes for each sampled speech data file. It accomplishes its task iteratively while the maximum number of iterations is fixed at 10. It takes the transcription of each `speechFile` from files present in the directory `Label`. The output label files are stored in the directory `BPCResults` for further processing. Also, it uses the file `BPCMappingList` that has been created beforehand and that contains duration statistics for each phoneme-like unit and the mapping from phoneme to broad phonetic class. Many default settings of the options are in force.

2. **SeqQuant** is invoked for the Sequence Constrained Vector Quantization step for each sampled speech data file. It takes the transcription of each `speechFile` from files present in the directory `Label`. It uses the file `SCVQMappingList` that has been created beforehand and that contains duration statistics for each phoneme-like unit and the mapping from phoneme to broad phonetic class. In contrast with the previous step, `SCVQMappingList` contains more restricted durations allowed for each phoneme-like unit. The broad phonetic class segmentation results are read in from the files in directory `BPCResults`. The output label files are stored in the directory `SCVQResults` for further processing. Many default settings of the options are in force.

```
#!/bin/csh
# UNIX C-Shell Script for Hierarchical Approach
# Steffen Pauws

echo " ..................."
echo " ... STEP 1 AND 2 ..."
echo " ... Broad Phonetic Class Segmentation ..."
echo " ... Sequence Constrained Vector Quantization ..."
foreach speechFile ( Speech/*.aiff )
   BroPhon -i 10 -L Label -S BPCResults BPCMappingList $speechFile
   SeqQuant -L Label -D SCVQMappingList -B BPCResults $speechFile
end
```

3. **FrontEnd** is invoked for converting each sampled speech data file into an observation sequence of filterbank vectors. Each analysis frame is explicitly Hamming-windowed. Each vector is augmented with delta, delta-square, and energy features. The result files are stored in the directory `AcVec`. Many default settings of the options are in force.

```
echo " ............."
echo " ... STEP 3 ..."
echo " ... Front End : Filterbank Analysis ..."
foreach speechFile ( Speech/*.aiff)
    set j="`basename $speechFile`"
    set base=`echo $j | cut -d. -f1`
    if ( -e AcVec/${base}.erb ) then
       echo "Data File ${base}.erb already created"
    else
       echo "FrontEnd $speechFile --> AcVec/${base}.erb"
       FrontEnd -h -d -D -e $speechFile AcVec/${base}.erb
   endif
end
```

4. **GenMarkov** is invoked to generate two HMM definition files describing the topology and structure for each HMM. `HMMDef` is a 3-state model that is defined into 4 data streams. The first three data streams representing respectively instantaneous filterbank, delta, and delta square features are 16-dimensional each. The last stream for the energy features is only 3-dimensional. Each data stream is modelled by a 4-mixture Gaussian density. `HMMBurst` is a one-state model with the same configuration at the state.

```
echo " ............."
echo " ... STEP 4 ..."
echo " ... Generating HMM topology and structure ..."
GenMarkov -s 3 -d 4 16 16 16 3 -m 4 HMMDef
GenMarkov -s 1 -d 4 16 16 16 3 -m 4 HMMBurst
```

5. **SegKMeans** is invoked for a Segmental K-Means run for each HMM and its goal is to create initial HMMs relying solely on the segmentation provided by **SeqQuant** as present in directory `SCVQResults`. The extension of these transcription files is `seg`. The tool requires parametrized speech data as present in directory `AcVec`. It must be emphasized that this tool has to be invoked for each HMM modelling a phoneme-like unit. For obvious reasons, we only show four examples. HMM definition `HMMDef` is used for the phoneme-like units `S` and `A`, whereas the one-state model `HMMBurst` is especially for the bursts `P2` and `T2`. The initialized HMMs are stored in a directory called `hmm.skm`. Many default settings of the options are in force.

```
echo " ............."
echo " ... STEP 5 ..."
echo " ... Segmental K-Means Initialization ..."
SegKMeans -X seg -L SCVQResults -l S -o hmm.skm/S -T 1 HMMDef AcVec/*.erb
SegKMeans -X seg -L SCVQResults -l A -o hmm.skm/A -T 1 HMMDef AcVec/*.erb
SegKMeans -X seg -L SCVQResults -l P2 -o hmm.skm/P2 -T 1 HMMBurst AcVec/*.erb
SegKMeans -X seg -L SCVQResults -l T2 -o hmm.skm/T2 -T 1 HMMBurst AcVec/*.erb
```

