Speech Processing

David Suendermann

http://suendermann.com

Baden-Wuerttemberg Cooperative State University
Stuttgart, Germany

General remarks

- **The most up-to-date version of this document as well as auxiliary material can be found online at**

  `http://suendermann.com`

Outline

- **introduction**
- **speech recognition**
  - **introduction**
  - **evaluation/Levenshtein distance/dynamic programming**
  - **Fourier transform**
  - **speech analysis**
  - **statistical models**
- **speech synthesis**
- **voice conversion**

Outline

- **introduction**
- **speech recognition**
  - **introduction**
  - **evaluation/Levenshtein distance/dynamic programming**
  - **Fourier transform**
  - **speech analysis**
  - **statistical models**
- **speech synthesis**
- **voice conversion**

- **Excerpt from the areas of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2012)**

  13: Speech processing

  - **13.1: Speech production**
  - **13.2: Speech perception and psychoacoustics**
  - **13.3: Speech analysis**
    - **13.3.1: Spectral and other time-frequency analysis techniques**
    - **13.3.2: Distortion measures**
    - **13.3.3: Pitch/fundamental frequency analysis**
    - **13.3.4: Timing/duration/speaking rate analysis**
    - **13.3.5: Acoustic-phonetic features (e.g. formants)**
    - **13.3.6: Non-linguistic information (e.g. gender, emotion)**
    - **13.3.7: Voice quality**

Areas of speech processing (cont.)

- **13.4: Speech synthesis and generation**
  - **13.4.1: Concatenative synthesis**
  - **13.4.2: Statistical synthesis**
  - **13.4.3: Articulatory synthesis**
  - **13.4.4: Parametric synthesis**
  - **13.4.5: Prosody, emotional, expressive synthesis**
  - **13.4.6: Text-to-phoneme conversion**
  - **13.4.8: Voice conversion**
- **13.5: Speech coding**
- **13.6: Speech enhancement**
- **13.7: Acoustic modeling for speech recognition**
  - **13.7.1: Feature extraction**
  - **13.7.2: Low-level feature modeling (e.g. Gaussians)**
- **13.8: Robust speech recognition**
- **13.9: Speech adaptation and normalization**
- **13.11: Multilingual recognition and identification**
- **13.12: Lexical modeling**

- **13.13: Search**
  - **13.13.1: Decoding algorithms**
  - **13.13.2: Lattices**
- **13.14: Speaker recognition**

14: Spoken Language Processing

- **14.1: Spoken language understanding [→KBS]**
  - **14.1.1: Semantic classification [→KBS]**
  - **14.1.3: Spoken document summarization**
  - **14.1.4: Topic spotting and classification**
  - **14.1.5: Question answering**
- **14.3: Spoken dialog systems [→KBS]**
  - **14.3.1: Systems, applications, and architectures [→KBS]**
  - **14.3.2: Stochastic learning for dialog modeling [→KBS]**
  - **14.3.3: Response generation**
  - **14.3.5: Evaluation metrics [→KBS]**
  - **14.3.6: Speech-based human-computer interfaces**

Areas of speech processing (cont.)

14.4: Speech data mining

14.5: Speech data retrieval

- **14.5.3: Voice search**

14.6: Machine translation of speech

14.7: Language modeling [→KBS, ML]

- **14.7.1: N-grams and smoothing methods [→ML]**
- **14.7.3: Grammars [→KBS]**

14.8: Spoken language resources and annotation [→KBS]

Outline

- **introduction**
- **speech recognition**
  - **introduction**
  - **evaluation/Levenshtein distance/dynamic programming**
  - **Fourier transform**
  - **speech analysis**
  - **statistical models**
- **speech synthesis**
- **voice conversion**

Outline

- **introduction**
- **speech recognition**
  - **introduction**
  - **evaluation/Levenshtein distance/dynamic programming**
  - **Fourier transform**
  - **speech analysis**
  - **statistical models**
- **speech synthesis**
- **voice conversion**

