PhD Thesis

presented in order to obtain the degree of

DOCTEUR EN SCIENCES
DE L'UNIVERSITÉ PARIS XI ORSAY

by

Jan Černocký

SUBJECT:

Speech Processing Using Automatically Derived Segmental Units: Applications to Very Low Rate Coding and Speaker Verification

defended on December 18th, 1998

Committee of Referees

Guy DEMOMENT	Chairman
Jean MENÉZ	Referees
Hynek HEŘMANSKÝ	
Gérard CHOLLET	Examiners
Vladimír ŠEBESTA	
Geneviève BAUDOIN	

Abstract

Current systems of automatic speech processing (ASP), including the recognition, synthesis, very low bit-rate (VLBR) coding and text-independent speaker verification, rely on sub-word units determined using phonetic knowledge. This thesis is aimed on an alternative approach – determination of speech units using ALISP (Automatic Language Independent Speech Processing) techniques. The proposed method consists of two steps: first, the speech unit set is defined and initial transcription of a speech database is obtained; the temporal decomposition (TD), unsupervised clustering and multigrams have been used. Then, the units are modeled using Hidden Markov Models (HMM). An iterative procedure for refinement of model set was developed. First tested application was the very low bit-rate speech coding. Issues specific to the re-synthesis in the decoder had to be solved. Using given set of units, intelligible speech was obtained at mean bit-rate of 120 bps (for unit encoding) in two sets of speaker-dependent experiments. The second application was the use of those units as a pre-processing for segmental speaker verification system based on multi-layer perceptrons (MLP). We have shown, that the segmental system reached comparable performances as the system with global scoring using one MLP. Correspondence of ALISP segmentation with phonetic transcriptions has been studied: a consistent mapping was found, which was however far from one-to-one correspondence. Possible ways of using ALISP units in speech recognition are discussed.

Résumé

Les systèmes courants de traitement automatique de la parole (TAP) – la reconnaissance, la synthèse, le codage à très bas débit et la vérification du locuteur indépendante du texte – sont basés sur des unités de type sous-mot, définies à l'aide d'un savoir-faire phonétique. Cette thèse porte sur une approche alternative : une détermination des unités à l'aide des techniques ALISP (Traitement Automatique de Parole, Indépendant de la Langue). La méthode proposée comporte deux étapes : premièrement, l'ensemble des unités est défini et une transcription initiale de la base de données de parole est obtenue; nous avons utilisé des techniques de décomposition temporelle (DT), de classification non-supervisée et des multigrammes. Ensuite, nous modélisons les unités à l'aide des Modèles de Markov Cachés (HMM). La première application testée a été en codage de parole à très bas débit. Nous avons dû résoudre des problèmes spécifiques à la re-synthèse dans le décodeur. En utilisant l'ensemble d'unités donné, nous avons obtenu un signal de parole intelligible au débit moyen de 120 bps (pour le codage des unités) dans deux jeux d'expériences dépendantes du locuteur. Dans une deuxième application, nous avons utilisé ces unités comme pré-traitement d'un système de vérification segmental, basé sur des perceptrons multi-couche (MLP). Nous avons montré que le système segmental obtenait des performances comparables à un système avec une détermination globale des scores. Nous avons étudié la correspondance entre une segmentation ALISP et une transcription phonétique : nous avons trouvé une correspondance consistante, si bien qu'éloignée d'une correspondance biunivoque. Nous discutons par ailleurs, différentes voix possibles d'utilisation des unités ALISP en reconnaissance de parole.

Acknowledgements

First, I would like to thank the members of jury for having accepted to assist at this PhD defense and for their precious time they spent to come to Orsay. I am especially grateful to the referees: to **Hynek Heřmanský** who has agreed to come from the US. Thank you also for having assisted at my pre-defense in Lausanne, for precious remarks and discussions about my work and also for the support to all speech people in Brno. I am also grateful to **Jean Menéz**, for having accepted to come from Nice and for the nice discussion, we had on the sea coast in Saint Raphaël. I owe a lot to **Guy Demoment**, not only for having accepted to become "Orsay's deputy" and the president of the jury, but also for excellent basis of stochastic signal processing he gave us during his DEA courses.

But the greatest "scientific" thanks come to the trinity of my thesis directors. **Geneviève Baudoin**, even if not officially the supervisor, has attracted me to the scientific work in speech processing and she was my first advisor throughout the thesis. I hope I will once be able to render her all her help.

Now comes the time to thank the "big boss", **Gérard Chollet**. Not only was he the French head of my PhD, but also the person who familiarized me with the speech processing community and who was an invisible originator of many my steps. Even if, in the first time, I had a hard time to to find a way to his charisma, now I am very grateful for all his manipulations[1], hundreds of emails, and hours in his smoky office at ENST.

I wish to thank also the Czech supervisor, **Vladimír Šebesta**, for precious remarks he made on my report, and especially for the organizing work he is doing in Brno. Thanks to him, the "Signal Processing Lab" was created and after a difficult start, it begins to be a good place for working.

Next thanks go to all the colleagues, I met during this thesis: I am especially grateful to **Frédéric Bimbot**, for having allowed us to use his temporal decomposition package, and also for all the remarks and discussions I had with him throughout this work. The same holds for **Sabine Deligne**, who has been a "scientific star" in my eyes and whose works on sequence modeling I appreciate a lot.

I owe a lot to **Dijana Petrovska-Delacrétaz** and to **Jean Hennebert** for the nice (although hard) time at EPFL Lausanne during the NIST'98 speaker verification campaign. Never can one learn so much on a topic, than when working intensively with people strong in the subject. Thanks also to all colleagues of Dijana and Jean and especially to the head of CIRC, **Martin Hassler**, for having received me in their group so nicely.

There are many others I wish to thank: the other PhD students of Gérard Chollet: **Guillaume Gravier** and **Andrei Constantinescu**, especially for the HTK support they gave me. My dear colleagues at ESIEE: **Pascale Jardin**, **Jean-François Bercher** and **Olivier Venard**, thank you for nice time I spent at the Department of Signals and Telecommunications, for the help you gave me and for all the nice moments we had together (special thanks to Olivier for having shared the office with me). The colleagues in Brno: **Milan Sigmund** for having given me the first education in speech processing, and the others: **Zbyněk Raida**, **Aleš Prokeš**, **Radek Plachejda**, **Milan Brejl**, **Říša Menšík**, for all the discussions we had either in the lab or in various Brno's pubs...

This work could not have been done without computers and computer people. I begin with the SMIG department of ESIEE: thank you, **Jean-Claude Arnouil**, **Ming Eng**, and **Frank Bonnet**. For Brno, I wish to thank **Zdeněk Kolka**, having so hard time administering all those PC's at the Institute, and the staff of the Center of Computing and Information Services (CVIS) of the University, especially **Ludvík Kania**.

[1] all PhD student and all co-workers of Gérard are subject to his manipulations...

The life would be nothing without friends. Impossible to mention all of them here, but some had nevertheless a direct influence on this PhD, as they taught me, how the French looked like in the real life. Thank you **Paco**, **Marie**, **Charlotte**, **Mehdí**, and the others, for everything.

But the life would be also impossible without money. I am grateful to the **French Government** for the scholarship No. 94/4516, which allowed me to do this magnificent Czech-French experience.

The last thanks go to my family: of course to my parents **Jarmila** and **Jaroslav Černočtí**, for having taught me to speak[2] and for having supported me throughout my studies. The very last and greatest thank is for my wife **Hanka** for all she has done, she is doing, and she will do and be for me.

[2]First obligatory step in the career of speech scientist.

Contents

List of Figures

Chapter 1

Introduction

Many modern systems for speech processing rely on sub-word units: in large vocabulary continuous speech recognizers (LVCSR), those units are the intermediate level between parametric description of speech and higher levels (word decoding, language modeling, syntactic parsing, etc.). In text-to-speech synthesizers (TTS), they limit the dictionary size and permit the synthesizer to "read" a previously unseen word. In speaker recognition and verification, a pre-classification of sub-word units and a use of multiple scoring systems leads often to schemes with better performances than those relying on modeling of "global" probability distribution functions. Finally, in very low bit-rate coders, one can no more transmit information about each frame: grouping frames into larger units, and labelling of those units is necessary for an efficient transmission and/or storage.

Classically, the set of such units is based on historically established description of language: on phonemes, their derivatives (diphones, context-dependent phonemes,...), syllables, or other entities linked with the *text*. To make an automatic system use those units, one needs to know their position in the speech signal: in the recognition, phonetically labelled or at least transcribed databases are widely used, while in the synthesis, an important amount of human efforts must be devoted to the creation of (usually) diphone dictionary. The same holds for the use of such units in identification/verification or coding: for a task, where a database with appropriate phonetic information is available, it is preferable to purchase it; if such a product is not available, one must start to create it himself.

This thesis deals with an alternative approach to that cited above: with an unsupervised determination of basic speech units using data-driven approaches, operating directly on *non-transcribed* speech corpora. Recent advances in Automatic Language Independent Speech Processing (ALISP) [20] have shown, that many tasks relying currently on human knowledge can be performed more efficiently using those data-driven approaches. From a practical point of view, this brings revolutionary changes to the methodology of speech processing: extensive human efforts can be replaced by an automated process.

We describe a generic algorithm for determination of speech units. It is based on a detection of quasi-stationary segments in the parametric representation, and on a classification of those segments to clusters obtained by unsupervised training. Optionally, characteristic sequences of such segments can be found, so that the resulting units are longer. All these steps are preparatory to Hidden Markov Model (HMM) modeling of units. However, in our case, HMMs are not the final product of the algorithm, but are used in an iterative procedure to refine the set of units.

The applications tested so far are the *very low bit-rate coding* and the *text-independent speaker verification*. Both of them were chosen for their relative simplicity in comparison to the speech recognition: the symbolic representation of speech is only an intermediary step between input signal and its synthesized version (coding), or a pre-processing for a set of scoring systems

(verification). However, this report contains first experimental results of a comparison of ALISP-units with their phonetic counterparts. The use of ALISP $\leftrightarrow$ phoneme mapping is one of possible ways leading to a recognizer based on automatically derived units.

1.1 Scope of chapters

The following Chapter 2 presents a typical application making use of sub-word units: a large vocabulary continuous speech recognizer. It deals with the steps of classical unit determination and criticizes its problematic aspects. As an alternative, the use of ALISP-like units is discussed.

Chapter 3 gives overview of bibliography devoted to segmental methods using data-driven approaches. It is mainly based on speech coding literature, but other domains pertinent for the topic of this thesis are not omitted.

The main theoretic part of this work is Chapter 4. It gives the synopsis of unit determination and concentrates on different "tools": the temporal decomposition, unsupervised clustering, multigrams and Hidden Markov Models. Sections are completed with remarks on the implementation.

The applications tested in experiments are covered by two following Chapters: 5 and 6. The former one deals with very-low bit rate coding. It starts with complements to unit determination specific to the coding. The experiments were carried out in speaker-dependent mode with the data from two databases: PolyVar and Boston University Radio Speech Corpus. Experimental setup, detailed results, and WWW-pointers to speech files are included.

The later chapter presents the application of ALISP techniques to speaker verification. The possible benefit of classification as front-end to scoring is discussed in the beginning of the chapter. Description of experiments conducted within the NIST-NSA'98 evaluation campaign follows.

Chapter 7 compares the ALISP transcription of corpus with the phonetic one. The methodology of such comparison is reported. Experimental results are given for one speaker of the BU database: possible ways leading to the construction of an ALISP-based recognizer are discussed.

Finally, Chapter 8 closes this thesis with the discussion and conclusions.

This work contains four appendices: Appendix A gives technical details on the determination of targets and interpolation functions in the temporal decomposition. Appendix B and Appendix C contain respectively the synopsis of this work in French and in Czech, as it is requested by the CO-TUTELLE convention between University of Paris XI – Orsay and Technical University of Brno. Appendix D includes the most important articles presented at conferences during this thesis. The following and final part of this dissertation is the list of references.

1.2 Original contributions of this thesis

Rather than on extensive solution of theoretical problems, this thesis was based on *integration* of existing tools having the final goal (automatic creation of set of speech units) in mind. The main efforts were therefore turned to the choice of such tools, their implementation, and primarily to their *evaluation in the experimental work*. It should however be noted, that only few of those tools could be used "as is": more or less severe modifications were necessary. So, the main claims of this thesis can be summarized as follows:

- definition and realization of an algorithm capable of automatic determination of speech units using raw speech data.
- experimental use of those units instead of phonemes in very low bit-rate coding based on recognition-synthesis.

- experimental use of those units instead of phonemes in a class-dependent speaker verification system.

- comparison of a segmentation obtained using such units, with the phonetic transcription.

The theoretic contributions in this work can be summarized as follows:

- introduction of *multigrams with distance* with a probabilistic formulation of this approach (in cooperation with Geneviève Baudoin).

- study of *refinement of unit set* using successive segmentations by HMMs and re-estimations of HMM parameters.

- study of *multigrams* cooperating at different levels with *HMMs*.

1.3 Notations, frequent abbreviations and glossary of terms

1.3.1 Notations

a, A	scalar value
$\mathbf{a}$	column vector
$\mathbf{A}$	matrix
$(\cdot)^T$	transpose of matrix or vector
$(\cdot)^H$	hermitian (complex conjugate) transpose of matrix or vector
$\mathrm{tr}(\cdot)$	trace of matrix
$\Re(\cdot)$	real part of complex number
$\Im(\cdot)$	imaginary part of complex number
$\lvert\cdot\rvert$	modulus of complex number
$\lVert\cdot\rVert^2$	2-norm of vector
$\measuredangle(\cdot)$	phase of complex number
$\star$	convolution
$\mathcal{N}$	Gaussian (normal) probability density function (PDF)
$\mathcal{L}$	linear likelihood
$\mathcal{P}$	linear probability. Often, the term "probability" is not justified from the statistical point of view, as we are using $\mathcal{P}$ as a value of PDF for given $\mathbf{x}$: $\mathcal{P}(\mathbf{x}) = \mathcal{N}(\mathbf{x})$.
$\mathcal{F}$	discrete Fourier transform (DFT).
N	length of signal (samples, parameter vectors, or others)
P	size of parameter vector, order of prediction filter
L	size of VQ dictionary
Z	size of coding units dictionary

1.3.2 Frequent abbreviations

ALISP	automatic language independent speech processing
CMS	cepstral mean subtraction
CU	coding unit
DB	database
DFT	discrete Fourier transform
DP	dynamic programming
DTW	dynamic time warping
EHMM	ergodic hidden Markov model
HMM	hidden Markov model
HTK	hidden Markov model toolkit
LBG	Linde-Buzo-Gray algorithm for VQ code-book training
LM	language model
LPC	linear predictive coding (coefficients)
LPCC	linear prediction (derived) cepstral coefficients
LVCSR	large vocabulary continuous speech recognition
MLP	multi-layer perceptron
MG	multigram
MGD	multigram with distance
ML	maximum likelihood
PDF	probability distribution function
SU	synthesis unit
TD	temporal decomposition
VLBR	very low bit rate
VAD	voice activity detector
VQ	vector quantization

A larger and regularly updated version of the list of abbreviations used in speech processing can be consulted at: `http://www.fee.vutbr.cz/~cernocky/FILES/abbreviations.txt`

1.3.3 Glossary of terms

Common term definitions are given according to Websters English dictionary:

phoneme one of a small set of speech sounds that are distinguished by the speakers and listeners of a particular language. Two sounds fall into the same phoneme category if the difference between them never distinguishes two words in the language.

word a unit of language that native speakers can identify.

morpheme minimal meaningful language unit; it cannot be divided into smaller meaningful units.

syllable a unit of spoken language larger than a phoneme.

sentence a string of words satisfying the grammatical rules of a language.

fenone sub-phonemic unit for speech recognition, modeled by 2-state HMM [28].

multone linear combination of fenones using multonic coefficients [7].

senone context-dependent sub-phonemic units for the synthesis, equivalent to one HMM state in a recognition system [50].

ALISP unit speech unit derived using data-driven ALISP approaches.

Chapter 2

Sub-word units: classical versus automatic determination

2.1 Classical approach: example of a speech recognizer

Linguists claim that human language is articulated into multiple levels of elementary units, which are themselves organized in systems. Sentences are made of words, which belong to different syntactic classes, words can be decomposed into morphemes of different kinds, morphemes themselves are articulated into phonemes, characterized by various features... Language thus appears as a complex code, in which several layers of symbolic representations are involved at different levels.

This formalism is then reflected in various applications of the automatic speech processing (ASP): the description of speech is based on an expert analysis of the linguistic material, which requires a certain number of *a priori* hypotheses on the nature of the relevant units. We are going to illustrate this approach on an example of large vocabulary continuous speech recognizer (LVCSR) and point out its problems.

A classical recognizer is based on finding best unit sequence maximizing the probability :

$$\arg\max \mathcal{P}(w_1^N|\mathbf{O}) \tag{2.1}$$

over all possible unit sequences w_1^N. Note, that $\mathbf{O}$ is a string of vector observations $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T]$ of length T. As it is not possible to evaluate this probability directly, one uses Bayes rule:

$$\mathcal{P}(w_1^N|\mathbf{O}) = \frac{\mathcal{P}(\mathbf{O}|w_1^N)\mathcal{P}(w_1^N)}{\mathcal{P}(\mathbf{O})}. \tag{2.2}$$

Here, the probability of data $\mathcal{P}(\mathbf{O})$ is unknown but constant, so that we can concentrate on the numerator of Equation 2.2. The term $\mathcal{P}(\mathbf{O}|w_1^N)$ determines, how plausible is a *production* of observation sequence $\mathbf{O}$ by unit string w_1^N while the term $\mathcal{P}(w_1^N)$ gives the probability a-priori of that string. The production of observations by units is often modeled using Hidden Markov Models trained on acoustic data (with complementary information) while the prior probabilities of word sequences are trained on a text corpus. Both mechanisms are shown on Figure 2.1.

As this work is concentrating on *acoustic units*, we are going to describe their determination and training in a classical recognizer in more detail. The HMMs are based on an *a-priori* chosen set of symbols: phonemes, context-dependent phonemes, syllables, or others. Their number is given by the choice of system developer, and the same holds for their structure (number of states, etc.). To train a model to represent given unit, one must dispose of several *examples* of this unit – so, in case of using phonemes, at least a part of the training database (DB) must be

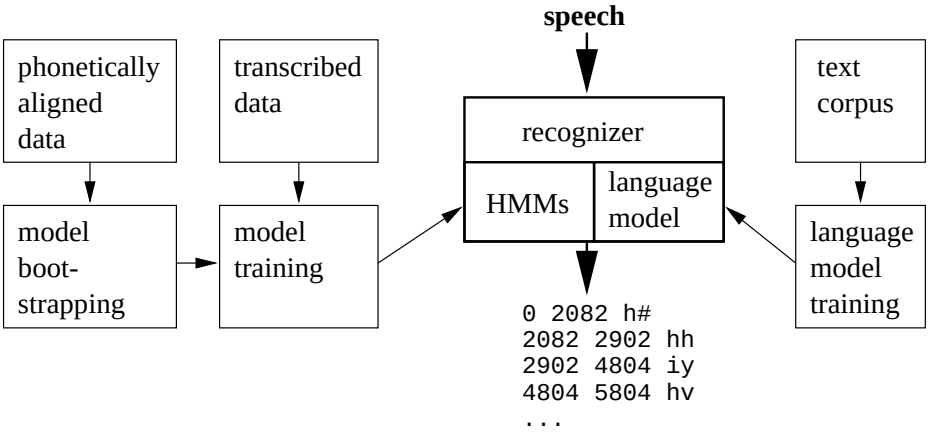


Figure 2.1: Training of a classical large vocabulary speech recognizer.

phonetically aligned. Once the models are "bootstrapped", it is sufficient, that the rest of the DB is only *transcribed* (only a string of units is available for each signal file, without their exact positions).

2.2 Problems of the classical approach

This classical concept of unit derivation suffers from several major drawbacks:

- Current speech technology is based on the massive use of hand-processed data. As this processing requires huge amounts of human efforts, it is very costly. Annotators and "labellers" of databases must be highly qualified, and they are prone to errors, as their work is quite monotonous and tiring. Moreover, if a group of co-workers is engaged, their results can be quite incoherent. Due to economic factors, such databases have been created only for major world languages: their creation in minor ones is slow and there is a serious risk, that for some languages in the world such database will never be created.

  On the other hand, a very large quantity of unlabelled speech data are available and easy to access. These *found* data can be obtained from radio or TV broadcasts, telephone conversations, CD-ROMs, etc. They are available in many languages, many speakers and many audio qualities. The cost for collecting them is negligible compared to the cost that would be needed to collect and annotate specific data. Moreover, if these data are taken from various sources, they are more representative of the general human spoken communication process.

- the *adequacy of phonemes* and their variants (diphones, phone segments) as basic units for speech description is questionable. There is surely an intermediate level between the lexical and acoustic representations of speech, but is has not been proven, that phonemes are the units constituting this level. They are more or less linked to the *textual representation* of speech.[1] While this description is established over centuries to capture the informative contents of speech, its comparison with the acoustic form is not straightforward: let us mention only the different pronunciation variants, or complicated grapheme to phoneme mapping in some languages.

[1]This would let suppose, that each person able to *speak* must be also automatically be able to *read*. In this case, children and analphabets could not talk...

- even if we admit the usefulness of phonetic description in speech *recognition*, where those units constitute an intermediate level between the acoustic form and text, it is not justified in other applications as low rate coding or speaker verification: in the coding, such symbolic description is only an intermediate step between two signals: the input one and the synthesized one. This suggests the set of units should be based on the *signal* rather than to be driven by the text. In the verification, one attempts to divide the signal into classes with different degree of discriminability with regard to the verification task. Here, it is preferable to construct the classes depending on the verification efficiency rather than on a-priori known forms.

2.3 Automatic determination: ALISP

Previous critical remarks have shown a great interest of methods based on the *data* and using as little a-priori knowledge as possible. Those techniques are generically named ALISP (Automatic Language Independent Speech Processing) [20]. In the context of ALISP, a key resource is a large database of speech signal in the target language. From this database, a fully automatic procedure is used to derive a representation, that satisfies a compromise between the accuracy of the description of the acoustic observations and the economy of the representation. When necessary, this intermediate representation has to be mapped to the linguistic level (for instance, words units): only this step may require the use of a set of annotated data and the help of general phonetic knowledge. Moreover, the mapping can be probabilistic rather than deterministic as is the case in conventional approaches. The phonetic representation should only *guide* this mapping rather than be the sole intermediate representation between the acoustic and linguistic levels.

2.4 Evaluation of the set of automatically determined units

In classical approaches, where the set of units is fixed a-priori, it is either impossible or very difficult to evaluate the impact of this choice on the overall system performances. A general technique for evaluation of quality of units does not exist, as well as a method giving their optimal number.

In case of ALISP, such evaluation is possible in the framework of *very low bit-rate coding*. At very low rates of 100–200 bits per second[2] (bps), a symbolic representation of the incoming speech is required. Only the symbols and an auxiliary information is then transmitted from the coder: the decoder performs a synthesis. If the decoded speech is intelligible, one must admit that the symbolic representation is capable of capturing the significant acoustic-phonetic structure of the message. Moreover, it is possible to *evaluate* the quality of the set of units using two measures:

- the *coding rate* in bps and *dictionary size* give an idea of the *efficiency* of speech description.
- the *quality* of decoded speech is related to its *precision*.

Therefore, the coding experiments conducted within this thesis can be seen not only as a search for technical solutions for speech transmission and storage, but also as a verification of general usefulness of tested ALISP approaches in the speech processing.

[2]Those rates are approaching the phonetic rate of speech, which is according to the literature [76, 74] about 50 bps.

Chapter 3

Automatically determined units in the literature

This chapter contains a bibliographic study of data-driven segmental methods throughout various fields of the speech processing. According to the importance given to coding in this thesis, the widest attention has been payed to segmental methods in the speech coding literature. Therefore, attention is payed also to phonetic vocoders, as this concept, with automatically derived units, was used in the experimental part of our work. Described phonetic vocoders are not examples of ALISP-like techniques, on the contrary, we demonstrate on their case the apparent problems of classical approaches.

Other fields, as the recognition using acoustically derived units, are covered by independent sections, too.

3.1 Segmental methods in speech coding

The coding literature contains a lot of methods which we could call "ALISP-like". On the way towards low bit rates while preserving the coded speech quality, the developers were forced to quit the classical frame-by-frame schemes and to use larger segments, represented by an entry in the *codebook*.

On contrary to frame-by-frame based parametric vocoders, the segmental schemes[1] make use of *dependencies* between successive frames. Rather than single parameter vector $\mathbf{x}$ (containing arbitrary set of LPC derived coefficients: LAR, LSF, ...), they work with *matrices* of such parameters:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m] = \begin{bmatrix} x_{1,1} & x_{2,1} & \cdots & x_{m,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,P} & x_{2,P} & \cdots & x_{m,P} \end{bmatrix}. \tag{3.1}$$

Depending on the way of encoding those matrices, we can roughly divide these schemes into three categories:

- *Matrix Quantization (MQ)* is equivalent to vector quantization (VQ) for single vectors. A best matching code-matrix is searched in a codebook and only its index is transmitted.
- in *Multi-Frame Coding (MFC)*, only a few parameter vectors are transmitted and the rest is obtained by interpolation.

[1] A partial bibliographic study about segmental coders in French is included in author's DEA report [94].

- *Temporal Decomposition Coding* attempts to represent the parameters using target vectors and interpolation functions.

The following three subsections deal with this topics in more detail. However, it should be noted, that those approaches are not mutually exclusive and that in one coder, one can easily find two or more of them cooperating. This explains, that one bibliographic reference can appear at multiple places in the following text.

3.1.1 Matrix quantization

The easiest way to represent parameters on segmental level is to build a codebook:

$$C = \{\mathbf{Y}_i\}_{i=1}^{Z}, \tag{3.2}$$

with all code-matrices $\mathbf{Y}_i$ of equal lengths m and to let them represent fixed-length input segments $\mathbf{X}_j$ of the same length m. This approach was tested as one variant by Roucos et al. in [82], but it is reported inefficient, as the characteristic patterns of speech do not have the same lengths. A solution would be to construct multiple codebooks C_q for different lengths $q = 1 \ldots n$ and compare an input segment $\mathbf{X}_j$ of length q only with entries of the corresponding codebook C_q. Even if this approach was adopted by Chou and Lookabaugh in [23] and by Jeanrenaud and Peterson in [54], in the majority of systems the code-matrices are of fixed length and the input segments are *time aligned* against them:

- by the *dynamic programming (DP)* or *Dynamic Time Warping (DTW)* which computes the distance between the two segments along the optimal alignment path [104].
- or the matrices of parameters are *interpolated* and *re-sampled* to the same number of vectors. This approach, including linear interpolation, has been used in [82, 83, 86].

The *determination of segments* in the input speech given a segment codebook is a more difficult task. There are two possible ways to perform this segmentation:

- *open loop* schemes. Here, the segment boundaries are determined using a discontinuity measure of parameter vectors. In [82], a simple time derivative estimations:

  $$\Delta_i(n) = \|\mathbf{x}(n) - \mathbf{x}(n-i)\|^2, \quad i = 1, 3 \tag{3.3}$$

  are suggested and the segments boundaries are placed to stable regions between the transitions. Wong in [104] proposes similar approach based on evaluating *spectral variance measure*:

  $$\sigma_d(n) = \frac{1}{2L} \sum_{j=n-L}^{n+L} d[\mathbf{x}(j), \mathbf{x}(n)], \tag{3.4}$$

  where $d[\cdot, \cdot]$ stands for the distance of two vectors and L is the half-length of interval, on which the variance measure is computed.

- *closed loop* schemes optimize the segmentation with the criterion of minimum overall distortion. In [86], the optimal segmentation B_{opt} of input matrix $\mathbf{X}$ into J segments $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J$ is written as:

  $$B_{opt} = \arg \min_{l_j;\ 1 \le j \le J} \left\{ \min_{\mathbf{Y}_{i(j)} \in C} \sum_{j=1}^{J} d(\mathbf{X}_j, \mathbf{Y}_{i(j)}) \right\}, \tag{3.5}$$

where l_j are the lengths of segments and C is the codebook of Z code-matrices $\mathbf{Y}_i$. The distance $d(\mathbf{X}_j, \mathbf{Y}_{i(j)})$ is already the time warped one. This optimal segmentation can be in practice found using the dynamic programming.

Yet another possibility is not only to minimize the distortion, but to optimize jointly the *distortion and rate*. For the case of single vectors, Chou and Lookabaugh define an equivalent of VQ: the *Entropy Constrained Vector Quantization (ECVQ)* in [24, 21], where in the criterion, the rate is linked to the distortion using a Lagrange multiplier:

$$\min(D + \lambda R). \tag{3.6}$$

In the following works, these authors further develop this approach to *Variable-to-Variable Length Vector Quantization (VVVQ)* [22, 23]. Here, the above mentioned Shiraki's criterion could be rewritten to:

$$B_{opt} = \arg \min_{l_j;\ 1 \le j \le J} \left\{ \min_{\mathbf{Y}_{i(j)} \in C} \sum_{j=1}^{J} \left[d(\mathbf{X}_j, \mathbf{Y}_{i(j)}) + \lambda \gamma(\mathbf{Y}_{i(j)}) \right] \right\}, \tag{3.7}$$

where $\gamma(\mathbf{Y}_{i(j)})$ stands for the number of bits necessary to encode the code-matrix $\mathbf{Y}_i$ quantizing the input segment $\mathbf{X}_j$. This number of bits is determined from the a-priori probability of $\mathbf{Y}_i$ as $\gamma = \log_2 \mathcal{P}(\mathbf{Y}_i)$. Also this optimal segmentation can be in practice obtained using dynamic programming. Similar criterion (distortion plus rate) coupled with *Matrix Product Quantization (MPQ)* is presented by Bruhn in [18].

The design of *segment codebook* is another important issue in this type of coding. Roucos et al. in [82] use *random quantizer* (arbitrarily chosen segments from the training string are memorized as code-matrices), but in the other articles we find uniquely *clustering algorithms*. Generally, such codebook is trained in the same way as VQ ones, using iterated segmentations and segment clusterings. However, the *initialization* of such such codebook is not straightforward as in the VQ case. Chou [23] suggests to initialize with previously trained fixed length MQ codebooks, while the other authors begin with a huge codebook and compact it using a minimax criterion [104], or apply a segmental equivalent of Linde-Buzo-Gray (LBG) algorithm [86].

It has been shown, that the introduction of further *dependencies between segments* using transition probabilities [81] or by creating a network of possible successors to each code-matrix $\mathbf{Y}_i$ [83] further improves the efficiency of coding and eventually limits the necessary search space. These techniques, widely used in recognition, are called *language models*.

3.1.2 Multi-frame coding

In multi-frame coding (MFC), the quantization of spectral parameters is slightly different from the matrix quantization. Here, a set of "important vectors" called *anchor points* is quantized and transmitted to the decoder, the rest of vectors being obtained by *interpolation*. Two modifications of this algorithm exist:

- in the first, the quantization begins with a full matrix of parameters. This matrix is "diluted" by choosing only a subset of vectors constituting a *template*, the non-transmitted vectors being interpolated in the decoder.

- the second approach sets up a "current interpolation contour" using previously transmitted vectors. The current vector is not coded if it corresponds well to this contour (e.g. if it does not cross a distance threshold).

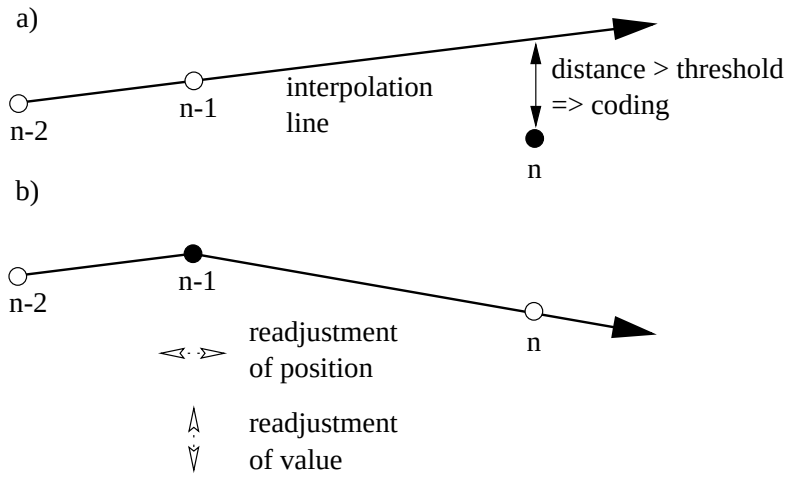


Figure 3.1: Multi-Frame Coding: a) selection of vector to be coded, b) re-adjustment of position and of value of the previous anchor vector.

Copperi in [27] performs a phonetic pre-classification of frames. The critical ones corresponding to probable perceptual cues (changes voiced–unvoiced and vice-versa, onsets for stop consonants, etc.) are always transmitted. The rest of anchor frames are determined as end points of interpolation intervals having a mean square spectral error below a threshold. In addition, the quantizer properties (for example the number of coded coefficients) change depending on the classification of segment, so that this scheme could be named also "phonetic segmentation vocoder".

Lopez-Soler and Farvardin in [59] adopt an algorithm which does not make use of any a-priori knowledge. An interpolation line (see Fig. 3.1) is traced using two previous coded vectors and the current one is not coded unless it is too far (> threshold) from this line. When such an outlier frame is found, it is registered to be coded and the position and value of previous anchor are readjusted in order to minimize the distortion on this segment. The previous anchor is then vector quantized using:

- a classical VQ minimizing the distance of anchor vector to one of code-vectors.
- choosing the code-vector in order to minimize the overall distortion on the segment.

Using the later approach, the segmentation and quantization are linked more closely than in the former one.

Svendsen in [90] presents a closed-loop approach of segment determination. There are no anchor points, but the vector $\mathbf{x}_n$ is represented by the centroid of segment:

$$\mathbf{r}_n = \frac{1}{k}\sum_{i=1}^{k}\mathbf{x}_i, \tag{3.8}$$

or by the first order approximation of the segment:

$$\mathbf{r}_n = n\mathbf{a} + \mathbf{b}, \tag{3.9}$$

where $\mathbf{b}$ is the segment centroid and $\mathbf{a}$ is the first order orthogonal polynomial coefficient vector. The segment boundaries are determined in similar way to Eq. 3.5:

$$D(m) = \min_{\{b_1,b_2,\ldots,b_m\}}\left\{\frac{1}{N}\sum_{k=1}^{m}\sum_{n=b_k}^{b_{k+1}-1} d(n)\right\}, \tag{3.10}$$

where N is the length of original parameter matrix, m is the number of segments, $b_1, b_2, \ldots, b_m$ are the segment boundaries and $d(n)$ stands for the distance of $\mathbf{x}_n$ and $\mathbf{r}_n$. The number of segments m is incremented until the overall distortion $D(m)$ falls below a pre-determined threshold: $D(m) < \Theta$. This article treats also practical considerations of finite-length buffer.

The *Frame-Repeat Vector Quantization* described by Wong in the first part of [104] can be considered as a simple multi-frame coding, too. Here, if the distance of current vector $\mathbf{x}_n$ from previously quantized vector $\hat{\mathbf{x}}_{n-1}$ is lower than a threshold, or the code-vector chosen is the same as that for $\mathbf{x}_{n-1}$, only a repeat flag is transmitted. This corresponds to [59] with a piecewise constant interpolation line, no anchor adjustment and no interpolation in the decoder.

The representative of the "diluting of parameter matrix" is the article [55] by Kemp et al. It presents a multi-frame post-processing to the standard LPC10 algorithm. The parameter vectors are grouped to matrices of length m=8. Out of each matrix, only 4 vectors are chosen for the transmission: always the last one to ensure good interpolation of the following matrix, plus three others selected by minimization of the total spectral error over the matrix.

Another approach falling to the first category of MFC coders was proposed by Mouy, de La Noue and Goudezeune [62] and was standardized as NATO STANAG 4479. This vocoder at 800 bps uses the same front-end processing as LPC10e vocoder. Out of 3 consecutive frames, a *super-frame* is constructed. In the spectral envelope coding, the vector of 10 LAR coefficients is first projected on 10 main axes obtained using eigen-value analysis. It is argued, that in case of stable envelope, the average spectrum should be well quantized. In case of unstable sound, the variations should be emphasized. Therefore, one or two filters are quantized using 32, 24 or 16 bits (if one is quantized using 24 bits, the additional 8 ones are used to encode differentially the second filter). Similar super-frame quantization is applied also to the energy and pitch/voicing.

3.1.3 Temporal decomposition based coding

The *temporal decomposition (TD)* attempts to decompose a matrix of spectral parameters $\mathbf{X}$ of length N into a limited number of *events*, each consisting of a *target vector* $\mathbf{a}_k$ with associated *interpolation function (IF)* $\phi_k(n)$, compact in time[2]. The number of events m is inferior to the original number of frames: $m < N$ and the parameter matrix is approximated by the sum of target vectors "spread" in time by interpolation functions:

$$\hat{x}_i(n) = \sum_{k=1}^{m} a_{ik}\phi_k(n), \tag{3.11}$$

where $\hat{x}_i(n)$ is approximation of the trajectory of one spectral coefficient and a_{ik} is the i-th component of k-th target vector. This equation can be written in more compact form using matrix notation:

$$\hat{\mathbf{X}} = \mathbf{A}\mathbf{\Phi}, \tag{3.12}$$

where $\hat{\mathbf{X}}$ is $P \times N$ matrix of approximated parameters, $\mathbf{A}$ is $P \times m$ matrix of target vectors (in its columns) and $\mathbf{\Phi}$ is $m \times N$ matrix of interpolation functions (in matrix lines).

The methods of estimating matrices $\mathbf{A}$ and $\mathbf{\Phi}$ will be discussed in detail in Section 4.4 and in Appendix A, here we are going to comment the use of TD directly in the coding, inducing the need of efficient representation of events. On contrary to other segmental schemes, in TD based coders, the information from neighboring segments is intrinsically merged to form a smooth transition, thus alleviating some transition problems of other schemes.

[2]Theoretically, this approach is justified by one of explanations of the articulatory process: the vocal tract receives *commands* to attain *articulatory targets* but due to its inertia, another command can come before the desired target is reached. The *coarticulation* may be easily explained in this way.

Atal in the original TD article [5] determines the bandwidth of interpolation functions $\phi_k(n)$ and finds it lower than that of original parameter trajectories. In this case, it is possible to down-sample functions $\phi_k(n)$ at the average of 4 samples per IF. The parameter vectors are quantized using standard scalar quantization.

Ghaemmaghami et al. in more recent articles [41, 42, 43] concentrate mainly on the approximation of IFs and on the choice of parameters. In their article published at ICASSP'96 [41], they approximate the IFs using rectangles or Gaussians. They find, that the global distance of coded vectors from the original ones decreases, when the later approximation is used. This is certainly linked to problems on transitions when using rectangles (in this case, one completely neglects the coarticulation). This article contains two another interesting points: the comparison of TD segment determination with the steadiness analysis of parameters using the formula:

$$d(n) = \sum_{j=n-\frac{l-1}{2}}^{n+\frac{l-1}{2}} \exp\left\{ -\left[\sum_{i=1}^{10} |g_i(n) - g_i(j)|^4 \right]^{\frac{1}{4}} \right\} - 1, \tag{3.13}$$

where l is the length of steadiness analysis window and $g_i(n)$ are the LAR parameters. The steady parts of speech are found by detecting valleys of $d(n)$ over the utterance of N frames. The authors find that both methods (TD and steadiness analysis) provide a similar segmentation. The second interesting point is the coding of pitch and energy using the IFs of temporal decomposition.

The ICASSP'97 paper [42] studies the usefulness of 7 different parameter sets (LPC, reflection coefficients, A – areas (in the tube model), LA – log areas, LAR – log area ratios, cepstra and filter-bank outputs) in the TD coding. The comparison of different parameters has been treated already in Van Dijk's article [93], which concentrated rather on the adequacy of IFs with phonemes and sub-phonemes. Here, the suitability is evaluated using three measures: the *parametric distance*, defined as Euclidean distance of LAR parameters, the *Bark-scaled filter-bank distance*:

$$d_s(n) = \sum_{k=1}^{15} |P_k(n) - \hat{P}_k(n)|, \tag{3.14}$$

where $P_k(n)$ denotes the power in k-th band for original frame number n, and $\hat{P}_k(n)$ this power of the synthetic signal. The third measure is the *energy distance*:

$$d_e(n) = |\log E(n) - \log \hat{E}(n)|, \tag{3.15}$$

where $E(n)$ denotes the relative energy, normalized to the peak energy of the utterance. These distances were evaluated not only globally, but also separately for voiced and unvoiced parts and their transitions. Furthermore, two sets of parameters were used in one experimental setup: the first for *event detection*, the second for the *synthesis*. The results show, that the most efficient combination are the LPCCs for detection and LARs for synthesis (certainly also due to their inherent stability).

The last work of Ghaemmaghami et al. presented at Eurospeech'97 [43] brings new approximation of IFs. The authors define the *width*, *curving* and *sharpening* parameters of an approximation function. The first one defines the number of frames under the corresponding IF, the second denotes the weight of parameters near the center of IF and the third stands for the concentration on the event centroid. The approximation function is obtained as a linear combination of a triangular function and a parametric function:

$$u_1(n) = \begin{cases} \dfrac{(1+m)a}{(n-n_c)^2 + a} & \text{for} \quad |n - n_c| < \dfrac{N-1}{2} \\ 0 & \text{elsewhere} \end{cases}, \tag{3.16}$$

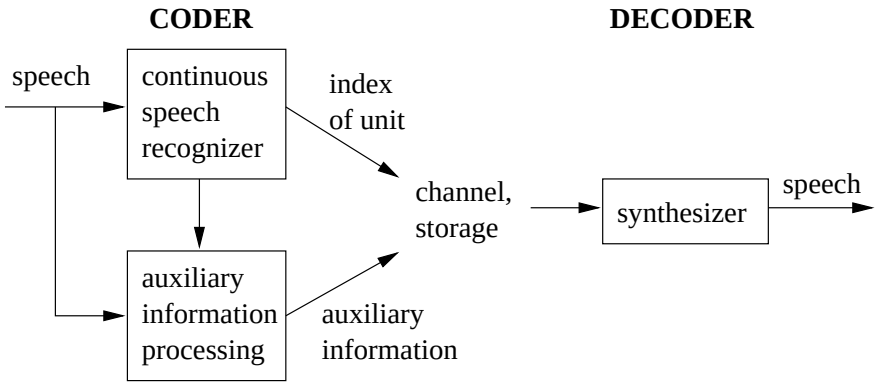


Figure 3.2: Synopsis of a phonetic vocoder.

with

$$m = \frac{4a}{(N-1)^2}, \tag{3.17}$$

where a is the curving parameter, N is the length of segment and n_c is the index of segment's central frame. The authors report the improvement of distortion in comparison with Gaussian approximation, again with LPCC–LAR as the best parameter set couple. Strangely, there is no possibility to make the IF approximation *asymmetric*, which, in our opinion, would lead to further increase of precision of the approximation.

3.2 Recognition – synthesis based methods

The schemes mentioned in this section have a lot of common with segmental coders of the previous one, but we attempted to separate the coders using standard recognizers at the input from the methods working directly with matrices of spectral parameters. However, these approaches are not heterogeneous and the concepts may be overlapping.

The front-end of *phonetic vocoders* (another name for recognition–synthesis based methods) is a continuous speech recognizer producing a segmentation and labelling input speech by *phonemes* or larger units. Only the index of the unit in dictionary, together with an auxiliary information is transmitted (see Fig 3.2). In the decoder, a synthesis takes place to reconstruct the speech.

The standard framework for recognizers in phonetic vocoders are the Hidden Markov Models (HMM)[3]. The auxiliary information includes, as it is the case for previously mentioned segmental schemes, the timing, pitch and energy, and optionally the alignment path between memorized speech unit and the input one. The synthesis can be performed either parametrically (for example using an LPC synthesizer) or in the time domain.

Picone and Doddington in the first article about phonetic vocoding known to us [73], state, that HMM systems are able to closely model the acoustic data (better in comparison with previously used DTW) and that the errors in speech recognition are often due to wrong symbolic assignment rather than to poor acoustic match. They built a vocoder with 60 phonemes of the TIMIT database [39] with left-right HMMs. For each phoneme, the HMM dictionary contains 2,4 or 8 models. As the models are left-right, the timing information is transmitted in the form of flags saying "stay in state" or "next state". With 8 models per phoneme, they report intelligible speech at the mean rate of 170 bps. Two interesting points are suggested:

[3]The article of Roucos et al. [81], which can be considered as a predecessor of those schemes, contains a matrix quantization, but also quite sophisticated language model (LM) to take into account the dependencies between segments.

- adding of several *larger units* to the recognition system should improve the quality with a minimal increase in complexity.
- the use of *grammar models* hypothesizing the phonemes should improve the recognition accuracy and decrease the bit rate. The authors report the rate decrease of 13 % when a phone-bigram grammar was used and 22 % decrease in case of phone-trigrams, but due to the computational complexity, this approach was not pursued.

Ismail and Ponting in [51] report results of phonetic vocoding at the average rate of 300 bps. They use the *acoustic segments* as the basic unit for recognition and a rule-driven formant synthesizer in the decoder. The set of acoustic segments is defined as an extension of SAMPA phoneme dictionary [4], where the *plosives* are expanded into several segments: closure, burst and aspiration for unvoiced, closure and burst for voiced. All expansions are subject to context. Out of 79 acoustic segments defined by Holmes et al. in [49], for which rules to drive a formant synthesizer have been derived, Ismail takes a subset of 64 segments, completed by diphthongs. Conventional left-right HMMs are used to model the acoustic segments. For plosive portions, only a single state is allocated to each segment to allow plosives be shorter (if standard 3-state HMM was used for each segment, a minimal duration required for a plosive would be too long: 90 ms[4]). The pitch and durations are sent to the decoder as side information. The scheme was tested in speaker dependent and task dependent condition with a limited vocabulary of 500 words. The best recognition performance in terms of segment errors was 3.8 % (7.4 % word errors), and the speech quality reported is better than that for a text-to-speech (TTS) system, working with the same text, as for which the original utterance was created. As important issues, the authors cite the *automatic derivation of rules* for the synthesizer from natural language and automatic determination of *speaker characteristics*.

The phonetic vocoding with *speaker adaptation* is described in two articles of Ribeiro and Trancoso [78, 79]. It is argued, that speaker recognizability is important even in VLBR schemes, and that a speaker modification module should be included. The first article [78] introduces two methods of speaker and voice adaptation, both based on normalization of parameters for *voiced parts* of speech. In the first, the radius and angle of z-transfer function poles (corresponding to formants) are normalized using means and standard deviations:

$$\begin{aligned} r' &= \frac{r - \bar{r}}{\sigma_r} \\ \theta' &= \frac{\theta - \bar{\theta}}{\sigma_\theta}, \end{aligned} \tag{3.18}$$

where r is the radius of pole, θ is the angle and $\bar{r}, \sigma_r, \bar{\theta}, \sigma_\theta$ are statistics of vowel classes of the speaker. These statistics are transmitted, and in the decoder, a de-normalization takes place:

$$\begin{aligned} r_s &= r'_s \sigma_{rs} + \bar{r}_s \\ \theta_s &= \theta'_s \sigma_{\theta s} + \bar{\theta}_s, \end{aligned} \tag{3.19}$$

where the subscript s stands for values appearing in the synthesizer. $\bar{r}_s, \sigma_{rs}, \bar{\theta}_s, \sigma_{\theta s}$ are the quantized statistics for each vowel class and r'_s, θ'_s are obtained from the synthesizer's phoneme dictionary. In the second approach, the LSP parameters are used instead of pole parameters: the authors argue, that the roots of $M(z)$ (first polynomial used to determine LSP's, see for example [59]) correspond to the locations of formant frequencies while the roots of $Q(z)$ (the second polynomial), called "difference coefficients", give their bandwidths. Similar equations

[4]Ismail uses centisecond vectors.

as 3.18 and 3.19 are used for the normalization and de-normalization of line frequencies. The experiments were done using TIMIT phones, completed with some Portuguese ones, and with the EUROM.1_P corpus. The quality of synthetic speech was found slightly inferior to DoD FS 1015 standard LPC-10 [92] operating at 2400 bps.