6. **HERest** is invoked three times to conclude the training phase by three Baum-Welch reestimations using only the transcription and parametrized speech. The bootstrapped set of models in `hmm.skm` are used and their updated version is written to `hmm.ebw`. Each intermediate set of HMMs is stored in a temporary directory called `tmp`. It requires a file that lists all HMM models that must be trained. For that means, the file `PhonList` is generated using the UNIX-utilities `awk`, `sort`, and `uniq`. The transcription file for each parametrized speech file is loaded from the directory `Label`. Many default settings of the options are in force.

```
echo " ............."
echo " ... STEP 6 ..."
echo " ... Baum Welch Reestimation ... "
set iter=3
if ( ! -d tmp ) mkdir tmp
set i=1
set srcdir="hmm.skm"
set tgtdir="hmm.ebw"
awk ' { print $3 } ' Label/*.lab | sort | uniq > PhonList
while ( $i <= $iter )
    echo "Iteration $i of Baum-Welch Reestimation"
    HERest -d $srcdir -e $tgtdir -L Label -T 1 -v 0.0001 PhonList AcVec/*.erb
    @ i++
    if ( $i <= $iter ) then
       rm -fr tmp
       mv hmm.2 tmp
       mkdir hmm.2
       set srcdir="tmp"
    endif
end
rm -fr tmp
```

7. **Viterbi** is invoked in order to apply Viterbi decoding that is controlled by a very simple syntax network in a file called **Network**: it contains the transcription of the utterance. Hence, this network definition represents a recognition network in which all of the HMMs are placed in cascade, reducing the Viterbi decoding to a segmentation process. This file is generated simply by using the UNIX-utility **awk**. Also, the file PhonList is required representing a list of all HMM models that must be loaded by **Viterbi**. Each parameterized speech data file in directory **AcVec** is processed along its own network by using the fully-trained HMMs in directory **hmm.ebw**, its output being a set of transcriptions with their inferred boundary positions. The output label files are stored in the directory **HMMResults**. Many default settings of the options are in force.

```
echo " ............."
echo " ... STEP 7 ..."
echo ".. Automatic segmentation by Viterbi and HMMs .."
foreach acvecFile ( AcVec/*.erb )
   set j="`basename $acvecFile`"
   set base=`echo $j | cut -d. -f1`
   awk ' { print $3 } ' Label/${base}.lab | sort | uniq > PhonList
   awk 'BEGIN { print "("} { print $3 } END { print ")"}' Label/${base}.lab > Network
   Viterbi -d hmm.ebw -L HMMResults -T 0200 PhonList Network SampaAcVec/${base}.erb
end
```

8. **Assess** is invoked in order to evaluate the accuracy. Both the automatically obtained boundaries provided by **SeqQuant** and **Viterbi** (as present in the directories SCVQResults and HMMResults) are compared with the manually positioned boundaries in directory Label. The output will be a table with global cumulative statistics about the accuracy. Moreover, a list of all HMM models PhonList is required.

```
echo " ............."
echo " ... STEP 8 ..."
```

```
awk ' { print $3 } ' Label/*.lab | sort | uniq > PhonList
echo ".. Segmentation performance SCVQ ... "
Assess -g 10 -L Label PhonList SCVQResults/*.seg
echo ".. Segmentation performance HMMs .."
Assess -g 10 -L Label PhonList HMMResults/*.rec
```

Appendix A
Phonological processes found in the speech inventory

As a result of a careful examination of the speech inventory several types of co-articulation such as assimilation has been detected. A relatively large set of utterances was suspected of having segments that were not labeled correctly in a phonetic sense. Special attention has to be paid to these phenomena in order to provide the automatic segmentation process with a transcription that correctly matches the acoustic realization.

Before giving an overview of the specific co-articulation realizations found in the inventory, we give a short description of some typical co-articulation processes[Jongenbur 91]

Syllable structure related phoneme modifications Awkward sequences of only consonants or only vowels are avoided by elision and insertion of consonants:

1. consonant cluster type such as degemination
2. schwa-insertion after l and r
3. r-elision
4. glottal stop

Assimilation Phonological features of segments are changed under the influence of their phonetic context.

