M A S T E R ' S   T H E S I S

Speech Recognition as a Retrieval Problem

University of Bonn
Department of Computer Science

Joscha Simon Rieber

May 6, 2013

Acknowledgments

This master's thesis has been written for the working group NetMedia headed by Dr. Rolf Bardeli at the Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS. It is the final thesis for my degree Master of Computer Science at the Institute for Computer Science at the University of Bonn.

First, I want to thank Dr. Rolf Bardeli for supervising my thesis. I strongly appreciate his effort in supporting my work. He and the other members of NetMedia helped me developing and presenting the results of my thesis. I would like to extend my particular thanks to Prof. Dr. Michael Clausen, heading the working group for multimedia signal processing at the University of Bonn, for supporting me during my studies. I owe Mr. Clausen a strong growth of interest in the field of research of signal processing and multimedia retrieval over the past years of study. I thank apl. Prof. Dr. Frank Kurth from the Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE for his support during my bachelor's thesis. His lessons helped me a lot when writing this master's thesis. I thank these persons and the other members of the working group for multimedia signal processing at the University of Bonn for very inspiring discussions and a lot of useful criticism that influenced my work.

My sincere thanks go to my parents, my brother and my partner Anja Diederich who always encouraged and supported me during my studies.

Joscha Simon Rieber

May 6, 2013

Contents

List of Figures

1 Introduction

Speech is the primary means of communication between people. Over the last six decades, *automatic speech recognition (ASR)* has been a challenging domain of science. Starting in 1952, the first ASR systems were built to automatically identify words spoken in isolation using time-aligning dynamic programming techniques [14]. Statistical approaches based on machine learning and *hidden Markov models (HMM)* have been established and outperform earlier technologies. However, these approaches are based on a training of statistical models of the dictionary [31, 32]. These models are then used to recognize speech. In many cases, additional knowledge about the language, such as a phoneme dictionary, is utilized, which describes the sound propagations of words.

This master's thesis deals with ASR systems that are training independent and do not employ any knowledge about language characteristics. These systems utilize a *retrieval-based approach*, recognizing a spoken word string by finding the most similar word utterance in a reference set for each word utterance contained in the test string. This method is based on the fundamental assumption that each possibly occurring word in a test string is contained in the set of references, as well. The problem is referred to as *connected word recognition*. If the best matching utterance for each word contained in the test string is known, it can be identified. This approach has the advantage to be mainly independent of the language and is more flexible than many common approaches. A connected word recognizer can be used, for example, to enable human-computer interactions by means of voice or to automatically transcribe a speech database. As shown in [25] such an ASR system is desired, for example, in case of the Sinhala language spoken in Sri Lanka. However, in this work of Priyadarshani et al. in 2012 only words spoken by a single speaker in isolation are considered, whereas in this work speaker-independent connected word recognizers are considered.

A word string is identified using techniques based on *dynamic time warping (DTW)*, which is a well-known retrieval technology for speech and music [1, 15, 18, 21, 25]. DTW can be used to find spoken words in a collection of known utterances that are similar to the content in a test string by finding the best-fitting time alignment, using a dynamic programming algorithm. Thus, it can handle variations in the time domain. A suitable DTW-based connected word recognizer was introduced by Myers and Rabiner in 1981. In the work [23] a *level-building DTW* algorithm is described. It builds up the test string, word-by-word, by iteratively comparing portions of the test string with a set of known reference utterances by finding the best-fitting time alignment. The algorithm determinates, if for each word in the test string a matching reference utterance is found. Thus, it can be used to identify the test string. This approach has been verified to reliably identify strings of connected digits in [22]. In the work, reference templates were utilized that are selected utterances of digits spoken in isolation.

In this work, we developed and evaluated three different variants of speech recognition systems based on level-building DTW. The first approach, called the *template recognizer*, uses reference templates. This follows the instructions given by Myers and Rabiner. Instead of using selected templates, our *database recognizer* uses the whole database of known utterances as references. The third recognizer works in two separate steps. Before recognizing the test string using level-building DTW, the references are retrieved from the database by finding utterances matching to uniformly distributed windowed frames taken from the test string. The retrieval technique used is *subsequence DTW*, which is detailed, for instance, in [21]. It is a DTW-based retrieval method that can find word utterances in a database similar to a given query utterance. It is referred to as the *retrieval-based recognizer*.

All concepts are based on a preprocessing step to extract *speech features* from the waveform of the test string, which are small sets of time-varying parameters representing speech characteristics compactly. The speech features are only extracted from time positions containing speech. To achieve this, we invented a new approach for a *speech-silence segmentation* obtained by analyzing the spectrogram. Spectral vectors containing a sufficient number of *speech candidate cells* in the spectrogram are classified as originating from speech. These cells have either a high amplitude or a high local amplitude variance. Following the speech-silence segmentation, either *Mel-frequency cepstral coefficients (MFCCs)* or *linear predictive coding (LPC)* features are extracted. MFCCs are obtained by analyzing the spectral characteristics according to properties of the human perception [19]. LPC speech features are obtained by estimating parameters of a simplified model of speech production [26].

A crucial point when using speech features is the distance between two feature vectors, which can be used as a *local cost measure* in DTW-based algorithms to quantify similarity. In this work, MFCCs are compared using the cosine distance, whereas LPC features are compared using a more complex measurement based on *LPC residuals*, introduced by Itakura in [13], quantifying model errors. The residual-based local cost measure is introduced by Rabiner and Schmidt for connected digit recognition in [27].

The three retrieval-based ASR systems are implemented and evaluated using the *Aurora-2 speech corpus* [11], which is a processed set based on *Texas Instruments Digits (TIDigits)* [16]. It contains short word strings of 1–7 digits uttered by various speakers. It is used to create a speaker-independent test scenario by choosing 770 test word strings uttered by 10 male and 10 female speakers and 2823 database strings uttered by 40 male and 40 female speakers. First, the speech retrieval system based on subsequence DTW is evaluated and optimized. Then, the three retrieval-based ASR systems are evaluated. For the retrieval-based recognizer the optimal speech retrieval system is used from the previous step. The results show that the retrieval-based recognizer has a weak performance. The template recognizer using 12 template references per digit obtains a *word error rate (WER)* [12] of 17.6 % and yields a fast processing time. The database recognizer demands inherently more processing time, but the WER reduces to 13.9 %, whereas the most frequent errors are additional insertions of short utterances „oh“ and „eight“. In case of known string lengths, the WER drops to 2.1 %, which is the same result a state-of-the-art phoneme recognizer using HMM achieves. Further evaluations show a significant accuracy deterioration in case of noise-affected speech recordings.

Chapter 2 describes the basic principles of digital signal processing including storing and analyzing speech. Chapter 3 introduces the feature extraction process including MFCCs, LPC and the speech-silence segmentation. The next Chapter 4 details the theory of DTW and the retrieval of references using subsequence DTW. In Chapter 5 the concept of retrieval-based ASR and level-building DTW are detailed. The evaluations are detailed in Chapter 6. A summary is given in Chapter 7.

2 Fundamentals

In this chapter, the basic principles for storing and analyzing speech within a digital computer system are introduced. Section 2.1 introduces the fundamental notation. Sections 2.2 and 2.3 introduce basic methods for analyzing speech. Further details can be found in [10, 12, 21, 26, 29].

2.1 Sound, Waveform, Analog and Digital Signal

A sound perceived by the human ear originates from a source, such as the vocal chords of a human. The vibration produced in the *vocal tract*, where speech is produced, excites an oscillation of the particles in the air. Their density and the local air pressure oscillates in the direction of speech. Air pressure is measured in Pascal [Pa], where 1 Pa = 1 N/m^2. It is the ratio of force and the size of the area it exerts on. A common measurement for the sound pressure is a relation between the local air pressure p_{signal} and a reference value, fixed as $p_0 := 20\ \mu\text{Pa}$. It is called the *sound pressure level (SPL)* and is measured in *decibel (dB).*

$$\text{SPL}\left(p_{\text{signal}}\right) := 20 \cdot \log_{10}\left(\frac{p_{\text{signal}}}{p_0}\right)\ \text{dB}. \tag{2.1}$$

The oscillating sound pressure propagates through the air as a *wave.* The wave can be captured by a microphone transforming it into an electrical, i.e., analog, *signal.* Thus, speech can be characterized in terms of its signal. An analog signal is also called a *continuous-time (CT) signal.* In practical applications, the sound of speech, i.e., the *desired signal*, originating from the vocal tract of a speaker interferes with additional sounds. These sounds are often classified as *background noise.* A measurement for the relative strength of the desired signal and the background noise is the *signal-to-noise ratio (SNR).* It is defined as

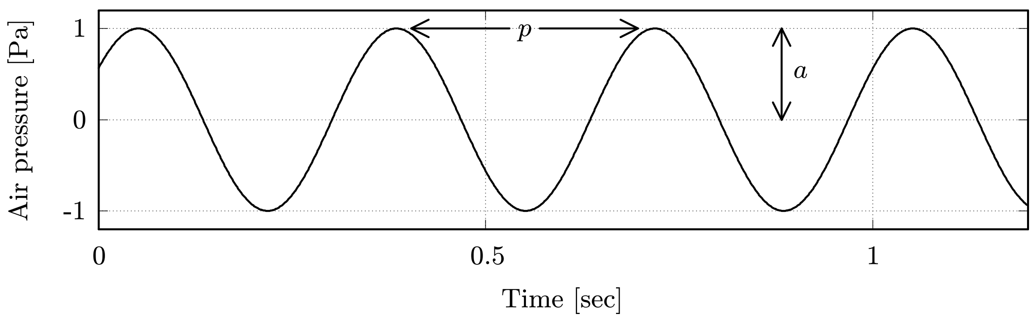

Figure 2.1: Waveform of a periodic sound with a frequency of 3 Hz. p shows the period and a the amplitude.

$$\mathrm{SNR}\left(p_{\text{signal}}, p_{\text{noise}}\right) := 20 \cdot \log_{10}\left(\frac{p_{\text{signal}}}{p_{\text{noise}}}\right) \text{ dB}. \tag{2.2}$$

The wave propagating through the air can be characterized by a plot of air pressure against time. This leads to the *waveform* of the signal. If the waveform describes a repetitive signal, it is called *periodic*. The smallest time duration of the repetitive structure defines the *period*. The reciprocal of the period is called the *frequency*. It is measured in Hertz [Hz]. A periodic waveform with a frequency of 3 Hz is illustrated in Figure 2.1.

A waveform of an analog signal cannot be processed by a digital computer system, because of its continuous nature. It has to be transformed into a discrete and finite digital signal. First, a concept for transforming an infinite analog signal into an infinite discrete signal is introduced. This step is referred to as *sampling*. Let $f : \mathbb{R} \to \mathbb{R}$ be the infinite waveform of a CT signal. A uniform sampling can be expressed as $x(n) := f(n \cdot R)$, where $n \in \mathbb{Z}$, the *sampling period* $R \in \mathbb{R}$ and $x : \mathbb{Z} \to \mathbb{R}$ is a discretized version of f. x is called a *discrete-time (DT) signal*. The reciprocal of the sampling period is called the *sampling frequency* or *sampling rate*. In case of a finite discrete waveform of length N, x is assumed to be equal to zero outside an interval of $[0 : N-1]$. To obtain a digital signal, the values of f have to be of discrete nature, as well. The procedure of discretizing the range of x is called *quantization*. The most common method of quantizing a discrete-time signal x is by using a uniform mapping to a finite set of values. The values are stored as bit strings. The number of bits reserved per sample value is called the *bit depth*. Figure 2.2 illustrates the process of discretizing a CT waveform.

In theory, signals are elements of *signal spaces* defining desired properties. The *Lebesgue spaces* [6] only contain signals of finite norms that are proportional to the energies

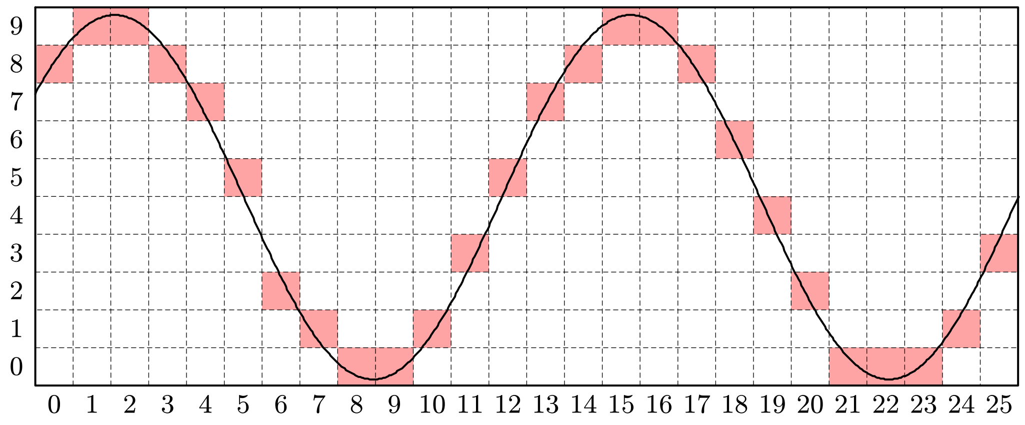

Figure 2.2: Discretization of an analog waveform. The light red boxes show the discretized version of the black continuous waveform.

transmitted by the waves. The spaces of CT signals fulfilling this property can be mathematically expressed for $p \in [1 : \infty)$ as

$$L^p(\mathbb{R}) := \left\{ f : \mathbb{R} \to \mathbb{C} \mid \|f\|_p < \infty \right\}, \tag{2.3}$$

where the p-norm of a CT signal is defined as

$$\|f\|_p = \left(\int_{\mathbb{R}} |f(t)|^p \, dt \right)^{1/p}. \tag{2.4}$$

The Lebesgue spaces of DT signals for $p \in [1 : \infty)$ are defined as

$$\ell^p(\mathbb{Z}) := \left\{ x : \mathbb{Z} \to \mathbb{C} \mid \|x\|_p < \infty \right\}, \tag{2.5}$$

where the p-norm of a DT signal is defined as

$$\|x\|_p = \left(\sum_{k \in \mathbb{Z}} x(k) \cdot \overline{x(k)} \right)^{1/p}. \tag{2.6}$$

In general, we assume that $p = 2$, leading to commonly used signal spaces, which are called *Hilbert spaces*. Note that the process of digitizing a CT waveform always accompanies a loss of information. The process of sampling can lead to artifacts, which are called *aliasing*.

It happens, when very high frequencies are involved that cannot be represented by the time-discretized version using the fixed sampling rate. The *Nyquist-Shannon sampling theorem* quantifies the relation between the highest frequency in the original signal and the sampling rate. If the original CT waveform $f \in L^2(\mathbb{R})$ is Ω-bandlimited, i.e., no frequency greater than Ω occurs in f, then f can be reconstructed from the discretized version $x := f(n \cdot T) \in \ell^2(\mathbb{Z})$, where the sampling period $T \leq 2 \cdot \Omega$. Thus, the sampling frequency $1/T$ has to be at least $1/(2 \cdot \Omega)$. The second step of digitizing, i.e., the quantization, leads to a distortion of the signal. The result of fixing the values in a discrete grid has a similar effect as adding random noise to the original signal. It is also called *quantization noise*. The most common audio format stores the waveform with a sampling rate of 44,100 Hz and each sample is represented with a bit depth of 16. Because of the very high sampling rate, all audible frequencies can be represented and the chosen bit depth leads to a hardly audible quantization noise.

2.2 Fourier Transform

A common method for analyzing digital audio signals is to transform a given time-domain signal into a frequency-domain signal. This can be done, for instance, by using the *Fourier transform*. It delivers a frequency-domain signal that is referred to the *spectrum* of the original signal. In general, the process can be compared to a prism splitting light into its constituent parts, which helps analyzing the spectral content of a ray. In terms of audio signal processing, the Fourier transform splits a given signal into frequency components composing the original signal. A comprehensive introduction into the Fourier transform can be found in [21]. To analyze a discrete-time signal $x \in \ell^2(\mathbb{Z})$ by means of periodic *frequency functions*

$$e_\omega(n) := e^{2\pi i \omega n} \qquad (n \in \mathbb{Z}, \omega \in [0,1]) \tag{2.7}$$

the Fourier transform is defined as

$$\widehat{x}(\omega) := \sum_{n \in \mathbb{Z}} x(n) \cdot \overline{e_\omega(n)} = \sum_{n \in \mathbb{Z}} x(n) \cdot e^{-2\pi i \omega n} \qquad (\omega \in [0,1]) \tag{2.8}$$

with $\widehat{x} : [0,1] \to \mathbb{C}$. In this expression, x is represented by a superposition of elementary frequency functions e_ω. A mapping back from a frequency-domain signal to a time-domain signal is given by the *inverse Fourier transform*. It is defined as

$$x(n) = \int_{\omega\in[0,1]} \widehat{x}(\omega) \cdot e_{\omega}(n)\,\mathrm{d}\omega = \int_{\omega\in[0,1]} \widehat{x}(\omega) \cdot e^{2\pi i\omega n}\mathrm{d}\omega. \tag{2.9}$$

Since computing the Fourier transform at an infinite set of values within the interval $[0,1]$ is computationally infeasible, it is evaluated on a finite set of frequencies, uniformly distributed within the interval. The *discrete Fourier transform (DFT)* solves the problem of approximately computing the Fourier transform. The DFT of length N is a linear mapping $\mathbb{C}^N \to \mathbb{C}^N$, given by the $N \times N$ matrix

$$DFT_N := \left(\Omega_N^{kj}\right)_{0\le k,j<N} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \Omega_N & \cdots & \Omega_N^{(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \Omega_N^{(N-1)} & \cdots & \Omega_N^{(N-1)(N-1)} \end{pmatrix}, \tag{2.10}$$

where $\Omega_N := e^{-2\pi i/N}$. For the input vector $v := (v(0), v(1), \ldots, v(N-1))^\top \in \mathbb{C}^N$, the matrix multiplication $\widehat{v} := \mathrm{DFT}_N \cdot v$ evaluates the DFT of v. The value $\widehat{v}(k)$ gives the approximate intensity of the frequency $k/(T\cdot N)$ involved in the original waveform, where $1/T$ is the sampling frequency. The upper half of the vector $\widehat{v}$ gives a mirrored version of the lower half. This effect arises from aliasing. Thus, only the lower half is regarded. The DFT can be computed efficiently in time $\mathcal{O}(N\log N)$ using the *fast Fourier transform (FFT)* if the DFT length N is a power of two. A method for transforming a frequency-domain signal back to a time-domain signal is the *inverse FFT*. For further details see [5].