The following work of Ribeiro and Trancoso [79] brings fully quantized version of the phonetic vocoder. For vowels, one normalized vowel is stored in the dictionary and de-normalization and time warping to the correct duration take place in the decoder. Considerable bit rate saving with only a little speech degradation is reported if only the means of LPSs are transmitted and the variances are kept constant. The non-vowel phones are not adapted. The experimental results confirmed, that even in presence of important recognition errors (32 % for context dependent phones), the synthetic speech quality is not seriously degraded in comparison with the case, where boundaries and labels of phonemes were obtained manually. The resulting bit rates obtained by the authors are summarized in Table 3.1.

The recent article of Tokuda et al. [91] deals with HMM recognition but also with the *synthesis* applied to VLBR coding. The recognition portion works with 3-state left-to-right HMMs trained on Japanese triphones. The set of triphone models shares 1800 tied distributions. The indices of HMMs are transmitted to the decoder using entropy encoding and the synthesis itself is done by *generating* MFCC vectors using HMMs. In case Δ and $\Delta\Delta$ coefficients were not included in feature vectors, the vector sequence maximizing the likelihood would be simply the concatenation of means of states. In case dynamic features:

$$\Delta c_t = \sum_{\tau=-L_1}^{L_1} w_1(\tau) c_{t+\tau} \tag{3.20}$$

and

$$\Delta\Delta c_t = \sum_{\tau=-L_2}^{L_2} w_2(\tau) c_{t+\tau} \tag{3.21}$$

are used, the optimal observation sequence can be found by solving a set of linear equations:

$$\frac{\delta \log \mathcal{P}(\mathbf{O}|Q,\lambda)}{\delta \mathbf{c}} = \mathbf{0}, \tag{3.22}$$

where $\mathbf{c} = [c_1, c_2, \ldots, c_P]^T$ is the cepstral coefficients vector, $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_N]$ is the sequence of observations, λ are HMM parameters and $Q = \{q_1, q_2, \ldots, q_N\}$ is the sequence of HMM states obtained as a concatenation of all HMMs belonging to the coded utterance. The authors report that this system of equations can be easily solved by a fast algorithm. When using phoneme bigrams as language model, the entropy coding rate of models is as low as 46 bps. An important point of this article is the handling of timing. The experiments were conducted for one male speaker from the ATR Japanese DB and 8 subjects were asked to compare the quality of presented scheme with a conventional VQ spectrum representation at 400 bps (50 frames per second à 8 bits). The results were comparable, when precise transmission of state durations was ensured, verifying the importance of timing in this type of schemes.

3.2.1 Problems of existing recognition – synthesis based schemes

On contrary to the schemes based uniquely on the signal, the phonetic vocoders need a *transcription* of the training database to be able to train models for phonemes, phone-segments or larger units. The optimal transcription is *phonetic labelling*, where the exact position and phoneme label are known. Widely used toolkits, (for example HTK [106]) read phonetic labelling from special files – see Fig. 3.3 for an example). The creation of such phonetic labelling is a tedious

Parameter	avg. rate [bps]
phone	91
phone duration	84
LSP average values	200
energy	236
voicing and pitch	226
TOTAL	840

Table 3.1: Average bit rates in the segmental coder with speaker adaptation of Ribeiro and Trancoso [79].

```
0       3050    h#
3050    4559    sh
4559    5723    ix
5723    6642    hv
6642    8772    eh
8772    9190    dcl
9190    10337   jh
10337   11517   ih
11517   12500   dcl
12500   12640   d
12640   14714   ah
14714   15870   kcl
15870   16334   k
...
```

Figure 3.3: Example of file with phonetic labelling: NIST-TIMIT format. The times are given in samples.

task and not many databases available are entirely labelled. More frequently, a part of the DB is labelled, the phoneme models are *bootstrapped* using this portion and in the rest of DB, where only a *transcription* is at our disposal, those models are aligned with the signal using Viterbi algorithm to produce the labelling. To summarize the drawbacks of phonemes for VLBR coding, one can mention the following points:

- time and money consuming hand labelling of DB.
- if automatic labelling methods are used, the corrections should be mostly done by hand.
- in case of bootstrapping, the alignment of the rest of corpus is never 100 % correct.
- the phones and their different modifications (context dependent phones, phone segments, etc.) have proven their efficiency in recognition, but their use as units in VLBR coding, where the symbolic transcription is only the intermediate level, is disputable.

Therefore, as it was already stated in Chapter 2, our aim is to work with units emerging directly from the speech data. Some approaches known not in the coding but used in other speech disciplines, are discussed in the following sections.

3.3 Automatic segmentation of speech

According to Sharma and Mammone [63], the applications of automatic speech segmentation include the speaker verification (sub-word segmentation of user defined passwords), the recognition (building of set of sub-word units for medium size vocabulary systems), the language identification (to obtain sub-word segmentation of multi-lingual corpora) and speech corpus creation (to obtain a coarse segmentation before phonetic labelling). From our point of view, the VLBR speech coding is one of candidate applications of automatic segmentation, too.

Svendsen in [89] defines three approaches to the automatic segmentation:

1. *Template matching*, where the templates for sub-word units and the transcription of utterance are known. The problem reduces to finding the optimal positions of templates (minimizing the distortion using the dynamic programming).

2. *Spectral transition measure based segmentation.* Here, the variations of spectrum are detected using a function and a peak-picking procedure is applied. For Svendsen, the *spectral variation function* is given as the approximation of short time spectrum variation using cepstral coefficients:
$$\frac{\delta \log |S(\omega,t)|}{\delta t} = \sum_{m=-\infty}^{\infty} \frac{\delta c_m(t)}{\delta t} e^{-j\omega m}, \tag{3.23}$$
where the derivatives of cepstra are estimated using the first order difference of the orthogonal polynomial fit to the cepstral coefficient trajectory (the fit is used instead of the original trajectory to prevent noisy estimates).

3. *Constrained clustering vector quantization.* Here, rather than detecting peaks of a variation function, a segmentation into quasi-stationary segments is searched. The centroids are dynamically defined and the segments are checked to maximize the overall likelihood of the utterance. The relation of frame probability and Itakura-Saito distance is used, and the likelihood is approximated as:
$$\mathcal{L} = \prod_{i=0}^{m-1} \prod_{n=b_i+1}^{b_{i+1}} exp\left[-\frac{\mathbf{a}_i^T \mathbf{R}_n \mathbf{a}_i}{2\sigma_n^2} \right], \tag{3.24}$$
where m is the number of segments, $\{b_0, b_1, \ldots, b_m\}$ are segment boundaries, $\mathbf{R}_n$ is the covariance matrix of n-th frame, and $\mathbf{a}_i$ is the vector of prediction coefficients for the centroid of i-th segment. The boundaries can be found by minimizing the likelihood distortion ratio $-\log(\mathcal{L})$ over the utterance. A level-building dynamic programming (LBDP) procedure is presented to find this minimum.

Sharma and Mammone in [63] use also LBDP to find the optimal segmentation, but they concentrate on the determination of optimal *number of segments* in the utterance. The initial estimate of minimal number of segments K_{min} is given by the convex-hull method, using the subjective loudness function (temporally smoothed log-energy of speech frames) to find the number of syllabic units. It is argued, that the segments searched are shorter than syllables so that their number is higher. The upper limit K_{max} is given by the number of peaks of spectral variation function, defined here as the 2-norm of delta-cepstral coefficients vector. Having K_{min} and K_{max}, a LBDP algorithm determining automatically segment centroids as means of associated vectors is run for all K's in the interval $[K_{min}, K_{max}]$. In the end, the *Normal Decomposition* is performed for all K's to find the optimal number of segments. This method maximizes the log

likelihood criterion:

$$Q_K = \sum_{j=1}^{N} \ln \mathcal{P}(\mathbf{x}_j), \tag{3.25}$$

where $\mathbf{x}_j$ are parameter vectors and N is the length of utterance. The probability distribution $\mathcal{P}(\cdot)$ is given as a sum of Gaussians:

$$\mathcal{P}(\mathbf{x}) = \sum_{i=1}^{K} P_i \mathcal{N}(\mathbf{x}, \mathbf{M}_i, \mathbf{\Sigma}_i), \tag{3.26}$$

where P_i are weights and $\mathbf{M}_i$ and $\mathbf{\Sigma}_i$ are respectively mean vectors and covariance matrices of Gaussians. These can be obtained using standard ML estimation formulae, but they are *initialized* by vectors from i-th segment. It is found, that when K increases, Q_K tends also to increase, but then it reaches a flat plateau, where the optimal number of segments K_0 is situated.

Beet and Baghai-Ravary in [10] define *Multi-step Adaptive Flux Interpolation (MAFI)* method as one of possible candidates to reduce the dimensionality of recognition task. The method aims to remove redundancy in speech spectra by omitting frames which can be accurately reconstructed from the retained ones. The method is thus comparable to Multi-Frame Coders described in subsection 3.1.2. Although this work is preliminary, the advantage of MAFI seems to be the ability to take into account the dependencies *between* coefficients of spectral vectors by means of *links* intercepting two known frames. These links need not to be only horizontal (e.g., linking coefficients with the same index), but can be also inclined: the conditions are, that the link must not "quit" the space between two known vectors, and that unrealistically rapid spectral transitions can not be modeled. The set of optimal non-crossing links is then obtained by dynamic programming. It would be interesting to compare this method with the Temporal Decomposition (section 4.4 and Appendix A), where the dependencies between coefficients are taken into account when the singular value decomposition of parameter matrix is performed.

Finally, Bonafonte et al. [16] investigate *segment refinement* method for the segmentation of labelled speech. The hypothesis is quite simple: speech frames of a phone should be more similar to that phone, than to context (neighboring) ones. However, this hypothesis is applicable only to some phonetic classes (vowels rather than plosives). The initial segmentation is obtained using Viterbi alignment with a sequence of pre-trained HMMs (the sequence is derived from the phonetic transcription). For each segment, an independent new model is estimated. The refinement then consists of iterative moving of segment boundaries, models re-estimations and overall likelihood re-evaluations. If the likelihood increases, the boundary is fixed in the new position. The HMM covariance matrices $\mathbf{\Sigma}$ have to be simplified as very few training data are available for their estimations (from one segment only). The authors worked with diagonal covariance matrix, constant covariance matrix estimated on the whole utterance, constant diagonal matrix and finally with simple identity matrix in place of $\mathbf{\Sigma}$.

3.4 Recognition with acoustically derived units

Two research groups attempt to fight against the drawbacks of phone based systems in the recognition: the IBM group working on *fenonic models* instead of phone-based HMMs and the Boston University and ATR groups combining the advantages of *segmental modeling* [5] and acoustic derivation of units.

[5] *Segmental modeling* should not be confused with *acoustic segments* used by Ponting and Ismail in [51]. The segmental modeling attempts to introduce the time dependency in states of HMMs by modeling the *trajectories* of parameters, while the acoustic segments are sub-phone units derived from SAMPA phonetic dictionary by splitting the plosives.

Das, in his article about the IBM Tangora dictation system [28], summarizes the practical issues of large vocabulary isolated word recognition system. From our point of view, the acoustic modeling is of greatest importance. This system uses *fenones* modeled by simple two-state Markov models (Fig 3.4a). There are three transition probabilities: p_s denoting the probability of looping in the state, p_f meaning "going on" and p_n which gives a possibility to skip a fenone. With p_s and p_f, discrete emission probabilities are associated, represented by vectors $\mathbf{s}$ and $\mathbf{f}$. Each element of those vectors is related to one code-word of VQ dictionary. No emission probabilities are associated with p_n. The fenone models are trained with no supervision, using only the VQ labelled parameters vectors[6]. A fenonic "pronunciation" of a word is then found as a fenone sequence $\hat{u}$ maximizing the probability of a set of different utterances of this word:

$$\hat{u} = \arg\max_{\{u\}} p(Y^{(1)}, Y^{(2)}, \ldots, Y^{(n)}|u), \tag{3.27}$$

where $Y^{(1)}, Y^{(2)}, \ldots, Y^{(n)}$ denote n different utterances of word Y and $\{u\}$ stands for all possible fenone strings. Assuming the independence of $Y^{(1)}, Y^{(2)}, \ldots, Y^{(n)}$, the joint probability can be replaced by a product.

Bahl et al. in [7] bring extension of fenonic models called *multones*. Instead of "hard" fenonic pronunciation of a word, the pronunciation variants can be taken into account using linear combination of fenones using *multonic coefficients* a_i. Such a parallel combination of fenones in a compact form is depicted on Fig. 3.4b. The estimation of multone parameters is done in four steps:

1. initial estimation of fenones using VQ labelled speech.
2. setting up multonic models for each word, determination of number of multones.
3. estimation of multonic coefficients a_i when keeping fenone parameters (transition and emission probabilities) unchanged.
4. pruning of multones. To avoid high computational load, the number of arcs in multones is reduced to a predefined number, or only those, for which a_i is sufficiently high, are retained.

As Bahl remarks, on contrary to fenones, where finding of optimal "pronunciation" is a discrete optimization problem, in case of multones it is a continuous one (optimal real-valued weighting coefficients are searched) .

The works of ATR and BU groups on the recognition with acoustically derived units are described in [37] and [6]. The former article of Fukada concentrates mainly on the adequacy of automatically derived units with the phonetic representation and on the lexical mapping. We will treat in more detail Bacchiani's article [6], giving the overview of unit set design. The units are based on *polynomial trajectory models* relaxing to some extent the frame independence supposition of classical HMMs[7]. The mean of a segment (represented by one state of HMM) is not supposed constant, but following a trajectory. This trajectory is for i-th segment given by a polynomial coefficient matrix $\mathbf{B}_i$ (of dimensions $P \times (R+1)$, where P is the parameter vector size and R is the polynomial order) and a time sampling matrix $\mathbf{Z}_{N_i}$ of dimensions $(R+1) \times N_i$, where N_i is the segment length. This matrix is filled with the powers of normalized time in its columns:

$$\mathbf{z}_{N_i,j} = \left[1, t_j, t_j^2, \ldots, t_j^R\right]^T, \tag{3.28}$$

[6]The feature extraction of Tangora is quite sophisticated, making use of different noise condition adaptations and deriving a generalized form of delta coefficients by maximization of the discriminative power of vectors in the recognition.

[7]For a tutorial on segmental models, see Ostendorf's excellent work [69].

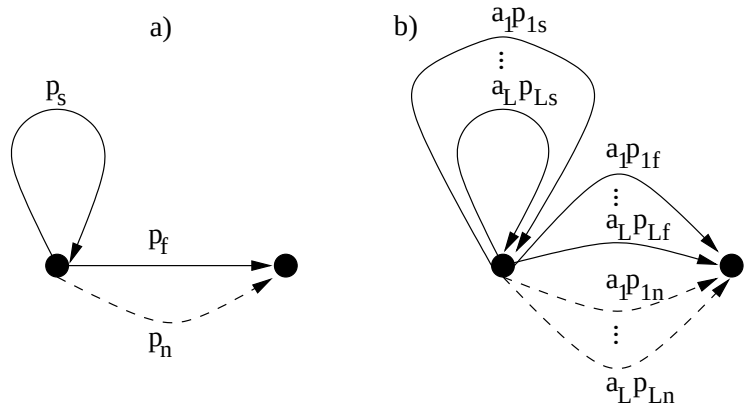


Figure 3.4: Two classes of fenonic models: a) Markov model for a fenone, b) Compact Markov topology of a multone.

where $t_j = (j-1)/(N_i - 1)$ and $j = 1, \ldots, N_i$. The unit design consists of the following steps:

1. *acoustic segmentation* as the initialization of the procedure. The segmentation is done with an assumption that segments are coherent in sense of polynomial modeling (piecewise constant for R=0, piecewise linear for R=1, etc.). Only one covariance matrix is estimated per utterance and a segmental model model is estimated for each segment, as in Bonafonte's segment refinement [16]. During iterations of this process, it was observed, that the likelihood increased monotonously. To prevent an utterance from being split into too many segments, a threshold is imposed on the average likelihood.

2. With segments from the acoustic segmentation, a *clustering* is done as the following step. The maximum likelihood approach is implemented as a minimization of *multivariate Gaussian distance measure* between original regression matrix $\mathbf{B}_s$ of segment and a cluster mean matrix $\mathbf{B}_c$:

$$N \log(|\mathbf{\Sigma}|) + \mathrm{tr}\{N\mathbf{S}\mathbf{\Sigma}^{-1}\} + (\mathbf{B}_s\mathbf{Z}_N - \mathbf{B}_c\mathbf{Z}_N)^T\mathbf{\Sigma}^{-1}(\mathbf{B}_s\mathbf{Z}_N - \mathbf{B}_c\mathbf{Z}_N), \tag{3.29}$$

   where N is the length of segment, $\mathbf{Z}_N$ is the time sampling matrix, $\mathbf{S}$ is the sample covariance matrix of segment and $\mathbf{\Sigma}$ is the covariance of cluster. With this distance, a clustering algorithm is run, beginning by assigning all segments to one cluster, then dividing this cluster until a desired number C is reached. The clusters resulting from this divisive procedure are then refined using a K-means algorithm.

3. *iterative re-estimation* of segment inventory. In this step, the corpus is iteratively re-segmented using the set of C models and the model parameters are re estimated. In the Viterbi segmentation, a length model and bigram "language" model (giving a probability of segment i following segment j) are used. The log-likelihood of observations is computed in a similar way to Eq. 3.29.

This article contains another interesting points concerning the lexicon building and pronunciation dictionary modeling, but they are beyond the scope of this chapter.

Yet another approach to the recognition using segmental models can be found in the work of Flammia [36]. Here, the sequence of observation vectors: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ is divided into segments using spectral variation function, given by averaging the differences of vectors and a moving average:

$$\mathbf{d}_k = \mathbf{x}_k - \mathbf{a}_k, \tag{3.30}$$

where the moving average is computed using a window of $2L + 1$ vectors:

$$\mathbf{a}_t = \frac{1}{2L+1} \sum_{k=0}^{2L} \mathbf{x}_{t-L+k}. \tag{3.31}$$

The spectral variation function is then defined as:

$$SVF(t) = \frac{1}{2} \left[1 - \frac{1}{L^2} \sum_{i,j} \cos(i,j) \right] \quad \text{for} \;\; t - L \leq i < t, \;\; t < j \leq t + L, \tag{3.32}$$

where the cosine is given by normalized scalar product of two difference vectors:

$$\cos(i,j) = \frac{\mathbf{d}_i \mathbf{d}_j}{\|\mathbf{d}_i\| \, \|\mathbf{d}_j\|}. \tag{3.33}$$

This SVF has low values for stable segments and presents peaks for segment transitions. For the recognition, the found segments are down-sampled to limit the complexity of the Viterbi algorithm. The vectors of one segment are selected in order to be maximally decorrelated from the frames of neighboring segments and within the segment itself. The authors report a reduction of 50 % in the number of observation vectors with no performance deterioration compared to centisecond vector based system.

3.5 Determination of the rate of speech

The *Rate of Speech (ROS)* determination is one of methods to compensate for elocution speed variations in HMM based systems. Verhasselt and Martens [103] estimate the ROS by accumulating phone boundary evidences in an interval and dividing this number by the length of interval[8]. Those evidences are detected using a Multi-Layer Perceptron (MLP), trained to produce, for each hypothesized boundary, an a-posteriori probability, that it is a phonetic segment boundary. The input of MLP are auditory spectral coefficients together with functions measuring spectral and energy changes. During the training, boundary hypothesis are presented to the MLP each 10 ms together with the correct output obtained from hand-labelled database. The MLP used was quite small, containing 11 hidden nodes and 50 input ones. The authors incorporate the ROS estimator into their MLP based recognition system as a mean to switch between perceptrons trained on fast or slow utterances (named "ROS-specific") or to weight their scores. As the result, they report the estimated ROS to be very closed to the value obtained by manual segmentation, but only a small improvement in recognition performances (35.9 % phone error rate with ROS correction versus 36.6 % without correction).

3.6 Text-to-speech synthesis

Even if the majority of Text-to-Speech (TTS) synthesis systems are based on phonemes or diphones, some works, concerning namely the organization of unit dictionary, transition handling and prosody representation, are interesting also for the ALISP framework.

Sagisaka in [85] presents an interesting alternative to synthesizer with a dictionary of independent units. Here, the *synthesis unit entry (SUE)* dictionary is proposed, where all phoneme *substrings* in a synthesis unit can be used themselves as synthesis units in the system. When

[8] Verhasselt uses *unnormalized* phone rate, given by the average number of phones per second. The normalized one is given by normalizing the phone durations by phone specific *expected* durations.

the dictionary contains for example the word "nagasa", one can use units "g", "ga", "gas", "gasa", etc. The units are ordered in a tree according to their phoneme length, so that a rapid generation of entry lattice for input phoneme string is possible. The entry lattice is generated by automatically choosing the longest right-matching unit for each phoneme on the input. The final choice of units is then done using the following criteria:

1. consonant-vowel transitions (C-V) should be preserved due to great risk of disfluency, if such transition were concatenated.
2. conservation of vowel successions.
3. preference of longest possible units (this makes the number of concatenations small).
4. template overlap. Choosing unit from a larger one, where its left and right contexts are equivalent to neighboring units, is preferred and leads to minimizing discontinuities in the synthetic speech.

In case of a system with automatically derived units, the notion of phones would be changed, but the idea of organization of the dictionary minimizing the number of transitions would have the same positive effects as in a Sagisaka's TTS.

The article of Huang et al. [50] on Microsoft TTS "Whistler" is interesting from the point of view of choice of synthesis units. This system uses *senones*, defined as context-dependent sub-phonemic units, equivalent to one HMM state in a recognition system (Microsoft "Whisper" in this case). The context decision trees are generated automatically from the training data in order to obtain minimum intra-unit distortion. To select sets of synthesis units for each senone (called here *instances*), first the statistics of units in terms of gain, pitch and duration are made and the outliers are removed. Then, for each senone, several instances maximizing the objective function (HMM likelihood) are chosen. In the phase of synthesis, the instances minimizing the distortions on junctures are concatenated. Moreover, to prevent too severe prosody (pitch and duration) modification, the instance with these values closed to desired prosody can be selected. Another feature of senones is the easiness to concatenate them in order to create longer units useful for difficult contexts (for example V-V concatenation) or frequently repeating triphones or words. Similarly as in Sagisaka's work [85], the individual senones in longer units can still be used independently.

Prosody handling is another point where the work done in synthesis may be useful for ALISP-based VLBR coding. Most issues are related to adequacy of prosody with the text, but an interesting point is the representation of prosody patterns. Nakai et al. in [64] introduce superpositional modeling of pitch using phrase commands superposed with accent commands. The pitch is derived from commands using Fujisaki's model.

More interesting for application in VLBR coding are the articles of Grigoriu et al. [45] and López-Gonzalo et al. [58]. In the former, related more to recognition improvement using prosody than to synthesis, the segment pitch is converted to linear segments and the pitch patterns in two neighboring segments is represented by three parameters:

`basic_type`	`slope`	`pattern_type`

where `basic_type` stands for basic contour type: 1–fall, 2–rise, 3–level, 4–fall-rise and 5–rise-fall, `slope` gives the relative slope of pitch change (1 to 3) and `pattern_type` identifies the overall pattern type with similar options as `basic_type`, plus 6–fall-level, 7–rise-level, 8–level-rise and 9–level-fall. In [58], the prosodic contour of a syllable is defined using 5 parameters: two duration values and three pitch values shown on Figure 3.5. These parameters form a prosodic syllabic pattern (PSP). The authors report that such patterns may be successfully clusters to only 64 classes.

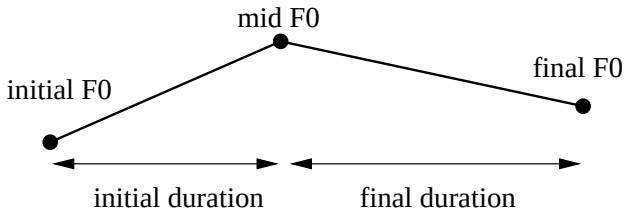


Figure 3.5: Prosodic contour in a syllable defined using 5 parameters according to López-Gonzalo [58].

Chapter 4

Proposed method and its "tools"

First two sections of this chapter give the overview of proposed algorithm of unit determination. Section 4.1 describes, how the different steps should look like and what should be at their input and output, while the following section 4.2 presents the technical solutions. The rest of this chapter is devoted to more detailed description of used ALISP "tools".

4.1 Set of units: steps of the determination ...

The aim of automatic unit set determination is to dispose of the following items after having run the algorithm on a *non-transcribed* database:

- of *transcription* of corpus using those units (Fig. 4.1a) as computer readable files (Fig. 4.1b).
- of *dictionary* of units containing their names[1] and a-priori probabilities (Fig. 4.1c).
- of set of trained *stochastic models*, one for each unit. Those are the most important product of the algorithm, as they allow to identify the units in any future incoming speech.

The processing chain should begin with a *parameterization* or *feature extraction*, in order to describe the speech signal by parameter (feature) vectors. As the temporal dependencies are taken into account in later steps, we decided for a classical feature extraction on fixed frames. Moreover, the parameters were restricted to the *spectral envelope* ones. With regard to clustering algorithms, the distance of two parameter sets should be easy to evaluate, and have a perceptual or at least physical meaning.

When deriving the units, one should begin with a simple hypothesis that the frames of one unit should be more similar one to another, than the to the frames of neighboring units[2]. From this hypothesis, we can derive a constraint of *quasi-stationarity* of signal within one unit, and of *transitions* between different units. Both criteria can be employed: in chapter 3 we have seen various definitions of spectral variation function, attempting to detect the discontinuities. Stochastic rupture detection methods come into mind as well. On the other hand, the stationarity can be detected for example by measuring the conformity of feature vectors to a linear or polynomial model. Using any of those methods, one can obtain a first *segmentation* of the corpus.

Next, it is necessary to assign obtained segments to *classes*. In classical approaches, this operation is rather simple, as one knows the number of classes, and "how should they look like". Again, we can suppose, that a segment within one class should exhibit more similarity

[1] Those names should be computer-readable: using ASCII sequences is more practical than: ♠, ♢, ♡, ☺, ♂, ♀, ...
[2] Bonafonte et al. [16] uses this hypothesis for segment refinement.

to segments from the same class, than to those of other classes. One thus recalls to different techniques of *unsupervised clustering*.

In many applications, it is worth to construct *larger units* by sequencing the basic units. This approach is often used in the speech synthesis, where it helps to overcome problems on transitions of units, hard to concatenate. Classically, those units are given by the analysis of *text:* one can concentrate on frequent words, words containing "difficult" phoneme transitions, etc. In the data-driven approach, no mapping to lexical units is a-priori known, so also this sequencing must be based on the *data.* We need therefore an automatic method of detecting *frequently repeating* sequences of basic units in the training corpus. It is advantageous if those sequences have *variable lengths*.

In the next step, stochastic models must be trained. It is however acceptable to consider those models not only as the final product[3], but as another *tool* for automatic unit determination.

Finally, those found units should be adapted and used according to the application: for example, in the coding we are going to create several "representatives" of each unit, serving for the re-synthesis in the decoder.

4.2 . . . and their technical solutions

For the steps outlined in the previous section, appropriate mathematical tools and their implementation had to be found. As there were many algorithms to implement, the choice of methods was driven not only by their mathematical pureness and efficiency, but also by the availability of software tools, and possibilities of their easy use and modifications.

The different techniques and their sequencing are depicted on Fig 4.2. It should be however noted, that ALISP tools are modular, and that a technique can be completely removed from the scheme (as multigrams and the entire HMM part in verification experiments), or can change its position (multigrams in the second set of coding experiments took place *after* HMM training).

The choice of *parameterization* was very classical: we used cepstral coefficients derived using linear predictive (LP) analysis: LPCC. The advantage of this description is, that the Euclidean distance of two cepstral vectors has a physical meaning:

$$D = \mu \sqrt{2 \sum_{i=1}^{\infty} (c_{1i} - c_{2i})^2}, \tag{4.1}$$

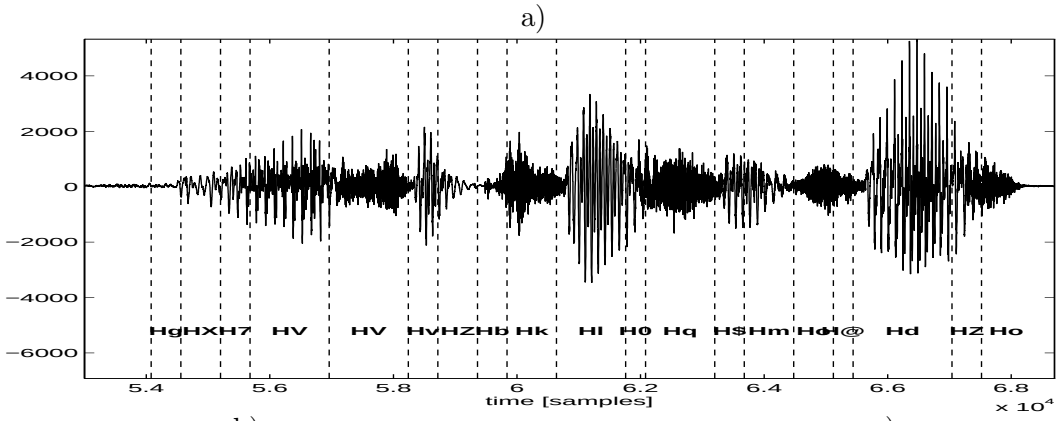
where the constant $\mu = \ln 10/10$ enables the conversion to decibels and $\mathbf{c}_1$ and $\mathbf{c}_2$ are two LPCC vectors, is equal to the *logarithmic distance* of power spectral densities:

$$D = \sqrt{\int_{-1/2}^{1/2} \left[10 \log S_1(f) - 10 \log S_2(f)\right]^2 df} \quad \text{in dB.} \tag{4.2}$$

Other parameterizations, obtained using data-analysis, as for example in Hermansky's works [47], are interesting alternative, but they were not tested experimentally.

For the detection of quasi-stationary parts of speech, there is a bunch of methods to choose from: different modifications of spectral variation function (SVF) and their minima and maxima explorations, "conformity" to the modeled trajectory in a segment model, etc. Those methods were described at different places in Chapter 3. Our choice for this step was the *temporal decomposition (TD)*. The interesting feature of this method is its ability to model the transitions

[3] In classical approaches, the trained models should best fit to given annotation/transcription of the corpus. In data-driven approach, the notion of annotation/transcription is *dynamic*, eg. it can itself be created by models.



b)

```
...
33100000   33800000   Hg
33800000   34100000   HX
34100000   34500000   H7
34500000   34800000   HV
34800000   35600000   HV
35600000   36400000   Hv
36400000   36700000   HZ
36700000   37100000   Hb
37100000   37400000   Hk
37400000   37900000   Hl
37900000   38600000   H0
38600000   38800000   Hq
38800000   39500000   H$
39500000   39800000   Hm
39800000   40300000   Ho
...
```

c)

```
HA   0.014126
HB   0.013483
HC   0.014514
HD   0.023177
HE   0.018745
HF   0.014072
HG   0.016469
HH   0.019160
HI   0.012733
HJ   0.017312
HK   0.013737
HL   0.016589
HM   0.018504
HN   0.018370
HO   0.013470
HP   0.017526
...
```

Figure 4.1: Example of automatic labelling of the corpus on words "Massachusetts has" from the BU Radio Speech Corpus: a) signal with boundaries and labels. b) part of transcription file (times are given in hundreds of nanoseconds). c) part of unit dictionary with associated linear a-priori probabilities.

of two segments by a sum of two interpolation functions. The segment border can thus be "softly" placed to the point, where the two neighboring segments have the same influence.

There are many possibilities of unsupervised clustering of data; some of them are described in Section 4.5 although all were not tested experimentally. The practical choice was the *vector quantization (VQ)*, a classical algorithm, even if we are conscious of its drawbacks (for example, the output codebook is not organized, and the neighboring code-vectors in the codebook are not necessarily closed to each other in the parametric space).

For the "lengthening" of units, *multigram (MG)* method (Section 4.6) was found suitable, as it parses a corpus of symbols and outputs the set of characteristic, frequently appearing sequences of variable length, along with their positions in the corpus. Therefore, we did not experiment with any fixed-length sequencing, even is some reflections were made on using diphone-like units for the VLBR coding. As already mentioned, multigrams were not always used, as they increase the dictionary size and therefore also the computational complexity and storage needs. However, it was interesting to study the difference between short and "lengthened" units in coding, where the lengthening decreases the number of unit transitions.

Finally, for the statistical modeling, the widely employed and theoretically well understood

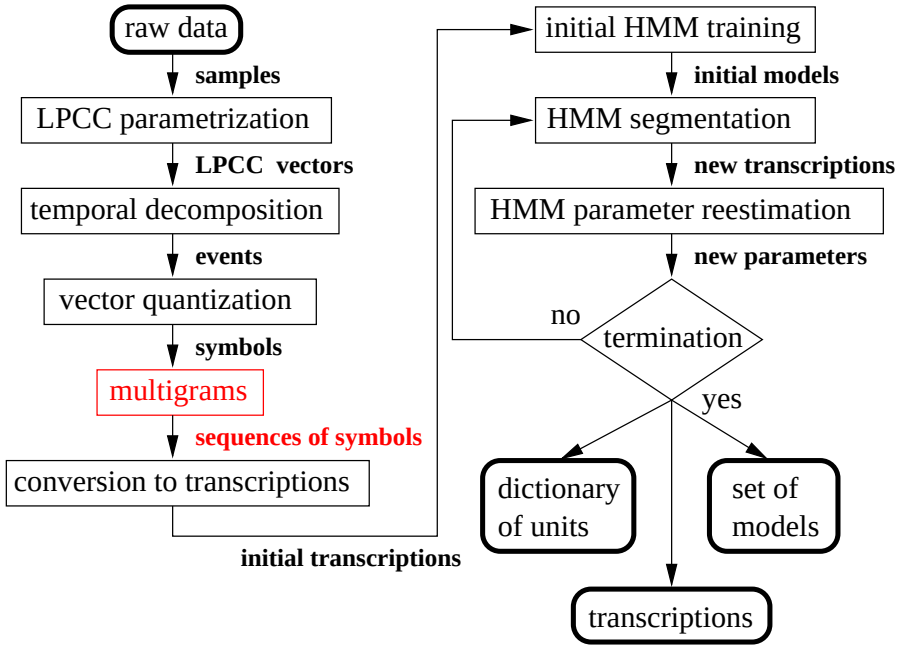


Figure 4.2: Techniques used for the automatic determination of units.

Hidden Markov Model (HMM) framework was used. We made use of the HTK Toolkit [4] for both the training phase and the corpora segmentation using HMMs.

Throughout this thesis, we claim, that the process of unit derivation if fully automatic. In practice, however, it is not true at 100% and the algorithm does contain a-priori human choices:

- the parameters of feature extraction must be set (length of frame, overlap, pre-emphasis, window, number of coefficients).
- it is necessary to specify signal *parts* to work on: we used a voice activity detector (of course, it was necessary to set its parameters), or the parts were determined by original files in the DB (so that in this case, designers of the DB made the choice for us).
- in the temporal decomposition step, the setting of average number of events per second is necessary. The TD was usually set up to produce approximately 15–18 events per second (closed to the phonetic rate).
- in the vector quantization, the a-priori setting of codebook size L is obligatory, as well as choosing the criterion to stop the training.
- for multigrams (if used), is it necessary to fix their maximal length n (we then call them n-multigrams), and the strategy to prune rare sequences out of the dictionary.
- in the HMM portion, a lot of choices are to be done: the number of states per unit, the choice of probability distribution function (PDF) model (Gaussian distribution, mixed Gaussians, discrete), and others: parameter tying, search space limitation (beam width), etc.
- finally, the iterative refinement of HMMs has also its parameters: the stopping criterion or fixed number of iterations. There is also a question of the *language model (LM)* weighting

[4] © Entropic Research Labs

the acoustic probabilities of HMMs during the segmentation: this LM can either be set to non-existent (NULL), or, as it was done in some experiments, the acoustic probabilities can be weighted by a-priori probabilities of models.

Even if those choices seem to deteriorate the concept of ALISP unit determination, we must note, that their number and importance is quite limited compared to classically used schemes.

4.3 Parameterization

As mentioned, the parameterization (or feature extraction) was done using LPCC coefficients. This step can certainly not be ranked as "ALISP", but as the processing chain begins with the parameterization, we found it worth of being included in this theoretical chapter. The subsection 4.3.2 is devoted to the extraction of prosody parameters. Out of these, only the energy and delta-energy were used in the determination of set of units (HMM part), the pitch was significant only in coding experiments.

4.3.1 Spectral envelope parameters – LPCC

According to classical works (for example Rabiner and Schaffer [76], Psutka [74]), the speech is divided into frames of l_f samples with overlapping r_f. Before this operation, the mean of each utterance is removed (centering) and the spectrum is flattened using first order pre-emphasis filter:

$$H_{preem}(z) = 1 - az^{-1}, \tag{4.3}$$

with a closed to 1. Then, each frame is weighted by Hamming window in order to attenuate transition effects on its boundaries:

$$s(n) = w(n)s_{orig}(n) \quad \text{for} \quad n = 0 \dots l_f - 1, \tag{4.4}$$

with the window $w(n)$ defined as:

$$w(n) = 0.54 - 0.46 \cos \frac{2\pi n}{l_f} \quad \text{for} \quad n = 0 \dots l_f - 1. \tag{4.5}$$

From each window, a set of $P + 1$ non-normalized autocorrelation coefficients are derived:

$$r(m) = \sum_{n=0}^{l_f - 1 - m} s(n)s(n+m), \tag{4.6}$$

serving as intermediary representation to obtain the coefficients of linear predictive filter by solving the set of linear equations:

$$r(m) + \sum_{i=1}^{P} a_i r(m-i) = 0. \tag{4.7}$$

This system is solved using the efficient Levinson–Durbin algorithm [76]. The resulting vector of LPC coefficients $\mathbf{a} = [a_1, a_2, \dots, a_P]^T$ is converted to LPCC using

$$c_n = -a_n - \sum_{k=1}^{n-1} \frac{k}{n}\, c_k a_{n-k}, \tag{4.8}$$

for $n = 1 \ldots \infty$, where the vector of LPC coefficients a_i is extended by zeros for $i > P$. Coefficients $c_1, c_2, \ldots, c_P$ then form the LPCC vector $\mathbf{c} = [c_1, c_2, \ldots, c_P]^T$ (zeroth cepstral coefficient carrying the energy information was never used).

In recognizers, the partial independence on varying channel characteristics is ensured using Cepstral Mean Subtraction (CMS) [38]. We used this technique too, subtracting from each cepstral vector the mean:

$$\bar{\mathbf{c}} = \sum_{i=0}^{N_f - 1} \mathbf{c}_i, \tag{4.9}$$

where N_f is the number of frames in utterance, recorded in presumably equal channel conditions (this can be one telephone call or a "story" in radio-recorded data). It should be mentioned, that when using the cepstral coefficients to return to parameters of filter in the synthesizer, the mean must be always re-introduced.

4.3.2 Prosody: voicing, pitch and energy

The *energy* of frame is defined as normalized sum of sample powers in pre-emphasized, but not windowed frame:

$$e = \frac{1}{l_f} \sum_{n=0}^{l_f - 1} s_{orig}^2(n). \tag{4.10}$$

The *voicing decision* for i-th frame is based on following three conditions (to declare a frame voiced, all three must be met):

- *zero crossing rate* which, for voiced frame, must be lower than pre-determined threshold: $z_i < z_{voi}$.
- *prediction gain* defined as the ratio of total frame energy over the energy of residual. This ratio can be computed using reflection coefficients obtained as side-product of Levinson–Durbin algorithm:

  $$GP = \left(\prod_{i=1}^{P} 1 - k_i^2 \right)^{-1}, \tag{4.11}$$

  and it must be greater than pre-determined threshold: $GP > GP_{voi}$.
- finally, a voiced frame must have the *relative energy* (normalized to maximum energy of the utterance) greater than a predetermined threshold:

  $$RE_i = \frac{e_i}{e_{max}} > RE_{voi}. \tag{4.12}$$

The *pitch period* was determined using FFT-cepstral method, with longer frames than those used for the LPCC parameters estimation. The FFT cepstrum is computed using definition equation:

$$c(n) = \mathcal{F}^{-1} \left\{ \ln \left| \mathcal{F}[s(n)] \right| \right\}, \tag{4.13}$$

and the pitch period is determined by searching the maximum of $c(n)$ in the region of allowable pitch lags $[T_{0min}, T_{0max}]$:

$$T_0 = \arg \max_{T_{0min} \le n \le T_{0max}} c(n). \tag{4.14}$$

4.3.3 Voice activity detection

The voice activity detector (VAD) was realized using simple two threshold scheme with following decision smoothing. The raw decisions are based on two criteria:

- frame is declared active only if its relative energy reaches a pre-determined threshold:

$$RE_i = \frac{e_i}{e_{max}} > RE_{vad}, \tag{4.15}$$

  where e_{max} is the maximum frame energy of the utterance.

- the absolute frame energy must reach a pre-determined threshold: $e_i > E_{vad}$.

These raw decisions are smoothed using "OR" filter preventing very short low-energy regions (such as stop parts of plosives) from being classified as silence. If for n-th frame, the raw voice activity decision is marked $RVAD(n)$, having value 1 for active frame and 0 for passive, the final decision is taken by computing logical sum over l_{vad} left-context and l_{vad} right-context frames:

$$VAD(n) = \bigoplus_{i=n-l_{vad}}^{n+l_{vad}} RVAD(i), \tag{4.16}$$

where the operator $\oplus$ stands for logical summation (OR).

4.3.4 Practical issues of speech parameterization

In handling databases or signal coming directly from audio input, it is often necessary to split it into pieces, as many tools (TD, HMMs) must work with finite portions of signal in order to be able to perform iterative refinement or back-tracking in the Viterbi algorithm. In a simplest view, we can say, that these operations need *the end* of signal for successful completion of their operation. Such pre-segmentation into "pieces" may be done in two ways:

1. using *voice activity detector* where at each beginning of "silence" region, the end of previous "speech portion" is declared.

2. for signals, where it is difficult to get reliable VAD decisions[5] it is possible to segment the signal into pre-defined length segments, by finding minimum energy in a "tolerance" region. This is illustrated in Figure 4.3, where prior to the minimum search, the energy contour is smoothed using median filter to prevent the boundary placement in a deep, but brief minimum (which may be a part of plosive).

4.4 Temporal decomposition

This method, mentioned already in subsection 3.1.3, was introduced in 1983 by Atal [5] as non-uniform sampling and interpolation procedure for efficient high quality speech coding. It was further argued by Bimbot et al. [12], that this technique is related to simplified production model of speech: the vocal tract receives commands, but as the sequence of commands is faster than corresponding physical changes in the vocal tract, there is an overlapping of reactions to one, two or more commands. In our work, the temporal decomposition (TD) was used as the first step in the quest for characteristic speech units. Due to the fact, that it is able of automatic

[5] for example for radio data, where the speakers seem to be payed for the cadence in words per second, and where even the breath-takings are barely noticeable...

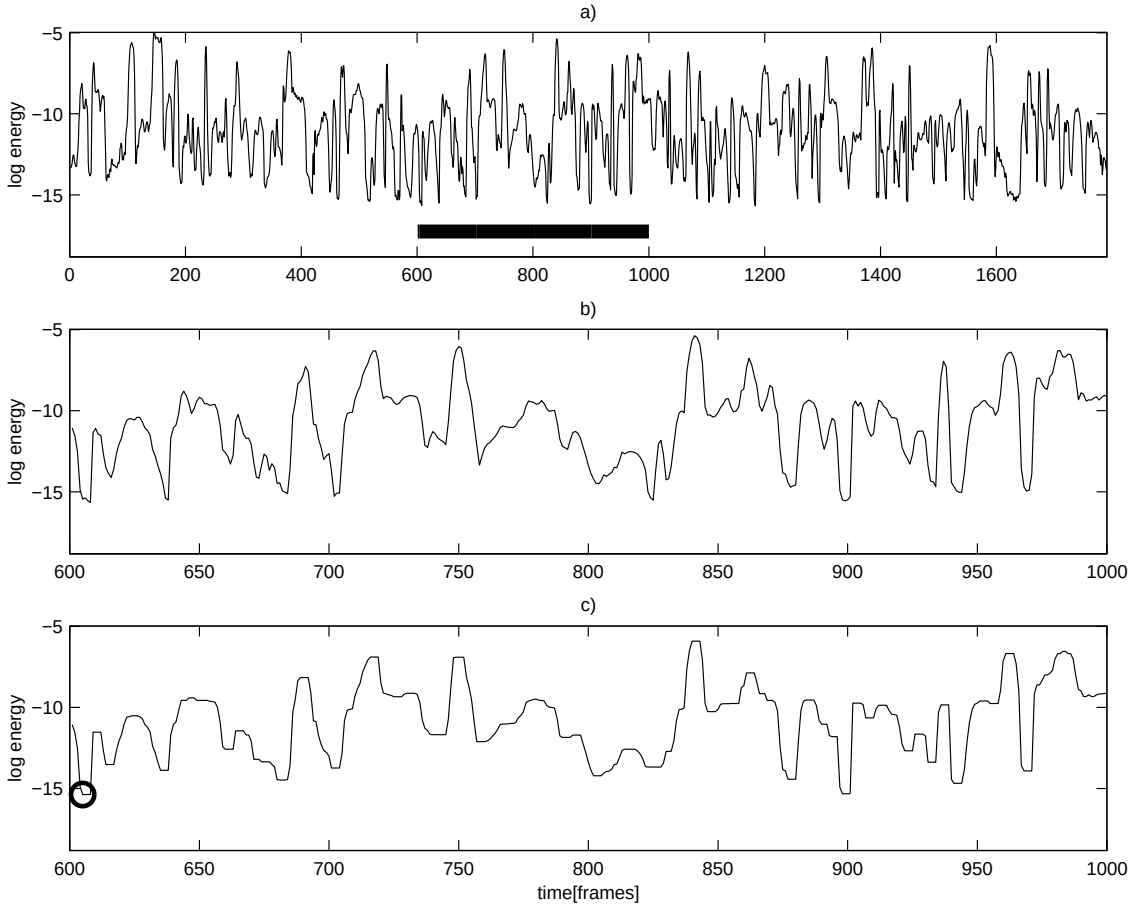


Figure 4.3: Pre-segmentation of signal into parts of approximately equal lengths using minimum energy criterion: a) energy contour with marked "tolerance region". b) zoom on the tolerance region. c) the same region after median filtering, with detected minimum.

detection of stable and transitional parts of speech parameter matrix, it is a valuable alternative to edge detection techniques described throughout Chapter 3. Segments obtained by TD can be further clustered and the boundaries together with labels used as initial transcription for HMM training[6].

