Short-term Time Series in Automatic Speech Processing

Radu Timofte

MASTER'S THESIS

University of Joensuu
Department of Computer Science and Statistics
P.O. Box 111, FIN-80101 Joensuu, Finland

October 31, 2007

Table of Contents

List of Figures

List of Tables

Abstract

Speech recognition along with speaker recognition and voice activity detection are fundamental tasks in speech processing. Since 1970s the performance of speech recognition engines made significant progress mainly because of the stochastic approach followed in acoustic modeling. The continuous, spontaneous speech recognition lacks an acceptable solution, noisy environments increasing the difficulty of the task. The actual systems take advantage of the statistical models of the target language and of the utterance context to reach a good performance.

This thesis tackles: keyword spotting, speaker identification and voice activity detection by introducing the concept of *short-term time series* (STS). STS models sequences of feature vectors extracted from the speech signal that describe the phonemes or pseudo-phonemes as acoustic units. It differs from the traditional frame-based processing in the sense that contextual information is modeled along a short interval, a pseudo-phoneme.

We give a short overview of speech processing basics and make a review of acoustic models along with the proposed short-term time series approach and the tools that are used. In the second part of the thesis we apply the short-term time series for solving keyword spotting, speaker recognition and voice activity detection as distinct problems in speech processing field.

Keyword spotting section sets the background of the field and uses the short-term time series as phoneme modeling technique. The words are considered in their phonetic transcription form and the search of those becomes matching of a pattern of phonemes on the speech material. The system created is a speaker and context independent word spotting system for continuous speech. Extensive tests are done using TIMIT database with good results.

Voice activity detection section offers an overview of the field, sets the comparison methods and introduces short-term time series as modeling technique for speech and non-speech segments. The tests are done on three datasets from different environments and the results are good, showing a robust behavior of the proposed method.

Speaker identification section studies the applicability of short-term time series in speaker modeling. We focus on text-independent closed set speaker identification task and make use of the TIMIT corpus structure for experiments. The results are worse than the baseline represented by a typical *vector quantization* (VQ) – based speaker identification system, showing that the approach followed here is not good.

The results obtained for short-term time series in the mentioned three speech processing tasks indicate the potential of the proposed approach and are a strong basement for further research.

Keywords: short-term time series, pseudo-phonemes, keyword spotting, speaker identification, voice activity detection

1. INTRODUCTION

Let us consider one example from daily life. You receive a phone call and the first thing to hear is “Hello, how are you?” and, after a while, you answer with “Well, fine!”. What happens here? First of all you distinguish that you hear a voice and not a noise, an environmental sound. Secondly, you are able to get the meaning of the speech, to receive the message, which happens because you know the language. Thirdly, you identify the person who is speaking to you. The human beings are doing speech processing all the time; it is a critical process in communication. This chapter describes the motivation of the thesis, the field of automatic speech processing, its applications, the main contributions and the structure of the thesis.

1.1 Motivation

For centuries scientists have dreamt to give machines the capabilities of speaking and understanding natural language. The first invention of a phonographic recording device dates to 1857 and was made by Patrick Kiernan. In 1930s, Homer Dudley at Bell Laboratories proposed a system model for speech analysis and synthesis [Dudley1939a, Dudley1939b], and since then, the speech processing problem has been treated progressively from low complexity machines that deal with a reduced set of sounds to complex systems that respond to fluently spoken natural language.

There is an increasing need for speech processing in the world of information. All audio materials need tools that permit to efficiently deal with the embedded informational content. In telecommunications, it is important to send only the speech part of the signal, to save bandwidth and to increase the speed of transmission. Thus, voice activity detection is crucial in labeling the signal as speech (information) and non-speech (dispensable data). For doing information retrieval from large audio materials it is necessary to have a text transcription obtained by speech recognition tools or to perform a search of particular keywords at a high-speed rate directly. Nevertheless, person authentication is a sensible issue and can be done based on speech as biometric. Speaker recognition involves recognizing persons by their voices.

1.2 Automatic speech processing

Speech processing refers to analysis and processing of speech signals with the aim to achieve maximum benefit in various practical scenarios like information retrieval or speaker recognition. Since the signals are usually processed in digital form, speech processing can be placed at the confluence of digital signal processing and natural language processing. Automation of speech processing is of great importance in practical applications where the human factor is involved only in the final stage of interpreting and using the processing results. In this way we save human power resources.

The speech processing covers a broad area that relates to the following important research directions:

- *speech recognition*: analysis of the linguistic content of the speech signal.
- *speaker recognition*: checks the identity of the speaker from speech signal.
- *speech signal enhancement*: focus on quality aspects of the speech signal.
- *speech coding*: specialized form of data compression.
- *voice analysis* with medical purposes, such as analysis of vocal loading and dysfunction of vocal cords.
- *speech synthesis*: artificial synthesis of speech.

Speech recognition

Speech recognition [Rabiner1993] is defined as the process of converting the speech signal to its linguistic content in form of digital data (usually plain text). In other words, the speech recognition implies the ability to match patterns of provided or acquired vocabularies against the speech signal in an appropriate form.

Speaker recognition

Identification of a speaker from a known collection of speakers based on an audio recording and the verification of a claimed identity makes the object of speaker recognition [Campbell1997]. In those processes it is usually not necessary to identify what the speaker says, but to extract the information from their voice that varies between speakers.

Speech enhancement

Usually there appear various different noises with the speech content on an audio recording. The speech could also be too low to be easily understood by people. The processes that involve noise reduction, removing effects due to recording equipments and enhancing the perceptual quality of the speech signal are grouped under speech enhancement [Ephraim1992] direction of speech processing.

Speech coding

Speech coding [Furui2000] is the application of data compression to digital audio signals containing speech. In order to achieve good performance (high rates of compression), the speech coding process utilizes sound related properties, speech models and intelligibility of the message. A lot of the initial speech signal is lost and the aim is that the decoded/reconstructed signal is still understandable.

Voice analysis

Voice analysis tries to use voice characteristics as loading and dysfunctions [Jilek2004] in vocal cords in order to find different possible health problems.

Speech synthesis

Speech synthesis [Furui2000] can be seen as the opposite direction to speech recognition; it has the aim to artificially create the speech signal that carries out the desired message (text). To

accomplish this task speech synthesizers use information about basic speech units, models of the language addressed and grammar rules.

1.3 Applications

Here we will discuss about practical applications related to the speech processing tasks addressed in this thesis: keyword spotting, voice activity detection and speaker recognition.

Keyword spotting

Keyword spotting (KWS) [Foote1999] as a part of automatic speech recognition aims at locating given keywords in audio recordings. The goal of a keyword spotting system is to provide the capability of searching through audio content based on queries, without necessarily to compute the full text transcription. The query can be formed from a text representation, a phonetic transcription or a speech sample. The search will return the positions of possible occurrences and the corresponding confidence scores. This topic is very important since the human factor is prohibitive to be used in dealing with large audio recordings to classify and identify the speech content that occurs. Applicability of KWS address:

- *person authorization* uses KWS to find out the password said by the claimer and eventually some other information used in the authorization/authentication process.
- *information retrieval* uses KWS to localize in a multimedia content information related to particular queries.
- *human-computer interaction* facilitates the interaction through basic words between the computer and the user in order to accomplish different tasks, like dialogue-based ticketing machine.
- *voice-command control*: KWS identifies the words that permit a user to control different devices using his voice.

Voice activity detection

Voice activity detection (VAD) [Davis2006] aims at classifying a given audio segment as speech or non-speech. In this way the focus of other speech processing tasks can be on the relevant segments and to make use of this information for updating the models for noise and speech. VAD is a critical front-end component in various voice-based applications:

- *telecommunications* uses VAD in speech coding process.
- *forensics*: VAD helps by pruning irrelevant audio material from the material containing speech.
- *speech recognition* moves focus only on the speech segments and reduce the recognition errors.
- *speaker recognition* by using only the audio material with speech for the speaker models and in this way bringing substantial improvements in the recognition.
- *speech enhancement* by using the classification of speech and non-speech segments, the models for noise and speech can be updated properly and the focus in enhancement can be targeted at the speech segments.

Speaker recognition

Speaker recognition (SR) [Campbell1997] technology includes the following applications:

- *person authentication* uses SR as a biometric authentication method. Examples are credit card transactions, computer login, access to physical facilities and border control.
- *forensics* uses SR as a tool for recognition and tracking of speakers in audio recordings.
- *speech recognition* uses speaker adaptation for speech models in order to use the current speaker information, and thus, to improve the performance of the recognition.
- *multi-speaker environments*: tracking speakers in audio recordings or teleconferences that involves several speakers.
- *speech user interfaces* uses SR to adapt the interfaces to the preferences/profile of the current user/speaker.

1.4 Main contributions and the outline of the Thesis

The goal of the thesis is to introduce an acoustic model based on *short-term time series* and to study how it can be used for solving speech processing tasks. First, we describe the tools needed for using the model. Once the fundamental concepts are defined we apply the model and analyze the results in keyword spotting, voice activity detection and speaker recognition tasks. The originality of thesis comes from the novel approach of using short-term time series as acoustic models and the proposed solutions for speech processing tasks.

The rest of the thesis is organized as follows. Chapter 2 gives the bases for speech processing introducing the speech production, acoustics and perception, the phonology and the feature extraction process for characterizing a speech signal. Chapter 3 focuses on the modeling of speech, describes the state of art models with an emphasis on Hidden Markov models. It also describes the proposed acoustic model based on short-term time series by giving also the similarity measures and clustering methods involved in the usage of the concepts in the thesis.

The second part of our work consists of applications of short-term time series. Chapter 4 is reserved to keyword spotting task, by setting the problem definition and background, introducing the proposed solution and reporting the experiments conducted. Chapter 5 covers the voice activity detection task and gives a substantial evaluation of the existing methods along with the proposed one. Chapter 6 considers the short-time time series approach in speaker identification. Conclusions and future work directions are given in Chapter 7.

2. SPEECH PROCESSING BASICS

This chapter reviews the basic concepts in speech signal processing. Firstly, speech production and acoustics aspects are described; after that, terms from phonology are introduced. The feature extraction part shows how the features used for the experiments were computed. Finally, concepts from the probability model of speech and the state-of-art modeling technique, namely hidden Markov models, are presented.

2.1 Speech production and acoustics

Speech is the main way to communicate in the human society. It can be seen as the process of creating vocal sounds, which are grouped together to form words, and those are used to express the human thoughts, ideas and feelings. The human civilization development as it is today would not have been possible without speech as a means of communication. The human's speech production apparatus is the one to create vocal sounds. In speech production all the apparatus' components (see Figure 2.1) contribute by interacting with each other and changing their shapes. The apparatus' components usage gives us the main characteristics of the voice: the *loudness* describes the magnitude of an auditory sensation, the *pitch* describes the perceived attribute of sounds that makes possible an ordering of sounds, usually connected to the frequency of a sound, and the *timbre* or *tone color* can be is seen as the quality of sound that makes the difference between two sounds with the same loudness and pitch level. The sounds can be classified in relation to the position of the articulators and analyzed using their spectrograms.

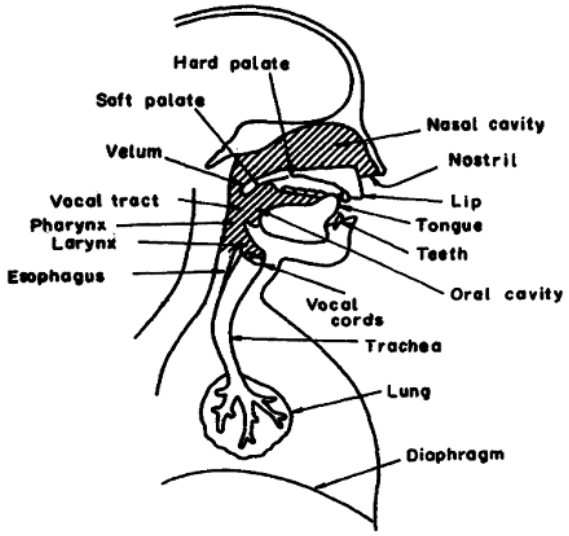


Figure 2.1: Human's speech production apparatus [Furui2000].

Speech production

Speech is produced as follows. The air, expelled from lungs goes through *trachea* and *esophagus* and reaches the *larynx*, where the vocal cords form a V-like opening called *glottis*. The vocal cords are responsible for the type of the sound produced (voiced/unvoiced). When the vocal cords are tensed, the air makes them to vibrate and the results are so-called *voiced* speech sounds: voiced consonants (like /m/, /z/) and all vowels. In the case when the vocal cords are completely open the *unvoiced* sounds like /f/, /p/ are produced. The intermediary situation corresponds to whispering.

The frequency of vibrations, in oscillation mode, is called the *fundamental frequency* and abbreviated *F0*. For males, the average value of *F0* is about 120 Hz, while for females and children it is about 220 Hz and 330 Hz respectively [Kinnunen2003]. Figure 2.2 depicts the glottis oscillation when periods of completely closed vocal cords alternate with opened ones.

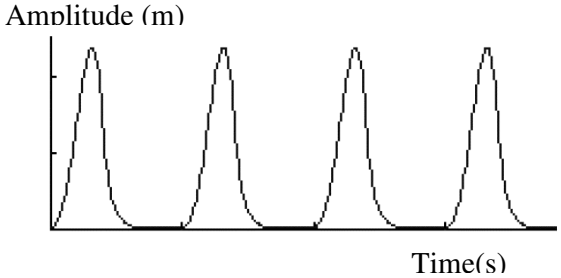


Figure 2.2: Glottis oscillation [Grebenskaya2005].

Next, the airflow goes through the *vocal tract* formed by the *pharynx* (throat cavity), the *velum* (*soft palate*), and the *oral* and *nasal cavities.* While the shape of the nasal cavity is fixed, the characteristics of the oral cavity can vary according the sounds pronounced. The articulators as a part of the vocal tract may change their lengths and shapes and so the vocal tract properties:

- *Pharynx*: between larynx and oral cavity.
- *Velum* closes/opens the nasal cavity for producing sounds as /m/ and /n/.
- *Tongue* changes the oral cavity shape, is the most flexible articulator involved in most of the sounds produced.
- *Teeth* used for pronunciation while contacting with the tongue.
- *Lips* are completely closed for some consonants (/p/, /m/), while for vowels can be rounded (/o/) or spread (/i/).
- *Hard palate* separates the mouth from the nasal cavity and allows producing some consonants.
- *Alveolar ridge:* between top teeth and hard palate, in conjunction with tongue creates sounds as /d/ or /t/.

The vocal tract can be modeled with a concatenation of tubes of varying cross-sectional area. At one end are the vocal cords while at the other one are the lips. The acoustic theory stands that the transfer function of this model can be described in terms of natural frequencies (*resonances*). The *formants* are the resonant frequencies of the vocal tract when vowels are pronounced. The first three formants (<3500 Hz) are the most significant in speech processing and carry out most of the acoustic energy.

Source-filter model

In speech analysis, it is useful to describe a vocal tract with a source-filter model [Huang2001]. The process of speech production is seen as a *sound source* (glottal air flow) passing through the *filter* (vocal tract). The filter shapes the source spectra as in Figure 2.3, where the example is for the neutral vowel /ə/ ("bird").

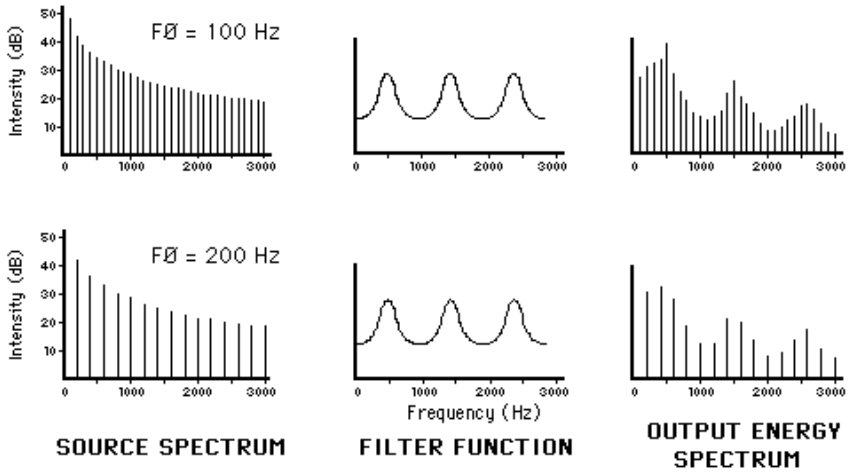


Figure 2.3: Source-filter model of speech production for *F0* = 100Hz and *F0* = 200 Hz [Grebenskaya2005].

From the filter transfer function we can see that the vocal tract has the first three formants equal to 500 Hz, 1500 Hz and 2500 Hz. These values can be calculated analytically using the following formula [Harrington1999]:

$$F_n = \frac{(2n-1)c}{4L}, \tag{2.1}$$

where L is the vocal tract length, n is the formant number and c is the velocity of sound.

As it is seen in Figure 2.3 the final spectrum is the result of two components: the source, or fast-varying part, and the filter, or slow-varying component. The filter transfer function changes with the pronunciation of other sounds because of modifications in vocal tract length caused by the changes in positions and shapes of the articulators. The independence between its two parts is the main assumption of the source-filter model, which is not generally true but gives a good approximation. Popular feature extraction methods based on *cepstral analysis* (discussed in this chapter) exploit this assumption by trying to separate excitation (source) from filter characteristics.

2.2 Phonology

According to linguistic dictionary [SIL2007], *phonology* is the study of how sounds are organized and used in natural languages. A phonological system of a language includes an inventory of sounds and their features and rules of sound interactions. The *phoneme* is the basic concept of phonetics and phonology, and it is seen as "the smallest contrastive unit in the sound system of a language". A phoneme can include slightly different sounds or phones. For example

in words “pin” and “spin” the /p/ sound is pronounced differently but it is still considered to be a single /p/ phoneme. The main characteristic of a phoneme is the ability to differentiate between words, like in the case of the pair “milk” and “silk” which differ by a single phoneme.

The phonemes properties like ways of production, features, interactions with other phonemes, play an important role in speech processing. Usually phonemes are grouped into *vowels* and *consonants*.

Vowels

The vibration of the vocal cords causes vowels when there is no closure of the vocal tract. An example of the waveform and spectrogram of the vowel /i/ in the word “kiitos” (“thank you” in Finnish language) pronounced by the author is shown in Figure 2.4. On the spectrogram we can notice that vowels have large dark regions corresponding to formant frequencies while the waveform demonstrates the periodic nature due to glottis oscillation.

.