When analyzing a speech waveform using the FFT, one general problem has to be discussed. The result of the FFT shows the spectrum for every time position at once. Thus, the frequencies involved can be analyzed, but the information about spectral variations over time gets lost. Since speech signals are approximately stationary for lengths around 10 to 30 msec, a method for analyzing a small portion of a waveform is desired. The process of evaluating the FFT on a small portion of the original signal is called the *short-time Fourier transform (STFT)*. It is defined for frequency $\omega \in [0,1]$ and time position $k \in \mathbb{Z}$ as

$$\widehat{x}(\omega, k) := \sum_{n\in\mathbb{Z}} x(n) \cdot w(n-k) \cdot e^{-2\pi i\omega n}, \tag{2.11}$$

where $w : \mathbb{Z} \to [0,1]$ is a *window function*. It has the property to pass only a small segment of x. The window is centered in $w(0)$. A common function is the *Hann window* w_H of length N, which is defined as

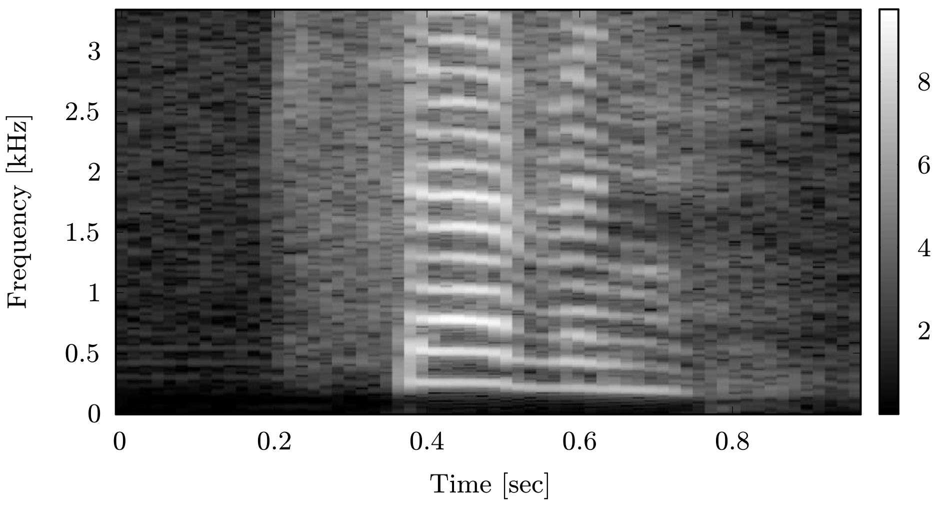

Figure 2.3: Spectrogram of an utterance of the word „seven“.

$$w_H(n) := 0.5 \cdot \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \qquad (n \in \mathbb{Z}). \tag{2.12}$$

The STFT can be computed efficiently by first framing and windowing a digital waveform and then computing the FFT of the segment. In most practical cases, instead of analyzing the signal by means of the complex-domain STFT, the squared magnitude, which is proportional to the energy, is considered. The squared magnitude STFT is also called the *spectrogram*, i.e.,

$$S_x(\omega, k) := \left|\hat{x}(\omega, k)\right|^2 \qquad (\omega \in [0, 1], k \in \mathbb{Z}), \tag{2.13}$$

where $S_x(\omega, k) \in \mathbb{R}$. Evaluating the FFT of uniformly distributed frames leads to a matrix $S : \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}$ describing the spectrogram. This matrix can be visualized by encoding the values as levels of gray. Since the span between extreme values can be very large in practical applications, the values of matrix S are compressed using a logarithmic function such as the common logarithm. This step is performed due to the fact that the human perception of amplitudes is scaled logarithmically. Figure 2.3 shows the spectrogram of an utterance of the word „seven“ rescaled by using the common logarithm.

2.3 Digital Filters

In this chapter, a brief introduction to the fundamentals of digital filters is given. For further details on filters, see, for example, [21, 26]. A *filter*, in general, is a tool that alters the sound characteristics of a waveform. Filters can affect the spectral properties by boosting or attenuating certain frequency bands. It belongs to the general group of processes influencing a waveform called a *system*. The most important class of systems can be mathematically expressed in terms of a *convolution*. The convolution of two DT signals x and y leads to a DT signal $x * y$. It is defined as

$$[x * y](n) := \sum_{k \in \mathbb{Z}} x(k) \cdot y(n-k). \tag{2.14}$$

Using the convolution operation, all *linear and time invariant (LTI)* systems can be represented, where time invariance indicates that the system output is independent of a time shift of the input signal, except of an identical time shift. Let the DT signal $h \in \ell^1(\mathbb{Z})$, then the operation $C_h : \ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z})$, given as $C_h[x] := h * x$, defines a *convolution filter*. The operator has the property that it outputs the signal h, if the input signal is equal to the *impulse signal* δ, where $\delta(0) = 1$ and $\delta(n) = 0$ for $n \neq 0$. Thus, h is called the *impulse response* of the filter with the *filter coefficients* $h(n)$. An important group of filters are those, where the filter coefficients $h(n) = 0$ for $n < 0$. Then, the output of the filter $C_h[x](n)$ is not affected by the coefficients of the input signal sequentially succeeding n. This means that the filter does not process signal states in the „future“. Therefore such a filter is called *causal*. If h has the property to be non-zero only on a finite set of coefficients, the filter is called *finite impulse response (FIR) filter*, otherwise it is called *infinite impulse response (IIR) filter*. A causal FIR filter of order N can be mathematically expressed as

$$C_h[x](n) = [h * x](n) = \sum_{k=0}^{N} h(k) \cdot x(n-k). \tag{2.15}$$

Since filters alter spectral properties, a tool for analyzing and designing modifications in the frequency domain is helpful. In Section 2.2 a method for transforming a DT signal to the frequency domain is given. To analyze the spectral properties of a convolution filter, the impulse response h can be transformed to its spectrum $\widehat{h}$ using the Fourier transform. $\widehat{h}$ is also referred to as the *frequency response* of the filter. For the question, how the frequency response influences the spectral properties of an input signal x when applying the convolution filter $C_h[x]$, the *convolution theorem* gives a suitable answer. It states

$$\widehat{C_h[x]} = \widehat{h * x} = \widehat{h} \cdot \widehat{x}. \tag{2.16}$$

This means that the spectrum of the convolution filter output can be expressed as a point-wise multiplication of the frequency response with the spectrum of the input signal. Besides the qualitative meaning of the convolution theorem, it gives a hint for computing a convolution in $\mathcal{O}(n \log n)$ time using the FFT and the inverse FFT. The frequency response can be split into two components

$$\widehat{y}(\omega) = |\widehat{y}(\omega)| \cdot e^{2\pi i \Phi_h(\omega)}, \tag{2.17}$$

where $|\widehat{y}(\cdot)|$ is called the *magnitude response* and $\Phi_h(\cdot)$ is called the *phase response*. Commonly, the magnitude response is analyzed in the decibel scale, i.e., $|\widehat{y}(\omega)| \mapsto 20\log_{10}(|\widehat{y}(\omega)|)$. The phase response is measured in radians.

3 Speech Features

One important aspect of speech research is to describe a speech waveform by means of a small set of time-varying parameters. These parameters represent certain speech characteristics and are called *speech features.* Section 3.1 introduces a common method for extracting features based on perceptual properties. In Section 3.2 an approach for estimating a set of parameters influencing the process of speech production within a simplified model is detailed. The final Section 3.3 discusses the problem of discriminating between speech and silence in a speech recording. This step is important to discard silent regions before analyzing the speech signal.

3.1 Mel-frequency Cepstral Coefficients

The common concept of extracting *Mel-frequency cepstral coefficients (MFCCs)* from a waveform as speech features is introduced in numerous publications, for instance, in [10,12,19,31] and has successfully been tested in various speech processing applications, such as, to recognize Sinhala utterances spoken in isolation [25]. It is also used in state-of-the-art continuous speech recognition systems [32]. MFCCs are obtained by first analyzing the magnitude STFT by means of triangular filters that are arranged according to perceptive properties, then a logarithmically scaled spectrum of the components yields the *cepstrum.* The procedure for extracting MFCCs from a waveform is illustrated in Figure 3.1.

In the first step, the magnitude STFT is computed on short, uniformly distributed frames of the given waveform. The frame size is chosen around 20 ms and a Hann window is commonly used for the STFT. In the next step, each value of the magnitude STFT is rescaled using a logarithm, because experiments have proven a logarithmically scaled human perception of loudness. The resulting spectral coefficients are analyzed by triangular filters arranged according to the *Mel scale.* Figure 3.2 (a) illustrates the Mel scale. It is based on experiments on a relationship between actual frequencies and perceived pitches. This mapping is approximately linear up to 1000 Hz, but then changes in a logarithmic

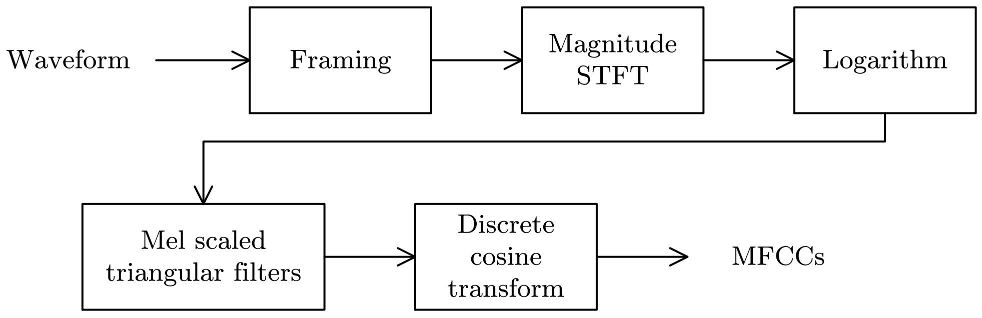

Figure 3.1: Process to extract MFCC features.

manner. Thus, the human auditory system is able to perceive lower frequencies with a higher resolution than higher frequencies. The relation between frequency ν_{Hz} in Hz and Mel value ν_{Mel} can be mathematically expressed as

$$\nu_{\mathrm{Mel}} = 2595 \cdot \log_{10}\left(1 + \frac{\nu_{\mathrm{Hz}}}{700}\right). \tag{3.1}$$

The Mel-distributed triangular filters simulate the distribution of this perceptual property. They are illustrated in Figure 3.2 (b). The filters are computed by binning spectral components within the triangles. This yields a Mel scaled spectrum. To reduce the number of features, in the last step a statistically decorrelating transform is used. It is called the *discrete cosine transform (DCT)*. Let $m_j, j = 1, 2, \ldots, N$ be the Mel scaled spectral coefficients, then

$$c_i := \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right) \tag{3.2}$$

defines the i^{th} cepstral coefficient resulting from the DCT. Commonly, besides the cepstral coefficients, the first and second derivatives of the features are taken into account, as well. This yields the *delta* und *delta-delta coefficients*. The delta coefficients can be approximated from the cepstral coefficients c_i by

$$\Delta c_i := c_{i-2} - c_{i+2}. \tag{3.3}$$

A recursive procedure yields the delta-delta coefficients.

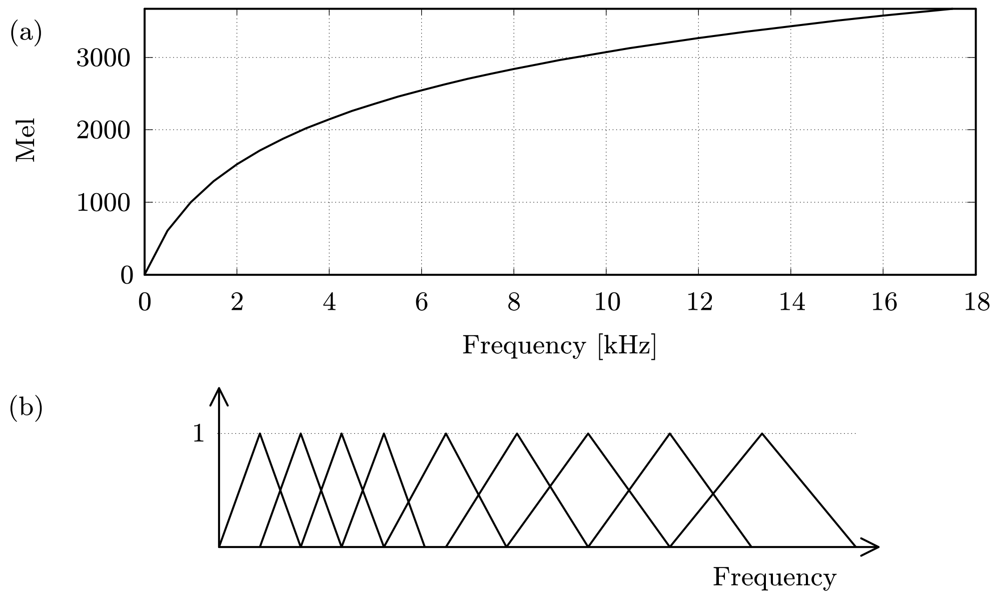

Figure 3.2: (a) Mel scale. (b) Scheme of Mel-distributed triangular filters.

3.2 Linear Predictive Coding

Linear predictive coding (LPC) is a well-known and powerful method for speech analysis and is described in numerous publications, such as, [2, 12, 13, 26, 30]. The fundamental concept of LPC is to analyze a simplified model of speech production by means of a small set of parameters. These parameters continuously change over time. To approximate the parameter changes, a given speech recording is analyzed on small time segments. For each segment, the parameters are estimated using an efficient algorithm. In contrast to common approaches based on analyzing the spectrum, like MFCCs, LPC estimates the speech features directly from the waveform. In the following, the simplified speech production model and the small set of estimated parameters used to describe the speech production process are introduced.

The process of human speech production occurs in the vocal tract. The vocal tract is excited by the *glottis*, which is the space where the vocal chords are located. The shape of the vocal tract affects the sound. In case of voiced sounds, such as the vowels „o“, „u“ and „a“, the human vocal tract is excited by quasiperiodic pulses generated within the glottis. Unvoiced sounds are generated by air flowing turbulently through the vocal tract. A simplified model for the speech production process is based on these assumptions. When creating voiced sounds, the excitation is approximated by an impulse train generator. Unvoiced sounds are approximated by a random noise generator. To model the influence

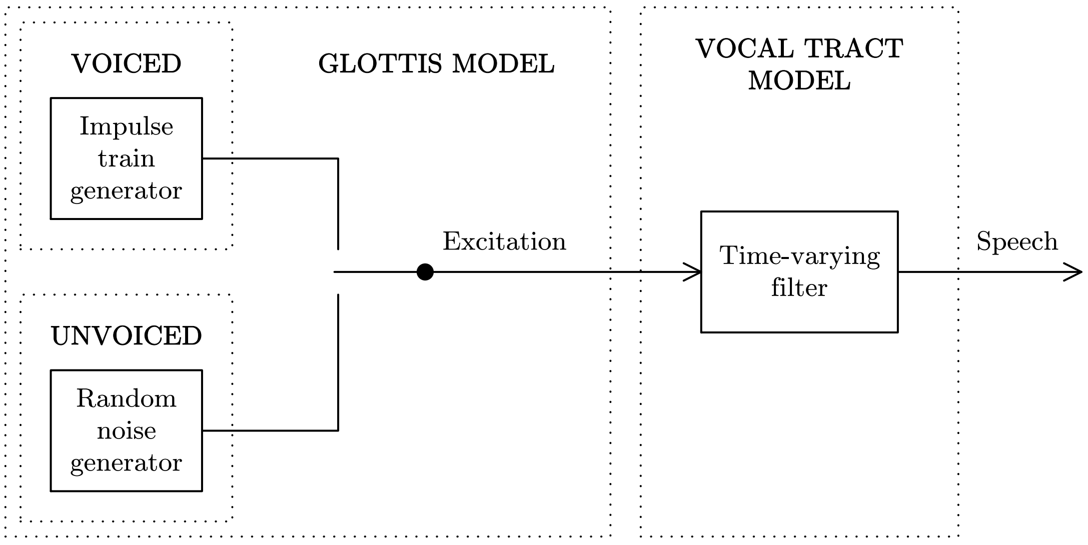

Figure 3.3: A simplified model for speech production.

of the vocal tract shape, the excitation signal is processed by a linear filter. For each temporal state the vocal tract shape is approximated by the filter parameters. This model is illustrated in Figure 3.3.

Let $s(n)$ be the speech created with a causal FIR convolution filter of order p with filter coefficients a_k, where a_0 is identically one and thus disregarded and let $u(n)$ denote the excitation. Then

$$s\left(n\right) = \sum_{k=1}^{p} a_k s\left(n-k\right) + u\left(n\right) \tag{3.4}$$

describes the process of speech production shown in Figure 3.3. The filter coefficients a_k are to be estimated using a linear prediction approach. Let

$$\overline{s}\left(n\right) = \sum_{k=1}^{p} \alpha_k s\left(n-k\right) \tag{3.5}$$

be the output of the linear predictor. The α_k define the predictor coefficients. p is called the *prediction order*. The *predictor error* is the difference between the predictor output and the filter output from Equation (3.4). It is defined as

$$\begin{aligned} e\left(n\right) &:= s\left(n\right) - \overline{s}\left(n\right) && (3.6)\\ &= s\left(n\right) - \sum_{k=1}^{p} \alpha_k s\left(n-k\right). && (3.7)\end{aligned}$$

$$\begin{aligned} E\left(n\right) &:= \sum_{m\in\mathbb{Z}} e^2\left(m+n\right) && (3.8)\\ &= \sum_{m\in\mathbb{Z}} \left(s\left(m+n\right) - \sum_{k=1}^{p} \alpha_k s\left(m+n-k\right)\right)^2 && (3.9)\end{aligned}$$

is the accumulated squared predictor error within a certain region around n. To determine the optimal predictor coefficients, it is supposed that

$$\frac{\partial E(n)}{\partial \alpha_i} = 0 \qquad \left(i = 1, 2, \ldots, p\right). \tag{3.10}$$

For all $1 \leq i \leq p$: holds

$$\sum_{m\in\mathbb{Z}} s\left(m-i+n\right) s\left(m+n\right) = \sum_{k=1}^{p} \widehat{\alpha}_k \sum_{m\in\mathbb{Z}} s\left(m-i+n\right) s\left(m-k+n\right), \tag{3.11}$$

where $\widehat{\alpha}_k$ are the optimal predictor coefficients. In the following, the optimal predictor coefficients are called α_k, as well. To simplify the equation, the subtitution

$$\phi_n\left(i,k\right) := \sum_{m\in\mathbb{Z}} s\left(m-i+n\right) s\left(m-k+n\right) \tag{3.12}$$

is made. Following, Equation (3.11) can be expressed as

$$\sum_{k=1}^{p} \alpha_k \phi_n\left(i,k\right) = \phi_n\left(i,0\right) \qquad \left(i = 1, 2, \ldots, p\right). \tag{3.13}$$

Since the range of the sum over m is not specified, Equation (3.13) cannot be solved efficiently. Thus, a windowed speech segment of length N is considered.

$$\Rightarrow \forall n \notin [0 : N-1] : s(n) = 0. \tag{3.14}$$

3.2.1 The Autocorrelation Method

For the accumulated squared error, it follows that the summation range over m can be shortened to a finite interval, i.e.,

$$E(n) = \sum_{m=0}^{N-1+p} e^2\left(m+n\right) \tag{3.15}$$

Considering the finite interval, Equation (3.12) becomes

$$\phi_n\left(i,k\right) = \sum_{m=0}^{N-1+p} s\left(m-i+n\right) s\left(m-k+n\right) \qquad \begin{array}{l}(1 \leq i \leq p)\\ (0 \leq k \leq p)\,.\end{array} \tag{3.16}$$