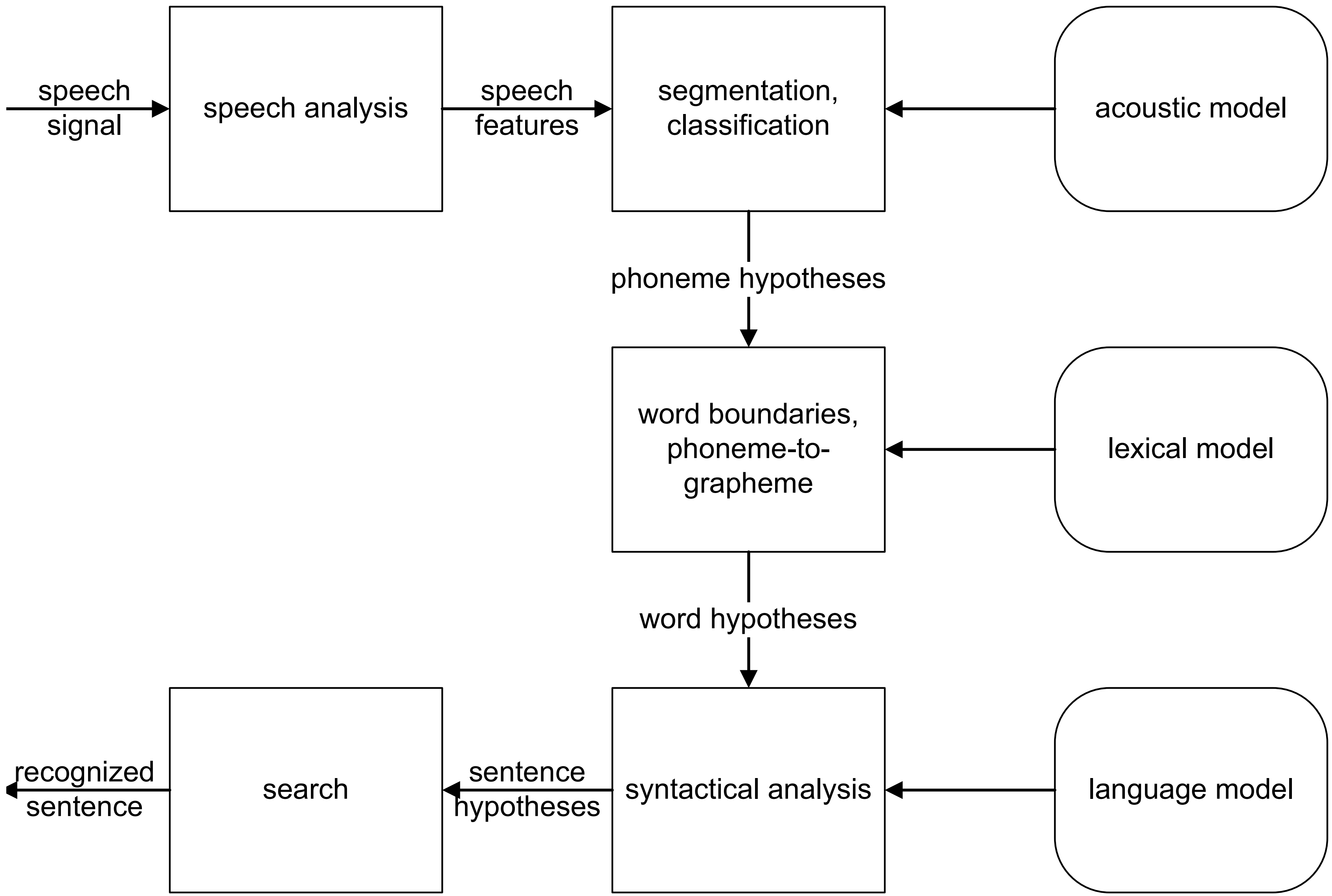
Applications of speech recognition

- **Spoken dialog systems**
  - **Call router**
  - **Phone banking**
  - **Technical support**
  - **Directory assistance**
  - **Train or flight schedule**
- **Command and control**
  - **Car system control**
  - **Voice dialing**
- **Dictation**
  - **Text messaging**
  - **Medical transcription**
- **Voice search**
- **Speech-to-speech translation**

Why is speech recognition not perfect?

- **Variability of the signal**
  - **Speaker differences (accent, age, gender)**
  - **Speech differences (speaking rate, hesitations, repetitions)**
  - **Evironmental differences (noise, microphone, channel)**
- **Too little training data**
  - **Acoustic model**
  - **Lexical model**
  - **Language model**
- **Models exclude potentially essential information**
  - **Phase**
  - **Spectral fine structure**
  - **Fundamental frequency**
  - **Voicing**
  - **[play example utterance]**

The holistic approach to speech recognition



Outline

- **introduction**
- **speech recognition**
  - **introduction**
  - **evaluation/Levenshtein distance/dynamic programming**
  - **Fourier transform**
  - **speech analysis**
  - **statistical models**
- **speech synthesis**
- **voice conversion**

Evaluation of a speech recognizer

- **To be able to evaluate performance of a recognizer in a fair fashion, the evaluation data (test data) must never have seen before.**
- **Often, also training and development data are standardized for multilateral evaluations.**
- **Most common performance metric is the word error rate:**

$$\text{WER} = \frac{\text{Levenshtein distance}}{\text{number of spoken words}} \tag{1}$$

- **The Levenshtein distance (aka edit distance) is the minimum number of substitutions, deletions, or insertions necessary to map a string of spoken words to the string of recognized words.**

- **Example from a call router:**
  - **Spoken words:**

    no i need repair on my phone its uh crackling and its ringing here where i live and uh its just dead other than the crackling no dial tone at all
  - **Recognized words:**

    no — need repair on my phone its not crackling and — ringing here where i live — — in my bed other than the crackling *there* **no dial tone at all**

  substitution, *insertion*, **— deletion**

$$\text{WER} = \frac{4 \text{ substitutions} + 1 \text{ insertion} + 4 \text{ deletions}}{31 \text{ spoken words}} = 29\% \qquad (2)$$

Word error rate: ambiguity

- **Consider another example:**
  - **Spoken words:**

    no need to repair my phone
  - **Recognized words:**

    no need repair my broken phone
  - **Determining word errors:**

    1) **no need repair my broken phone [3 errors]**
    2) **no need repair — my *broken* phone [3 errors]**
    3) **no need — repair my *broken* phone [2 errors]**
- **Apparently, determining the minimum number of errors may not be as trivial as it first seems.**

- **Examplifying the Levenshtein distance on the character level:**

		S	A	T	U	R	D	A	Y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
U	2	1	1	2	2	3	4	5	6
N	3	2	2	2	3	3	4	5	6
D	4	3	3	3	3	4	3	4	5
A	5	4	3	4	4	4	4	3	4
Y	6	5	4	4	5	5	5	4	3

- **In this matrix, penalties for**
  - **a vertical step is 1 (insertion),**
  - **a horizontal step is 1 (deletion),**
  - **a diagonal step is 0 if the target fields' letters are identical, otherwise 1 (substitution).**

- **Formally, the Levenshtein distance can be computed by an application of dynamic programming (DP).**
- **DP is a method to solve complex problems by breaking them down into simpler subproblems.**
- **In our case, the problem is to determine the minimal cost c of a path through a grid spanned by the two symbol sequences (letters, characters, feature vectors, or the like)**

$$x_1^n := x_1, \ldots, x_n \quad \text{and} \quad y_1^m := y_1, \ldots, y_m. \tag{3}$$

- **The cost c is nothing but the minimal cost of a path ending at the grid node (n, m) where both sequences terminate:**

$$c = l(n, m). \tag{4}$$

- **Now, let us inductively define the cost $l(i,j)$ for the minimal cost of a path ending at the grid node (i,j):**
  - **The start cost, i.e., the cost at the beginning of the path, is set to zero, and the costs of initial deletions and insertions are defined as boundary conditions:**

$$l(0,0)=0; \quad l(i,0)=i \text{ for } i\in\{1,\dots,n\}; \quad l(0,j)=j \text{ for } j\in\{1,\dots,m\}.$$

  - **For every grid node, the minimum cost is that one obtained by coming from the vertical neighbor by insertion, from the horizontal neighbor by deletion, or from the diagonal neighbor by potential substitution:**

$$\begin{aligned} l(i,j) \quad = \quad & \min\left(\{l(i,j-1)+1,\right. \\ & l(i-1,j)+1, \\ & \left. l(i-1,j-1)+1-\delta_{x_i,y_j}\}\right) \end{aligned} \tag{5}$$

with the Kronecker delta

$$\delta_{x,y} = \begin{cases} 1 & \text{if } x=y \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

- **Calculate the Levenshtein distance between the sequences**

$$x_1^{10} = 1, 0, 0, 1, 1, 0, 1, 0, 0, 1 \quad (7)$$

and

$$y_1^{10} = 0, 1, 1, 0, 0, 1, 0, 1, 0, 1. \quad (8)$$

Outline

- **introduction**
- **speech recognition**
  - **introduction**
  - **evaluation/Levenshtein distance/dynamic programming**
  - **Fourier transform**
  - **speech analysis**
  - **statistical models**
- **speech synthesis**
- **voice conversion**

- **The Fourier transform decomposes a function into sinusoids of different frequency that sum to the original function.**
- **Definition of the Fourier transform:**