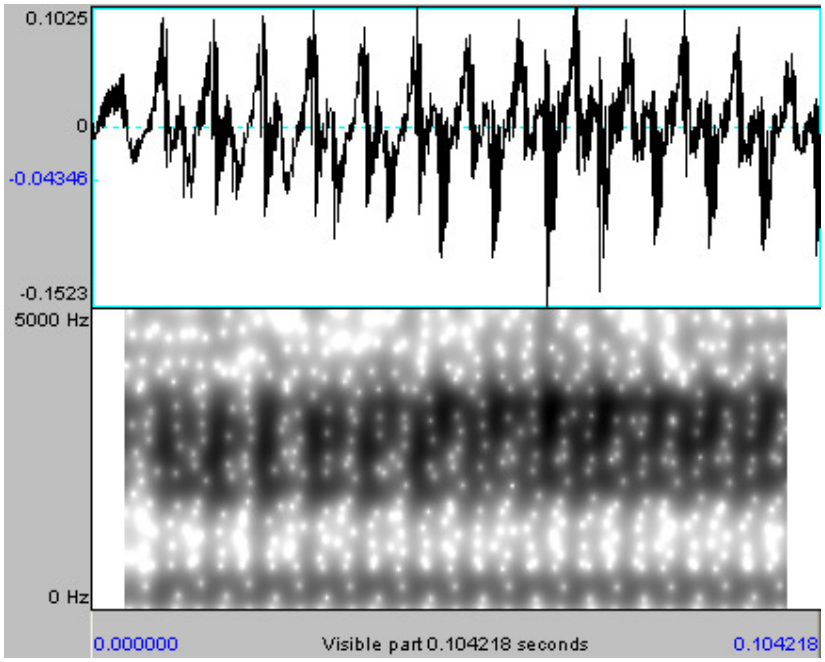


Figure 2.4: Waveform (up) and spectrogram (down) for the phoneme /i/.

The articulators (especially the tongue) are responsible for the individuality of the vowels through the changes performed in the shape of the oral cavity. Depending on the position of the tongue, vowels are classified as *low* (/aa/, /ae/ etc.) and *high* (/iy/, /uw/ etc.). Vowels, like /ao/ and /ow/, are called *rounded* because of the shape of lips.

Consonants

Consonants differ from vowels by an obstruction or constriction that occurs in the pharyngeal or oral parts of the vocal tracts. According to the way of production consonants can be classified as:

- Nasals produced by the air passing the nasal cavity (i.e. /m/, /n/).
- Semivowels similar to vowels, influenced by the context (i.e. /l/, /w/).

- Fricatives conditioned by the constriction position in the vocal tract (i.e. /v/, /s/).
- Stops product by sudden release of air from closed oral cavity (i.e. /b/, /p/).
- Affricates, the class of complex articulations, are combinations of two types (i.e.: /ch/ in 'church' is a combination of the phonemes /t/ and /sh/).

Another criterion for classification is the articulation position [Huang2001].

The information about phonemes classes became substantially important for speech recognition systems based on complex units of sub-words such as triphones.

2.3 Feature extraction

Feature extraction aims at giving a useful representation of the speech signal by capturing the important information from it. A common division of the feature extraction approaches is into *production-based* and *perception-based* methods. *Linear predictive coding* (*LPC*) is an example from the first group while *Mel-frequency cepstral coefficients* (MFCC) and *Perceptual Linear Prediction* (PLP) belongs to the perception-based approaches family. In this section, we outline the computation of MFCC features that involves first applying a pre-emphasis over the signal and a windowing process.

Pre-emphasis

Pre-emphasis is the process of emphasizing higher frequencies by passing a signal through a filter. Since in speech signal the low frequencies carries the most part of the energy, the pre-emphasis also balances the energy of the signal. The formant peaks became more visible in spectrum after pre-emphasis. Figure 2.5 shows an example of pre-emphasis taken from [Grebenskaya2005]. An unprocessed signal is plotted along with its short-time magnitude spectrum and spectrogram on the left side of the figure, while the same characteristics for the emphasized signal are plotted on the right side.

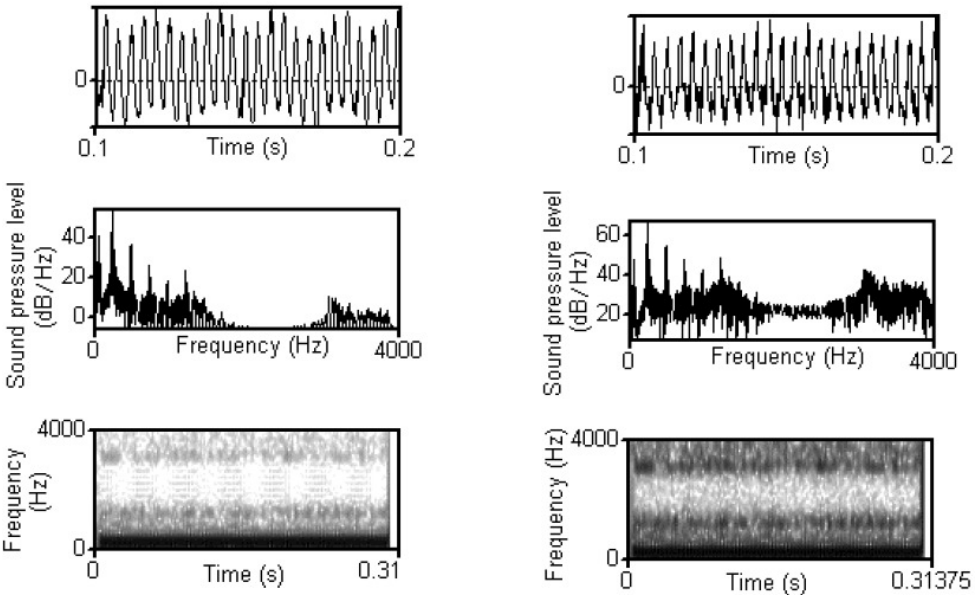


Figure 2.5: Unprocessed (left) and pre-emphasized (right) signals [Grebenskaya2005].

After pre-emphasis the spectrum becomes more flat because of the rise in energy in the high frequency region, effect well seen on the spectrogram where to higher energy corresponds to the darker parts. Pre-emphasis filter is designed to compensate the effect of the glottal-source and energy radiation from the lips [Kinnunen2003] when the spectrum of the recorded speech signal is with 6 dB/octave lower than the original, i.e. vocal tract, spectrum. The filter is implemented by the equation:

$$y[n] = x[n] - ax[n-1], \tag{2.2}$$

while its transfer function is:

$$H(z) = 1 - az^{-1}, \tag{2.3}$$

with $a \approx 1.0$.

Windowing

Under the assumption that the signal is *stationary* within a short time interval, a *short-time analysis* is applied for extracting the spectral features. These intervals are known as *frames* and their length is usually around 20-25 ms. The frame length is a trade-off between the assumption which implies a short length and the number of samples captured to compute the parameters. The frames are also overlapped by approximately 25 to 50% of their length. For each frame a *windowing function* is applied to reduce the effect of discontinuities at the edges of the frames. The most used window in speech analysis is the *Hamming* window [Furui2000]:

$$w[n] = \begin{cases} 0.54 - 0.46\cos\dfrac{2\pi n}{N}, 0 \le n < N \\ 0, otherwise \end{cases} \tag{2.4}$$

All the feature extraction techniques analyze the output of the windowing process.

Mel-frequency cepstral coefficients (MFCC)

The human perception of the frequency spectrum of sound does not have a linear behavior. The mel-scale frequency axis tries to approximate this behavior and is the reason why it is often used. The relation between mel- and linear frequency in KHz is given by [Furui2000]:

$$Mel = 1000\log_2(1+f) \tag{2.5}$$

The mel-frequency cepstral coefficients are defined as a discrete cosine transform of the log filterbank amplitudes. Every filter computes the average spectrum around each central frequency. An example of a filterbank used in MFCC calculations is shown in Figure 2.6.

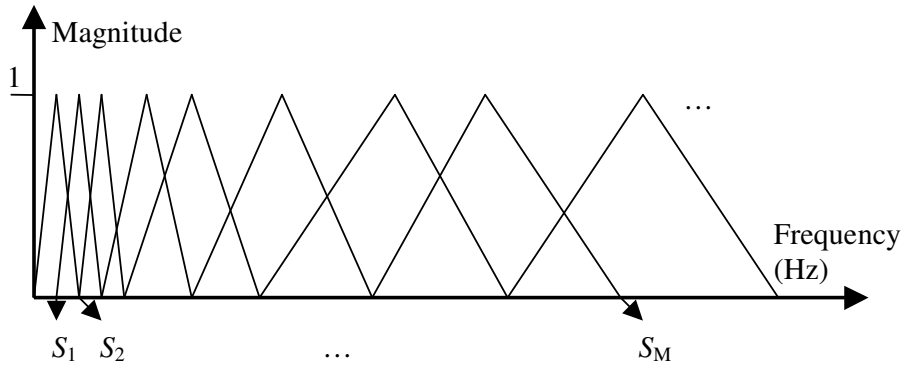


Figure 2.6: Critical band filters used in MFCC computation and their outputs (S_1, S_2,…,S_M).

In order to compute MFCC, the following steps are needed:

1. Compute the FFT-based spectrum

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi nk}{N}}, 0 \le k < N \tag{2.6}$$

2. Pass the magnitude spectrum *X[k]* through the mel filterbank, which is equivalent to multiplying each Fourier transform magnitude coefficient by the corresponding filter value. The result of the step is the set of *M* values representing the energy in each band, where *M* is the number of filters in the filterbank.

3. Compute log-energy at the output of each filter

$$s[m] = \ln\left[\sum_{k=0}^{N-1} |X[k]|^2 H_m[k]\right], 0 \le m < M \tag{2.7}$$

4. Convert log-energies to cepstral coefficients using *discrete cosine transform* (DCT)

$$c[n] = \sum_{m-0}^{M-1} s[m] \cos\left(\frac{\pi n(m-0.5)}{M}\right) \tag{2.8}$$

From the equation (2.8) we see that *c[0]* can be considered as a total log energy. For speech recognition, the first 13 cepstrum coefficients are usually used. MFCCs are weakly correlated and they have a good discrimination property.

The main idea of the cepstral analysis is to separate the source part from the filter part in the signal and represent them as a linear combination [Deller2000]. This is important since the main interest in speech processing is to estimate the parameters of the vocal tract (see section 2.1) represented by the filter. The cepstrum involves logarithm of the magnitude, which results in summation of logarithms of source and filter magnitudes. After applying DCT (in the case of MFCC) we get the characteristics of the slow-varying part (filter/vocal tract) concentrated in the low cepstral coefficients.

Dynamic characteristics

The vocal tract is not characterized only by "static" parameters (like LPCC or MFCC). During speech production, the articulators change their positions continuously, and this information might be useful for speech processing. The dynamic information is estimated by so-called *delta-features* computed as [Huang2000]:

$$\Delta c_k = c_{k-2} - c_{k+2}, \tag{2.9}$$

where c_k is the vector of cepstral coefficients at the frame k.

Replacing c_k by Δc_k in the equation (2.9) we get the second-order derivatives, which are called *delta-delta* or *acceleration* coefficients.

2.4 Probabilistic models for speech

State-of-art *automatic speech recognition* (ASR) systems are based on probabilistic modeling of the speech signal and use hidden Markov models (discussed further in this chapter), *artificial neural networks* (ANN) [Rabiner1993] and their variants. The probabilistic models are successfully used for medium to large vocabulary ASR systems by significantly increasing the recognition rate [Furui2000].

Given a word sequence W, which produces an observation sequence X, a probabilistic ASR system decodes the sequence to a string of words. According to Bayes' rule the decoded string will have the maximum a posteriori probability [Rabiner1993, Juravsky2000, Furui2000]:

$$\hat{W} = \arg\max_W P(W \mid X) = \arg\max_W \frac{P(X \mid W)P(W)}{P(X)} = \arg\max_W P(X \mid W)P(W) \tag{2.10}$$

where $P(X)$ is independent of W and is omitted in the final form in a forced notation. The term $P(X|W)$, the *likelihood*, represents an *acoustic model* and reflects the probability for an observation X given a word sequence W. The term $P(W)$, the *prior probability*, represents a *language model* and gives the probability for specific word sequences to occur.

The automatic speech recognition process can be seen as in Figure 2.7, where the language and acoustic models are estimated from large training materials.

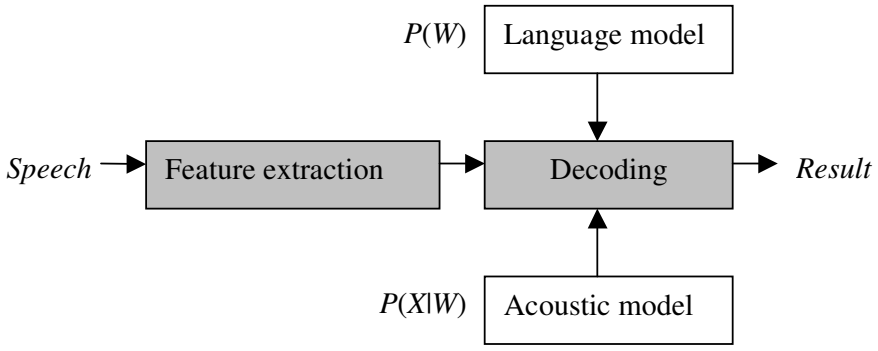


Figure 2.7: Probabilistic speech recognition model.

Under the assumption that all the words are independent from each other in the sequence, $P(X|W)$ can be expressed as

$$P(X \mid W) = \prod_{i=1}^{K} P(X^i \mid w_i) \tag{2.11}$$

where $X=\{X^1,X^2,...,X^K\}$ and w_i is the i-th word of the sequence W. The expression (2.11) is a *whole-word modeling*. Furthermore, by assuming the independence of phonemes, as acoustic units of the words, the term $P(X^i \mid w_i)$ can be rewritten as

$$P(X^i \mid w_i) = \prod_{j=1}^{M} P(X_j^i \mid ph_{j,i}) \tag{2.12}$$

where M is the number of phonemes in the word w_i, and $ph_{j,i}$ is the j-th phoneme of the word. The equation (2.12) depicts a *phoneme-based* word model.

In modeling, elements of variability in pronunciation, context as well the adaptability to new words, speakers and environments are usually taken into account.

Statistical Language Model (SLM) assigns a probability to a sequence of words W, shows how often W is met as sentence in the language. SLM can be written as

$$P(W) = p(w_1)\prod_{i=2}^{M} p(w_i \mid w_{i-1}, w_{i-2},..., w_1) \tag{2.13}$$

where sequence $w_{i-1}, w_{i-2},\ldots, w_1$ is the *history* of the word w_i, and M is the number of words from W. There is no good method to compute the probability of a word given a long history; we cannot just count how many times it happened in our material for our history to be followed by the word given. The solution is to approximate it by considering short histories of maximum N words in the case of *N-grams language model*:

$$P(W) \approx p(w_1)\prod_{i=2}^{M} p(w_i \mid w_{i-1}, w_{i-2},..., w_{i-N+1}) \tag{2.14}$$

2.5 Hidden Markov model

2.5.1 Fundamentals of HMMs

Hidden Markov Model is a stochastic finite state machine. It can be seen as a pair of stochastic processes: a *hidden Markov chain* and an *observable* process which is a probabilistic function of the chain states [Trentin2001]. Based on the current state, the *probability density functions* (pdf) attached to the state and the *state transition probabilities* the HMM changes its state and emits an *observation.* Since we can have a sequence of observations after a number of steps but without knowing exactly by which state each observation was produced, the state sequence is "hidden" and justifies the name of HMM.

In speech processing, in order to apply HMMs, we make two main assumptions. A *first-order Markov assumption* considers that the transition depends only on the current and the destination states. An *output-independent assumption* takes all observations as independent from

each other and dependent only on the state of generation. Both assumptions do not hold for speech but in practice, usage of second-order HMMs and speech frames correlations do not improve the performance significantly and they increase the computational complexity [Huang2001].

According to [Rabiner1989], an HMM is defined by:

- Number of states, N
- Set of all states, $S = \{s_1, s_2 \ldots s_N\}$
- Number of different observation symbols per state, M
- Set of all observations, $V = \{v_1, v_2 \ldots v_M\}$
- State transition probability distribution $A = \{a_{ij}\}$, where $a_{ij} = P\,(q_{t+1} = s_j \mid q_t = s_i)$ $(i,j \leq N)$
- Observation probability distribution in state j, $B = \{b_j(k)\}$, where $b_j(k) = P(o_t = v_k \mid q_t = s_j)$
- Initial state distribution $\pi = \{\pi_i\}$, where $\pi_i = P\,(q_1 = s_i)$.

We use the compact notation $\lambda = \{A,B,\pi\}$ to define a HMM. The parameter B also refers "state output function" and usually defines the type of the HMM. The modeling of output spectral distributions is made by:

- discrete modeling using vector quantization (VQ) approach [Rabiner1983], constructing codebooks from observations via clustering methods (see next section).
- continuous modeling using pdf as Gaussian distributions [Huang2001].
- semi-continuous modeling, a compromise between discrete and continuous modeling [Huang1990].

In this chapter, we focus only on the discrete case.

In speech processing, HMM is a *template* for *recognition unit*s. For example, HMMs can be templates for each phoneme or word used as recognition unit by ASR. For a set of HMMs (or units) and a sequence of observations, we must compute the *most likely* models to produce the specific sequence. For this have to solve the three basic problems of HMMs [Rabiner1993]:

1) *Recognition problem*: given a model λ and sequence of observations $\boldsymbol{O}$ how to efficiently compute $P(\boldsymbol{O}|\lambda)$?
2) *Decoding problem*: given a model λ and sequence of observations $\boldsymbol{O}$, how to choose a state sequence that most likely produced $\boldsymbol{O}$?
3) *Training problem*: how to adjust HMM parameters $\lambda = \{A, B, \pi\}$ to maximize $P(\boldsymbol{O}|\lambda)$?

2.5.2 Recognition with HMM

Considering $Q = q_1 q_2 \ldots q_T$ to be a fixed state sequence the probability of the observation $\boldsymbol{O}$ and Q given the model λ is

$$P(O,Q \mid \lambda) = P(Q \mid \lambda)P(O \mid Q,\lambda) = \left(\pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T}\right)\left(b_{q_1}(o_1) b_{q_2}(o_2) \ldots b_{q_T}(o_T)\right) \quad (2.15)$$

The probability of $\boldsymbol{O}$ given the λ is obtained by summation $P(\boldsymbol{O},Q|\lambda)$ over all state sequences:

$$P(O \mid \lambda) = \sum_{allQ} P(O,Q \mid \lambda) = \sum_{q_1 q_2 \ldots q_T} \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) \quad (2.16)$$

Instead of using the formula there are efficient algorithms for calculating the $P(O|\lambda)$ value, namely *forward* and *backward* approaches [Rabiner1989]. Both of them are based on calculating auxiliary variables called forward and backward variables correspondingly.