This term can be rearranged to

$$\phi_n\left(i,k\right) = \sum_{m=0}^{N-1-(i-k)} s\left(m+n\right) s\left(m+i-k+n\right). \tag{3.17}$$

Let $s_n\left(\cdot\right) := s\left(\cdot+n\right)$ be the n-shifted version of s and let

$$R_{s_n}\left(t\right) = \sum_{m=0}^{N-1-t} s_n\left(m\right) s_n\left(m+t\right) \tag{3.18}$$

be the *autocorrelation* of s_n at position t. Then ϕ_n in Equation (3.17) can be expressed by means of the autocorrelation function[1], i.e.,

$$\phi_n\left(i,k\right) = R_{s_n}\left(i-k\right). \qquad \begin{array}{l}(i = 1,2,\ldots,p)\\ (k = 0,1,\ldots,p)\end{array} \tag{3.19}$$

Since $R_{s_n}\left(\cdot\right)$ is even,

$$\phi_n\left(i,k\right) = R_{s_n}\left(\left|i-k\right|\right) \qquad \begin{array}{l}(i = 1,2,\ldots,p)\\ (k = 0,1,\ldots,p)\end{array} \tag{3.20}$$

holds. Therefore, Equation (3.13) can be expressed as

[1]The autocorrelation of a signal frame of length N can be computed efficiently in $\mathcal{O}\left(N \log N\right)$ time, using the fast Fourier transform [26].

$$\sum_{k=1}^{p} \alpha_k R_{s_n}\left(|i-k|\right) = R_{s_n}\left(i\right). \tag{3.21}$$

This expression can be represented in matrix form as

$$\begin{pmatrix} R_{s_n}(0) & R_{s_n}(1) & R_{s_n}(2) & \cdots & R_{s_n}(p-1) \\ R_{s_n}(1) & R_{s_n}(0) & R_{s_n}(1) & \cdots & R_{s_n}(p-2) \\ R_{s_n}(2) & R_{s_n}(1) & R_{s_n}(0) & \cdots & R_{s_n}(p-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{s_n}(p-1) & R_{s_n}(p-2) & R_{s_n}(p-3) & \cdots & R_{s_n}(0) \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_p \end{pmatrix} = \begin{pmatrix} R_{s_n}(1) \\ R_{s_n}(2) \\ R_{s_n}(3) \\ \vdots \\ R_{s_n}(p) \end{pmatrix}. \tag{3.22}$$

The $p \times p$ matrix containing autocorrelation values is a *Toeplitz matrix*, i.e., it is symmetric and for each diagonal all entries are equal. This certainty is used by the *Levinson-Durbin recursion*, which estimates the predictor coefficients α_k by means of an efficient algorithm running in $\Theta\left(p^2\right)$ time. The algorithm is detailed in [12, 26]. The introduced method is referred to as the *autocorrelation method*. In the following, another slightly more complex method for evaluating LPC coefficients is introduced.

3.2.2 The Covariance Method

First, the accumulated squared error is similarly defined as in Equation (3.15), but the summation is fixed over the range $m = 0, 1, \ldots, N-1$, i.e.,

$$E\left(n\right) = \sum_{m=0}^{N-1} e^2\left(m+n\right). \tag{3.23}$$

The term $\phi_n\left(i,k\right)$, in Equation (3.16), here becomes

$$\phi_n\left(i,k\right) = \sum_{m=0}^{N-1} s\left(m-i+n\right) s_n\left(m-k+n\right). \qquad \begin{matrix} (1 \le i \le p) \\ (0 \le k \le p) \end{matrix} \tag{3.24}$$

Equation (3.11) can be fixed to

$$\phi_n\left(i,0\right) = \sum_{k=1}^{p} \alpha_k \phi_n\left(i,k\right) \tag{3.25}$$

or in matrix form as

$$\begin{pmatrix} \phi_n(1,1) & \phi_n(1,2) & \phi_n(1,3) & \cdots & \phi_n(1,p) \\ \phi_n(2,1) & \phi_n(2,2) & \phi_n(2,3) & \cdots & \phi_n(2,p) \\ \phi_n(3,1) & \phi_n(3,2) & \phi_n(3,3) & \cdots & \phi_n(3,p) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_n(p,1) & \phi_n(p,2) & \phi_n(p,3) & \cdots & \phi_n(p,p) \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_p \end{pmatrix} = \begin{pmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ \phi_n(3,0) \\ \vdots \\ \phi_n(p,0) \end{pmatrix}. \tag{3.26}$$

The $p \times p$ matrix has the properties of a covariance matrix. Thus, this technique is referred to as the *covariance method*. In this variant, the predictor coefficients α_k can be computed in time $\Theta(p^3)$ using the *Cholesky decomposition*, which is described, for instance, in [12,26].

3.3 Discriminating Between Speech and Silence

For simplification purposes, we define that all time regions within a speech recording not containing speech are *silence*. In practical applications, silence in a speech recording is rather background noise originating from sound sources in the environment of the microphone capturing the speech signal. Thus, silence does not necessarily imply that the components of a waveform are close to zero. In this work, we assume that silence means random noise, which in many cases has a significantly lower amplitude than the average amplitude of the speech signal. Figure 3.4 (a) and (b) show a waveform and a logarithmically scaled spectrogram of a speech recording, which demonstrate the noisy character of silent regions.

When extracting speech features, silence inherently distorts the results, since extracting speech features from random noise can lead to randomly distributed features. Therefore, before analyzing the waveform of a speech recording, silence has to be discarded. Since silence does not necessarily mean that the amplitude of the waveform is close to zero, the naive approach of discriminating high and low amplitudes to distinguish between speech and silence cannot be a suitable solution in general.

3.3.1 Amplitude-modulation Features

In this work, each cell in the logarithmically scaled spectrogram is analyzed to classify a spectral vector to originate either from a speech or silence signal. This is done by our newly invented algorithm that analyzes each cell within the spectrogram. The algorithm marks cells that possibly originate from speech as speech candidates. If a spectral vector contains several cells marked as speech candidates, the time region in the waveform, from

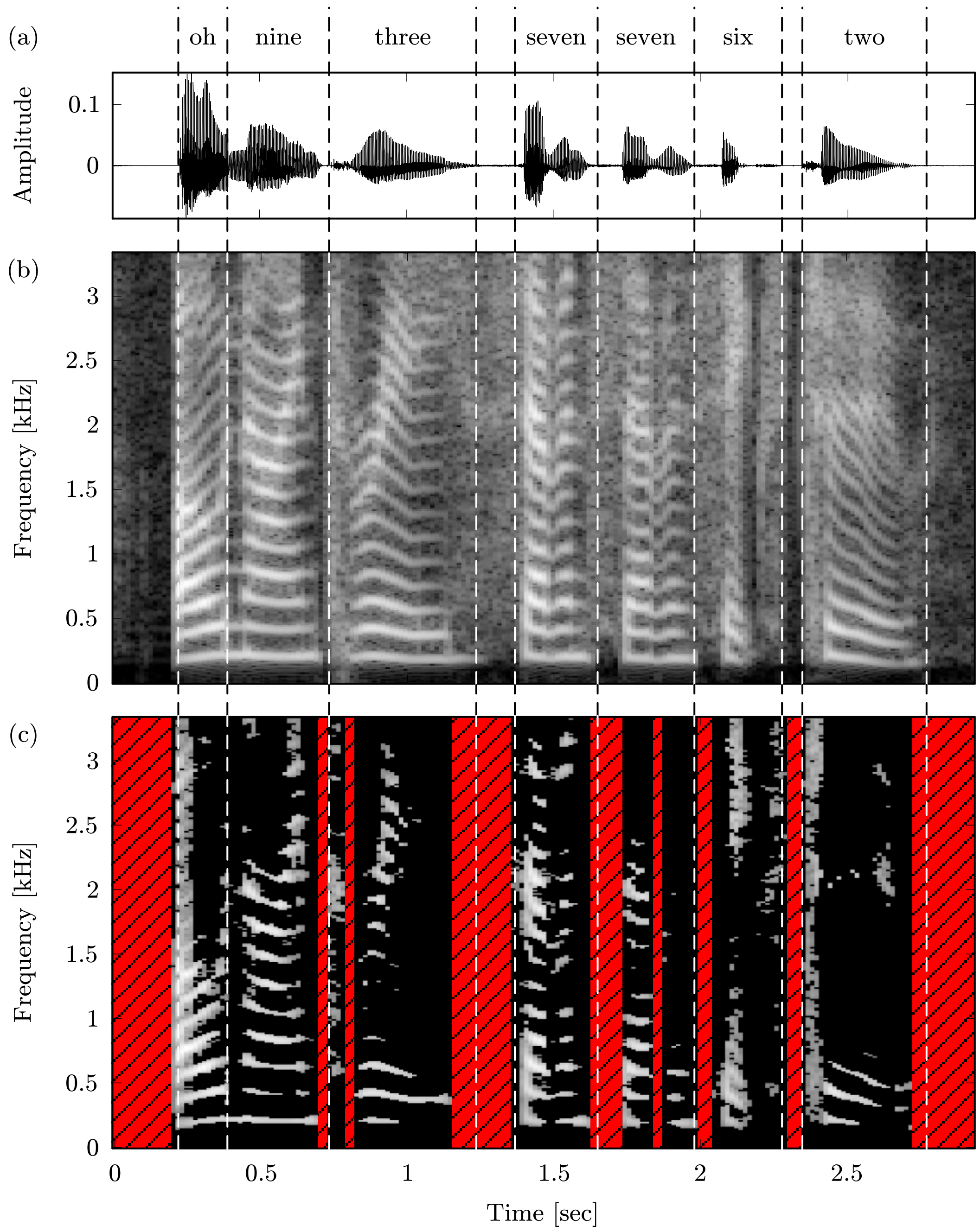

Figure 3.4: (a) Waveform and (b) the spectrogram of a speech recording containing the message „oh, nine, three, seven, seven, six, two“. (c) Spectrogram cells that are marked as speech candidates using Algorithm 3.1. Red marked regions are automatically classified as silence, because of an insufficient number of speech candidates.

which the spectral vector is computed, is classified as speech. If the time region is not classified as speech, it is assumed to be silence. Thus, a *speech-silence segmentation* of the waveform is provided. Before we investigate the algorithm, a formal definition has to be introduced. To analyze each cell within the spectrogram, the *optimal homogeneous path* is considered. It is a powerful tool and is utilized by the algorithm.

Let c be the analyzed cell within spectrogram S. c is called the *center cell.* Let L be an odd length greater than one of the optimal path, which is unknown. Then, we consider a window W of size $L \times L$ and centered in c. The exceptional cases of having c at the border of S are now disregarded for simplification, but should be taken into account in practical applications, as well. The window may shrink at borders.

Now we define a path $p : [0 : L-1] \to \mathbb{Z} \times \mathbb{Z}$ through c fulfilling the following constraints.

$$p(t) = (x_t, y_t) \in W \qquad (t = 0, 1, \ldots, L-1) \tag{3.27}$$

$$x_{t+1} - x_t = 1 \qquad (t = 0, 1, \ldots, L-2) \tag{3.28}$$

$$y_{t+1} - y_t \in \{-1, 0, 1\} \qquad (t = 0, 1, \ldots, L-2) \tag{3.29}$$

$$p(\lfloor L/2 \rfloor + 1) = c \tag{3.30}$$

where Equations (3.28) and (3.29) are constraints regarding the *step sizes.* Equation (3.30) fixes the center of path p to be identical c, i.e., the center cell. The accumulated squared amplitude distance is

$$D(p) := \sum_{t=0}^{L-1} \left(S\left(p\left(t\right)\right) - S\left(c\right)\right)^2. \tag{3.31}$$

The optimal path $\widehat{p}$, which minimizes the accumulated squared amplitude distance, is defined as

$$\widehat{p} := \operatorname*{argmin}_{p} \{D(p)\}. \tag{3.32}$$

A dynamic programming algorithm for finding $\widehat{p}$ is given in the following Section 3.3.1.1.

Keeping the definition of the optimal homogeneous path in mind, we can introduce the algorithm for computing the speech-silence segmentation. In the first step all cells in the spectrogram having high amplitudes are marked as speech candidates. All cells having relatively low amplitudes are not regarded. All remaining cells that neither have low amplitudes, nor are peak cells, are processed as follows. A small neighboring region is analyzed by extracting the optimal homogeneous path through each of these cells. The cell is classified as a speech candidate, if its optimal path lies within a heterogeneous area. In case of silence the path would lie within a homogeneous area. To discriminate between heterogeneous and homogeneous areas, the amplitude variance of the optimal path is calculated. The local amplitude variance of a path is defined as

$$v(p) := \sum_{t=0}^{L-1} \left(S(p(t)) - \mu_p\right)^2 \tag{3.33}$$

$$\mu_p := \frac{1}{L} \sum_{t=0}^{L-1} S(p(t)) \tag{3.34}$$

and is called the *amplitude-modulation* of the cell. In case of a heterogeneous area, the amplitude modulation is high. Thus, all remaining cells having a high amplitude-modulation are marked as speech candidates. The speech candidate cells in a spectrogram of an example word string are shown in Figure 3.4 (c). As illustrated, many speech candidates are recognized within regions of speech and only very few candidates are marked within silent regions. The resulting speech-silence segmentation is indicated, as well. Obviously, silent regions are recognized, but few speech regions are wrongly marked as silence, as well, because of missing speech candidates. Such errors are not as bad as silent regions classified as speech, because missing speech features do not affect further processing as much as random speech features produced from silent regions do. The pseudo code for the speech-silence segmentation is given in Algorithm 3.1. The fundamental idea of analyzing modulation features is based on a publication by Mubarak et al. from 2006 [20]. In that work, an approach for speech-music segmentation is described.

The algorithm employs 3 threshold parameters. $\tau_p \in (0,1)$ defines the relative peak threshold. Each spectrogram cell is marked as a speech candidate, if its amplitude relative to the maximal amplitude occurring in the spectrogram is higher than τ_p. Each cell having a higher amplitude variance than $\tau_v \in \mathbb{R}_{>0}$ is marked as a speech candidate. The threshold $\tau_n \in \mathbb{Z}_{>0}$ gives the minimal number of speech candidates in a spectrogram column to mark the spectral vector as originating from a speech wave. Empirical experiments showed that $\tau_p = 0.5, \tau_v = 1, \tau_n = 3$ works well in many cases. These parameters are used to obtain the results depicted in Figure 3.4.

Algorithm 3.1: Finding a speech-silence segmentation using modulation features.

```
Input:   Speech waveform
         Relative peak threshold $\tau_p$
         Amplitude variance threshold $\tau_v$
         Threshold for minimal number of speech candidates per column $\tau_n$
Output:  Speech-silence segmentation

Obtain logarithmically scaled spectrogram $S$ from the input speech waveform

Mark each peak cell in $S$ having a higher ratio of amplitude to the maximal amplitude
   than $\tau_p$ as speech candidates

Disregard all cells in $S$ with relatively low amplitudes

for each remaining cell $c$ in $S$ do
    Compute the optimal path $p$ through $c$ within a small window around $c$
    Compute the amplitude variance $v(p)$
    if $v(p)$ is greater than the threshold $\tau_v$ then
        Mark $c$ as a speech candidate
    end
end

for each spectral vector $t$ in $S$ do
    Count the number $n(t)$ of speech candidates in $t$
    if $n(t)$ is greater than the threshold $\tau_n$ then
        Classify $t$ as originating from a speech signal
    else
        Classify $t$ as originating from a silence signal
    end
end

Determine the speech-silence segmentation $q$ from the classified spectral vectors

return $q$
```

3.3.1.1 Finding the optimal path

We recapitulate that a path $p : [0 : L-1] \to \mathbb{Z} \times \mathbb{Z}$, within window W of size $L \times L$, has to fulfill the conditions stated in Equations (3.27) to (3.30). The optimal path $\widehat{p}$ is a path that minimizes the accumulated squared amplitude distance D, which is introduced in Equation (3.31). Thus, $\widehat{p} := \operatorname{argmin}_p \{D(p)\}$. The optimal path can be computed in three steps using a dynamic programming approach. In the first step, the spectrogram excerpt within window W is copied into a separate matrix $M : \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}$ of size $L \times L$. Outside the window, M is assumed to be identically ∞. Window cells that are unreachable because of the step size conditions can be set to ∞, as well. This step is illustrated for an 11×11 window and an example spectrogram in Figure 3.5 (a) and (b). In the second step, for each window cell, the squared distance to the center cell is computed.

$$M(x,y) := (M(x,y) - M(c_M))^2 \qquad (x,y = 0,1,\dots,L-1). \tag{3.35}$$

Here, the center cell in M has the local coordinates $c_M := (\lfloor L/2 \rfloor, \lfloor L/2 \rfloor)$. This is illustrated in Figure 3.5 (c). In the third step, beginning at the center cell, accumulated distances to the left- and right-hand sides are computed. This is done, by iterating correspondingly to the step size conditions of possible paths and taking the minimal accumulated distance in each step. For the left-hand side, this step can be expressed, for $x = \lfloor L/2 \rfloor - 1, \lfloor L/2 \rfloor - 2, \dots, 0$ and for $y = 0,1,\dots,L-1$, as

$$M(x,y) := M(x,y) + \min\{M(x+1,y-1), M(x+1,y), M(x+1,y+1)\}, \tag{3.36}$$

and for the right-hand side, for $x = \lfloor L/2 \rfloor + 1, \lfloor L/2 \rfloor + 2, \dots, L-1$ and for $y = 0,1,\dots,L-1$, as

$$M(x,y) := M(x,y) + \min\{M(x-1,y-1), M(x-1,y), M(x-1,y+1)\}. \tag{3.37}$$

Recall that for all exceptional cases of indexing M outside the window, the value of M is assumed to be ∞. From the *accumulated cost matrix* M, the optimal path can be computed using backtracking from the smallest element at the left and right sides, respectively, to the center cell c_M. The result for an example is given in Figure 3.5 (d). The resulting path is the optimal path $\widehat{p}$, which then can be transformed back into the coordinate system of the spectrogram, which is shown in Figure 3.5 (e).

Note that unreachable path cells in M that are set to ∞ in the first step stay ∞ in the whole procedure. They actually do not have to be processed at all, but the equations become simpler.