4.4.1 Temporal decomposition principles

As it was already mentioned in subsection on TD-based coding (3.1.3), the temporal decomposition is capable of approximation of matrix $\mathbf{X}$ of N successive parameter vectors, called *spectral trajectory matrix,* by m *target vectors*[7] with associated *interpolation functions (IF).* So, the trajectory of i-th parameter can be approximated by:

$$\hat{x}_i(n) = \sum_{k=1}^{m} a_{ik}\phi_k(n), \tag{4.17}$$

[6]in traditional recognition/VLBR coding systems, a phonetically labelled portion of DB is used to *bootstrap* phone models, but here, no such information is available.

[7]Ghaemmeghami et al. in [41, 42, 43] are calling the elements $a_{i,k}$ of target vectors "weighting factors". In our opinion, it is more convenient to say, that to approximate parameters of t-th frame $\hat{\mathbf{x}}(t)$, the target contributions are weighted by interpolation functions values for t: $\phi_1(t), \phi_2(t), \ldots, \phi_m(t)$.

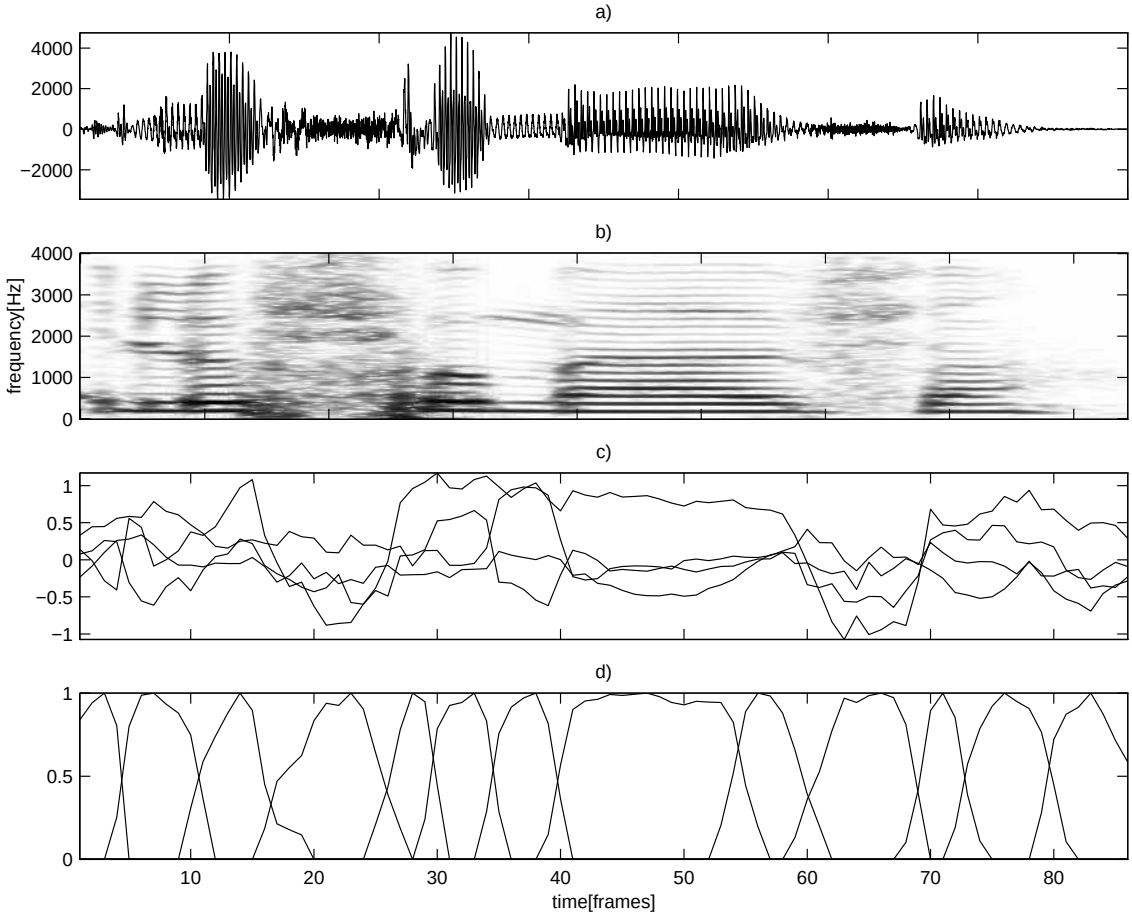


Figure 4.4: Illustration of temporal decomposition of French word "le chômage": a) signal. b) spectrogram. c) trajectories of first 4 LPCC parameters. d) TD interpolation functions.

where a_{ik} is the i-th element of k-th target vector $\mathbf{a}_k$ and $\phi_k(n)$ is its associated IF. Functions $\phi_k(n)$ can be overlapping but each one should be concentrated in time:

$$\phi_k(t) = \begin{cases} \neq 0 & \text{for} \quad b_k \leq t \leq e_k \\ \\ 0 & \text{otherwise,} \end{cases} \tag{4.18}$$

where b_k and e_k stand respectively for the beginning and end of k-th IF. The influence of k-th target $\mathbf{a}_k$ is thus limited in time only to the region $[b_k, e_k]$ corresponding to the reality, where an articulatory command issued at time t_0 can not influence the vocal tract 5 minutes later.

This principle can be written more compactly using three matrices: $\hat{\mathbf{X}}$ as the full approximated parameter matrix ($P \times N$), $\mathbf{A}$ as the matrix of target-vectors (dimensions $P \times m$ – one target per column), and a $m \times N$ sparse matrix $\mathbf{\Phi}$ with IFs in lines (so that, in reality, only a small portion of each line is non-zero). The following relation links the three matrices:

$$\hat{\mathbf{X}} = \mathbf{A}\mathbf{\Phi}. \tag{4.19}$$

Graphically, the principle of TD is illustrated in Figure 4.4.

4.4.2 Finding targets and interpolation functions

The goal of TD is to approximate the spectral trajectory matrix $\mathbf{X}$ by matrix $\hat{\mathbf{X}}$ as precisely as possible, while using limited number of events. Moreover, the segments should represent stationary parts of spectral evolution, while transitions should be taken into account by IF-weighted "passage" from one segment to another.

Atal, in his original article [5], proposes using of long-window singular value decomposition (SVD) of matrix and timing function locating the events. Rather than this method, we were extensively using short-time SVD with adaptive windowing and post-processing of IFs and targets, defined by Bimbot et al. in [12] and thoroughly described in Bimbot's report [11]. This method is also the key-point of Bimbot's software package `td95`, which we were using in our experiments[8].

The algorithm consists of the following steps:

1. initial search of IFs using adaptive rectangular window and short-time singular-value decomposition (SVD).
2. post-processing of IFs (smoothing, de-correlation and normalization).
3. target computation:
$$\mathbf{A} = \mathbf{X}\mathbf{\Phi}^{\#}, \tag{4.20}$$
where $\mathbf{\Phi}^{\#}$ is the pseudo-inverse of initial IF matrix.
4. local adaptive refinement of IFs and targets.

The detailed algorithm is presented in Appendix A.

4.4.3 From TD interpolation functions to segments

This subsection describes rather "technical" than "scientific" aspect of TD: the determination of segment boundaries from interpolation functions and their limits. From the preceding steps, we dispose of matrix $\mathbf{\Phi}$, with one IF in each line: $\phi_i(t)$. The number of IFs is m. Each IF has its nonzero portion situated between its hypothetical beginning b_i and end e_i (denoted τ_{1i} and τ_{2i} in Appendix A). Each IF has also its gravity center denoted by gc_i. Our task is to determine the borders of segments: bb_i and ee_i, so that the segments are adjacent: $bb_i = ee_{i-1} + 1$ [9].

Two methods for boundaries determination were introduced. In the first, simplified one, the left boundary is given as mean of beginning of current and end of previous IF:

$$bb_i = \left\lfloor \frac{b_i + e_{i-1}}{2} \right\rfloor, \tag{4.21}$$

where the operator $\lfloor \cdot \rfloor$ denotes the flooring to nearest integer towards zero.

In the second method, the left boundary is determined as the *intersection* of previous IF with current IF. As those intersections can be multiple, it is precisely defined as mean of the first left intersection and first right intersection:

- to find the left intersection I_l: initialize time t to b_i. While $\phi_i(t) < \phi_{i-1}(t)$, $t = t + 1$. Record the final time $I_l = t$.

[8] the author made this package public for research and education purposes, the requests should be addressed to `bimbot@irisa.fr`.

[9] in practical realizations, the "ends" are made to point to one frame after the segment, so that $bb_{c_i} = ee_{c_{i-1}}$, where the subscript c means "computer use".

- to find the right intersection I_r: initialize time t to e_{i-1}. While $\phi_i(t) > \phi_{i-1}(t)$, $t = t - 1$. Record the final time $I_r = t$.

The boundary is then defined as:

$$bb_i = \left\lfloor \frac{I_l + I_r}{2} \right\rfloor . \tag{4.22}$$

After any of those two computations, it is checked, if bb_i does not fall into the previous segment: if $bb_i \leq bb_{i-1}$, the current event is discarded (this can happen if three or more IFs are overlapping, or if one IF is too short). Finally, the beginnings of segments are completed by setting $bb_1 = 1$ and associated ends are computed:

$$ee_i = bb_{i+1} - 1 \quad \text{for} \quad 1 \leq i \leq m - 1, \tag{4.23}$$

and $ee_m = N$. Variables bb_i and ee_i can be directly converted to computer readable form.

This step of TD post-processing should also output vectors suitable for clustering. As we will show in subsection 4.5.3, two kinds of vectors are requested: targets and original vectors situated in gravity centers of IFs. The *target matrix* is one of the results of TD. Only in case an event was discarded[10] in the previous step, the corresponding target vector is deleted from matrix **A**.

Before the collection of *matrix of original vectors*, the gravity centers are checked to be situated within respective segment:

$$bb_i \leq gc_i \leq ee_i. \tag{4.24}$$

If this is not the case, an attempt is made to use the maximum of IF instead of gravity center:

$$gc_i = t_{im}, \tag{4.25}$$

where t_{im} denotes the time index of the maximum. Also this maximum is checked to lie inside the segment (Inequality 4.24), and if it is not the case, the mean of beginning and end of segment is taken as last resort:

$$gc_i = \left\lfloor \frac{bb_i + ee_i}{2} \right\rfloor . \tag{4.26}$$

The matrix of original vectors is then defined as:

$$\mathbf{X}_o = [\mathbf{x}_{gc_1}, \mathbf{x}_{gc_2}, \ldots, \mathbf{x}_{gc_m}] . \tag{4.27}$$

4.4.4 Practical issues of the temporal decomposition

Bimbot's method is stable and producing satisfactory results for different parameter sets. The only parameter, necessitating hand-adjustment, is the threshold D_{thr} (Eq. A.8) limiting the number of parameter trajectories in the new space (after SVD), participating on the creation of new interpolation function. This value influences the final number of IFs: unfortunately, there is not an analytical form for the determination of D_{thr} from, for example, desired average number of events per second. The threshold has to be set heuristically using several speech files and trial-and-miss method.

[10] Fortunately, when processing real speech data, the discarding of events is usually not frequent.

4.5 Clustering

On the way from raw speech data towards set of ALISP units, it is necessary to convert continuous domain representation of segments to symbolic one. To be able to label segments of data by symbols, one must dispose:

- of *dictionary*, each class being sufficiently coherent to describe acoustically similar speech segments and in the same time sufficiently different from other classes (intra-class variability should be minimal, while the inter-class one should be maximal). Each class must have associated model or set of values comparable to input data and a symbol for convenient segment labelling.
- of *method* for associating the closest class to input segment. This can be performed by evaluating distance between two vectors or two sets of vectors and by finding the minimum distance over all classes, or by choosing the class with maximal model likelihood.

In this section, we are going to deal only with the initial clustering of segments, with boundaries pre-determined by TD. We are going to treat the joint classification and segment boundaries determination by HMMs in special section (4.7).

This initial clustering was performed by Vector Quantization (VQ) or by one of its modifications. Another possibility is the use of Ergodic Hidden Markov Models (EHMM) for this operation. EHMMs have not been used in experiments, but for completeness, their description is included in the section devoted to HMMs (subsection 4.7.4).

4.5.1 Vector quantization and its modifications

VQ is the most straightforward method to cluster events using parameter vectors, providing us with the class dictionary as well as with labelling of segments. According to classical works of Gersho [40] and Makhoul [60], the population of P-dimensional vectors $\mathbf{x} = [x_1, \ldots, x_P]^T$ is represented by finite set of L "reconstruction", "output" or "code" vectors called *codebook*:

$$\mathbf{Y} = \{\mathbf{y}_i; 1 \leq i \leq L\}. \tag{4.28}$$

To design such a codebook, the P-dimensional space is partitioned into L cells $\{C_i; 1 \leq i \leq L\}$, and we call *quantization* the process of assignment one of code-vectors $\mathbf{y}_i$ to each $\mathbf{x}$ according to appertaining of $\mathbf{x}$ to cell C_i:

$$q(\mathbf{x}) = \mathbf{y}_i \quad \text{if} \quad \mathbf{x} \in C_i. \tag{4.29}$$

To formalize this classification of vectors, we must dispose:

- of *centroids* of cells. Those are mostly equivalent to reconstruction vectors and for mean square error criterion (minimizing the mean Euclidean distance of all vectors belonging to C_i to the centroid), it is defined as mean value:

$$\mathbf{y}_i = \frac{1}{M_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}, \tag{4.30}$$

  where M_i is the number of vectors belonging to i-th cell.

- of *metric* which may be defined in several ways [60]. In our work, the Euclidean distance of P-dimensional LPCC vectors was used:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})} = \sqrt{\sum_{k=1}^{P} |x_k - y_k|^2}. \tag{4.31}$$

This distance is an approximation of logarithmic spectral distortion (see Eqs. 4.1 and 4.2), so it was used as a plausible way to evaluate differences of two spectral envelopes.

Having centroids and metric, the appertaining of vector to cell is given by evaluating all distances to centroids and by finding the minimal one:

$$q(\mathbf{x}) = \mathbf{y}_i \quad \text{if} \quad d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j), \quad j \neq i, \ 1 \leq j \leq L. \tag{4.32}$$

4.5.2 Codebook training

The training is based on K-means algorithm working with set of N training vectors $\{\mathbf{x}(n)\}$, with successive splitting of codebook vectors so that their number in step s is $L^s = 2^{s-1} = 1, 2, 4, 8, \ldots, L$. This method is called Linde–Buzo–Gray or LBG. It is obvious, that in this algorithm, the final codebook size L must be a power of 2. The procedure is written as follows:

1. initialization: set step number $s = 1$. Set initial codebook size $L^1 = 2^0 = 1$. Compute initial centroid $\mathbf{y}_1^1$ as mean of all vectors in the training set:

$$\mathbf{y}_1^1 = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}(n). \tag{4.33}$$

2. multiplication of number of classes: set $s = s + 1$. Set $L^s = 2L^{s-1}$. Split centroids by moving them in opposite directions by a vectorial variable $\mathbf{e}$:

$$\mathbf{y}_i^{s-1} \Rightarrow \begin{cases} \mathbf{y}_{2i-1}^s = \mathbf{y}_i^{s-1} + \mathbf{e} \\ \\ \mathbf{y}_{2i}^s = \mathbf{y}_i^{s-1} - \mathbf{e} \end{cases} \quad \text{for} \quad 1 \leq i \leq L^{s-1}. \tag{4.34}$$

3. training of L^s quantizer using the LBG method:

   (a) initialization: set number of iterations $m = 0$. Take freshly split centroids as initial values for the algorithm: $\mathbf{y}_i(0) = \mathbf{y}_i^s$ for $1 \leq i \leq L^s$.

   (b) classification of vectors into clusters by nearest neighbor rule:

$$\mathbf{x} \in C_i(m) \quad \text{if} \quad d[\mathbf{x}, \mathbf{y}_i(m)] \leq d[\mathbf{x}, \mathbf{y}_j(m)], \quad \text{for} \quad \forall j \neq i. \tag{4.35}$$

   (c) code vector updating: set $m = m + 1$. For each cluster, compute new centroid:

$$\mathbf{y}_i = \text{cent}[C_i(m)], \tag{4.36}$$

   in our case using mean computation (Equation 4.30).

   (d) termination test: if overall distortion $D(m)$, calculated as sum of distances of $\mathbf{x}(n)$ to respective centroids, changes from previous distortion more than a pre-specified relative threshold:

$$\frac{D(m-1) - D(m)}{D(m-1)} > RD_{thr}, \tag{4.37}$$

   return to Step (b), otherwise update centroids of outer loop with results: $\mathbf{y}_i^s = \mathbf{y}_i(m)$ for $1 \leq i \leq L^s$.

4. termination test: if $L^s < L$, go to Step 2, otherwise terminate.

4.5.3 Vector quantization classifying TD events

The codebook was always *trained* using one parameter vector per event. Depending on the experiment, the spectral targets $\mathbf{a}_n$ (Equation 4.19) or original vectors situated in gravity centers of interpolation functions were taken as input to the codebook training. In the former case, one can argue, that the value of target is more representative for the segment, but it can be also strongly influenced by the transition, as the contributions of i-th and $i+1$-th segment are merged by IF overlapping. For the later choice, the vectors are certainly more prone to noise and estimation errors, but they are situated in central stationary part of TD event, where neighboring events should have minimal impact (see Figure 4.4). With any of those representations, the codebook training is done by LBG method with successive splitting, as described in the previous paragraph.

In the *quantization*, the character of used vectors must correspond to that used in the training. If the codebook is trained with targets, the quantization must also use targets. In case of using original vectors, we experimented with two possible quantizations:

- as in the codebook creation, for classification of segment $\mathbf{X}(n)$, take into account only vector situated in gravity center of n-th IF:

$$\mathbf{X}(n) \in C_i \quad \text{if} \quad d[\mathbf{x}(gc_n), \mathbf{y}_i] \leq d[\mathbf{x}(gc_n), \mathbf{y}_j] \quad \text{for} \quad \forall j \neq i, \tag{4.38}$$

  where gc_n stands for gravity center of n-th IF.

- using *cumulated distance* of all vectors of segment:

$$\mathbf{X}(n) \in C_i \quad \text{if} \quad d_c[\mathbf{X}(n), \mathbf{y}_i] \leq d_c[\mathbf{X}(n), \mathbf{y}_j] \quad \text{for} \quad \forall j \neq i, \tag{4.39}$$

  where the cumulated distance is defined as non-normalized sum of distances of all segment vectors to the centroid:

$$d_c[\mathbf{X}(n), \mathbf{y}] = \sum_{t=bb_n}^{ee_n} d[\mathbf{x}(t), \mathbf{y}], \tag{4.40}$$

  and bb_n and ee_n are respectively the beginning (first vector index) and end (last vector index) of n-th segment.

4.6 Multigrams

In our work, the multigrams (MGs) were used as an important tool to find regularities in the symbolic description of speech and to advance from "symbol"-like units to larger ones; to be coherent with the large vocabulary continuous speech recognition (LVCSR) terminology, they can be called "words". Although the application of MGs directly to vector quantized parameter vectors was also tested [98], the main use of MGs was in the post-processing of temporal decomposition events, in order to find their characteristic sequences, and to act as word-generation model (we can not say directly "language model") for word-based recognition.

The MG framework was originally developed for text processing (finding regularities within a string of symbols) by Bimbot, Deligne et al. [13, 14, 30], then reformulated by Deligne as a production model [29, 32]. The theory of joint MGs capable of describing events in two parallel streams of observations was also developed by Deligne [29, 32, 33], but we do not make use of joint MGs in our work.

4.6.1 Definition and statistical formulation

We suppose, that an observation sequence $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T]$ is generated by a source emitting multigrams: variable length sequences drawn from a limited set $\{x_i\}$ of Z units. The sequence of MGs is denoted: $X = [x_{i1}, x_{i2}, \ldots, x_{iN}]$, and the segmentation of observations into sequences is denoted: $S = [s_1, s_2, \ldots, s_N]$. The schematic view of $(\mathbf{O}, S, X)$ can be seen in Figure 4.5

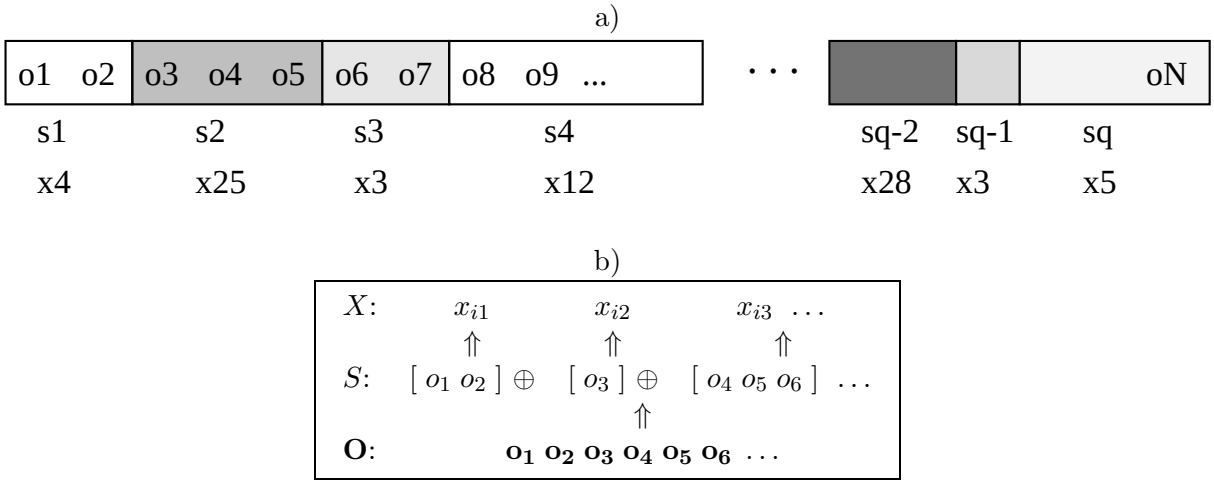


Figure 4.5: Two examples of underlying multigram structure (S, X) of observation sequence $\mathbf{O}$.

Neither the set of MGs $\{x_i\}$, nor the segmentation S are known. Our task is to identify them using solely the observation sequence $\mathbf{O}$.

Unlikely to traditional recognition approaches, where the set of underlying units $\{x_i\}$ is known and fixed a-priori, in case of MGs we are confronted with the task of *determination* of this set. From the statistical point of view, the optimal set could be found by maximizing the likelihood:

$$\{x_i\}^\star = \arg\max_{\forall\{x_i\}} \mathcal{L}(\mathbf{O}|\{x_i\})\mathcal{L}(\{x_i\}), \tag{4.41}$$

where the likelihood $\mathcal{L}(\mathbf{O}|\{x_i\})$ is similar to standard recognition framework and says, how well the data fit to given set of units. The likelihood of the set itself $\mathcal{L}(\{x_i\})$ is in "standard" cases supposed fixed and is not taken into account. Here, it must be evaluated, which is however not possible to do directly. This likelihood can be approximated by the inverse of number of bits necessary for describing of $\{x_i\}$ (its description length). Such Minimum Description Length (MDL) criterion attempts to find a trade-off between the adequacy of the set of units with observations data, and the complexity of the set, which in ideal case leads to a reduction of the risk of over-learning. If we adopt this approach, we can define the number of bits necessary to represent multigram x_i as $\log_2 \mathcal{P}(x_i)$, where $\mathcal{P}(x_i)$ is the a-priori probability of the multigram. The complexity of $\{x_i\}$ can then be defined as its entropy:

$$H(\{x_i\}) = -\sum_{i=1}^{Z} \mathcal{P}(x_i) \log_2 \mathcal{P}(x_i), \tag{4.42}$$

and it is obvious, that this complexity can be decreased by discarding MGs with low probabilities.

The likelihood of data given the set of units $\{x_i\}$ can be defined as:

$$\mathcal{L}(\mathbf{O}|\{x_i\}) = \sum_{S,X} \mathcal{L}(\mathbf{O}, S, X|\{x_i\}). \tag{4.43}$$

In case the multigrams are supposed independent, the joint likelihood of data $\mathbf{O}$, segmentation S and associated MG string X, given the set $\{x_i\}$, can be evaluated as:

$$\mathcal{L}(\mathbf{O}, S, X|\{x_i\}) = \prod_{t=1}^{c(X,S)} \mathcal{P}(s_t|x_{it})\mathcal{P}(x_{it}), \tag{4.44}$$

where $c(S, X)$ is the number of sequences in the segmentation S (or of MGs in the string X), $\{\mathcal{P}(x_{it})\}$ is the set of prior probabilities of MGs and $\mathcal{P}(s|x_i)$ is the probability of observation sequence given underlying multigram x_i. The optimal underlying segmentation and string of MGs are determined as:

$$(S^\star, X^\star) = \arg \max_{\forall(S,X)} \mathcal{L}(\mathbf{O}, S, X|\{x_i\}). \tag{4.45}$$

4.6.2 Estimation of the multigram model

The optimization problem given in Equation 4.41 can not be solved directly. Instead, we search for the optimal set $\{x_i\}^\star$ in a two-step iterative procedure:

1. "upgrading" of set $\{x_i\}^k$ to $\{x_i\}^{k+1}$ by reducing its complexity. A heuristic is applied to discard MGs with low probabilities.

2. re-estimation of MG parameters, e.g. of their a-priori probabilities $\mathcal{P}(x_i)$ and of distributions $\mathcal{P}(s|x_i)$ in order to maximize the data likelihood $\mathcal{L}(\mathbf{O}|\{x_i\}^{k+1})$.

First, we will outline the Step 2. of this algorithm. The parameters of MGs can be obtained by ML estimation from incomplete data — similarly as the parameters in HMM estimation. Here, the observed sequence $\mathbf{O}$ is known, while the segmentation S and MG string X are not. The maximization of $\mathcal{L}(\mathbf{O}|\{x_i\}^{k+1})$ is achieved by maximizing the following auxiliary function[11]:

$$Q(k, k+1) = \sum_{S,X} \mathcal{L}^k(\mathbf{O}, S, X) \log \mathcal{L}^{k+1}(\mathbf{O}, S, X). \tag{4.46}$$

The re-estimation formula of parameters at (k+1)-th iteration can be obtained by maximization of $Q(k, k+1)$ with respect to multigram parameters. Having developed [32] the criterion in Equation 4.46, it can be shown, that it can be slit into two criteria: $Q_1(k, k+1)$ and $Q_2(k, k+1)$ where the former depends on the priors $\mathcal{P}^{k+1}(x_i)$ and the later on the a-posteriori distributions $\mathcal{P}^{k+1}(s|x_i)$:

- the *estimation of priors* is in the most general case given in (k+1)-th iteration as:

$$\mathcal{P}^{k+1}(x_i) = \frac{\sum_{S,X} c(x_i|S, X)\mathcal{L}^k(\mathbf{O}, S, X)}{\sum_{S,X} c(S, X)\mathcal{L}^k(\mathbf{O}, S, X)}, \tag{4.47}$$

  which means the averaging of counts of given multigram x_i in all possible segmentations, with all possible MG chains, weighted by the likelihood of the triple $(\mathbf{O}, S, X)$ given in Equation 4.44. This probability evaluation can be called Baum-Welch re-estimation, as all possible combinations of X and S are taken into account. In case of *discrete multigrams*,

[11] as in the ML step, the set $\{x_i\}^{k+1}$ is supposed fixed, it is omitted in the conditional likelihoods $\mathcal{L}(\mathbf{O}, S, X|\{x_i\}^{k+1})$.

where there is no ambiguity between S and X, e.g., where each multigram x_{it} is fully determined by the sequence s_t, the a-posteriori distribution is a Dirac function:

$$\mathcal{P}(s_t|x_{it}) = \begin{cases} 1 & \text{if} \quad s_t \equiv x_{it}, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{4.48}$$

Therefore, the segmentation is equivalent to MG string $S \equiv X$ and the Equation 4.47 can be rewritten:

$$\mathcal{P}^{k+1}(x_i) = \frac{\sum\limits_{S} c(x_i|S)\mathcal{L}^k(\mathbf{O}, S)}{\sum\limits_{S} c(S)\mathcal{L}^k(\mathbf{O}, S)}. \tag{4.49}$$

For both cases of a-posteriori distributions (Dirac or continuous one), the probability can also be re-estimated not by averaging all S and X combinations, but by taking into account only the counts of the optimal (Equation 4.45) combination $(S^{\star k}, X^{\star k})$:

$$\mathcal{P}^{\star k+1}(x_i) = \frac{c(x_i|S^{\star k}, X^{\star k})}{c(S^{\star k}, X^{\star k})}. \tag{4.50}$$

This prior estimation can be called "Viterbi" and it was the only one used in our experiments.

- the estimation of *a-posteriori* probability distributions $\mathcal{P}(s|x_i)$. In case of *discrete multigrams*, there is nothing to estimate, as the sequences s_t are equivalent to multigrams x_{it}. The PDF is thus given by the Dirac distribution given in Equation 4.48. In *continuous case*, each multigram must have an associated model. This model can be given by a set of vectors compared with observation sequence similarly as in Matrix Quantization (see subsection 3.1.1), or by an HMM associated with each MG. Those models can be re-estimated using optimal "transcriptions" $S^{\star k}, X^{\star k}$ from the previous iteration. Subsection 4.6.4 is dealing with those re-estimations in detail.

The Step 1. of above mentioned algorithm: "upgrading" of MG set $\{x_i\}^k$ to $\{x_i\}^{k+1}$ is simple, when Viterbi re-estimation of priors (Equation 4.50) is used. This equation can be modified in several ways to reflect the elimination of rare MGs. One of methods, described in [13], is the use of *penalized probability*:

$$\mathcal{P}_a^{\star k+1}(x_i) = \frac{c(x_i)}{C}\left(1 - a\sqrt{\frac{C - c(x_i)}{C c(x_i)}}\right), \tag{4.51}$$

where $c(x_i)$ is a shortcut for $c(x_i|S^{\star k}, X^{\star k})$ and C stands for the total number of sequences $c(S^{\star k}, X^{\star k})$. The constant $a > 0$ is a pruning factor helping to eliminate rare MGs from the dictionary: if the count $c(x_i)$ is too small, the term $a\sqrt{\cdot}$ becomes greater than one, and the probability $\mathcal{P}_a^{\star k+1}(x_i)$ falls below 0. In this case, the MG is eliminated and the probabilities are re-normalized to sum up to one for all retained MGs.

Another possibility is the setting of *count threshold*. If the number of occurrences of multigram x_i falls below a pre-determined value:

$$c(x_i) < c_{thr}, \tag{4.52}$$

this MG is discarded and the other probabilities are re-normalized. The threshold c_{thr} should be set with respect to the following step of HMM training or segment mean determination. To let the estimation be statistically reliable, the values of c_{thr} should be at least several tens.

When one of above methods of dictionary pruning is applied to discrete MGs estimation, it is necessary to pay attention to the "realizability" of the segmentation $S^{k+1} \equiv X^{k+1}$: as an example, let us consider the observation sequence $S = [ABCAADAB]$. If the MG dictionary $\{x_i\}$ contains MGs: $x_1 = A, x_2 = B, x_3 = C, x_4 = D, x_5 = AB, x_6 = CAA$, the segmentation is possible and would have for example the form: $X = [x_5, x_6, x_4, x_5]$. The likelihood of this segmentation is $\mathcal{L}(O|X) = \mathcal{P}(x_5)\mathcal{P}(x_6)\mathcal{P}(x_4)\mathcal{P}(x_5)$. In case we deleted the MG x_4, due to its low probability, the observation sequence could not be segmented anymore and the likelihood would be 0. Therefore, it is a good idea to keep at least 1-grams[12] x_1, x_2, x_3, x_4 in the MG dictionary, to allow the sequence to be segmented in the worst case symbol-by-symbol.

4.6.3 Discrete multigrams

As already mentioned in the previous subsection, discrete multigrams [13, 14, 30] operate on strings of symbols, the PDF of sequence depending on MG being given by a Dirac distribution (Equation 4.48) and the segmentation S is equivalent to MG string: $S \equiv X$. The sequence of observations is given as a string of symbols: $O = [o_1, o_2, \ldots, o_T]$. The optimal segmentation $S^\star$ (equivalent to optimal MG chain $X^\star$) is given by:

$$S^\star = \arg\max_{\forall S} \mathcal{L}(\mathbf{O}, S|\{x_i\}), \tag{4.53}$$

which is a simplification of Equation 4.45. This segmentation can be found by Viterbi procedure. Considering MGs with maximal length n (n-multigrams), for each time $t+1 : 1 \leq t \leq T-1$, the maximal partial likelihood is found:

$$\mathcal{L}^\star(o_1, \ldots, o_{t+1}) = \max_{1 \leq i \leq n} \mathcal{P}(o_{t-i+2}, \ldots, o_{t+1}) \mathcal{L}^\star(o_1, \ldots, o_{t-i+1}), \tag{4.54}$$

and the optimal "cutting point" corresponding to maximal $\mathcal{P}(\cdot)\mathcal{L}^\star(\cdot)$ product is memorized. When finished with evaluations of $\mathcal{L}^\star$ (for the time $t = T-1$), the optimal segmentation can be found by back-tracing these "cutting points". We see, that we can freely replace prior probability $\mathcal{P}(x_i)$ by $\mathcal{P}(o_{t-i+2}, \ldots, o_{t+1})$, as there is one-to-one correspondence between MG and the sequence.

For the re-estimation of prior probabilities in Baum-Welch style, a similar framework as for HMMs was developed by Deligne [30], using forward likelihood α and backward likelihood β. However, this formalism was not used in our work and we re-estimated the prior probabilities using the optimal segmentation and Equation 4.50.

With above mentioned mathematical framework, the estimation process of discrete MGs using training observation sequence O can be summarized as follows:

1. *Initialization.* Find all unique sequences of observation symbols of length 1 to n, initialize the MG dictionary $\{x_i\}^0$. Set prior probabilities of MGs to:

$$\mathcal{P}^0(x_i) = \frac{c_{tot}(x_i|O)}{c_{tot}(O)}, \tag{4.55}$$

   where $c_{tot}(x_i|O)$ is the number of occurrences of sequence $s = x_i$ in the training non-segmented string, and $c_{tot} = n \times T$ is the total number of sequences of length 1 to n in the training string. Set the iteration number $k = 0$.

2. *Segmentation.* Using the set of probabilities $\{\mathcal{P}^k(x_i)\}$ and the Viterbi algorithm (Equation 4.54), find the optimal segmentation of the training string $S^{\star k}$.

[12] "keeping" in the dictionary means attributing of small non-zero probability.

3. *Re-estimation of probabilities.* New probabilities are given by:

$$\mathcal{P}^{k+1}(x_i) = \frac{c(x_i|S^{\star k})}{c(S^{\star k})} \tag{4.56}$$

(simplification of Equation 4.50), where $c(x_i|S^{\star k})$ is the number of occurrences of x_i in the optimal segmentation and $c(S^{\star k})$ is the total number of sequences in this segmentation. Pruning using penalized probability or count thresholds may also be applied. Retained MGs constitute the new set $\{x_i\}^{k+1}$.

4. *Termination test.* Stop iterations, if the difference of likelihoods $\mathcal{L}^{\star}(O|\{x_i\}^k)$ and $\mathcal{L}^{\star}(O|\{x_i\}^{k+1})$ is no more significant, or if a pre-set number of iterations was reached. Otherwise set $k = k + 1$ and return to Step 2.

4.6.4 Continuous multigrams

Continuous MGs are more interested framework than discrete ones, as we consider a multigram x_i to emit sequence s with certain emission probability $\mathcal{P}(s|x_i)$. The formalism of re-estimation of prior probabilities $\mathcal{P}(x_i)$ and emission PDFs $\mathcal{P}(s|x_i)$ was already outlined in subsection 4.6.2, here we are going to concentrate on practical realizations of alignment of sequences with MGs.

4.6.4.1 Multigrams with distance

In our early works [97, 96, 98], we used the multigrams with distance notion as alternative to discrete MGs. From the statistical point of view, the solution is "heretic", as the emission probability $\mathcal{P}(s|x_i)$ is not evaluated, and instead, the a-priori probability of MG is weighted by a function of distance between the MG and the represented sequence.

On contrary to discrete MGs, here, a MG is not defined as a sequence of *symbols*, but a sequence of *parameter vectors:* $x_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \ldots, \mathbf{x}_{i,l(x_i)}]$. To compare the t-th sequence in segmentation S with multigram x_i, it must have the same length: $s_t = [\mathbf{o}_{t,1}, \mathbf{o}_{t,2}, \ldots, \mathbf{o}_{t,l(x_i)}]$. Then, the distance of sequence from MG can be defined as the average of vector distances:

$$D(s_t, x_i) = \frac{1}{l(x_i)} \sum_{j=1}^{l(x_i)} d(\mathbf{o}_{t,j}, \mathbf{x}_{i,j}). \tag{4.57}$$

We used simple Euclidean distance to compare two parameter vectors:

$$d(\mathbf{o}, \mathbf{x}) = \sqrt{(\mathbf{o} - \mathbf{x})^T (\mathbf{o} - \mathbf{x})}. \tag{4.58}$$

The a-priori probability of multigram $\mathcal{P}(x_i)$ is *penalized* by a function of distance of MG and sequence x_t:

$$\mathcal{P}'(x_i) = Q[D(s_t, x_i)]\mathcal{P}(x_i). \tag{4.59}$$

In our experiments, the penalizing function $Q[\cdot]$ was defined as a simple partially linear function:

$$Q[D] = \begin{cases} 1 - \dfrac{D}{D_{max}} & \text{for} \quad D \leq D_{max}, \\ \\ 0 & \text{otherwise}, \end{cases} \tag{4.60}$$

with D_{max} giving the maximal distance for which $\mathcal{P}'$ may be non-zero. For similar reasons, as cited in the end of subsection 4.6.2, a minimum value should be attributed instead of zero to the function $Q[D]$ at least for uni-grams, to ensure the feasibility of the segmentation.

With such formalism, the evaluation of likelihood[13] of a segmentation with underlying MG string given in Equation 4.44 can be rewritten:

$$\mathcal{L}'(\mathbf{O}, S, X|\{x_i\}) = \prod_{t=1}^{c(X,S)} \mathcal{P}'(x_{it}), \tag{4.61}$$

with penalized probability $\mathcal{P}'(x_{it})$ evaluated according to Equation 4.59. When searching the optimal combination $(S^\star, X^\star)$, the Viterbi algorithm given in previous subsection is modified. The partial likelihood of observation up to the time $t+1$ is now given as:

$$\mathcal{L}'^\star(\mathbf{o}_1, \ldots, \mathbf{o}_{t+1}) = \max_{1 \le j \le n} \mathcal{P}'^\star(x_i)\mathcal{L}'^\star(\mathbf{o}_1, \ldots, \mathbf{o}_{t-j+1}), \tag{4.62}$$

where the "winning" penalized probability among all MGs of length j is given as:

$$\mathcal{P}'^\star(x_i) = \max_{\forall x_i; l(x_i)=j} Q[D(\mathbf{o}_{t-j+2}, \ldots, \mathbf{o}_{t+1}, x_i)]\mathcal{P}(x_i). \tag{4.63}$$

The re-estimation algorithm of multigrams with distance differs from the previous one in the fact, that not only MGs' a-priori probabilities, but also MGs themselves (their vectors $\mathbf{x}_{i,j}$) must be re-estimated:

1. *Initialization.* There is no straightforward method of initializing multigrams-with-distance dictionary. Experiments were done with initialization by beforehand trained dictionary of discrete MGs, while replacing symbols by code-vectors of corresponding VQ codebook. The advantage of discrete MGs is that they provide a-priori probabilities. Another possibility would be to use random initialization. Denote initial set of multigrams $\{x_i\}^0$, and their probabilities $\mathcal{P}^0(x_i)$.

2. *Segmentation.* Using the set of probabilities $\{\mathcal{P}^k(x_i)\}$ and code-multigrams[14] $\{x_i\}^k$, segment the observation string $\mathbf{O}$ by maximizing the "penalized likelihood" (Equation 4.61) using Viterbi algorithm (Equation 4.62).

3. Re-estimation of probabilities. New probabilities are given by:

$$\mathcal{P}^{k+1}(x_i) = \frac{c(x_i|X^{\star k})}{c(X^{\star k})}. \tag{4.64}$$

   where $c(x_i|X^{\star k})$ is the count of multigram x_i in the optimal MG string and $c(X^{\star k})$ is the total count of MGs in this string. Note, that we can no more use similar notations as in Equation 4.56, as here the MGs are not equivalent to sequences. Pruning schemes can be applied.

4. *Re-estimation of code-multigrams.* New code-multigram x_i^{k+1} (of course, only if it was retained in the pruning) is determined as the mean of sequences represented by x_i^k in the optimal segmentation. Mathematically:

$$\mathbf{x}_{i,j}^{k+1} = \frac{1}{c(x_i|X^{\star k})} \sum_{\forall s_t \approx x_i} \mathbf{o}_{t,j} \quad \text{for} \quad 1 \le j \le l(x_i), \tag{4.65}$$

   where $s_t \approx x_i$ denotes the representation of sequence s_t by multigram x_i.

5. *Termination test.* Stop iterations if the difference of "penalized likelihoods" $\mathcal{L}'^\star(\mathbf{O}|\{x_i\}^k)$ and $\mathcal{L}'^\star(\mathbf{O}|\{x_i\}^{k+1})$ is no more significant, or if pre-determined number of iterations was reached. Otherwise, set $k = k+1$ and return to Step 2.

[13] the quantity $\mathcal{L}'(\cdot)$ could be called "penalized likelihood", but due to non-linear probability modification, it does not have too much to do with statistically "proper" view of a likelihood.

[14] as the dictionary of multigrams-with-distance contains parameter vectors, the sequences of vectors can be called similarly to VQ "code-multigrams".

4.6.4.2 Multigrams as Gaussian sources

Due to lack of mathematical rigorousness of the previous definition of multigrams-with-distance, Baudoin et al. [9, 8] have reformulated them as Gaussian sources. In this formalism, multigrams are supposed to generate variable length sequences of parameter vectors, which are Gaussian with covariance matrix $\mathbf{\Sigma}$ equal to $\sigma^2\mathbf{I}$, where $\mathbf{I}$ is $P \times P$ identity matrix (P is the parameter vector length). Similarly as in the previous case, the multigram x_i is represented by a sequence of *vector means*: $x_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \ldots, \mathbf{x}_{i,l(x_i)}]$. The emission probability for sequence s_t of the same length is then given by normal PDF (defined in the following section in Equation 4.80):

$$\mathcal{P}(s_t|x_i) = \prod_{j=1}^{l(x_i)} \mathcal{N}(\mathbf{s}_{t,j}; \mathbf{x}_{i,j}, \sigma^2\mathbf{I}). \tag{4.66}$$

When evaluating the log probability, we obtain:

$$\log \mathcal{P}(s_t|x_i) = l(x_i) \log \frac{1}{(2\pi)^{\frac{n}{2}}\sigma^n} - \frac{1}{2\sigma^2} \sum_{j=1}^{l(x_i)} \|\mathbf{s}_{t,j} - \mathbf{x}_{i,j}\|^2. \tag{4.67}$$

Finding the optimal segmentation and MG string (Equation 4.45) is then equivalent to the maximization of:

$$(S^\star, X^\star) = \arg\max_{\forall(S,X)} \sum_{t=1}^{c(S,X)} \left[\log \mathcal{P}(s_t|x_i) - \frac{1}{2\sigma^2} \sum_{j=1}^{l(x_i)} \|\mathbf{s}_{t,j} - \mathbf{x}_{i,j}\|^2 \right]. \tag{4.68}$$

When returning back to the linear domain:

$$(S^\star, X^\star) = \arg\max_{\forall(S,X)} \prod_{t=1}^{c(S,X)} \mathcal{P}(s_t|x_i) \exp\left[\frac{1}{2\sigma^2} \sum_{j=1}^{l(x_i)} \|\mathbf{s}_{t,j} - \mathbf{x}_{i,j}\|^2 \right]. \tag{4.69}$$

The Gaussian function $\exp[\cdot]$ can be compared to triangular penalizing function $Q[\cdot]$ used in the previous paragraph.

Baudoin [9, 8] has investigated also "lengthened" multigrams, where only multigrams of lengths 1 to $\frac{n}{2}$ are stored, and lengths $\frac{n}{2}+1$ to n are given by linear interpolation of $\frac{n}{2}$-grams. This modification limits the size of MG dictionary, but greatly increases the complexity of the algorithm. Rather than this modification, we used MGs represented by HMMs, as described in the following paragraph.

4.6.4.3 Multigrams represented by HMMs

In comparison to all previous representations of MGs, the HMMs generating the "emission probabilities" $\mathcal{P}(s_t, x_i)$ are by far the most general framework for MGs. There are two possible approaches to HMM incorporation into MGs:

- the first based on discrete MGs, where each symbol is replaced by "small" or "symbol-like" HMM. A multigram x_i of length $l(x_i)$ is thus represented by a concatenation of several HMMs, each standing for a symbol and capable of representing small number of observation vectors:

  $$x_i \equiv [M_{i,1}, M_{i,2}, \ldots, M_{i,l(x_i)}], \tag{4.70}$$

  where $M_{i,j}$ denotes the j-th model in i-th multigram. The MG segmentation can be compared to connected word recognition: each MG stands for a word and the "symbol"-models can be viewed as phones in standard LVCSR system. The language model is

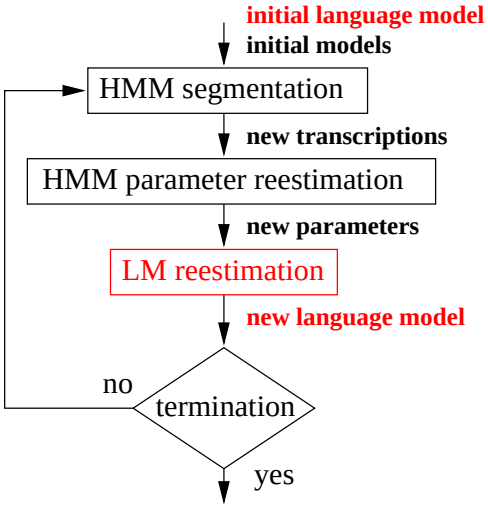


Figure 4.6: Re-estimation of a-priori probabilities of models (uni-gram language model) within the iterative refinement of HMMs.

uni-gram, as each MG has its associated a-priori probability and the MGs are supposed independent. The recognition can be performed using token passing paradigm, as described in subsection 4.7.3. If the information about "visited" HMMs is needed (for example for the re-estimation of models or for the synthesis), the algorithm should be modified to keep track of models, not only of word-ends.

Technically, it is also possible to implement this formalism by *a-posteriori* processing of "symbol-like" HMM segmentation by discrete multigrams.

- the second approach considers one MG being represented by one HMM, so that:

$$x_i \equiv M_i. \tag{4.71}$$

To reflect the variability of MG lengths, these models can have different numbers of states and the proportions of "next-state" ($a_{i,i+1}$) and "stay-in-state" ($a_{i,i}$) transition probabilities may vary[15]. The recognition works in similar way as in the previous case, but here, all words are composed of only one model, so that we could call this approach "connected model recognition". As each model (multigram) has an a-priori probability which must be taken into account, the language model is once again uni-gram.