Forward algorithm

The forward approach iteratively computes the forward variable $\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = s_i \mid \lambda)$, the probability at the time t of being in the state i and observing a partial sequence $o_1 o_2 \ldots o_t$ with the model λ. The algorithm follows the steps:

1. *Initialization*

$$\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N \quad (2.17)$$

2. *Induction*

$$\alpha_{t+1}(i) = \left(\sum_{j=1}^{N} \alpha_t(j) a_{ji}\right) b_i(o_{t+1}) \quad 1 \le i \le N, \; 1 \le t \le T-1 \quad (2.18)$$

3. *Termination*

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad 1 \le i \le N \quad (2.19)$$

Backward algorithm

The backward algorithm is similar to the forward one. It uses a backward variable, $\beta_t(i) = P(o_{t+1} o_{t+2} \ldots o_T \mid q_t = s_i, \lambda)$, the probability of the partial sequence, given the model λ and state s_i at time t.

1. *Initialization*

$$\beta_T(i) = 1 \quad 1 \le i \le N \tag{2.20}$$

2. *Induction*

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad 1 \le i \le N, 1 \le t \le T-1 \tag{2.21}$$

3. *Termination*

$$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \quad 1 \le i \le N \tag{2.22}$$

The principal difference between forward and backward approaches is that the backward case requires information about entire sequence of observations before starting calculations. The order of calculations is TN^2 in both cases.

2.5.3 Viterbi decoding

For decoding problem we use *Viterbi algorithm* [Viterbi1967]. In order to find the single best state sequence, $Q = q_1 q_2 \ldots q_T$, for a given sequence of observations $O = o_1 o_2 \ldots o_T$, we first define the quantity [Rabiner1989]:

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P(q_1 q_2 \ldots q_t = s_i, o_1 o_2 \ldots o_t \mid \lambda) \tag{2.23}$$

where $\delta_t(i)$ represents the highest probability at time t along a single path for the first t observations and ends in state s_i. An array $\psi_t(i)$ is used to keep the argument which maximizes the $\delta_t(i)$ in order to be able to retrieve the best state sequence. The Viterbi decoding process is similar to the forward algorithm (the summation is replaced with maximization) and goes as follows:

1. *Initialization*

$$\delta_1(i) = \pi_i b_i(o_1), \ \psi_1(i) = 0 \quad 1 \le i \le N \tag{2.24}$$

2. *Induction*

$$\delta_t(i) = \left(\max_{1 \le j \le N} \delta_{t-1}(j) a_{ji} \right) b_i(o_t), \ 1 \le i \le N, \ 2 \le t \le T \tag{2.25}$$

$$\psi_t(i) = \arg\max_{1 \le j \le N} \left(\delta_{t-1}(j) a_{ji} \right), \ 1 \le i \le N, \ 2 \le t \le T \tag{2.26}$$

3. *Termination*

$$P^*(O \mid \lambda) = \max_{1 \le i \le N} \delta_T(i) \tag{2.27}$$

$$q_T^* = \arg\max_{1 \le i \le N} \delta_T(i) \tag{2.28}$$

4. *Path backtracking*

$$q_t^* = \psi_{t+1}(q_{t+1}^*),\ T-1 \ge t \ge 1 \tag{2.29}$$

The Viterbi algorithm is a form of the *dynamic programming* method [Furui2000]. Using the formulas given above leads to extremely small probability values during calculations, and therefore, the underflow problem occurs. In order to avoid the underflow problem, the observation probability can be replaced with its logarithmic form [Young2001]:

$$\delta_t(i) = \max_{1 \le j \le N} \left(\delta_{t-1}(j) + \log(a_{ji}) \right) + \log(b_i(o_t)) \tag{2.30}$$

2.5.4 HMM training

The goal of the training step is to maximize $P(O|\lambda)$ for the training data by finding the parameters of the HMM model $\lambda=\{\pi, A, B\}$. There are numerous ways to initialize the HMM parameters including "flat start" where the models are initialized to be identical and have the global mean and variance, manual segmentation or *segmental k-means* segmentation with clustering [Rabiner1989]. Segmental k-means algorithm can also be used for HMM training. *Baum-Welch* [Baum1970] is another broadly used algorithm for training of HMMs and will be described in the following. Both the Baum-Welch and the segmental *k*-means approaches are examples of the *maximum likelihood estimation (MLE)* principle [Huang2001]. MLE works under the assumption that the data is large enough to find robust estimates of the model parameters.

Baum-Welch algorithm

An iterative *Baum-Welch algorithm* [Baum1970] is used for maximizing $P(O|\lambda)$ because we do not know a closed-form analytical solution. Baum-Welch algorithm is a generalized *expectation maximization* (*EM*) algorithm [Dempster1977]. For describing the re-estimation process of parameters, we define two auxiliary variables [Rabiner1993]. The first one is the probability of being in *i-th* state at time *t,* and in *j-th* state at time *t+1*, given the model and the observation sequence:

$$\xi_t(i,j) = P(s_t = i, s_{t+1} = j \mid O, \lambda) \tag{2.31}$$

The second variable, usually referred as a *state occupation*, is the probability of being in state *i* at time *t* given the model and the observation sequence:

$$\gamma_t(i) = P(s_t = i \mid O, \lambda) \tag{2.32}$$

By using forward and the backward variables previously introduced we have

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(O \mid \lambda)} = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \tag{2.33}$$

and

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)} \tag{2.34}$$

where $P(O \mid \lambda)$ is a normalization factor. A relation between these variables is:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j) \tag{2.35}$$

If $\gamma_t(i)$ are summed up over t, we get the expected number of visits in the state s_i. Similarly, by summing $\xi_t(i,j)$ we get the expected number of transitions from state s_i to s_j. By using the introduced variables the re-estimated parameters for the HMM are:

$$\overline{\pi_i} = \gamma_1(i) \tag{2.36}$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1}P\left(q_t = i, q_{t+1} = j, O \mid \lambda^{(i-1)}\right)}{\sum_{t=1}^{T-1}P\left(q_t = i, O \mid \lambda^{(i-1)}\right)} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{2.37}$$

$$b_j(k) = \frac{\sum_{\substack{t=1 \\ s.t.o_t=v_k}}^{T}P\left(q_t = j, O \mid \lambda^{(i-1)}\right)}{\sum_{t=1}^{T}P\left(q_t = j, O \mid \lambda^{(i-1)}\right)} = \frac{\sum_{\substack{t=1 \\ s.t.o_t=v_k}}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)} \tag{2.38}$$

The iterations of the Baum-Welch algorithm are guaranteed to convergence to a stable solution, each re-estimated model λ' will have $P(O \mid \lambda') \geq P(O \mid \lambda)$.

3. SHORT-TERM TIME SERIES ACOUSTIC MODEL

The main contribution of this thesis is the introduction and usage of short-term time series as an acoustic model in speech processing. A *time series* is a sequence of data points measured typically at successive times. In our notation, a *short-term time series* is a short sequence of data points taken at successive times, spaced at uniform time intervals. In analysis of audio signals the intuition is the following. A short-term time series covers a sequence of feature vectors extracted at uniform time intervals. In this approach, the order is important and the feature vectors taken as sequence is expected to convey better a segment of the signal.

In order to work with short-term time series that can be of different lengths we use the classic dynamic time warping alignment method and introduce two new Euclidean type distance functions. Based on those similarity distances we can perform clustering. The result of clustering will be the basis to acoustic models in the case of audio signals.

3.1 Phoneme and pseudo-phoneme

For short-term time series we distinguish two cases. When the group of subsequent feature vectors are extracted from a segment of audio signal with a known phonetic significance then we call the short-term time series as being a *phoneme*. On the other hand, when the sequence has no relevant, particular, phonetic meaning, and this is the general case, it is called *pseudo-phoneme*. If in the following subsections, even when we use the term “phoneme”, it is evident that the proposed similarity measures and the clustering methods will also work also for the pseudo-phonemes.

3.2 Similarity measures

The distance measures used between phonemes that are represented by feature vector sequences of different length are:

- *proportional centred distance* (PCD),
- *extended Euclidean distance* (EED), and
- symmetric *dynamic time warping* (DTW).

While PCD and EED are proposed distances and are Euclidean type distance functions, DTW is an alignment method largely used in the field of speech recognition. The calculation of the first two distances has linear time complexity. On the other hand, the DTW distance has a very good accuracy although quadratic time complexity.

3.2.1 Dynamic Time Warping (DTW)

In the symmetric case, dynamic time warping alignment method for two sequences X and Y (of lengths n and m) can be recursively computed as in expression (3.1). The recursion is typically solved using dynamic programming.

$$D_{i,j} = SED(X_i, Y_j) + \min\{D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}\}, i = \overline{1,n},\ j = \overline{1,m}, \tag{3.1}$$

where SED is the *standard Euclidian distance*

$$SED(A,B) = \sqrt{\sum_{k=1}^{p} |a_k - b_k|^2}, A, B \in R^p \tag{3.2}$$

Given a start and an end point of a sequence, the purpose of DTW is to determine, in an optimal manner, the warping function that provides the best time alignment between the two sequences. The DTW distance score is defined as the Euclidean distance, equation (3.2), of the optimal alignment. In our case, the score is normalized by dividing it by the sum of the sequence lengths (n+m) in order for the distance to be independent of the lengths of the sequences. The time complexity for computing DTW is $O(nm)$ and the space complexity is $O(n)$.

There are numerous approximation methods for the reduction of the time complexity [Itakura1975, Salvador2004, Keogh2000]. We prefer the original form (3.1) that assures not only the optimality of the solution but also the optimality of sub-solutions. In other words, if we have computed the matrix of DTW distance for two sequences, we have the DTW distances between any pair of sub-sequences that keeps the starting point of the initial sequence. An example of DTW computation is shown in Figure 3.1.

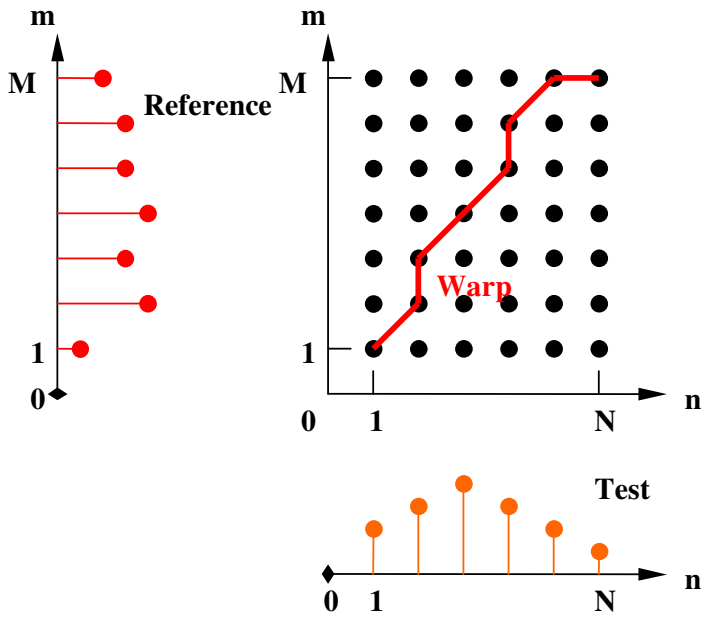


Figure 3.1: An example of computing DTW between two sequences of lengths M and N, respectively.

3.2.2 Proportional Centred Distance (PCD)

The idea behind PCD (Figure 3.2) is to provide a measure that works for sequences with different lengths. We divide the sequences into two halves, and find how many points from the longer sequence *Y* correspond to one point in the shorter sequence *X*. The distances are computed from the end points to the centre, and the remaining of the division between *m* and *n* will be the number of points on *Y* that correspond to the middle point of *X*. The time complexity for computing PCD is O(*n*) and the space complexity is O(*1*).

PCD_DISTANCE(*X*, *Y*): *PCD*

$step = \left\lfloor \frac{\lfloor m/2 \rfloor}{\lfloor n/2 \rfloor} \right\rfloor$;

$D = 0$;

for $i = 1$ to $\lfloor n/2 \rfloor$ do

  for $j = (i\text{-}1)*step+1$ to $i*step$ do

    $D = D + SED(X_i, Y_j) + SED(X_{n-i+1}, Y_{m-j+1})$;

for $j = 1$ to $m\text{-}step*n$ do

  $D = D + SED(X_{\lfloor n/2 \rfloor+1}, Y_{\lfloor n/2 \rfloor*step+j})$;

$PCD = D / m$;

Figure 3.2: Pseudocode for the proportional centred distance (PCD), where *m* and *n* are the lengths of the sequences *X* and *Y* (*m* > *n*).

An example of using PCD is demonstrated in Figure 3.3 where *n*=3, *m*=7, *step*=3, D_i is the Euclidean distance between two corresponding points. The result can be defined mathematically as follows:

$$PCD(X,Y) = \frac{1}{m}\sum_{i=1}^{m} D_i \tag{3.3}$$

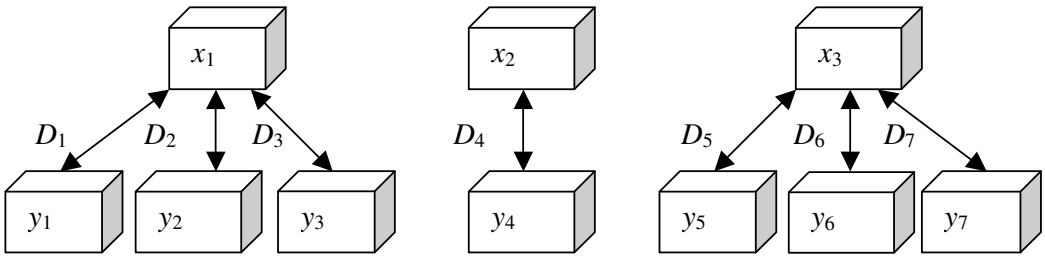


Figure 3.3: Example of computing PCD for two sequences *X* and *Y*.

3.2.3 Extended Euclidean Distance (EED)

EED compromises between a straightforward approximation of PCD and the Euclidean distance. In consequence, the proportionality of PCD is kept also in EED, but it operates on real values so

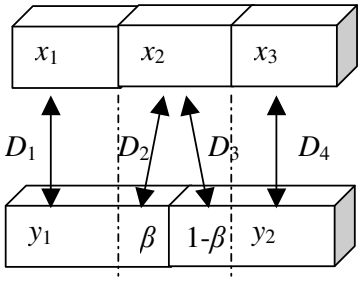
that a point in the longer sequence may correspond to one or at most two points in the shorter sequence. When there is two points corresponding to a point then the distances are computed between the points and added proportionally with how much correspond each point to the point from the longer sequence. Pseudocode for calculating EED is given in Figure 3.4.

```
EED_DISTANCE(X, Y): EED

p = n/m;
D = 0;
pp = p;
j = 1;
for i = 1 to n do
        pr = i-pp;
        if (pr = 0)
                D = D + SED(X_i,Y_j);
        else
                if (pr = 1)
                        D = D + SED(X_i,Y_j+1);
                else
                        D = D + (1-pr) SED(X_i,Y_j) + prSED(X_i,Y_j+1);
                j = j + 1;
                pp = pp + p;
EED = D / n;
```

Figure 3.4: Pseudocode for the extended Euclidean distance (EED). Here m and n are the lengths of the sequences X and Y ($n > m$).

Figure 3.5 shows an example of EED computation for two sequences of length 2 and 3. In the worst case, EED computes two times more point-to-point distances than PCD. Both PCD and EED have linear time complexity and constant space complexity.



$$EED = (D_1 + (\beta D_2 + (1-\beta)D_3) + D_4)/3$$

Figure 3.5: An example of computing EED for sequences X of length 3 and Y of length 2 ($\beta = 0.5$).

3.3 Clustering methods

We are using clustering methods to reduce the number of samples so that we keep the most representative samples. This is necessary since working with all the samples that we have is time consuming. The samples (phonemes) are sequences of the feature vectors of different lengths. The input for clustering is formed by the sets of samples for each phoneme, while the output will contain the most representative samples for each phoneme (Figure 3.6). We implement the following clustering algorithms: several iterations of a *random clustering*, the *k-means algorithm*

[MacQueen1967] repeated for a number of random initial codebooks and the *k-medoids algorithm* [Kaufman1987] repeated for a number of randomly initial codebooks. The random clustering algorithm and the k-medoids are suitable for all the distances discussed. The k-means algorithm cannot be used with DTW because there is no good method to compute the cluster centroids. Taking of the average feature vector for each sample/phoneme in the clustering process is unsatisfactory since the information given by the order of feature vectors is not used.

In the clustering, we want to have representative sequences of the features for each phoneme. The distance between a phoneme and a codebook (DPCB) is computed using the formula (3.4) where PCD, EED or DTW can be used as the distance function between the two phonemes ($dist(ph_i,s_k)$).

$$DPCB(ph_i, cb_j) = \min\{dist(ph_i, s_k) \mid s_k \in cb_j\} \tag{3.4}$$

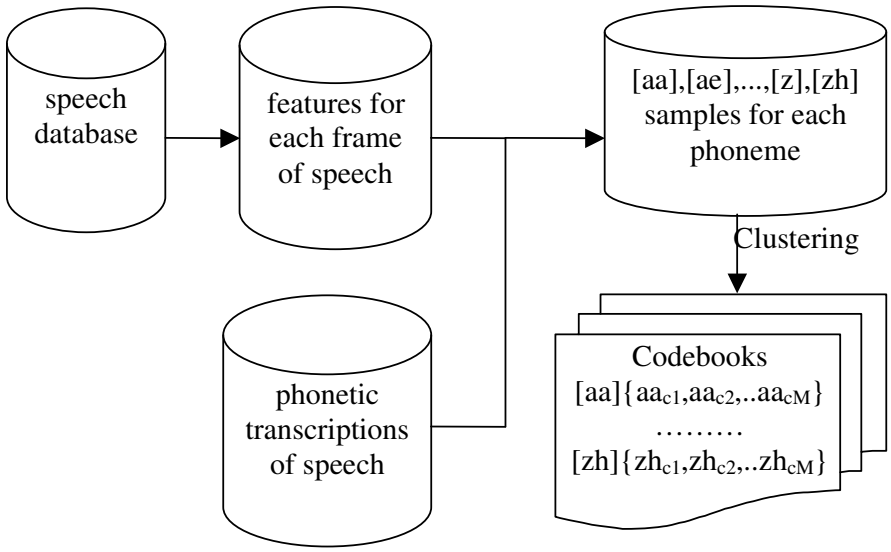


Figure 3.6: Generation of the phoneme codebooks from the phonetic transcriptions.

3.3.1 Random clustering

We apply repeatedly a random selection in the set of samples, evaluate the candidate codebook and maintain the best solution with the least distortion in terms of *mean absolute error* (MAE) (see Figure 3.7).