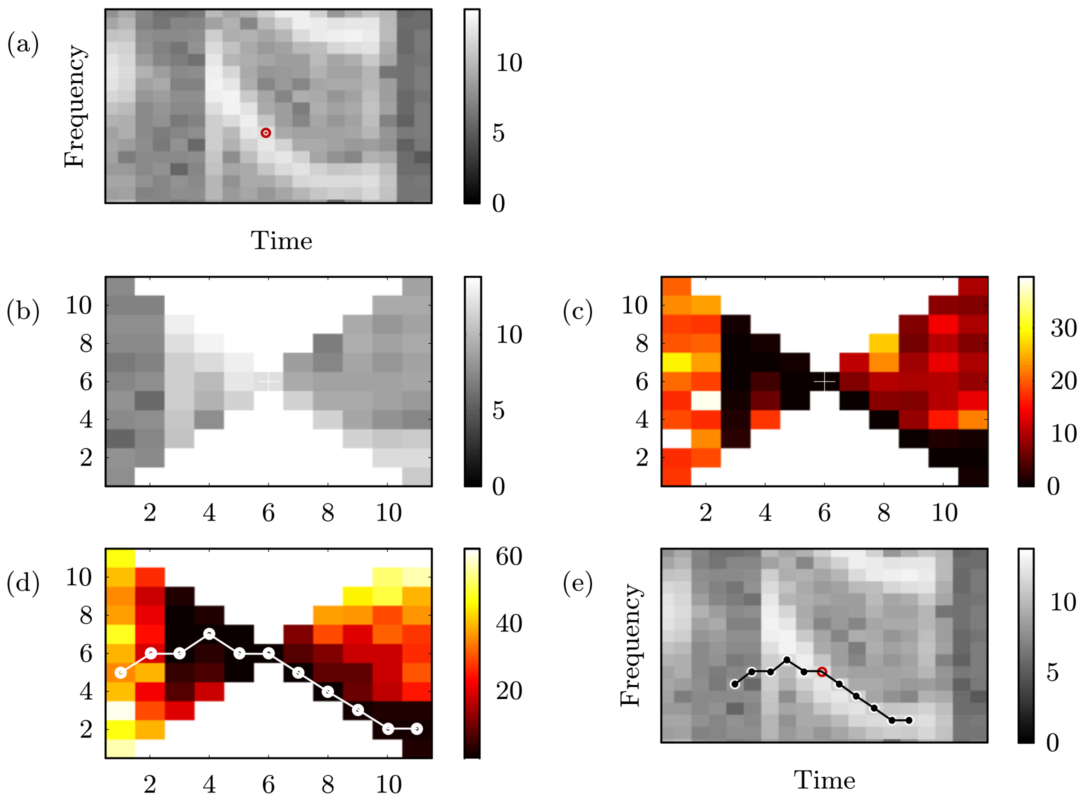

Figure 3.5: Finding the optimal homogeneous path. (a) Excerpt of a spectrogram and a marked center cell. Unreachable path cells are set to ∞. (b) Spectrogram excerpt within an 11×11 window around the center cell. (c) Squared amplitude distances to the center cell. (d) Accumulated distances and optimal cost-minimizing path. (e) Optimal path in the spectrogram excerpt.

4 Information Retrieval for Speech

Given a query item, the problem of finding similar items within a set of known items is called *information retrieval*. A well-known system for text retrieval for the Internet is Google, described in [7]. It preprocesses each website and creates an efficient structure enabling to find it, when searching for text given on the website. The matching results are listed with a ranking, which defines the relevance of matching items. The advantage of this approach is, that the user does not have to browse for a long time, but can directly pick one of the best results and has probably found an item of interest. In this work, a comparable procedure is utilized to find similar word utterances of a given set. Given a query utterance, the speech retrieval system matches it with similar items of an annotated database of known word utterances. Each match has a rank quantifying the similarity between the query utterance and the matching word utterance. If the query utterance has a high similarity to a known database item, they might contain the same message, i.e., word. Figure 4.1 illustrates an example of relations between an unknown query item and known word utterances in the database.

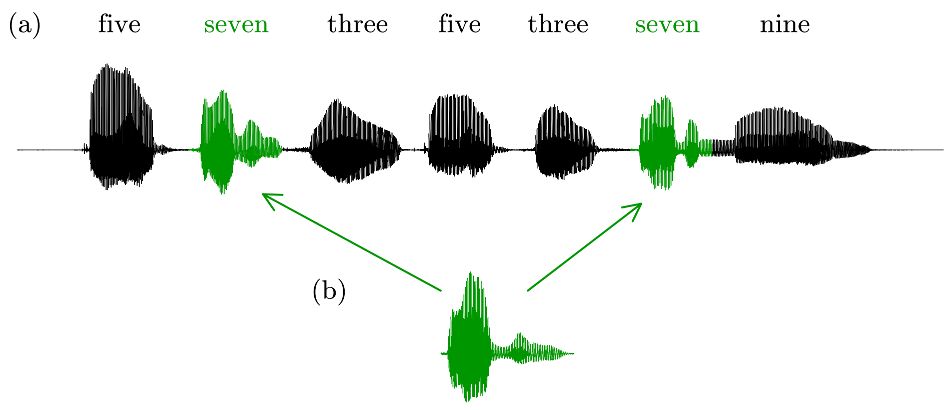

Figure 4.1: Concept of a retrieval system for word utterances. (a) Annotated database of known word utterances. (b) Unknown query utterance. Each arrow illustrates a relation between the unknown query utterance and a similar word utterance in the database. Here, the query is identified as the word „seven“.

The speech retrieval system used in this work is based on preprocessing the database and the query. First, the query is uniformly framed, then each frame is transformed into its feature representation. The database is given as a set of features, as well. Then, an efficient method for quantifying the similarity between each window and all items in the database is used. The method is introduced in Section 4.1. A step-by-step procedure for the speech retrieval system is described in Section 4.2. The evaluation of the performance of the speech retrieval system is based on tools and measurements that are defined in Section 4.3.

4.1 Dynamic Time Warping

Dynamic time warping (DTW), also referred to as *classical DTW*, is a well-known technique that can be used to determine the similarity between two utterances, even in case of different durations. It is described in numerous publications, for example in [21]. Since two utterances of the same word can strongly differ in the time-domain, they first have to be time-aligned. DTW determines the optimal time alignment by means of a dynamic programming approach that is similar to the one for finding the optimal path in Section 3.3. If the alignment is found, the similarity between the words can be measured by accumulating the costs along the alignment. A low accumulated cost means high similarity. DTW is utilized in various scenarios to time-align speech, for example, to identify Sinhala words spoken in isolation in [25], to recognize speech in [1, 18, 27], to align speech to text in [8] and for key-phrase detection in [33].

$\mathcal{F}$ depicts a *feature space*, which is a set including all possible feature vectors. Let $X := (x_1, x_2, \ldots, x_N)$ and $Y := (y_1, y_2, \ldots, y_M)$, with $x_i, y_i \in \mathcal{F}$ are feature vectors, be two feature sequences of lengths N and M, respectively. N and M have the relationship, that $\lfloor M/2 \rfloor < N \leq M \cdot 2 - 1$ (see Figure 4.2 depicting legal values for N and M). Then a *local cost measure* is defined as $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}$. The *local cost matrix* $C \in \mathbb{R}_{\geq 0}^{N \times M}$, with $C(n,m) := c(x_n, y_m)$, contains the local costs for all pairs of feature vectors.

We recapitulate that the goal of DTW is to find an optimal alignment. It can be described as a *warping path* within C. This is a sequence $p := (p_1, p_2, \ldots, p_L)$ with $p_\ell := (n_\ell, m_\ell) \in [1:N] \times [1:M]$ of pairs of feature vectors fulfilling the constraints

$$p_1 = (1,1)\,,\ p_L = (N,M) \tag{4.1}$$

$$p_{\ell+1} - p_\ell \in \{(1,1)\,,(2,1)\,,(1,2)\} \qquad (\ell = 1, 2, \ldots, L-1) \tag{4.2}$$

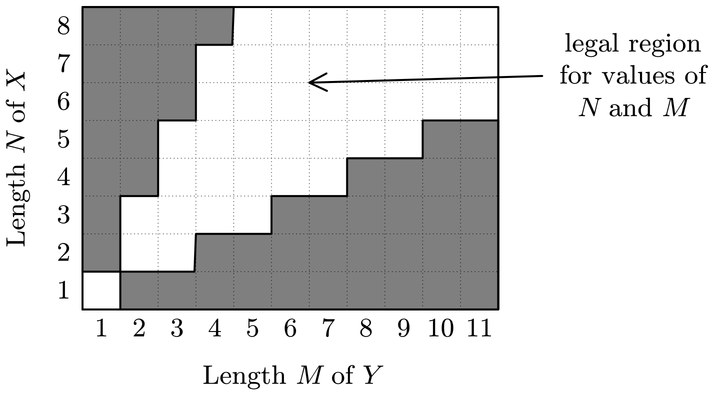

Figure 4.2: Legal region for feature sequence lengths N and M of X and Y, respectively.

where Equation (4.1) defines the boundaries of the warping path and Equation (4.2) gives constraints regarding the step sizes. The optimal alignment is represented by a warping path minimizing the *total cost*, which is an accumulation of the local costs along the path. It is defined for X and Y as

$$c_p\left(X,Y\right) := \sum_{\ell=1}^{L} c\left(x_{n_\ell}, y_{m_\ell}\right) = \sum_{\ell=1}^{L} C\left(n_\ell, m_\ell\right). \tag{4.3}$$

Thus, the optimal warping path can be mathematically expressed as

$$\widehat{p} := \operatorname*{argmin}_{p} \left\{ c_p\left(X,Y\right) \mid p \text{ is a warping path} \right\}. \tag{4.4}$$

The total cost of the optimal warping path is expressed as $c_{\widehat{p}}\left(X,Y\right) =: \mathrm{DTW}\left(X,Y\right)$ and is called the *DTW cost*. In the following, an efficient algorithm is described for computing the optimal path $\widehat{p}$ from a given local cost matrix C using dynamic programming. It is based on the theorem, that the DTW cost $\mathrm{DTW}\left(X,Y\right)$ can be determined by iteratively accumulating the local costs for $m = 2,3,\ldots,M$ and $n = 2,3,\ldots,N$ regarding the step size conditions. This leads to the *accumulated cost matrix*. Initially, let $A := C$ be a matrix containing the local costs. Set $A(n,1) := \infty, n = 2,3,\ldots,N$ and $A(1,m) := \infty, m = 2,3,\ldots,M$. If A is indexed outside the boundaries, the value is assumed to be ∞. Then, for $n = 2,3,\ldots,N$ and $m = 2,3,\ldots,M$ iterate

$$A\left(n,m\right) := A\left(n,m\right) + \min\left\{A\left(n-1,m-1\right), A\left(n-2,m-1\right), A\left(n-1,m-2\right)\right\}. \tag{4.5}$$

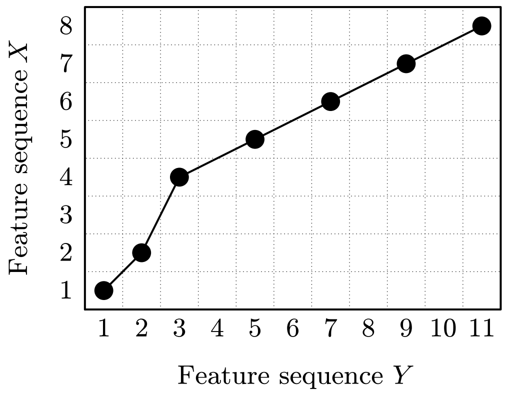

Figure 4.3: A warping path fulfilling the boundary and step size constraints.

Theorem. The matrix A contains the DTW costs for the optimal warping paths from cell $(1,1)$ to any other possible position. Let $X\,(a:b) := (x_a, x_{a+1}, \ldots, x_b)$, where $a \leq b$. Considering the matrix A, $\mathrm{DTW}\,(X\,(1:n)\,, Y\,(1:m)) = A\,(n,m)$ for $n \in [1:N]$ and $m \in [1:M]$, where $\lfloor m/2 \rfloor < n \leq m \cdot 2 - 1$, and particularly $\mathrm{DTW}\,(X,Y) = A\,(N,M)$. Implying that the matrix A contains the DTW cost of the optimal warping path $\widehat{p}$.

Proof. Obviously $\mathrm{DTW}\,(X\,(1:1)\,, Y\,(1,1)) = c\,(x_1, y_1)$. Now let $n > 1$ or $m > 1$, where $\lfloor m/2 \rfloor < n \leq m \cdot 2 - 1$, and let $p = (p1, \ldots, p_{L-1}, p_L)$ be the optimal warping path for $X\,(1:n)$ and $Y\,(1:m)$. Then $p_L = (n,m)$ according to the boundary condition. Let $(a,b) := p_{L-1}$, then $(a,b) \in \{(n-1,m-1)\,, (n-1,m-2)\,, (n-2,m-1)\}$ according to the step size constraints. Since p is an optimal warping path for X and Y, $(p_1, \ldots, p_{L-1})$ must be an optimal warping path for $X\,(1:a)$ and $Y\,(1:b)$, as well. Since $\mathrm{DTW}\,(X\,(1:n)\,, Y\,(1:m)) = c_{p_1,\ldots,p_{L-1}}\,(X\,(1:a)\,, Y\,(1:b)) + c\,(x_n, y_m)$, the accumulated cost matrix yields the claims. □

The optimal warping path can be obtained from the accumulated cost matrix by means of backtracking beginning at cell (N,M) down to $(1,1)$. Starting at $p_L := (N,M)$ perform recursively

$$p_{\ell-1} := \operatorname{argmin}\,\{A\,(n-1,m-1)\,, A\,(n-2,m-1)\,, A\,(n-1,m-2)\} \qquad (4.6)$$

until $(n,m) = (1,1)$. Since $\lfloor M/2 \rfloor < N \leq M \cdot 2 - 1$ holds, using the step size conditions each possible warping path begins at cell $(1,1)$, where the backtracking terminates. The running time of the algorithm lies in $\mathcal{O}\,(N \cdot M)$.

4.1.1 Subsequence DTW

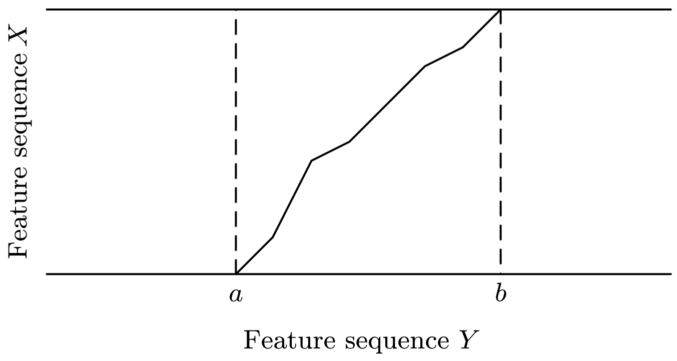

Figure 4.4: Warping path in subsequence DTW. Indices a and b show the beginning and the end of the subsequence, respectively.

In this work, one feature sequence comes from a database of known word utterances. Since the database contains a large number of utterances, the feature sequence is long, as well. The problem is to find a *subsequence* in the database feature sequence that best fits a given unknown query frame. Thus, a method for determining a subsequence of the database, which can be optimally time-aligned with minimal total cost, is desired. In this case, classical DTW is not feasible, because only a single global warping path can be found. *Subsequence DTW* is a variant of DTW, altering the computation of the optimal warping path to allow for time-aligning subsequences of the database with a short query. It is described in [21] and used, for example, for key-phrase detection in [33].

Let $X := (x_1, x_2, \ldots, x_N)$ and $Y := (y_1, y_2, \ldots, y_M)$ be the two feature sequences, where X is short and Y is long, i.e., $N \ll M$. The goal is to find a subsequence from feature vector index $\widehat{a}$ to $\widehat{b}$, where $1 \leq \widehat{a} \leq \widehat{b} \leq M$, of Y, i.e., $Y\left(\widehat{a} : \widehat{b}\right) := (y_{\hat{a}}, y_{\hat{a}+1}, \ldots, y_{\hat{b}})$, where $Y\left(\widehat{a} : \widehat{b}\right)$ and X can be time-aligned with minimal total cost. The goal can be expressed as finding the indices

$$\left(\widehat{a}, \widehat{b}\right) := \underset{(a,b):1\leq a\leq b\leq N}{\operatorname{argmin}} \left\{\mathrm{DTW}\left(X, Y\left(a : b\right)\right)\right\}, \tag{4.7}$$

where the DTW cost is based on the classical DTW approach. Thus, $\mathrm{DTW}\left(X, Y\left(a : b\right)\right) = c_{\hat{p}}\left(X, Y\left(a : b\right)\right)$ holds, where $\widehat{p}$ is the optimal warping path for X and the subsequence of Y. An important step for finding the optimal subsequence is the computation of the accumulated cost matrix A. The procedure is similar to the one described for classical

DTW. We add additional rows to the local cost matrix that are zero, to allow for all paths beginning somewhere at the bottom of the local cost matrix and end at the top. Thus, the costs are now accumulated within locally optimal regions of Y instead of a global warping. This is done as follows. A is initialized with the local cost matrix C, where $C(n,m) = c(x_n, y_m)$ concatenated with a row of zeros, i.e., $A(0,m) = 0$ and $A(n,m) = C(n,m)$ for $n = 1,2,\dots,N$ and $m = 1,2,\dots,M$. This means, A is of size $N+1 \times M$. For the exceptional case that A is indexed outside the boundaries, the value is assumed to be ∞. The accumulated costs are now iteratively computed, for $n = 1,2,\dots,N$ and $m = 1,2,\dots,M$, by

$$A(n,m) := A(n,m) + \min\left\{A(n-1,m-1), A(n-2,m-1), A(n-1,m-2)\right\}. \tag{4.8}$$

In the classical DTW algorithm, the DTW cost can be found at a single position in the accumulated cost matrix, where all possible warping paths end. In the subsequence variant, the DTW costs are all values at the top of the matrix, i.e., $A(N,m)$ for $m = 1,2,\dots,M$. The minimal DTW cost is

$$\mathrm{MINDTW}(X,Y) := \min_{m=1,2,\dots,M}\left\{A(N,m)\right\}. \tag{4.9}$$

The optimal m defines the right border of the optimal subsequence of Y with respect to X, i.e.,

$$\widehat{b} := \operatorname*{argmin}_{m=1,2,\dots,M}\left\{A(N,m)\right\}. \tag{4.10}$$

The index $\widehat{a}$ can then be found via backtracking beginning at position $\left(N,\widehat{b}\right)$ in the matrix A back to the bottom of A. Let the warping path p end at $p_L := \left(N,\widehat{b}\right)$ and recursively compute

$$p_{\ell-1} := \operatorname{argmin}\left\{A(n-1,m-1), A(n-2,m-1), A(n-1,m-2)\right\} \tag{4.11}$$

until $n \leq 1$. Now, the index $\widehat{a}$ is the position of m, where the algorithm stops. A warping path in subsequence DTW is shown in Figure 4.4. This leads to the optimal subsequence $\left(\widehat{a},\widehat{b}\right)$. By setting the minimal DTW cost to ∞ in the matrix A and performing the procedure again, the second-best match can be found. Iterating these steps leads to a ranked list of matching subsequences.

4.1.2 Local Cost Measure

We now specify a local cost measure for subsequence DTW. Considering the definition of the local cost matrix $C(n,m) := c(x_n, y_m)$, the local cost measure is given as $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}$ for a given feature space $\mathcal{F}$. A common method for comparing two D-dimensional real-valued feature vectors x and y is the *cosine distance*. It is also used in a DTW-based retrieval scenario in [15] and is defined as

$$c_{\cos}(x,y) := \frac{\langle x,y\rangle}{\|x\|_2 \cdot \|y\|_2} = \frac{1}{\|x\|_2 \cdot \|y\|_2} \cdot \sum_{i=0}^{D-1} x(i) \cdot y(i) = \cos(\sphericalangle(x,y)) . \qquad (4.12)$$

It gives the cosine of the angle between x and y.