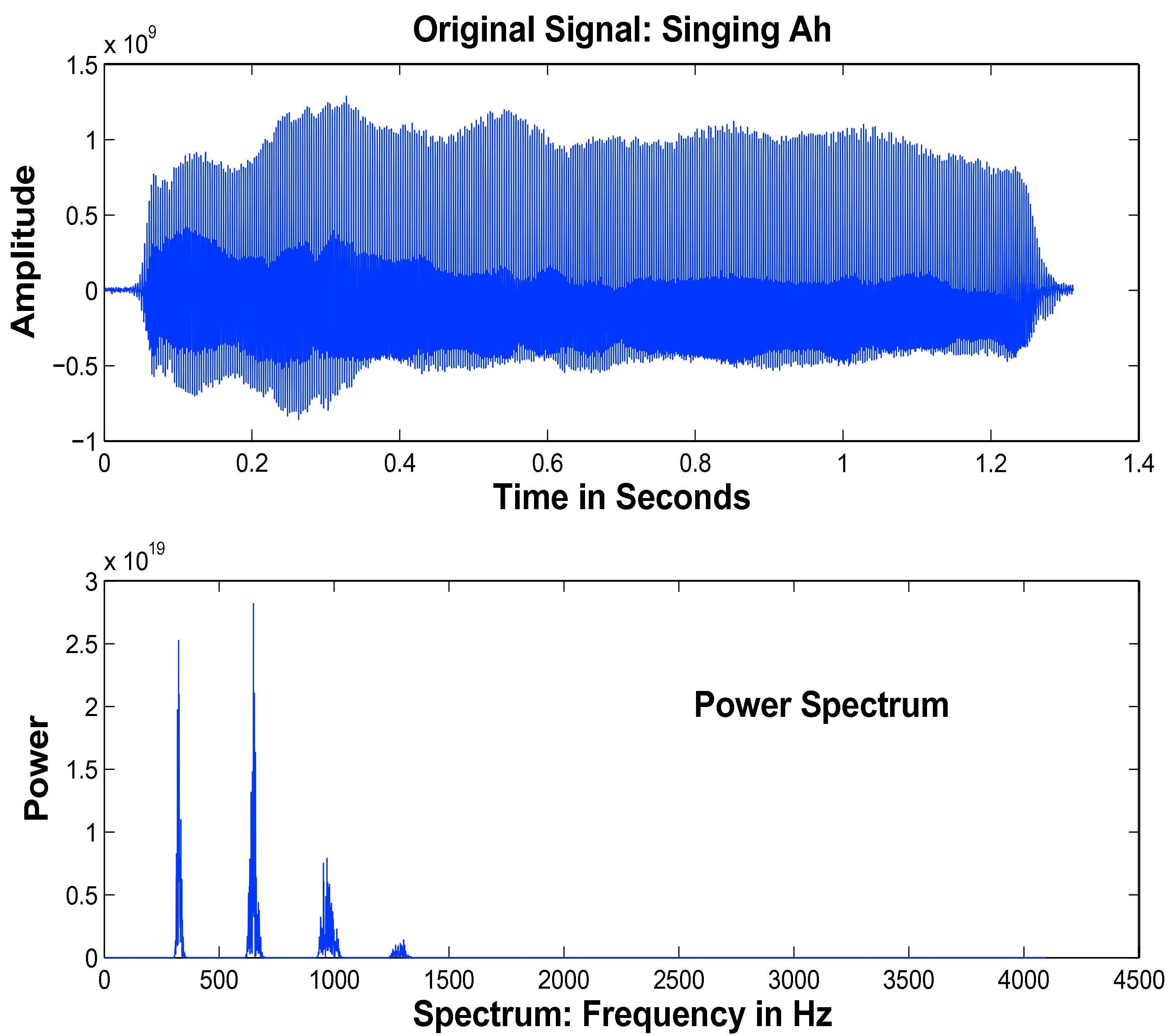
$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \mathrm{d}t; \quad \omega \in \mathbb{R}. \tag{9}$$

- **f and F occupy two domains (upper and lower) [Bracewell, 1965]:**

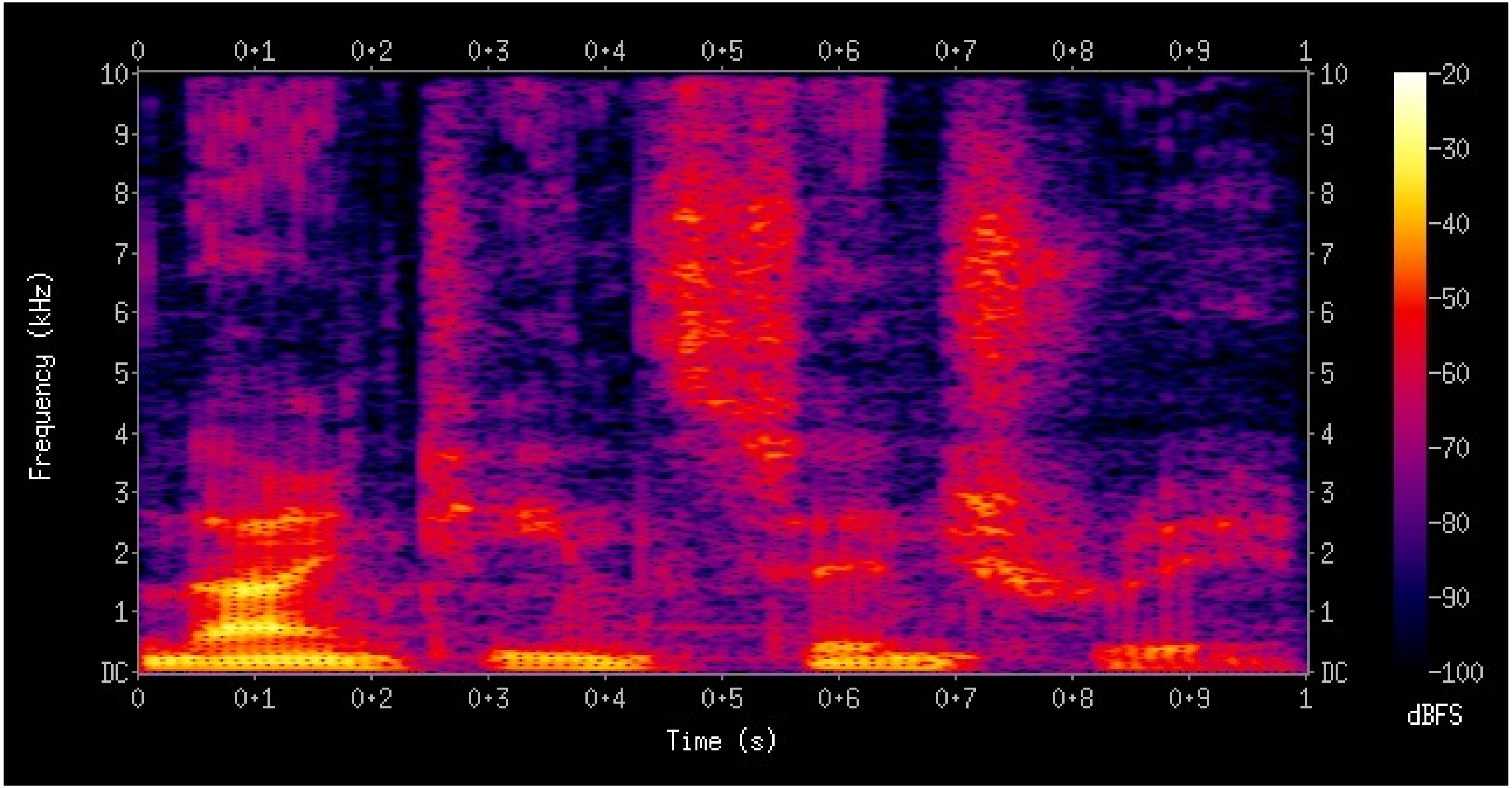
  Functions circulate[...] at ground level and their transforms in the underworld.