RANDOM_CLUSTERING(X, T): S, S_{MAE}

S = RandomCodebook;

$$S_{MAE} = \frac{1}{N}\sum_{i=1}^{N} \min_{p=1,M} dist(X_i, c_p);$$

for k = 1 to T-1 do

C = RandomCodebook;

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \min_{p=1,M} dist(X_i, c_p);$$

if ($MAE < S_{MAE}$)

$S = C$;

$S_{MAE} = MAE$;

Figure 3.7: Pseudocode for the random clustering algorithm repeated for T times. Here N is the number of samples to be clustered, M is the number of clusters, C is the codebook and c_p are the samples from the codebook. *MAE* stands for *mean absolute error*, and S is the best codebook with the distortion S_{MAE}.

3.3.2 K-means clustering

The k-means algorithm has two main steps: partition step and centroid step. While in the partition step the samples are grouped around the centroids by using the minimum distance criteria, the centroid step suppose computing the average vector values for the clusters/partitions found in the previous step. With our proposed distances it is difficult for samples of different lengths to compute the average, the mean of the samples. The solution comes from bringing the samples at the same length. The concatenation of feature vectors will not solve the problem unless it is a dimensionality reduction step to bring the samples at the same length, followed by a reverse procedure to take the corresponding phoneme. Anyway, the basic concept of STS modeling is to preserve and to use the information given by the order in sequence, which is not the case for the concatenation approach.

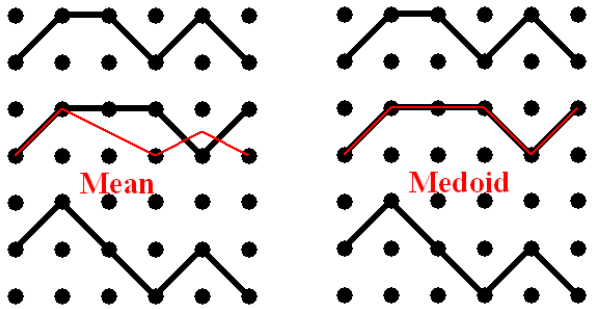


Figure 3.8: The mean and the medoid computed for a set of 3 sequences of same lengths.

In the k-means algorithm (Figure 3.11) with PCD or EED, we project each phoneme from the entire set of samples to the longest length from the entire set. In this way the averaging of samples for the construction of centroids (mean samples, Figure 3.8) is done on the projected samples. The projection (Figure 3.9) is done similarly with the PCD computation (Figure 3.3). Also EED projection (Figure 3.10) follows the EED computation (Figure 3.4).

```
PCD_PROJECTION( X, n, m): Y

step = ⌊ ⌊m/2⌋ / ⌊n/2⌋ ⌋;
D = 0;
for i = 1 to ⌊n / 2⌋ do
        for j = (i-1)*step+1 to i*step do
                Y_j = X_i;
                Y_{m-j+1} = X_{n-i+1};
for j = 1 to m-step*n do
        Y_{⌊n/2⌋*step+j} = X_{⌊n/2⌋+1};
```

Figure 3.9: Pseudocode for projecting in PCD sense of a sequence X (length n) to a sequence Y (length m, $m > n$).

```
EED_PROJECTION( X, n, m): Y

p = m/n;
pp = p;
j = 1;
for i = 1 to m do
        pr = i - p;
        if (pr = 0)
                Y_i = X_j;
        else
                if (pr = 1)
                        Y_i = X_{j+1};
                else
                        Y_i = (1-pr) X_j + pr X_{j+1};
        j = j + 1;
        pp = pp + p;
```

Figure 3.10: Pseudocode for projecting in EED sense of a sequence X (length n) to a sequence Y (length m, $m > n$).

K-MEANS_CLUSTERING(X, T): S, S_{MAE}

$S_{MAE} = +\infty$;
for k = 1 to T do
    $MAE = +\infty$;
    C = RandomCodebook;
    do
        for i = 1 to N do
            $p_i = \arg\min_{j=1,M} dist(X_i, c_j)$;
        for j = 1 to M do
            $c_j = \text{average}_{p_i=j} X_i$;
        $MAE_{\text{prec}} = MAE$;
        $MAE = \frac{1}{N}\sum_{i=1}^{N} \min_{j=1,M} dist(X_i, c_j)$;
    while $\left|MAE_{prec} - MAE\right| > \varepsilon$;
    if ($MAE < S_{MAE}$)
        $S = C$;
        $S_{MAE} = MAE$;

Figure 3.11: Pseudocode for the k-means clustering algorithm repeated for T times. Here N is the number of samples to be clustered, M is the number of clusters, p_i is the cluster number of the sample X_i, C is the codebook and c_j are the centroids from the codebook. *MAE* stands for *mean absolute error*, and S is the best codebook with the lowest mean absolute error, S_{MAE}.

3.3.3 K-medoids clustering

The k-medoids (Figure 3.12), known also as *Partitioning Around Medoids* (PAM) algorithm [Kaufman1987], is a variant of the k-means algorithm (Figure 3.11). Instead of computing centroids by averaging the samples from clusters at each iteration, k-medoids computes the medoids (Figure 3.8), which represent the samples from clusters that minimize the distortion (MAE) in each cluster.

K-MEDOIDS_CLUSTERING(X, T): S, S_{MAE}

$S_{MAE} = +\infty$;
for k = 1 to T do
    $MAE = +\infty$;
    C = RandomCodebook;
    do
        for i = 1 to N do
            $p_i = \arg\min_{j=\overline{1,M}} dist(X_i, c_j)$;
        for j = 1 to M do
            $t = \arg\min_{s=\overline{1,N}, p_s = j} \sum_{p_i = j} dist(X_i, X_s)$;
            $c_j = X_t$;
        $MAE_{prev} = MAE$;
        $MAE = \frac{1}{N} \sum_{i=1}^{N} \min_{j=\overline{1,M}} dist(X_i, c_j)$;
    while $\left| MAE_{prev} - MAE \right| > \varepsilon$;
    if ($MAE < S_{MAE}$)
        $S = C$;
        $S_{MAE} = MAE$;

Figure 3.12: Pseudocode for the k-medoids clustering algorithm repeated for T times. N is the number of samples for clustering. M is the number of clusters (size of codebook). p_i is the cluster number for sample X_i, C is the codebook, c_j are the medoids from codebook. S is the best codebook with the lowest mean absolute error, S_{MAE}.

4. APPLICATION TO KEYWORD SPOTTING

The goal of a word spotting system is to provide the capability of searching through audio content based on queries. The query can be formed from a text representation, a phonetic transcription or a speech sample. The search will return the positions of possible occurrences and the corresponding confidence scores.

4.1 Background

The easiest way to perform word spotting is to obtain transcription of the speech by *large vocabulary continuous speech recognizers* (LVCSR), and then performing text retrieval on the transcription. In the application domain of broadcast news clips, this approach has been found sufficient [Garofolo2000].

Unfortunately this approach does not generalize to the problem of searching arbitrary keywords from continuous speech. A phoneme-lattice based approach to solve vocabulary independent audio search has been proposed in [Seide2004]. The *speech recognition lattice* is an intermediate format, usually an acyclic graph, in which the hypotheses for phonemes, sub-words or words are scored. They also show comparative results using phoneme lattices and word lattices. In [Yu2004], the authors combine both of these methods in a word/phoneme hybrid model.

A comparison between acoustic keyword spotting, spotting in word lattices generated by a LVCSR and a hybrid approach making use of phoneme lattices generated by a phoneme recognizer can be found in [Szöke2005]. The most widely used model in the current approaches is the HMM, using GMMs to model the distribution of the features for each phoneme state.

4.2 Word spotting system

The word spotting systems usually are based on laborious acoustic models (HMM/GMM, HMM/ANN, lattice) using complex linguistic units (phoneme groups, sub-words, words) and linguistic models in order to have high performances. All those make the system to be dependent on the language spoken and on the speech context (the decision in one point uses the surrounding contextual information) and low scalable for words that were not present in the training vocabulary.

We introduce a new approach based on short-term time series to solve the problem in the form of a speaker, vocabulary and context independent word spotting system for continuous speech. This approach made the subject of [Timofte2006]. The approach is based on the following elements:

- The basic block is the phoneme, seen as a short-term time series of feature vectors.

- The acoustic models used are represented by characteristics groups of samples or codebooks for each utilized phoneme.
- The phoneme is the linguistic unit, taken independent of the context represented by neighbor phonemes.
- In order to achieve context independence we do not integrate language models or other a priori information related to specific language statistics.
- For keyword searching from audio material we extract the same feature vectors used in the trained phoneme codebooks.
- By using the phoneme codebooks we compute for each possible start position and length in the audio material similarities that are normalized. Those create a space of estimated probabilities for phonemes.
- In the space of probabilities we match the pattern of the keyword represented by its phonetic transcription in order to obtain the maximum probability.
- We take the decision if the keyword occurs or not at a specific position in the audio material according to a threshold.

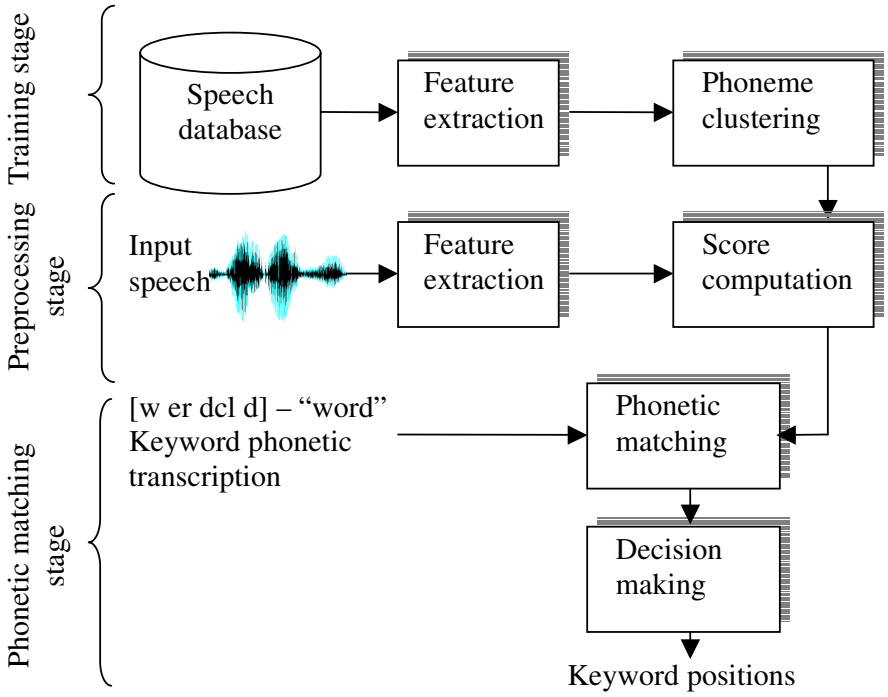


Figure 4.1: Structure of the three-stage word spotting system.

To sum up (see Figure 4.1), the word spotting system consists of three stages: training stage, preprocessing stage and phonetic search. In the training stage, we create an acoustical model of each phoneme, which are the clustered feature vector sequences. Phonemes are extracted independently from the linguistic context. In the preprocessing stage, we assign scores for each possible phoneme occurrence by creating a phoneme-level representation of the background speech. The phonetic search consists of pattern matching of the keyword phonetic transcription in the previous matrix of scores and computing scores to decide accurately the possible presence of the keyword. Due to the modularity of these stages it is possible to use

previously stored phonetic information from the preprocessing stage of an audio recording and to run the phonetic matching to obtain results faster without recalculating the scores. In this case, the system would be suitable for audio indexing and information retrieval as well.

Acoustic modelling of phonemes

In the training stage, short-term time series of feature vectors are extracted from the speech database according to the phonetic transcriptions for each phoneme. We make codebooks of representative samples for each phoneme individually by clustering methods to form the acoustic model of the phoneme. The clustering process is described in the chapter 3.

Score computation

In the preprocessing stage (see Figure 4.2), we create hypotheses for each phoneme, for each start frame in the background speech, and for each possible length of the phoneme.

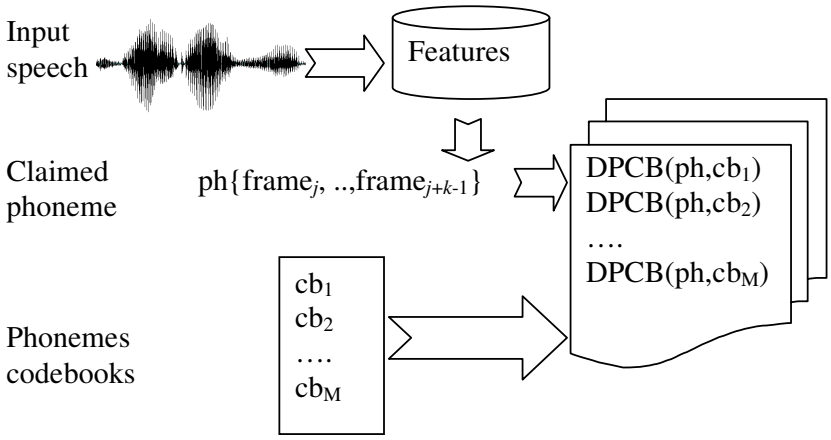


Figure 4.2: Computing distances for each phoneme codebook starting with frame *j* and the possible phoneme of length *k*.

Starting with frame *j*, we compute distances between each phoneme codebook and the possible phoneme that starts from this particular position in the background speech with a fixed length. After computing the scores we normalize those scores. For this purpose, we sort the scores in ascending order. Phoneme with the minimum score will get the probability estimate 1 and the rest according to their ranks:

$$probability(ph_i) = 1 - \frac{rank(ph_i) - 1}{numberPhonemes} \tag{4.1}$$

We replace the distances with the assigned probabilities. Finally, we will have a 3-dimensional space: the 1st dimension is for the codebook/phoneme, the 2nd is the length of the possible phoneme, and the 3rd is the starting frame in the input speech.

Phonetic matching

The phonetic matching algorithm takes the previous matrices of probabilities constructed on the background speech and the keyword phonetic transcription as input. If the keyword is in another format we need to convert it to its phonetic transcription:

$$W = ph_1 ph_2 .... ph_n \tag{4.2}$$

The result of the phonetic matching for a start time (or frame) t_s is the estimated probability:

$$P(W, t_s) = \max\{ {}^{1}\!/\!_{n} \sum_{i=1}^{n} p(ph_i, t_{s_i}, l_i) \mid t_{s_1} = t_s, t_{s_i} = t_{s_{i-1}} + l_{i-1} \} \tag{4.3}$$

The length of a phoneme can vary between fixed values. Here $p(ph_i, t_{si}, l_i)$ is the probability that the phoneme ph_i of length l_i from the keyword phonetic transcription appears at the time/frame position t_{si} in the background speech. This probability is extracted from the computed values at the preprocessing stage. To compute $P(W, t_s)$ we use recursive definition, which can be solved by exhaustive search using dynamic programming in the space of all solutions, see Figure 4.3. The time complexity of the algorithm is $O(n^2\beta^2)$.

PHONETIC_MATCHING(α, β, W, t_s): $P(W, t_s)$

α = minimum number of frames per phoneme (e.g. 1)
β = maximum number of frames per phoneme (e.g. 12)
W = keyword phonetic transcription, length of n phonemes
t_s = start frame position in background speech
R = cumulative probability matrix

$$R_{1,j} = p(ph_1, t_s, j),\ j = \overline{\alpha, \beta}$$

$$R_{i,j} = \max\{R_{i-1,k} + p(ph_i, t_s + k + 1, j - k) \mid$$

$$k \geq \alpha(i-1),\ j - k \leq \beta, k < j\}, i > 1$$

$$P(W, t_s) = {}^{1}\!/\!_{n} \max\{R_{n,j} \mid j = \overline{n\alpha, n\beta}\}$$

Figure 4.3: Problem definition of the phonetic pattern matching.

4.3 Experimental setup

All the experiments are carried out on a notebook with the configuration: 1.4 GHz, 512Mb RAM, Windows XP SP2.

Speech database

We evaluate our system using the TIMIT corpus (Table 4.1), which consists of two parts: train data and test data. The acoustic models are trained on the train data excluding the SA1 and SA2 files.

Table 4.1: Structure of the TIMIT corpus.

TIMIT	TRAIN	TEST
No. of sentences	3696	1344
No. of uttered words	30132	11025
No. of distinct words	4891	2373
No. of male speakers	438	112
No. of female speakers	192	56
No. of dialect regions	8	8
Total speech time	~188 minutes	~69 minutes

Feature extraction

In speech recognition usually the *cepstral features* are used (LPC, MFCC). In our experiments we will see the effect of MFCCs and we will investigate if the *formants* (pitches) could also be used as features in keyword spotting. The formants as resonant frequencies of the vocal tract characterize the vocal tract shape in the moment of sound pronunciation.

For formants extraction we use *Burg's algorithm* [Press1992] as implemented in *PRAAT* application [Boersma2006]. We set the time step to 10 ms, the maximum number of formants at 5 for robust estimation of the first 3 formants used, the maximum formant to 5500Hz, the window length to 25 ms and the pre-emphasis is done starting 50 Hz.

To extract MFCCs we use the same PRAAT application [Boersma2006] with the following settings: 10 ms time step, 15 ms window, 100 mel as position of the first filter and 100 mel distance between filters between in the filterbank. We use only the first 12 MFCCs without the energy coefficient.

Acoustic models training

In training phase, the feature vectors (MFCCs or formants) are extracted from the training material and grouped for each phoneme according the phonetic transcriptions. Clustering is performed on the extracted samples from training material for each phoneme. The acoustic models (codebooks) are the result of the clustering step and they contain a number of

representative training samples. Clustering methods and measures were presented in the previous chapter.

Keyword selection

We choose a large set of keywords that permits a good evaluation of the system and brings relevance to the tests. In order to obtain the set of keywords we do the following. Firstly, we use only test words with at least 4 letters and that are not substrings of other words in the speech material. After that we dismiss those that have more than one or none occurrences in the entire test material. The final set has 862 keywords as summarized in Table 4.2. The word is defined as an *in-vocabulary* word (INV) if it occurs in the training material, and as an *out-of-vocabulary* word (OOV) if it appears only in the testing material. The notion of vocabulary is relative to the training/testing material, but in order to demonstrate the independence from context and training material of the system and its results, this division is necessary.

Table 4.2: Information about keywords selected for testing.