4.2 Speech Retrieval System for Word Utterances

In this work, a speech retrieval system for word utterances is developed. The query speech waveform is uniformly framed and windowed using the Hann window. Each window is then processed to determine the speech features, which includes a speech-silence segmentation and the extraction of either MFCCs or LPC features. Following, the features are matched with the database features using subsequence DTW to obtain a list of best-matching word utterances.

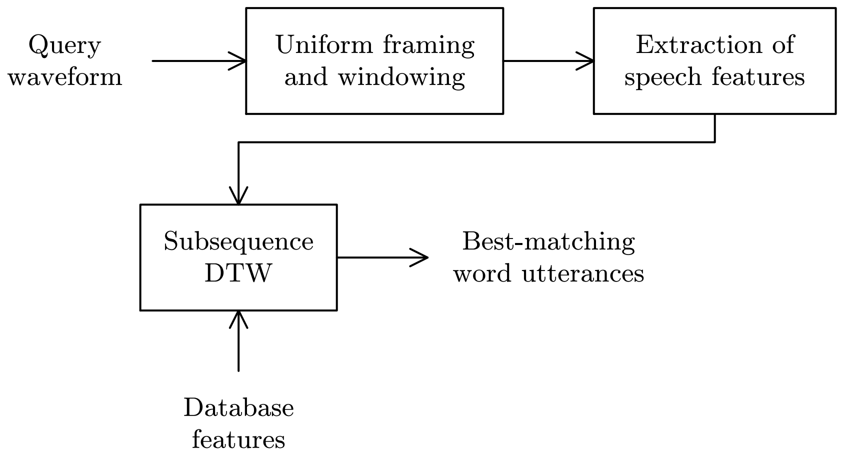
Query waveform
Uniform framing and windowing
Extraction of speech features
Subsequence DTW
Best-matching word utterances
Database features

Figure 4.5: Step-by-step procedure of the retrieval system for word utterances.

4.3 How to Evaluate the Speech Retrieval System

Subsequence DTW has been introduced as an information retrieval technique for word utterances. It can be used to match a query utterance with similar word utterances from a database. The matches are given as a ranked list, i.e., each match is assigned to a ranking value, indicating the amount of similarity. The performance of the system is evaluated in this work by iteratively processing numerous word strings. We recapitulate that the retrieval system used in this work initially uniformly frames the given string and queries each frame using subsequence DTW. Thus, for each frame a best-ranked match can be determined. This leads to a set of retrieved words that the system predicts to be contained in the original string. If a predicted word actually is contained in the original string it is classified as a hit, otherwise it is a mismatch. This Section introduces several techniques for evaluating the performance of the system by means of this classification.

4.3.1 Precision and Recall

In the following, a common method for evaluating information retrieval systems is introduced with respect to the special case of the subsequence-DTW-based system used in this work. Further details can be found, for instance, in [4, 24]. Let $\mathcal{P}(X) := \{a_1, a_2, \dots, a_N\}$ be the set of predicted words a_i the system retrieves from an input string X and let $X := \{x_1, x_2, \dots, x_M\}$ be the original set of words x_i. The desired case is that $\mathcal{P}(X) = X$, because it implies a correct performance of the system. In practical applications, this is often not true. Four possible cases may occur. The first case is that a word a_i is contained in X, then a_i can be classified as a *true positive.* If the predicted word a_i is not contained in X it is called a *false positive.* In case of a word x_i contained in X that is missed in the set of predicted words $\mathcal{P}(X)$, x_i is a *false negative.* All words not predicted and not contained in X are called *true negatives.*

The numbers of true positives (tp), false positives (fp) and false negatives (fn) are accumulated while processing numerous strings by the retrieval system. A measurement for the proportion of real positive cases that are correctly predicted as positive is the *recall* or *sensitivity* of the system. It is defined as

$$\text{recall} := \frac{\text{tp}}{\text{tp} + \text{fn}}. \tag{4.13}$$

It reflects the number of relevant cases the system picks up. A measurement for the *precision* or *confidence* determines the proportion of predicted positive cases that are real positives. It can be expressed in terms of the definitions given above as

$$\text{precision} := \frac{\text{tp}}{\text{tp} + \text{fp}}. \tag{4.14}$$

Both, precision and recall, are values between zero and one. A desired property of an information retrieval system is to obtain a high precision and a high recall.

4.3.2 F-measure

Commonly, the two values of precision and recall are combined to obtain a single value quantifying the performance of an information retrieval system. The advantage is that when changing a parameter of the evaluated system, a two-dimensional diagram can show the effect of the parameter changes on the performance. Typically the harmonic mean of precision and recall is considered. It is called the *F_1-measure* and is defined as

$$\text{F}_1 := 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{4.15}$$

The general *F_β-measure* was introduced by van Rijsbergen in [28] and is defined as

$$\text{F}_\beta = \left(1 + \beta^2\right) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}. \tag{4.16}$$

By choosing β to be greater than one, recall is weighted higher than precision.

4.3.3 Confusion Matrix

We recapitulate that the introduced speech retrieval system automatically matches each frame of a given input word string with similar word utterances in a database. Using a given alignment of the input string as the *ground truth*, for each frame the correct content can be estimated. Thus, for each frame a predicted word and an actual word according to the ground truth alignment are known. This information can be recorded for all tests during the evaluation process and summed up in a matrix. The scheme for W different words is given in Table 4.1, where $n : [1 : W] \times [1 : W] \to \mathbb{N}$ contains the number $n(p, a)$ of occurrences, where p is the index of the predicted word and a the actual word. The matrix $(n\,(p, a))_{1 \leq p,a \leq W}$ is called *confusion matrix*. A similar definition can be found in [4].

		Predicted word			
		word_1	word_2	$\cdots$	word_W
Actual word	word_1	$n(1,1)$	$n(1,2)$	$\cdots$	$n(1,W)$
	word_2	$n(2,1)$	$n(2,2)$	$\cdots$	$n(2,W)$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	word_W	$n(W,1)$	$n(W,2)$	$\cdots$	$n(W,W)$

Table 4.1: Scheme of a confusion matrix.

5 Retrieval-based Speech Recognition

Automatic speech recognition (ASR) is the process of determining linguistic information given in a speech waveform by means of a computer system or electronic circuits. In this work, we will only consider computer systems. It is also referred to as *speech to text (STT)*, because it converts the speech waveform into text. It is an important field of research in computer science, since 1952, when the first spoken digit recognizer was published. Further details can be found, for instance, in [10, 12, 26, 29].

In general, spectral properties of speech can differ from speaker to speaker. An ASR system that is only able to deliver a suitable STT performance for specific speakers is called *speaker dependent*. However, since in general ASR systems should cope with every speaker, a desired property is *speaker independence*. The problem of a speaker independent scenario is more complex than a speaker dependent scenario, because it assumes a higher degree of flexibility and thus demands a smarter concept or additional processing time.

Many concepts for ASR systems have been published within the past six decades. In 1962, a recognizer has been proposed that utilizes a time-aligning technique to identify words spoken in isolation. The first ASR systems were able to cope only with small vocabularies, i.e., 10–100 words. In 1981, Myers and Rabiner proposed a speaker-independent technique that is able to recognize *connected words* by means of linear predictive analysis and a DTW-based pattern recognition technology [22, 23]. The problem of connected word recognition is not identical to the more general *continuous speech recognition*. The fundamental difference is that each word string can be seen as a sequence of isolated words. All of them can be represented completely by a set of references. The ASR system by Myers and Rabiner is based on an algorithm building up the test string word-by-word by measuring the similarity of short sequences of LPC coefficients to each item in a set of references by means of an optimal time-alignment. Later in the 1980's, large vocabulary (over 20,000 words) continuous speech recognizers have been proposed that are based on *hidden Markov models (HMM)* and stochastic language modeling. Instead of directly comparing with references, these techniques describe the phoneme dictionary by means of a model that is trained according to a speech database. Further details on the history of ASR are given, for example, in [14].

In this work, a speaker-independent retrieval-based connected word recognition system is considered. The goal is to find a mapping for each word contained in a test string to a known word utterance reference within a database. If such a mapping is given, the test string can be annotated by identifying each word with the most similar reference. Figure 5.1 illustrates this mapping by means of an example. Since the proposed DTW-based pattern recognition technology by Myers and Rabiner builds up the string word-by-word by iterative comparisons with the references, such a mapping can be achieved as a result. The technique is detailed in Section 5.1. In this work, three different connected word recognizers based on this technique are developed and evaluated. They are introduced in Section 5.2. Common methods to evaluate ASR systems are introduced in Section 5.3.

A possible application scenario for such an ASR system can be a human-computer interaction system, which enables, for instance, controlling a car computer by means of voice. This has the advantage that the driver does not need to use a keyboard or touch screen, which are controlled by the hands and thus can lead to dangerous situations in traffic. Voice control is the most intuitive interface, since speech is the primary means of communication between people. Another possible application scenario is the automatic annotation of a speech database.

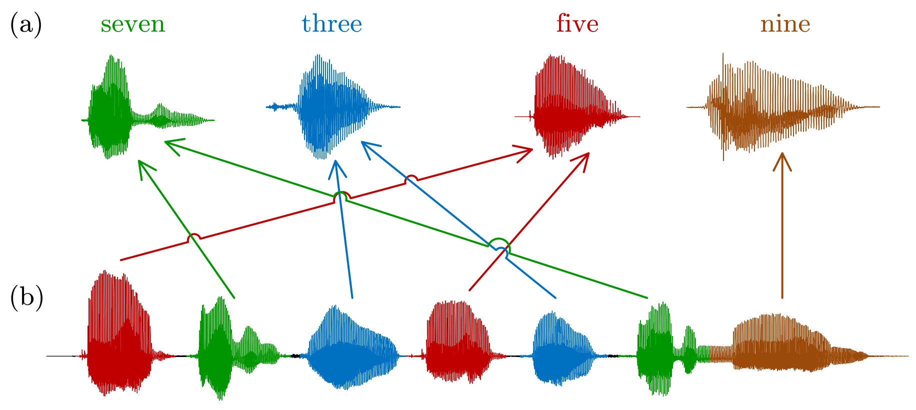

Figure 5.1: Concept of the goal of a retrieval-based ASR system. (a) Set of reference word utterances. (b) Test string containing the message „five, seven, three, five, three, seven, nine“. The colored arrows show a mapping from each word utterance within the test string to the most similar reference utterance. Given the mapping, the test string can be identified.

5.1 Level-building DTW

In 1981, Myers and Rabiner proposed a technique for annotating a test string by means of a word recognizer. The technology is introduced in [23] and is called *level-building DTW*. The algorithm finds the optimal time-alignment between the test string and concatenated reference word utterances from a database. The DTW cost is minimized by iteratively choosing the most similar reference word to each word in the test string. Each concatenated reference is considered as a level. The name of the algorithm is motivated by the process that word-by-word portions of the test string are compared with all reference words. This leads to a level-wise creation of the concatenated reference string. Since all references are known, an automatically produced reference string can be used in an STT step to annotate the test string as shown in Figure 5.1.

Let $Y = (y_1, y_2, \ldots, y_M)$, with $y_i \in \mathcal{F}$, be an input feature sequence originating from a connected word sequence, called the *test sequence* of length M. R_v for $v = 1, 2, \ldots, V$ are V references, where each R_v is a sequence of N_v feature vectors, i.e., $R_v(n) \in \mathcal{F}$. Level-building DTW finds a *super reference pattern* $R^s_{q(1)q(2)\ldots q(L)}$, which is produced by concatenating the reference patterns $R_{q(1)}, R_{q(2)}, \ldots, R_{q(L)}$ of L selected references. The sequence with elements $q(\ell) \in [1 : V]$, with $\ell \in [1 : L]$, defines the order of the concatenated references. The super reference pattern is abbreviated by R^s and can be mathematically expressed as

$$R^s(n) = \begin{cases} R_{q(1)}(n - h_0), & \text{if } h_0 < n \leq h_1 \\ R_{q(2)}(n - h_1), & \text{if } h_1 < n \leq h_2 \\ \vdots & \\ R_{q(\ell)}(n - h_{\ell-1}), & \text{if } h_{\ell-1} < n \leq h_\ell \\ \vdots & \\ R_{q(L)}(n - h_{L-1}), & \text{if } h_{L-1} < n \leq h_L \end{cases} \qquad (n \in [1 : h_L]), \tag{5.1}$$

where the function h_ℓ gives the accumulated length of the first ℓ reference feature sequences, i.e.,

$$h_\ell := \sum_{k=1}^{\ell} N_{q(k)} \tag{5.2}$$

$$h_0 = 0. \tag{5.3}$$

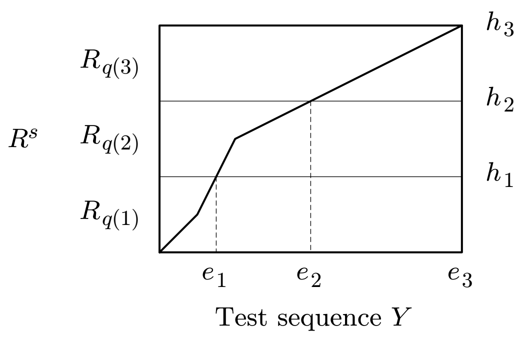

Figure 5.2: Example of a warping path in level-building DTW for a super reference pattern of 3 concatenated references.

Given a feature space $\mathcal{F}$ and a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}$, the optimal warping path $\widehat{p}$ can be computed according to the procedure of classical DTW. By means of the introduced definitions, the goal of level-building DTW is to find an optimal super reference pattern $\widehat{R^s}$, which minimizes the DTW cost $\mathrm{DTW}\left(\widehat{R^s}, Y\right)$ according to classical DTW. Figure 5.2 shows an example of a possible warping path for a given test sequence Y and a super reference pattern R^s consisting of a concatenated sequence of 3 references $R_{q(1)}, R_{q(2)}$ and $R_{q(3)}$.

The level-building algorithm is based on the assumption that if the best sequence of ℓ references is found, then these ℓ references will be the first ℓ references within the optimal super reference pattern. For the following theorem, the definition of the ending frame of a super reference pattern of level ℓ has to be introduced. It is the position $e_\ell \in [1 : M]$, where h_ℓ intersects the optimal warping path. It defines the ending frame of the time-alignment between the reference of the ℓ^{th} level $R_{q(\ell)}$ in the super reference pattern R^s and the corresponding subsequence of the test sequence Y. e_ℓ is depicted in Figure 5.2 for the example of a 3-level super reference pattern.

Theorem. Let $R^s_{r_1,r_2,\ldots,r_\ell}$ be a super reference pattern that minimizes the DTW cost up to level ℓ, i.e., $\mathrm{DTW}\left(R^s_{r_1,r_2,\ldots,r_\ell}, Y(1 : e_\ell)\right) \leq \mathrm{DTW}\left(R^s_{s_1,s_2,\ldots,s_\ell}, Y(1 : e_\ell)\right)$ for any possible super reference pattern $R^s_{s_1,s_2,\ldots,s_\ell}$. If $R^s_{r_1,\ldots,r_\ell}$ ends at the same frame e_ℓ as $R^s_{s_1,\ldots,s_\ell}$, concatenating both with a reference $R_{r_{\ell+1}}$ yields the same relation, i.e., $\mathrm{DTW}\left(R^s_{r_1,r_2,\ldots,r_\ell,r_{\ell+1}}, Y(1 : e_{\ell+1})\right) \leq \mathrm{DTW}\left(R^s_{s_1,s_2,\ldots,s_\ell,r_{\ell+1}}, Y(1 : e_{\ell+1})\right)$ for any $R^s_{s_1,s_2,\ldots,s_\ell}$ and any $R_{r_{\ell+1}}$. This theorem is not true for the case that $R^s_{r_1,\ldots,r_\ell}$ does not end at the same frame e_ℓ as $R^s_{s_1,\ldots,s_\ell}$.

Proof. Let p denote the optimal warping path, then

$$\begin{aligned}
\mathrm{DTW}\left(R^s_{s_1,s_2,\ldots,s_\ell,r_{\ell+1}}, Y\left(1:e_{\ell+1}\right)\right) &= c_p\left(R^s_{s_1,s_2,\ldots,s_\ell,r_{\ell+1}}, Y\left(1:e_{\ell+1}\right)\right) && (5.4)\\
&= c_p\left(R^s_{s_1,s_2,\ldots,s_\ell}, Y\left(1:e_\ell\right)\right) + c_p\left(R_{r_{\ell+1}}, Y\left(e_\ell:e_{\ell+1}\right)\right) - c\left(p\left(e_\ell\right)\right) && (5.5)\\
&= \mathrm{DTW}\left(R^s_{s_1,s_2,\ldots,s_\ell}, Y\left(1:e_\ell\right)\right) + c_p\left(R_{r_{\ell+1}}, Y\left(e_\ell:e_{\ell+1}\right)\right) - c\left(p\left(e_\ell\right)\right) && (5.6)\\
&\geq \mathrm{DTW}\left(R^s_{r_1,r_2,\ldots,r_\ell}, Y\left(1:e_\ell\right)\right) + c_p\left(R_{r_{\ell+1}}, Y\left(e_\ell:e_{\ell+1}\right)\right) - c\left(p\left(e_\ell\right)\right) && (5.7)\\
&\Leftrightarrow \mathrm{DTW}\left(R^s_{r_1,r_2,\ldots,r_\ell}, Y\left(1:e_\ell\right)\right) \leq \mathrm{DTW}\left(R^s_{s_1,s_2,\ldots,s_\ell}, Y\left(1:e_\ell\right)\right) \quad \square && (5.8)
\end{aligned}$$

Level-building DTW roughly proceeds as follows. First, the local cost matrices $C_v \in \mathbb{R}^{N_v \times M}_{\geq 0}$, where $v \in [1:V]$, are computed. This is done by evaluating the local cost $C_v(n,m) = c\left(R_v(n), y_m\right)$ for $n = 1,2,\ldots,N_v$ and $m = 1,2,\ldots,M$. Given a maximal number of levels $L_{\max}$, for each level $\ell = 1,2,\ldots,L_{\max}$, each reference R_v for $v = 1,2,\ldots,V$, and for $m = 1,2,\ldots,M$, the minimal accumulated cost is computed using the minimal accumulated cost from the previous level. A backtracking pointer always documents the covered path and additionally for each minimal accumulated cost the corresponding reference is stored. Then, the optimal number of levels $\widehat{L} \in [1:L_{\max}]$, i.e., amount of words contained in the test phrase, is estimated by determining the minimal accumulated cost at position $m = M$. This value defines the DTW cost D. Utilizing the stored backtracking pointers, the endpoint sequence $e : \left[1:\widehat{L}\right] \to [1:M]$ for each level and the sequence of chosen cost-minimizing references $q : \left[1:\widehat{L}\right] \to [1:V]$ can be obtained. From these parameters the test string can be annotated. The whole procedure is detailed in Algorithm 5.1.