1. carry over assimilation of voice
2. anticipatory assimilation of voice

We have attempted to categorize all co-articulation realizations. Unfortunately, some factors that influence some sound adjustments were hard to highlight (in some case it was just a 'wrong' pronunciation), so some rather broad categories are maintained.

- Schwa-insertion/elision. In Dutch a schwa can be inserted, for instance between a non-nasal sonorant and a nasal if both consonants belong to the same syllable. Schwas in unstressed syllables can be elided when they occur before liquids.
- *r*-elision. In Dutch the elision of *r* is very common. Especially in the first (unstressed) syllable of a word.
- Glottal stops. In general, a glottal stop is inserted when a vowel is in hiatus position, for instance in a sequence of two (identical) vowels.

- Confusion voiced⟶unvoiced. No attempt has been made to identify whether it was a carry-over or anticipating type or just a 'wrong' pronunciation.
- Confusion unvoiced⟶voiced. No attempt has been made to identify whether it was a carry-over or anticipating type or just a 'wrong' pronunciation.
- Consonant clusters. In Dutch two adjacent identical consonants are reduced to one (degemination). It also holds for many sound-a-like consonants.
- Artefacts (corruption). A corruption of the waveform has occurred during preparation of the file.

schwa-insertion/elision

index	transcription	co-articulation	corrected?
D1129	`S T1 T2 CC K1 K2 W E R - K1 K2`	/@/-insertion	yes
D1130	`V EE L W E R - K1 K2`	/@/-insertion	yes
D223	`- Z CC Z C Z C`	/@/-insertion	yes
D250	`- Z C Z CC Z C`	/@/-insertion	yes
D570	`K1 K2 - N II C N`	/@/-insertion	yes
D662	`K1 K2 E R - K C R`	/@/-insertion	yes
D692	`K1 K2 E R - M C`	/@/-insertion	yes
D706	`K1 K2 A R - N C`	/@/-insertion	yes
D859	`S C S OO S` $\underline{\texttt{C}}$	/@/-elision	yes
D903	`D1 D2 O R - P1 P2 C`	/@/-insertion	yes
D989	`D1 D2 U` $\underline{\texttt{C}}$ `K1 K2 C N`	/@/-deletion	yes
E60	`N E T1 T2 W E R - K1 K2`	/@/-insertion	yes

r-elision

index	transcription	co-articulation	corrected?
D1230	`V C` $\underline{\texttt{R}}$ `Z A K1 K2 C`	/*r*/ elision	yes
D380	`Z EI1 EI2 E R X E` $\underline{\texttt{R}}$ `T1 T2`	/*r*/ elision	yes
D468	`G EE L F I L T C` $\underline{\texttt{R}}$	/*r*/ elision	yes
D648	`V C` $\underline{\texttt{R}}$ `J AA X C N`	/*r*/ elision	yes
D903	`D O` $\underline{\texttt{R}}$ `P C`	/*r*/ elision	yes
D928	`K1 K2 EI1 EI2 K1 K2 S EI1 EI2 F C` $\underline{\texttt{R}}$ `S`	/*r*/ elision	yes
E35	`T1 T2 UI1 UI2 N V EI1 EI2 V C` $\underline{\texttt{R}}$	/*r*/ elision	yes

Glottal stop

index	transcription	co-articulation	corrected?
D320	`V EE - AU1 AU2 T1 T2 OO`	glottal stop	no
D339	`N EE - O N D1 D2 C R`	glottal stop	no
D374	`B EI1 EI2 - AU1 AU2 D1 D2 C R S`	vocal fry	no
D397	`Z EI1 EI2 - O P1 P2 C R T`	glottal stop	no
D398	`Z EI1 EI2 - OO X T`	glottal stop	no
D409	`Z EI1 EI2 - U F C N T`	glottal stop	no
D410	`V R EI1 EI2 - UI1 UI2 T1 T2`	glottal stop	no
D414	`B EI1 EI2 - Y R C`	glottal stop	no
D607	`D II - UI1 UI2 T1 T2`	glottal stop	no
D865	`Z OO - UI1 UI2 T1 T2`	glottal stop	no