	Number of keywords	Average length (phonemes)
In-vocabulary (INV)	238	7.03
Out-of-vocabulary (OOV)	624	7.96
All keywords (INV+OOV)	862	7.7

Evaluation criteria

For testing our system, we consider a positive match as the case when the computed score corresponds to the starting position of the keyword, or the score is computed for a possible word that ends at the keyword ending position and its length is between 90% and 120% of that of the keyword. The rest of the matches are considered as negative matches. We use the transcriptions for information about the keyword occurrence. After computing the best scores for each start frame, we split the entire set of scores into segments of 8 times of the number of phonemes in the keyword. This is chosen to match the average number of frames per phonemes in the training material.

We use *Receiver Operating Characteristics (ROC)* curves [Fawcett2004] for evaluating the accuracy of the method. The basic ROC graph plots the pairs of *true positive rates* (*tpr*) and *false positive rates* (*fpr*) at different thresholds. The true positive rate is defined as the number of correctly classified positives divided by the total number of positives, and the false positive rate (*fpr*) is defined as the number of incorrectly classified negatives divided by the total number of negatives.

$$tpr = \frac{positives \quad correctly \quad classified}{total \quad positives} \tag{4.4}$$

$$fpr = \frac{negatives \quad incorrectly \quad classified}{total \quad negatives}, \tag{4.5}$$

As the first evaluation criterion, we measure the word-spotting accuracy by *Figure of Merit* (FOM) as proposed in [Rohlicek1989] and then defined by NIST (National Institute of Standards and Technology) as the average of the detection/false-alarm curve over the range [0..10] *fa/kw/h* (*false alarms per keyword per hour*). For each keyword in the test material, we use linear interpolation for a lower approximation of FOM so that FOM equals to 0 at 0 fa/kw/h if the keyword is not spotted with 0 false alarms. The ROC curve is plotted for detection rate or *tpr* and number of fa/kw/h. The FOM for one set of keywords will be represented by the average value of all keyword FOMs computed separately.

The second evaluation criterion we used is *Equal Error Rate (EER)*. It is defined as the error rate at the point where the false rejection rate (1 - *tpr*) and the false acceptance rate (*fpr*) are equal. The EER for one set of keywords will be represented by the average value of all keyword EERs computed separately.

4.4 Results

Comparison of the features

For the comparison, we fixed the minimum phoneme length at 1 frame and the maximum at 12 frames. The acoustic models are computed using a random clustering algorithm. We set the codebook size to 5. The keyword test set is obtained by random selection of 34 keywords from the test material. We use the first three formants (F1, F2, F3) and the first 12 MFCC coefficients.

Table 4.3: Comparison of the formants and the MFCC features.

Similarity measure	FOM(%)		EER(%)	
	Formants	MFCC	Formants	MFCC
PCD	10.42	42.77	17.22	4.39
DTW	12.84	39.72	12.21	5.22

We observe a big gap between the results obtained using formants and those with MFCC (see Table 4.3). Using MFCC, the keyword spotting system performs almost three to four times better than using formants (see Figure 4.4). The results indicate that the formants are not sufficient for accurate keyword spotting but wider representation of the spectrum, such as MFCC features, should be used.

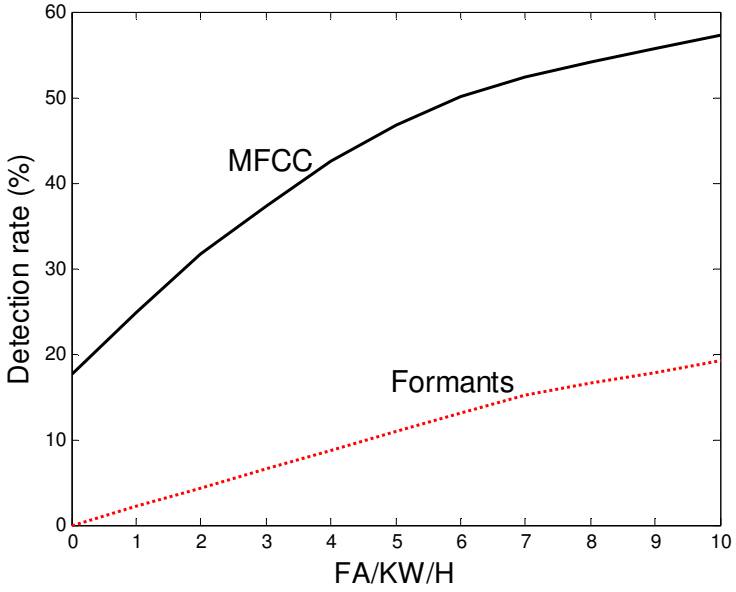


Figure 4.4: Comparative word detection results (ROC curve) between the formants and MFCC, for 34 keywords set using PCD.

Comparison of design parameters

For the tests, we use the first 12 MFCC coefficients and the set of 862 keywords selected previously. Each codebook contains 5 phoneme samples.

The first experiment aims at finding out a proper maximum phoneme length setting for the phonetic matching algorithm. From Figure 4.5 we can see that the FOM reaches the maximum (45.58%) for maximum phoneme length of 12. The best EER (3.02%) is achieved for maximum phoneme length of 13, as seen in Figure 4.6.

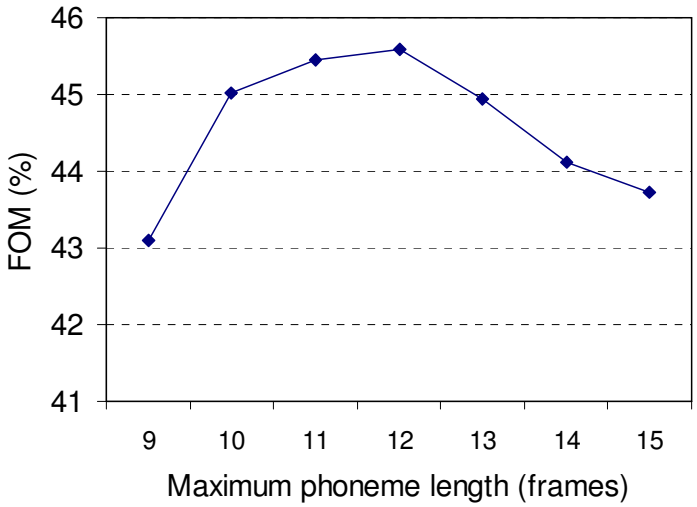


Figure 4.5: FOM variation of the matching algorithm as a function of the maximum phoneme length. The entire set of keywords and the DTW distance are used.

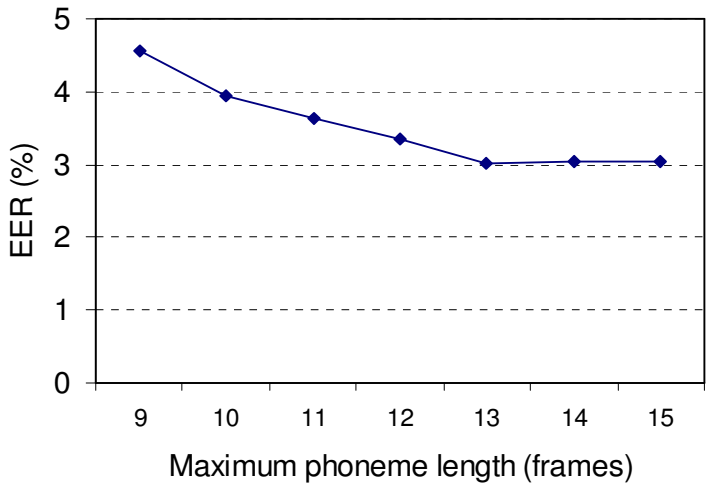


Figure 4.6: EER variation of the matching algorithm as a function of the maximum phoneme length. The entire set of keywords and the DTW distance are used.

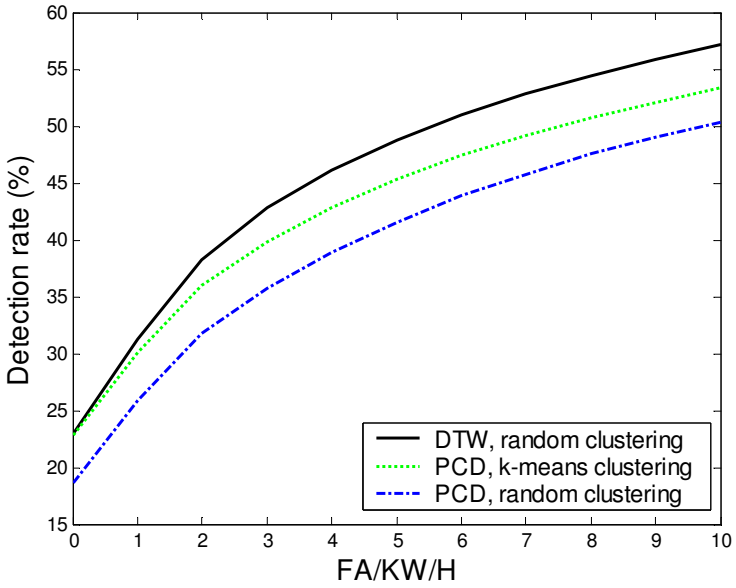


Figure 4.7: Comparison (ROC curve) between different clustering and similarity measures in terms of FOM. The entire set of keywords and the maximum phoneme length of 12 are used.

From Figure 4.7 we can see that our word spotting system performs better with the DTW distance than with the PCD distance in terms of higher FOM. A lower EER is also achieved with the DTW distance (Table 4.4). The main observation here is that the better codebooks are obtained the better will be the matching results. The codebooks were computed with the time and iterations constraints set up so that the processing time would not exceed 10 hours.

Table 4.4: Search accuracy (FOM) and equal error rates (EER) of the different clustering methods and similarity measures. The entire set of keywords and the maximum phoneme length of 12 are used.

	PCD K-means clustering	PCD Random clustering	DTW Random clustering
FOM (%)	42.68	38.99	45.58
EER (%)	3.98	4.57	3.34

The performance of our word spotting system is directly influenced by the length of the keywords (measured as the number of phonemes), see Figures 4.8 and 4.9. The longer the keywords the better is the performance in terms of both FOM and EER. This trend is more clearly visible when more data is used (OOV and INV+OOV) whereas the smaller INV set with only few keywords in the length range [10…13], which explains the high variation in the results in this part. From these graphics we can also conclude a less significant difference between INV and OOV keywords performances, than is shown in the Table 4.5 where due to the relatively reduced number of keywords and unbalance in keywords' lengths over the INV and OOV sets is a big difference in favour of OOV keywords.

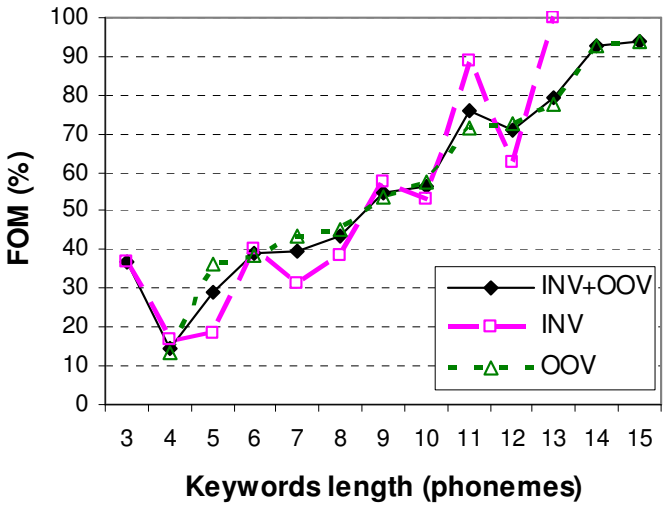


Figure 4.8: FOM variation of the matching algorithm as a function of keywords' length in phonemes for INV, OOV and combined INV+OOV keywords using the DTW distance and the maximum phoneme length of 12.

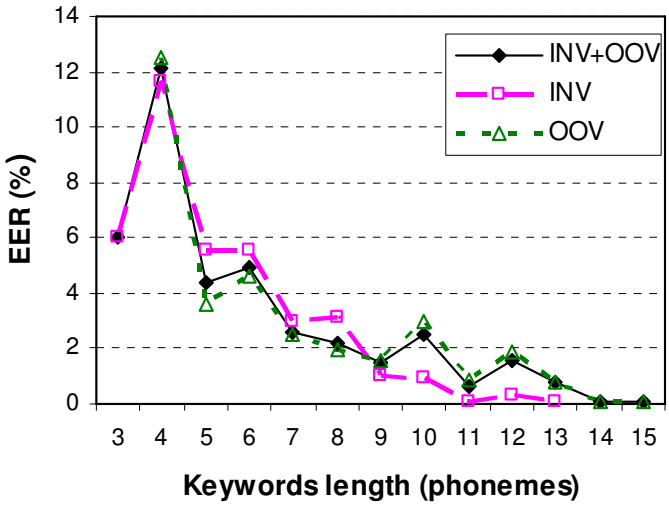


Figure 4.9: EER variation of the matching algorithm as a function of keywords' length in phonemes for INV, OOV and combined INV+OOV keywords using the DTW distance and the maximum phoneme length of 12.

Table 4.5: Comparison between word spotting results for in- and out-of-vocabulary keywords separately and jointly. The DTW distance and maximum phoneme length of 12 are used.

Keywords	FOM (%)	EER (%)
INV	38.62	4.17
OOV	48.23	3.02
INV+OOV	45.58	3.34

The best results

The results shown here are published in [Timofte2006]. The best result in terms of FOM (45.58%) is obtained with the DTW distance and a maximum phoneme length of 12 frames (120 ms in our case). The corresponding EER is less than 3.34%.

More recent experiments based on the introduction of Extended Euclidean Distance (EED) along with k-medoids clustering bring substantial improvements in performance in the conditions of a decreased number of representative samples per codebook being 4 instead of 5.

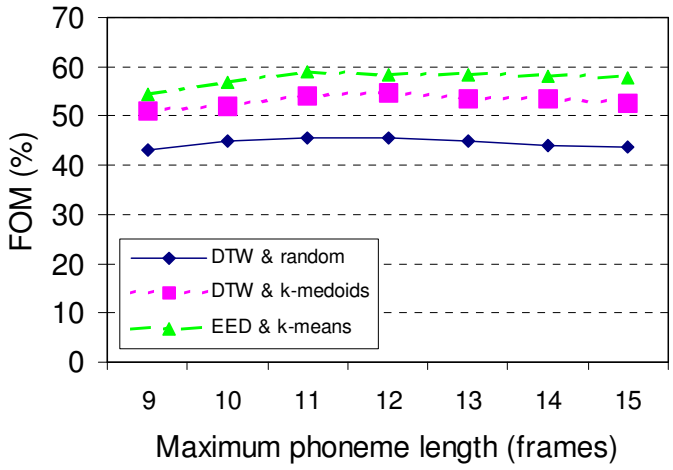


Figure 4.10: FOM variation of the matching algorithm as a function of the maximum phoneme length. Comparison of DTW with random clustering, DTW with k-medoids and EED with k-means clustering. The entire set of keywords is used.

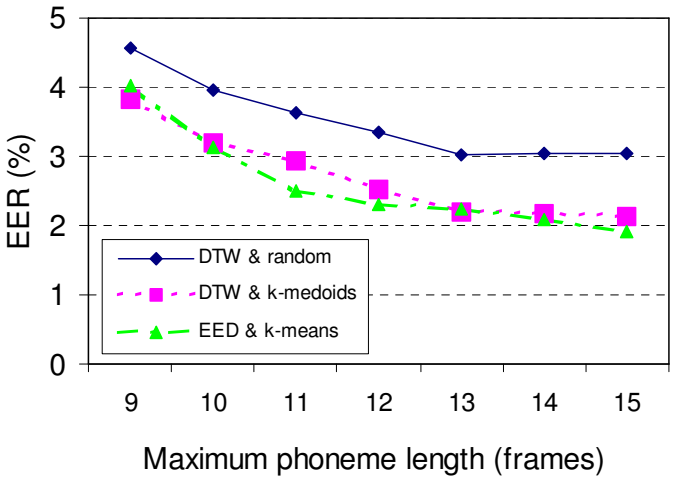


Figure 4.11: EER variation of the matching algorithm as a function of the maximum phoneme length. Comparison of DTW with random clustering, DTW with k-medoids and EED with k-means clustering. The entire set of keywords is used.

In Figures 4.10 and 4.11, the results for DTW and random clustering are the ones computed from the previous experiments with a codebook size of 5 samples, and the other results for DTW and EED are computed for a size of 4 samples per phoneme codebook.

The clustering method is a critical part for the entire system; better clustering method provides better performance of the overall system (Figure 4.10). DTW and k-medoids reach a FOM of 54.58% for 12 frames maximum length for the phonemes comparative with only 45.58% for DTW with random clustering. The improvement is significant, but the newly proposed similarity measure EED brings superior performance in both FOM (58.99% instead of 54.58%) and EER (2.49% instead of 2.93%). The EER decrease with the maximum length

setting for phonemes and also FOM has a decreasing trend after the maximum reached for a maximum phoneme length of 12 frames.

5. APPLICATION TO VOICE ACTIVITY DETECTION

Voice activity detection (VAD) is a critical front-end component in various voice-based applications such as modern telecommunication, speech recognition, speech enhancement and speaker recognition. In forensic analysis, VAD can also be used to track and annotate the speech parts from large audio recordings [Tuononen2007]. The applicability of this tool is directly addressed in our experiments.

The goal of VAD is simply to classify a given sound frame either as speech or non-speech. The result of this decision can provide considerable savings in data processing by coding only the speech frames, power saving in mobile devices [Digital1998], co-channel interference reduction in mobile telephony [Beritelli2001], greater noise suppression in speech enhancement and significant improvement in recognition accuracy in automatic speech recognition or speaker recognition.

5.1 Background

For solving VAD, a considerable number of methods exist in literature. Simple ones are based on comparing the frame energy [Tong2006], periodicity [Hautamäki2007], zero crossing rate [ITU1996] and spectral entropy [Renevey2001] with a detection threshold to classify the audio segments into speech/non-speech. More complex approaches consider combinations of the previous ones: long-term spectral divergence measure [Ramirez2004], statistical hypothesis testing [Chang2006], low-variance spectrum estimation [Davis2006] amplitude probability distribution [Tanyer2000], Gaussian mixture models [Kay1998] and support vector machines [Kinnunen2007].

The *International Telecommunication Union* (ITU) standard G729 Annex B [ITU1996] uses the *zero-crossing rate* (ZCR), the full-band energy, the low-band (0-1 KHz) energy and the *linear predictive coding* (LPC) for the spectral envelope. The *European Telecommunications Standards Institute* (ETSI) standard AMR [ETSI1999] specifies two options for VAD. The first option makes computation of the signal levels in each band after passing the signal through a filterbank and taking the VAD decision based on *signal-to-noise ratio* (SNR), pitches, tone and a correlated complex signal analysis module. The option 2 is an enhanced version that is expected to be more robust against environmental noise.