5.1.1 Running Time

In [23] Myers and Rabiner give some further details to improve the running time. This is done, for instance, by reducing the computation range of the accumulated cost. In case of considering these slight modifications, the following computation time can be achieved. We recapitulate that V is the number of references, $L_{\max}$ the maximal number levels and M is the length of the test sequence. Let $\overline{N}$ be the average length of a reference pattern, then $V \cdot L_{\max} \cdot \overline{N} \cdot M/3$ gives the average number of distance calculations. In general an upper bound is $\mathcal{O}\left(V \cdot L_{\max} \cdot \max_{v\in[1:V]}\left\{N_v\right\} \cdot M\right)$, where N_v is the length of the v^{th} reference.

Algorithm 5.1: Finding a cost-minimizing reference sequence.

```
Input:   References $R_v$    $(v = 1, 2, \ldots, V)$
         Test sequence $Y$
         Maximal number levels $L_{\max}$
Output:  Sequence $q$ of cost-minimizing references
         Sequence $e$ of endpoints
         DTW cost $D$

// Initialize variables
Call InitMatrices in Algorithm 5.2

// Main loop
minAccumulatedCostPerLevel, backtrackPerLevel, minReferencePerLevel
    = Call LevelBuildingIteration in Algorithm 5.3 using all initialized variables

// Find the optimal number of levels
$\widehat{L} := \mathrm{argmin}_{\ell\in[1:L_{\max}]} \{\mathrm{minAccumulatedCostPerLevel}\,(\ell, M)\}$

// DTW cost
$D := \mathrm{minAccumulatedCostPerLevel}\left(\widehat{L}, M\right)$

// Backtracking
// Find endpoints for each level
Init. array $e : \left[1 : \widehat{L}\right] \to [1 : M]$
$e_{\widehat{L}} := M$
for $\ell := \widehat{L} - 1, \widehat{L} - 2, \ldots, 1$ do
    $e_\ell := \mathrm{backtrackPerLevel}\,(\ell + 1, e_{\ell+1})$
end

// Find cost-minimizing references per level
Init. array $q : \left[1 : \widehat{L}\right] \to [1 : V]$
for $\ell := 1, 2, \ldots, \widehat{L}$ do
    $q\,(\ell) := \mathrm{minReferencePerLevel}\,(\ell, e_\ell)$
end

return $q, e, D$
```

Algorithm 5.2: Initialization function InitMatrices

```
// Initialize local cost matrices
Init. local cost matrices $C_v$ for $v = 1, 2, \ldots, V$

// Storage for minimal accumulated costs for each level
Init. matrix minAccumulatedCostPerLevel : $[0 : L_{\max}] \times [0 : M] \to \mathbb{R}_{\geq 0}$
Set all entries of minAccumulatedCostPerLevel to $\infty$
minAccumulatedCostPerLevel $(0, 0) := 0$

// Storage for accumulated costs for each reference
Init. matrix levelAccumulatedCostPerReference : $[1 : V] \times [1 : M] \to \mathbb{R}_{\geq 0}$
Set all entries of levelAccumulatedCostPerReference to $\infty$

// Storage for backtracking pointer for each reference
Init. matrix levelBacktrackPerReference : $[1 : V] \times [1 : M] \to \mathbb{Z}$
Set all entries of levelBacktrackPerReference to 0

// Storage for backtracking pointer for each level
Init. matrix backtrackPerLevel : $[0 : L_{\max}] \times [0 : M] \to \mathbb{Z}$
Set all entries of backtrackPerLevel to 0

// Storage for references providing minimal accumulated cost
Init. matrix minReferencePerLevel : $[0 : L_{\max}] \times [0 : M] \to \mathbb{R}_{\geq 0}$
```

Algorithm 5.3: Main function LevelBuildingIteration

```
// Level-building iteration
for $\ell := 1, 2, \ldots, L_{max}$ do
    for $v := 1, 2, \ldots, V$ do

        // Initilization
        Init. matrix accumulatedCost : $[0 : N_v] \times [0 : M] \to \mathbb{R}_{\geq 0}$
        Init. matrix backtrack : $[0 : N_v] \times [0 : M] \to \mathbb{Z}$
        for $m := 0, 1, \ldots, M$ do
            accumulatedCost $(0, m)$ := minAccumulatedCostPerLevel $(\ell - 1, m)$
            backtrack $(0, m) := m$
        end
        for $n := 1, 2, \ldots, N_v$ do
            for $m := 1, 2, \ldots, M$ do
                accumulatedCost $(n, m) := C_v(n, m)$
            end
        end

        // Compute accumulated cost
        Call ComputeAccumulatedCost in Algorithm 5.4

        // Store last row of accumulated cost matrix
        for $m := 1, 2, \ldots, M$ do
            levelAccumulatedCostPerReference $(v, m)$ := accumulatedCost $(N_v, m)$
            levelBacktrackPerReference $(v, m)$ := backtrack $(N_v, m)$
        end
    end

    // Store optimal configurations for this level
    for $m := 1, 2, \ldots, M$ do
        minReferencePerLevel $(\ell, m)$ :=
            $\mathrm{argmin}_v$ {levelAccumulatedCostPerReference $(v, m)$}
        minAccumulatedCostPerLevel $(\ell, m)$ :=
            levelAccumulatedCostPerReference (minReferencePerLevel $(\ell, m)$ , $m$)
        backtrackPerLevel $(\ell, m)$ :=
            levelBacktrackPerReference (minReferencePerLevel $(\ell, m)$ , $m$)
    end
end
```

Algorithm 5.4: Core function ComputeAccumulatedCost

```
// Compute accumulated cost
// The matrix values are assumed to be $\infty$ outside the boundaries
for $n := 1, 2, \ldots, N$ do
    for $m := 1, 2, \ldots, M$ do
        $(n', m') := \operatorname{argmin} \left\{ \begin{array}{l} \text{accumulatedCost}\,(n-1, m-1)\,, \\ \text{accumulatedCost}\,(n-2, m-1)\,, \\ \text{accumulatedCost}\,(n-1, m-2) \end{array} \right\}$
        $\text{accumulatedCost}\,(n, m) := \text{accumulatedCost}\,(n, m) +$
            $\text{accumulatedCost}\,(n', m')$
        $\text{backtrack}\,(n, m) := \text{backtrack}\,(n', m')$
    end
end
```

5.1.2 KNN Decision Rule

The level-building DTW concept introduced above provides a procedure to efficiently determine a sequence $q : \left[1 : \widehat{L}\right] \rightarrow [1 : V]$ of cost-minimizing references using level- and column-wise, i.e., for each $m \in [1 : M]$, determination of cost-minimizing references. However, in case of a speaker-independent word recognizer, Myers and Rabiner [22] propose a slight variation of this procedure to enhance the robustness. The fundamental assumption is that not just a single reference should be taken into account, but a number of K references. The accumulated costs for the K references are to be averaged for each level and each m, if the backtracking pointers are similar, i.e., both accumulated costs stem from the same area. In this work, the robustness could be improved significantly by considering K references from different speakers. Myers and Rabiner proposed K to be equal 2. The K references are referred to as nearest neighbors of the corresponding test sequence, thus they are called the *K-nearest neighbors (KNN)*. In terms of this expression, the level-wise and column-wise determination of minimal averaged accumulated costs is called the *KNN decision rule*.

5.1.3 Using LPC in Level-building DTW

We recapitulate that a test sequence $Y \in \mathcal{F}^M$ and references $R_v \in \mathcal{F}^{N_v}$, with $v \in [1 : V]$, are processed, whereas the feature space $\mathcal{F}$ is unspecified. Myers and Rabiner proposed using LPC coefficients in [22] for a suitable connected digit recognizer. Additionally, a concept for a local cost measure $c : \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$ in case of LPC suitable for word recognition is to be introduced. Since the space of LPC coefficients is complex, the definition of a suitable distance between a pair of LPC-based feature vectors is not trivial. Itakura discussed this problem in [13] published in 1975. Later, in 1980, Rabiner and Schmidt proposed a distance measure based on the concepts of Itakura in [27] suitable for DTW-based connected digit recognition. The concept is based on Itakura's definition of *prediction residuals*. The prediction residual is a measure for the model error made by LPC and is defined as

$$Q := v^\top X v, \tag{5.9}$$

where v is a column vector holding the LPC filter coefficients and X is the Toeplitz matrix from Equation (3.22) of the autocorrelation values. The distance measure proposed by Rabiner and Schmidt is defined for a reference item r and a test item t as

$$c_{\text{residual}}(v_r, v_t) := \ln\left(Q_r \cdot Q_t\right), \tag{5.10}$$

where

$$Q_r := v_r^\top X_t v_r \tag{5.11}$$
$$Q_t := v_t^\top X_t v_t = x_t^\top v_t. \tag{5.12}$$

x_t is a column vector holding the autocorrelation values. Equation (5.12) expresses a hint to efficiently compute the prediction residual Q_t of the test item.

5.2 Speech Recognition Based on Level-building DTW

Level-building DTW can be utilized for speaker-independent connected word recognition. In the setup of Myers and Rabiner [22] from 1981, first, the waveforms are *resampled* to 6.67 kHz, which means that frequencies higher than the Nyquist frequency are removed by a filter and afterwards the sampling rate is reduced by linear interpolation and a uniform sample deletion (further details are given, for example, in [26]). Then a mechanism for detecting endpoints of each word within the waveforms is used. Each 100 samples, i.e., 15 ms, a window of 300 samples, i.e., 45 ms, is processed to extract LPC features using the autocorrelation method with a prediction order of 8. This leads to a feature rate of 67 frames per second. Then level-building DTW is used to recognize a given test string. In the speaker-independent case the utilized references are based on 12 template word utterances from 6 male and 6 female speakers per word.

In this work, three different prototypes of speaker-independent word recognizers based on level-building DTW are developed. All of them use a resampling to 6.67 kHz and a feature extraction of windows of size 300 samples every 100 samples. The feature extraction is based on the speech-silence segmentation introduced in Section 3.3. Those feature vectors belonging to time positions containing no speech are erased. The features can be either MFCCs or LPC coefficients. Then the test sequence is processed using level-building DTW. The processing is illustrated in Figure 5.3. The three prototype recognizers use different references. Given an annotated database containing word utterances from different speakers, the first recognizer, called the *database recognizer*, uses the whole database as the reference set. The second mechanism, called the *template recognizer*, uses 12 template utterances from 6 male and 6 female speakers per word. The third recognizer is referred to as the *retrieval-based recognizer*, which in a first step uses the speech retrieval system described in Chapter 4 to find a set of references fitting the linguistic content given in the test phrase. These references are then employed in the level-building DTW algorithm to recognize the test string. The three different concepts are illustrated in Figure 5.4.

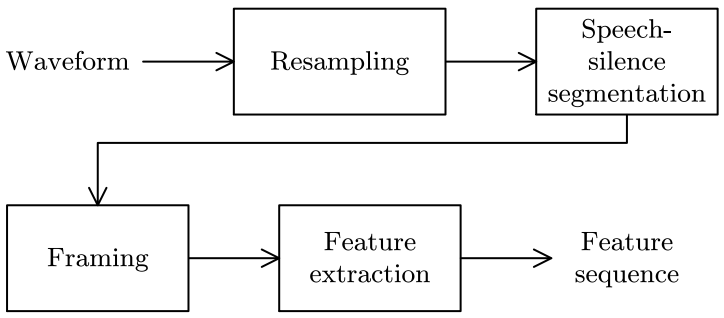

Figure 5.3: Preprocessing of a given waveform for the use in level-building DTW.

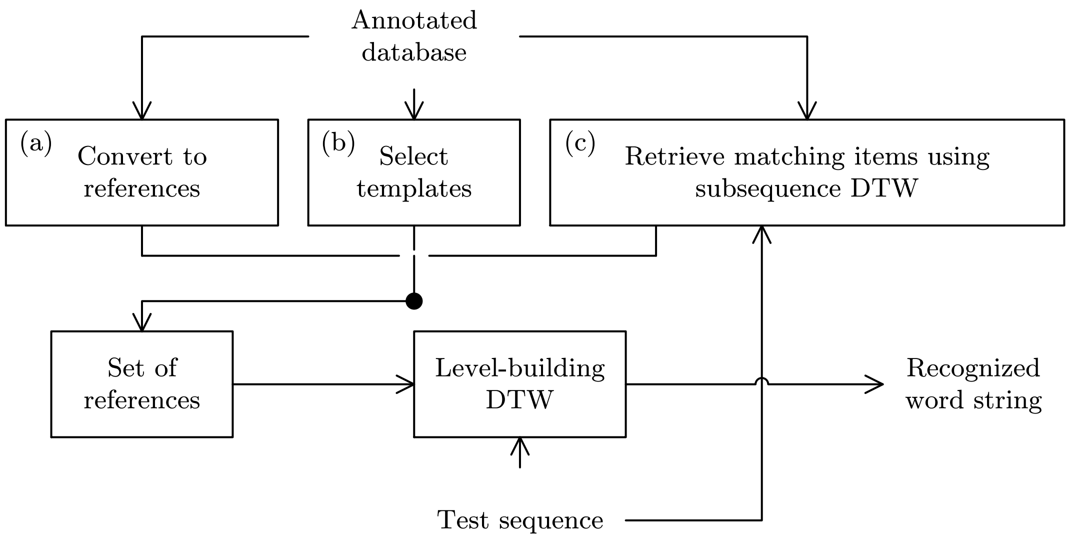

Figure 5.4: Three variants of word recognizers using different reference concepts. (a) Database recognizer. (b) Template-based. (c) Retrieval-based.

5.3 How to Evaluate ASR Systems

An automatic STT system, such as the level-building-DTW-based ASR system introduced above, transforms a given speech waveform into text. If the correct linguistic information is known as the ground truth, the performance of an STT system can be quantified by means of the following measurement. Commonly, the *word error rate (WER)* [12] is used to describe the accuracy of determining a certain word string. It gives the ratio of word errors generated by the STT system to the number of words according to the ground truth. Word errors are divided into three categories. If a word is predicted that is not contained in the actual word string, the error is referred to as a *word insertion*. Missing words are counted as *word deletions*. The third class of word errors are *word substitutions*. If the number of word insertions I, deletions D and substitutions S are given, the WER can be calculated by

$$\text{WER} := \frac{S + D + I}{W}, \tag{5.13}$$

where W is the number of actual words. Commonly, the WER is expressed as a percentage. Given an actual word string and a predicted word string resulting from an STT process, the automatic classification of word errors is not trivial. Both word strings have to be aligned to match correctly predicted words within both sequences. Word insertions and deletions lead to missing relations in the alignment. Word substitutions are aligned words that do not match. The three word error classes are depicted with the help of a small example in Figure 5.5. An efficient algorithm for computing the alignment and to classify and count word errors by means of dynamic programming is detailed in Algorithm 5.5.

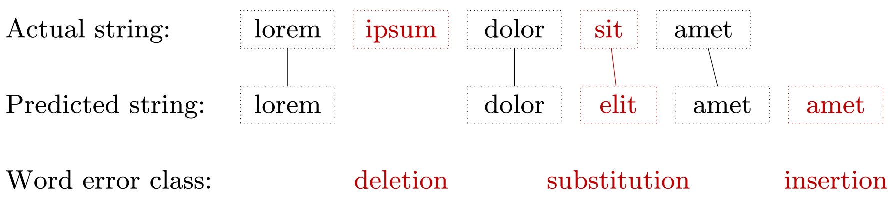

Figure 5.5: Word error classes depicted by means of a word alignment.

Algorithm 5.5: Automatic word error classification.

```
Input:    Non-empty reference word string $X = (x_1, x_2, \ldots, x_W)$
          Non-empty predicted word string $\mathcal{P}(X) = (p_1, p_2, \ldots, p_V)$
Output:   Number of word insertions $I$
          Number of word deletions $D$
          Number of word substitutions $S$

// Initialization
$I, D, S := 0$
Init. backtracking matrix $B : [0 : W] \times [0 : V] \rightarrow \mathbb{Z}$
Init. accumulated cost matrix $A : [0 : W] \times [0 : V] \rightarrow \mathbb{Z}$
Set all entries of $A$ to $\infty$ and set $A(0,0) := 0$

// Compute accumulated cost and backtracking information
for $i := 1$ to $W$ do
    for $j := 1$ to $V$ do
        $A(i,j) := \min \left\{ \begin{array}{lll} A(i-1,j)+1 & & \text{(deletion)} \\ A(i,j-1)+1 & & \text{(insertion)} \\ A(i-1,j-1) & \text{if } x_i = p_j & \text{(match)} \\ A(i-1,j-1)+1 & \text{if } x_i \neq p_j & \text{(substitution)} \end{array} \right\}$
        $B(i,j) := \begin{cases} 1 & \text{if deletion} \\ 2 & \text{if insertion} \\ 3 & \text{if match} \\ 4 & \text{if substitution} \end{cases}$
    end
end

// Backtracking and word error classification
optimal warping path $p = (p_L, p_{L-1}, \ldots, p_0)$, beginning with $p_L = (W, V)$
Iterate $p_{\ell-1} := \begin{cases} (i-1,j) & \text{if } B(p_\ell) = 1 & \text{(increment } D) \\ (i,j-1) & \text{if } B(p_\ell) = 2 & \text{(increment } I) \\ (i-1,j-1) & \text{if } B(p_\ell) = 3 & \\ (i-1,j-1) & \text{if } B(p_\ell) = 4 & \text{(increment } S) \end{cases}$
Terminate at $p_0 = (0,0)$

return $I, D, S$
```

Usually, there is a trade-off between STT accuracy and processing time. Parameter tweaks yielding a lower WER often accompany a longer processing time. Thus, when evaluating different configurations of parameters, a measurement for the processing time is necessary to indicate, for instance, considerably high increases in computation times. Commonly, the *real time factor (RTF)* is considered. Given a processing time P for evaluating a set of test strings of total time duration D the RTF is defined as the ratio

$$\text{RTF} := \frac{P}{D}. \tag{5.14}$$

6 Evaluations

In the previous Chapter 5, three different variants of speech recognition systems have been introduced. All concepts are implemented as prototype systems using MATLAB 7 with the help of VOICEBOX [3] by Mike Brookes and a utility library [9] by Daniel P. W. Ellis. The core functions are implemented in C and integrated into the MATLAB system using the interface MEX to provide a fast processing. The goal of this work is to find an optimal configuration, which delivers a suitable word recognition system. First, the speech retrieval system based on subsequence DTW is evaluated using various parameter configurations. Then, the three speech recognition systems are evaluated and compared with a state-of-the-art recognizer. All evaluations are speaker-independent using a large set of speech recordings introduced in Section 6.1. The speech retrieval evaluations are described in Section 6.2 and the speech recognition systems are evaluated in Section 6.3. Section 6.4 summarizes the evaluation results and gives an outlook for further research.