As it is stated in subsection 4.7.5, the language model (or, as we have seen, simply a-priori probabilities of models or of sequences of models) can be re-estimated within the iterative refinement of HMM set. In both cases, simple re-estimation formula 4.50, based on counts of MGs in the optimal segmentation $S^{\star k}$ with optimal MG string $X^{\star k}$, can be used. Figure 4.6 presents this adaptation of a-priori probabilities.

4.6.5 Practical issues of multigram implementation

4.6.5.1 Organization of discrete MG dictionary

The most straightforward method for storing discrete MGs in memory would be to record for each multigram x_i the corresponding sequence of $l(x_i)$ symbols: $x_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,l_{x_i}}]$, with

[15] some authors use HMMs with explicit duration modeling, but it was not used in our work.

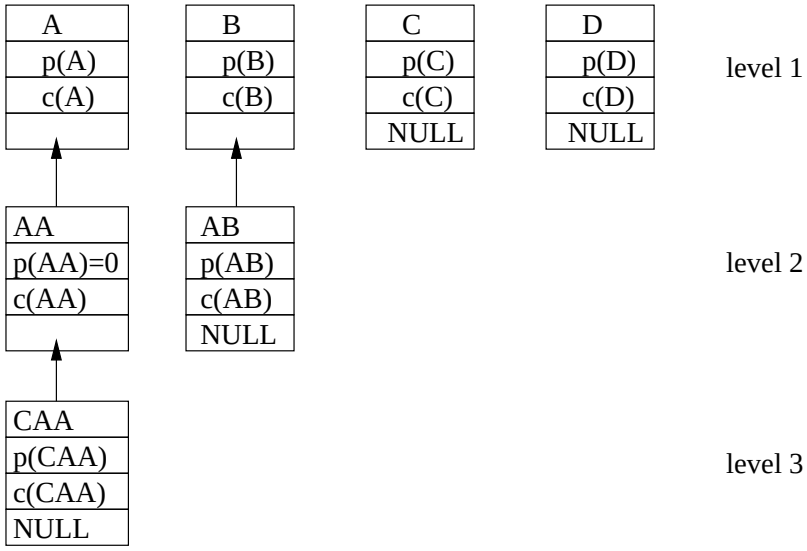


Figure 4.7: Hierarchical structure of MG dictionary. Represented multigrams correspond to example given in the end of subsection 4.6.2: $x_1 = A$, $x_2 = B$, $x_3 = C$, $x_4 = D$, $x_5 = AA$, $x_6 = AB$, $x_7 = CAA$

.

associated a-priori probability and with a counter for memorizing its count in the MG string X (Equations 4.55 and 4.56). Each evaluation of partial likelihood 4.54 would then require a search done by comparison of sequence $[o_{t-i+2}, \ldots, o_{t+1}]$ to all i-grams in the dictionary. We have developed more efficient structure with tree-like organization of MG dictionary. Note, that in Equation 4.54, one needs to sequentially look for probabilities of sequences: $[o_{t+1}]$, $[o_t, o_{t+1}]$, $\ldots$, $[o_{t-n+2}, \ldots, o_{t+1}]$, so that each $(i+1)$-gram ends with previously investigated i-gram. Therefore, multigrams are read in memory from the end to the beginning by stepping in hierarchical structure depicted in Figure 4.7. Each node in level i, representing an i-gram, has an associated probability (which may be zero to indicate, that this MG was discarded), a counter, and a pointer to next level (NULL if no $(i+1)$-gram with the end given by current i-gram exists). When evaluating partial likelihoods in Equation 4.54, the number of search steps in each level is at most L, where L is the length of symbol alphabet. Generally, this number is smaller than L, because for an i-gram, the number of $(i+1)$-grams ending with this i-gram is limited.

4.6.5.2 Implementation of the segmentation

The segmentation and optimal MG string search is governed by Viterbi algorithm — Equation 4.54 for the discrete case, Equation 4.62 for MGs with distance, and it is similarly defined also for HMM representation of MGs, where it can be efficiently implemented by token passing. The disadvantage of this search is that the entire utterance must be processed until back-tracing can be started and until actual segmentation $S^\star$ and MG string $X^\star$ can be found. For discrete MG case, we developed an algorithm working with dynamic buffering, capable of producing the segmentation with user defined maximal delay LB_{max}. The optimal "cut-length" for time $t+1$ (according to maximal $\mathcal{PL}^\star$ product) is denoted l_{t+1}^{opt}. The actual length of buffer is denoted LB and the buffer contains history of last $LB-1$ symbols: $O_{hist} = o_{t-LB+2}, \ldots, o_t$ and a history of $LB-1$ optimal "cut-lengths" $l_{hist}^{opt} = l_{t-LB+2}^{opt}, \ldots, l_t^{opt}$. For the time $t+1$, the algorithm is written as follows:

1. increment the length of buffer $LB = LB + 1$.
2. add the symbol o_{t+1} to the end of symbol history. The history becomes: $O_{hist} = o_{t-LB+2}, \ldots, o_{t+1}$.
3. Determine the optimal "cut-length" by maximizing the $\mathcal{PL}^\star$ product (Equation 4.54). Write it to the end of history of optimal lengths, which becomes: $l^{opt}_{hist} = l^{opt}_{t-LB+2}, \ldots, l^{opt}_{t+1}$.
4. search a *common point* CP of n segmentations, beginning by $l^{opt}_{t-n+2}, \ldots, l^{opt}_{t+1}$ by backward search in the history of optimal lengths.
5. if common point CP was not found and the buffer length reached its limit $LB = LB_{max}$, force the common point to the current time: $CP = t + 1$.
6. if CP was found or forced, declare the segmentation and optimal MG string *found* for the substring $o_{t-LB+2}, \ldots, o_{CP}$, and shorten the buffer $LB = k + 1 - CP$.
7. if not yet at the end of utterance ($t + 1 \leq T$), increment t and return to Step 1.

The meaning of naturally found (not forced) common point is that from the beginning of buffer: o_{t-LB+2} up to this point (o_{CP}), the segmentation and MG string will not change for any time $\tau > t$. Therefore, they can be directly used (written to file, or, if in re-estimation phase, the counters can be updated, etc.). The forcing of common point is mostly necessary if $l^{opt} = n$ (maximal MG length) for many successive symbols: in this case, all segmentations are unique and do not "meet". This is often a proof of over-learning, where too many long MGs are retained in the dictionary.

4.6.6 Use of multigrams in automatic determination of units

In our early experiments, we used MGs directly for the processing of parameter vectors. Discrete MGs were tested with vector-quantized vectors (VQ providing the conversion to symbols), while multigrams-with-distance were applied to raw vectors. Although those experiments familiarized us with the multigram method, their results did not bring significant difference compared to other techniques. The main use of MGs in our work was in word-generation for symbol-HMMs and in post-processing of TD events to create large HMMs.

The first approach consisted of training multigrams not on sequences of quantized TD events, but already with previous symbol-HMM segmentation as input. To obtain this initial segmentation, the symbol-HMMs were first themselves trained on original TD events. This approach corresponds to "word" model, where each MG is composed of several HMMs shared among MGs.

The second approach was based on searching discrete MGs of quantized TD events. The transcriptions were modified to reflect these longer units, and to prepare a set of variable state number HMMs for the training. An assumption was done, that each TD event is composed of a stable part and of two transitional parts connecting it to the neighboring events. Therefore, for a sequence of i original TD events (an i-gram), the number of states of corresponding HMM was set to $2i + 1$ (not taking into account entry and exit non-emitting states), in order to cover each stable part and each transition with one state. These HMMs were then viewed as independent, each with its own a-priori probability.

The main benefit of both approaches is obvious in very low bit-rate coding using automatically derived units: by making the units capable of representing longer and variable-length sequences, the mean rate given in number of bits per second, necessary for unit indices transmission, decreases. The second benefit is linked to transition handling in synthesis. When preparing synthesis units by choosing them in the training corpus, we are motivated to have as

little transitions of those units as possible. Multigrams are intrinsically lowering the number of transitions, as they represent longer sequences than original TD units or symbol-trained HMMs.

4.7 Hidden Markov Models

HMMs are probably the most widespread tool used in nowadays systems of speech recognition (SR) described in numerous books [75], tutorial articles [77] and other publications. HMMs provide unified stochastic view of segmental processing[16] based on likelihood evaluations and maximization. Besides the fact, that those models have proved their efficiency in SR, that the theory is well established and that numerous computer tools for HMMs are available, their stochastic nature permits their incorporation as inferior (front-end) or superior level to other processings based on probabilities and likelihoods, as for example multigrams. The main advantage of HMMs is their ability to represent segments variable in time as well as in spectral trajectory, which is difficult, when working with simple VQ, or with the TD-VQ tandem.

The following subsections give a brief overview of HMM basis and describe their engagement in the unit determination. In the experimental part of work, we were using `HTK` toolkit developed at Entropic Research Labs; the manual of this toolkit, Young's HTK-book [106] contains also exhaustive description of HMM theory. Subsections, dealing with HMM basis, are based on this book and often use the same notations.

4.7.1 HMM basis

The easiest way to introduce HMMs is the case of isolated segment recognition, where only one model is representing one segment at time. The segment is defined as sequence of *observation vectors* $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T]$ of length T. Those may be vectors of parameters (LPCC, MFCC and others) with associated energies and possibly velocity (Δ) and acceleration ($\Delta\Delta$) coefficients. We dispose also of vocabulary of segments $\{w_i\}$ and the recognition can be viewed as finding segment w_i that maximizes the likelihood:

$$\arg\max_{\forall i} \{\mathcal{L}(w_i|\mathbf{O})\}, \tag{4.72}$$

where the observation sequence $\mathbf{O}$ is known. This likelihood must be computed indirectly using Bayes rule:

$$\mathcal{L}(w_i|\mathbf{O}) = \frac{\mathcal{L}(\mathbf{O}|w_i)\mathcal{P}(w_i)}{\mathcal{L}(\mathbf{O})}. \tag{4.73}$$

The likelihood of observation data $\mathcal{L}(\mathbf{O})$ is not known, but is constant and does not need to be considered. $\mathcal{P}(w_i)$ is the a-priori probability of segment w_i, and finally, the likelihood that observations are generated by i-th segment $\mathcal{L}(\mathbf{O}|w_i)$, is to be calculated. In case of HMM, each segment has an associated model M. This model is a finite state machine, shown in Figure 4.8, with N states. With each state[17] j, an emission probability density function (PDF) is associated: $b_j(\mathbf{o}_t)$ and each arc connecting two states has a transition probability a_{ij}. To observation sequence $\mathbf{O}$ of length T, one may associate state sequence X of length T giving numbers of states associated to observations[18]. So, for the example observation sequence depicted in Figure 4.8,

[16] we can freely replace "word" widely used in HMM literature also for phones, syllables, demi-syllables, etc., by "segment".

[17] emission probabilities may also be associated with arcs.

[18] in the definition, we use state No. 1 as non-emitting "entry point" to the HMM, so that no observations are associated with it and it does not carry any emission PDF. The same holds for exit point: state number N.

the state sequence would be: $X = [1, 2, 2, 3, 4, 4, 5, 6]$. The joint likelihood that sequence $\mathbf{O}$ is generated by model M along path X is given:

$$\mathcal{L}(\mathbf{O}, X|M) = a_{x(o)x(1)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}_t) a_{x(t)x(t+1)}, \tag{4.74}$$

in our example it is:

$$\mathcal{L}(\mathbf{O}, X|M) = a_{12} b_2(\mathbf{o}_1) a_{22} b_2(\mathbf{o}_2) a_{23} b_3(\mathbf{o}_3) \dots \tag{4.75}$$

In practice, the observation sequence is known, but the path along which observations are emitted, and probabilities a_{ij} and distributions b_j are not. This is why the model is called *hidden*.

The likelihood, that $\mathbf{O}$ was emitted by model M, is given by the sum of terms $\mathcal{L}(\mathbf{O}, X|M)$ over the set of all possible paths:

$$\mathcal{L}(\mathbf{O}|M) = \sum_{\{X\}} \mathcal{L}(\mathbf{O}, X|M), \tag{4.76}$$

or by finding the path with maximum likelihood:

$$\mathcal{L}^\star(\mathbf{O}|M) = \max_{\{X\}} \mathcal{L}(\mathbf{O}, X|M). \tag{4.77}$$

Training algorithm, based on the former definition (Equation 4.76) if often referred as Baum–Welch, while the later (4.77) is named Viterbi. Efficient algorithms have been developed to compute both $\mathcal{L}(\mathbf{O}|M)$ and $\mathcal{L}^\star(\mathbf{O}|M)$.

While the transition probabilities are simple values grouped in matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix}, \tag{4.78}$$

the output probability distributions $b_j(\mathbf{o}_t)$ must be given parametrically. Widely used and the most general is modeling by mixtures of Gaussians with splitting of observation vectors $\mathbf{o}$ into streams $\mathbf{o}_s$ (sub-vectors of $\mathbf{o}$). Emission PDF is then given by:

$$b_j(\mathbf{o}_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} \mathcal{N}(\mathbf{o}_{st}; \mu_{jsm}, \mathbf{\Sigma}_{jsm}) \right]^{\gamma_s}, \tag{4.79}$$

where S is the number of non-overlapping streams in observation vector $\mathbf{o}$, M_s is the number of Gaussian components in s-th stream and $\mathcal{N}(\cdot; \mu, \mathbf{\Sigma})$ is multivariate Gaussian PDF with mean vector μ and covariance matrix $\mathbf{\Sigma}$:

$$\mathcal{N}(\mathbf{o}; \mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{o}-\mu)^T \mathbf{\Sigma}^{-1} (\mathbf{o}-\mu)}, \tag{4.80}$$

where n is the size of one stream. Coefficients c_{jsm} are weights of m-th mixture component for s-th stream of j-th state and finally, γ_s are stream weight constants.

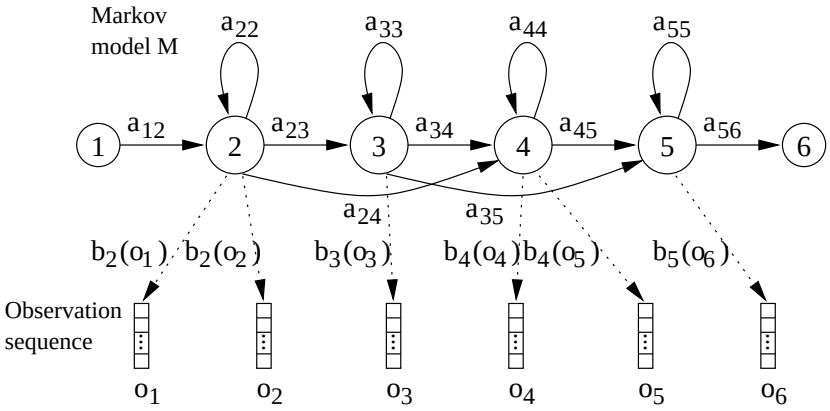


Figure 4.8: Hidden Markov model: structure and example of assignment of observation vectors to states. States Nos. 1 and 6 are non-emitting.

4.7.2 Estimation of HMM parameters

The model parameters are not known a-priori and must be estimated using training data set. In the simplest case, this consists of huge number of acoustic vectors with associated transcriptions, so that for training of model M, we dispose of number of observation sequences $\{\mathbf{O}\}$. The fact that multiple streams come into account does not alter the mathematical apparatus as the streams are supposed independent. Generally, the training of parameters can be divided into three steps:

1. *Initialization of parameters.* In the beginning of this step, observation vectors are uniformly assigned to states and means and variances are computed using simple formulae:

$$\hat{\mu}_j = \frac{1}{T_j} \sum_{i=1}^{T_j} \mathbf{o}_t^{(j)}, \tag{4.81}$$

$$\hat{\mathbf{\Sigma}}_j = \frac{1}{T_j} \sum_{i=1}^{T_j} (\mathbf{o}_t^{(j)} - \mu_j)(\mathbf{o}_t^{(j)} - \mu_j)^T, \tag{4.82}$$

   where T_j denotes the number of observations $\mathbf{o}_t^{(j)}$ assigned to j-th state. Note, that here, the assignment is "hard", so that one vector can "belong" only to one state. Then, this assignment is reevaluated by finding the optimal Viterbi alignment of observations and states (Equation 4.77). Means and covariances are then re-estimated and those two steps are iterated until the increase of likelihood over all training sequences for this model:

$$\sum_{\{\mathbf{O}\}} \mathcal{L}^\star(\mathbf{O}|M) \tag{4.83}$$

   does not change significantly. This step does not use transition probabilities and does not estimate them.

2. *Context independent re-estimation.* When speaking about context, one must take into account, that training segments are parts of segment sequences as for example `Hg, HX, H7, HV,...` in Figure 4.1. Context independent re-estimation means, that neighboring segments (for example `Hg` and `H7` for `HX` in our example) are not taken into account and that segment boundaries (`33800000` and `34100000` in hundreds of ns) will not be changed.

Here again, model M is re-estimated using collection $\{\mathbf{O}\}$ of training observation sequences. On contrary to previous step, here, the appertaining of vector $\mathbf{o}_t$ to state j is not "yes" or "not", but is expressed by likelihood $L_j(t)$ measuring "being in state j at time t". Those evaluations are based on Baum-Welch formula (Equation 4.76), where the set of all possible paths $\{X\}$ is taken into account. Using this "soft" appertaining function, one may reformulate Equations 4.81 and 4.82 to:

$$\hat{\mu}_j = \frac{1}{T}\sum_{i=1}^{T}\frac{L_j(t)\mathbf{o}_t}{L_j(t)}, \tag{4.84}$$

$$\hat{\mu}_j = \frac{1}{T}\sum_{i=1}^{T}\frac{L_j(t)(\mathbf{o}_t-\mu_j)(\mathbf{o}_t-\mu_j)^T}{L_j(t)}. \tag{4.85}$$

In above mentioned equations, we have omitted the sum for all associated training sequences $\{\mathbf{O}\}$ and indices i denoting the index of sequence. Similar formulae can be found for transition probabilities a_{ij}, taking into account "soft" transitions from one state to another. For working with "appertaining" likelihoods $L_j(t)$, efficient formalism using forward and backward alignment costs was developed, but as it is not the key-point of this thesis, we refer the reader to excellent explication in [75, 106, 77].

3. *Context dependent re-estimation.* In this step, not only model parameters, but also boundaries of segments are changed. This is also the only re-estimation method necessitating only of *transcriptions* and no more of *labelling* of the training corpus. For each training utterance, huge composite HMM is created by concatenation of all models appearing in the utterance. For our example of Figure 4.1, such "global" model would be given as: $\mathbf{M} = [\ldots, M_{Hg}, M_{HX}, M_{H7}, M_{HV}, \ldots]$. The non-emitting entry and exit states play the important role of "glue" in this concatenation. All utterances are taken into account to reestimate jointly parameters of *all* models using similar formulae as 4.84 and 4.85.

4.7.3 Recognition and token passing

In isolated segment (or word) recognition, the task equals to computation of Viterbi alignment likelihood $\mathcal{L}^\star(\mathbf{O}|M_i)$ for all models M_i and choosing the model maximizing the likelihood. In connected word recognition, boundaries of segments are not known and it is necessary to perform the segmentation and assignment of models to segments jointly. This is realized by connecting all available models to a *network*, where one may easily integrate language model (LM) parameters, and of application of *Token Passing*[19] [107]. The algorithm is articulated as follows:

- Initialization: put token with value 0 to initial state of each model. Put tokens with values $-\infty$ to the other states.
- Algorithm:

for $t = 1$ **to** T **do**
  for each state i **do**
    Pass a copy of the token in state i to all connecting states j
    (nonzero probability a_{ij}) and increment its value by $\log a_{ij} + \log b_j(\mathbf{o}_t)$.
  end

[19] token passing is not the only available method for connected word recognition: one pass (OP) and level building dynamic programming (LBDP) have also been investigated, but according to [107], the token passing may be considered as common formalism for both LBDP and OP.

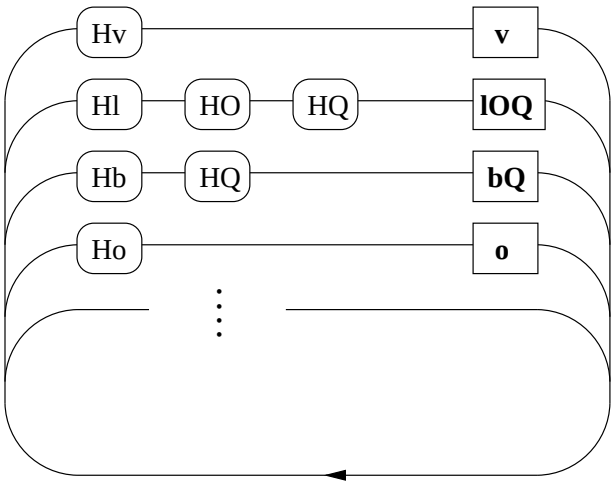


Figure 4.9: Example of "word" network. Round boxes denote models while square boxes stand for word-end nodes.

```
        Discard original tokens.
        for each state i do
            Compare values of all tokens in state i, find the token with greatest
            value and discard the rest.
        end
    end
```

- Termination: Examine all final states of all models; the model carrying the token with maximal value wins.

Evidently, on contrary to Equation 4.74, logarithms of probabilities are used to allow for easy summations instead of multiplications.

In case of *continuous segment recognition*, segments can be grouped into *words* [20] composed of one or more segments. Such words constitute a network depicted in Figure 4.9 with so called word-end nodes in square boxes. The only difference from previous algorithm is that each time a token is passed through a word-end node, the identity of this node must be recorded in order to be able to back-track "words" visited by the winning token of the utterance. To allow for this, each token carries a word-end link and each time it crosses a word-end node, new structure called *word-link record (WLR)* is created and the word-end link is made to point to it. WLR itself contains the time it was created, partial alignment likelihood of "creator" token at that time, and a pointer to another WLR, where the contents of old token's word-end link is copied. In this way, we obtain a linked set of WLRs which may be efficiently back-traced, when we are finished with the utterance and when the winning token is found. This structure allows for other options, not investigated in our work (N-best list, when multiple tokens in state coming from different predecessor words are allowed, etc.). The important feature is, that when a token is entering word w_i, its value can be directly augmented by the log-probability $\log \mathcal{P}(w_i)$, which allows for transparent incorporation of language models.

4.7.4 Ergodic HMM

This model mentioned already in the section dealing with clustering (4.5) and depicted in Figure 4.10 differs from standardly used left-right HMMs (Figure 4.8) by its transition probability

[20] we have seen the utility of the "word" representation later the section on multigrams.

matrix **A**, which is full. For the sake of compatibility with previously used HTK conventions, a non-emitting entry state 1 having equal transition probabilities to all emitting states:

$$a_{1i} = k = \frac{1}{N-2} \quad \text{for} \quad 2 \leq i \leq N-1, \tag{4.86}$$

and exit state N with probabilities on incoming arcs equal to the same constant:

$$a_{iN} = k = \frac{1}{N-2} \quad \text{for} \quad 2 \leq i \leq N-1, \tag{4.87}$$

should be added, so that the transition probability matrix **A** is written:

$$\mathbf{A} = \begin{bmatrix} 0 & k & \cdots & k & 0 \\ 0 & a_{22} & \cdots & a_{2N-1} & k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n-1,2} & \cdots & a_{N-1,N-1} & k \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{4.88}$$

In automatic unit determination, ergodic HMM is a valuable alternative to vector quantization (subsection 4.5.1) for clustering of vectors because:

1. it allows for more complex cells in P-dimensional parameter space, than simple Voronoi regions based on minimum distance. If the emission PDFs of EHMM are given by mixtures of Gaussians, it is possible to construct arbitrarily complex cell.

2. the dependencies between vectors (or TD events) to be clustered are taken into account by transition probability matrix **A**. If we want to use only the first feature of EHMM (e.g. obtaining complex cells) and do not want to work with inter-frame dependencies, it is sufficient to fix all transition probabilities equal to a constant:

$$a_{ij} = c \quad \text{for} \quad 2 \leq i \leq N-1,\ 2 \leq j \leq N-1, \tag{4.89}$$

   and do not allow the algorithms to reestimate them.

The working with EHMM is somehow different from "classical" left-right HMMs described above. One must take into account, that only *one* model is at our disposal and that rather than finding maximum likelihood alignment path among several models, we are interested by state-to-observations alignments along the optimal path in EHMM. Also, the EHMM *training* presents differences compared to other HMMs: as we want to cluster the data into a-priori unknown classes, the EHMM training consists of the following steps:

1. *Initialization* of means, variances and transition probabilities. As we do not dispose of transcription, it is good idea to initialize the means using previously trained VQ or randomly, to put reasonably chosen constants to covariance matrices and to set all transitions equiprobable (Equation 4.89).

2. The *estimation* itself is very similar to VQ codebook training described in paragraph 4.5.2 (steps (a)–(d) of the inner loop in the algorithm). If the set of all parameters of ergodic model M is denoted λ (including transition probabilities and emission PDF parameters), the algorithm is equivalent to context-free re-estimation (subsection 4.7.2):

   (a) denote initialized parameters λ^0, set iteration number $m = 0$.

   (b) with parameter set λ^m, evaluate "appertaining" functions $L_j(\mathbf{o}_t)$ using forward and backward alignment costs.

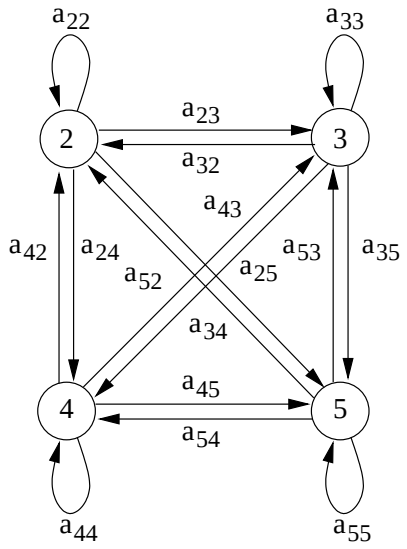


Figure 4.10: Ergodic HMM with 4 emitting states. Non-emitting states Nos. 1 (entry) and 6 (exit) with equal transition probabilities to and from all emitting states are not shown. Each emitting state carries emission PDF $b_i(\mathbf{o}_t)$.

(c) reestimate parameters using Equations 4.84 and 4.85. Denote new parameters λ^{m+1}.

(d) examine the change of likelihoods $\mathcal{L}(\mathbf{O}|\lambda^m)$ and $\mathcal{L}(\mathbf{O}|\lambda^{m+1})$; if this change is insignificant, or if maximal number of iterations was reached, exit, otherwise set $m = m + 1$ and return to Step (b).

In Step (c), transition probabilities can be re-estimated or imposed to be fixed.

The recognition (or quantization) is performed simply by searching the alignment path X corresponding to likelihood $\mathcal{L}^\star(\mathbf{O}|M)$. If the token passing paradigm is employed, it is necessary to complete the token contents with information about "visited" states. Similar structure as word-link record can be created and named *state-link record (SLR)*.

An important advantage of EHMM clustering is the possibility to "unfold" partial paths within EHMM to create initial "multi-symbol" HMMs. In our example of Figure 4.10, an HMM with three emitting states $[2, 4, 5]$ and skip of the central state could be constructed by taking into account the transition probabilities a_{22}, a_{24}; a_{44}, a_{45}; a_{55} and emission PDFs b_2, b_4, b_5 of pre-trained EHMM. This interesting feature was investigated by Deligne [31, 32, 29] in her joint-multigram recognition experiments with HMMs representing continuous multigrams (subsection 4.6.4).

4.7.5 HMMs in automatic determination of units, refinement

This subsection deals with engagement of HMMs in our unit derivation scheme and mainly with the feature of iterative refinement of model set by successive re-segmentations of training corpus and re-estimations of parameters. The most important resource for HMM training are automatically derived transcriptions of the training DB. They can be obtained by temporal decomposition with successive clustering of events into L classes. In this case, we should train L models. In some experiments, this initial segmentation and labelling were post-processed by discrete multigrams (see subsection 4.6.3), so that the segments were containing 1 to n original TD events (where n is the maximal multigram length) and the total number of models to train equals to multigram dictionary size Z. From the point of view of HMM training, those two

approaches do not differ except for language model alterations in the HMM segmentation step, if LM use is allowed in the training. To formalize the training algorithm, denote the initial set of transcriptions of training corpus $\mathbf{T}^0$, the initial number of models to train Z^0 and the initial language model LM^0. The procedure can be then written as follows[21]:

1. *Initialization.* Using original transcriptions $\mathbf{T}^0$, train Z^0 models using all three training steps described in subsection 4.7.2: initialization, context-free and context-dependent training. Denote the set of all HMM parameters

$$\mathbf{\Lambda}^0 = \{\lambda_i^0\} \quad \text{for} \quad 1 \leq i \leq Z^0. \tag{4.90}$$

   Set iteration number $m = 0$.

2. *Segmentation.* Using Z^m models with parameters $\mathbf{\Lambda}^m$ recognize, e.g. segment the training corpus to produce transcriptions $\mathbf{T}^{m+1}$. Denote the total alignment likelihood $\mathcal{L}_{tot}^{\star m+1}$. According to the type of experiment, the language model LM^m may be used or not in the recognition. In case it is used, the alignment likelihood $\mathcal{L}_{tot}^{\star m+1}$ reflects also LM probabilities.

3. *Parameter re-estimation.* Determine new number of models Z^{m+1}: according to count of segments assigned to model M_i in the recognition step, model M_i can be retained (if sufficient number of segments carry its label) or discarded (if the number is insufficient for the following re-estimation step). Reestimate new set of parameters $\mathbf{\Lambda}^{m+1}$ using transcriptions $\mathbf{T}^{m+1}$. Old parameters $\mathbf{\Lambda}^m$ can be used as initial values, or the estimation can be started from scratch (e.g. including initialization of parameters — Step 1 in subsection 4.7.2). Denote the total Baum-Welch likelihood obtained on the set of all training utterances $\mathcal{L}_{tot}^{m+1}$.

4. *Language model re-estimation.* If desired, the LM can be "upgraded" using counts from the recognition step. The parameters of resulting re-estimated LM are denoted LM^{m+1}. Subsections 4.7.6 and 4.6.4.3 deal with this point in more detail.

5. *Termination test.* Termination of the algorithm can be based on different criteria. As the above procedure is not very modest concerning time and computer resources, often a fixed and rather limited (such as 5) number of iterations was imposed. Another possibility is to investigate differences of recognition likelihoods $\mathcal{L}_{tot}^{\star m+1} - \mathcal{L}_{tot}^{\star m}$ or re-estimation likelihoods $\mathcal{L}_{tot}^{m+1} - \mathcal{L}_{tot}^{m}$, and to stop the algorithm if the change is relatively small. The optimal solution would a measure determining the suitability of resulting model set for the target application, but such criterion was unfortunately not found. For any of above mentioned methods, if the decision is to continue, set $m = m + 1$ and return to Step 2.

When comparing the integrity of clusters derived by above mentioned method in listening tests (comparing signal segments labelled by the same model M_i), we found, that the consistency increased.

Another possibility for stochastic representation of segments would be segmental HMMs described in articles of Mari Ostendorf and her group [69, 37, 6] (see also section 3.4). Here, the means and variances of one HMM state are not constant but following a polynomial trajectory, which allows for more precise alignment with observation vectors. This type of HMMs has not been used in our experimental work but we hope to work on this topic in the future.

[21] Similar algorithm of iterative refinement of units was described by Bacchiani et al. in [6], with maximum likelihood (ML) as the sole measure of description efficiency. The ML criterion was presumably used also in termination decision taking.

4.7.6 Language models used with HMMs

Language models (LM) are important part of continuous speech recognizers [66, 65] responsible for finding the a-priori probability of words. On contrary to isolated word case, where these probabilities are unconditioned (see Equation 4.73), in continuous speech recognition (CSR), the Bayes rule is written:

$$\mathcal{L}(w_1^N|\mathbf{O}) = \frac{\mathcal{L}(\mathbf{O}|w_1^N)\mathcal{L}(w_1^N)}{\mathcal{L}(\mathbf{O})}. \tag{4.91}$$

Here again, the likelihood of data $\mathcal{L}(\mathbf{O})$ is unknown but constant, so that we can concentrate on the numerator of Equation 4.91. The symbol w_1^N denotes the sequence of words $[w_1, w_2, \ldots, w_N]$ being aligned with the data. $\mathcal{L}(\mathbf{O}|w_1^N)$ is the likelihood of observation sequence $\mathbf{O}$ while produced by these words and $\mathcal{L}(w_1^N)$ is the key-point of language models: a-priori likelihood of word sequence w_1^N. We suppose, that words are conditioned by all predecessor words[22] from the beginning of utterance, so that the joint likelihood $\mathcal{L}(w_1^N)$ can be rewritten:

$$\mathcal{L}(w_1^N) = \prod_{n=1}^{N} \mathcal{P}(w_n|w_1^{n-1}), \tag{4.92}$$

In practice, the entire history:

$$h_n = w_1^{n-1} = [w_1, w_2, \ldots, w_{n-1}] \tag{4.93}$$

can not be taken into account and is *truncated*. For so called word m-grams it is truncated to $(m-1)$ predecessor words, for grammar models to the current sentence and for trigger or cache models to the current paragraph. Here, only m-grams are discussed.

The simplest in the class of m-gram language models are *unigrams* or one-grams. As the history is of zero length, the probability of word w_n depends only on the word itself and the likelihood of word sequence w_1^N is given by a product of a-priori probabilities of words:

$$\mathcal{L}(w_1^N) = \prod_{n=1}^{N} \mathcal{P}(w_n). \tag{4.94}$$

This LM can be estimated easily on the training set by evaluating numbers of occurrences of words in transcriptions:

$$\mathcal{P}(w_i) = \frac{N(w_i)}{N}, \tag{4.95}$$

where $N(w_i)$ is the number of occurrences of word w_i and N is the total number of words in the corpus. Such LM can be easily re-estimated within iterative refinement of model set, as described in previous subsection (Step 4). For this type of models, we suppose, that the recognizer can not be confronted with an unknown word w_i, so that all probabilities $\mathcal{P}(w_i)$ must have non-zero values. Unigram LM can be transparently incorporated into token passing paradigm (subsection 4.7.3): each time a token enters word w_i, its value is augmented by log a-priori probability of that word: $\log \mathcal{P}(w_i)$.

Widely used LM, already taking into account the history, are word *bigrams*. The a-priori probability of word w_n depends on its predecessor:

$$\mathcal{P}(w_n|h_n) = \mathcal{P}(w_n|w_{n-1}). \tag{4.96}$$

[22] Humans seem to be able to make use also of *future* words in their "human language models", but this seems to be still outside the capabilities of computer algorithms...

The estimation of bigrams is an interesting problem, but as bigrams were not used in our work, we shall refer reader to books [75, 74], or for example to articles of H. Ney [66].

Multigrams differ considerably from the two previously described LMs. The term "multigram" would suggest that on contrary to m-grams, we are handling "variable history m-grams" . This approach called x-grams was investigated by Bonafonte and Mariño in [15], but multigrams are providing segmentation into variable length units rather than overlapping variable histories of words. As they are assumed to be statistically independent, they lead to use of *unigram LM* when incorporated into the HMM framework. However, they play an important role in the creation of units (HMMs based on several original TD events) or in making "words" of units, approaching in this the automatically found units to *words* in a large vocabulary continuous speech recognizer.

4.7.7 Practical issues of HMMs

There are important choices to be done in the experimental training and recognition using HMMs. We have not performed exhaustive investigation on different features of HMM, the "standard" options were chosen most of the time.

First choice to be done is the parameter set. Classically used are 39-element feature vectors of 12 parameters, 12 Δparameters and 12 $\Delta\Delta$parameters, energy, Δenergy and $\Delta\Delta$energy. In the experiments, we have used simpler features with only Δcoefficients. Those data formed three streams: 1) stream of LPCC coefficients of width P, where P is the number of coefficients, 2) stream of ΔLPCC coefficients (width P) and 3) stream of E and ΔE (width 2).

The emission distributions were chosen mono-Gaussians, as the experiments were speaker-dependent. An important issue in LVCSR is the *tying* of parameters, limiting the storage and computational load of both training and Viterbi alignment. In our experiments, the tying was not used, and all parameters were independent. Finally, there are methods for limitation of "beam width", in other words, of number of models and states active in the re-estimation or recognition. We have used a simple mechanism offered by the HTK toolkit: during the computation of backward alignment probabilities β, the states for which β falls below a threshold (relative with regard to the maximum value β_{max} of the utterance), are "switched off" and not taken into account during forward probability α computations. This method can speed-up both the training and the recognition.

When the uni-gram language was used in the Viterbi decoding, it was not used with weight equivalent to acoustic modeling, but its contribution was weighted by *language model factor* γ, so that Equation 4.91 can be re-written:

$$\mathcal{L}(w_1^N|\mathbf{O}) = \frac{\mathcal{L}(\mathbf{O}|w_1^N)\mathcal{L}(w_1^N)^\gamma}{\mathcal{L}(\mathbf{O})}. \tag{4.97}$$

In the coding experiments, the influence of LM scale factor γ on resulting bit rate and decoded speech quality was investigated.

Chapter 5

Application No. 1: Very Low Bit-Rate Coding

This chapter is devoted to the first application of automatically derived units, tested in experiments: the very low bit-rate (VLBR) coding. Its first two sections section can be understood as a complement of the theoretical Chapter 4. They concentrate on the implementation of ALISP framework in the coding. The following sections describe two sets of experiments, conducted with the data from databases PolyVar and Boston University Radio Speech Corpus. Our early experiments concerning the direct application of multigrams to spectral envelope coding are not covered in this chapter: interested reader should refer to our papers, presented at Rencontres de Jeunes Chercheurs en Parole'95 [95], IAPR workshop'96 [97], Journées d'Étude sur la Parole'96 [96] and at ICASSP'97 [98]. The last mentioned article is included in Appendix D.

5.1 Recognition–synthesis ALISP vocoder

When studying the historical development of speech coding methods [53, 87], one finds, that the very-low bit rate region (hundreds of bits per second (bps)), is populated uniquely by *recognition-synthesis* (or *phonetic*) vocoders. Those coders approach the lower limit of coding rate, imposed by the *phonetic rate*. This figure is giving the average number of bits per second necessary for encoding the phonetic information of spoken speech, without any prosody or voice quality encoding. According to classical works [76, 74], it is approximately 50 bps. Some phonetic vocoders have been mentioned in the bibliography chapter (section 3.2), along with their drawbacks: mainly the need of phonetically labelled database for their training.

In our work, the recognition-synthesis framework was adopted, but the determination of units is done automatically, by ALISP tools. The synopsis of our coder is given in Figure 5.1. The main difference from classically used schemes is the availability of parts of the training corpus to the decoder.

Possible *applications* of such coder are:

- *archiving of speech material*. There are many applications necessitating storing of hours of mono-speaker data where the intelligibility is requested, but the quality is a secondary issue: black boxes of aircrafts, recordings of parliamentary or justice court debates, monitoring of TV and radio stations, etc. In addition, these tasks are often connected to the *recognition* (word spotting, search for key-words in a talk or in a telephone call). Using this representation of the signal, we can provide in the same time the information necessary to *reconstruct* the speech and a *symbolic representation* useful for the recognition.

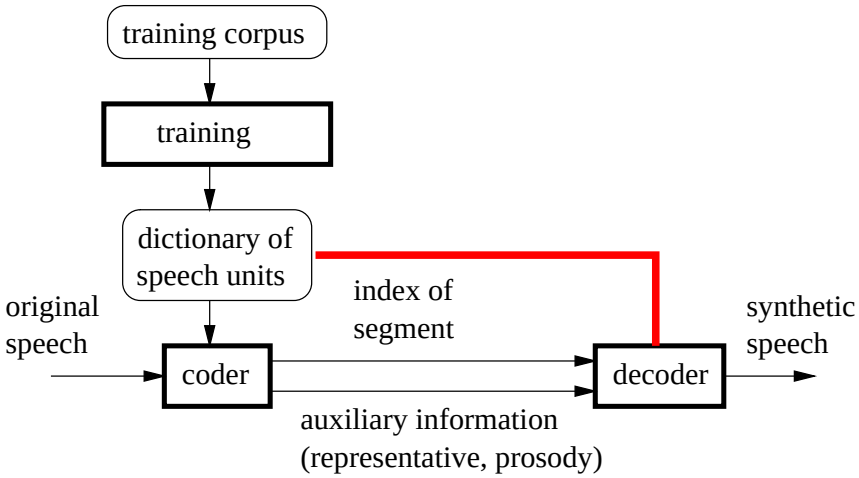


Figure 5.1: Synopsis of the proposed coder.

- *Internet telephony.* A lot of network applications make use of the voice transmission (NetTalk, CoolTalk, MultiCast, ...) [2], but the coding schemes employed are generally based on low-delay CELP developed for Global System of Mobile Communications (GSM) and do not allow for voice coding below several kbps. In case of limited network bandwidth or simultaneous transmission of more types of information (speech, text, graphics, video), there is a need for decreasing the rate allocated for speech.

- *protection of packet speech transmission.* In high-quality speech transmission over the Internet, a packet loss can appear in case of channel saturation or of changing the packet way from one "route" to another. If, for example, packets of one second of high quality speech are lost, the decoder is not able to recover the information. In case we complete the high quality coding with one packet containing the information for the *entire second* of speech (with worse quality) each 100 ms, there is a chance, that the information will be recovered in the decoder. Moreover, the speech unit dictionary can be derived dynamically from high-quality packets. This back-up scheme could thus replace existing algorithms of forward error correction (FEC) relying currently only on parity and Reed-Solomon codes [80].

- the application cited above are closely linked to *scalable coders*, where the high and low rate information are not transmitted *simultaneously* but where a high quality coder can "back-up" to lower rate scheme in case of problems in the channel.

5.2 Technical aspects of ALISP vocoder

Some definitions should be made before any description of technical details of the proposed scheme. In this chapter, the automatically derived speech units are called *coding units (CU)*. In experiments, the speech was always re-synthesized using *examples* of those units drawn from the training corpus called *representatives*. However, theoretically it is possible (and we believe, that useful) to define *synthesis units* derived from coding ones. Schematically, this approach is presented in Fig. 5.2. The following subsections contain details on different levels of units, and on two approaches of *synthesis* used in the experiments.

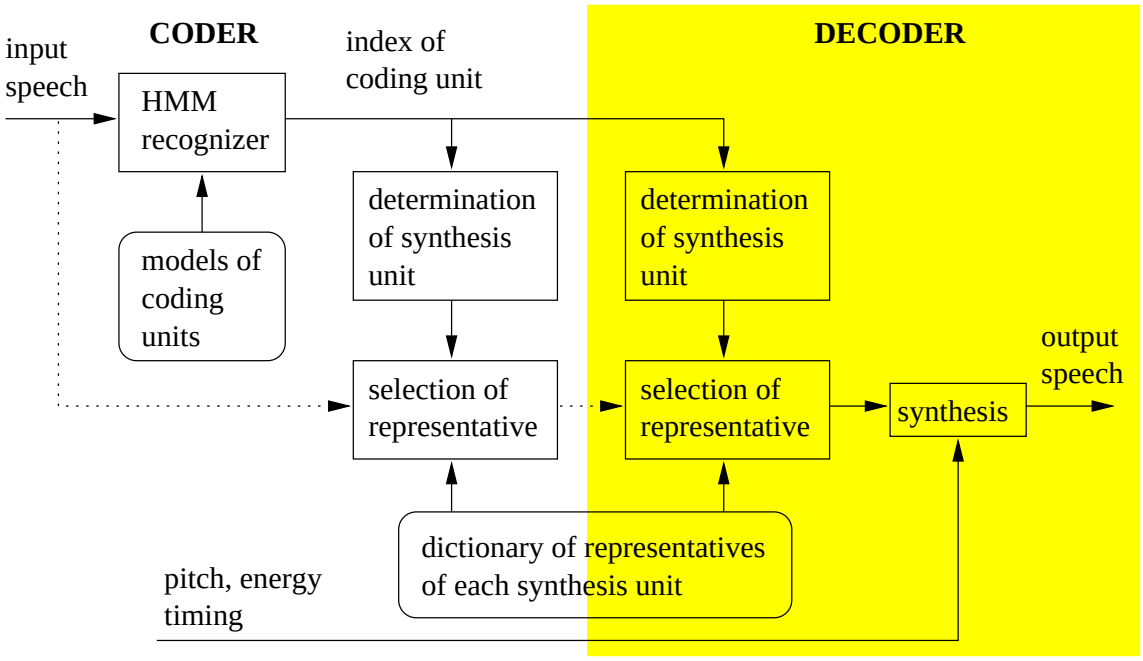


Figure 5.2: Coding units, synthesis units and representatives in the coder and decoder. The information about chosen representative can be either transmitted (dotted line) or re-created in the decoder.

5.2.1 Choice of synthesis units and representatives

Synthesis units (SU) are derived from coding units (CU). Each synthesis unit u_i can have one or more *representatives* in the training corpus $r_{i,j}$, whose number may be variable and depending on the unit: n_{ri}. For the set of coding units $\{x_i\}$, the choice of u_i's may be of two kinds:

1. synthesis unit u_i corresponds to coding unit x_i. Representatives of u_i are found in the training corpus by searching signal parts (or parameter matrix segments) labelled by x_i. Their choice is driven by one or more criteria described below.

2. Synthesis units u_i correspond to *transitions* of coding units x_i. This approach is not useless, if one takes into account, that coding units are mostly derived from TD events, having central stable part and transitions on its borders. If, for a string of coding units: $X = [ABADDBAC]$, we define synthesis units not as: $u_1 = A$, $u_2 = B$, $u_3 = C$, $u_4 = D$, but as transitions: $u_1 = AB$, $u_2 = BA$, $u_3 = AD$, $u_4 = DD$, $u_5 = DB$, $u_6 = AC$, we will hopefully obtain smoother concatenation in the decoder than when using units driven directly by the coding. The disadvantage of this approach is an increased number of synthesis units and not straightforward algorithm of their generation when larger coding units (such as multigrams of original TD events, or HMMs trained on those multigrams) are used.

When the methodology of synthesis units is fixed, another important question is how, for given synthesis unit u_i, choose its representatives $r_{i,j}$ in the training corpus. We suppose, that the number of representatives per unit n_{ri} is fixed a-priori and that it can be only decreased, if the number of representatives in corpus is not sufficient. Several approaches come into account for the choice of representatives:

1. If the synthesis method does not allow for *time modifications*, representatives $r_{i,j}$ should have different times to encounter for elocution speed variations. However, those primitive

synthesis techniques were used only in early steps of our work for segment set testing, so that this criterion is not of great importance.

2. The representatives should be chosen as long as possible. When converted to shorter ones, the down-sampling is easier and less error-prone than interpolation or frame repetition of shorter segments to longer ones. On the other hand, by choosing long representatives, one risks to introduce spurious sounds sometimes appearing in those long segments, which are "averaged-out" by a favorable global distance (or likelihood).