- **$f \longrightarrow$ time (or spacial) domain;** **$F \longrightarrow$ frequency domain**

Time signal vs. power spectrum



Original signal: “nineteenth century”



Discrete Fourier transform (DFT)

- The (continuous) Fourier transform cannot directly be handled by numerical computation that requires discrete sample values of $f(t)$.
- The same applies to the frequency domain where a computer can compute $F(\omega)$ only at discrete values of ω.
- Using the discretization $f_k = f(kT)$ and $F_r = F(r\omega_0)$, the DFT is defined as:

$$\begin{aligned} F_r &= \sum_{k=0}^{N_0-1} f_k e^{-irk\frac{2\pi}{N_0}}; \quad r \in \{0, \ldots, N_0 - 1\} \\ &= \sum_{k=0}^{N_0-1} f_k \cos\left(rk\frac{2\pi}{N_0}\right) - i \sin\left(rk\frac{2\pi}{N_0}\right) \end{aligned} \tag{10}$$

- In order to be able to reconstruct f from F_r, the Nyquist criterion has to be satisfied, i.e., $F(\omega) = 0$ for $|\omega| > \frac{1}{T}$.

Fast Fourier transform (FFT)

- **FFT was developed by [Cooley and Tukey, 1965] (even though [Heideman^{+}, 1984] discovered that Carl Friedrich Gauss had invented the algorithm already in 1805).**
- **The most popular algorithm divides the N-point transform into two transforms of size $N/2$ each.**
- **The algorithm is then recursively applied to each subdivision.**
- **This reduces the algorithm's complexity from $O(N^2)$ (DFT) to $O(N \log N)$ (FFT).**

General Applications

- optics (spectroscopy)
- medical imaging (nuclear magnetic resonance)
- solution of partial differential equations (spectral method)
- multiplication of very large integers (using FFT)

Applications to Signal Processing of Digital Media

- image and video processing
- audio processing
- speech processing
- telecommunications

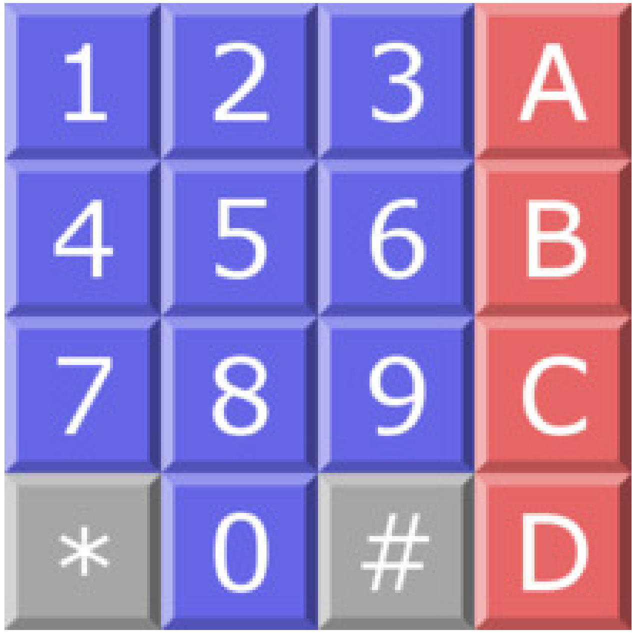
Applications to Signal Processing of Digital Media (Details)

- **image and video processing**
  - **filtering, smoothing, sharpening**
  - **restoration, blur removal, enhancement**
  - **pattern recognition**
  - **compression (JPG, MPEG)**
- **audio processing**
  - **filtering, up- and down-sampling**
  - **psycho-acoustic compression (mp3)**
  - **noise reduction**
  - **encryption**

Applications to Signal Processing of Digital Media (Details Cont.)

- speech processing
  - speech recognition (MFCC, PLP, RASTA)
  - voice conversion
  - speech synthesis (FD-PSOLA, HMM-based)
  - vocal tract length normalization
- telecommunications
  - cellular communcation
  - artificial bandwidth extension
  - recovery of lost packets in VoIP communication
  - touch tone (DTMF)

Dual-tone multi-frequency (DTMF)



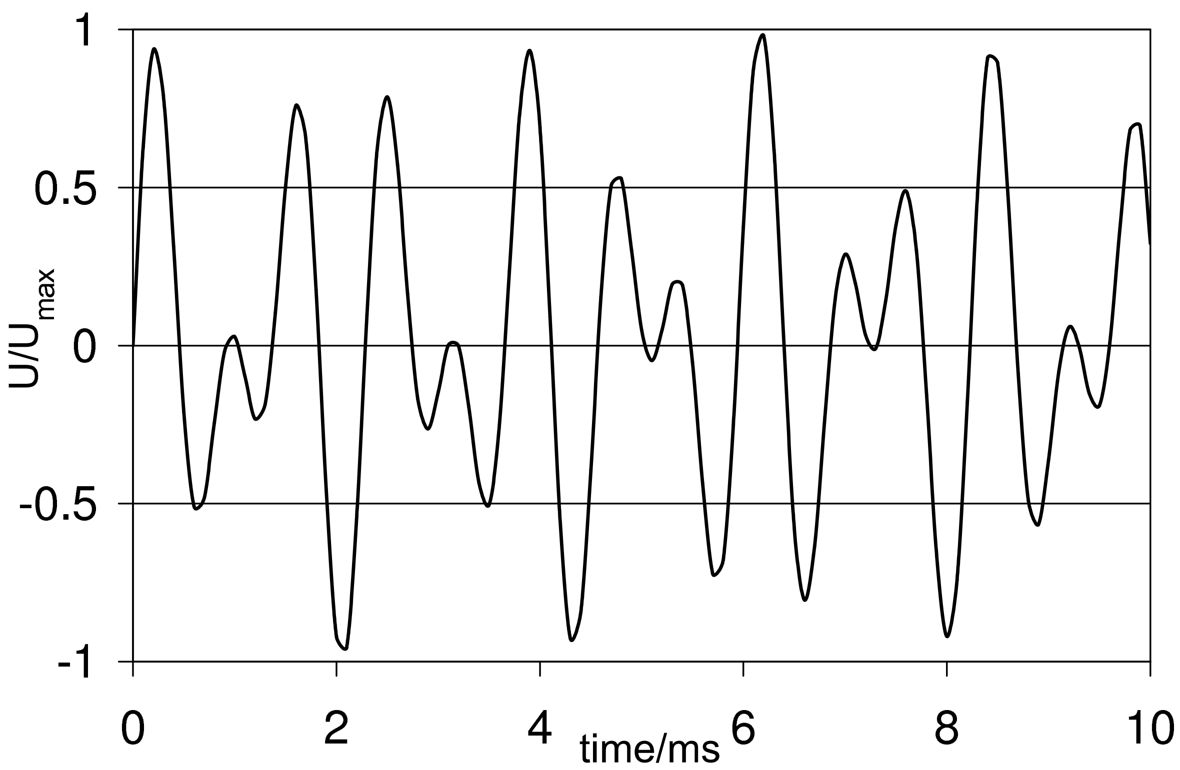
- **DTMF is used for telecommunication signaling over analog telephone lines.**
- **It gradually replaced pulse dialing starting with the introduction of the telephone keypad by AT&T in 1963.**
- **DTMF is based on a mixture of two sinusoids.**
- **(play example call)**
- **A, B, C, and D were used by the U.S. military to give some calls priority.**

DTMF (cont.)