Energy-based method first measures the energy of each frame in the file and then sets the speech detection threshold relative to the maximum energy level. This method includes parameters optimized on NIST datasets by Institute for Infocomm Research [Tong2006]. The *periodicity-based* method works under the assumption that the noise is aperiodic. The periodicity detector defined in the YIN pitch estimation algorithm [deCheveigne2002] is used in our experiments as described in [Hautamäki2007].

The *long-term spectral divergence* method [Ramirez2004] uses long-term spectral divergence to separate between speech and noise, and its parameters are set up as explained in [Tuononen2007]. The speech/non-speech decision rule is formulated by comparing the long-term

spectral envelope to the average noise spectrum. The noise model is initialized using the beginning part of each file. We also consider trained variant of the LTSD method, in which the parameters are tuned from the training material.

The concept of adapted GMMs is used in the *GMM-based* method [Kay1998]. A *universal background model* (UBM) of 256 diagonal-covariance Gaussian components is trained from the training data. To obtain the adapted speech and non-speech models we apply *maximum a posteriori* (MAP) adaptation [Gauvin1994, Reynolds 2000] of the mean vectors. The VAD decision is taken based on the log likelihood ratio computed by the fast *N*-top scoring algorithm [Kay1998].

A *content-based* approach analyzes whether the audio signal contains any recognizable speech using standard *automatic speech recognition* (ASR) component [Zhang2002]. The problem is that the language models should be known and trained in advance, and it would be vulnerable to the change of language and training conditions. Moreover, the ASR itself needs VAD as a front-end to make the speech recognition more reliable. A two-state *hidden Markov model* (HMM) was considered in [Pfau2001] so that one state was trained for speech and the other one for non-speech.

In this Chapter, we propose a content- and language-independent method for voice activity detection based on *short-term time series* (STS) of the audio signal based on the work presented in [Timofte2007]. We use *mel-frequency cepstral coefficients* (MFCC) as the basic features, and group the subsequent MFCC features to form so called *pseudo-phonemes*. The method exploits temporal information of the signal but without the need to know the exact speech content or the language spoken. The system needs to be trained using representative training samples annotated as speech/non-speech for the operating conditions. The VAD will then classify given input material frame-by-frame using the trained speech and non-speech models.

The proposed approach is also suitable for forensic analysis by doing annotation for a part of the material, and then using the model as an automatic tool for the rest of the material. In this way, each time the conditions from audio material change the training is performed and the only parameter to be set for the system will be the detection threshold. It is expected that the proposed method trained on a specific working conditions have robust behavior on further material from the same environment and to give better performance than most of the comparative VAD methods.

5.2 Voice activity detection based on short-term time series

Overall scheme

VAD can be seen as a classification problem with two classes: one for speech and another one for non-speech. In our approach, we train models for both classes and use them to take the appropriate decision in the classification phase.

In the proposed approach, we use sequences of feature vectors measured at successive time moments. This follows the idea in [Timofte2006] originally used for keyword spotting. A time series can be considered as *phoneme* if the sequence of feature vectors has a distinctive known phonetic interpretation. However, since phonetic significance of the time series is unknown, we refer the features as *pseudo-phoneme*; the phonetic level information is irrelevant for the task.

Short-term time series (STS) based VAD system can be divided into training phase and classification phase as in Figure 5.1. In the training phase, pseudo-phonemes are extracted from the audio material annotated at segment-level as speech or non-speech with 1-second resolution. Because it is unfeasible to maintain models with all possible samples for speech and non-speech, we reduce the number of possibilities and keep only the most representative ones by using a clustering algorithm. As a result, we will have two separate codebooks, one to model the speech sequences and another one to model the non-speech sequences. In the classification phase, likelihoods originate from the trained speech and non-speech models and are computed for every frame, and the VAD decision is then taken as the one with the higher likelihood.

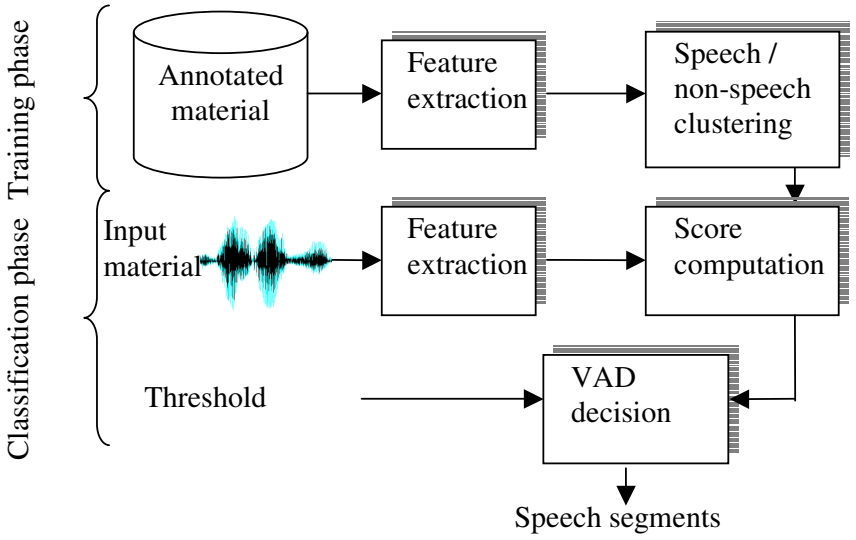


Figure 5.1: Structure of time series based VAD system.

Time series similarity measure

To operate with the time series, particularly with phonemes and pseudo-phonemes, we need to define and use a similarity measure. Since time series can be sequences of different lengths, commonly used distance metrics do not apply. We therefore use symmetric *dynamic time warping* (DTW) [Myers1981], an alignment method that has good accuracy, although quadratic time complexity described in Chapter 3.

Training phase

In the training phase, audio material should be annotated as speech/non-speech at segment level. The better the quality of the annotations for the material and the relevance of this for the testing conditions, the better will be the results. Based on the annotations, we extract samples for speech and non-speech as fixed length pseudo-phonemes (usually 100 ms or 100 frames/feature vectors) as shown in Figure 5.2, where we can see a speech part divided for speech samples and a non-speech part with a noise that provides samples for the non-speech model.

The pseudo-phoneme samples are the input for the clustering algorithm. We use a hierarchical clustering method followed by repeated *k-medoids* iterations for fine-tuning the

solutions. K-medoids is used since we use DTW as a similarity measure, and we do not have a good method for computing centroids as would be required in k-means. The codebooks consist of 100 representative samples; one codebook for the model of speech, and one for the model of non-speech used in the VAD classification. The clustering algorithms are described more detailed in chapter 3.

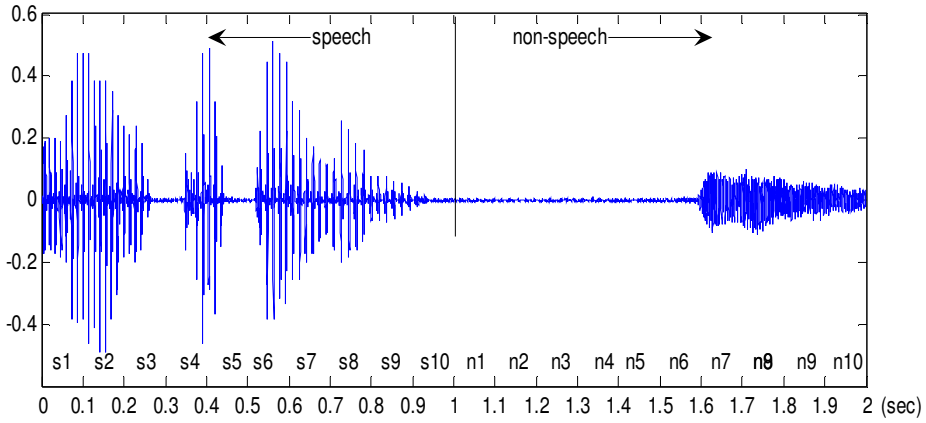


Figure 5.2: Pseudo-phonemes marked for extraction from a given annotated material at 1-second resolution as speech/non-speech.

Classification phase

In VAD classification, we calculate distances between the pseudo-phonemes centered on the current frame $ph_k = (f_{i-k+1}\ f_{i-k}\ \ldots\ f_{i+k-1})$ to the speech codebook *cb*. A frame f_i is classified as speech if the cumulated sum S_i of the distances is less than the corresponding distances to the corresponding non-speech codebook. Thus, we actually take the context of the frame into account in the decision by using the formula (5.1), where *L* is empirically set to 6, we consider the centred pseudo-phonemes with at most 11 frames/110 ms length:

$$S_i(cb) = \sum_{k=1}^{L} Dist(f_{i-k+1} f_{i-k} \ldots f_{i+k-1}, cb) \tag{5.1}$$

The distance between a sequence of feature vectors (pseudo-phoneme *ph*) and the given codebook *cb* is defined as in equation (5.2) where s_k are the representative pseudo-phoneme samples from the codebook. The distance between the two pseudo-phonemes is obtained by DTW.

$$Dist(ph, cb) = \min\{DTW(ph, s_k) \mid s_k \in cb\} \tag{5.2}$$

After having the decisions at frame level, we combine the decisions at 1-second level as follows. The number of speech frames is counted for each 1-second interval. If the proportion of the speech frames is higher than a given threshold (e.g. 50 %), the corresponding 1-second segment is considered as speech; otherwise it is labelled as non-speech. Furthermore, the 1-second neighbour segments are concatenated if they both are classified as speech. The concatenated segments are then output as the result of the VAD.

5.3 Experimental setup

Datasets and features

Experiments are conducted on three datasets whose main characteristics are listed in Table 5.1. The first one is a subset of the NIST 2005 speaker recognition evaluation corpus, consisting of conversational telephone-quality speech having a sampling rate of 8 kHz. We selected 15 files from different speakers for our purposes having duration of 5 minutes per file.

The "Bus stop" dataset consists of timetable system dialogues recorded in 8 kHz sampling rate [Turunen2005]. The material consists of human speech commands that are mostly very short, and synthesized speech that provides rather long explanations about bus schedules.

The "Lab" dataset consist of one long continuous recording from the lounge of our laboratory in 44.1 kHz, re-sampled at 8kHz. The goal of the material was to simulate wiretapping material collected by detectives. For more details on the "Bus stop" and "Lab" datasets, refer to [Tuononen2007].

Table 5.1: Datasets used in the experiments and their partitioning into training and testing sections

	NIST2005	Bus stop	Lab
Recording equipment	Telephone	Telephone	Labtec PC microphone
Training section	25 min (5 spk. × 5 min)	61 min	85 min
Test section	50 min (10 spk.× 5 min)	105 min	170 min
Speech-to-non-speech ratio	53%:47%	75%:25%	12%:88%

Each dataset has been manually annotated using a resolution of 1 second as described in [Tuononen2007]. We divide each dataset into non-overlapping training and test sections of 1/3 and 2/3, respectively. For the MFCC features, we use the frame length of 25 milliseconds and frame shift of 10 milliseconds. We use only the first 12 cepstral coefficients, without the energy coefficient.

Evaluation methodology

In forensic skimming, it is important to point out correctly the speech segments whereas results at the frame level are not very relevant. All our errors are therefore computed at segment level, where the segments are annotated using 1-second resolution.

Two types of errors can happen. A *miss* denotes a speech segment incorrectly classified as non-speech by VAD. A *false alarm* denotes a non-speech segment that is incorrectly classified

as speech. According to the application, the importance of the considered error measures varies. The operating point of the VAD system can be selected by fixing the threshold. A lower threshold reduces the number of missed speech segments at the cost of increasing the number of false alarms.

The detection error trade-off curve (DET) is used as an evaluation tool. The DET curve shows the *probability of miss* (P_{miss}) as a function of the *probability of false alarm* (P_{fa}) on a normal deviate scale. We use also the *equal error rate* (EER), which is the case when the two errors are equal, $P_{miss} = P_{fa}$.

For a better estimation of the differences between the methods in a realistic application scenario, we include two extreme operating points to minimize the probability of either miss or false alarm, in addition to the EER. We consider the case of $P_{miss} = 2\%$ (and $P_{fa} = 2\%$) by setting the threshold accordingly and measure the other error rate at this operating point.

Influence of parameters

In our approach, an important parameter is the length of the pseudo-phoneme that is used as the basic unit of computation. To investigate its effect on the detection accuracy, we run tests for lengths varying from 60 ms to 120 ms in the case of "Lab" material. The corresponding DET curves are plotted in Figure 5.3. We can see that the best behavior is obtained when the length of sequence is set to 100 ms (10 frames). This is the parameter value that will be used in the following comparative tests.

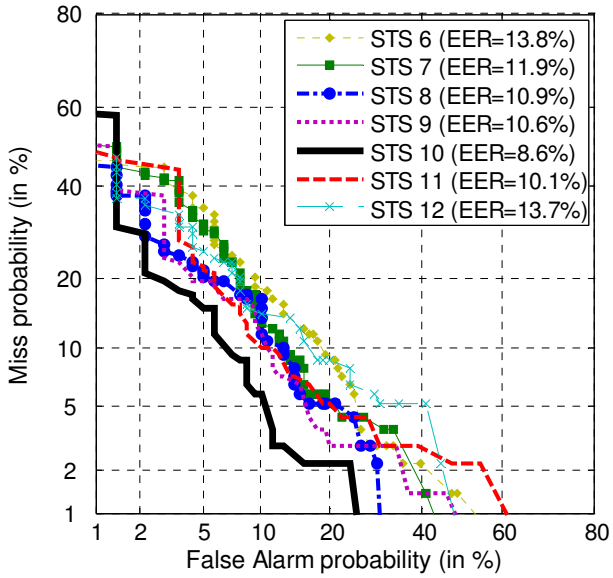


Figure 5.3: Comparative DET plots for STS trained using pseudo-phonemes of length varying from 6 to 12 frames.

5.4 Results

Table 5.2: Comparison of the EER (E_1), $P_{miss}@P_{fa}=2\%$ (E_2), $P_{fa}@P_{miss}=2\%$ (E_3) and the average performance for the VAD methods on the datasets used in experiments.

	VAD method	NIST 2005			Bus stop			Lab			Average error			
		E_1	E_2	E_3	E_1	E_2	E_3	E_1	E_2	E_3	E_1	E_2	E_3	All
Adaptive	Energy [Tong2006]	1.5	1.4	1.2	14.6	62.3	27.2	16.8	80.6	65.3	11.0	48.1	31.2	30.1
	LTSD [Ramirez2004]	40.0	40.0	62.5	19.2	100.0	36.0	14.4	76.8	48.6	24.5	72.3	49.0	48.6
	Periodicity [Hautamäki2007]	3.2	4.3	5.8	21.9	92.9	100.0	9.9	34.5	36.6	11.7	43.9	47.5	34.3
	G729B [ITU1996]	8.9	39.2	10.8	6.5	10.0	19.4	7.9	**18.4**	10.9	7.8	**22.5**	13.7	**14.7**
	AMR1 [ETSI1999]	5.5	27.4	8.0	5.7	10.3	17.4	7.2	38.6	49.5	**6.1**	25.4	25.0	18.8
	AMR2 [ETSI1999]	8.4	47.7	11.1	7.4	74.6	18.0	**5.1**	23.9	**10.4**	7.0	48.7	**13.2**	23.0
Trained	SVM [Kinnunen2007]	11.6	36.6	60.0	5.2	10.0	23.1	19.5	65.9	64.5	12.1	37.5	49.2	32.9
	GMM [Kay1998]	8.8	24.3	19.6	7.5	53.1	22.2	9.7	44.4	26.6	8.2	38.7	20.9	24.0
	LTSD [Ramirez2004]	**1.3**	**0.3**	**1.0**	6.2	28.7	28.4	14.9	83.7	55.0	7.5	37.6	28.1	24.4
	STS (proposed)	7.1	37.7	8.8	**3.9**	**5.3**	**7.7**	8.6	29.4	24.0	6.5	24.1	13.5	**14.7**

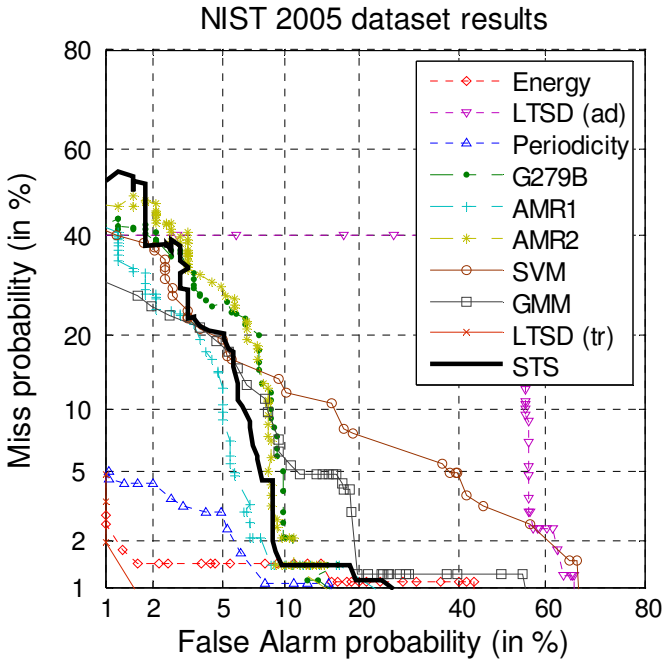


Figure 5.4: Comparative DET plots for each VAD method on NIST 2005 dataset.

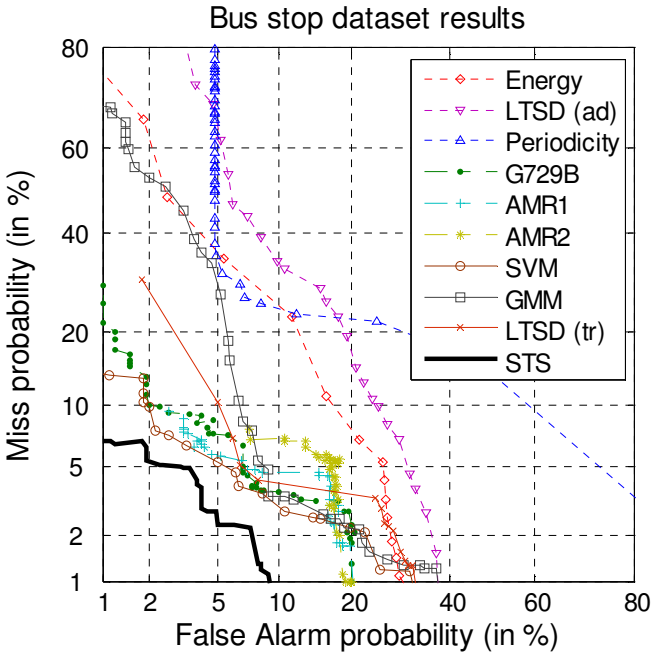


Figure 5.5: Comparative DET plots for each VAD method on Bus stop dataset.