3. To let representatives $r_{i,j}$ be consistent with given synthesis unit u_i, they should be chosen with a criterion of minimum distance (or maximum likelihood) when compared to synthesis unit centroid (or model). It is easy to evaluate this criterion when synthesis units correspond to coding ones: $u_i \equiv x_i$, so that the sequence of means (multigrams with distance, MGs as Gaussian sources) or an HMM is readily available. In case of synthesis units "crossing" transitions of two coding ones, this template must be artificially created by concatenation of "half" mean-sequences or by creation of new HMM from two "half"-HMMs (Figure 5.3). When concatenating parts of two HMMs, attention should be payed to duration distributions of "cut" states. In our case (no explicit duration distributions), these are given implicitly as:

$$\mathcal{P}(t_i) = (a_{ii})^{t_i - 1}(1 - a_{ii}), \tag{5.1}$$

where t_i is the occupation time of i-th state. The mean occupation time is then according to Rabiner [77]:

$$\bar{t}_i = \frac{1}{1 - a_{ii}}. \tag{5.2}$$

If we want first and last emitting states of transition model AB to have half mean durations than original 3rd states of models A and B, their loop transition probabilities should be determined in order to satisfy:

$$\bar{t}^{AB}_{22} = \frac{\bar{t}^{A}_{33}}{2}, \quad \bar{t}^{AB}_{55} = \frac{\bar{t}^{B}_{33}}{2}. \tag{5.3}$$

The solution of this problem is to set transition probabilities a^{AB}_{22} and a^{AB}_{55} to:

$$a^{AB}_{22} = 2a^{A}_{33} - 1, \quad a^{AB}_{55} = 2a^{B}_{33} - 1. \tag{5.4}$$

Precautions should be taken if the term $2a_{33} - 1 < 0$. In this case, it is worth to set corresponding transition probability a^{AB}_{22} or a^{AB}_{55} to zero, ensuring in this way only one vector to be emitted by the corresponding state with an immediate transition to the next one.

4. to let the representatives be able to account for different neighboring events, $r_{i,j}$'s for an u_i should be chosen as different as possible.

It can be seen, that those four criteria are not only non-exclusive, but some (3 and 4) are even contradictory. The ultimate criterion would be an objective measure of decoded speech quality, but, as it has been mentioned in the end of subsection 4.7.5, such criterion has not been found.

5.2.2 Representatives determination for the coded speech

The first step in the coding is the transcription of input speech with coding units performed by the Viterbi search (outlined in section 4.7). The following step is not of less importance:

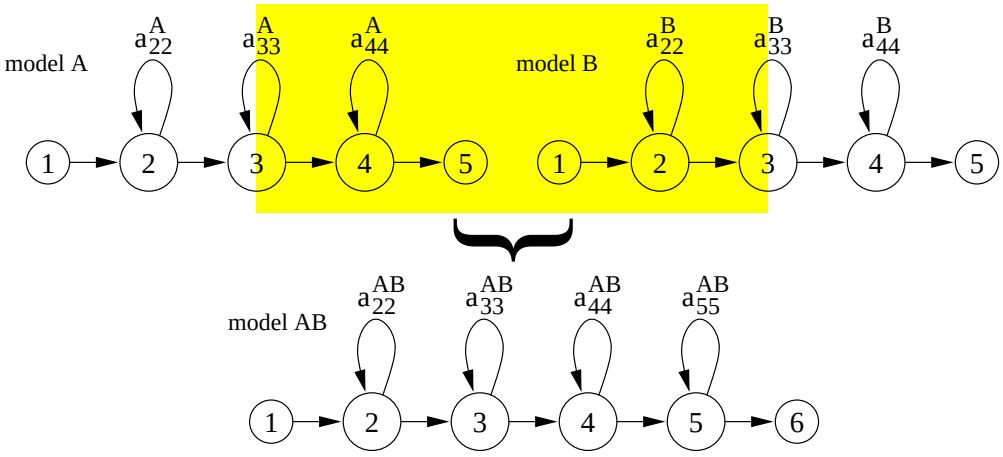


Figure 5.3: Elaboration of an HMM for a transition of two original HMMs used in evaluation of representative suitability using maximum likelihood.

assigning a representative to each synthesis unit (as we have seen, those can be equivalent to coding ones or to their transitions). The choice of representative is transmitted to the decoder together with coding unit identification. While the transcription into coding units is a pure "recognition-like" operation, in the generation of representatives we are returning to notions of similarity of representatives with the incoming speech and to smoothness criteria.

We are going to denote the signals of representatives $r_{ij}(t)$, their parametric form $\mathbf{R}_{ij}$, while the segment of input speech to be compared with a representative will be denoted $s(t)$ (signal) and $\mathbf{S}$ (parameter matrix). We also suppose, that the synthesis unit u_i is already known, so that i is fixed. With this notation, the criteria for representatives choosing can be summarized as follows:

1. When working with a method unable to modify the timing of segments, the representative with the length most closed to the original should be chosen:

$$r_{ij}(t) = \arg\min_{\forall j} |l[s(t)] - l[r_{ij}(t)]|, \tag{5.5}$$

   where $l[\cdot]$ denotes the length of signal in samples. As we have already mentioned, this primitive synthesis method was used only in early test steps.

2. The parameter matrix of representative should be as closed as possible to the input speech segment. The maximal likelihood approach is not useful here, as we do not dispose of model for given incoming segment (note, that models are available only for coding units and those are already known). Therefore, minimal distance approach comes into account. As the lengths of matrices $\mathbf{S}$ and $\mathbf{R}_{ij}$ are generally not equal, their distance is evaluated using Dynamic Time Warping (DTW) [52] as:

$$D_{DTW}(\mathbf{S}, \mathbf{R}_{ij}) = \frac{d_c(K, L)}{K + L}, \tag{5.6}$$

   where K and L are respectively lengths of matrices $\mathbf{S}$ and $\mathbf{R}_{ij}$. For the cumulated distance $d_c(\cdot, \cdot)$ evaluation, one must define elementary distance of two vectors belonging to $\mathbf{S}$ and $\mathbf{R}_{ij}$ as:

$$d(k, l) = d_e(\mathbf{s}_k, \mathbf{r}_{ijk}), \tag{5.7}$$

where $d_e(\cdot,\cdot)$ stands for standard Euclidean distance. The cumulated distance can then be evaluated for $1 \leq k \leq K$ and $1 \leq l \leq L$, with initial settings: $d_c(0,0) = 0$, $d_c(0, l \neq 0) = \infty$, $d_c(k \neq 0, 0) = \infty$, as:

$$d_c(k,l) = \min \left\{ \begin{array}{l} d_c(k-1,l) + d(k,l), \\ d_c(k,l-1) + d(k,l), \\ d_c(k-1,l-1) + 2d(k,l) \end{array} \right\}. \tag{5.8}$$

The optimal representative should minimize the DTW distance given in Equation 5.6:

$$r_{ij} = \arg\min_{\forall j} D_{DTW}(\mathbf{S}, \mathbf{R}_{ij}). \tag{5.9}$$

3. The concatenations of representatives should be as *smooth* as possible. We will denote the counter of representatives t, and the representative for t-th segment r_{i_t,j_t} (similar notation was for example used to denote t-th coding unit x_{it} in Equation 4.44). This representative should minimize the distortion on the transition from the previous representative to the current one:

$$r_{i_t,j_t} = \arg\min_{\forall j} d_e(\mathbf{r}_{i_{t-1},j_{t-1},L_{t-1}}, \mathbf{r}_{i_t,j_t,1}), \tag{5.10}$$

where $\mathbf{r}_{i_{t-1},j_{t-1},L_{t-1}}$ is the last vector of previous optimal representative of length L_{t-1}, and $\mathbf{r}_{i_t,j_t,1}$ is the first vector of the current representative. The distance $d_e(\cdot)$ is Euclidean.

The above described minimization checks only for the transition in the beginning of current representative, so that its end can be "hard to concatenate". Optimal solution is a minimization of global transition distortion given for chain R of N concatenated segments:

$$D_{tr}(R) = \sum_{t=2}^{N} d_{en}(\mathbf{r}_{i_{t-1},j_{t-1},L_{t-1}}, \mathbf{r}_{i_t,j_t,1}). \tag{5.11}$$

The optimal string of representatives is then given as:

$$R^\star = \arg\min_{\{R\}} D_{tr}(R), \tag{5.12}$$

where $\{R\}$ is the set of all possible representative chains. This problem can be efficiently solved by Viterbi algorithm. Note that the distance used in Equation 5.11 must be normalized, in order to sum up comparable distances to obtain the global transition distortion $D_{tr}(R)$:

$$d_{en}(\mathbf{x},\mathbf{y}) = \sqrt{\frac{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})}{\|\mathbf{x}\|\ \|\mathbf{y}\|}}. \tag{5.13}$$

As we can see, the criteria 2 and 3 are once again not exclusive, and suggest an algorithm optimizing jointly the global distance of representatives and input sequences, and transitions distortions.

5.2.3 Synthesis using concatenation of signal segments

The synthesis consists in concatenating signals of representatives to create the output speech. The first and simplest tested method was the concatenation of *signal segments* belonging to representatives. One can see that this method is quite primitive, not allowing for any timing and/or pitch modification. It was used in early experiments to verify the usefulness of automatically

derived units in the coding. Only energy modification can be performed; simple mean energy correction was employed, where the correction factor was calculated as:

$$c_{et} = \frac{\frac{1}{L}\sum_{l=1}^{L} e(\mathbf{r}_{i_t,j_t,l})}{\frac{1}{K}\sum_{k=1}^{K} e(\mathbf{s}_{t,k})}, \tag{5.14}$$

where $e(\cdot)$ denotes the frame energy defined in the beginning of this chapter in Equation 4.10. Then, the synthetic signal is given by concatenation of representative signals multiplied by corresponding correction factors:

$$y(t) = [\sqrt{c_{e1}} r_{i_1,j_1}(t), \sqrt{c_{e2}} r_{i_2,j_2}(t), \ldots, \sqrt{c_{eN}} r_{i_N,j_N}(t)]. \tag{5.15}$$

5.2.4 Linear prediction based synthesis

The most straightforward method of a parametric synthesis, more suitable for timing and pitch alterations, is the using of simple speech production model composed of excitation source and filter, as it was outlined in the introductory section on speech parameterization 4.3.

5.2.4.1 Principles of LP synthesis

In the easiest implementation, the excitation source is producing Dirac pulses with pitch period T_0 for voiced sounds and white noise for the unvoiced ones. The synthesis filter $\hat{H}(z)$ is given as:

$$\hat{H}(z) = \frac{1}{\hat{A}(z)}, \tag{5.16}$$

where the hat denotes, that coefficients of polynomial $\hat{A}(z)$: $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_P$ are not original but quantized. In our case, this quantization involves their derivation not from the input speech, but from the parametric representation (LPC-cepstrum) of the corresponding representative. The c to a conversion formula can be easily derived from Equation 4.8 and is written:

$$\hat{a}_n = -\hat{c}_n - \sum_{k=1}^{N} \frac{k}{n} \hat{c}_k \hat{a}_{n-k}, \tag{5.17}$$

for $1 \leq n \leq P$, where P is the filter order and $\hat{c}_n$ are LPC-cepstral coefficients of the representative. If the original parameterization (subsection 4.3.1) was done on frames of length l_f with overlapping r_f, it is sufficient when the synthesis produces only the number of samples equivalent to the window shift $w_s = l_f - r_f$ for each frame. For simplicity, we will call those non-overlapping parts of frames "ws-frames" (see illustration in Figure 5.4). The excitation $x_n(t)$ for n-th ws-frame is given as follows: if the voicing flag is 0, it is given by w_s samples of Gaussian white noise with unit variance. If the n-th frame is voiced, the excitation is given as a periodic sequence of Dirac pulses $\delta_{T_0}(t)$, where the samples for $t < 0$ are zero:

$$\delta_{T_0}(t) = \begin{cases} 1 & \text{for} \quad t = kT_0,\ k \geq 0, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{5.18}$$

This sequence has total length w_s and initial "phase" τ, which is given as the difference of pitch period and the distance of previous Dirac pulse (time T_{ld}) from the end of already synthesized signal T_e:

$$\tau = T_0 - (T_e - T_{ld}). \tag{5.19}$$

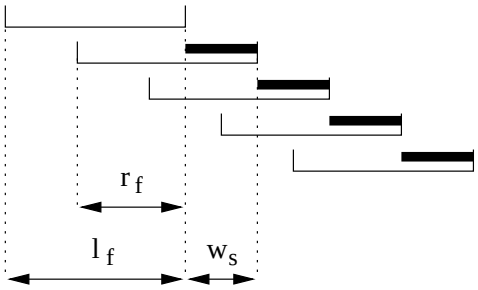


Figure 5.4: Illustration of frames and "window-shift" ws-frames in the synthesis. l_f denotes the frame length, r_f the frame overlapping and w_s the window shift. ws-frames are drawn as solid bold lines.

It is obvious, that the interval of previous pulse and the first one in currently synthesized ws-frame must be equal to the pitch period. If the previous frame is unvoiced, we set $\tau = T_0$.

The excitation $x_n(t)$ is then filtered by synthesis filter $\hat{H}(z)$. The initial conditions of this filter are kept from the synthesis of last ws-frame. If the pre-emphasis was applied before computation of LPC coefficients (Equation 4.3), it is worth to deemphasize the signal by inverse filter:

$$H_{deem}(z) = \frac{1}{1 - az^{-1}}. \tag{5.20}$$

The last step is the energy adjustment. If we denote the energy of filtered and deemphasized excitation e_{hd} (normalized by ws-frame length), we can obtain the energy correction constant as:

$$c_{en} = \frac{\hat{e}_n}{e_{hd}}, \tag{5.21}$$

where $\hat{e}_n$ is the desired energy.

The resulting signal is then given by the concatenation of synthesized ws-frames multiplied by corresponding amplitude corrections:

$$y(t) = [\sqrt{c_{e1}} y_{hd_1}(t), \sqrt{c_{e2}} y_{hd_2}(t), \ldots, \sqrt{c_{eN}} y_{hd_N}(t)], \tag{5.22}$$

where $y_{hd_n}(t)$ stands for excitation $x_n(t)$ processed by synthesis and deemphasis filters.

5.2.4.2 Timing modification in LP synthesis

When a representative r_{ij} is chosen for the input sequence s, the lengths of their parametric representations K (length of $\mathbf{S}$) and L (length of $\mathbf{R}_{ij}$) are generally different. In LP synthesis, it is possible to correct easily $\mathbf{R}_{ij}$ in order to correspond to $\mathbf{S}$ in length.

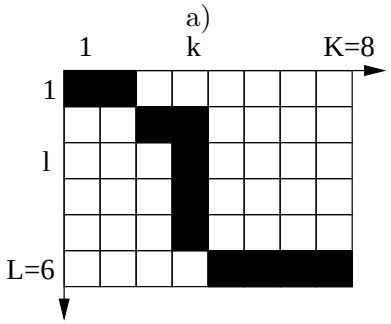
The most efficient way is to redistribute vectors $\mathbf{r}_{ijk}$ to the length K along the optimal DTW path, obtained in Equation 5.8. The aligned representative $\mathbf{R}^a_{ij}$ is then given by the following transformation:

$$\mathbf{R}^a_{ij} = \mathbf{R}_{ij}\mathbf{T}, \tag{5.23}$$

where $\mathbf{R}^a_{ij}$ has dimensions $P \times L$, the representative $\mathbf{R}_{ij}$ has dimensions $P \times K$ and $\mathbf{T}$ is a $L \times K$ transform matrix. This matrix is sparse and contains ones distributed according to the optimal DTW path (see Figure 5.5 for illustration). On contrary to the trajectory, this matrix can hold only one "1" in each column to prevent summation of neighboring parameter vectors.

This method of aligning representatives with input segments was used in the tests, but in practice, the transmission of transformation matrices would necessitate considerable rate[1].

[1] more than 1 bit/frame, if one takes into account, that due to shrinking vertical parts in the the path into a single "1" in the transition matrix, the step on representative vectors may be not only 0 or 1, but also greater.



b)

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

c)

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 5.5: Illustration of DTW path (panel a) and of its conversion to transformation matrix $\mathbf{T}$ (panel b). Note, that a vertical segment in the path must be converted to only one "1" in corresponding column of $\mathbf{T}$. Panel c) shows the linear path obtained for the same dimensions K and L using Equation 5.24.

A more economic method of timing modification is the construction of *linear alignment path* between $\mathbf{S}$ and $\mathbf{R}_{ij}$ using only respective lengths K and L. In this case, the position of "1" in k-th column of transformation matrix $\mathbf{T}$ is given by simple linear alignment function:

$$l_k = \left\langle (k-1)\frac{L-1}{K-1} \right\rangle + 1, \tag{5.24}$$

where $\langle \cdot \rangle$ denotes the rounding to closest integer. Figure 5.5c illustrates the construction of transformation matrix $\mathbf{T}$ for lengths $K = 8$, $L = 6$. On contrary to the previous case, only the transmission of the length K of input segment is necessary, the decoder knows the representative's length L.

5.2.5 Rate considerations

The coding *rate* is the number of bits per second (bps) necessary to transmit the speech. It can be either *fixed* or *variable*. For variable rate coders, it is useful to define the *mean rate*. In case of coding indices of units from a dictionary of finite length Z, the units are denoted M_i and their lengths in frames $l(M_i)$ (with the frame shift T_f). There are two possibilities of encoding the indices of those units:

- using *uniform coding* where $\log_2 Z$ bits are necessary to encode each unit M_i. The mean rate in bps is then given by:

$$R_u = \frac{\log_2 Z \sum_{i=1}^{Z} c(M_i)}{T_f \sum_{i=1}^{Z} c(M_i) l(M_i)}, \tag{5.25}$$

where $c(M_i)$ is the number of occurrences of M_i in the encoded string. We recognize the total number of bits necessary in the numerator and the total length in the denominator.

- using *entropy coding*, where the number of bits to encode each M_i depends on its a-priori probability: $-\log_2 \mathcal{P}(M_i)$. The mean rate in bps is then given by:

$$R_e = -\frac{\sum_{i=1}^{Z} c(M_i) \log_2 \mathcal{P}(M_i)}{T_f \sum_{i=1}^{Z} c(M_i) l(M_i)}. \tag{5.26}$$

  In case the distribution of units in the encoded signal is the same as in that used for determination of a-priori probabilities, this quantity should be closed to the entropy of dictionary (evaluated on the training corpus) normalized by the mean length of sequences:

$$H_{norm} = -\frac{\sum_{i=1}^{Z} \mathcal{P}(M_i) \log_2 \mathcal{P}(M_i)}{T_f \sum_{i=1}^{Z} \mathcal{P}(M_i) l(M_i)} \quad \text{in bps.} \tag{5.27}$$

In the coding experiments, we have concentrated mainly on the *rate of coding units*. It gives the information amount needed for symbolic transcription of speech using automatically derived units and thus can be compared to the *phonetic rate* (according to [76, 74], this rate is about 50 bps). The other information streams (each with its own bit-rate) are important in a real application of the coder: the information about representatives[2] and prosody information.

5.3 First set of experiments – PolyVar

This section presents speaker-dependent recognition-synthesis coding experiments conducted with one speaker of the PolyVar database. For the unit determination, the temporal decomposition (TD), vector quantization (VQ) and multigrams were used to derive initial transcriptions of training corpus. Using those transcriptions and dictionary of multigrams as coding units, a set of HMMs was trained, and further refined in one segmentation–re-estimation loop. Language model factor γ played an important role in this refinement. The synthesis was done using simple concatenation of signal segments from the training corpus, with a simple criterion based on best time and DTW match with original segments. Partial results of this work were presented at NATO Summer School'97 [99] (also as a part of Chollet's chapter [20]), at Radioelektronika'98 [101], and finally at ICASSP'98 [100].

5.3.1 Database

PolyVar [26] was recorded at IDIAP (Martigny, Switzerland) as a complement to PolyPhone in order to study intra-speaker variability of speech. Its language is French and it was recorded over the telephone. Each call consists of 38 items summarized in Table 5.1. The data were collected using analog telephone interface, with sampling frequency $F_s = 8000$ Hz. The original 8 bit A-law quantization was converted to 16 bit words. This database (DB) contains also annotations, but those were not used.

[2] The representatives can be determined uniquely using the minimization of transition distortion: in this case, the representative choice does not need to be transmitted, as the decoder can perform the same operation using only synthesis unit information.

1	sequence of 6 single digits including the hash (#) and star ($\star$) symbols
1	sheet id number (5 connected digits / 1 natural number)
1	telephone number (spontaneous)
1	16-digit credit card number
2	natural numbers (1 + sheet id)
2	currency amount
1	quantity
3	spelled words
1	time phrase (prompted, word style)
1	date (spontaneous)
1	date (prompted)
1	yes/no question
1	city name (spontaneous)
1	city name (prompted)
5	application words from a list of 105
1	name (spelling table)
1	mother tongue (spontaneous)
1	education level
1	telephone type
10	sentences (read)
1	query to telephone directory (given the name and city of subject)
1	free comment to the call

Table 5.1: Contents of one PolyVar call.

For our experiments, we choose the speaker who recorded the greatest number of telephone calls: Eric Martin with 225 calls. The entire set of calls was used in those experiments, but we found 7 of them defective:

- strange byte order (corresponding probably to unfortunate conversion between different processor architectures - Little Endian vs. Big Endian): call numbers 41691351, 42571034.
- defective markers (markers delimiting useful signals point outside signal files): call numbers 40862102, 40862237, 40871238, 40881915, 40891947.

The set of 218 correct calls was divided into training set: 174 calls ($\frac{4}{5}$) and test set: 44 calls ($\frac{1}{5}$).

5.3.2 Parameterization

In the parameterization, we had to cope with the problem of very long signal files, which are difficult to process with TD and HMM software. Therefore, signals were segmented into smaller parts using voice activity detector (VAD) decisions.

First, the LPCC parameters were extracted for both the training and the test corpus. The signal was first pre-emphasized by a $1 - 0.95z^{-1}$ filter, then divided into 20 ms frames with 10 ms overlap (centisecond frames) and weighted by a Hamming window. Then, 10 LPC coefficients were extracted by standard autocorrelation method realized by Levinson-Durbin algorithm. Each LPC vector was then converted to 10 LPC-cepstrum coefficients.

The VAD was based on two criteria mentioned in subsection 4.3.3: one relative and one absolute threshold. The relative threshold (difference of current frame energy and the most energetic frame in file) was set to -35 dB. The absolute threshold (difference of frame energy and energy of a sinusoid with maximal amplitude: $A^2/2$) was set to -84 dB. Raw decision were

smoothed using 11-tap "all-around-must-be-unvoiced" filter. The speech segments with the same type of VAD decision are called "parts". Only active parts were taken into account in further processing.

The cepstral mean subtraction (CMS) was done on call basis: in the first step, the cepstral mean was computed using all active parts from a telephone call, in the second step, this mean was subtracted from all LPCC vectors. In the following steps, we were working with those CMS'ed files.

5.3.3 Temporal decomposition, post-processing

The software package `td95` of Frédéric Bimbot was used for this step. LPCC matrix (with subtracted mean) of each active part in the training corpus was processed by TD with the following parameters:

1. by far the most important parameter is the threshold determining the number of principal components of $\mathbf{U}$ in Equation A.6. An absolute threshold was used (Equation A.8) with $D_{thr} = 1.1$.

2. thresholds for truncation of interpolation functions (IF) after in initial search were set to $\mu_1 = 0.85$ (absolute) and $\mu_2 = 0.5$ (for local minima). The same thresholds for iterative refinement of targets and IFs were set to $\mu_1^r = 0.5$ and $\mu_2^r = 0.1$.

3. smoothing coefficient (Equation A.12) set to $\gamma = 0.25$.

4. threshold to discriminate similar functions (Equation A.20) set to 0.5

5. range for inversion in the iterative refinement (section A.3) set to $k = 3$.

6. maximal error rate (expressed by RMS in Equation A.2), for which in the iterative refinement is declared non-convergent and stopped, set to 0.5.

7. relative change in RMS for stopping of iterative refinement (convergence) set to 0.1%.

8. minimal number of refinement iterations set to 5. In these 5 iterations, the deletion of events is prohibited.

9. centering of local parameter matrix before SVD allowed, re-normalization of IFs, forcing their sum to 1 (Equation A.2) allowed.

Note that points 2 to 9 are default settings of `td95` software. The only parameter which is to be adjusted in practice is the threshold D_{thr} determining the number of principal components used for construction of initial IF. This threshold was determined experimentally on several parameter files in order to obtain mean number of events per second closed to phonetic rate (approximately 15 phonemes per second). An example of temporal decomposition of PolyVar word "le chômage" was used to illustrate TD results in theoretic chapter of this thesis in Figure 4.4. The summary of parameterization, division into parts and TD for training and test sets is given in Table 5.2.

As the next step, the limits of IFs were converted to limits of segments. For finding the boundary of i-th and $(i-1)$-th event, a simple mean computation of beginning of current IF and end of previous IF was done, according to Equation 4.21. In case this boundary fell inside previously found segment (this happened mostly for overlapping of more than 2 interpolation functions), the current event was discarded.

	training	test	total
calls	174	44	218
active parts	15813	3936	19749
frames in active parts	1864936	463860	2328796
mean nb. of frames in 1 active part	117.9	117.8	117.9
total active speech [h]	5.2	1.3	6.5
TD events	271078	-	-
average rate events per sec.	15.26	-	-

Table 5.2: Summary of parameterization and temporal decomposition in PolyVar recognition–synthesis experiments. TD was done only for the training corpus.

5.3.4 Vector quantization

The vector quantization is useful for the original clustering of TD events. Two possibilities have been tested: clustering of original vectors situated in gravity centers of TD interpolation functions and clustering of spectral target vectors (columns of resulting TD matrix **A** from Equation 4.19). For both options, a codebook with $L = 64$ code-vectors was trained using LBG algorithm with successive splitting of codebook. The Euclidean distance of cepstral vectors was converted to spectral distance using Equation 4.1. Only $P = 10$ LPCC coefficients were used in this evaluation. The criterion of stopping iterations of the LBG algorithm was the relative change of two overall spectral distortions: for $< 0.5\%$, iterations were stopped. Generally, 3 or 4 iterations were sufficient to reach this relative change threshold.

As result of the quantization, we obtained 2 strings of labels in the range 0-63. Those were attributed to TD segments. We investigated the coherence of segments by listening tests (by comparing signal segments with the same associated label). The coherence was found better for original vectors situated in gravity centers, so that this option was retained for further work[3]. For practical reasons, the segment labels were converted to alphanumeric notation according to Table 5.3.

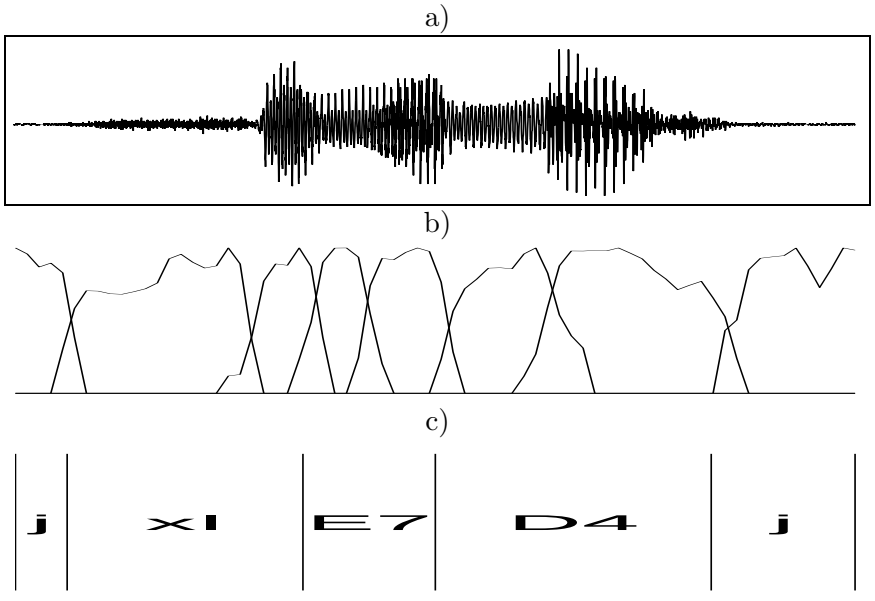
Numeric label	Alphanumeric symbol
0 . . . 25	A, B, C,. . . , Z
26 . . . 51	a, b, c,. . . , z
52 . . . 61	0, 1, 2,. . . , 9
62	@
63	$

Table 5.3: Conversion of numeric VQ labels to alphanumeric ones.

5.3.5 Multigrams, dictionary of coding units and initial transcriptions

The multigram (MG) method was used to post-process symbol strings resulting from VQ in order to lengthen basic coding units and thus to capture even more regularities in the training string. Discrete MGs were used. Their input consisted of concatenation of all VQ symbol files from the

[3]target vectors should be less influenced by LPCC estimation errors, as they could be understood as long-term mean, but they are strongly depending on neighboring segments. This is our explanation of better coherence obtained with original vectors in gravity centers.



0	450000	j
450000	2500000	xI
2500000	3650000	E7
3650000	6050000	D4
6050000	7300000	j

Figure 5.6: Example for the French word "cinema". a) signal, b) TD interpolation functions, c) MG segmentation, d) phonetic-like transcription (times are given in hundreds of ns).

training corpus. However, to prevent multigrams from crossing part boundaries, the classical method was modified by the introduction of *forced segmentation* on part boundaries. Practically, this was done by forced flushing of the buffer each time the partial likelihood $\mathcal{PL}^\star$ evaluation (Equation 4.54) was finished for the current part. The common point CP (see paragraph 4.6.5.2) was always set to the beginning of buffer. The buffer with dynamically changing length was not used.

For the pruning of MG dictionary, not the penalized probability, but *count thresholds* were employed (see subsection 4.6.2). Two MG count thresholds were applied, the first directly after the dictionary initialization, the second during the segmentation-re-estimation iterations. With the following step of HMM training in mind, both were set to 20, providing thus a sufficient number of examples for each HMM in the training data.

The MG dictionary was trained for maximal MG length $n = 5$, with 10 iterations of segmentation-re-estimation. The result of this algorithm are: a *dictionary* of typical sequences and a *segmentation* of the input string. In the resulting dictionary, the numbers of 4- and 5-grams were zero; it consisted of 64 1-grams, 1514 2-grams and 88 3-grams. The total amount of multigrams was thus 1666. The segmentation, together with TD segment boundary information, was converted to phonetic-like transcriptions. An example of TD and MG processing, and of conversion of their results to transcription for PolyVar word "cinema" is given in Figure 5.6.

5.3.6 Initial HMM training

The HTK toolkit version 1.4 from Entropic Research Labs [105] was used for all experiments related with HMMs. Three important choices had to be done:

- The *parameterization* for HMMs was created as extension of the original LPCC vectors. First parameter *stream* consists of LPCC coefficients with subtracted mean. Second stream includes approximations of first derivation of those coefficients (Δ or "velocity" parameters). The third stream includes energy and Δ-energy computed frame-by-frame using Equation 4.10. The weights of streams (Equation 4.79) were set equal: $\gamma_1 = \gamma_2 = \gamma_3 = 1.0$.

- The choice of HMM *state number*. This number was determined accordingly to the length of original multigram. For an i-gram, an HMM with $2i + 1$ emitting states was created accounting for the fact, that each stable part of original TD event and each event-event transition should be covered with one state.

- The HMM *structure* was chosen as classical left-right, with no state skip, so that only transition probabilities $a_{j,j}$ and $a_{j,j+1}$ (except for first and last non-emitting states) are non-zero.

One HMM corresponds to one MG, so that there were 1666 models to train. The initial set of HMMs was trained using HTK tools `HVite` (initialization), `HRest` (context free re-estimation) and `HERest` (context dependent re-estimation). We performed five iterations of the last mentioned step (`HERest`). The resulting set of trained HMMs is called *first generation HMM dictionary*.

5.3.7 Refinement of HMM set

The coherence of HMM set with the data was further improved by HMM segmentation (equivalent to recognition in standard tasks) and re-estimation of model parameters.

The *segmentation* was performed using Viterbi algorithm implemented in HTK tool `HVite`. However, as we wanted to experiment with the use of unigram language model (LM), weighting the likelihoods of models, we modified `HVite` to be able to make use of this LM (standardly, `HVite` can work either in LM-free manner, or use a bigram LM). On contrary to Equation 4.91 in subsection 4.7.6, the language model probability is not directly multiplied with the product of emission and transition probabilities of HMMs, but it is weighted by a *language model scale factor* γ. With higher values of γ, the influence of a-priori probabilities of models is greater, so that more probable models are likely to be used, even if their associated emission probabilities are low. The likelihood product to maximize over the set of all possible segment chains is therefore:

$$\max \mathcal{L}(\mathbf{O}|w_1^N)\mathcal{L}(w_1^N)^\gamma. \tag{5.28}$$

Three LM scale factors[4] $\gamma = 0.0$ (grammar-free), 5.0 and 10.0 were tested. A-priori probabilities of *1st generation LM* were given by probabilities of corresponding multigrams in the MG dictionary. Using the Viterbi alignment of training set data with models with those 3 different LM factors, we obtained three different sets of transcriptions. If we call the transcriptions obtained using TD and MG "initial ones", we can call these sets "1st generation transcriptions".

Using 1st generation transcriptions, we were estimating 2nd generation models. However, before running the estimations, it was necessary to prune the initial MG dictionary by deleting models whose labels appear only rarely in 1st generation transcriptions. This pruning was again based on occurrence threshold, which was set to 20, similarly as for multigrams in subsection 5.3.5. Table 5.4 gives counts of 1st and 2nd generation HMMs. Using the counts in 1st generation transcriptions, 2nd generation language models were estimated using counts of labels divided by the total number of labels in the training string. Having finished those preparatory

[4]HTK documentation [105] uses the term "grammar scale factor".

Numbers of HMMs	1	2	3	total
First generation	64	1514	88	1666
Second generation $\gamma = 0.0$	63	1364	87	1514
Second generation $\gamma = 5.0$	45	1069	87	1201
Second generation $\gamma = 10.0$	41	769	84	894

Table 5.4: Numbers of HMMs in the initial and refined sets.

	train. set			test set		
γ	R_e [bps]	R_u [bps]	R_s [seq/sec]	R_e [bps]	R_u [bps]	R_s [seq/sec]
0.0	113	117	11.07	116	120	11.40
5.0	68	74	7.19	69	74	7.29
10.0	51	58	5.95	50	58	5.95

Table 5.5: Resulting rates for sequence indices coding with different language model (LM) factors γ, assuming uniform coding (R_u) or entropy coding with a-priori probabilities from corresponding LM (R_e). R_s is the average number of sequences per second.

works, 2nd generation HMMs could be trained by 5 iterations of context-dependent re-estimation (`HERest`). First generation models were used for initialization of `HERest`.

Using 2nd generation models and LMs, we created 2nd generation transcriptions, which were supposed as final product of segmental coding. With the same sets of models and language models, the test data set was processed too. We then evaluated the coding unit rates on the training and test corpora; they are summarized in Table 5.5.

5.3.8 Representatives and synthesis

To reconstruct the speech in the decoder, a simple synthesis based on concatenation of signal segments was used. Prior to this step, synthesis units and their representatives (see subsection 5.2.1) had to be chosen: in those experiments, *synthesis units* (SU) were equivalent to the coding ones. There were therefore 1514 SUs for $\gamma = 0.0$, 1201 SUs for $\gamma = 5.0$ and 894 SUs for $\gamma = 10.0$. For each SU, 8 representatives with the corresponding label were extracted from the training corpus. There was no special choice of representatives, simply the 8 first occurrences in the corpus were taken. Example of those 8 representatives for unit `JDV` can be seen in Figure 5.7.

In the coding, the following algorithm was used to assign a representative to an input segment:

1. as the synthesis method did not allow for any time modifications, the first choice was based on time match. Among the 8 representatives, 4 time candidates minimizing the length difference between input segment and representative (Equation 5.5) were chosen.

2. among these 4 candidates, the winner was chosen by evaluating the DTW distance between candidates and input sequence in the LPCC parameter space.

The choice of representative had to be encoded by 3 bits per segment, so that a term $3R_s$ must be added to the bit rate (R_s is the mean rate of sequences per second).

As we have already mentioned, the synthesis was done as concatenation of chosen representatives. This method allows for no timing nor pitch modification. Only a mean energy correction was done: the mean energy of representative $\bar{E}_r$ and of input segment $\bar{E}_s$ were computed. The correction $\bar{E}_s - \bar{E}_r$ was quantized using 32-level scalar quantizer. Energy correction transmission

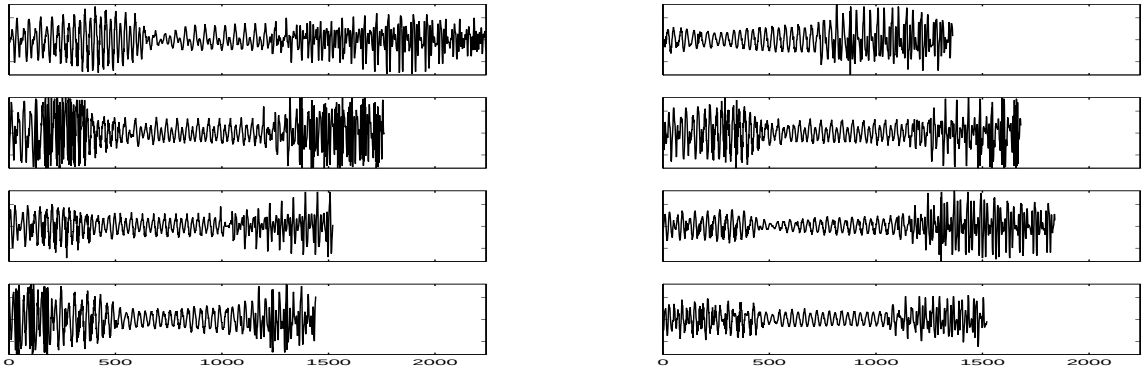


Figure 5.7: Eight representatives of unit JDV (phonetically corresponds approximately to "imo"). The longest representative has duration 0.28 sec, the shortest one 0.17 sec.

γ	train. set	test set
0.0	**205.6**	**211.2**
5.0	131.5	132.3
10.0	105.6	105.6

Table 5.6: Resulting total rates [bps] of segmental coding including representative identity (3 bits per sequence) and energy correction (5 bits per sequence) transmission. Uniform coding of sequence indices (R_u) is considered.

adds the term $5R_s$ to the total bit rate. The rates for training and test strings including representative choice and energy correction are summarized in Table 5.6. An example of original and synthetic utterance "information consommateur" can be seen in Figure 5.8. Informal listening evaluations were done with several original signals and synthetic versions for all three used LM factors $\gamma = 0.0$, 0.5, 10.0.

5.3.9 Results

The resulting bit rates are shown in Table 5.6. In listening tests we found, that for all three LM factors, the output speech sounded quite naturally. However, for $\gamma = 5.0$ and 10.0, the speech is not intelligible, even if it sounds naturally (a paradox in comparison with standard coding schemes). Therefore, the coding with $\gamma = 0.0$ should be used, so that the rate obtained is 205.6 bps for the training set and 211.2 bps for the test set. For this LM factor, the intelligibility was good for digits, names, command words and short phrases appearing often in the PolyVar corpus. It was worse for longer utterances. The examples of original and coded speech for an entire PolyVar call, with all three mentioned LM scale factors, can be downloaded in WAV or AU audio formats from the Web:
`http://www.fee.vutbr.cz/~cernocky/Icassp98.html`

5.3.10 Discussion

Those experiments with recognition-synthesis coding using automatically automatically derived units with their stochastic modeling have proven feasibility and efficiency of this approach, as we obtained mostly intelligible and quite naturally sounding speech at very low bit rates. It is clear, that many simplifications had to be introduced in these experiments: primarily the speaker-dependent mode. Our scheme did not include any speaker adaptation or normalization

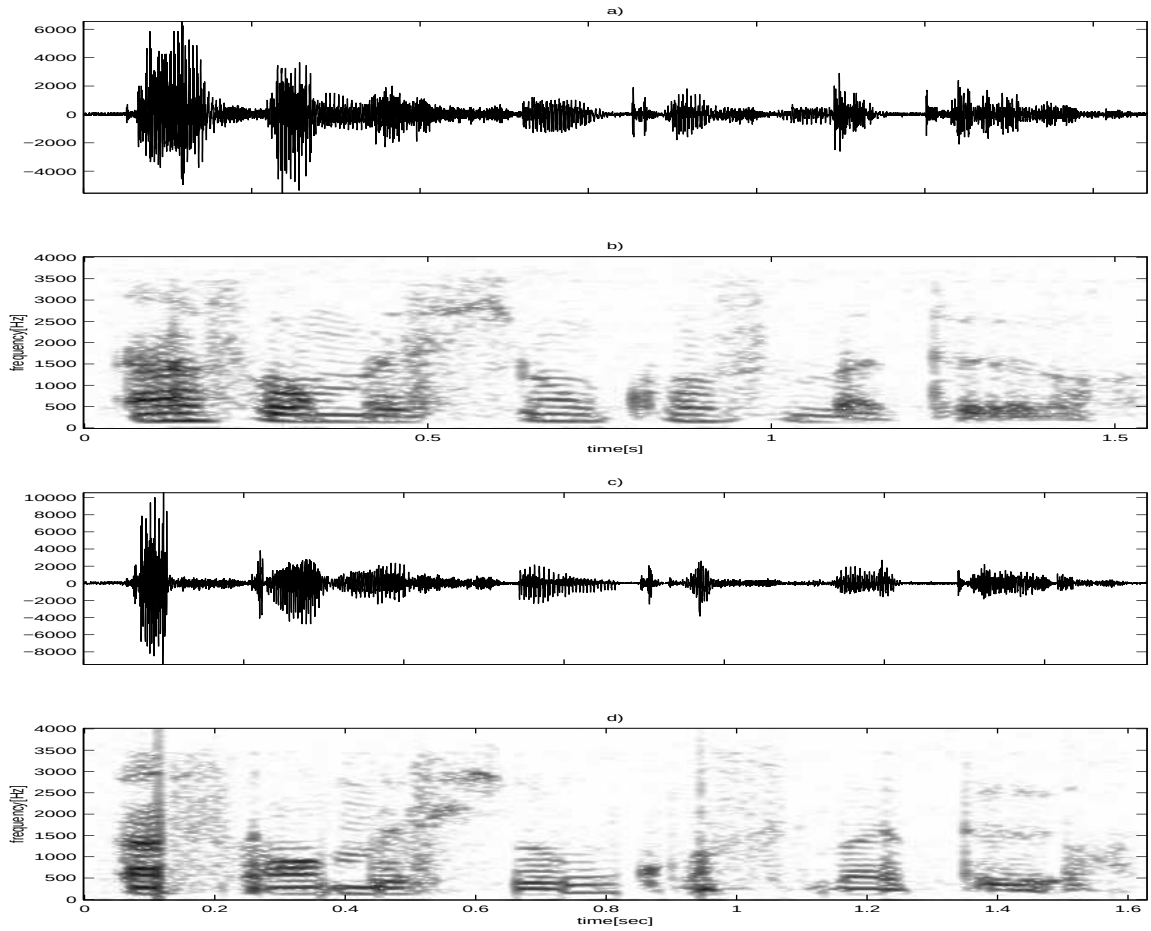


Figure 5.8: French words "information consommateur": a) original signal, b) original spectrogram, c) signal of the synthetic version with $\gamma = 0.0$, d) synthetic version – spectrogram.

module: with the set of units trained on Eric Martin's data from PolyVar, we did preliminary experiments with the TIMIT database [39], but those were not successful. But even if we admit speaker-dependent nature of experiments, the results were not as good as expected, namely for longer phrases in the corpus. We found three possible reasons and suggested the following improvements:

1. For tests, where a principle of method should be verified, the conditions should be as positive as possible. However, the quality of used database is far from "Hi-Fi": it is noisy, recorded over the telephone with all known distortions (microphone, channel, etc.). Background noises appears often (strange voices, bell, door flapping, ...). But first of all, the speaking style of the subject is not very clear, and native French speakers had often problems to understand, even when listening to *original signals*. Therefore, in future works, we suggested to use a *database of better quality*.

2. The use of *multigrams before HMMs* makes basic coding units longer, than if quantized TD events were used directly as the first transcriptions. As we will see later, this alleviates a number of transition problems, as there are fewer transitions in the synthesized speech. On the other hand, the number of free parameters in multigram-based HMMs is considerable, makes the system very complex and lowers the reliability of parameter estimations (few examples for a model in the training corpus). Therefore, for the following experiments, we suggested to work with *lower number of simpler models* based on quantized TD events.

3. The *concatenation synthesis* was employed only for the sake of simplicity. Thanks to quite monotonous speaking style of the subject, pitch errors and discontinuities were less audible than in experiments with Boston University Corpus (see the following section). However, the parametric synthesis with pitch and duration modification is a necessary step to able to fully use the coding efficiency of segments.

5.4 Recognition – synthesis coding with Boston University Radio Speech Corpus

In those experiments, the goal was to address the problems mentioned in above discussion of previous section. The main issue was the choice of database. From few commercially available databases with sufficient data from one speaker, Boston University Radio Speech Corpus (BURSC) distributed by LDC[5] was chosen. The experiments were carried out for one male and one female speaker. The data were processed by temporal decomposition and vector quantization, without the use of multigrams prior to HMMs. Greater attention was payed to the iterative refinement of HMM set with recording of likelihood change. The language model was not used in these experiments. The speech was synthesized using LPC synthesis with the original prosody, and representatives were aligned with original segments using the DTW path.

5.4.1 Database

Boston University Radio Speech Corpus[6] was collected in 1995 by the group of Mari Ostendorf at BU primarily to support research in text-to-speech synthesis (prosody patterns). It contains data from 7 professional FM-radio speakers. The whole DB contains annotations, and its parts are completed with phonetic alignments, part-of-speech tags and prosodic labelling. None of these was used in the coding experiments, but phonetic alignments were later used in the study of correspondence of ALISP units and phonemes (Chapter 7). The data are divided into two portions: the main *radio speech* data were recorded in the WBUR radio station during broadcasts. They are completed with *laboratory news* portion recorded at BU laboratory. Detailed description of this DB is given in its documentation [70].

In our experiments, the data of one male speaker (M2B) and of one female speaker (F2B) were used. The radio news portions were used as training corpora and laboratory news one as test set. Clean data and noisy data are distinguished in the DB. For training as well as for tests, we took into account only the clean ones. The data are organized within directories related to radio *stories*, and are divided into paragraphs. For F2B and M2B clean data, the longest paragraph has 75 seconds. For experimental tools, such as temporal decomposition or Viterbi decoder in HTK, paragraphs were too long, so that they had to be split into shorter *parts* (see following subsection). Their amounts for both speakers are summarized in the first part of Table 5.7.

5.4.2 Parameterization

As this DB is sampled at $F_s = 16$ kHz, the number of cepstral parameters was increased in comparison to PolyVar. The LPC analysis was done with order of $P = 16$ and 16 LPCC coefficients were derived for each frame. The frame length and overlapping were 20 and 10 ms,

[5]LDC stands for Linguistic Data Consortium, an organization collecting and distributing speech and text databases, lexicons, and other resources. LDC is located at the University of Pennsylvania, `http://www.ldc.upenn.edu/`.

[6]also called Boston University Radio News Corpus.

similarly as in the previous experiments. Cepstral mean subtraction was performed on story basis and CMS'ed parameters were used in the following work (except for the parametric synthesis).