DTMF keypad frequencies

[Hz]	1209	1336	1477
697	1	2	3
770	4	5	6
852	7	8	9
941	*	0	#

Example DTMF, time domain

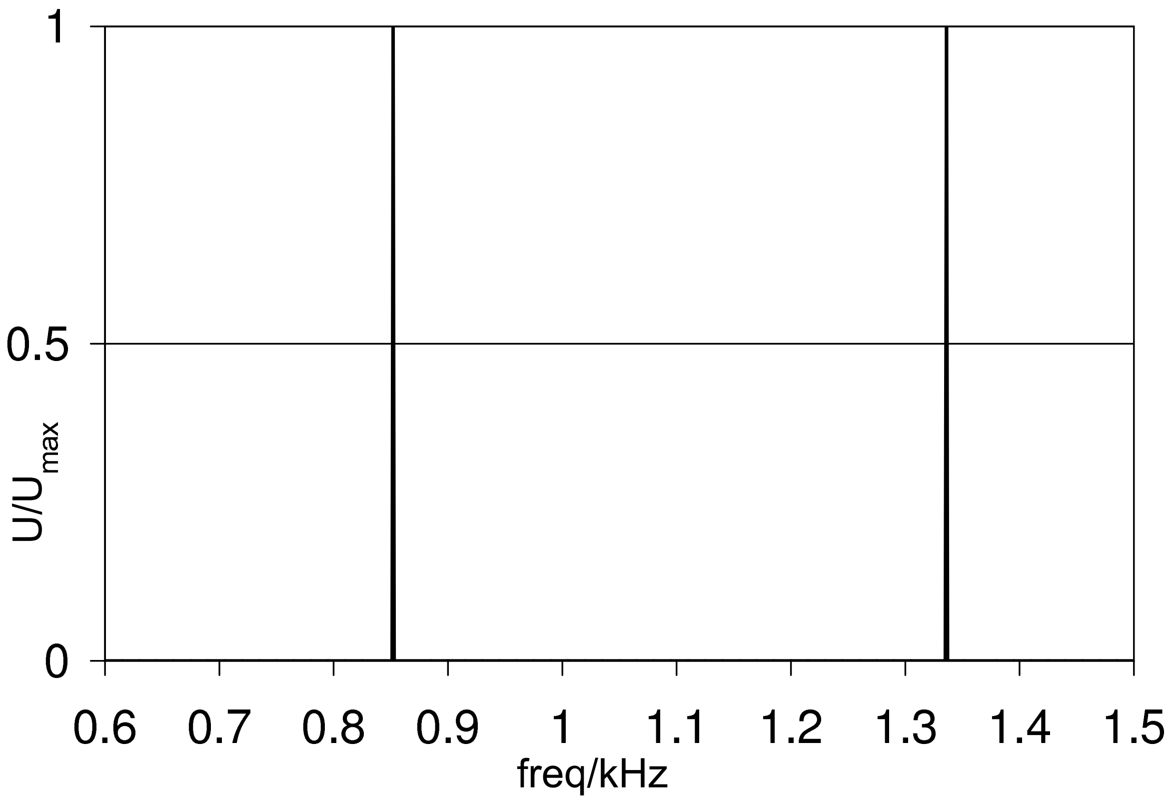


DTMF (cont.)

DTMF keypad frequencies

[Hz]	1209	1336	1477
697	1	2	3
770	4	5	6
852	7	8	9
941	*	0	#

Example DTMF, frequency domain



- **A scrambler is a device or algorithm that inverts or transposes a signal (audio, video, etc.).**
- **It renders the signal unintelligible at a receiver not equipped with a proper descrambler.**
- **The first scramblers were invented at Bell Labs just before World War II.**
- **For example, one of the devices was used by Winston Churchill and Franklin D. Roosevelt, however, it was intercepted and unscrambled by the Germans.**
- **Nowadays, scramblers are used for, e.g.**
  - **cable TV (to prevent casual signal theft),**
  - **satellite communication, and**
  - **PSTN modems.**

Audio and video encryption via scrambling (cont.)

- **A simple scrambling circuit for voice communication works as follows:**
  - **Consider the frequency band from 0 to 4kHz ($\approx$ telephony bandwidth).**
  - **Subdivide the frequency band into 4 equal sub-bands: A (0-1kHz); B (1-2kHz); C (2-3kHz); D (3-4kHz).**
  - **The original order of sub-bands is ABCD.**
  - **To render the signal incomprehensible, we re-order the sub-bands to CBDA (as an example).**
- **(play example scrambled signal)**

Audio and video encryption via scrambling (cont.)

- **After descrambling:**

- **After descrambling:**

  We shall fight in France, we shall fight on the seas and oceans, we shall fight with growing confidence and growing strength in the air, we shall defend our island, whatever the cost may be. We shall fight on the beaches, we shall fight on the landing grounds, we shall fight in the fields and in the streets, we shall fight in the hills; we shall never surrender.

- **After descrambling:**

  We shall fight in France, we shall fight on the seas and oceans, we shall fight with growing confidence and growing strength in the air, we shall defend our island, whatever the cost may be. We shall fight on the beaches, we shall fight on the landing grounds, we shall fight in the fields and in the streets, we shall fight in the hills; we shall never surrender.

  Winston Churchill at the House of Commons, June 4, 1940

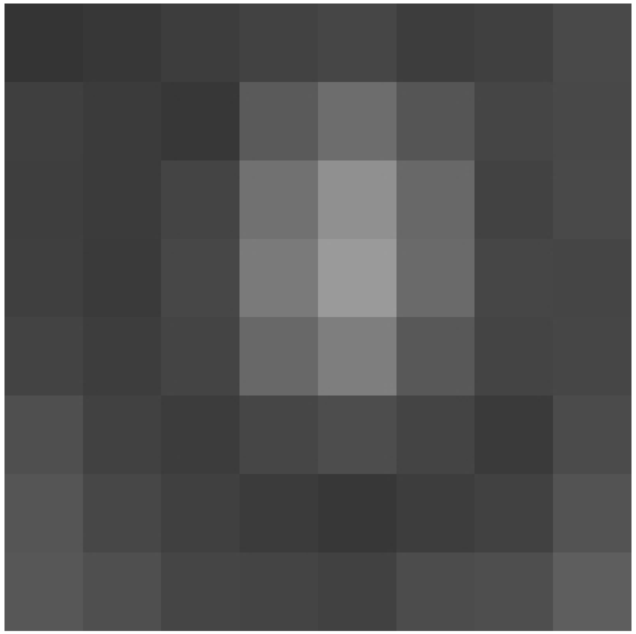
- **The objective of image compression is to reduce redundancy and irrelevance of image data.**