Confusion voiced⟶unvoiced

index	transcription	co-articulation	corrected?
D1112	A F V A Q C N	/*v*/ unvoiced	yes
D1121	J A S V A L C	/*v*/ unvoiced	yes
D1123	D1 D2 U SJ V AA K1 K2	/*v*/ unvoiced	yes
D1124	UI1 UI2 T1 T2 V EE X C	/*v*/ unvoiced	yes
D1125	W E X V A L C	/*v*/ unvoiced	yes
D1222	P1 P2 A K1 K2 Z A K1 K2 C	/*z*/ unvoiced	yes
D1223	S T1 T2 O F Z A K1 K2	/*z*/ unvoiced	yes
D1231	W A S Z A K1 K2	/*z*/ unvoiced	yes
D1233	D1 D2 U SJ Z E L C F	/*z* unvoiced	yes
D1235	W E X Z A K1 K2 C	/*z* unvoiced	yes
D124	G1 G2 E G1 G2 AU1 AU2 G1 G2 C	/*g*/ unvoiced	yes
D143	V C V AU1 AU2 V C	/*v*/ unvoiced	yes
D146	Z C Z AU1 AU2 Z C	/*z*/ unvoiced	yes
D180	V EE L B1 B2 EE T C	/*v*/ unvoiced	yes
D223	Z CC Z C Z C	/*z*/ unvoiced	yes
D313	V C V E V C	/*v*/ unvoiced	yes
D320	V EE AU1 AU2 T1 T2 OO	/*v*/ unvoiced	yes
D352	V C V EE V C	/*v*/ unvoiced	yes
D356	Z C Z EE Z C	/*z*/ unvoiced	yes
D397	Z EI1 EI2 O P1 P2 C R T	/*z* unvoiced	yes
D411	V C V EI1 EI2 V C	/*v*/ unvoiced	yes
D561	V C V I V C	/*v* unvoiced	yes
D648	V C J AA X C	/*v*/ unvoiced	yes
D685	D1 D2 R CC G1 G2 M AA K C	/*g*/ unvoiced	yes
D736	V C V O V C	/*v*/ unvoiced	yes
D739	Z E Z O Z C	/*z*/ unvoiced	yes
D740	ZJ C ZJ O ZJ C	/*zj*/ unvoiced	yes
D765	V C V OE V C	/*v*/ unvoiced	yes
D775	Z C Z OH Z C	/*z*/ unvoiced	yes
D855	Z OO OO L OO X II	/*z*/ unvoiced	yes
D866	V C V OO V C	/*v*/ unvoiced	yes
D869	Z C Z OO Z C	/*z* unvoiced	yes
D896	D1 D2 R CC P1 P2 A K1 K2 C	/*d*/ unvoiced	yes
D938	Z E X S AA M C	/*z*/ unvoiced	yes
D940	G1 G2 C G1 G2 CC G1 G2	/*g*/ unvoiced	yes
D997	G1 G2 C G1 G2 U G1 G2 C	/*g*/ unvoiced	yes
E11	V CC V C V C	/*v*/ unvoiced	yes
E35	T1 T2 UI1 UI2 N V EI1 EI2 V C R	/*v*/ unvoiced	yes
E37	V EI1 EI2 N Z C	/*v*/ unvoiced	yes
E37	V EI1 EI2 N Z C	/*z*/ unvoiced	yes
E66	Z C Z A Z C	/*z*/ unvoiced	yes
F227	D1 D2 ZJ C D1 D2 ZJ CC D ZJ C	/*zj*/ unvoiced	yes

Confusion unvoiced⟶voiced

index	transcription	co-articulation	corrected?
D288	D1 D2 U SJ D1 D2 EE K1 K2 C	/*sj*/ voiced	yes
D406	SJ C SJ EI1 EI2 SJ C	/*sj*/ voiced	yes
D763	SJ C SJ OE SJ C	/*sj*/ voiced	yes
E39	S C S OE S C	/*s*/ voiced	yes
E40	S T1 T2 OO II S EI1 EI2 N S	/*s*/ voiced	yes
E50	B1 B2 O S B1 B2 AU1 AU2 W	/*s*/ voiced	yes