6.1 Aurora-2 Speech Corpus

For all evaluations, a large collection of speech recordings is used. It only contains word utterances of digits, i.e., „oh“, „one“, „two“, „three“, . . ., „eight“, „nine“ and „zero“, spoken by American English speakers. It is a *speech corpus* based on *Texas Instruments Digits (TIDigits)*, which is described in [16] published by Leonard in 1984. The used corpus has been processed by Hirsch and Pearce in 2000 to additionally provide real-world-noise-affected material. The processed version of TIDigits used in this work is called *Aurora-2* and is described in [11]. In this work, for the database, 2823 word strings uttered by 40 male and 40 female speakers are used. The test material consists of 770 word strings produced by 10 male and 10 female speakers. The number of words contained in a single word string lies in a range between 1 and 7. The speaker sets are disjoint to obtain a speaker-independent scenario. For the database only clean speech is used. The database is annotated and aligned using the *Hidden Markov Model Toolkit (HTK)* [31], which can be used to perform *forced alignment* to determine time regions of occurring words within an annotated speech waveform.

6.2 Speech Retrieval System

In this Section, the speech retrieval system based on subsequence DTW is evaluated. The goal of the evaluation is to obtain a speech retrieval system that reliably finds references for the speech recognition system based on level-building DTW. Thus, a desired property of the system is a high average recall over all evaluations. To measure the retrieval performance of a given configuration, the average F_2-measure is considered, which weights the recall higher than the precision. All evaluations are performed using clean speech only.

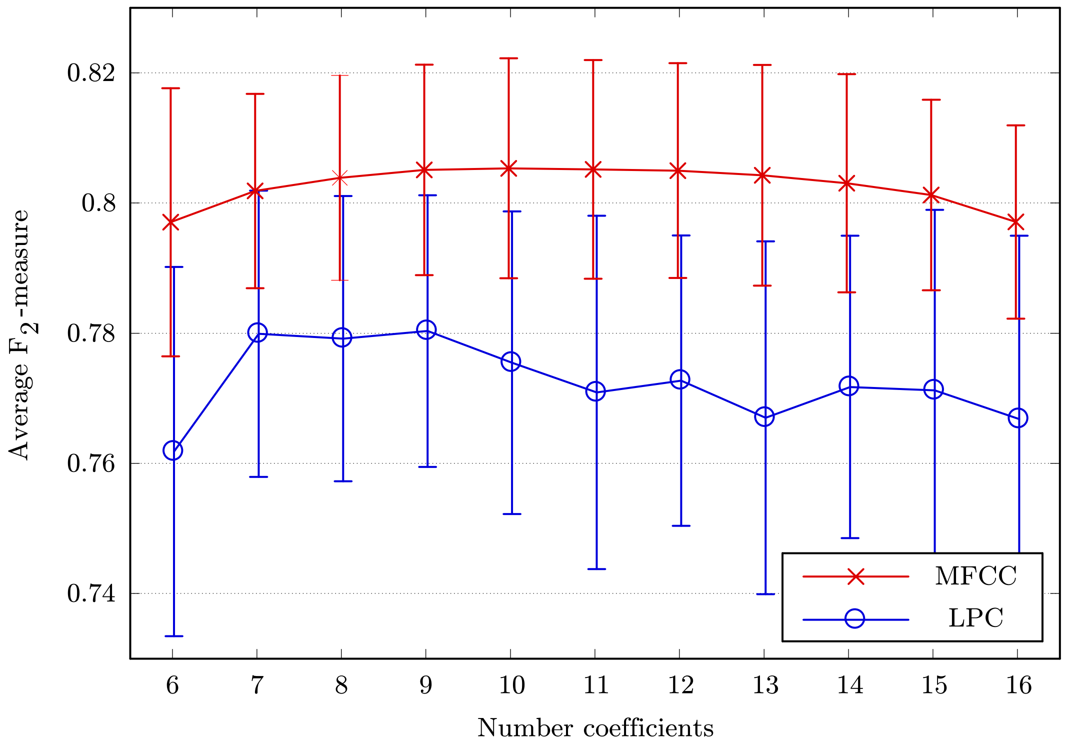

Figure 6.1: Average F_2-measures and variances of the speech retrieval system based on MFCCs and LPC with various numbers of feature coefficients using the cosine distance as local cost measure.

In the first step, the test string is uniformly framed. Each 156.25 ms a Hann window of size 312.5 ms is transformed into features and queried using subsequence DTW. Using this window size yields suitable results. For each window, the best-ranked match is taken. The average precision and recall values and the variance of this data are measured to compute the average F_2-measure and its variance. For the first evaluations, this is performed for MFCCs and LPC features. The MFCCs are computed and the 0^{th} coefficient is discarded. Afterwards the delta and delta-delta coefficients are computed. The LPC coefficients are

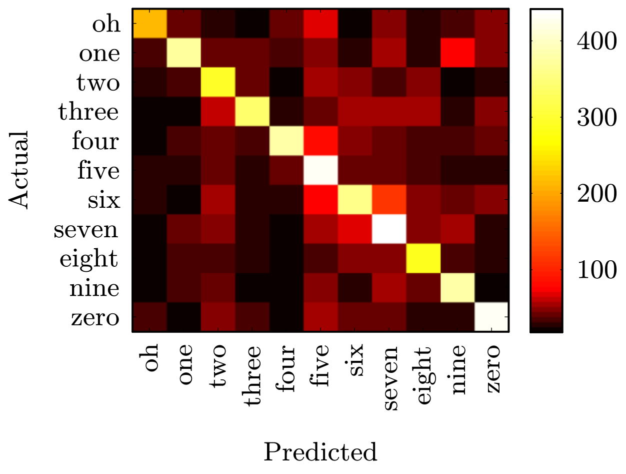

Figure 6.2: Confusion matrix of the speech retrieval system using MFCC10.

estimated using the covariance method and the 0th coefficient is discarded, as well. The utilized local cost measure is the cosine distance. The evaluation results are depicted in Figure 6.1, where in case of MFCCs the number of coefficients exclude delta and delta-delta coefficients. Using 10 MFCCs plus delta and delta-delta coefficients yields the best result for the analyzed configurations with an F_2-measure of 0.805. This configuration now is called *MFCC10*. The average recall for this configuration is very high with 0.966 and the average precision is 0.484. The high recall shows that the method reliably finds correct words in the database that actually occur within the test string. The lower precision shows that some false positives have to be accepted. In case of using delta and delta-delta coefficients for the LPC feature extraction with an LPC order of 9, as well, the average F_2-measure does not change.

To get an impression of the errors made by the speech retrieval system using MFCC10, Figure 6.2 depicts the confusion matrix as a colored image. In some cases, an utterance of the word „six“ is predicted as „five“. Sometimes „one“ is substituted by „nine“. Most substitutions stem from unsuitable positions of the windows. Since the windows are not adapted to positions of word utterances, in some cases a window contains only a short portion of a word or it may contain sounds from two different words. In such a case, subsequence DTW does not work properly, which leads to a possible mismatch. Thus, an increase of the amount of false positives has to be accepted. During the evaluations, in 112 cases the algorithm is unable to predict any word from the windows, because they do not contain any speech content. These cases are excluded in the diagram.

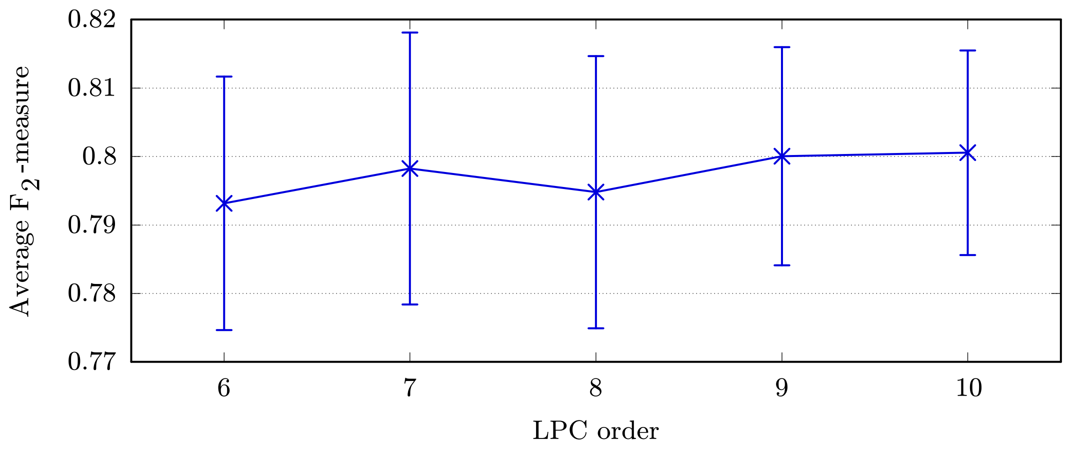

Figure 6.3: Average F_2-measures and variances of the speech retrieval system based on LPC with various prediction orders using the residual-based local cost measure.

Further evaluations are performed using the LPC feature extraction combined with the residual-based local cost measure proposed by Rabiner and Schmidt for connected digit recognition. In this case, the 0^{th} LPC component is not excluded. The F_2-measures for various LPC orders are depicted in Figure 6.3. The highest F_2-measure occurs in case of an LPC order of 10 with a value of 0.801. This configuration is defined as *LPC10*. It delivers an average recall of 0.968 and an average precision of 0.474. Thus, using LPC features can yield the same results as MFCCs in this test scenario.

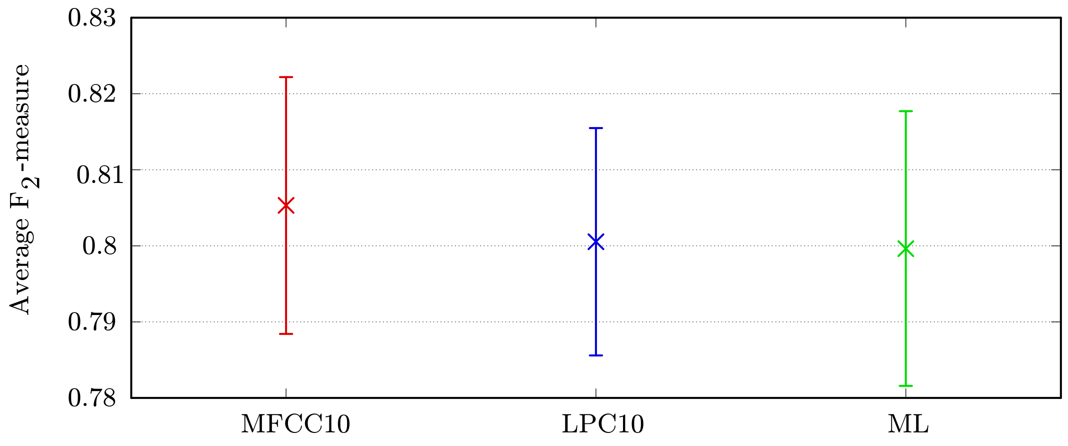

Figure 6.4: Average F_2-measures and variances of the speech retrieval system using MFCC10, LPC10 and ML.

In the last evaluations, both MFCCs and LPC features are used in combination. The feature vectors are built up using MFCC10 including its delta und delta-delta coefficients concatenated with LPC coefficients using a prediction order of 8. The utilized local cost measure is the cosine distance. This configuration is called *ML*. Figure 6.4 depicts the F_2-measures for MFCC10, LPC10 and ML. Although all three configurations are different, they perform similarly in this retrieval scenario.

6.3 Speech Recognition Systems

In the following, various configurations of the three introduced speech recognition systems are evaluated. The optimal configuration yielding a low WER and a possibly low RTF is sought. First evaluations are performed for the template and the database recognizer. We recapitulate that both systems are based on level-building DTW using a set of references. The template recognizer works with a set of 12 templates per word, spoken by various speakers. Each 12 templates are uttered in isolation by 6 male speakers and 6 female speakers. All in all, 132 templates are used. The word utterances are collected from the database set. The database recognizer uses all digit utterances from the database set that are not too short. Each utterance producing a feature sequence of a length smaller than 10 is discarded to provide a robust performance of level-building DTW. This leads to a set of 8305 references. Level-building DTW is configured to recognize word strings of lengths up to 20 words. The actual maximal amount of words contained in a string is 7. For the KNN decision rule, the 2 nearest neighbor references from different speakers are selected and averaged. Figure 6.5 depicts WERs for three different recognizer concepts. The LPC-based template and database recognizers use the residual-based local cost measure. The MFCC-based recognizer discards the 0^{th} MFCC and uses the cosine distance as the local cost measure. The results show that MFCCs are not suitable when using level-building DTW for connected word recognition. The best result with the LPC-based template recognizer is obtained using a prediction order of 7 yielding a WER of 25.6 %. The LPC-based database recognizer with a prediction order of 6 obtains the lowest WER with a value of 23.0 %. Here, not every number of LPC coefficients is checked, because of a long computation time. The LPC features are estimated using the covariance method.

To check the relevance of the KNN decision rule, all evaluations are done for the template recognizer and the database recognizer using LPC features again for $K = 1$, i.e., no pair-wise averaging of accumulated costs is performed. The residual-based local cost measure is used. Figure 6.6 depicts the WERs. It appears that the WER drops for $K = 1$. For instance, the WER of the template recognizer using LPC features of order 7 decreases from a value of 25.6 % ($K = 2$) to 25.5 % ($K = 1$). In the following, this feature setup is

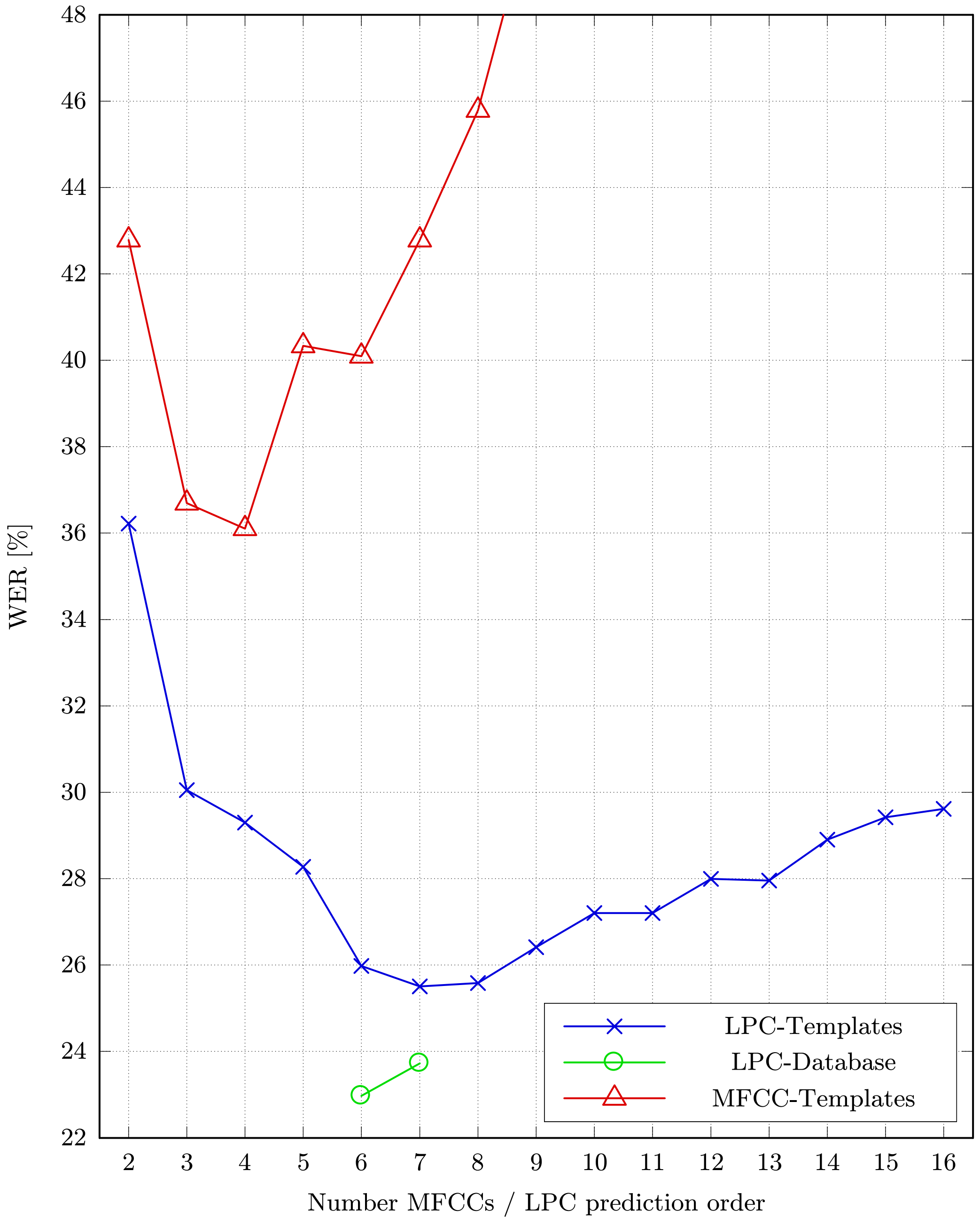

Figure 6.5: WERs for various template and database recognizers using MFCCs and LPC features obtained using the covariance method and the KNN decision rule with $K = 2$.

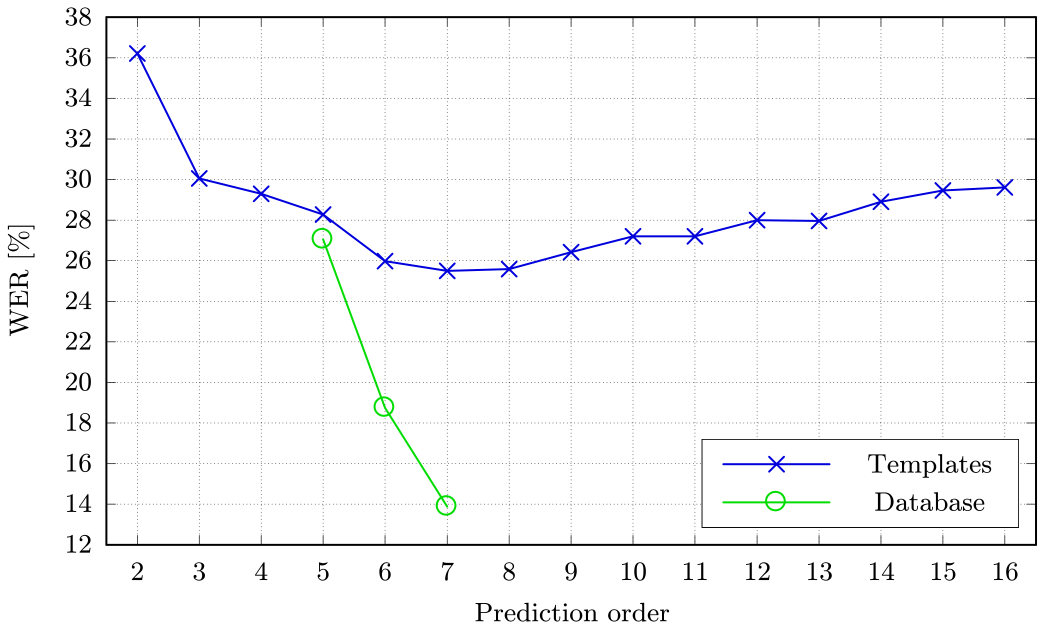

Figure 6.6: WERs for various template and database recognizers using LPC features estimated using the covariance method with $K = 1$ for the KNN decision rule.

abbreviated with *LPC7*. For the database recognizer using LPC7, the WER drops from 23.7 % ($K = 2$) to 13.9 % ($K = 1$). The results refute the suitability of the KNN decision rule. It is not helpful and accordingly not utilized in the following evaluations. The LPC7 database recognizer is much more accurate, but the computation time increases compared to the template recognizer, because of the huge amount of references. The RTF of the LPC7 template recognizer in the prototype system lies around 0.7, whereas the LPC7 database recognizer has an RTF of 22.1 in the evaluations. Thus, not every number of LPC coefficients is checked for the database recognizer.