- **Lossless compression (reduces redudancy)**
  - **run-length encoding [BMP, TGA, TIFF] (WWWWWWWWBBWWWWWW $\longrightarrow$ 8W2B6W)**
  - **entropy/deflation/Huffman encoding [PNG, TIFF]**
  - **adaptive dictionary algorithms [GIF, TIFF]**

- **Lossy compression (reduces irrelevance)**
  - **reducing the color space**
  - **chroma subsampling (give less resolution to chroma than to luma)**
  - **transform coding [JPEG, MPEG]**

1. **chroma subsampling**
2. **split the image in 8x8 pixel blocks**
3. **Each channel (RGB) or component (YCbCr) undergoes a discrete Cosine transform (DCT).**
4. **Amplitudes of the frequency components are quantized:**
   - **Human vision is more sensitive to variations in color and brightness over large areas than to high-frequency variations.**
   - **Consequently, high-frequency components are stored with less resolution than low-frequency components.**
   - **The JPEG quality setting affects the strength of the resolution reduction.**
5. **apply Huffman encoding**

- **taking an 8-bit example block centering its values around 0 (i.e., the value range is $[-127, 128]$)**



$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

- **definition of the one-dimensional DCT:**

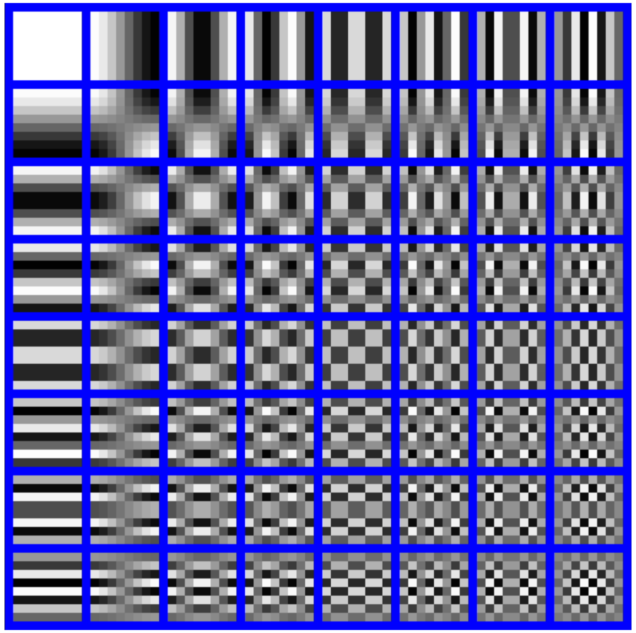
$$F_r = \sum_{k=0}^{N_0-1} f_k \cos\left[\frac{\pi}{N_0}\left(k+\frac{1}{2}\right)r\right]; \quad r \in \{0,\ldots,N_0-1\} \qquad (11)$$

- **This is equivalent to a DFT of $4N_0$ inputs of even symmetry where the even-indexed elements are zero.**

- **definition of the two-dimensional DCT and application to the 8x8 pixel block:**

$$F_{r,s} = \sum_{k=0}^{7}\sum_{l=0}^{7} f_{k,l} \cos\left[\frac{\pi}{8}\left(k+\frac{1}{2}\right)r\right] \cos\left[\frac{\pi}{8}\left(l+\frac{1}{2}\right)s\right]; \; r,s \in \{0,\ldots,7\}. \quad (12)$$

- **The 2-dimensional DCT converts the 8x8 input block into a matrix of values to a linear combination of the below 64 patterns.**
- **These patterns are referred to as basis functions.**



$$F = \begin{bmatrix} -415.38 & -30.19 & -61.20 & \dots & 0.46 \\ 4.47 & -21.86 & -60.76 & \dots & 4.88 \\ -46.83 & 7.37 & 77.13 & \dots & -5.65 \\ -48.53 & 12.07 & 34.10 & \dots & 1.95 \\ 12.12 & -6.55 & -13.20 & \dots & 3.14 \\ -7.73 & 2.91 & 2.38 & \dots & 1.85 \\ -1.03 & 0.18 & 0.42 & \dots & -0.66 \\ -0.17 & 0.14 & -1.07 & \dots & 1.68 \end{bmatrix}$$

- **divide each value in the frequency domain by a constant for that component and round to the nearest integer**
- $B_{r,s} = \text{round}\left(\frac{F_{r,s}}{Q_{r,s}}\right); \quad r,s \in \{1,\ldots,7\}$
- **a typical quantization matrix:**

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantization (cont.)

- **Typically, many higher-frequency values are rounded to 0, and the other values become small positive or negative integers.**

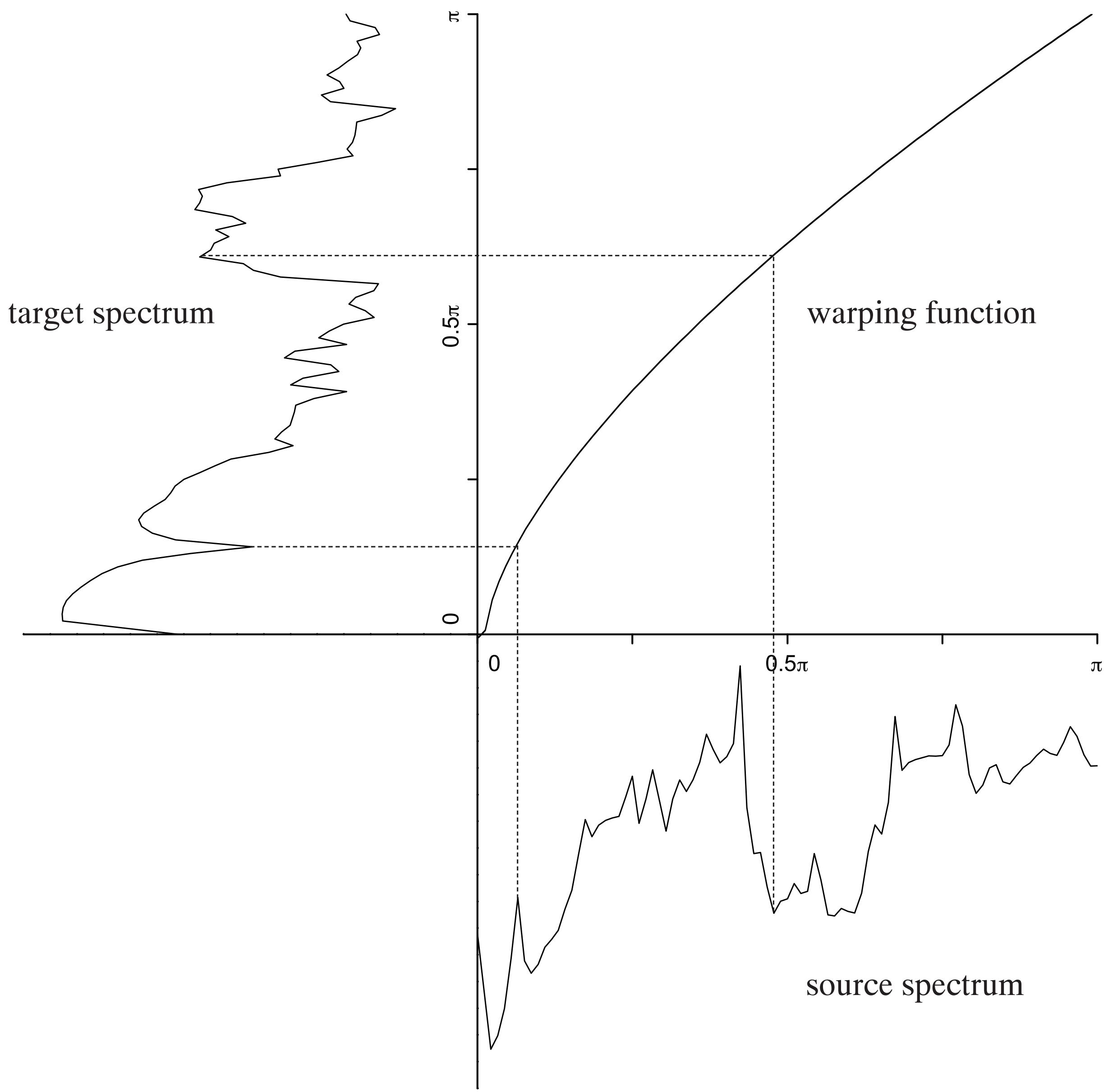
⟶ **Many fewer bits are required for their representation.**