Consonant clusters

index	transcription	co-articulation	corrected?
D184	O P1 P2 B1 B2 A K1 K2 C	*/p/* elision	yes
D1240	T1 T2 II N ZJ U R N AA L	*/n/* elision	no
D277	D1 D2 R CC G1 G2 D1 D2 II L C R	unclear cluster */gd/*	no
D289	UI1 UI2 T1 T2 D1 D2 AA X C	*/t/* elision	yes
D481	M AA K1 K2 G1 G2 OO L S	*/k/* elision	yes
D486	O P1 P2 G1 G2 OO L S	*/p/* elision	yes
D671	X EE L L UI1 UI2 K1 K2	degemination	yes
D896	D1 D2 R U G1 G2 P1 P2 A K1 K2 C	*/g/* elision	yes
D965	W A S SJ AA K1 K2	unclear cluster	no
D967	D1 D2 U SJ SJ AA K1 K2	degemination	no
E30	UI1 UI2 M H UI1 UI2 T1 T2 C	*/h/* elision	yes
E31	Z W A M N OE S	unclear cluster */mn/*	no
F481	M AA G1 G2 K1 K2 OO L S	*/g/* elision	yes
F620	H C H II W H C	*/h/* elision	yes

Artefact (corruption)

index	transcription	co-articulation	corrected?
D214	T1 T2 CC T1 T2 C	no */t/*-closure present	yes
F633	T1 T2 C T1 T2 II W T1 T2 C	no */t/*-closure present	yes

Bibliography

[Alphen 92] ALPHEN, P. VAN, *HMM-based continuous-speech recognition, Systematic evaluation of various system components*, Thesis, PTT Research (1992)

[Angelini 93] ANGELINI B., BRUGNARA F., FALAVIGNA D., GIULIANI D., GRETTER R., OMOLOGO M., *Automatic Segmentation and Labeling of English and Italian Speech Databases*, EuroSpeech 93, VOL.1, pp.653-656.

[Boëffard 92] BOËFFARD O., MICLET L., WHITE S., *Automatic Generation of Optimized Unit Dictionaries for Text to Speech Synthesis*, ICSLP, Bauff 1211-1215, 1992.

[Boëffard 93] BOËFFARD O., CHERBONNEL B., EMERARD F., WHITE S., *Automatic Segmentation and Quality Evaluation of Speech Unit Inventories for Concatenation-based, Multilingual PSOLA Text-to-Speech Systems*, EuroSpeech 93, VOL.2, pp.1449-1452.

[Brugnara 92] BRUGNARA F., FALAVIGNA D., OMOLOGO M., *A HMM-based System for Automatic Segmentation and Labeling of Speech*, Proc. Int. Conf. Spok. Lang. Syst., VOL.1, Alberta, 1992.

[Cosi 91] COSI P., FALAVIGNA D., OMOLOGE M., *A Preliminary Statistical Evaluation of Manual and Automatic Segmentation Discrepancies*, EuroSpeech 91, VOL.2, pp.693-696.

[Davis 80] DAVIS S.B., MERMELSTEIN P., *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences*, IEEE Trans. Acoustics, Speech, Signal Proc.,ASSP-28(4), pp.357-366, August 1980.

[ESPRIT 92] ESPRIT PROJECT 2589, *SAM Multi-lingual speech input/output assessment methodology and standardisation*, Final report, Year Three, SAM-UCL-G004, Central Services, University College London, 1992.

[Farhat 93] FARHAT A., PÉRENNOU G., ANDRÉ-OBRECHT R., *A Segmental Approach versus a Centisecond One for Automatic Phone Time-Alignment*, EuroSpeech 93, VOL.1, pp.657-660.

[Fukunaga 90] FUKUNAGA F., *Introducion to Statistical Pattern Recognition*, Academic Press, San Diego ,1990.

[Gray 84] GRAY R.M., *Vector Quantization*, IEEE ASSP Magazine April 1984.

[Glasberg 90] Glasberg B.R., Moore B.C.J., *Derivation of auditory filter shapes from notched-noise data*, Hearing Research, No. 47, pp.103-138,1990.