All previous evaluations based on LPC features are performed using the covariance method, which is more complex than the autocorrelation method used to estimate LPC coefficients. However, using the autocorrelation method increases the accuracy significantly. In case of the LPC7 template recognizer, the WER drops from a value of 25.5 % to 17.6 % using the autocorrelation method. The RTF has a value of 0.7. The LPC7 database recognizer achieves a WER of 13.9 % and an RTF of 21.6.

Next, the retrieval-based recognizer is configured and evaluated. It uses the MFCC10 configuration to retrieve references and then performs level-building DTW using LPC7. This setup should solve each step optimally. However, the performance is not satisfying with a WER of 79.3 %. When using LPC7 for both, the reference retrieval and the speech

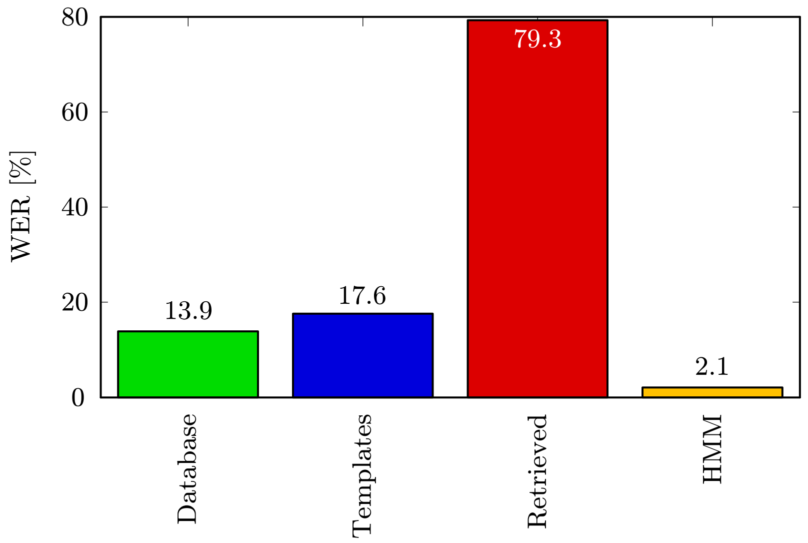

Figure 6.7: WERs for the database, template and retrieval-based recognizers in comparison to the HMM recognizer.

recognition, the WER is 75.2 %. Thus, the retrieval-based concept is not suitable for connected word recognition using level-building DTW. This surprises, because level-building DTW is provided with suitable references, which is shown in the previous section. Figure 6.7 depicts the WER of the LPC7 database recognizer, the LPC7 template recognizer, the retrieval-based recognizer and the result using a state-of-the-art HMM recognizer. The latter system is a phoneme recognizer built up with HTK [31] using 4 GMMs per state. The HMM recognizer achieves a WER of 2.1 %.

The minimum and maximum number of levels permitted using level-building DTW can be adapted to a certain range. This allows for forcing level-building DTW to always predict a word string with the correct number of levels. In this case, the algorithm is forced to find the best-fitting word string of fixed length. Evaluating the database recognizer in this setup, the WER drops significantly from 13.9 % to a value of 2.1 %, which is the same result the state-of-the-art HMM recognizer yields in this scenario. The RTF drops, as well, to 9.0.

For the LPC7 database recognizer having a WER of 13.9 %, 77 % of the word errors are additional insertions. Figure 6.8 shows the five most frequent additionally inserted words. Short utterances „oh“ and „eight“ are most frequently inserted.

To investigate the performance of the LPC7 template recognizer using speech recordings affected by noise, the system is evaluated using a test set with additive natural noise

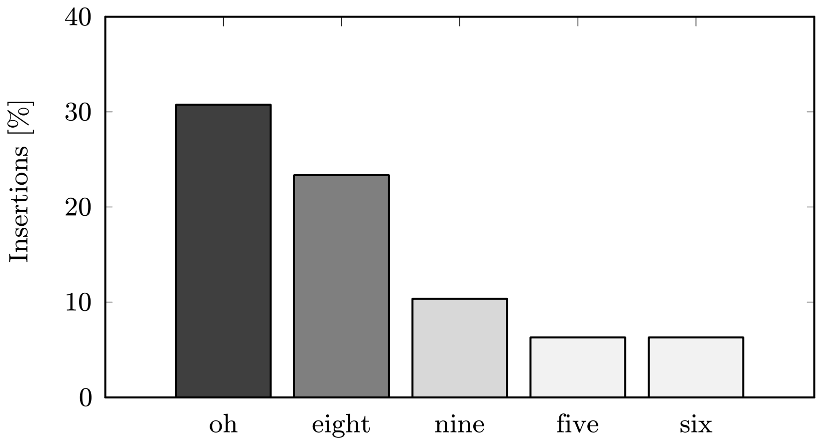

Figure 6.8: Most frequent additional word insertions made by the LPC7 database recognizer.

provided by the Aurora-2 corpus. An SNR of 10 dB affects the template recognizer. The WER rises from 17.6 % to 80.1 %. The lack of accuracy can stem from both, an inaccurate speech-silence segmentation, as well as distorted LPC feature coefficients. The performance of both algorithms for estimating LPC coefficients deteriorates significantly when dealing with noise-affected speech recordings.

6.4 Conclusion and Outlook

In this chapter, both, the subsequence-DTW-based retrieval system and the level-building-DTW-based connected word recognition system, have been evaluated to find the optimal parameters. The function of the speech retrieval system is to find suitable references for level-building DTW, thus a desired property is a high recall. The goal of the speech recognition system is to produce as few word errors as possible. The Aurora-2 speech corpus is used to build up a speaker-independent test scenario using digit strings of lengths between 1 and 7. The database contains 2823 word strings, uttered by 40 male and 40 female speakers and the test material consists of 770 strings, produced by 10 male and 10 female speakers. The database word strings are annotated and aligned.

The speech retrieval system is evaluated for clean speech using both, MFCCs and LPC features. When using the cosine distance as the local cost measure, the optimal configuration is achieved using 10 MFCCs. This yields an average F_2-measure of 0.805. The average recall is very high with a value of 0.966, indicating a suitable retrieval of references for level-building DTW. Similar retrieval results can be achieved using LPC feature coefficients with a prediction order of 10 estimated using the covariance method in combination with the residual-based local cost measure.

To find the optimal level-building-DTW-based speech recognition system, three different concepts are evaluated using clean speech. It is shown that the proposed KNN decision rule by Myers and Rabiner can be dropped. First, the template recognizer using 12 reference templates per word spoken by different speakers in isolation is analyzed. All in all, 132 references are used, which are level-wise compared with the test string. The best performance is achieved using LPC features of order 7 estimated using the autocorrelation method in combination with the residual-based local cost measure (LPC7), yielding a connected digit recognition system with a WER of 17.6 % and an RTF of 0.7. A better accuracy can be measured for the database recognizer that uses the whole database as references, except for very short utterances. This leads to a set of 8305 references. The LPC7 database recognizer yields a significantly lower WER with a value of 13.9 %. The higher amount of references lead to a rise of the RTF to a value of 21.6. Most frequently, the recognizer predicts additional words „oh“ and „eight“ that are actually not contained in the word string. In case of known string lengths, the WER drops to a value of 2.1 %, which is the same WER achieved by a state-of-the-art phoneme-based HMM recognizer built using HTK. The RTF drops, as well, to 9.0 in this scenario. This shows that the algorithm works much better if a good estimation of the number of words contained in the test string is given beforehand.

A significant accuracy deterioration of the level-building-DTW-based speech recognizer using LPC features is determined in case of noise-affected speech recordings. It shows the main problem of the speech recognition approach. Both, the speech-silence segmentation, as well as the LPC estimation methods have to be revisited to improve the accuracy in noisy environments. A possible improvement for the estimation of LPC features is given by Swain and Abdulla, who proposed a new approach in [30], in 2004.

All evaluations are performed using a speech corpus containing digits only, which provides a very small dictionary of 11 different words. Larger dictionaries can be evaluated, as well, since the algorithm can easily be adapted. Since the algorithm is not using any further knowledge about the language except for the LPC filter parameters, the recognition accuracy for other languages can be measured, as well. Since the recognition concept does not depend on training, it provides a very high degree of flexibility. In the evaluations, it is used to recognize connected word strings. Further research may include the investigation of syllable and phoneme recognition. Since the system uses a high feature density, it is possibly suitable to work with these shorter units, as well. Another important aspect is the choice of references. Levinson and Rabiner proposed a clustering technique for selecting speaker-independent reference templates in [17], in 1979. This concept is also used in the level-building-DTW-based connected digit recognition system proposed by Rabiner and Schmidt, in 1980. In that work, clustering is used to detect suitable reference templates in a large set of utterances. The reference set has a considerable influence on the recognition results, as well as the computation time.

7 Summary

In this work, three different retrievel-based speech recognition systems have been developed, optimized and evaluated. The goal of the systems is to automatically match each word utterance within a test word string with a similar word utterance within a set of references. Given such a mapping, the test string can be annotated, which leads to an STT system. The fundamental assumption of this approach is that for each word that can occur in a test string, a set of reference utterances exists. The concept is referred to as connected word recognition. Level-building DTW is shown to be a suitable concept for connected word recognition, for the case that the words are digits. It builds up the test string, word-by-word, by iteratively comparing portions of the test string with each reference. Each comparison is performed using a time alignment determined with a warping path, which is used to quantify the similarity. The three retrieval-based speech recognition systems built up in this work are based on level-building DTW using different concepts of references. The template recognizer works with a small set of selected isolated word utterances spoken from different speakers, the database recognizer uses the whole database as references and the third recognizer uses automatically retrieved references that have similar feature characteristics. The speech retrieval system, which finds suitable references for level-building DTW, is based on uniform framing and windowing, following a window-wise query matching based on subsequence DTW.

For determining the speech features, first the waveform is processed to obtain a speech-silence segmentation using our new approach, which detects time positions containing speech within a spectrogram by determining speech candidates. A speech candidate can be a spectrogram cell with high amplitude or high local amplitude variance. A suitable configuration is found empirically. Two concepts for extracting features are introduced and evaluated for both scenarios, the speech retrieval and the speech recognition. The concepts are MFCCs and LPC.

After implementing the concepts in MATLAB, the speech retrieval and the speech recognition systems have been evaluated and optimized using the Aurora-2 speech corpus in a speaker-independent test scenario. The best configuration for the speech retrieval system is achieved by extracting 10 MFCCs excluding the 0^{th} component for each window of size

312.5 ms using a step size of 156.25 ms. The local cost measure is the cosine distance. The retrieval system achieves an average F_2-measure of 0.805. The average recall is very high with a value of 0.966, which guarantees a retrieval of suitable references for level-building DTW.

The evaluation of the three speech recognition systems indicates optimal results using LPC features of order 7, extracted with the autocorrelation method, in combination with the residual-based local cost measure introduced by Rabiner and Schmidt. The LPC7 template recognizer, using 12 template references per word spoken from various speakers, achieves a WER of 17.6 % with an RTF of 0.7. The LPC7 database recognizer obtains a lower WER of 13.9 %, but takes more processing time with an RTF of 21.6. The most frequent word errors are additional insertions of „oh“ and „eight“. In case of known string lengths, the database recognizer achieves a WER of 2.1 %, which is the same result a state-of-the-art HMM recognizer built up using HTK achieves. This shows that the performance of level-building DTW increases, if a suitable estimate for the range of numbers of words contained in the test string is given beforehand. Additionally, the RTF decreases to a value of 9.0. Further evaluations show the inaccuracy of the retrieval-based recognizer that uses subsequence DTW to determine references.

A significant accuracy deterioration is measured for the template recognizer in case of noise-affected speech recordings. The performance lack stems from both, the speech-silence segmentation, as well as the LPC feature extraction that, in further research, have to be revisited to improve the noise robustness.

The main advantage of level-building-DTW-based speech recognition is the high degree of flexibility. This retrieval-based approach provides language independence. No training of language models is necessary. The performance of the recognizer can directly be influenced by the choice of the set of references. Further research may include the investigation of syllable and phoneme recognition.

8 Bibliography

[1] Abdulla, Waleed H., David Chow, and Gary Sin: *Cross-words reference template for DTW-based speech recognition systems.* In *TENCON. Conference on Convergent Technologies for the Asia-Pacific Region*, volume 4, pages 1576–1579. IEEE, 2003.

[2] Atal, Bishnu S. and Suzanne L. Hanauer: *Speech analysis and synthesis by linear prediction of the speech wave.* The Journal of the Acoustical Society of America, 50, Issue 2B:637–655, 1971.

[3] Brookes, Mike: *VOICEBOX: Speech Processing Toolbox for MATLAB.* Online web resource: `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`, 2005 (Accessed October 2012).

[4] Caruana, Rich: *Lecture notes „Performance Measures for Machine Learning“.* `http://www.cs.cornell.edu/courses/cs678/2006sp/performance_measures.4up.pdf`. Cornell University, Department of Computer Science, Spring 2006 (Accessed February 2013).

[5] Clausen, Michael and Ulrich Baum: *Fast Fourier Transforms.* BI Wissenschaftsverlag, 1993.

[6] Clausen, Michael and Meinard Müller: *Lecture notes „Basic Concepts of Digital Signal Processing“.* University of Bonn, 2003.

[7] Clausen, Michael and Meinard Müller: *Lecture notes for „Inhaltsbasiertes Multimedia-retrieval“.* University of Bonn, 2007.

[8] Damm, David, Harald G. Grohganz, Frank Kurth, Sebastian Ewert, and Michael Clausen: *SyncTS: Automatic synchronization of speech and text documents.* In *Semantic Audio. Proceedings of the AES 42nd International Conference*, pages 98–107. Audio Engine Society (AES), 2011.

[9] Ellis, Daniel P. W.: *PLP and RASTA (and MFCC, and inversion) in MATLAB.* Online web resource: `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`, 2005 (Accessed October 2012).

[10] Furui, Sadaoki: *Digital Speech Processing, Synthesis and Recognition.* Marcel Dekker, Inc., 2000.

[11] Hirsch, Hans Günter and David Pearce: *The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions.* In *Automatic Speech Recognition: Challenges for the Next Millenium.* ISCA ITRW, 2000.

[12] Huang, Xuedong, Alex Acero, and Hsiao Wuen Hon: *Spoken Language Processing: A Guide to Theory, Algorithm and System Development.* Prentice Hall, 2001.

[13] Itakura, Fumitada: *Minimum prediction residual principle applied to speech recognition.* IEEE Transactions on Acoustics, Speech and Signal Processing, 23:67–72, 1975.

[14] Juang, Biing Hwang and Lawrence R. Rabiner: *Automatic speech recognition–a brief history of the technology.* Elsevier Encyclopedia of Language and Linguistics, Second Edition, 2005.

[15] Kurth, Frank and Dirk von Zeddelmann: *An analysis of MFCC-like parametric audio features for keyphrase spotting applications.* In *Sprachkommunikation, 9. ITG-Fachtagung.* VDE Verlag, 2010.

[16] Leonard, Robert G.: *A database for speaker-independent digit recognition.* In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 9, pages 328–331, 1984.

[17] Levinson, Stephen E. and Lawrence R. Rabiner: *Interactive clustering techniques for selecting speaker-independent reference templates for isolated word recognition.* IEEE Transactions on Acoustics Speech and Signal Processing, 27, Issue 2:134–141, 1979.

[18] Liu, Yadong, Yee Chun Lee, Hsing Hen Chen, and Guo Zheng Sun: *Speech recognition using dynamic time warping with neural network trained templates.* In *International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 326–331, 1992.

[19] Logan, Beth: *Mel frequency cepstral coefficients for music modeling.* In *International Symposium on Music Information Retrieval (ISMIR)*, 2000.

[20] Mubarak, Omer Mohsin, Eliathamby Ambikairajah, Julien Epps, and Teddy Surya Gunawan: *Modulation features for speech and music classification.* In *IEEE Singapore International Conference on Communication Systems (ICCS)*, 2006.

[21] Müller, Meinard: *Information Retrieval for Music and Motion.* Springer, 2007.

[22] Myers, Cory S. and Lawrence R. Rabiner: *Connected digit recognition using a level-building DTW algorithm.* IEEE Transactions on Acoustics, Speech and Signal Processing, 29, Issue 3(3):351–363, 1981.

[23] Myers, Cory S. and Lawrence R. Rabiner: *A level building dynamic time warping algorithm for connected word recognition.* IEEE Transactions on Acoustics, Speech and Signal Processing, 29, Issue 2:284–297, 1981.

[24] Powers, David M. W.: *Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness & Correlation.* Journal of Machine Learning Technologies, 2, Issue 1:37–63, 2011.

[25] Priyadarshani, P. G. N. and N. G. J. Dias: *Dynamic time warping based speech recognition for isolated Sinhala words.* In *55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 892–895. IEEE, 2012.

[26] Rabiner, Lawrence R. and Ronald W. Schafer: *Digital Processing of Speech Signals.* Prentice Hall, 1978.

[27] Rabiner, Lawrence R. and Carolyn E. Schmidt: *Application of dynamic time warping to connected digit recognition.* IEEE Transactions on Acoustics Speech and Signal Processing, 28, Issue 4:377–388, 1980.

[28] Rijsbergen, C. J. van: *Information Retrieval.* Information Retrieval Group, University of Glasgow, 1979.

[29] Schukat-Talamazzini, Ernst Günter: *Automatische Spracherkennung.* Vieweg Verlag, 1995.

[30] Swain, Akshya K. and Waleed H. Abdulla: *Estimation of LPC parameters of speech signals in noisy environment.* In *TENCON*, volume 1, pages 139–142. IEEE, 2004.

[31] Young, S. J., G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland: *The HTK Book, version 3.4.* Cambridge University Engineering Department, Cambridge, UK, 2006.

[32] Young, Steve: *A review of large-vocabulary continuous-speech recognition.* IEEE Signal Processing Magazine, 13, Issue 5, 1996.

[33] Zeddelmann, Dirk von, Frank Kurth, and Meinard Müller: *Perceptual audio features for unsupervised key-phrase detection.* In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 257–260. IEEE, 2010.

Statement of Affirmation

I hereby declare that the master's thesis submitted was in all parts exclusively prepared on my own, and that other resources or other means (including electronic media and online sources), than those explicitly referred to, have not been utilized.

All implemented fragments of text, employed in a literal and/or analogous manner, have been marked as such.

Bonn, May 6, 2013

Signature

Joscha Simon Rieber