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Vocal tract length normalization

- Vocal tract length normalization (VTLN) tries to compensate for the effect of speaker-dependent vocal tract lengths.
- This is done by warping the frequency axis of the spectrum.
- speech recognition: normalization of a speaker's voice to remove individual speaker characteristics $\longrightarrow$ recognition performance gain
- voice conversion: transformation of a standard speaker to
  - several well-distinguishable individuals or to
  - a given target speaker

VTLN (cont.)


target spectrum
warping function
source spectrum
0
0.5π
π
0
0.5π
π

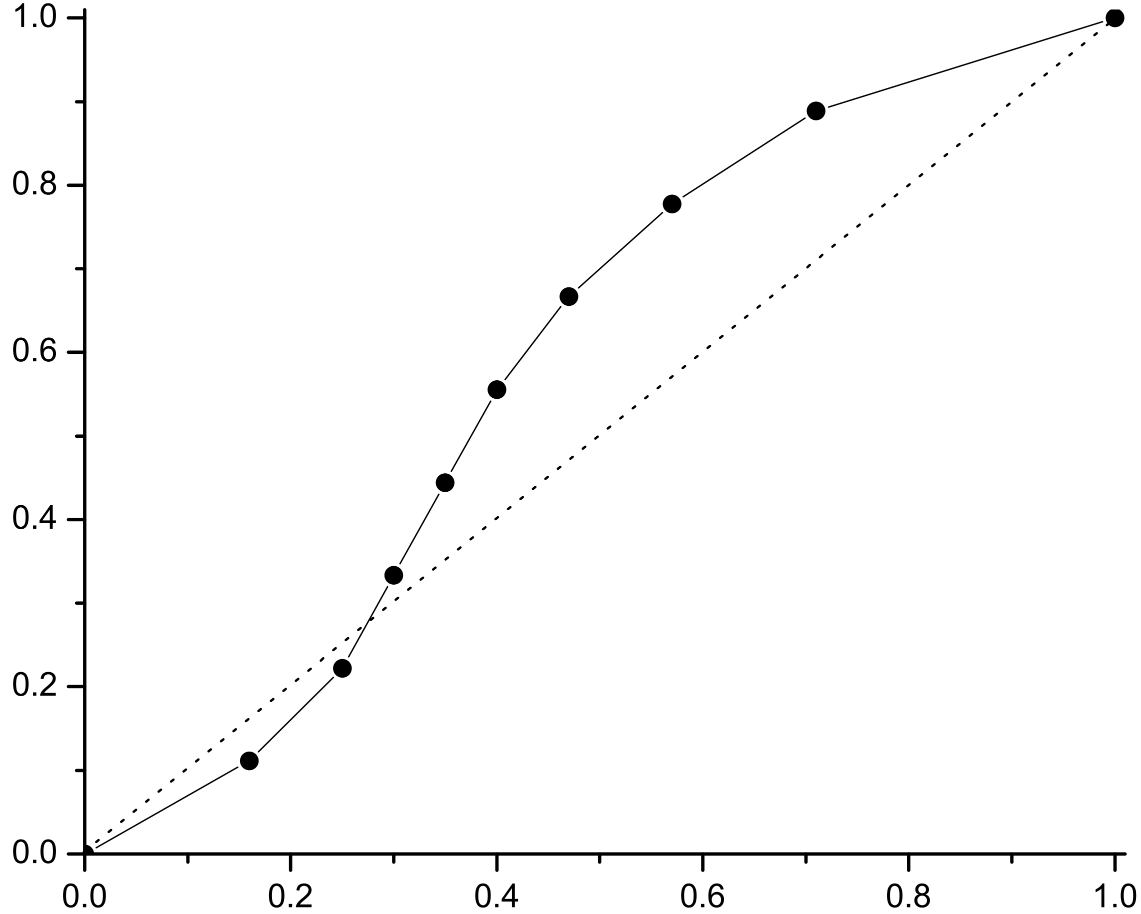
Categorization of VTLN warping functions

parameters	linear	nonlinear
one	• piece-wise linear with two segments – asymmetric – symmetric	• bilinear • power • quadratic
several	• piece-wise linear with several segments	• allpass transform

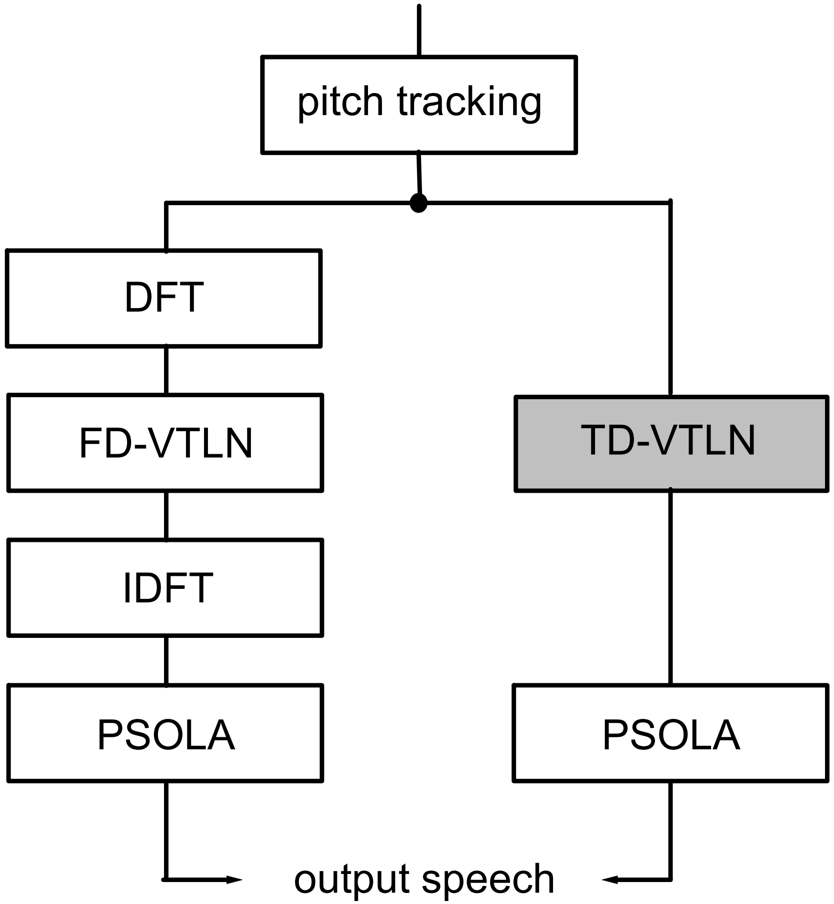
Categorization of VTLN warping functions (cont.)