The *pitch* was determined as well, using FFT-cepstral method described in subsection 4.3.2. For the voicing detection, the relative energy threshold (Equation 4.12) was set to $RE_{voi} = 0.015$. The maximal zero passages count in 20 ms frame for voiced frame was set to 80. The threshold for prediction gain (Equation 4.11) was set to 0.08. The frame length for FFT-cepstra computation was 700 samples and this frame was centered on the "normal" analysis frame. The lags were searched in the interval 42 to 284 samples, which corresponds to F_0 in the range $56 \leq F_0 \leq 381$ Hz. The *energy* was simply evaluated frame-by-frame, using Equation 4.10.

The paragraphs had to be segmented into smaller parts, in order to work with reasonable lengths of parameter matrices in TD and HMMs. First, this segmentation was done using voice activity detector, similarly as in PolyVar experiments. In the stories spoken by professional speakers, there are however too few silences, so that almost all the data were classified as active. Instead of VAD, we finally used the division into parts of predetermined approximate length using energy contours. The part length was set to 800 frames with allowed deviation of ±400 frames. The minimal length of part was imposed to be 200 frames. Figure 4.3 on page 42 gives an illustration of this division. The total numbers of parts and frames are summarized in Table 5.7.

5.4.3 Temporal decomposition

The temporal decomposition was once again done using Bimbot's `td95`. Its parameters were the same as in previous experiments, except for the threshold D_{thr} controlling the number of singular vectors participating on the creation of one interpolation function (Equation A.8). This threshold was experimentally set to have slightly higher number of events per second than in PolyVar experiments: with $D_{thr} = 0.9$, this number was approximately 17-18 events/sec. The number of events per second was increased, as in the previous experiments, we sometimes observed very distinct sounds being "covered" by one TD event. The entire training set was processed by TD with the results summarized in Table 5.7.

The segment boundaries were determined more precisely than in previous experiments as intersections of adjacent interpolation functions (see subsection 4.4.3 for details).

5.4.4 Vector quantization

The vector quantization training was done uniquely with original vectors situated in gravity centers of TD interpolation functions. The codebook with length $L = 64$ was created using LBG algorithm with successive splitting of code-vectors, independently for each of speakers M2B and F2B.

On contrary to previous experiments, for the vector quantization, not only one (original) vector per segment, by the whole segment was taken into account using cumulated distance method, as it is described in subsection 4.5.3. This option should alleviate classification errors encountered in the previous work, where a noisy original vector in gravity center could cause misclassification of the entire segment. However, the drawback of cumulated distance approach is the taking into account also of vectors near segment boundaries, where the influence of neighboring segments is strong.

No multigram lengthening of segments was used, so that from TD-derived segment boundaries and VQ labels, initial transcriptions of the training set could be directly created. These were used in the next step for HMM training.

F2B	training	test	total
stories	32	7	39
clean paragraphs	124	41	165
time	56 min	22 min	78 min
parts	394	150	544
frames	336743	129919	466662
TD events	61489	-	-
average frames/event	5.47	-	-
average events/sec	18.25	-	-

M2B	training	test	total
stories	33	4	37
clean paragraphs	214	24	238
time	71 min	12 min	83 min
parts	499	77	576
frames	428929	69566	498495
TD events	76256	-	-
average frames/event	5.62	-	-
average events/sec	17.77	-	-

Table 5.7: Summary of data amounts and temporal decomposition results for speakers F2B and M2B of BU corpus. TD was done only for the training corpus.

5.4.5 HMM training and refinement

HMMs were related to original VQ symbols, so that their number was 64. The number of emitting states per model was fixed to 3, and the models were initialized as left-right, without state skipping. The feature vectors consisted, similarly as for PolyVar, of 3 streams: LPCC parameters with cepstral mean subtracted in the first, Δ-parameters in the second and energy and Δ-energy in the third. HMMs were trained using HTK tools `HVite` (initialization), `HRest` (context-free re-estimation) and `HERest` (context-dependent re-estimation). We performed five iterations of the last mentioned step (`HERest`). The resulting set of trained HMMs is called *first generation HMM dictionary*.

With this 1st generation HMM set, the Viterbi alignment of data with models was performed (`HVite`). Language model was not used, but fixed model-model transition log probability was experimentally set to 9.0 in order to have approximately the same rate of segments per second as we obtained by TD. After this step, 5 iterations of HMM refinement were run, each consisting of HMM re-estimation (5×`HERest`) and of corpus segmentation. The likelihoods of final context-dependent re-estimation (`HERest`) in each iteration were recorded and are shown in Table 5.8, together with Viterbi alignment likelihood of one part of the test set (the same for which synthesis tests were done). During those iteration, no language models was used and there was no change in HMM lists, as all segments appeared with sufficient number of occurrences in all segmentations.

In each iteration, not only the training data set, but also the test one was aligned with models. We could thus evaluate *sequence rates* on both corpora, as shown in Table 5.9.

5.4.6 Representatives and synthesis

Also in these experiments, the synthesis units (SU) were equivalent to coding ones. Two synthesis methods were tested: the concatenation of representative signals, similarly as in the previous case, and parametric LPC synthesis using the original prosody.

HMM generation	**F2B**		**M2B**	
	$\log \mathcal{L}_{HERest}$	$\log \mathcal{L}^1_{HVite}$	$\log \mathcal{L}_{HERest}$	$\log \mathcal{L}^1_{HVite}$
1.	35.22	33.76	34.22	37.27
2.	37.56	33.95	36.44	38.09
3.	38.01	34.26	36.81	38.25
4.	38.24	34.39	36.97	38.33
5.	38.37	34.47	37.05	38.40
6.	38.44	34.48	37.13	38.54

Table 5.8: Likelihoods change during the training (`HERest`) and recognition (`HVite`). $\log \mathcal{L}_{HERest}$ is the mean log-likelihood per frame calculated on the entire training corpus. $\log \mathcal{L}^1_{HVite}$ are mean log-likelihoods related to one test file (`f2bjrop1.0` for F2B and `m2bjrlp1.0` for M2B).

HMM generation	**F2B**				**M2B**			
	train. set		test set		train. set		test set	
	R_u	R_s	R_u	R_s	R_u	R_s	R_u	R_s
1.	107.44	17.90	109.45	18.24	104.53	17.42	104.86	17.47
2.	112.59	18.76	114.23	19.03	110.03	18.33	112.09	18.68
3.	117.28	19.54	118.25	19.70	114.68	19.11	117.55	19.59
4.	120.75	20.12	121.55	20.25	118.79	19.79	122.15	20.36
5.	123.65	20.60	124.46	20.74	122.89	20.48	126.51	21.08
6.	126.18	21.03	126.86	21.14	126.51	21.08	130.35	21.72
MG	103.27	10.88	110.26	11.61	108.95	10.97	119.81	12.07

Table 5.9: Iterations of HMMs: sequence coding rates evaluated on training and test sets during HMM refinement. Only rates assuming uniform coding of sequence indices R_u [bps] and mean rates of sequences per second R_s are shown. The table is completed by rates obtained with multigram-lengthened units (subsection 5.4.7).

For the *concatenation synthesis*, the representatives were chosen in the training corpus without any special criteria. 8 first occurrences of each SU were taken. The choice of representative for incoming segment was done similarly as for PolyVar: first, 4 candidates with the best time match were chosen, then, the winner minimizing the DTW distance from the input was selected. The choice of representative must be encoded by 3 bits per sequence, so that the term $3R_s$ must be added to the rate. We performed listening tests with one example file of F2b (`f2bjrop1.0`) and one file for M2B (`m2bjrlp1.0`).

The main attention was however payed to the *parametric synthesis*. The choice of representatives in the training corpus was done differently than before: 8 *longest* representatives from the training corpus were chosen, so that they are mostly down-sampled when being converted to shorter segments. For parameter matrices of representatives, original LPCC files (without cepstral mean subtraction) had to be taken into account. The synthesized versions of files `f2bjrop1.0` and `m2bjrlp1.0` were created for all generations of HMMs, so that we could evaluate the synthetic speech quality for different degrees of model set refinement. The choice of representatives for input segments was done using one of two following criteria:

- minimum DTW distance between a representative and an input segment.
- minimum distortion on boundaries of representatives, computed as Euclidean distance of

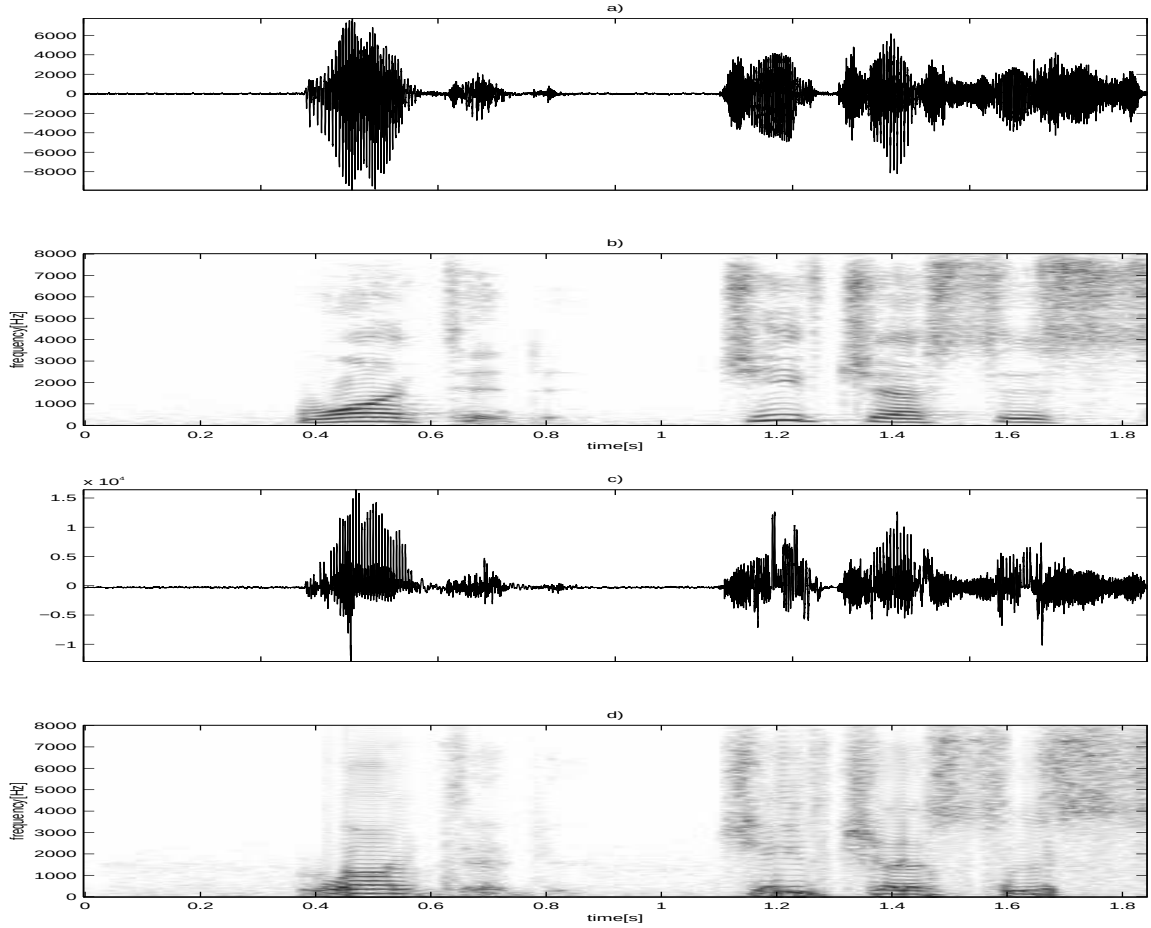


Figure 5.9: Example of synthesis of BU file `m2bjrlp1.0`, containing "wanted, chief justice": a) original signal, b) original spectrogram, c) signal of the synthetic version with 4th generation HMMs and DTW choice of optimal representative, d) synthetic version – spectrogram.

last LPCC vector of the "left" representative and first vector of the right one. Optimal representatives were chosen from left to right (e.g. the first minimizing the DTW distance, the second best concatenating with the first, the third best concatenating with the second, etc.), so that the minimization of global distortion distance given in Equation 5.11 is not ensured.

Important part of the synthesis is the prosody used. The original prosody was used in those experiments, we copied original pitch and energy contours, lengths of original segments and DTW alignment paths of original segments and representatives. In the synthesizer, the representative LPCC parameter matrix was aligned with the original segment following Equation 5.23. The speech was then produced using a simple LP-synthesizer, described in paragraph 5.2.4.1: excitation was either white noise or periodic sequence of Dirac pulses, synthesis filter parameters were derived from aligned representative's LPCC coefficients, and energy rectification with deemphasis were performed in the end. An example of original and synthesized signal for 4th generation HMMs, with representative choice using the 1st option (optimal DTW distance) is shown in Figure 5.9.

n	1	2	3	4	5	6	total
F2B	64	311	260	68	16	3	722
M2B	64	485	349	71	3	0	972

Table 5.10: Numbers of multigrams in dictionaries trained on transcriptions obtained by 6-th generation HMMs.

5.4.7 Multigrams used for lengthening of synthesis units

In order to decrease the number of segments per second and in this way to alleviate some transition problems, experiments were conducted with MG "lengthening" of sequences. Note however, that on contrary to previous works with PolyVar, MGs were not used *prior* to HMMs, but as a post-processing, so that the number of parameters in HMM did not increase.

The transcription of training string obtained with 6th generation HMMs were converted to symbol strings and used as input to MG method. Discrete multigram dictionaries training, with maximal length $n = 6$ and imposed segmentation on part borders, was run with 10 iterations of segmentation and probabilities re-estimation loop. The obtained numbers of 1- to 6-grams are summarized in Table 5.10. With those dictionaries, training and test sets were segmented and the multigram segmentations were evaluated in terms of mean bit rates. The results are joined to Table 5.9, so that they can be easily compared with those for coding without multigrams.

The representatives were chosen only for the parametric synthesis: 8 longest segments in the training set were chosen for each SU. This choice is more complex and representatives take more disk space, as their numbers are $8 \times 722 = 5776$ for F2B and $8 \times 972 = 7776$ for F2B (compare with $8 \times 64 = 512$ in case multigrams are not used). The choice of optimal representative for an input segment was done using one of two possible criteria, described in previous subsection: minimum DTW distance or minimum distortion on transitions. Again, the original prosody was used, with copying of pitch, energy, length and DTW path directly to the decoder.

5.4.8 Results

Informal listening tests were conducted for two above mentioned files (one per speaker), comparing original and coded version for different generations of HMMs, two types of the synthesis (concatenation, parametric), simple or multigram-lengthened synthesis units and different methods of representatives choice. The rates obtained with different generations of HMMs and finally with MG-like units are summarized in Table 5.9. It should however be noted, that those rates do not include the choice of representative (one out of 8, rate increase $3R_s$), and that the prosody representation is not taken into account.

The listening tests with *concatenation synthesis* have shown very poor quality of output speech, which was mostly incomprehensible. On contrary, the quality of *LPC synthesized* speech was found not excellent, but acceptable. Slight improvement of resulting speech quality was observed when "upgrading" the HMM set from one generation to another. On the other hand, as the rate of sequences per second increases for newer generations (see R_s figures in Table 5.9), more transition discontinuities appear and are audible. When comparing two methods of representative assignment to input sequences, no significant differences were found, the second method (minimizing transition distortions) produces slightly smoother speech.

When comparing coding and synthesis using *multigram* units with the previous HMM-like SUs, one observes, that the fluency of output speech increases. This is certainly caused by lower number of transitions (compare sequence rates R_s for original and MG units in Table 5.9). However, it seems, that this amelioration is counter-balanced by the decrease of acoustic match between units and original segments. From the subjective point of view, the best quality is

obtained by synthesis with MG units, with representative choice of type 2 (minimization of transition distortions).

5.4.9 Discussion

The goal of those experiments was to test the coding with automatically derived units on a high quality database, with simplifications in comparison to previous work with PolyVar. One of main issues was to test parametric synthesis with use of representatives' LPCC matrices, where the pitch and energy can be modified. Again, the usefulness and efficiency of this approach was verified, as we obtained mostly intelligible speech at sequence rates below 120 bps. It must be however mentioned, that these are the last experiments performed, so that the results are not final and are subject to further work. There are numerous problems we are conscious about:

1. The *synthesis units* used were equivalent to coding ones. As we have outlined in subsection 5.2.1, this is not necessarily the best choice due to a great risk of transition problems when concatenating parametric representations of those units. For single HMM units, a simple method comes into mind, where SUs would correspond to *transitions* of original coding units, in similar way, as *diphones* are constructed in text-to-speech synthesis. The theoretical number of those SUs is $64^2 = 4096$, but this ensemble can be pruned and for rare combinations, a concatenation of two shorter units is possible.

2. The *assignment of representatives to input sequences* was done using exclusively by one of two criteria: best DTW match or minimum distortion. In our future work, we will recombine them to obtain a joint optimization of match of representatives with the input, and of transition distortions. Work is also to be done on the *smoothing* of transitions.

3. The *LPC analysis and synthesis*, we used, suffer from many problems. The pitch detector produces often multiples and fractions of real pitch, which has negative effect on the synthetic speech quality, even if only copy LPC synthesis is performed. As for the synthesizer, the excitation creation for voiced sounds (sequence of Dirac pulses) is the most primitive one, implemented for simplicity reasons. Its negative effects are constant amplitudes of harmonics up to $F_s/2$ (see Figure 5.9d) in comparison with the FT of inverse LP filtered signal, and a DC bias introduced to the signal (Figure 5.9c).

4. The above mentioned problems should be overcome by *higher quality synthesis*. Experiments with *Harmonic Noise Model* (HNM) [88], or *Pitch-Synchronized Overlap And Add* (PSOLA) synthesis are envisaged.

5. The prosody coding was not yet investigated. Pitch and energy patterns should be found in segment-dependent way.

When attempting to evaluate the possible resulting bit-rate, including the coding of representatives and of prosody, we took the highest figures of Table 5.9 (including multigrams): the rate for unit encoding of 130 bps and the rate of sequences 12 per second. The choice of representative must be encoded by 3 bits per unit. In case we used similar technique for pitch and energy encoding as Mouy et al. [62] in their 800 bps vocoder, we would need additional 19 bits per unit. The total rate would therefore be $130+22\times12=$**394** bps. This figure holds for the speaker-dependent case; if the transmission of voice characteristics was requested, another information (means of spectral parameters, transformation matrix or other) would be needed.

5.5 VLBR Coding: Conclusions

The very low bit-rate coding was the first application, on which automatically derived speech units were tested. The ALISP approach allowed us to overcome the problems of existing phonetic vocoders (need of an annotated training database). The derivation of coding units and of their derivatives (synthesis units and representatives) is fully automatic.

The experimental results obtained so far are promising: in both sets of experiments, intelligible speech was obtained at mean rate of coding units of about 120 bps. We can state, that the rate of this speech representation approaches the phonetic rate.

Certainly, for real applications of this scheme, many problems have to be resolved any many issues are still open: in subsections 5.3.10 and 5.4.9 we have mentioned the determination of synthesis units, the smoothing of their transitions in the decoder, the synthesis itself, and mainly the speaker adaptation. We believe, that solutions of these more or less technical issues exist; many of them were not investigated because of the restricted time of this thesis, and because of other interesting emerging applications of studied algorithms.

As we have mentioned in section 2.4, the main benefit of coding experiments was the possibility to *evaluate* the set of obtained ALISP units. In our opinion, this goal was fulfilled, as the experimental work allowed for the *assessment* of different ALISP techniques. The success of their use in VLBR coding predicts their usability in other speech domains.

Chapter 6

Application No. 2: Text-Independent Speaker Verification

The second tested application of automatically derived speech units was the *speaker verification independent on the text*. Here, a division into segments and their assignment to broad classes were used as a pre-processing to a set of scoring systems. In the experimental part, only a subset of ALISP tools was used: the *temporal decomposition (TD)* and *vector quantization (VQ)*. The scoring was implemented by *Multi-Layer Perceptrons (MLP)*. This approach was verified on the NIST-NSA data during the NIST'98 evaluation campaign. Performances of the segmental system were compared with a global one, with the same MLP scoring, and with a base-line system based on Gaussian-Mixture Models (GMM).

6.1 Introduction

In recent years, speaker recognition technology has made quite a lot of progress, but open research problems still remain. The generic term of speaker recognition comprises all of the many different tasks of distinguishing people on the basis of their voices. There are speaker identification tasks – who among many candidate speakers pronounced the available test speech sequence; or speaker verification tasks – whether a specific candidate speaker said the available test speech sequence. In this paper we focus on speaker verification, which is actually a decision problem between two classes: the *true* speaker (also denominated as *client* or *target* speaker) and the *other* speakers (usually noted as *impostors*). As far as the speech mode is concerned, speaker recognition systems can be *text-dependent* or *text-independent*. In text-dependent experiments, the text transcription of the speech sequence used to distinguish the speaker is known. In text-independent tasks, the foreknowledge of what the speaker said is not available.

As far as the accuracy is concerned, text-dependent systems perform generally better than text-independent systems. There are two reasons for this:

- the knowledge of what has been said can be exploited to align the speech signal into more discriminating classes (words or sub-word speech units).
- an optimized recombination of these class decisions can be done. Several studies on text-dependent systems [35, 68, 71, 46] have demonstrated that some phones show more speaker discriminative power than others[1], suggesting that a weighting of individual class decisions should be performed when computing the global decision.

[1]This fact has been previously validated by phonetic studies [67] that have also shown that the information about the identity of a speaker is not uniformly distributed among speech segments (phones).

6.2 Global versus segmental approach

We are interested here in building robust text-independent systems. Since the content of the speech signal is not accessible, text-independent speaker verification is usually based on modeling the *global* probability distribution function (PDF) of speakers in the acoustic vector space. We believe that such global approaches are reaching their limits because speaker modeling is too coarse in this case.

Therefore, we have proposed to investigate a *segmental* approach in which the speech signal is pre-classified into more specific speech units. This approach should recover text-dependent advantages since the speech signal is aligned into classes. The implementation is however different, since we have no clue about what has been said. As for text-dependent systems, we can underline two potential advantages: first, if the speech units are relevant, then speaker modeling is more precise and the system should have better performances than the global approach. Second, if speech units present different discriminative power, then better recombination of the decisions per class can be done. The disadvantage of this method is, that it requires an accurate recognition of speech segments. Two alternative procedures can be employed:

- Large Vocabulary Continuous Speech Recognition (LVCSR) provides the hypothesized contents of the speech signal on which classical text-dependent techniques can be applied. LVCSR uses previously trained phone models and a language model, generally a bigram or trigram stochastic grammar.
- ALISP (Automatic Language Independent Speech Processing) tools provide a general framework for creating sets of acoustically coherent units with little or no supervision.

LVCSR systems, although they are promising for segmental approaches, suffer from the same problems as all classical schemes based on sub-word units (section 2.2). For the training, they require large annotated databases, which are either costly or not available. Furthermore, LVCSR are often dependent on speech signal characteristics (language, speech quality, bandwidth, ...) making them difficult to adapt to new tasks. Finally, sub-word units derived for *speech* recognition are not necessarily the best choice for *speaker* recognition.

These reasons, and the focus of this thesis, led us to investigate the later approach. As we have seen in several places of previous chapters, ALISP tools offer an alternative to classical sub-word approaches, when no annotated training data are available. It has however to be proved, that such automatically derived classes and segmentation of input speech can be useful in the verification: the necessity of determination of the *discriminative power* of classes and of development of algorithms able to *merge* efficiently class-dependent scores, is a common issue to both LVCSR and ALISP approaches.

6.3 System description

A classical (non-segmental) text-independent speaker verification system, illustrated in Figure 6.1, can be represented in three blocks:

1. *Feature Analysis*: similarly to what is done for speech recognition, the speech signal is cut into frames undergoing feature extraction. The signal can thus be represented by a sequence of feature vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ of length N. Classical feature extraction algorithms are LPC-cepstral or MFCC (mel-frequency cepstral) analysis.
2. *Scoring*: The sequence of feature vectors is fed into a classifier, which outputs a likelihood score of the client model and of the world model[2]: S_c and S_w.

[2]In the verification, "world" stands for non-client speakers used to train the competing model to "client".

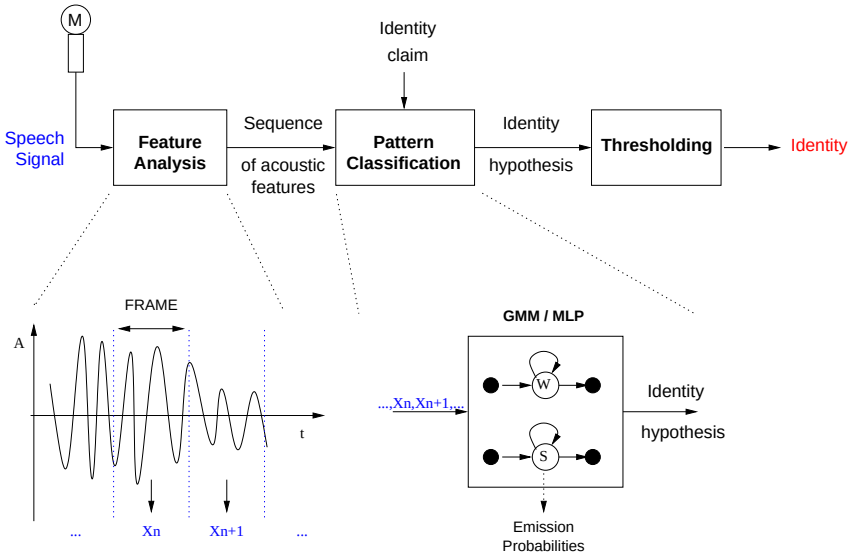


Figure 6.1: Speaker verification system.

3. *Decision taking*: the verification (reject/accept) of the speaker is performed by comparing the ratio (difference of logs) of client and world scores to a threshold:

$$\log(S_c) - \log(S_w) > T \quad \Rightarrow accept \tag{6.1}$$
$$\log(S_c) - \log(S_w) \leq T \quad \Rightarrow reject \tag{6.2}$$

The threshold should be set to minimize the cost function $C_{det}(T)$ of false rejection and false accepting:

$$C_{det} = C_{fr}\mathcal{P}(reject|client)\mathcal{P}(client) + C_{fa}\mathcal{P}(accept|\overline{client})\mathcal{P}(\overline{client}) \tag{6.3}$$

where C_{fr} is the cost of false rejection, C_{fa} is the cost of false acceptance, $\mathcal{P}(reject|client)$ is the probability to reject a client, $\mathcal{P}(accept|\overline{client})$ is the probability to accept an impostor, $\mathcal{P}(client)$ is the a-priori probability of client and $\mathcal{P}(\overline{client})$ is the a-priori probability of impostor. $\mathcal{P}(reject|client)$ and $\mathcal{P}(accept|\overline{client})$ are both functions of the threshold value T.

6.3.1 Global systems

Figure 6.1 shows a classical way to do pattern classification in text-independent systems, referred here as the *global* method. For both the client and the world a unique probability distribution function (PDF) is estimated using all available training feature vectors.

6.3.1.1 Gaussian Mixture Model (GMM)

One way to model the probability distribution functions is to use mixtures of Gaussians. The multivariate distribution is given by:

$$\mathcal{L}(\mathbf{x}_n|M) = \sum_{j=1}^{J} w_j \mathcal{N}(\mathbf{x}_n; \mu_j, \Sigma_j) \tag{6.4}$$

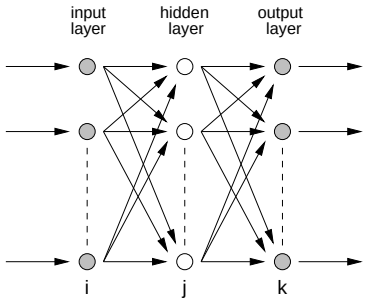


Figure 6.2: Multi-layer perceptron with 3 layers. The role of the input layer is to distribute the input features. No computation is performed at this stage. Hidden and output layers are the computational components with non-linear activation functions (generally sigmoids or tanh).

with the constraint

$$\sum_{j=1}^{J} w_j = 1. \tag{6.5}$$

As illustrated in Figure 6.1, this can be viewed as a particular case of Hidden Markov Model (HMM) with only one state. The HMM emission probability is then equal to the output of the unique probability density function tied to the state. Two models are built, one for the client, one for the world. Scores are equal to the product of frame likelihoods over the whole test utterance:

$$S_c = \mathcal{L}(\mathbf{X}|M_{client}) = \prod_{n=1}^{N} \mathcal{L}(\mathbf{x}_n|M_{client}) \tag{6.6}$$

$$S_w = \mathcal{L}(\mathbf{X}|M_{world}) = \prod_{n=1}^{N} \mathcal{L}(\mathbf{x}_n|M_{world}) \tag{6.7}$$

6.3.1.2 Multi-Layer Perceptron (MLP)

Another possibility to model speakers is to build discriminant models with Artificial Neural Nets (ANN), for example Multi-Layer Perceptrons (MLP). The MLP (Figure 6.2) is an artificial neural network composed of hierarchical layers of neurons arranged so that information flows from the input layer to the output layer of the network, i.e. no feedback connections are allowed. A good introductory book on artificial neural networks and MLP architectures is [48]. The main advantages of MLP against other systems, include discriminant capabilities, weaker hypotheses on the acoustic vector distributions and possibility to include easily a larger acoustic frame window as input to the classifier. The main drawback using MLPs is that their optimal architecture must be selected by trials and errors.

MLPs, one per client speaker, are discriminatively trained to distinguish between the client speaker and a background world model. MLPs with two outputs are generally used: one output is for the client and the other for the world class. In [17] it has been proved that if each output unit k of the MLP is associated to class categories C_k, it is possible to train the MLP to generate a-posteriori probabilities $\mathcal{P}(C_k|\mathbf{x}_n)$.

As explained in [84], the parameters of the MLP (weight matrices) are iteratively updated via a gradient descent procedure in order to minimize the difference between actual outputs and desired targets. In our case, in the training, target vectors $d(\mathbf{x}_n)$ are set to $[1, 0]$ and $[0, 1]$ when the input vector $\mathbf{x}_n$ is produced respectively by the client and by the world speaker. The training

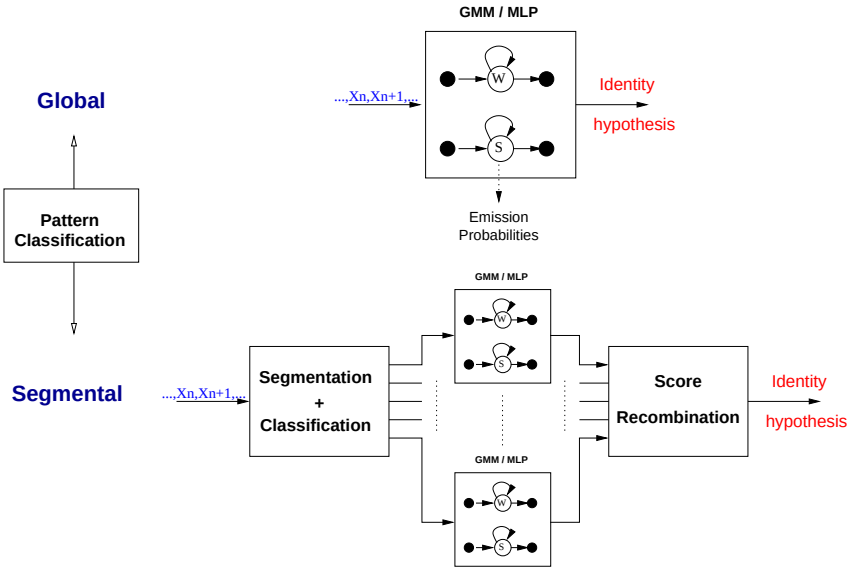


Figure 6.3: Global and segmental speaker verification system.

is said to be *discriminative* because it minimizes the likelihood of incorrect models (through the zeros of the target vector) and maximizes the likelihood of the correct model. The network attempts to model class boundaries, rather than to estimate accurately the probability density functions for each class. As shown with Bayes rule:

$$\mathcal{L}(C_k|\mathbf{x}_n) = \frac{\mathcal{L}(\mathbf{x}_n|C_k)\mathcal{P}(C_k)}{\mathcal{P}(\mathbf{x}_n)}, \tag{6.8}$$

the MLP training incorporates priors $\mathcal{P}(C_k)$ when modeling posterior estimates. These priors are not equal to real-life class prior probabilities but are dependent on the number of samples in the training set:

$$\mathcal{P}(C_k) = \frac{N_k^{ts}}{N^{ts}}, \tag{6.9}$$

where N^{ts} is the total number of training patterns and N_k^{ts} is the number of training patterns of the class k. Posteriors have then to be scaled to remove the dependency on the number of patterns in the training set[3]. Posteriors $\mathcal{L}(C_k|\mathbf{x}_n)$ are then usually divided by priors $\mathcal{P}(C_k)$ to obtain the so-called *scaled likelihoods* $\mathcal{L}(\mathbf{x}_n|C_k)/\mathcal{P}(\mathbf{x})$. Scaled likelihoods can be used in place of likelihoods at recognition time since the factor $\mathcal{P}(\mathbf{x})$ is not dependent on the class. Client and world scores are then computed according to the Eqs. 6.6 and 6.7 where likelihoods $\mathcal{L}(\mathbf{x}_n|client)$ and $\mathcal{L}(\mathbf{x}_n|world)$ are replaced with scaled likelihoods.

6.3.2 Segmental system

Our aim is to develop a segmental text-independent speaker modeling system (see Figure 6.3) where the speech sequence is segmented and segments are assigned to *categories*[4]. Categories are automatically determined by unsupervised clustering. Each category has its own model to distinguish clients from world. A segment is assigned to model corresponding to its category label. Finally, a score recombination of the individual models is performed.

[3]In speaker verification, this is very important, as the number of patterns in the client class is extremely low in comparison to the number of patterns in the world class.

[4]Below, we will use "category" instead of "class" to avoid confusion with client and world classes.

6.3.2.1 Segmental pre-processing

The segmentation is achieved using the Temporal Decomposition (TD), defined in section 4.4, finding quasi-stationary parts in the parametric representation. Intersections of TD interpolation functions permit to define speech segments $\mathbf{X}_a^b = [\mathbf{x}_a, \ldots, \mathbf{x}_b]$ and the utterance is decomposed into I non-overlapping segments:

$$\mathbf{X} = [\mathbf{X}_{s_1}^{s_2-1}, \mathbf{X}_{s_2}^{s_3-1}, \ldots, \mathbf{X}_{s_I}^{N}], \tag{6.10}$$

with $s_1 = 1$ and $s_1 \leq s_2 \leq \ldots \leq s_I \leq N$.

The next step is *unsupervised clustering*, implemented by Vector Quantization (VQ). The VQ codebook $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_L\}$ is trained by K-means algorithm with binary splitting (subsection 4.5.2). The training is performed using vectors positioned in gravity centers of the temporal decomposition interpolation functions, while the *quantization* takes into account entire segments using cumulated distances between all vectors of a segment and a code-vector (Equation 4.40). Temporal decomposition and vector quantization provide a symbolic transcription of the data in unsupervised way. Each vector of the acoustic sequence is declared as a member of a category C_l determined using the segmentation and the labelling. The number of categories is fixed by the number of centroids in the VQ codebook.

6.3.2.2 MLP Modeling

The same technique as for global modeling (section 6.3.1.2) is applied, but this time, L MLPs, where L is the number of categories, are used. They are respectively fed (both in the training and the test phase) with feature vectors having corresponding labels. For example, the MLP associated with category C_l provides a segmental score as follows:

$$S_{cl} = \prod_{x \in C_l} \mathcal{L}(M_{cl}|\mathbf{x})/\mathcal{P}(M_{cl}) \tag{6.11}$$

$$S_{wl} = \prod_{x \in C_l} \mathcal{L}(M_{wl}|\mathbf{x})/\mathcal{P}(M_{wl}), \tag{6.12}$$

where products encompass vectors being previously labelled as members of category C_l. Subscripts cl and wl denote respectively the client model for category C_l and world model for category C_l. Posterior probabilities in Eqs. 6.11 and 6.12 are also divided by prior probabilities, computed similarly as in Equation 6.9.

6.3.2.3 Score recombination

Among several techniques for score recombination the *linear recombination* was investigated in this study. The MLP scores are recombined linearly by simple additions:

$$S_c = \sum_{l=1}^{L} S_{cl} \tag{6.13}$$

$$S_w = \sum_{l=1}^{L} S_{wl} \tag{6.14}$$

$$\tag{6.15}$$

6.4 Experiments

6.4.1 Task description

The speaker verification system performances depend on many factors. Among the most influencing are the amount of training data, the duration of test segments, the microphone difference between training and testing data, the noise and the temporal drift[5], as pointed out by Doddington [34]. The data from National Institute of Standards and Technology (NIST) are an excellent tool to study the influence of many of those factors. Moreover, they offer a standardized training/test framework for multiple laboratories, so that the results can be quantitatively compared.

Both segmental and global systems were compared in the framework of the NIST-NSA'98 evaluation campaign[6] (NSA stands for National Security Agency). The data are selected from the SWITCHBOARD database, recorded over the telephone. The speech is spontaneous and no transcriptions, neither orthographic nor phonetic, are available. The *training set* consists of 250 male and 250 female subjects representing *clients* of the system (the sex mismatch is not studied in these evaluations, so that all experiences are strictly sex-dependent). For each client, the system is trained under three training conditions. Depending on the amount of data, they are denoted: 1S for one session, 2S for two sessions and 2F for two sessions with supplementary data ("full"). The *test set* comprises 2500 test files per sex, and per test condition, each "pretending" to be 10 clients. Three test conditions are defined depending on the available amount of test data: 3, 10 or 30 seconds. The total number of trials to do within NIST evaluations is: 2 sexes $\times$ 9 train–test conditions $\times$ 2500 test files $\times$ 10 trials per test file = 450000. For this study, only one training-test configuration is considered: two minutes or more for the training (condition 2F) and 30 s of speech for the tests. To evaluate the robustness of the new proposed segmental method, the tests are evaluated separately for matched and mismatched conditions, noted respectively SN (same telephone number) and DT (different microphone types). For modeling the world speakers, an independent set of 100 female and 100 male speakers with mixed carbon and electret microphones, was selected in the 1997 database of the previous evaluation.

6.4.2 Experimental setup

LPC-cepstral parameters were used in the feature extraction. A 30 ms Hamming window was applied to the speech every 10 ms in order to extract 12 LPC-cepstrum coefficients. The order of the LPC analysis was set to 10. A liftering was applied to the cepstral vectors followed by cepstral mean subtraction. The GMM-based system operated with 64 mixture components; the feature vector for GMM consisted of 12 LPCC, 12 ΔLPCC coefficients and Δenergy.

The MLPs used for the global systems had three layers, with 120 neurons in the hidden layer. For the segmental MLPs, the number of neurons in the hidden layer was reduced to 20. An important issue with MLPs is the choice of context of acoustic frames put on the input of MLP together with the current frame. While previous studies of Hennebert and Petrovska have shown, that larger context improved the verification efficiency [46, 71], in the segmental system, this context is limited by relatively small lengths of segments. Therefore, the context for the segmental system was set to 2 left-context and 2 right-context frames (C22), while the global system operated with the context of 5 left and 5 right frames (C55).

For the segmental system, the temporal decomposition was set up to detect 15 events per second in average. The vector quantization was trained on 1997 data with codebook size of $L = 8$. This small size was chosen because of relatively small amount of training data. With

[5] the delay between recording of training and test utterances.

[6] Detailed description of the verification task can be found in NIST report [61].

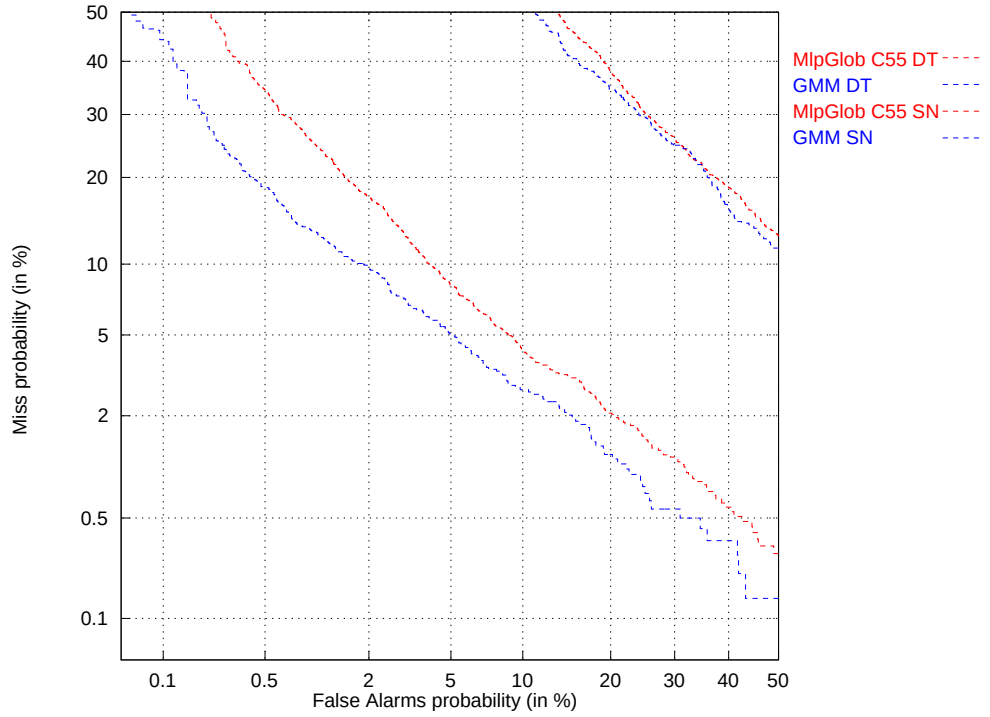


Figure 6.4: Global systems, GMM and MLP modeling, training condition 2F (2 minutes or more), test duration 30 sec, same number (SN) and different type (DT) training-test conditions.

2 minutes or more of speech, it is not possible to set this number to several tens (usual number of phonemes). One can thus consider the 8 categories as broad phonetic categories. The coherence of acoustic labelling among speakers was verified in informal listening tests.

6.4.3 Results

The results are presented as detection error tradeoff (DET) curves, giving the ratio of probabilities of false acceptings and false rejections on the test corpus, with the value of threshold T varying as parameter. In analogy with digital radio-electronics, these curves use to be named also ROC (receiver operating characteristics).

On Figure 6.4, one can see the comparison of GMM system with the global MLP one. The importance of mismatched training and testing conditions, as far as the microphone differences are considered, is visible on this figure. When the test segments come from a different handset type than the training speech material (DT curves), the error rates are increased roughly by a factor of four. The GMM system presents slightly better performances than the MLP one, so that it was used as the *base-line* system in all evaluations. However, taking into account the discriminant possibilities we can use with MLP, we adopted them for the segmental experiments.

For the segmental system, we begin with the presentation of performances of different categories given in Figure 6.5. Only the categories with dissimilar performances are shown. This figure demonstrates the fact that the categories perform differently and convey more or less informations about the speakers.

Finally, we give the results of the overall segmental system with linear recombination of the 8 category-dependent scores (Figure 6.6). With this simple recombination technique, we observe slightly worse performances of the segmental system for the easier training-test condition (SN). As far as the more difficult experimental condition (DT) is concerned, the segmental system

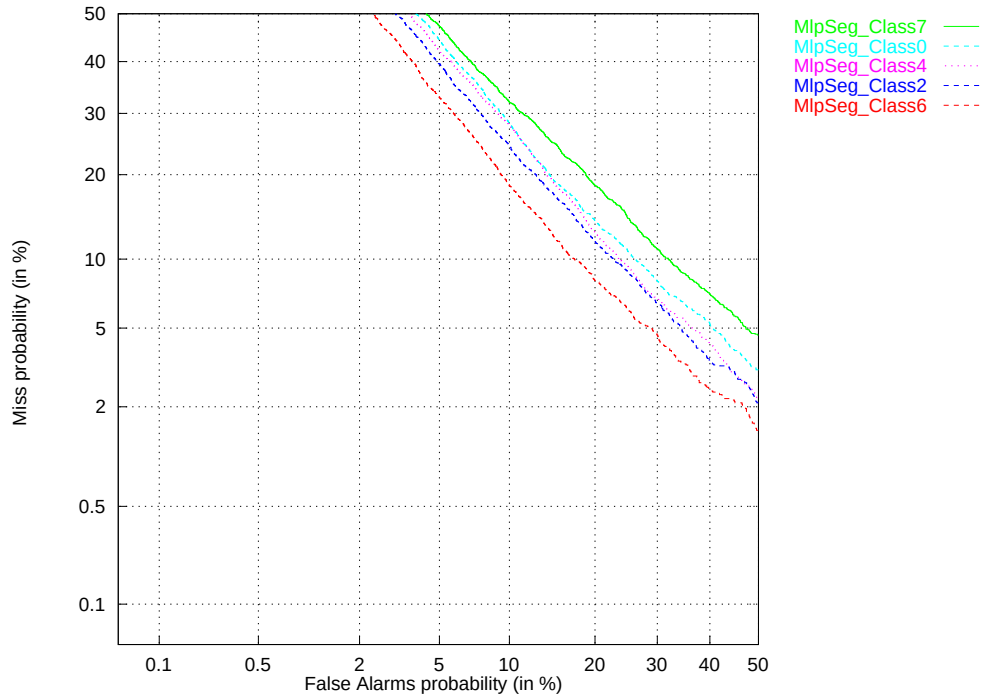


Figure 6.5: Segmental system, results by categories, training condition 2F, test duration 30 sec, same telephone number for training and test (SN).

outperforms the global one. This ensures us, that the segmentation and labelling are consistent, and opens a way to more sophisticated fusion techniques, where individual tuning of parameters corresponding to each category can be done.

6.5 Verification: Conclusions

This chapter describes the application of ALISP techniques of automatic derivation of speech units to text-independent segmental speaker verification task. The experimental results were obtained on NIST'98 evaluation data. We have verified, that the segmental system reaches similar performances as the global one, and even outperforms it in mismatched training/test conditions. However, in comparison with the baseline GMM system, GMMs showed still superior performance.

The main open issue of this approach is the merging of category-dependent results in order to obtain the global score, taking into account the discriminant performances of categories. Some computations, done standardly at the end of the speaker verification chain (normalization, threshold setting) must be moved towards category-dependent score computations. More general issue of this work is the determination of categories, currently done independently on the following steps. The categories should ideally be determined as optimal for the given scoring system in an iterative refinement.

Besides very low bit-rate coding, we demonstrated that ALISP techniques are potentially useful also in speaker verification, as they limit the human interaction necessary (and hence the number of errors introduced by humans, and the cost) and they approach the system to the data rather than to units more or less related with the text. However, work remains to be done when applying these methods efficiently in practice.