[Haeb 92] Haeb-Umbach R., Ney H., *Linear Discriminant Analysis for Improved Large Vocabulary Continuous Speech Recognition*, Proc. IEEE Int. Conf. Acoustics, Speech & Signal Processing, ICASSP-92, vol. S1, pp. 13-16.

[Hemert 85] Hemert J.P. van, *Automatic diphone preparation*, IPO Annual Progress Report 20, 1985.

[Hemert 87] Hemert J.P. van, *Automatische segmentering van spraak in difonen*, Philips Technisch Tijdschrift, Jaargang 43, nr.3, feb. 1987.

[Itakura 75] Itakura F., *Minimum Prediction Residual Principle Applied to Speech Recognition*, IEEE Trans. Acoustics, Speech, and Signal Proc., VOL. ASSP-23, No.1, pp.67-72, February 1975.

[Jongenbur 91] Jongenburger W., *Sandhi processes*, Final Report FONWET, Report no.35, Speech Technology Foundation, 1991.

[Kernigha 88] Kernighan B., Ritchie D., *The C Programming Language*, 2nd edition, Prentice-Hall, 1988.

[Lamel 87] Lamel L., Kassel R., Seneff S., *Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus*, Proc. DARPA Speech Recognition Workshop, pp. 26-32, March 1987.

[Leung 84] Leung H.C., Zue V.W., *A procedure for automatic alignment of phonetic transcriptions with continuous speech*, IEEE ICASSP, San Diego CA, 1984.

[Ljolje 91] Ljolje A., Riley M.D., *Automatic Segmentation and Labeling of Speech*, Proc. IEEE Int. Conf. on Acoust. Speech and Sig. Proc., Toronto, pp. 473-476, 1991

[Ljolje 93] Ljolje A., Riley M.D., *Automatic Segmentation of Speech for TTS*, EuroSpeech 93, VOL.2, pp.1445-1448.

[Ma 94] Ma C., Kamp Y., Willems L.F., *A Frobenius Norm Approach to Glottal Closure Detection from the Speech Signal*, IEEE Trans. Speech and Audio Proc., VOL 2, NO 2, April 1994.

[Pauws 93] Pauws S.C., Ceelen M., *MARS: Medical Application for Recognition of Speech*, Final report of the post-graduate programme Software Technology, Eindhoven University of Technology, september 1993.

[PHONDAT 92] Pompino-Marschall, B. (ed.), *PHONDAT: Verbundvorhaben zum Aufbau einer Sprachsignaldatenbank für gesprochenes Deutsch*, Forschungsberichte Institut für Phonetik und Sprachliche Kommunikation (FIPKM), 30, pp.99-128, 1992.

[Rabiner 89] Rabiner L.R., *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recogition*, Proc. IEEE Trans. on Acoust., Speech, and Sign. Proc., 77(2), pp.267-295, 1989.

[Rabiner 93] RABINER L.R., JUANG B. *Fundamentals of Speech Recognition*, Prentice-Hall Inc.

[Schmidt 91] SCHMIDT M.S., WATSON G.S. *Evaluation and optimization of automatic speech segmentation*, EuroSpeech 91, VOL.2, p.701-704.

[Svendsen 87] SVENDSEN T., SOONG F.K., *On the Automatic Segmentation of Speech Signals*, Proc. ICASSP, pp.77-80, April 1987.

[Taylor 91] TAYLOR P.A., ISARD, *Automatic Diphone Segmentation*, EuroSpeech 91, VOL.2, p.709-711.

[Veenker 94] VEENKER T.J.G., *Programmer's Guide IPO/OTS IO Software Library*, IPO report no. 129.

[Vidal 90] VIDAL E., MARZAL A., *A Review and New Approaches for Automatic Segmentation of Speech Signals*, SIGNAL PROCESSING V: Theories and Applications, Elsevier Science Publications, 1990, pp.43-53.

[Wilpon 85] WILPON J.G., RABINER L.R., *A Modified K-Means Clustering Algorithm for Use in Isolated Work Recognition*, IEEE Trans. Acoust., Speech, and Sign. Proc., VOL. ASSP-33, June, 1985.

[Young 93] YOUNG, S.J., *HTK: Hidden Markov Model Toolkit*, University of Cambridge, Entropic Research Laboratory, February 1993.