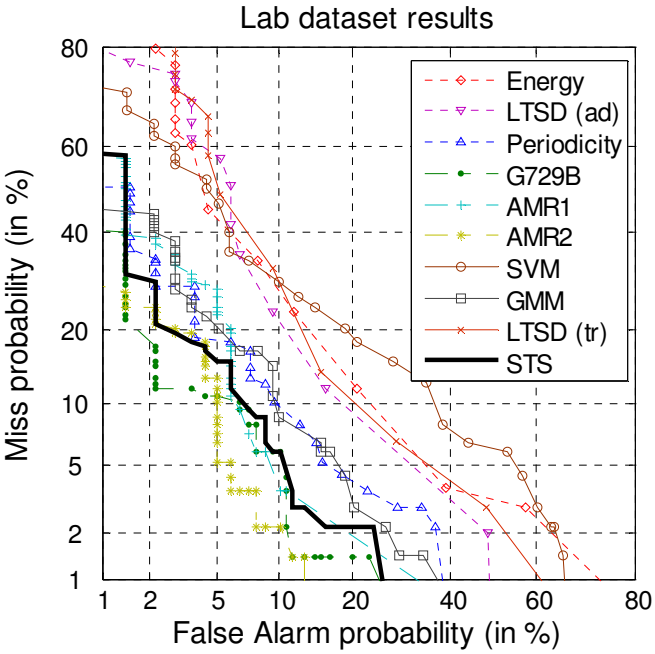


Figure 5.6: Comparative DET plots for each VAD method on Lab dataset.

Evaluation results

Comparison results of different VAD methods are shown in Table 5.2 and the corresponding DET plots for each dataset are shown through Figures 5.4 to 5.6. In order to have a DET curve for the comparative methods (G729B and AMR), we use the same approach as for STS: get the decisions at each frame and use the same thresholding method for concluding the result for the 1-second resolution by counting the proportion of speech frames in each segment.

The energy-based VAD and the trained LTSD that uses the energy provide the best results on the NIST dataset. This is not surprising since the parameters of the method have been optimized for earlier NIST corpuses through extensive testing. Moreover, the trained LTSD clearly outperforms the adaptive LTSD because the noise model initialization failed on some of the NIST files considering speech as non-speech in the beginning, and caused high error values.

In the case of Lab dataset, two methods perform well: G729B and AMR1, whereas the results of the proposed STS method are almost as good. The periodicity-based VAD gives good results on NIST and Lab datasets but fails on Bus stop recordings probably due to the periodicity of the noises closed to the speech. Trained LTSD VAD gives also bad results on Lab recording but performs best on the NIST dataset.

STS and AMR2 are both very good choices when one wants to keep the speech miss rate low and do not have many false alarms. This is the case in forensics application and in voice-dialogue systems. On the other hand, if one wishes to have a low false alarm rate (e.g. automatic speaker verification) the standard G729B method is slightly better than the proposed STS-based method.

The proposed time series based VAD clearly outperforms its competitors on Bus stop dataset. This could happen because of the temporal patterns that are modelled in the training phase and are also found in the testing material. It also performs well on Lab recording and gives good results on the NIST dataset. We can see that it exhibits a robust behaviour yielding, if not the best results, at least good enough for the purpose in all cases. The robustness across different datasets is the key advantage of the proposed method.

Conclusions

Voice activity detection using time series of MFCC features is proposed. The method works well when small miss rate is desired, which is the case in forensic skimming, for example. The main advantage of the proposed VAD is that it works consistently in the same manner with different corpora, and outperforms the other methods in most of our tests regardless the corpora.

The other methods, except the standards G729B and AMR, were more prone to the change of dataset and variations of their parameters. Our main conclusion is that, according to our experiments, STS-based VAD is easy to implement and adapts to the new dataset providing that we have a reasonable long training audio sample (usually 10 minutes is enough) from the recording environment; and that it provides better overall (average) performance than all the studied methods, except G729B for which the results are comparable.

6. APPLICATION TO SPEAKER IDENTIFICATION

In the speaker identification task, we search for the best match between a given unknown speaker and a set of speakers with known identities from a database. The database contains speech samples from the speakers. MFCC are used as features extracted from samples and further processed by clustering methods in order to create models for the speakers by retaining the representatives of the clusters. We study the role of short-term time series seen as sequences of feature vectors in speaker identification system. We want to find out if the use of short-term time series acoustic models brings improvements in the process over on single feature vectors and what are the influences of the length of the short-term time series and the size of the codebook in the process.

Speaker recognition research direction in speech processing is divided into two different tasks: *speaker identification* and *speaker verification*. Identification of a speaker from a known collection of speakers based on an audio recording is the task of speaker identification, while the verification of a claimed identity makes the object of speaker verification task. In those processes, it is usually not necessary to identify what the speaker says, but to extract the information that differs between speakers.

6.1 Background

Speaker recognition is the process of automatically recognizing who is speaking on the basis of individual information (bio-print) from the speech signals. It can be divided into two problems: speaker identification and speaker verification. In the speaker verification task the purpose is to establish if an identity of a claimer is the right one. In the speaker identification task the goal is to recognize the unknown speaker from a set of known speakers. The speaker identification task can be further classified into text-dependent and text-independent if in the process the text embedded into speech samples must match or not. This chapter is focused on the text-independent speaker identification task. More, since we have a database of finite number of speakers' models we are in the situation of identification from a closed set.

For speaker identification system the input is a sampled speech data while the output should be the identified speaker. The main parts of a speaker recognition system (see Figure 6.1) are: feature extraction, speaker modelling and the matching algorithm. The purpose of the feature extractor is to derive a set of speaker specific vectors from the input signal. The speaker model is constructed having these vectors for each speaker. The matching performs the comparison between two speaker models, or between the models and feature vectors extracted from the input signal.

Gathering information from the speech signal that captures most of the speaker's individuality is very complex. The individual speaker's characteristics occur at the lexical, segmental and prosodic levels [Rose2002, Weber2002]. Use of different word patterns is an example of lexical level individuality. At segmental level, the differences occur in phoneme realization as result of the speaker physiology of the speech apparatus. The prosodic characteristics are reflected by the use of pitches, stress and timing. The feature extraction part of

the speaker recognition system is responsible with measuring the *acoustic parameters* or *features* from the speech signal in order to capture the speaker characteristics. In the automatic speaker recognition task the commonly used features are MFCC, LPCC [Campbell1997], line spectral frequencies [Nguyen2003], subband processing [Kinnunen2002], and dynamic cepstral parameters [Soong1988]. The cepstrum carries only one source of evidence and for achieving better recognition accuracy other supplementary information sources should be used.

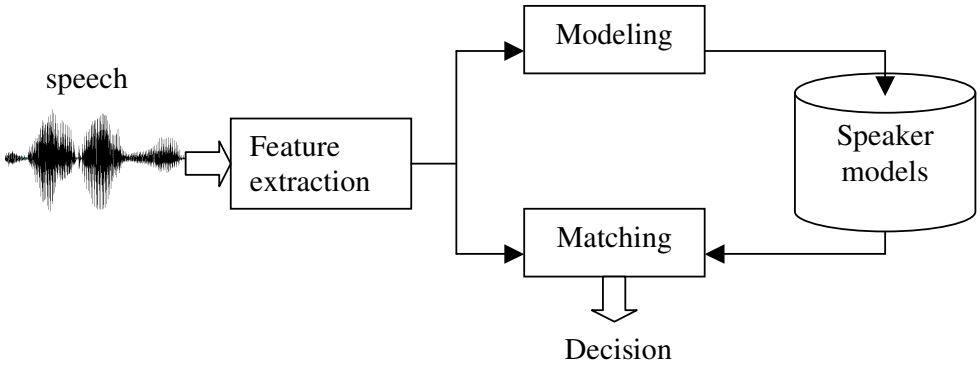


Figure 6.1: Speaker recognition system schema.

A well-known data fusion strategy is to concatenate the cepstral vector with the dynamic coefficients (delta and delta-delta) into a long feature vector [Campbell1997]. Also, the fundamental frequency could be added. Vector concatenation is also known as *classifier input fusion* [Slomka1998]. Another way of performing data fusion is to combine different classifiers, known as *classifier output fusion*. Feature concatenation for speaker identification along with *linear discriminant analysis* (LDA) is studied in [Zilca2000].

Typical approaches in speaker recognition that uses the segmental (phonemic) information from the speech signal are HMMs [Parthasarathy1996], artificial neural networks (ANNs) [Yiu2002], GMM, SVM [Campbell2003] or combinations of those in phoneme/speaker modeling and decision taking.

In speaker modeling, we can distinguish two main directions for estimating the speaker dependent feature distributions: *parametric* (stochastic) and *non-parametric* (template) [Fukunaga1990, Campbell1997, Duda2000]. In the parametric approaches, the aim is to fit known type distribution to the training data and to find the best parameters of the distribution that maximize/minimize some quality criterion. The non-parametric approaches, on the other hand, make minimal assumption about the distribution of features.

The pattern matching consists of computing similarity scores between the unknown speaker's feature vectors and all the speaker models available. The similarity measure depends on the type of the speaker model.

In this work, we study the role of the pseudo-phonemes as sequences of feature vectors in opposition to the widely approach to use single feature vectors individually. For this we consider one popular non-parametric approach to text independent speaker recognition, vector quantization (VQ) in a typical VQ-based system [Soong1987]. We want to find out what is the impact of the codebook size, the pseudo-phoneme length and the similarity measures used in the recognition accuracy of the identification system.

6.2 Speaker identification system

The structure of our VQ-based closed set speaker identification system is shown in Figure 6.2. The identification process is split into training and recognition (identification) phases. In the training, a model (a VQ codebook in our case) is created for each speaker starting from his or her speech samples. In the recognition phase, the speech samples are analyzed to find best matching models among the known previously trained speakers.

The extracted feature vectors from speech data are used to create speaker models through clustering. The purpose is to reduce the amount of data by locating the clusters in the feature space and keeping only relevant representative samples. To formalize: given a set of feature vectors (X) as input output a *codebook* (C). The codebook consists of the centroids of the clusters, called *code vectors*. The number of code vectors represents the codebook size. The codebook obtained is the speaker model and approximate the distribution of his feature vectors in the feature space.

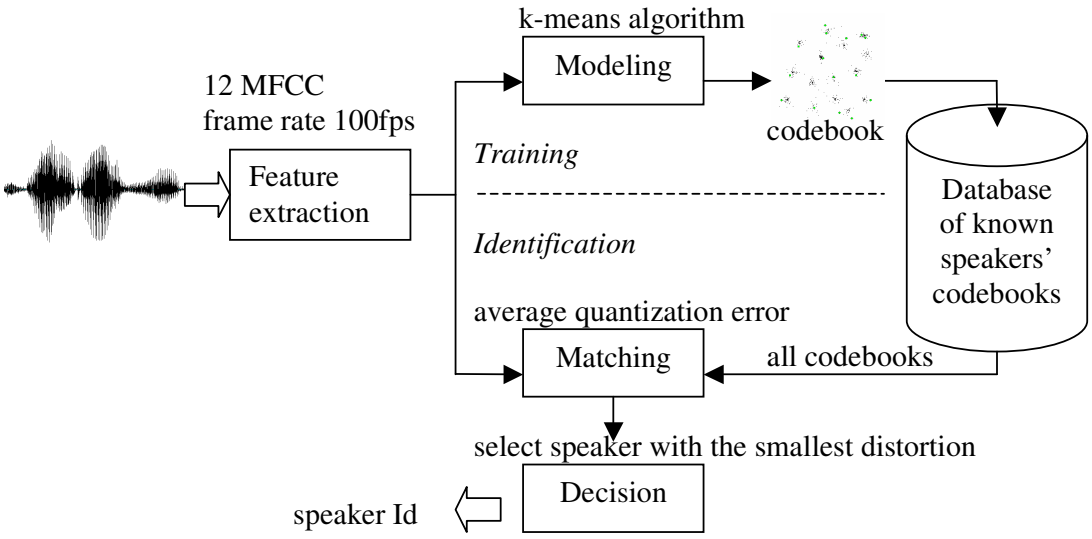


Figure 6.2: Structure of the VQ-based closed set speaker identification system.

In the recognition phase, the identification procedure is as follows:

1. Extract the set of feature vectors $X=\{x_1, x_2,\ldots, x_L\}$
2. Compute the distortion between the set of feature vectors extracted and all the speaker models (codebooks) C_i in the database:

$$D_i = d(X, C_i) \tag{6.1}$$

3. Identify the speaker that gives the smallest distortion:

$$Id = \arg\min_j \left(D_j\right) \tag{6.2}$$

The *distortion measure d* from the second step approximates the *dissimilarity* between the codebook $C_i = \{c_{i1}, c_{i2}, ..., c_{iK}\}$ and the vector set $X = \{x_1, x_2, \ldots, x_L\}$. We use the most used, intuitive distortion measure; for each vector in X, find the nearest code vector in C_i, compute the distances and take the average of the distances:

$$d(X, C_i) = \frac{1}{L} \sum_{j=1}^{L} \min_{k=1,K} d_E^2(x_j, c_{ik}) \tag{6.3}$$

where d_E is the Euclidean metric:

$$d_E(A, B) = \sqrt{\sum_{i=1}^{p} |a_i - b_i|^2}, A, B \in R^p \tag{6.4}$$

The distortion measure (6.3) is known as the *mean squared error* (MSE) and is also a measure for the quality of a codebook created from a given training set. The clustering problem of minimizing the MSE distortion is a NP-hard problem. In practice, approximation methods as k-means are used (see section 3.3.2) .The clustering of the feature vectors is necessary since it is infeasible to keep all the feature vectors and compute the distortion between those and the feature vectors extracted from the speech sample of an unknown speaker. Use of the full set of feature vectors as speaker model employs high memory consumption and the identification process takes too long when we have a large database of speakers to check.

6.3 Speaker modeling

The VQ-based speaker identification system is the base of our system. For short-term time series we introduce slight changes to the flow of the VQ process illustrated in Figure 6.2.

1. We replace the feature vectors by sequences of feature vectors of a fixed length. The order of the feature vectors in a sequence is important and follows the order in the speech data. Those are the pseudo-phonemes.
2. The distance between two pseudo-phonemes is EED or DTW (see section 3.2).
3. The clustering methods of section 3.3 use the distance between pseudo-phonemes and the result is a set of pseudo-phonemes called also codebook that represents the model of the speaker.
4. The distortion measure *d* of (6.3) becomes:

$$d(X, C_i) = \frac{1}{L} \sum_{j=1}^{L} \min_{k=1,K} dist(ph_j, c_{ik}), X = \{ph_1, ph_2, \ldots, ph_L\} \tag{6.5}$$

where *dist* is the distance between two pseudo-phonemes.

We notice that the flow of operations from the VQ-based speaker identification system remains mainly the same. The changes are in replacing the feature vectors by sequences of feature vectors (or pseudo-phonemes) of fixed length and the adaptation of the distance/similarity measures and clustering methods in order to handle the pseudo-phonemes as basic units.

We expect that the performance of the system would be better if we used instead of pseudo-phonemes, the phonemes spotted by a phoneme recognizer, and in this way, make use of better basic units with language content at the segmental level. The purpose of the presented

approach is to check if the short-term time series could be used in a simplified manner in text-independent speaker identification task.

6.4 Experimental setup

Speech material

The corpus used in our experiments is TIMIT, an American English corpus [Garofolo1993]. The corpus contains in total 630 American English native speakers from 8 dialect regions, of which 438 are males (70 %) and 192 females (30 %). For each speaker are 10 speech files. Two of the files ("sa" and "sx" files) have the same linguistic content for all speakers, the rest of the files being phonetically diverse. The corpus is noise free (53dB SNR) and has been recorded using a high-quality microphone. Speech files are stored in NIST/Sphere "wav"-file format with a sampling frequency of 16 kHz and a quantization resolution of 16 bits per sample. In our experiments the files were down-sampled at 8 kHz.

For the training phase, the speaker models are created using 7 files ("sa" and "sx" files) with 21.9 seconds length on average per speaker while for testing purposes 3 files ("si" files) are used with 8.9 seconds length on average.

In order to improve the results we drop the frames classified as non-speech by the standard voice activity detector G729B [ITU1996]. In this way, around 10% from the frames are ignored in our experiments.

Feature extraction, modeling and matching

In our experiments, we extract the first 12 MFCCs (without the energy coefficient) from a window of 25 ms with a frame shift of 10 ms (see section 2.3). The speaker models are generated by k-means algorithm (section 3.3.2) using standard or extended Euclidean distance. K-means is also applied for clustering the feature vectors in the special case of length 1, instead of sequences. K-medoids algorithm (section 3.3.3) is used when dynamic time warping is applied. The quantization distortion, equation (6.5), is our matching function where the distance is replaced with EED or DTW (see section 3.2).

Performance evaluation

The recognition accuracy of identification is measured by *identification error rate* (IER) and is defined by the number of incorrectly identified speakers reported to the total number of speaker identifications trials. For each speaker testing sample we match it against all 630 speaker models from the database and consider that it is identified correctly only if the distortion between the testing sample and the correct speaker model is the smallest one out of all 630 computed. All the experiments were carried out on a Dell Optiplex GX620 computer having a 3GHz processor and 1024 MB of RAM. The operating system is Windows XP SP2. All the methods were implemented using C/C++ languages.

6.5 Results

We used the following combinations: standard VQ-based system, the system adapted for short-term time series using extended Euclidean distance (EED), and the system adapted using dynamic time warping (DTW). We make the observation that when the pseudo-phoneme length is fixed to 1, the situation is assimilated with the case of the standard VQ-based system.

Baseline system

As baseline, we consider the standard case of a VQ-based closed set speaker identification system where the feature vectors (the pseudo-phoneme has fixed the length to 1 frame/feature vector) are clusterized using k-means and form the speaker models. The results of this approach are presented in the Table 6.1 and shows that the performance depends directly of the codebook size used in speaker modelling.

Table 6.1: Baseline system identification error rate (IER) variation with codebook size.

Codebook size	Identification error rate (%)
8	26.8
16	9.4
32	4.8
64	5.1

Influence of the parameters

In Figure 6.3, the variation of the identification error rate with the pseudo-phoneme length is illustrated when the codebook size is fixed to 32. The system based on EED performs much better than the system based on DTW, probably because the clustering algorithm in the case of EED uses k-means while the one in DTW uses k-medoids without having mean samples.