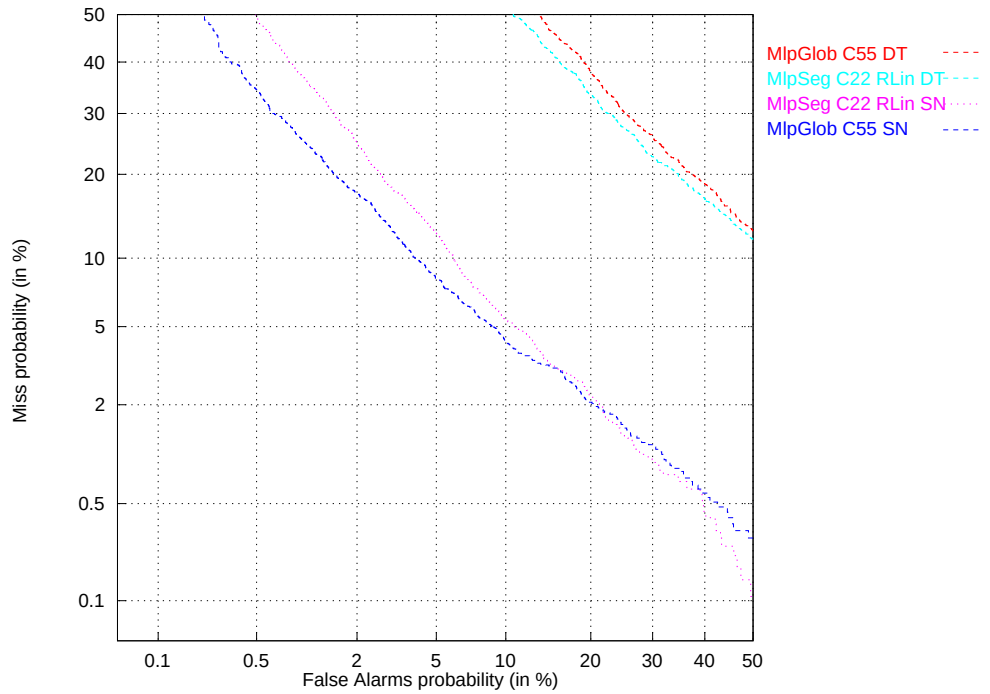


Figure 6.6: Global and segmental system, training condition 2F, test duration 30 sec, same number (SN) and different type (DT).

Chapter 7

Comparison of ALISP units with phonetic representation

The previous two chapters described applications of automatically derived units to coding and verification, with no need to convert those units to phonetic or lexical representations. However, the ultimate goal of our efforts is to build a *recognizer* making use of those units instead of currently used sub-word segments (such as phonemes, diphones, etc.). In a recognizer, the passage to lexical representations is needed at some place. Therefore, this chapter concentrates on possible ways of mapping ALISP units to "standard" ones. First experiments were done in 1↔1 mapping of ALISP units to phonemes: results were obtained using the phonetic labelling of one speaker of the BU corpus. It is however shown, that this mapping is only the first simple step – for real applications, more elaborated techniques of ALISP unit *composition* or *sequencing* have to be developed.

7.1 Motivations and methodology

The availability of phonetic transcriptions of some speech corpora led us to the idea of *comparing* the phonetic alignments using the ALISP segmentations on the same data. Two reasons are favorable for such experiments:

- the comparison of ALISP units with historically established ones can help us in the *evaluation* of used techniques. Traditional units exhibit consistency as far as their acoustic forms are concerned (at least for the human ear...). Regularities in the mapping of those units to automatically derived ones can be one of the criteria to evaluate the efficiency of ALISP techniques, besides the VLBR coding or the evaluation of obtained acoustic segments in listening tests.
- The phonetic-ALISP *mapping*, if successful, can in turn be useful for designing a recognizer. It is however necessary to note, that this mapping is only one possibility to use ALISP units in the speech recognition. The other, and in our opinion more interesting possibility, is to use ALISP units as the only intermediary level between signal (or feature vectors) and lexical representation.

The example of phonetic alignment of an utterance and of parallel ALISP segmentation is given in Figure 7.1. Our aim is to study the correspondence of these two sorts of units using all available data. We suppose, that phonemes are drawn from a finite set of size Z_p and that ALISP units are drawn from similar set of size Z_a. For the mathematical formulas[1], we are

[1]In computer processing, both types of units have their names given by ASCII sequences.

going to denote phonemes p_i, with $1 \leq i \leq Z_p$ and ALISP units a_i with $1 \leq i \leq Z_a$. The k-th occurrence of p_i or a_i in the corpus is denoted respectively p_{i_k} or a_{i_k}.

One stream of units has to be chosen as a *reference* one: in our case it was the phonetic alignment. We can then measure the correspondence of one reference unit with its ALISP counterparts by their *overlaps*. If for example the i-th occurrence of phoneme p_r overlaps with three ALISP units a_x, a_y, a_z, we define three absolute overlaps: $R(p_{r_i}, a_x)$, $R(p_{r_i}, a_y)$, $R(p_{r_i}, a_z)$. They can be expressed in any time unit (seconds, samples, frames). It is however desirable to compute *normalized* overlaps by dividing the absolute ones by the phoneme length:

$$r(p_{r_i}, a_x) = \frac{R(p_{r_i}, a_x)}{l(p_{r_i})}, \quad \text{etc.} \tag{7.1}$$

To evaluate the correspondence of all phonemes to all ALISP units globally, one may consider two possibilities:

1. *only one* ALISP unit is said to correspond to the i-th occurrence of phoneme p_r: that with the greatest overlap among all ALISP units overlapping with p_r (in our example, it is the unit `HD`). We will denote this unit a_{max} and the correspondence measure is given:

$$c_h(p_{r_i}, a_{max}) = 1. \tag{7.2}$$

   We will call this approach "hard" correspondence.

2. *all* ALISP units corresponding to a phoneme are considered, and a "soft" correspondence measure is given by their relative overlaps:

$$c_s(p_{r_i}, a_x) = r(p_{r_i}, a_x), \quad \text{etc.} \tag{7.3}$$

It is obvious, that the drawback of the first method is a penalizing of short ALISP units, one could even think about an ALISP unit with *no correspondence* to phonemes, because it was simply always "too short". Therefore, we used the later "soft" approach in the experimental work.

Using the above defined measure, the data of the whole corpus can be analyzed and the correspondence of phonemes and ALISP units enumerated in a *confusion matrix* $\mathbf{C}$. This matrix contains Z_p rows corresponding to phonemes and Z_a columns corresponding to ALISP units. The element $c_{i,j}$ is given by summing-up the correspondences over all N_{p_i} occurrences of phoneme p_i in the corpus:

$$c_{i,j} = \frac{\sum_{k=1}^{N_{p_i}} c_s(p_{r_i}, a_j)}{N_{p_i}}. \tag{7.4}$$

Obviously, this matrix is not arranged when evaluating it on the data. For the visualization, we have decided to convert the matrix to a pseudo-diagonal form, to make it more "readable". This can be ensured by re-arranging the order of columns; we should not re-arrange the rows, as phonemes have usually a meaningful order – they may be sorted for example according to phonetic classes. The re-arranging is done as follows: First, an ideal pseudo-diagonal matrix is constructed (it can be diagonal, only if $Z_p = Z_a$), and columns of the re-arranged matrix are allocated and marked empty. Then, the maxima of confusion matrix columns are searched. Next, the columns are processed from the one with greatest maximum to the lowest:

- the list of free columns of the re-arranged matrix is done.

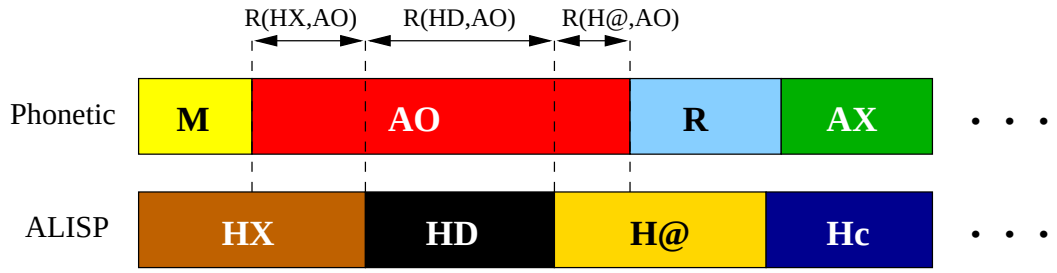


Figure 7.1: Phonetic alignment in parallel with ALISP segmentation. The absolute overlaps of ALISP units HX, HD, H@ with a reference phoneme AO are shown.

- Euclidean distance of each column of the ideal matrix corresponding to free columns of the re-arranged matrix, and of current column to sort-in, is computed.
- the current column is sorted-in at the place, where this distance is minimal.
- the newly filled column of the re-arranged matrix is marked occupied.

This process was implemented by a Matlab code, whose part is presented in Figure 7.2.

7.2 Experiments

7.2.1 Experimental setup

The experiments of comparison were done with the speaker F2B (female) from the Boston University Radio Speech Corpus, used in the VLBR coding experiments (section 5.4).

The *phonetic alignments* were created by the BU group and are included in the corpus. The group of Mari Ostendorf used segmental HMM recognizer[2] constrained by possible pronunciations of utterances [70]. In our comparison, the alignment files without hand-corrections (extension .lba) were taken for the phonetic-ALISP comparison. The phoneme set corresponds to the TIMIT one [39], but BU people distinguished stressed and unstressed vowels (for example AO and AO+1). We have not taken those differences into account, so we converted the labels of stressed vowels to the unstressed ones (for example, AO+1 to AO). The complete phoneme set is in Table 7.1.

The *ALISP segmentations* were those obtained by the 6-th generation of HMMs not post-processed by multigrams, as outlined in subsection 5.4.5. Their labels are all beginning with an H, the second letter is one of A...Z, a...z, 0...9, @, $.

7.2.2 Results and discussion

The resulting re-arranged confusion matrix grouping the correspondences $c_{i,j}$ defined in Equation 7.4 is given in Figure 7.3. Graphical form, more convenient for viewing, has been chosen, rather than classical textual representation. White color corresponds to zero correspondence, black to maximum value $c_{i,j_{max}}$=0.806, found for the correspondence of fricative SH to the ALISP unit H$.

It is obvious, that the found mapping is quite coherent, even if it is far from pure $1 \leftrightarrow 1$ correspondence. One can for example observe, that ALISP unit HA corresponds to all closures, but also to the pause and that the unit H$ with very pronounced correlation to SH is also linked to

[2] In this specific task, we can say *segmental aligner*.

```
% sort maxima of columns:
[dummy, imaxcol]=sort(max(conf));  % this is ascending, wants desc.
imaxcol=imaxcol(ncol:-1:1);
% create empty matrix and vector with new indices. 0 will mean that
% the position if free
newconf=zeros(nlin,ncol);
newind=zeros(1,ncol);
% go from the best maxima, and put the vector where it best coincides
% with the ideal pseudo-diagonal
for ii=1:ncol,
  col_to_put=conf(:,imaxcol(ii));
  % retrieve free positions
  free_pos=find(newind==0);
  % compute Euclidean distance of vector with all columns on free
  % positions
  aux=idig(:,free_pos);
  eucl=sqrt(sum( (col_to_put*ones(1,length(free_pos))-aux).^2 ));
  % put the vector where we've minimal eucl distance
  [dummy, mini] = min(eucl);
  index_to_put=free_pos(mini);
  newconf(:, index_to_put) = col_to_put;
  newind(index_to_put) = ii;
end
```

Figure 7.2: Matlab code for re-arranging of the confusion matrix for the visualization.

closures	BCL, DCL, GCL, PCL, KCL, TCL
stops	B, D, G, P, T, K, DX
affricates	JH, CH
fricatives	S, SH, Z, ZH, F, TH, V, DH
nasals	M, N, NG, EM, EN, NX
semi-vowels and glides	L, R, W, Y, HH, HV, EL
stressed vowels	IY+1, IH+1, EH+1, EY+1, AE+1, AA+1, AW+1, AY+1 AH+1, AO+1, OY+1, OW+1, UH+1, UW+1, ER+1
unstressed vowels	IY, IH, EH, EY, AE, AA, AW, AY, AH AO, OY, OW, UH, UW, ER, AX, AXR
others	PAU, H#, H, brth

Table 7.1: Original phoneme set of BU-corpus phonetic alignments. In our experiments, stressed and unstressed vowels were not distinguished (no +1 labels). Label H appears only once in the corpus, it should be H#.

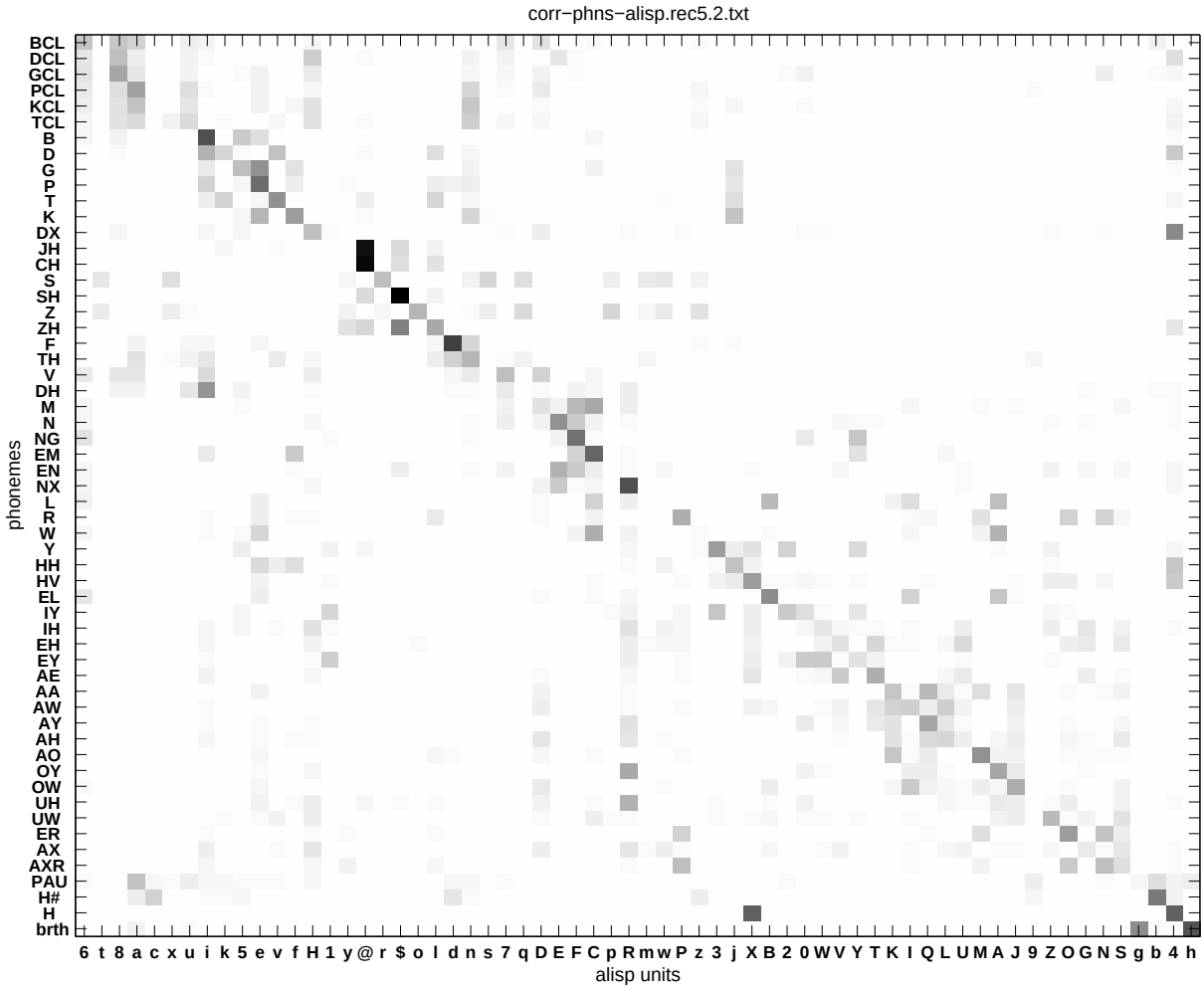


Figure 7.3: Confusion matrix between sets of ALISP units and of phonemes, estimated on the data of speaker F2B (Boston University Corpus). White color stands for zero correspondence, black for maximal correspondence $c_{i,j_{max}}$=0.806.

its voiced counterpart `ZH`, and to acoustically very closed affricates `JH` and `CH`. Another example is the unit `HF` with roughly the same correspondence to all nasals, and so on.

As expected, the $1 \leftrightarrow 1$ correspondence of ALISP units to phonemes was not found, so that we a recognizer could not be built by direct mapping of automatically derived units to phonemes, having kept the same pronunciation dictionaries (providing the lexical to phonetic level mapping). On the other hand, those pronunciation dictionaries, or *stochastic pronunciation networks* can be constructed using ALISP units. The following section gives some reflections on this topic.

7.3 Towards an ALISP-based recognizer

We have seen, that the use of classical pronunciation dictionary with a one-to-one mapping of ALISP units to phonemes is not possible. There are several possibilities to cope with this problem:

- stochastic mapping of *sequences* of phonemes to *sequences* of ALISP units can be done. This approach was studied by Deligne in [29]: likelihood maximization is applied to joint

segmentation of two streams of observations, where one can be the phonetic labelling and the other the sequence of automatically derived units. When the testing data are processed, the method finds the segmentation together with optimal "transcription" into sequences of phonemes. As we have seen in the section on multigrams (4.6), the observations can be either symbolic (in this case, the method is "discrete") or vectorial (here, not only statistics of sequences, but also emission PDFs come into mind).

- *composition* of ALISP units into word and phone models, proposed by Fukada in [37]. First, the basic units are derived in an unsupervised manner, similar to our work [6]. Then, the phonetic transcription is compared to ALISP segmentation[3] and composite models are constructed for the *words* of the corpus. In case the data do not contain sufficient number of examples of a word, the method can "back-up" to *phoneme models* composed in similar manner as the word ones.

The drawback of the two above mentioned methods is that they require an undetermined amount of phonetically rich material, which ought to be transcribed. Moreover, the need for pronunciation dictionaries and their inherent inaccuracy remains at heart of the problem. Constantinescu in [19] proposes an alternative method: *example-based word coding* using genetic algorithm. This method requires an acceptable low number of utterances of the target vocabulary. Two core ideas of this approach are vocabulary dependent word modeling (the sequence of sub-word models, describing a word, is determined based on an intra-dictionary similarity measure between the different vocabulary elements) and iterative combination of the best automatically generated word representations into new models, which may be implemented in a *genetic algorithm (GA)*.

We can conclude this chapter by stating, that the building of ALISP-based recognizer is not a straightforward task. The invested efforts should however be generously recompensed by the limitation of human efforts needed to create or modify such a recognizer. As it was mentioned in coding and verification chapters, if we obtain as efficient scheme, and in the same time we succeed in limiting the human labor (annotations, pronunciation dictionaries, etc.), it will be a great step towards the *real automatization* of speech processing, and it will also open the way to its easier implementation in different languages.

[3] Fukada et al. use the abbreviation ASU (acoustically derived segmental units).

Chapter 8

Discussion and Conclusions

Many partial conclusions have already been drawn in previous chapters. This final part of the thesis attempts to summarize the obtained results, to point out open problems, and to conclude by suggesting the possible future research directions in this domain.

8.1 Summary of results

This work was aimed at the research of techniques able to provide automatic segmentation of speech using a set of automatically derived segmental units. The method, we investigated, was based on the temporal decomposition, vector quantization and optionally on multigrams, and on stochastic modeling of sequences using Hidden Markov Models. Special attention was devoted to the refinement of set of models in iterations of segmentations and parameter re-estimations.

First tested application was the very low bit-rate coding, where issues specific to the re-synthesis of speech in the decoder had to be solved. Using given set of units, intelligible speech was obtained at mean bit-rate of 120 bps (for unit encoding) in two sets of speaker-dependent experiments.

The second application was the use of those units as a pre-processing for segmental speaker verification systems based on multi-layer perceptrons. In the experiments conducted within the NIST'98 campaign, the segmental system has shown some advantages in comparison to the "global" one based on MLPs, but it did not outperform the state-of-art GMM modeling approach.

Correspondence of ALISP units with phonetic transcriptions was studied with one speaker data from the Boston University corpus. Here, we have found a consistent phoneme to ALISP mapping, but as expected, this mapping was far from one-to-one correspondence.

8.2 Open issues

Of course, this thesis could not bring solution to all problems of automatic unit derivation and their use in speech processing. Some problems were known before the beginning of this work and have not been resolved, some emerged during this thesis. We can divide them into "general" ALISP issues, and to application-specific problems.

8.2.1 General ALISP issues

There are many techniques common in the ALISP approaches and in the classical speech processing, thus one finds also similar problems. On the beginning of the speech processing chain, there are the *data*. The choice of database is a crucial problem, even if the constraints are much more

relaxed in ALISP processing, than is approaches, where annotations or phonetic alignments are needed. There are questions of quality of the acoustic signals (telephone or wide-band), speaker-dependent vs. speaker-independent approach, language of speakers, dialects, speaking styles, background noises (recording in studio, in office, or in the street ...), and first of all the amount of the data: the constraints of reliability of statistical estimations (demanding as much data as possible) and physical constraints (capacity of storage medias and speed of processors) are classically contradictory.

Follows the *parameterization*. In our studies, the classical approach based on LPCC feature extraction in centi-second frames was used, but as it was pointed out by Hermansky [47] and others, this is a deliberate choice: while we are relying on the data as much as possible in the rest of the processing chain, the parameterization is often the place, where we introduce the most of heuristic knowledge. It would be thus desirable to verify the studied approaches with feature extractions more closed to the data (relative spectral coefficients (RASTA), linear discriminant analysis (LDA), temporal filtering, and others).

Next comes the key-point of our work: the determination of initial speech transcriptions and of the set of units. Here, a lot of issues stay open. If we agree, that the initial segmentation should be done accordingly to the stationarity of signal in the parts (and this is already a deliberate choice), there are more methods for determination of such stable portions than the temporal decompositions. Section 3.3 cites some of them: spectral variation function evaluation, measuring the conformity of feature vectors to the mean trajectory of a segmental stochastic model, etc. Their performances should be compared to the temporal decomposition.

Even more challenging is the following problem of determination of *classes*. The first step is the determination of their *number*, done a-priori in our experiments. More sophisticated would be a spectral distortion measure with successive splitting of classes until obtaining the desired global distortion. The optimal solution would be a determination of this number depending on the target application and a possibility to change it dynamically during the training (and maybe also during the operation) by merging and splitting of classes. Another issue is the *measure* for determination of classes: Euclidean distance of LPCC features is certainly the very common one, but again, it should be tuned according to the application and its final goal (discrimination of speakers in the verification is not the same problem, as preserving the speech quality in coding, nor as the recognition accuracy).

The sequencing of units by multigrams has its counterpart in standard speech processing, where the use of tri-phones, quint-phones in the recognition, or of di-phones in the synthesis, is current. However, also this sequencing is not without problems, as it increases the number of parameters in the system, decreases in the same way the relative amount of data available for training a parameter, and so one.

The stochastic modeling is closely linked to the previous sections. In the HMM framework, there are important choices to be done, including the feature vector (shall we use Δparameters ?), of numbers of states, initialization of transition probability matrices, and of probability distribution function modeling (discrete/continuous, single Gaussian/mixtures, etc.). The iterative refinement of models, which has been justified by the increase of likelihood, should be rather linked with a overall quality criterion of the entire system.

After having cited more or less *technical problems*, we are coming to the general *global* question on ALISP systems. The first is the **objective evaluation of quality of units**. Of course, we have seen the evaluation of this quality by low rate coding, subjective listening tests of representatives of found classes, and by the comparison of automatic segmentation with phonetic alignments. Those are however *indirect* with regard to a target application. The final objectives (for example the word error rate (WER) in the recognition, or the equal error rate (EER) in the verification) would be the ultimate measures of suitability of units. It is however difficult

to find a direct dependency of those measures on the choice of the set of units, so that we will probably not able to quit completely the auxiliary criteria cited above.

Another global question is the use of standard speech processing (mainly recognition) techniques in the automatic derivation of units. The used techniques have proved their efficiency when used with heuristically derived units. An almost philosophical question is, whether those techniques will perform as well when employed to derive the speech description, and if a completely other formalism would not be needed ...

8.2.2 Coding specific issues

Even if the coding is the most "direct" application of ALISP units and the most "closed" to the data, there is still an amount of open problems:

The most important one is the *evaluation of decoded speech quality.* In our experiments, this was done in informal listening tests. Surely, even for subjective evaluation, formal listening tests should be performed, evaluating DRT (diagnostic rhyme test) and DAM (diagnostic acceptability measure). However, those tests always involve extensive human efforts and are thus costly (if a professional company is engaged) or difficult to organize. Therefore, *objective* criteria, not necessitating the presence of listeners, and though giving a measure of human acceptability are needed. The works of Ghitza [44] on the use of perceptual models for estimation of the quality of coded speech can bring new insights on this problematic. An interesting topic is to study the influence of different factors (precision of spectral envelope coding, transitions, prosody) on the overall subjective speech quality. Not only the absolute values of those parameters, but also their temporal derivatives are important as expressed by Knagenhjelm and Kleijn [56]: "spectral dynamics is more important than spectral distortion".

Not less challenging are issues of the ratio of *coding rate* to the *quality* of decoded speech. As we have seen, it is difficult to evaluate it as we do not dispose of objective measure of the quality. In the experimental work, some tuning of this ratio was done using the unigram language model (accentuating the emission probabilities of frequent models), and the related language model weighting factor γ, but this was certainly too simplist. Optimal scheme would adjust this ratio automatically according to the desired bit-rate, or to the characteristics of the channel.

As mentioned in the respective chapter, the main technical issues of VLBR coding include the derivation of *synthesis units* from the coding ones, the choice of *representatives* for the resynthesis of speech, the smoothing on the boundaries of segments, and the synthesis itself. A lot has to be done especially in the investigation of diphone-like units (definition of a synthesis unit as transition of two coding ones), which was theoretically previewed, but not yet tested in experiments.

Also, there is a need to code efficiently the other parameters of speech (pitch, energy, timing) on segmental basis. In our opinion, it is possible to use the same formalism as that used for unit derivation (stability detection, clustering) for finding characteristic prosodic patterns, and to associate them with coding units.

All experiments conducted so far in the coding were speaker-dependent. There are therefore all the issues of *speaker adaptation* (which can be resolved by the normalization of voices to a generic speaker, or by the adaptation of models) which have to be investigated. Fortunately, here we can rely on many references, as those issues have been extensively studied in the recognition [25, 57]. It is however to be verified, how the standard approaches (maximum-likelihood linear regression (MLLR) and others) do perform in the coding with ALISP units, where the objectives are different from the recognition.

Linked to the previous problem is the *speaker modification.* If not used, the voice of decoder will always be that of the speaker(s), who created the training set. There is a possibility to transmit modification parameters, allowing to transform the voice of a generic speaker to the

desired one [88]. It has however been shown by Ribeiro and Trancoso [78, 79], that this information increases considerably the bit-rate. In some applications (military for example), we would accept the alteration of voice at the price of very low rate.

Having developed a speaker-independent coder, it will be necessary to verify its functionality in language-independent mode. The availability of speech corpora in different languages [3, 1] makes this verification easier. More difficult will be to conduct the subjective tests in multiple languages.

The real-world implementations of proposed coding will concern also other parameters of the coder, than the speech quality: the *complexity* and *delay* are the most important. While one accepts a longer delay in very low rate schemes, and in some applications (storage) the delay is not crucial, the complexity is among the most important points. While the determination of unit set can be run off-line, the coder and especially its HMM portion should be optimized. The questions of state sharing, parameter tying, search space limitation, and efficient search, emerge as very important ones.

8.2.3 Verification specific issues

Besides generic problems, as is the choice of number of units, parameterization, and others, there are two principal issues in the segmental verification with automatically derived units:

First, the determination of units should be done with respect to the ultimate goal of the verification: to *discriminate the speakers.* Until now, those units were derived using the general criteria: stationarity of parameters, and minimization of intra-class and maximization of inter-class variability. Ideally, the units should be selected to accentuate differences between speakers.

Another issue is closely linked to the previous one: as soon as the classes are defined, it is necessary to determine their discriminative power and to use those measures in the merging to obtain final client–world scores. The fact, that the reference systems were not outperformed by the segmental one in the experiments, were in our opinion caused mainly by this not taking into account the different discriminative powers of classes. It is necessary to define a performance measure, to evaluate this measure for different classes on the training data (possibly independently for each speaker, as the classes can behave differently for different speakers) and to use them in the recombination of the class-dependent scores.

Finally, as it was mentioned in chapter 6, we should move some operations done classically at the end of verification chain (normalization, threshold determination), towards the class-dependent score determination. This would increase the number of operations needed in the training, but should improve the overall performances of the system.

Similarly as in the previous subsection, the issues of implementation of this system in real world (computational complexity, storage needs, etc.) come into mind, but those are similar for both the "global", and phonetic- or ALISP-based segmental systems.

8.3 Future research directions

Generally speaking, all the problems mentioned above are worth to be resolved in order to allow the full implementation of ALISP units in the current speech processing. The solution of all of them is however beyond of what can do one person or even one entire laboratory in several years; some of the cited issues having been substantial for the speech processing research since it started. In our efforts, we can define two perspectives:

In short-time perspective, we are going to concentrate on practical aspects of the VLBR coding, and hope to cooperate with industrial companies to implement proposed approaches in

practice[1].

In long-time perspective, the goal is to work on an ALISP-based recognizer. This work will certainly have to be done in cooperation with other laboratories, as the unit determination is only one of steps in a recognizer design.

[1]A proposal to French "Réseau National de Recherches en Télécommunications" (RNRT) has been submitted.

Appendix A

Temporal decomposition – details

This appendix presents in detail the computation of target parameter vectors and interpolation functions (IF) in the temporal decomposition (TD). The algorithm is based on Bimbot's article [12] and on the technical report [11], which is implemented in the software package `td95`.

Recall, that TD attempts to approximate the parameter matrix $\mathbf{X}$ by the product of target matrix $\mathbf{A}$ and interpolation functions matrix $\mathbf{\Phi}$:

$$\underset{(P\times N)}{\hat{\mathbf{X}}} = \underset{(P\times m)}{\mathbf{A}} \; \underset{(m\times N)}{\mathbf{\Phi}}\,, \tag{A.1}$$

with matrix sizes in the bottom line. The original matrix $\mathbf{X}$ should be approximated with minimum square error:

$$\min \sqrt{\frac{\sum_{t=1}^{N} \|\mathbf{x}(t)-\hat{\mathbf{x}}(t)\|^2}{\sum_{t=1}^{N} \|\mathbf{x}(t)\|^2}}, \tag{A.2}$$

with the constraint of compactness of IFs in time. The algorithm consists of initial search of IF for each $t \in [1, N]$ (all time variables in this Appendix are indexing spectral vectors), post processing of IFs, initial computation of targets, and finally of local iterative refinement of targets and corresponding sub-matrix of IFs.

A.1 Initial search of one interpolation function

The variable t defines spectral sampling instant, for which the initial IF is computed. $[t_1, t_2]$ is a time interval centered around t: $t_1 = t - a$, $t_2 = t + a$ and T is its length: $T = t_2 - t_1 + 1$. Also defined is a prototype function $W(t)$, centered around t and included in $[t_1, t_2]$:

$$W(t) = \begin{cases} 1 & \text{for} \quad \theta_1 < t < \theta_2 \\ \\ 0 & \text{otherwise,} \end{cases} \tag{A.3}$$

where $\theta_1 = t - \alpha$, $\theta_2 = t + \alpha$ with $\alpha < a$, so that $t_1 < \theta_1 < t < \theta_2 < t_2$. This is depicted in Figure A.1a. All following processing works with sub-matrix $\mathbf{Y}$ spanning the interval $[t_1, t_2]$: $\mathbf{Y} = [\mathbf{x}_{t_1}, \mathbf{x}_{t_1+1}, \ldots, \mathbf{x}_{t_2}]$ of dimensions $P \times T$.

The first step in search of IF is a singular-value decomposition (SVD) of this sub-matrix:

$$\underset{(T\times P)}{\mathbf{Y}^T} = \underset{(T\times P)}{\mathbf{U}^T} \; \underset{(P\times P)}{\mathbf{D}} \; \underset{(P\times P),}{\mathbf{V}} \tag{A.4}$$

where $\mathbf{V}$ is orthogonal matrix of principal components of $\mathbf{Y}$, $\mathbf{D}$ is diagonal matrix of singular values, giving the "importance" of vectors of $\mathbf{V}$, and $\mathbf{U}$ can be understood as normalized parameter trajectories in the new space. It should be noted, that $\mathbf{U}$ and $\mathbf{V}$ are both orthonormal, so that $\mathbf{D}$ is the only one carrying magnitude information. The IF can be calculated as linear combination of rows of $\mathbf{U}$ with addition of a constant:

$$\phi(t) = b_0 + \sum_{i=1}^{P} b_i u_i(t), \tag{A.5}$$

where b_i are weighting factors and $u_i(t)$ are rows of $\mathbf{U}$. As those rows do not have the same importance in approximation of $\mathbf{Y}$, it is desirable to truncate $\mathbf{U}$ only to first p rows:

$$\phi(t) = b_0 + \sum_{i=1}^{p} b_i u_i(t), \tag{A.6}$$

with $p < P$. In matrix notation:

$$\phi(t) = \mathbf{b}^T \hat{\mathbf{U}}, \tag{A.7}$$

where $\mathbf{b}$ is column vector of dimension $p+1$ with coefficient b_0 in the end and $\hat{\mathbf{U}}$ is truncated matrix $\mathbf{U}$ of dimensions $(p+1) \times T$, with constants $1/\sqrt{p}$ in the $(p+1)$-th row. For determination of p, various criteria are admissible, but absolute threshold is satisfactory:

$$\sum_{i=1}^{p} d_i < D_{thr}, \tag{A.8}$$

where d_i are the diagonal elements of matrix $\mathbf{D}$. The coefficients $\mathbf{b}$ should ensure desired form of $\phi(t)$, e.g. its concentration around prototype function $W(t)$. The similarity criterion can be written as:

$$S(\phi, W) = \frac{\sum_{t=t_1}^{t_2} W^2(t)\phi^2(t)}{\sum_{t=t_1}^{t_2} W^2(t) \sum_{t=t_1}^{t_2} \phi^2(t)}, \tag{A.9}$$

and it can be shown, that maximization of this measure is equivalent to finding $\mathbf{b}$ as eigenvector associated with the maximal eigenvalue of:

$$\mathbf{Rb} = \lambda \mathbf{b}, \tag{A.10}$$

where the elements of $(p+1) \times (p+1)$ matrix $\mathbf{R}$ are given as:

$$r_{ij} = \sum_{t=t_1}^{t_2} W^2(t)\hat{u}_i(t)\hat{u}_j(t). \tag{A.11}$$

With vector $\mathbf{b}$ evaluated, $\phi(t)$ can be computed using Equation A.7. The final step consists of smoothing, truncation and normalization of initial IF. The smoothing is done using 3-tap MA filter:

$$\bar{\phi}(t) = \frac{1}{2\gamma + 1}[\gamma\phi(t-1) + \phi(t) + \gamma\phi(t+1)], \tag{A.12}$$

with $\gamma \geq 0$. The smoothed version is used in the following steps: $\phi(t) = \bar{\phi}(t)$. The truncation operates with two thresholds: one absolute, another for local minima. Left and right truncation points t_{t1} and t_{t2} are found in similar way, only the determination of the left one is described here:

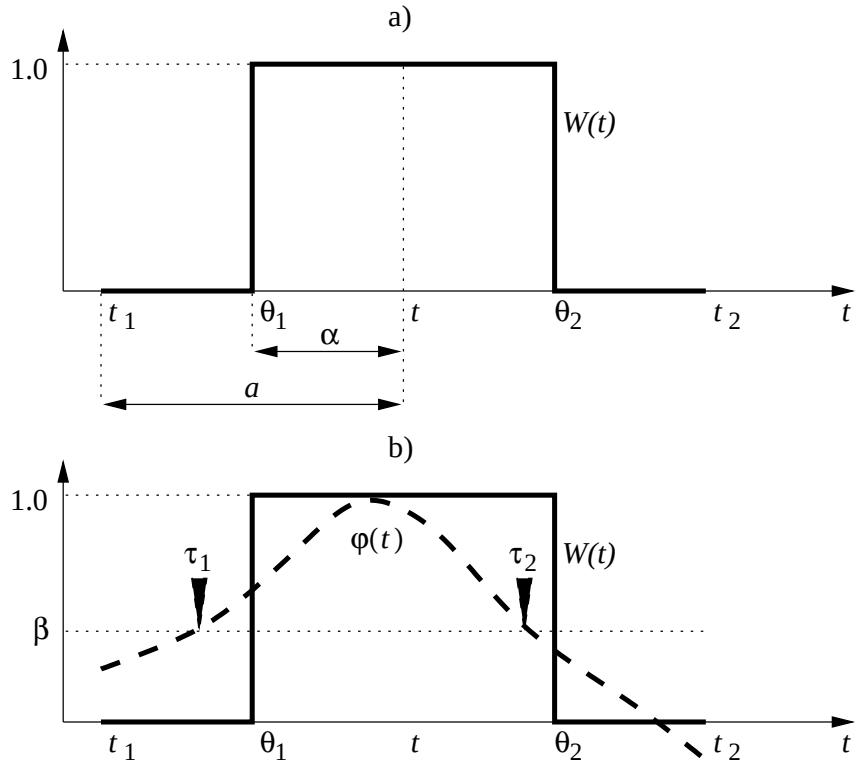


Figure A.1: Finding one interpolation function of TD: a) initial window parameters. b) initially found IF and its limit values for adaptive windowing.

- set t to t_m (index of maximum of $\phi(t)$).
- while $\phi(t) > \mu_1 \phi(t_m)$, decrement t.
- while $\phi(t) > \phi(t-1)$ (not yet in local minimum) and $\phi(t) > \mu_2 \phi(t_m)$, decrement t.
- set $t_{t1} = t$.

Then, truncated version of $\phi(t)$ is defined as:

$$\tilde{\phi}(t) = \begin{cases} \phi(t) & \text{for} \quad t_{t1} < t < t_{t2} \\ \\ 0 & \text{otherwise,} \end{cases} \tag{A.13}$$

The truncated version is used in the suite: $\phi(t) = \tilde{\phi}(t)$, and in the final step, it is normalized so that its maximum value equals to 1.

A.2 Adaptive windowing

It is very probable, that having chosen $W(t)$ deliberately centered around t, the IF will not match with this window. Therefore, adaptive windowing is introduced to maximize the fit of $\phi(t)$ to $W(t)$.

The computation of IF in iteration h is governed by number of parameters: t_h, t_{1h}, t_{2h}, θ_{1h}, θ_{2h}, a_h, α_h. The function calculated using these limits is denoted $\phi_h(t)$. With computed IF, the instants delimiting region, where ϕ_h is above a threshold β, are found (see Figure A.1b):

$$\phi_h(t) > \beta \quad \text{for} \quad \tau_{1h} < t < \tau_{2h}. \tag{A.14}$$

The parameters governing the computation of $(h+1)$-th IF are then obtained as follows:

$$\theta_{1h+1} = \begin{cases} \theta_{1h} - \delta & \text{if} \quad \tau_{1h} < \theta_{1h} \\ \theta_{1h} + \delta & \text{if} \quad \tau_{1h} > \theta_{1h} \\ \theta_{1h} & \text{if} \quad \tau_{1h} = \theta_{1h}, \end{cases} \tag{A.15}$$

$$\theta_{2h+1} = \begin{cases} \theta_{2h} - \delta & \text{if} \quad \tau_{2h} < \theta_{2h} \\ \theta_{2h} + \delta & \text{if} \quad \tau_{2h} > \theta_{2h} \\ \theta_{2h} & \text{if} \quad \tau_{2h} = \theta_{2h}, \end{cases} \tag{A.16}$$

which means: "move the concentration of prototype function towards concentration of ϕ". Then compute new time:

$$t_{h+1} = \frac{1}{2}(\theta_{1h+1} + \theta_{2h+1}) \quad \text{and} \quad \alpha_{h+1} = \frac{1}{2}(\theta_{2h+1} + \theta_{1h+1}), \tag{A.17}$$

meaning "get center and half-width of concentrated part of prototype function", and finally:

$$a_{h+1} = \rho\alpha_{h+1} + \eta, \tag{A.18}$$

where the half-length of prototype function is obtained. The limits are then calculated:

$$t_{1h+1} = t_{h+1} - a_{h+1}, \quad t_{2h+1} = t_{h+1} + a_{h+1}. \tag{A.19}$$

The constant δ stands for adaptation step while ρ and η determine the total width of prototype function as function of its "concentrated" portion.

Several possibilities can occur when comparing W_{h+1} to W_h:

- $W_{h+1} \equiv W_h$: W_{h+1} and ϕ_{h+1} are adapted. Keep ϕ_h and restart with new conditions ($t = t+1$, $h = 0$, initial a and α, etc.).
- W_{h+1} never seen before: memorize conditions, continue with W_{h+1} as new prototype function.
- $W_{h+1} \equiv W_q$, where $q < h+1$: cyclic behavior. Observe all ϕ's between q and $h+1$, and keep the one with best match between $W(t)$ and $\phi(t)$ (criterion in Equation A.9).
- W_{h+1} has the same parameters as one of W's observed ever before: we know the continuation, do not continue as this "path" was already investigated and the final IF was recorded. Restart with new initial conditions.

A.3 De-correlation and adaptive refinement

The result of previous step is number of IFs. Many of them can be strongly correlated. Therefore, a similarity measure is evaluated for each pair (i, j) of IFs:

$$c_{ij} = \frac{\sum_{t=1}^{N} \phi_i(t)\phi_j(t)}{\sqrt{\sum_{t=1}^{N} \phi_i^2(t)} \sqrt{\sum_{t=1}^{N} \phi_j^2(t)}}, \tag{A.20}$$

and if this measure is above a threshold (typically 0.5), one of IFs is discarded (that with lower coherence of $\phi(t)$ and $W(t)$ – Equation A.9).

The final step of TD is *iterative refinement* of IFs and targets. First, targets are computed:

$$\mathbf{A} = \mathbf{X}\mathbf{\Phi}^{\#}, \tag{A.21}$$

where $\mathbf{\Phi}^{\#}$ is pseudo-inverse of $\mathbf{\Phi}$. The refinement of i-th IF is then conducted locally: if τ_{1i} and τ_{2i} denote left and right border of IF, a *local* matrix $\mathbf{Y}_i$ is built by taking vectors: $\mathbf{Y}_i = [\mathbf{x}_{\tau_{1i}-k}, \ldots, \mathbf{x}_{\tau_{2i}+k}]$, so that its dimensions are $P \times N_i$, where $N_i = \tau_{2i} - \tau_{1i} + 1 + 2k$. Local target matrix $\mathbf{A}_i$ is filled with targets corresponding to events "active"[1] in interval $[\tau_{1i}-k, \tau_{2i}+k]$ whose number is denoted n_i. New local matrix $\mathbf{\Phi}_i$ can then be computed:

$$\mathbf{\Phi}_i = \mathbf{A}_i^{\#}\mathbf{Y}_i, \tag{A.22}$$

with $\mathbf{A}_i^{\#}$ standing for pseudo-inverse of $\mathbf{A}_i$. Only i-th IF is updated using this refined local matrix and the process is continued for all IFs. As following step, the IFs are renormalized to let their sum be 1 for all times:

$$\sum_{i=1}^{m} \phi_i(t) \equiv 1 \quad \text{for} \quad 1 \leq t \leq N. \tag{A.23}$$

The refinement (computation of targets – Equation A.21, and local refinements of ϕ's) is iterated and stopped if pre-determined number iterations is reached, or when the change of mean square error between original and approximated parameters (Eq. A.2) is no more significant.

[1] "active" means, that intersection of non-zero portion of interpolation function $\phi_j(t)$ with interval $[\tau_{1i}-k, \tau_{2i}+k]$ is nonempty.

Appendix B

Résumé en français

B.1 Introduction

Beaucoup de systèmes modernes de Traitement Automatique de la Parole (TAP) sont basés sur des unités de type *sous-mot*. Dans les systèmes de reconnaissance vocale (à vocabulaire illimité), ces unités constituent le niveau intermédiaire entre la description acoustique (ou paramétrique) et le niveau lexical, dans la reconnaissance du locuteur, une pré-segmentation avec ces unités avec de multiples systèmes de décision permet d'obtenir de meilleurs résultats que les systèmes utilisant une modélisation "globale", et finalement dans le codage à très bas débit (very low bit-rate, VLBR), la transmission de l'information sur chaque trame n'étant plus possible, ces unités déterminent l'information symbolique transmise dans le canal ou stockée.

Classiquement, ces unités sont basées sur celles établies historiquement : sur des phonèmes, leurs dérivés (diphones, phonèmes en contexte, ...), syllabes, ou autres. Pour la définition d'un jeu d'unités et pour la détermination de leurs positions dans le signal de parole (un alignement), des connaissances detaillées de phonétique et de linguistique sont requises. De plus, on doit investir des efforts humains considérables pour les transcriptions et/ou annotations des bases de données (BD); il est connu, que ces étapes sont les plus coûteuses et les plus sensibles aux erreur humaines dans le procédé de création des BD.

B.2 Solution proposée : ALISP

Les travaux effectués pendant cette thèse ont porté sur une approche alternative : détermination *automatique* des jeux d'unités ainsi que des transcriptions des BD. Les techniques regroupées sous un nom générique ALISP (Automatic Language Independent Speech Processing – Traitement Automatique de Parole, Indépendant de la Langue) se basent sur des *données* et tentent de limiter au minimum les connaissances a-priori nécessaires. Recherchant un équilibre entre la précision de la description et son économie, ces techniques détectent des régularités dans le signal (ou sa paramétrisation) pour faire émerger sa structure. L'utilisation de ces unités est directe dans les domaines, où la représentation symbolique ne constitue qu'un niveau intermédiaire entre deux signaux (le codage), ou auxiliaire (pré-segmentation dans la vérification). Au cas, où l'on exige une transcription linguistique (reconnaissance vocale), des techniques de mise en correspondance (mapping) des unités ALISP et des phonèmes doivent être élaborées. Mieux encore, il est possible de remplacer des dictionnaires de prononciation classiques par ses homologues constitués avec des unités déterminées automatiquement.

B.3 Techniques utilisées

Sur un corpus de parole donné, la détermination des unités s'effectue en deux étapes principales : dans la première, nous définissons le jeu d'unités et nous recherchons une segmentation initiale de la BD. Dans la deuxième, ces unités sont modélisées par des modèles stochastiques. Le système est ainsi *appris* et peut continuer à fonctionner avec un signal de parole inconnu.

La première étape englobe quatre pas : des vecteurs de paramètres sont extraits des signaux; nous utilisons une paramétrisation classique LPCC (cepstres calculés par la prédiction linéaire). Ensuite, nous appliquons la *décomposition temporelle (DT)*. Cette méthode, introduite par Atal et perfectionnée par Bimbot, approxime une matrice de paramètres par des vecteurs-cibles et des fonctions d'interpolation (FI). Les FI, déterminant des parties quasi-stationnaires du signal, peuvent définir une première *segmentation* de la parole. Les segments trouvés subissent une classification non-supervisée, réalisée ici par une *quantification vectorielle (QV)*. En appliquant la DT et QV, nous obtenons une *transcription* du signal. Des unités trouvées peuvent éventuellement être traitées par une autre technique, dite de *multigrammes (MG)*. Cette méthode, dont nous connaissons plusieurs variantes (discrète, des définitions différentes de multigrammes continus) permet de détecter des *séquences caractéristiques* d'unités dans le corpus d'apprentissage. En utilisant cette technique sur les unités déterminées automatiquement, nous obtenons un parallèle à des approches syllabiques ou diphoniques dans le traitement classique.

Dans la deuxième étape, les unités trouvées par la combinaison DT+QV ou DT+QV+MG sont *modélisées* par les *Modèles de Markov Cachés (HMM)*. Cependant, ce formalisme, utilisé largement en reconnaissance de parole, ne sert pas qu'à produire des modèles, mais contribue lui-même à un affinement du jeu d'unités par des itérations de segmentation du corpus (un alignement des HMM avec les données) et de ré-estimation des paramètres des modèles.

Les sorties de la technique proposée sont donc triples : un *dictionnaire d'unités,* avec (si exigées) leurs probabilités a-priori déterminées sur le corpus d'apprentissage, une *segmentation* du corpus d'apprentissage et un *jeu de modèles.*

B.4 Application No. 1 : Codage à très bas débit

Dans le codage à très bas débit, nous nous sommes positionnés dans le cadre des vocodeurs *phonétiques*, étudiés par Ribeiro, Trancoso, Ismail, Tokuda et autres. Contrairement à ces auteurs, qui utilisent à l'unanimité des phonèmes, nous avons choisi des unités ALISP pour les *unités de codage.* Nous pouvons les détecter dans la parole à l'entrée du codeur, *transcrire* la parole en termes de ces unités, et envoyer leurs indices dans le dictionnaire au décodeur. Nous devons également transmettre l'information supplémentaire sur la prosodie (fréquence fondamentale, énergie, longueur des segments). Au décodeur, la parole de sortie s'obtient par une *synthèse vocale.* D'abord, nous dérivons l'information sur des *unités de synthèse* à partir des unités de codage. Ensuite, nous recherchons des *représentants* dans un dictionnaire, créé dans la phase d'apprentissage. L'ensemble de paramètres du représentant choisi contrôle le synthétiseur en association avec l'information prosodique.

Nous avons effectué les expériences en mode mono-locuteur, avec les données de deux BD. Dans le premier jeu d'expériences, nous avons utilisé la BD française PolyVar créée à l'IDIAP (Martigny, Suisse). Cette base de données a été enregistrée à travers le téléphone. Nous avons utilisé les données du locuteur, qui avait fait le plus d'appels (218), que nous avons divisés en 174 pour l'apprentissage et 44 pour les tests. Nous avons effectué une paramétrisation avec 10 coefficients cepstraux en trames de 20 ms (recouvrement 10 ms). La soustraction de la moyenne cepstrale (CMS) a été faite pour chaque appel. Nous avons ensuite appliqué la DT, ajustée afin de produire 15 cibles par seconde en moyenne. Sur les segments obtenus, nous avons

appris un dictionnaire (code-book) de QV à 64 vecteurs-codes. Dans l'étape suivante, les MG ont détecté 1514 séquences caractéristiques de longueur 2, et 88 3-grammes. Le nombre total d'unités a donc été 64+1514+88=1666. Dans l'étape HMM, nous avons défini des modèles ayant $2i+1$ états émetteurs (i étant la longueur d'unités en cibles de la DT). Nous avons entraîné ces modèles par une initialisation, un apprentissage Baum-Welch sans contexte et par cinq itérations d'apprentissage en contexte. Une itération de segmentation par les HMM et de ré-apprentissage des paramètres a été effectuée.

Pour le décodage, nous avons défini des unités de synthèse équivalentes à celles de codage et nous avons sélectionné 8 représentants pour chacune. La synthèse utilisée a été une simple concaténation des signaux des représentants, sans modification ni du pitch ni du timing. Seul un ajustement de l'énergie moyenne a été encodé par 5 bits par unité. Le débit binaire total ainsi obtenu a été de 205.6 bps sur l'ensemble d'apprentissage et 211 bps sur celui de test. Nous avons évalué la qualité de la parole décodée dans les tests subjectifs : nous avons trouvé les signaux de sortie intelligibles, bien que leur qualité n'ait pas atteint celle de vocodeurs à plusieurs kbps.

Dans le deuxième jeu d'expériences, nous avons utilisé les données de deux locuteurs de la BD américaine Boston University Radio Speech Corpus. Ici, les données sont de qualité "Hi-Fi" (fréquence d'échantillonnage 16 kHz). Le pré-traitement a été similaire aux expériences précédentes, au nombre de coefficients dans les vecteurs cepstraux près (ce nombre a été porté à 16). Par contre, nous n'avons pas utilisé de MG avant la modélisation stochastique, les HMM étant appris directement sur les transcriptions DT+QV. Leur nombre réduit (64) a permis un affinement avec 5 itérations de segmentation et de ré-estimation. Nous avons vérifié que la vraisemblance d'alignement des données avec les modèles augmentait. Nous avons ensuite testé une application des MG sur la dernière segmentation HMM, et nous avons obtenu des dictionnaires de séquences de longueur variable 1 à 6, de tailles 722 (pour le locuteur féminin) et 972 (pour le sujet masculin).

Pour le décodage, nous avons utilisé des unités de synthèse équivalentes à celles de codage, et nous avons disposé de 8 représentants pour chacune. Ici, nous avons testé une synthèse LPC et nous n'avons pas considéré le codage de la prosodie, les contours de F_0 et de l'énergie originaux étant introduits directement dans le synthétiseur. Les débits binaires (seulement pour le codage des unités) obtenus sont donnés dans la Table suivante:

locuteur	F2B		M2B	
débit binaire sur l'ensemble de	apprentissage	test	apprentissage	test
HMM 6-ème génération	126	127	126	130
HMM 6-ème génération+MG	103	110	108	119

En évaluant la qualité de la parole obtenu, nous l'avons jugée intelligible, avec une meilleure qualité pour les multigrammes (moins de distorsions sur les transitions).

B.4.1 Étude de correspondance d'une segmentation ALISP et d'un alignement phonétique

Nous avons comparé la segmentation obtenue sur les données du locuteur féminin avec les alignements phonétiques. Ces alignement ont été obtenus à Boston University et sont joints à la BD. Nous avons mesuré la correspondance par des recouvrements des unités ALISP avec des phonèmes, et avons évalué une matrice de confusion. Celle-ci montre, que la correspondance est consistante, mais pas biunivoque. Ce fait ne facilite pas une éventuelle conception d'un système de reconnaissance basé sur des unités ALISP. Les méthodes de mise en correspondance de séquences d'unités, ou de composition des modèles doivent être étudiées.

B.5 Application No. 2 : Vérification du locuteur indépendante du texte

Dans ce domaine, l'application des unités ALISP peut approcher une vérification indépendante du texte aux méthodes dépendantes du texte, plus précises, car le signal peut être aligné dans les classes plus discriminantes. Cette approche peut être vérifiée avec plusieurs algorithmes de calcul des scores (locuteur client versus "le monde") et de décision. Nous l'avons testé en combinaison avec des perceptrons multi-couche (MLP).

Nous avons effectué les expériences dans le cadre de l'évaluation internationale NIST-NSA'98. Les données d'entraînement contiennent des fichiers de 250 clients masculins et de 250 clients féminins. Quant aux données de test, elles comportent 2500 fichiers par sexe, chacun "prétendant" être une dizaine de clients. Différentes conditions sont définies, en ce qui concerne les durées des signaux; nous avons étudié uniquement le cas avec plus de deux minutes de parole pour l'apprentissage et 30 s pour les tests.

Le signal a été analysé par la méthode LPC-cepstrale. Nous avons appliqué la DT produisant 15 cibles par seconde en moyenne. La QV suivante a déterminé 8 classes de segments. Cette information a été utilisée pour apprendre pour chaque client 8 MLPs discriminant le client et le monde (chacun avec 20 nœuds dans la couche cachée). Dans le calcul du score final, les scores dépendants des classes ont été re-combinés avec les mêmes poids.

L'évaluation des résultats a été faite en termes de courbes DET (detection error tradeoff) donnant le rapport de la probabilité de fausse acceptation en fonction de la probabilité de faux rejet avec le seuil de décision pour paramètre. Nous avons comparé des résultats pour deux catégories des couples entraînement-test: SN (same number), où les signaux proviennent du même No. de téléphone, et DT (different type) où les signaux proviennent non seulement des appareils différents mais aussi des types de microphone différents (électret vs. charbon). Les résultats ont été comparés avec des performances d'un système global, basé sur un seul MLP, et avec un système de référence avec une modélisation des clients par le mélange des Gaussiennes (GMM).

Nous avons montré que l'approche segmentale ne dégradait pas les performances par rapport à une modélisation globale MLP. Pour la catégorie DT (plus difficile), les unités ALISP ont donné de meilleurs résultats. Par contre, nous n'avons pas atteint le pouvoir discriminant des GMM.

B.6 Discussion, conclusions

Cette thèse avait pour but d'étudier les techniques de détermination automatique d'unités de parole et de les tester expérimentalement. Nous avons conçu un algorithme comportant la décomposition temporelle, la quantification vectorielle et les multigrammes. Pour modéliser les unités, nous avons fait appel aux Modèles de Markov Cachés.

L'application dans le codage à très bas débit a montré l'efficacité des techniques utilisées. Dans les deux jeux d'expériences, nous avons obtenu la parole intelligible aux débit moyens de codage des unités de $\sim$120 bps. Ces expériences n'étaient cependant pas sans difficultés : nous avons choisi des méthodes de synthèse très simples, qui devraient être remplacées par des schémas de meilleure qualité (PSOLA, HNM). La détermination des unités de synthèse et un lissage à leurs bornes n'ont pas été entièrement résolus. Pour l'application dans des systèmes réels, l'algorithme proposé doit être complété par une adaptation au locuteur et éventuellement par un module de modification de la voix dans le synthétiseur.

Quant aux travaux effectués en vérification du locuteur, nous avons montré que des unités ALISP constituaient une alternative valable à des systèmes basés sur la reconnaissance vocale

appliquée pour aligner le signal dans les classes. Les expériences ont confirmé que l'application d'une pré-segmentation ALISP ne dégradait pas les performances par rapport à un système avec le calcul global des scores. La difficulté principale se trouve dans l'agrégation des scores des classes pour obtenir le score final. Cette composition devrait être basée sur une mesure des pouvoirs discriminants des classes. D'autres opérations, effectuées classiquement à la fin de la chaîne de traitement (normalisation, détermination des seuils), devraient être décalées vers les calculs des scores par classe.

En plus des problèmes "techniques" mentionnés plus haut, la méthodologie ALISP ouvre des questions plus générales. La plus importante est celle de la détermination objective de qualité des unités. Certes, les résultats du codage à très bas débit, l'évaluation subjective des signaux correspondant aux classes et le mapping ALISP-phonétique donnent quelques informations, mais il s'agit des méthodes indirectes. Le problème suivant est le choix d'un nombre optimal d'unités qui devrait être guidé par l'application cible. Une question plus générale concerne la pertinence des techniques classiquement utilisées en reconnaissance de la parole dans la détermination automatique.

Nos futures recherches vont s'articuler autour de deux axes principaux : à court terme, nous allons améliorer les codeurs à très bas débit avec une éventuelle application industrielle en vue. À long terme, nous allons étudier les possibilités d'applications des unités ALISP dans un système de reconnaissance vocale à très grand vocabulaire.

Appendix C

Souhrn v českém jazyce

C.1 Úvod

Mnohé moderní systémy pro automatické zpracování řeči (ASŘ) jsou založeny na jednotkách typu *slovních segmentů*. V systémech pro rozpoznávání řeči (s nelimitovanou slovní zásobou) tvoří tyto jednotky přechodnou úroveň mezi akustickým (případně parametrickým) popisem signálu a lexikální úrovní, v případě rozpoznávání mluvčího můžeme zlepšit výsledky dosažené "globálními" systémy tak, že řečový signál před-rozdělíme do tříd a vytvoříme několik paralelních rozhodovacích systémů, a konečně v kódování na velmi nízkých bitových rychlostech (very low bit rate, VLBR), kde se již nedá přenášet informace o každém rámci, určují tyto jednotky symbolickou informaci, která je vyslána do přenosového kanálu či uložena.

V klasických přístupech jsou tyto jednotky založeny na historicky známých pojmech: na fonémech, jednotkách odvozených (difóny, fonémy s kontextem, ...), slabikách, nebo jiných. K definici sady takových jednotek a k určení jejich posic v řečové databázi (transkripci) jsou nutné detailní znalosti fonetiky a lingvistiky. Do tvorby anotací či transkripcí musíme navíc investovat mnoho lidského úsilí, a je známo, že právě tyto etapy jsou v celém procesu tvorby databází (DB) nejdražší, nemluvě o jejich náchylnosti k chybám způsobeným lidským faktorem.

C.2 Navržené řešení: ALISP

Tato disertace byla zaměřena na alternativní přístup: *automatické* určování sad jednotek, jakož i transkripcí databází. Techniky, které můžeme označit obecným názvem ALISP (Automatic Language Independent Speech Processing – automatické zpracování řeči nezávislé na jazyku) jsou založeny na *datech* a snaží se využívat apriorních informací v nejnižší možné míře. Tyto metody hledají rovnováhu mezi přesností popisu řeči a úsporností tohoto popisu, detekují pravidelné vzory v signálu či v jeho parametrisaci, a tím odkrývají jeho vnitřní strukturu. Použití těchto jednotek je bezproblémové v oborech, kde je symbolická representace mezistupněm mezi dvěma signály (kódování), nebo kde je pouze pomocná (segmentace v ověřování mluvčího). V případě, že na výstupu očekáváme lexikální popis, je nutné hledat techniky pro zobrazení (mapování) mezi ALISP-jednotkami a fonémy. Klasické výslovnostní slovníky můžeme dokonce nahradit jejich ekvivalenty vytvořenými pomocí těchto automaticky získaných jednotek.

C.3 Použité techniky

Pro danou řečovou databázi je určení jednotek provedeno ve dvou etapách: v první definujeme jejich soubor a hledáme první popis databáze. Ve druhé je modelujeme pomocí statistických

modelů. Na konci těchto dvou etap je systém natrénován a může nadále fungovat s neznámými řečovými signály.

První etapa zahrnuje čtyři kroky: nejprve je třeba pro každý řečový rámec určit vektor parametrů (příznakový vektor). V tomto kroku jsme použili klasickou metodu LPCC (kepstrum z lineární predikce). Pak používáme *časovou dekomposici (TD).* Tato metoda, navržená Atalem a zdokonalená Bimbotem, aproximuje matici parametrů pomocí cílových vektorů a interpolačních funkcí (IF). Tyto funkce, které určují kvazi-stacionární oblasti v signálu, mohou posloužit pro jeho prvotní *segmentaci.* Takto nalezené segmenty pak podstupují shlukování bez učitele, realisované v našem případě *vektorovým kvantováním (VQ).* Aplikujeme li TD a VQ, dostaneme základní transkripci signálu. Nalezené jednotky mohou být dále zpracovávány další metodou, zvanou *multigramy (MG).* Tento algoritmus (známe několik jeho variant – diskrétní, různé definice spojitých multigramů) umožňuje detekovat *charakteristické sekvence* jednotek v trénovací databázi. Aplikujeme-li tuto techniku na automaticky určené jednotky, získáme paralelu klasických přístupů založených na slabikách či difónech.

Ve druhé etapě jsou jednotky určené pomocí kombinace TD+VQ nebo TD+VQ+MG *modelovány* pomocí *skrytých Markovových modelů (HMM).* Tato metoda, široce používaná v rozpoznávání řeči, však nemá za úkol pouze vyprodukovat modely, ale sama se zúčastňuje tvorby sady jednotek: iterace segmentace databáze pomocí HMM a následného přetrénování parametrů HMM slouží ke zlepšování kvality této sady.

Výstupy navržené metody jsou tedy trojího druhu: *slovník jednotek* s jejich apriorními pravděpodobnostmi (pokud jsou žádány, dají se odhadnout na trénovací databázi), *segmentace* trénovací databáze a sada *modelů.*

C.4 První aplikace: kódování na velmi nízkých rychlostech

V kódování na nízkých rychlostech jsme pracovali s technikou *fonetických vokodérů*, popsanou Ribeirem, Trancosovou, Ismailem, Tokudou a jinými. Na rozdíl od těchto autorů, kteří bezvýhradně využívají fonémy, jsme použili automaticky odvozené základní jednotky, které zde budeme nazývat *kódovacími jednotkami.* V příchozí řeči je můžeme detekovat, řeč pomocí nich přepsat, a vyslat jejich indexy do dekodéru. Musíme se též zabývat přenosem dalších údajů, a to o prosodii (základní tón, energie, časování). V dekodéru je řeč rekonstruována pomocí *syntézy*: nejprve musíme ze sledu kódovacích jednotek odvodit informaci o *syntezačních jednotkách.* Pro každou z nich máme pak ve slovníku uložených několik *representantů*, kteří byli získáni v trénovací fázi. Syntezátor je tedy řízen parametrickým popisem representanta a přídavnou informací o prosodii.

Experimentální práce zahrnovala testy závislé na mluvčím, pracovali jsme postupně se dvěma databázemi. V první sadě pokusů jsme využili francouzskou databázi PolyVar vytvořenou v ústavy IDIAP (Martigny, Švýcarsko). Tato databáze je telefonní. Použili jsme data od mluvčího, od něhož obsahovala DB nejvíce hovorů. Těchto bylo 218 a rozdělili jsme je na 174 pro trénování a 44 pro testování. Parametrisaci jsme provedli pomocí 10 LPC-kepstrálních koeficientů ve 20 milisekundových rámcích s 10 ms překryvem. Pro každý hovor jsme vypočetli a odečetli průměrnou hodnotu kepstra (CMS). Pak jsme aplikovali TD, která byla nastavena tak, aby produkovala v průměru 15 cílů za vteřinu. Pomocí získaných segmentů jsme naučili kódovou knihu VQ o 64 kódových vektorech. V další etapě jsme použili multigramy: tyto detekovaly 1514 charakteristických 2-gramů a 88 3-gramů, takže celkový počet jednotek byl 64+1514+88=1666. V etapě "HMM" jsme nejprve definovali modely o $2i + 1$ stavech, kde i je počet původních cílů TD v sekvenci. Tyto modely jsme natrénovali pomocí inicialisace, Baum-Welchovy reestimace bez kontextu a 5-ti iterací učení s kontextem. Provedli jsme jednu iteraci segmentace HMM a přeučení modelů.

Pro dekódování jsme definovali syntezační jednotky ekvivalentní kódovacím a pro každou

jsme vybrali 8 representantů. Syntéza se prováděla pouhým napojováním úseků signálu, takže nebyly možné změny základního tónu ani časování. Kódovali jsme pouze opravnou konstantu pro korekci průměrné energie segmentu, a to 5-ti bity. Celkový dosažený bitový tok byl 205.6 bps na trénovací části databáze a 211 bps na části testovací. Kvalitu dekódované řeči jsme vyhodnocovali v subjektivních poslechových testech: shledali jsme výstupní signály srozumitelné, i když je jejich kvalita nesrovnatelná s kodéry pracujícími na bitových rychlostech řádu několika kbps.

Ve druhé sadě pokusů jsme pracovali se dvěma mluvčími z americké databáze Boston University Radio Speech Corpus. V totmo případě mají data "Hi-Fi" kvalitu (jsou navzorkována na 16 kHz). Předzpracování bylo podobné jako v předcházejícím případě, až na počet kepstrálních koeficientů (zde jich bylo použito 16). Nepoužili jsme však multigramy před statistickým modelováním, takže HMM byly trénovány přímo na transkripci získané pomocí TD+VQ. Jejich nízký počet (64) nám umožnil 5 iterací segmentace a reestimace parametrů. Ověřili jsme, že věrohodnost srovnání modelů s daty během iterací roste. Na závěr jsme testovali aplikaci multigramů na poslední segmentaci pomocí HMM, dostali jsme tak slovník sekvencí o délce 1 až 6, o velikosti 722 (pro mluvčí ženského pohlaví) a 972 (pro muže).

Pro dekódování jsme definovali syntezační jednotky opět ekvivalentní kódovacím a pro každou jsme vybrali 8 representantů. V těchto pokusech jsme testovali syntézu LPC a neuvažovali jsme kódování prosodie. Originální průběhy základního tónu a energie byly zavedeny přímo do syntezátoru. Bitové toky (pouze pro kódování jednotek) uvádíme v následující tabulce:

mluvčí	F2B		M2B	
bitový tok pro část	trénovací	testovací	trénovací	testovací
6. generace HMM	126	127	126	130
6. generace HMM+MG	103	110	108	119

Dekódovaná řeč byla hodnocena jako srozumitelná, kvalitnější byla při použití multigramů (projevuje se zde méně zkreslení na přechodech jednotek).

C.4.1 Srovnání segmentace ALISP s fonetickou transkripcí

Segmentaci pomocí automaticky odvozených jednotek jsme srovnali s fonetickou transkripcí pro ženskou mluvčí z BU databáze. Fonetické transkripce byly vytvořeny na BU a jsou obsaženy v databázi. Vzájemné vztahy ALISP jednotek a fonémů jsme studovali pomocí jejich relativních překrytí, z nich jsme vyhodnotili konfusní matici. Tato ukazuje, že zobrazení obou druhů jednotek je celkově konsistentní, ale není jednojednoznačné. Je tedy třeba zaměřit se na metody mapování sekvencí jednotek, nebo na komposici jednotlivých modelů.

C.5 Druhá aplikace: Ověřování mluvčího nezávislé na textu

V této oblasti může aplikace ALISP jednotek přiblížit ověřování nezávislé na textu k metodám závislým na textu, které mohou díky přiřazení úseků signálu do tříd dosahovat lepších výsledků. Tato metoda může být použita ve spojení s různými systémy pro výpočet skóre (klient versus "svět") a pro rozhodování. My jsme ji testovali v kombinaci s mnohavrstvými perceptrony (MLP).

Experimentální práci jsme prováděli během ověřovací kampoaně NIST-NSA'98. Trénovací data obsahovala 250 souborů pro mužské klienty a 250 souborů pro ženské klienty. Testovací data obsahovala 2500 souborů pro každé pohlaví a každý z těchto souborů bylo třeba vyzkoušet "jako" 10 různých klientů. Na základě množství dat dostupných pro trénování a testování jsou definovány různé trénovací-testovací podmínky. My jsme studovali pouze případ, kdy byly k disposici více než 2 minuty řeči pro trénování a 30 s pro test.

Vyhodnocení výsledků jsme provedli na křivkách DET (detection error tradeoff), které zobrazují pravděpodobnost mylných odmítnutí jako funkci pravděpodobnosti mylných přijetí, s rozhodovacím prahem jako parametrem. Porovnávali jsme výsledky pro dvě kategorie párů trénovací-testovací soubor: SN (same number), kde tyto soubory pocházely ze stejného telefonního čísla, a DT (different type), kde byl rozdílný nejen telefonní aparát, ale dokonce i typ mikrofonu (elektretový nebo uhlíkový). Výsledky byly porovnány s globálním systémem, založeným na MLP, a s referenčním ověřovačem, kde se klienti modelují směsí Gaussovských rozložení (GMM).

Ukázali jsme, že navžený segmentální přístup nezhoršuje chybovost systému systému oproti globálnímu modelu. Pro "těžší" kategorii DT jsme dokonce s ALISP jednotkami dosáhli lepších výsledků. Nepodařilo se však překonat diskriminační schopnosti GMM.

C.6 Diskuse, závěry

Tato disertace si kladla za cíl prostudovat automatickou tvorbu řečových jednotek a experimentálně ji otestovat. Vytvořili jsme algoritmus obsahující časovou dekomposici, vektorové kvantování a multigramy. Pro modelování jednotek jsme použili skryté Markovovy modely.

Aplikace v kódování řeči na velmi nízkých rychlostech prokázala efektivitu použitých technik. V obou sadách experimentů jsme obdrželi po dekódování srozumitelnou řeč, a to při bitových tocích pro kódování jednotek okolo 120 bps. Tyto experimenty se samozřejmě neobešly bez obtíží: použili jsme velmi jednoduchou syntézu, která musí být v budoucnu nahrazena kvalitnější metodou (PSOLA nebo HNM). Odvozování syntezačních jednotek a vyhlazování na jejich přechodech též nebylo uspokojivě vyřešeno. Pro aplikaci v reálných systémech musí být uvedený algoritmus doplněn adaptací na mluvčího a eventuálně modulem pro modifikaci hlasu v syntezátoru.

Co se týče aktivit v ověřování mluvčího, ukázali jsme, že jednotky ALISP jsou hodnotnou alternativou k rozpoznávání řeči, jež se používá k segmentaci a zatřiďování segmentů. Pokusy potvrdily, že ALISP segmentace nezhoršuje chybovost systému vzhledem ke globálnímu výpočtu skóre. Hlavní problém našeho přístupu spočívá v rekombinaci skóre jednotlivých tříd při tvorbě "globálního" skóre. Tato operace by měla být založena na diskriminačních schopnostech jednotlivých tříd. Také další operace, které se v klasických systémech provádějí na konci zpracování (normalisace, výpočet rozhodovacího prahu) musí být "přiblíženy" výpočtům skóre v jednotlivých třídách.

Kromě těchto spíše "technických" problémů otevírá metodologie ALISP také obecnější otázky. Z nich nejdůležitější je objektivní určování kvality odvozených jednotek. Povšimli jsme si, že k tomuto určení lze použít kódování, subjektivní posouzení signálů odpovídajících jednotlivým jednotkám, či mapování s fonémy, ale ve všech případech se jedná o nepřímé metody. Dalším problémem je určení optimálního počtu jednotek, který by měl odpovídat cílové aplikaci. Ještě obecnější otázkou je vhodnost technik, které se standardně používají v rozpoznávání, pro automatické určování jednotek.

Naše budoucí výzkumy se budou ubírat dvěma hlavními směry: v krátkodobé perspektivě bychom chtěli vylepšit vlastnosti kodéru na velmi nízkých rychlostech tak, aby případně nalezl průmyslové uplatnění. V dlouhodobé perspektivě chceme studovat možné aplikace ALISP jednotek v rozpoznávání řeči s velkou slovní zásobou.

Appendix D

Selected publications

- **ICASSP 97** [98]: J. Černocký, G. Baudoin, and G. Chollet. Speech spectrum representation and coding using multigrams with distance. In *Proc. IEEE ICASSP 97*, pages 1343–1346, Munich, Germany, April 1997.
- **ICASSP 98** [100]: J. Černocký, G. Baudoin, and G. Chollet. Segmental vocoder - going beyond the phonetic approach. In *Proc. IEEE ICASSP 98*, pages 605–608, Seattle, WA, May 1998. http://www.fee.vutbr.cz/~cernocky/Icassp98.html.
- **TSD 98** [102]: J. Černocký, G. Baudoin, D. Petrovska-Delacretaz, J. Hennebert, and G. Chollet. Automatically derived speech units: applications to very low rate coding and speaker verification. In P. Sojka, V. Matoušek, K. Pala, and I. Kopeček, editors, *Proc. of Workshop on Text Speech and Dialog (TSD'98)*, pages 183–188, Brno, Czech Republic, September 1998.
- **ICSLP 98** [72]: D. Petrovska-Delacrétaz, J. Černocký, J. Hennebert, and G. Chollet. Text-independent speaker verification using automatically labelled acoustic segments. In *accepted to International Conference on Spoken Language Processing (ICSLP)*, Sydney, Australia, December 1998.

Bibliography

[1] European Language Resources Association. http://www.icp.grenet.fr/ELRA/home.html.

[2] Internet telephony resource list. `http://anansi.panix.com/userdirs/charl/telephony.html`.

[3] Linguistic Data Consortium homepage. http://www.ldc.upenn.edu/.

[4] SAMPA – computer readable phonetic alphabet. homepage. http://www.phon.ucl.ac.uk/home/sampa/home.htm.

[5] B. S. Atal. Efficient coding of LPC parameters by temporal decomposition. In *Proc. IEEE ICASSP 83*, pages 81–84, 1983.

[6] M. Bacchiani, M. Ostendorf, Y. Sagisaka, and K. Paliwal. Design of a speech recognition system based on acoustically derived segmental units. In *Proc. IEEE ICASSP 96*, pages 443–446, 1996.

[7] L.R. Bahl, J.R. Bellagarda, P.V. de Souza, P.S. Gopalakrishnan, D. Nahamo, and M.A. Picheny. A new class of fenonic Markov word models for large vocabulary continuous speech recognition. In *Proc. IEEE ICASSP 91*, pages 117–180, 1991.

[8] G. Baudoin, J. Černocký, and G. Chollet. Quantification de séquences spectrales de longueur variable pour le codage de la parole à très bas débit (Quantization of variable length spectral sequences for very low rate speech coding). In *16e colloque GRETSI*, pages 1093–1096, Grenoble, France, September 1997.

[9] G. Baudoin, J. Černocký, and G. Chollet. Quantization of spectral sequences using variable length spectral segments for speech coding at very low bit rate. In *Proc. EUROSPEECH 97*, pages 1295–1298, Rhodes, Greece, September 1997.

[10] S.W. Beet and L. Baghai-Ravary. Automatic segmentation: data-driven units of speech. In *Proc. Eurospeech 97*, pages 505–508, 1997.

[11] F. Bimbot. An evaluation of temporal decomposition. Technical report, Acoustic research department AT&T Bell Labs, 1990.

[12] F. Bimbot, G. Chollet, P. Deleglise, and C. Montacié. Temporal decomposition and acoustic-phonetic decoding of speech. In *Proc. IEEE ICASSP 88*, pages 445–448, New York, 1988.

[13] F. Bimbot, R. Pieraccini, E. Levin, and B. Atal. Modèles de séquences à horizon variable: Multigrams. In *Proc. XX-èmes Journées d'Etude sur la Parole*, pages 467–472, Trégastel, France, June 1994.

[14] F. Bimbot, R. Pieraccini, E. Levin, and B. Atal. Variable length sequence modeling: Multigrams. *IEEE Signal Processing Letters*, 2(6):111–113, June 1995.

[15] A. Bonafonte and J.B. Mariño. Language modeling using x-grams. In *Proc. ICSLP 96*, pages 394–397, 1996.

[16] A. Bonafonte, A. Nogueiras, and A. Rodriguez-Garrido. Explicit segmentation of speech using Gaussian models. In *Proc. ICSLP 96*, pages 1269–1272, 1996.

[17] H. Bourlard and C. Wellekens. Links between Markov models and multilayer perceptrons. In D. Touretzky, editor, *Advances in Neural Information Processing*, volume 1, pages 502–510, San Mateo CA, 1988. Moran Kaufmann.

[18] S. Bruhn. Matrix product quantization for very low rate speech coding. In *Proc. IEEE ICASSP 95*, pages 724–727, Detroit, 1995.

[19] G. Chollet, A. Constantinescu, and J. Černocký. Automatic language independent speech processing (ALISP): Some tools for acoustic-phonetic research. to appear in Proc. International Society of Phonetic Sciences (IPS) Conference, June 1998.

[20] G. Chollet, J. Černocký, A. Constantinescu, S. Deligne, and F. Bimbot. *NATO ASI: Computational models of speech pattern processing*, chapter Towards ALISP: a proposal for Automatic Language Independent Speech Processing. Springer Verlag, in press.

[21] P. A. Chou and T. Lookabaugh. Conditional entropy-constrained vector quantization of linear predictive coefficients. In *Proc. IEEE ICASSP 90*, pages 197–200, Albuquerque, 1990.

[22] P. A. Chou and T. Lookabaugh. Locally optimal variable-to-variable length source coding with respect to fidelity criterion. In *Proc. IEEE Int. symposium on information theory*, page 238, 1991.

[23] P. A. Chou and T. Lookabaugh. Variable dimension vector quantization of linear predictive coefficients of speech. In *Proc. IEEE ICASSP 94*, pages I–505–508, Adelaide, June 1994.

[24] P. A. Chou, T. Lookabaugh, and R.M. Gray. Entropy-constrained vector quantization. *IEEE Trans. Acoust., Speech, Signal Processing*, 37:31–42, January 1989.

[25] K. Choukri. *Quelques approches pour l'adaptation aux locuteurs en reconnaissance automatique de la parole*. PhD thesis, École nationale supérieure des télécommunications (ENST), Paris, November 1987.

[26] A. Constantinescu and G. Chollet. Swiss PolyPhone and PolyVar: building databases for speech recognition and speaker verification. In *Speech and image understanding, Proc. of 3rd Slovenian-German and 2nd SDRV Workshop*, pages 27–36, Ljubljana, Slovenia, 1996.

[27] M. Copperi. Rule-based speech analysis and its application to CELP coding. In *Proc. IEEE ICASSP 88*, pages 143–146, 1988.

[28] S.K. Das and M.A. Picheny. Issues in practical large vocabulary isolated word recognition: the IBM Tangora system. In *Automatic Speech and Speaker Recognition: Advanced Topics*, chapter 19. Kluwer Academic Publishers, 1996.

[29] S. Deligne. *Modèles de séquences de longueurs variables: Application au traitement du langage écrit et de la parole*. PhD thesis, École nationale supérieure des télécommunications (ENST), Paris, 1996.

[30] S. Deligne and F. Bimbot. Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams. In *Proc. IEEE ICASSP 95*, pages 169–172, Detroit, USA, 1995.

[31] S. Deligne and F. Bimbot. Inference of variable-length acoustic units for continuous speech recognition. In *Proc. IEEE ICASSP 97*, pages 1731–1734, Munich, Germany, 1997.

[32] S. Deligne and F. Bimbot. Inference of variable-length linguistic and acoustic units by multigrams. *Free Speech Journal*, 1(4), June 1997. http://www.cse.ogi.edu/CSLU/fsj/.

[33] S. Deligne, F. Yvon, and F. Bimbot. Variable-length sequence matching for phonetic transcription using joint multigrams. In *Proc. EUROSPEECH 95*, pages 2243–2246, Madrid, Spain, 1995.

[34] G. Doddington. Speaker recognition evaluation methodology – an overview and perspective. In *Speaker Recognition and its Commercial and Forensic Apllications (RLA2C)*, pages 60–66, Avignon, France, 1998.

[35] J. P. Eatock and J. S. Mason. A quantitative assessment of the relative speaker discriminant properties of phonemes. In *ICASSP*, volume 1, pages 133–136, 1994.

[36] G. Flammia, P. Dalsgaard, O. Andersen, and B. Lindberg. Segment based variable frame rate speech analysis and recognition using spectral variation function. In *Proc. ICSLP 92*, pages 983–986, 1992.

[37] T. Fukada, M. Bacchiani, K. Paliwal, and Y. Sagisaka. Speech recognition based on acoustically derived segment units. In *Proc. ICSLP 96*, pages 1077–1080, 1996.

[38] S. Furui. Recent advances in robust speech recognition. In *Proc. ESCA-NATO workshop on robust speech recognition for unknown communication channels*, Pont-à-Mousson, France, April 1997.

[39] J.S. Garofolo, L.F.Lamel, W.M. Fisher, J.G. Fiscus, D.S. Pallett, and N.L. Dahlgren. DARPA–TIMIT acoustic–phonetic speech corpus. Technical Report NISTIR 4930, U.S. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory, February 1993.

[40] A. Gersho. *Vector quantization and signal compression.* Kluwer Academic Publishers, 1996.

[41] S. Ghaemmaghami and M. Deriche. A new approach to very low rate speech coding using temporal decomposition. In *Proc. IEEE ICASSP 96*, pages 224–227, Atlanta, 1996.

[42] S. Ghaemmaghami, M. Deriche, and B. Boashash. Comparative study of different parameters for temporal decomposition based speech coding. In *Proc. IEEE ICASSP 97*, pages III–1703–1706, Munich, 1997.

[43] S. Ghaemmaghami, M. Deriche, and B. Boashash. On modeling event functions in temporal decomposition based speech coding. In *Proc. ESCA Eurospeech 97*, pages 1299–1302, Rhodes, 1997.

[44] O. Ghitza. Auditory models and human performance in tasks related to speech coding and speech recognition. *IEEE Trans. Speech and Audio Processing*, 2(1, part II):115–132, January 1994.

[45] A. Grigoriu, J.P. Vonwiller, and R.W. King. An automatic intonation tone labelling and classification algorithm. In *Proc. IEEE ICASSP 94*, pages II–181–184, 1994.

[46] J. Hennebert and D. Petrovska-Delacrétaz. Phoneme based text-prompted speaker verification with multi-layer perceptrons. In *Proc. RLA2C 98*, Avignon, France, April 1998.

[47] H. Hermansky. Data-driven speech analysis for ASR. In P. Sojka, V. Matoušek, K. Pala, and I. Kopeček, editors, *Proc. of Workshop on Text Speech and Dialog (TSD'98)*, pages 213–218, Brno, Czech Republic, September 1998.

[48] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of Neural Computation*. Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley, 1991.

[49] J.N. Holmes, I.G. Mattingly, and J.N. Shearme. Speech synthesis by rule. *Language and speech*, 7:127–143, 1964.

[50] X. Huang, A. Acero, J. Adcock, H.W. Hon, J. Goldsmith, J. Liu, and M. Plumpe. WHISTLER: a trainable text-to-speech system. In *Proc. ICSLP 96*, pages 2387–2390, 1996.

[51] M. Ismail and K. Ponting. Between recognition and synthesis – 300 bits/second speech coding. In *Proc. EUROSPEECH 97*, pages 441–444, Rhodes, Greece, 1997.

[52] P. Jardin. Reconnaissance de la parole. polycopié I4, École Supérieure d'Ingénieurs en Électrotechnique et Électronique (ESIEE), Noisy-le-Grand, France, 1995.

[53] N. S. Jayant and P. Noll. *Digital coding of waveforms*. Prentice Hall, New Jersey, 1984.

[54] P. Jeanrenaud and P. Peterson. Segment vocoder based on reconstruction with natural segments. In *Proc. IEEE ICASSP 91*, pages 605–608, 1991.

[55] D. P. Kemp, J. S. Collura, and T. E. Tremain. Multi-frame coding of LPC parameters at 600-800 bps. In *Proc. IEEE ICASSP 91*, pages 609–612, Toronto, 1991.

[56] H. P. Knagenhjelm and W. B. Kleijn. Spectral dynamics is more important than spectral distortion. In *Proc. IEEE ICASSP 95*, pages 732–735, Detroit, 1995.

[57] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, (9):171–185, 1995.

[58] E. López-Gonzalo, J.M. Rodríguez-García, L. Hernández-Gómez, and J.M. Villar. Automatic prosodic modeling for speaker and task adaptation in text-to-speech. In *Proc. IEEE ICASSP 97*, pages 927–930, 1997.

[59] J. M. López-Soler and N. Farvardin. A combined quantization-interpolation scheme for very low bit rate coding of speech LSP parameters. In *Proc. IEEE ICASSP 93*, pages II–21–24, Minneapolis, 1993.

[60] J. Makhoul, S. Roucos, and H. Gish. Vector quantization in speech coding. *Proc. IEEE*, 73(11):1551–1589, November 1985.

[61] A. Martin. The 1998 speaker recognition evaluation plan. Technical report, NIST, March 1998. http://www.jaguar.nist.gov/evaluations/.

[62] B. Mouy, P. de La Noue, and G. Goudezeune. NATO STANAG 4479: A standard for an 800 bps vocoder and channel coding in HF-ECCM system. In *Proc. ICASSP 95*, pages I–480–483, Detroit, 1995.

[63] M. Sharma nad R. Mammone. "Blind" speech segmentation: automatic segmentation of speech without linguistic knowledge. In *Proc. ICSLP 96*, pages 1237–1240, 1996.

[64] M. Nakai, H. Singer, Y. Sagisaka, and H. Shimodara. Automatic prosodic segmentation by f_0 clustering using superpositional modeling. In *Proc. IEEE ICASSP 95*, pages 624–627, 1995.

[65] H. Ney. Language modeling for large vocabulary continuous speech recognition. Slides of seminar given at ENST Paris on 10th March 1998.

[66] H. Ney and S. Martin. Maximum likelihood criterion in language modeling. In *Informal Proc. NATO Advanced Study Institute: Computational models of speech pattern processing*, St. Helier, Jersey, UK, July 1997.

[67] F. Nolan. *The Phonetic Bases of Speaker Recognition*. Cambridge University Press, 1983.

[68] J. Olsen. A two-stage procedure for phone based speaker verification. In G. Borgefors J. Bigün, G. Chollet, editor, *First International Conference on Audio and Video Based Biometric Person Authentication (AVBPA)*, pages 219–226, Crans, Switzerland, 1997. Springer Verlag: Lecture Notes in computer Science 1206.

[69] M. Ostendorf, V.V. Digalakis, and O.A. Kimball. From HMM's to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Trans. Speech and Audio Processing*, 4(5):360–378, September 1996.

[70] M. Ostendorf, P.J. Price, and S. Shattuck-Hufnagel. The Boston University radio news corpus. Technical report, Boston University, February 1995.

[71] D. Petrovska-Delacrétaz and J. Hennebert. Text-prompted speaker verification experiments with phoneme specific MLP's. In *Proc. IEEE ICASSP98*, pages II–777–780, Seattle, WA, USA, May 1998.

[72] D. Petrovska-Delacrétaz, J. Černocký, J. Hennebert, and G. Chollet. Text-independent speaker verification using automatically labelled acoustic segments. In *accepted to International Conference on Spoken Language Processing (ICSLP)*, Sydney, Australia, December 1998.

[73] J. Picone and G. R. Doddington. A phonetic vocoder. In *Proc. IEEE ICASSP 89*, pages 580–583, Glasgow, 1989.

[74] J. Psutka. *Komunikace s počítačem mluvenou řečí*. Academia, Praha, 1995.

[75] L. Rabiner and B.H. Juang. *Fundamentals of speech recognition*. Signal Processing. Prentice Hall, Engelwood Cliffs, NJ, 1993.

[76] L. R. Rabiner and L. W. Schaeffer. *Digital processing of speech signals*. Prentice Hall, 1978.

[77] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, February 1989.

[78] C.M. Ribeiro and I.M. Trancoso. Application of speaker modification techniques to phonetic vocoding. In *Proc. ICSLP 96*, pages 306–309, Philadelphia, 1996.

[79] C.M. Ribeiro and I.M. Trancoso. Phonetic vocoding with speaker adaptation. In *Proc. EUROSPEECH 97*, pages 1291–1294, Rhodes, Greece, 1997.

[80] J. Rosenberg and H. Schulzrinn. An RTP payload format for generic forward error correction. Internet draft, Internet Engineering Task Force (IETF), Audio Video Transport Workgroup, July 1998. http://www.ietf.cnri.reston.va.us/.

[81] S. Roucos, J. Makhoul, and R. Schwartz. A variable-order Markov chain for coding of speech spectra. In *Proc. IEEE ICASSP 82*, pages 582–585, Paris, 1982.

[82] S. Roucos, R. Schwarz, and J. Makhoul. Segment quantization for very-low-rate speech coding. In *Proc. IEEE ICASSP 82*, pages 1565–1568, Paris, 1982.

[83] S. Roucos, R. Schwarz, and J. Makhoul. A segment vocoder at 150 b/s. In *Proc. IEEE ICASSP 83*, pages 61–64, Boston, 1983.

[84] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Exploration in the Microstructure of Cognition*, volume 1. MIT Press, 1986.

[85] Y. Sagisaka. Speech synthesis by rule using an optimal selection of non-uniform synthesis units. In *Proc. IEEE ICASSP 88*, pages 679–682, 1988.

[86] Y. Shiraki and M. Honda. LPC speech coding based on variable length segment quantization. *IEEE Trans. Acoust., Speech, Signal Processing*, 36(9):1437–1444, September 1988.

[87] A. S. Spanias. Speech coding: A tutorial review. *Proc. IEEE*, 82(10):1541–1582, October 1994.

[88] I. Stylianou. *Modèles harmoniques plus bruit combinés avec des méthodes statistiques, pour la modification de la parole et du locuteur*. PhD thesis, École nationale supérieure des télécommunications (ENST), Paris, January 1996.

[89] T. Svendsen. On the automatic segmentation of speech signals. In *Proc. IEEE ICASSP 87*, Dallas, 1987.

[90] T. Svendsen. Segmental quantization of speech spectral information. In *Proc. IEEE ICASSP 94*, pages I–517–520, Adelaide, 1994.

[91] K. Tokuda, T. Masuko, J. Hiroi, T. Kobayashi, and T. Kitamara. A very low bit rate speech coder using HMM-based speech recognition/synthesis techniques. In *Proc. IEEE ICASSP 98*, pages 609–612, Seattle, WA, May 1998.

[92] T.E. Tremain. The government standard linear predictive coding algorithm: LPC-10. *Speech Technology Magazine*, pages 40–49, April 1982.

[93] A. M. L. van Dijk-Kappers. Comparison of parameter sets for temporal decomposition. *Speech Communication*, 8(3):204–220, 1989.

[94] J. Černocký. Codeurs de parole segmentaux, rapport du stage de DEA. Technical report, ESIEE Paris, September 1995.

[95] J. Černocký. Recherche des séquences caractéristiques de vecteurs spectraux, application au codage. In *Proc. Rencontres jeunes chercheurs en parole*, pages 40–42, ENST Paris, France, November 1995.

[96] J. Černocký and G. Baudoin. Représentation du spectre de parole par les multigrammes. In *Proc. XXI-es Journées d'Étude sur la Parole*, pages 239–242, Avignon, France, June 1996.

[97] J. Černocký, G. Baudoin, and G. Chollet. Efficient method of speech spectrum description using multigrams. In *Proc. International association of pattern recognition (IAPR) Workshop*, pages 139–148, University of Ljubljana, Slovenia, April 1996.

[98] J. Černocký, G. Baudoin, and G. Chollet. Speech spectrum representation and coding using multigrams with distance. In *Proc. IEEE ICASSP 97*, pages 1343–1346, Munich, Germany, April 1997.

[99] J. Černocký, G. Baudoin, and G. Chollet. Towards a very low bit rate segmental speech coder. In *Informal Proc. NATO Advanced Study Institute: Computational models of speech pattern processing*, St. Helier, Jersey, UK, July 1997.

[100] J. Černocký, G. Baudoin, and G. Chollet. Segmental vocoder - going beyond the phonetic approach. In *Proc. IEEE ICASSP 98*, pages 605–608, Seattle, WA, May 1998. http://www.fee.vutbr.cz/~cernocky/Icassp98.html.

[101] J. Černocký, G. Baudoin, and G. Chollet. Very low bit rate segmental speech coding using automatically derived units. In *Proc. Radioelektronika'98*, pages 224–227, Brno, Czech Republic, April 1998.

[102] J. Černocký, G. Baudoin, D. Petrovska-Delacrétaz, J. Hennebert, and G. Chollet. Automatically derived speech units: applications to very low rate coding and speaker verification. In P. Sojka, V. Matoušek, K. Pala, and I. Kopeček, editors, *Proc. of Workshop on Text Speech and Dialogue (TSD'98)*, pages 183–188, Brno, Czech Republic, September 1998.

[103] J.P. Verhasselt and J.P. Martens. A fast and reliable rate of speech detector. In *Proc. ICSLP 96*, pages 2258–2261, 1996.

[104] D.Y. Wong, B.H. Juang, and D.Y. Cheng. Very low data rate speech compression using LPC vector and matrix quantization. In *Proc. IEEE ICASSP 83*, pages I–65–68, April 1983.

[105] S. Young. *HTK: hidden Markov model toolkit V1.4, Reference manual*. Cambridge University Engineering Department, Speech Group, Cambridge, UK, September 1992.

[106] S. Young, J. Jansen, J. Odell, D. Ollason, and P. Woodland. *The HTK book*. Entropics Cambridge Research Lab., Cambridge, UK, 1996.

[107] S.J. Young, N.H. Russell, and J.H.S. Thornton. Token passing: a simple conceptual model for connected speech recognition systems. Technical Report CUED/F-INFENG/TR.38, Cambridge University Engineering Department, July 1989.