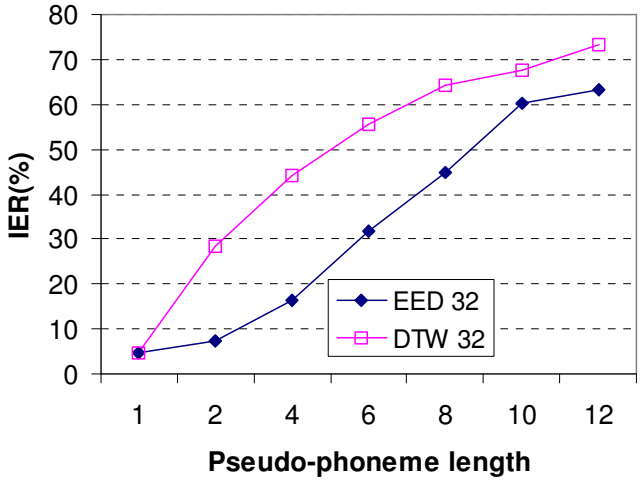


Figure 6.3: The influence of the length of the pseudo-phoneme on IER for the system using a codebook size of 32 in the case of EED and DTW.

The codebook size influences the results in an expected way like in the case of the baseline system (Table 6.1). The bigger the size of the codebook, the better the accuracy of the system. In Figure 6.4, the graphs for the system using EED and for the system using DTW are plotted when the pseudo-phoneme length is fixed at 2 frames. Again the performance of the EED-based system surpasses the one of the DTW-based system. We see that the performance decrease with the length of the pseudo-phoneme. The best performance is reached for the pseudo-phoneme length set to 1, which means that the baseline system provides the best results.

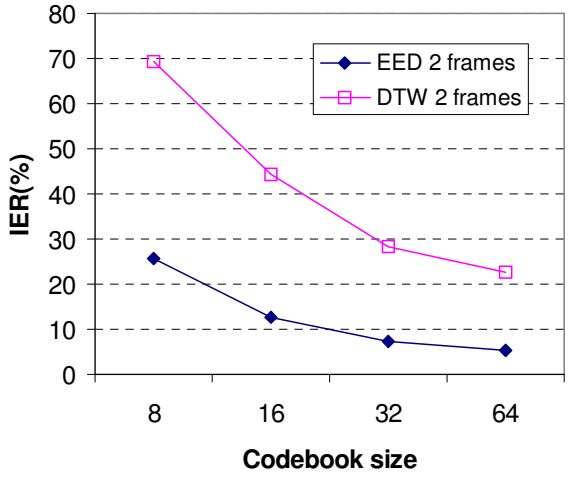


Figure 6.4: The influence of the codebook size on IER for the system using a length of the pseudo-phoneme of 2 frames in the case of EED and DTW.

In Table 6.2, are the IER for all the parameters combination considered in our tests. The pseudo-phoneme length is varied from 1 to 12 frames. In the case of the length 1 for the pseudo-phonemes the clustering method applied is k-means.

Table 6.2: The identification error rate (%) achieved in the different settings of the parameters. The results for DTW with pseudo-phoneme length set to 1 are the same as for EED and for the baseline system.

Pseudo-phoneme length (frames)	EED (repeated k-means) Codebook size				DTW (repeated k-medoids) Codebook size			
	8	16	32	64	8	16	32	64
1	26.82	9.36	4.76	5.08	26.82	9.36	4.76	5.08
2	25.56	12.70	7.46	5.24	69.21	44.29	28.41	22.70
4	31.11	19.84	16.51	17.14	74.60	57.94	44.29	37.78
6	41.59	36.35	31.91	32.38	76.67	67.62	55.56	49.52
8	52.22	45.56	44.76	45.87	83.65	73.33	64.13	60.32
10	60.79	58.41	60.32	61.75	84.44	76.83	67.46	65.08
12	65.71	66.03	63.18	69.37	86.83	80.00	73.18	63.97

Conclusions

In our experiments on TIMIT corpus, we found out that the short-term time series approach based on pseudo-phonemes of fixed length is not suitable for speaker modeling. The baseline system results based on vector quantization are better than the other results obtained by using pseudo-phonemes of 2 to 12 frames length (20ms-120ms). By increasing the codebook size the performance of the system improves. One reason for the poor performance of the short-term time series approach could be the small length of the training (21.9 seconds) and testing material (8.9 seconds) for each speaker. It is expected that short-term time series approach combined with a good phoneme recognizer will permit dealing with pseudo-phonemes/phonemes of variable length that reflects better the speech signal nature and in consequence will give better results. Also, the use of dynamic coefficients along with MFCCs could improve the performance.

7. CONCLUSIONS

In this thesis, we addressed specific speech processing tasks, namely keyword spotting, voice activity detection and speaker identification. The key aspect of the thesis is the introduction of the concept of short-term time series (pseudo-phonemes) as acoustic units. We cover theoretical aspects of speech signal processing science with emphasis on the elements used in our applications. Also, we set the usage details for the proposed short-term time series and we check the validity of the model by applications.

For keyword spotting we used short-term time series in phoneme modeling and we came up with solutions in matching a phonetic transcription as word pattern against a speech signal. The good results (FOM 58.99%, EER 2.49%) obtained in our experiments gives us a good starting point for future improvements. Comparison results on the same benchmark setting with the state-of-art methods are on our short-term plan.

Voice activity detection using short-term time series is proposed. We create models for speech and non-speech from the training material by making use of the pseudo-phonemes. We directly addressed the applicability in forensics. The method works well when small miss rate is desired, which is the case in forensic skimming, for example. The main advantage of the proposed VAD is that it works consistently in the same manner with all three different corpora tested, and outperforms the other methods in most of our tests regardless the corpora.

Speaker modeling using pseudo-phonemes of fixed length has been found to have poor results on speaker identification task on TIMIT corpus. The VQ-based speaker identification system provides better results than the proposed one. This is mainly due to the lack in covering the main characteristics embedded at the segmental level of the speech and too few training material per speaker. It is expected that short-term time series in combination with a phoneme recognizer will work with phonemes that cover better the speaker individuality and, thus, will lead to better recognition accuracy.

The short-term time series model showed its potential in keyword spotting and voice activity detection tasks while in speaker recognition there is room for further improvements.

REFERENCES

[Beritelli2001] F. Beritelli, S. Casale and G. Ruggeri, "Performance evaluation and comparison for ITU-T/ETSI voice activity detectors", in *Proc. IEEE ICASSP*, vol.3, pp. 1425-1428, Salt Lake City, UT, 2001.

[Boersma2006] P. Boersma and D. Weenink, "Praat: doing phonetics by computer" (Version 4.4.13)[Computer program]. Retrieved March 8, 2006, from http://www.praat.org/

[Campbell1997] J. Campbell, "Speaker Recognition: A Tutorial", in *Proc. of the IEEE*, 85(9), pp. 1437-1462, 1997.

[Campbell2003] W. M. Campbell, J. P. Campbell, D. A. Reynolds, D. A. Jones and T. R. Leek, "Phonetic Speaker Recognition with Support Vector Machines", in *Proc. Neural Information Processing Systems Conference in Vancouver*, British Columbia, pp. 1377-1384, 2003.

[Chang2006] J.-H. Chang, N.S. Kim and S.K. Mitra, "Voice Activity Detection Based on Multiple Statistical Models", *IEEE Trans. Signal Processing*, vol.54(6), pp. 1965-1976, June 2006.

[Davis2006] A. Davis, S. Nordholm and R. Togneri, "Statistical Voice Activity Detection Using Low-Variance Spectrum Estimation and an Adaptive Threshold", *IEEE Trans. Speech and Audio Processing* vol. 14(2), pp.412-424, March 2006.

[deCheveigne2002] A. de Cheveigne and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music", *The J. of Acoustical Society of America*, vol. 111, no.4, pp. 1917-1930, April 2002.

[Deller2000] J.R. Deller, J.H.L. Hansen and J.G. Proakis, *Discrete-time processing of speech signals*, IEEE Press, 2nd edition, New York, 2000.

[Digital1998] Digital Cellular Telecommunications Systems (Phase 2+); Adaptive Multi Rate (AMR); Speech Processing Functions; General Description, 1998.

[Duda2000] R. Duda, P., Hart and D. Stork, *Pattern Classification*, second ed. Wiley Interscience, New York, 2000.

[Dudley1939a] H. Dudley, "The Vocoder", *Bell Labs Record*, vol. 17, pp. 122-126, 1939.

[Dudley1939b] H. Dudley, R.R. Riesz and S.A. Watkins, "A synthetic speaker", *J. Franklin Institute*, vol. 227, pp. 7390764, 1939.

[Ephraim1992] Y. Ephraim, "Statistical-model-based speech enhancement systems", in *Proc. IEEE*, vol. 80, no. 10, pp. 1526-1555, 1992.

[ETSI1999] ETSI, "Voice Activity Detector (VAD) for Adaptive Multi-Rate (AMR) Speech Traffic Channels", ETSI EN 301 708 Recommandation, 1999.

[Fawcett2004] T. Fawcett, ”ROC Graphs: Notes and Practical Considerations for Researchers”, *Machine Learning*, 2004.

[Foote1999] J. T. Foote, “An Overview of Audio Information Retrieval”, *Multimedia Systems*, vol. 7 no. 1, pp. 2-11, ACM Press/Springer-Verlag, 1999.

[Fukunaga1990] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, second ed. Academic Press, London, 1990.

[Furui2000] S. Furui, *Digital speech processing, synthesis, and recognition - Second edition, revised and expanded*, Marcel Dekker, Inc., New York, 2000.

[Garofolo1993] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren and V. Zue, *TIMIT Acoustic-Phonetic Continuous Speech Corpus,* Linguistic Data Consortium, Philadelphia, USA, 1993.

[Garofolo2000] J. Garofolo, “TREC-9 Spoken Document Retrieval Track. National Institute of Standards and Technology”, http://trec.nist.gov/pubs/trec9/sdrt9_slides/sld001.htm (2007.08.23)

[Gauvin1994] J.L. Gauvin and C.H. Lee, ”Maximum-a-posteriori estimation for multivariate Gaussian observations of Markov Chains”, *IEEE Trans. Speech and Audio Processing*, vol. 2, no. 2, pp. 291-298, 1994.

[Gold1999] B. Gold and N. Morgan, *Speech and audio signal processing*, Wiley Press, 1999.

[Grebenskaya2005] O. Grebenskaya, “Speaker clustering in speech recognition”, Master’s thesis, University of Joensuu, 2005.

[Harringtion1999] J. Harrington and S. Cassidy, Techniques in speech acoustics, Kluwer Academic Publishers, Dordrecht, 1999.

[Hautamäki2007] V. Hautamäki, M. Tuononen, T. Niemi-Laitinen and P. Fränti, “Improving Speaker Verification by Periodicity Based Voice Activity Detection”, in *Proc. SPECOM*, vol. 2, pp. 645-650, Moscow, Russia, 2007.

[Huang2001] X. Huang, A. Acero and H.-W. Hon, *Spoken language processing*, Prentice Hall, Upper Saddle River, New Jersey, USA, 2001.

[Huang1990] X. Huang, K.-F. Lee and H.-W. Hon, “On semi-continuous hidden Markov modeling”, in *Proc. ICASSP*, vol. 2, pp. 689-692, Albuquerque, USA, 1990.

[ITU1996] ITU, “A Silence Compression Scheme for G.729 Optimized for Terminals Conforming to Recommendation V.70”, *ITU-T Recommandation G.729-Annex B*, 1996.

[Jilek2004] C. Jilek, J. Marienhagen and T. Hacki, “Vocal stability in functional dysphonic versus healthy voices at different times of voice loading”, *Journal of voice*, vol. 18, no.4, pp. 443-453, 2004.

[Juravsky2000] D. Juravsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing*, Computational Linguistics and Speech Recognition, Prentice Hall, San Francisco, USA, 2000.

[Kay1998] S.M. Kay, *Fundamentals of Statistical Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1998.

[Kaufman1987] L. Kaufman and P.J. Rousseeuw, "Clustering by means of medoids", *Statistical Data Analysis based on the L1 Norm*, Elsevier, pp. 405-416, 1987.

[Kinnunen2002] T. Kinnunen, "Designing a Speaker-Discriminative Adaptive Filter Bank for Speaker Recognition", in *Proc. ICSLP 2002*, pp. 2325-2328, Denver, USA, 2002.

[Kinnunen2003] T. Kinnunen, "Spectral features for automatic text-independent speaker recognition", Licentiate's thesis, University of Joensuu, 2003.

[Kinnunen2007] T. Kinnunen, E. Chernenko, M. Tuononen, P. Fränti and H. Li, "Voice Activity Detection Using MFCC Features and Support Vector Machine", in *Proc. SPECOM*, vol. 2, pp. 556-561, Moscow, Russia, 2007 (accepted).

[MacQueen1967] J.B. MacQueen, "Some methods for classification and analysis of multivariate observations", in Proc. 5th Berkley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, pp. 1:281-297, USA, 1967

[Myers1981] C.S. Myers and L.R. Rabiner, "A comparative study of several dynamic time-warping algorithms for connected word recognition", *The Bell system Technical Journal*, 60(7), pp1389-1409, September 1981.

[Nguyen2003] P.C. Nguyen, M. Akagi, T.B. Ho, "Temporal Decomposition: A Promising Approach to VQ-Based Speaker Identification", in *Proc. ICASSP 2003,* Hong Kong, 2003.

[Parthasarathy1996] S. PArthasarathy and A. E. Rosenberg. "General phrase speaker verification using sub-word background models and likelihood-ratio scoring", in *Proc. ICSLP'96*, vol. 4, pp. 2403-2406, Philadelphia, USA, 1996.

[Pfau2001] T. Pfau, D. Ellis and A. Stolcke, "Multispeaker Speech Activity Detection for the ICSI Meeting Recorder", in *Proc. ASRU*, Italy, December 2001.

[Press1992] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C: the art of scientific computing*, Second Edition, Cambridge University Press, 1992.

[Rabiner1993] L.R. Rabiner and B.W. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1993.

[Rabiner1983] L.R. Rabiner, S.E. Levinson and M.M. Sondhi, "On the application of vector quantization and hidden Markov models to speaker-independent isolated word recognition", Bell Syst. Tech. J., vol. 62, no. 4, pp. 1075-1105, 1983.

[Ramirez2004] J. Ramirez, J.C. Segura, C. Benitez, A. de la Torre and A. Rubio, "Efficient voice activity detection algorithms using long-term speech information", *Speech Comm.* vol. 42(3-4), pp. 271-287, 2004.

[Renevey2001] P. Renevey and A. Drygajlo, "Entropy based voice activity detection in very noisy conditions", in *Proc. EUROSPEECH*, USA, September 2001, pp. 1887-1890.

[Rohlicek1989] J.R. Rohlicek, W. Russell, S. Roukos, H. Gish, "Continuous hidden Markov modeling for speaker-independent word spotting", in *Proc. ICASSP*, pp. 627-630, 1989.

[Rose2002] P. Rose, *Forensic Speaker Identification*, Taylor &Francis, London, 2002.

[Seide2004] F. Seide, P. Yu, C. Ma and E. Chang, "Vocabulary-independent search in spontaneous speech", in *Proc. ICASSP*, vol.1, pp. 253-256, Montreal, Canada, 2004.

[SIL2007] Summer Institute of Linguistics (SIL), glossary of linguistic terms, http://www.sil.org/linguistics/glossaryoflinguisticterms/ (2007.10.03)

[Slomka1998] S. Slomka, S. Sridharan and V. Chandran, "A Comparison of Fusion Techniques in Mel-Cepstral Based Speaker Identification", in *Proc. ICSLP 1998*, Sydney, Australia, 1998.

[Soong1987] F. K. Soong, A. E. Rosenberg, B.-H. Juang and L. R. Rabiner, "A vector quantization approach to speaker recognition", *AT&T Technical Journal*, vol. 66, pp. 14-26, 1987.

[Soong1988] F.K. Soong and A.E. Rosenberg, "On the Use of Instantaneous and Transitional Spectral Information in Speaker Recognition", *IEEE Trans. Acoustics, Speech and Signal Processing*, 36(6), pp. 871-879, 1988.

[Szöke2005] I. Szöke, L. Burget, J. Černocký, M. Fapšo, M. Karafiát, P. Matějka and P. Schwarz, "Comparison of keyword spotting approaches for informal continuous speech", in *Proc. INTERSPEECH*, pp. 633-636, Lisabon, Portugal, 2005.

[Tanyer2000] S.G. Tanyer and H. Ozer, "Voice Activity Detection in Nonstationary Noise", *IEEE Trans. Speech and Audio Processing*, vol.8(4), July 2000.

[Timofte2006] R. Timofte, V. Hautamäki and P. Fränti, "Speaker, Vocabulary and Context Independent Word Spotting System in Continuous Speech", in *Proc. ISCSLP*, pp. 396-407, Singapore, December 2006.

[Timofte2007] R. Timofte and P. Fränti, "Voice Activity Detection Based on Short-term Time Series", submitted.

[Tong2006] R. Tong, B. Ma, K.A. Lee, C.H. You, D.L. Zou, T. Kinnunen, H.W. Sun, M.H. Dong, E.S. Ching and H.Z. Li, "Fusion of acoustic and tokenization features for speaker recognition", in *Proc. ISCSLP*, pp. 566-577, Singapore, 2006.

[Tuononen2007] M. Tuononen, R. Gonzales-Hautamäki and P. Fränti, "Applicability and Performance Evaluation of Voice Activity Detection", submitted.

[Turunen2005] M. Turunen, J. Hakulinen, K.-J. Räihä, E.-P. Salonen, A. Kainulainen, and P. Prusi, "An architecture and applications for speech-based accessibility systems", *IBM Systems Journal*, vol. 44, pp. 485-504, 2005.

[Viterbi1967] A. Viterbi, “Error bound for convolutional codes and an asymptotically optimum decoding algorithm”, *IEEE Trans. on Information Theory*, vol. 13, i. 2, pp.260-269, 1967.

[Yiu2002] K.K. Yiu, M.W. Mak and S.Y. Kung, "A Comparative Study on Kernel-Based Probabilistic Neural Networks for Speaker Verification", *International Journal of Neural Systems*, vol. 12, No. 5, 381-391, 2002.

[Yu2004] P. Yu and F. Seide, ”A hybrid word/phoneme-based approach for improved vocabulary-independent search in spontaneous speech”, in *Proc. INTERSPEECH*, pp. 293-296, Jeju, Korea, 2004.

[Weber2002] F. Weber, L. Manganaro, B. Peskin and E. Shriberg, ”Using Prosodic and Lexical Information for Speaker Identification”, in *Proc. ICASSP 2002*, pp. 141-144, Orlando, USA, 2002.

[Zhang2002] J. Zhang, W. Ward and B. Pellom, “Phone Based Voice Activity Detection Using Online Bayesian Adaptation with Conjugate Normal Distributions”, in *Proc. ICASSP*, Orlando Florida, USA, May 2002.

[Zilca2000] R. D. Zilca and Y. Bistritz, “Feature concatenation for speaker identification”, in *Proc. European Signal Processing Conference*, Tampere, Finland, 2000.