parameters	linear	nonlinear
one	• piece-wise linear with two segments – asymmetric – symmetric	• bilinear • power • quadratic
several	• piece-wise linear with several segments	• allpass trans-form

Example of a the piece-wise linear warping function



Time-domain VTLN



- **VTLN is directly applied to time frames of a speech signal.**
- **This is done by exploiting time domain correspondences of the Fourier transformation.**
- **Discrete Fourier transformation and its inverse are omitted.**
- **This leads to a considerable real-time factor boost, while speech quality is preserved.**

Time-Domain VTLN – Details

- **The piece-wise linearly warped spectrum can be written as**

$$\begin{aligned}\tilde{X}(\omega|\omega_1^I, \tilde{\omega}_1^I) &= \sum_{i=0}^{I} X\left(\frac{\omega - \beta_i}{\alpha_i}\right) R(\omega|\omega_i, \omega_{i+1}) \\ &=: \sum_{i=0}^{I} X^{(i)}(\omega) R^{(i)}(\omega)\end{aligned}$$

$$\text{with} \quad R(\omega|\omega_i, \omega_{i+1}) = \begin{cases} 1 : & \omega_i < \omega < \omega_{i+1} \\ \frac{1}{2} : & \omega = \omega_i \vee \omega = \omega_{i+1} \\ 0 : & \text{otherwise,} \end{cases} \tag{13}$$

- **and its time correspondence is**

$$\tilde{x} = \sum_{i=0}^{I} \mathcal{F}^{-1}(X^{(i)} R^{(i)}) = \sum_{i=0}^{I} x^{(i)} * r^{(i)}. \tag{14}$$

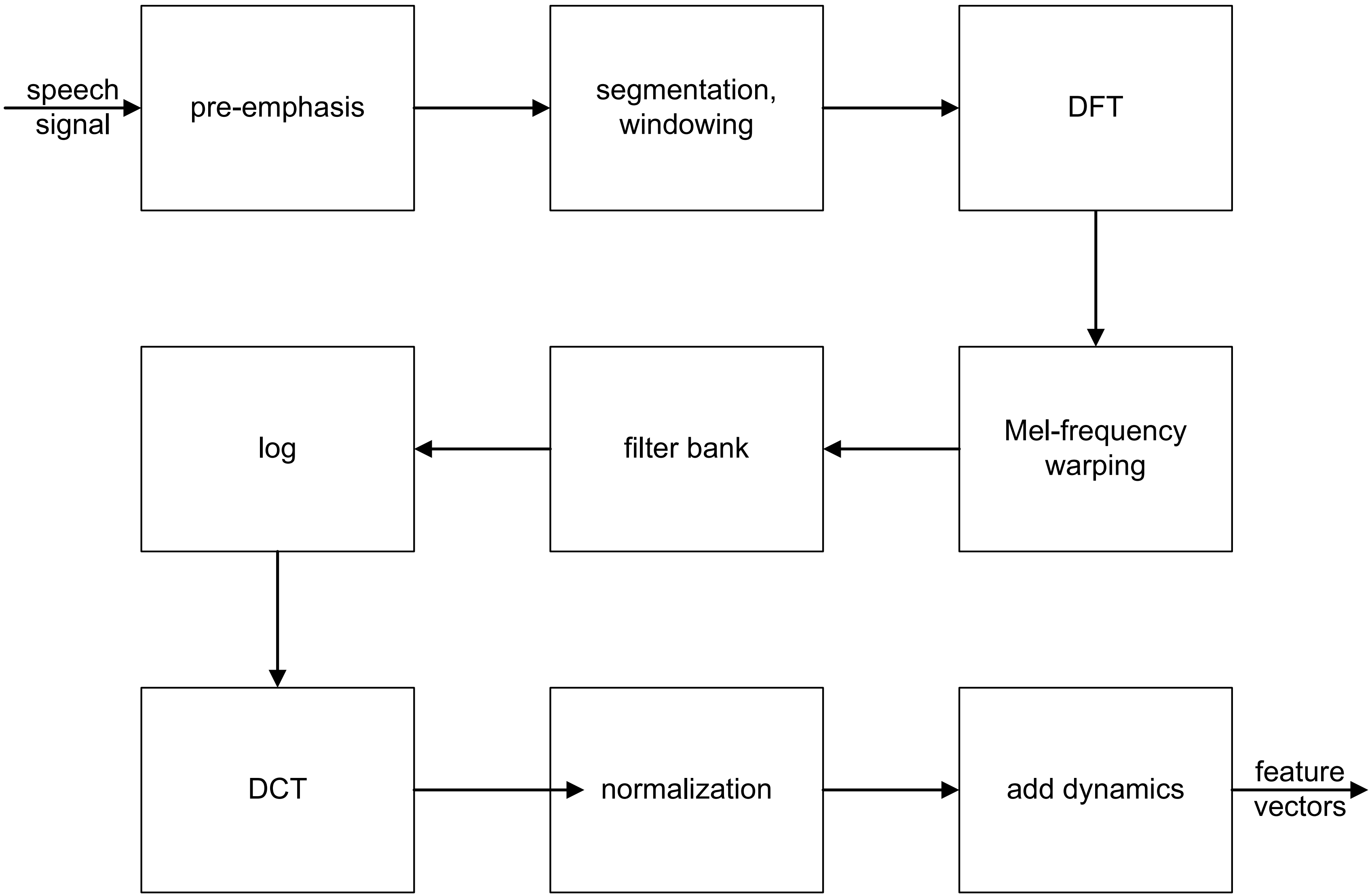
VTLN: sound samples

(sound samples)

Outline

- **introduction**
- **speech recognition**
  - **introduction**
  - **evaluation/Levenshtein distance/dynamic programming**
  - **Fourier transform**
  - **speech analysis**
  - **statistical models**
- **speech synthesis**
- **voice conversion**

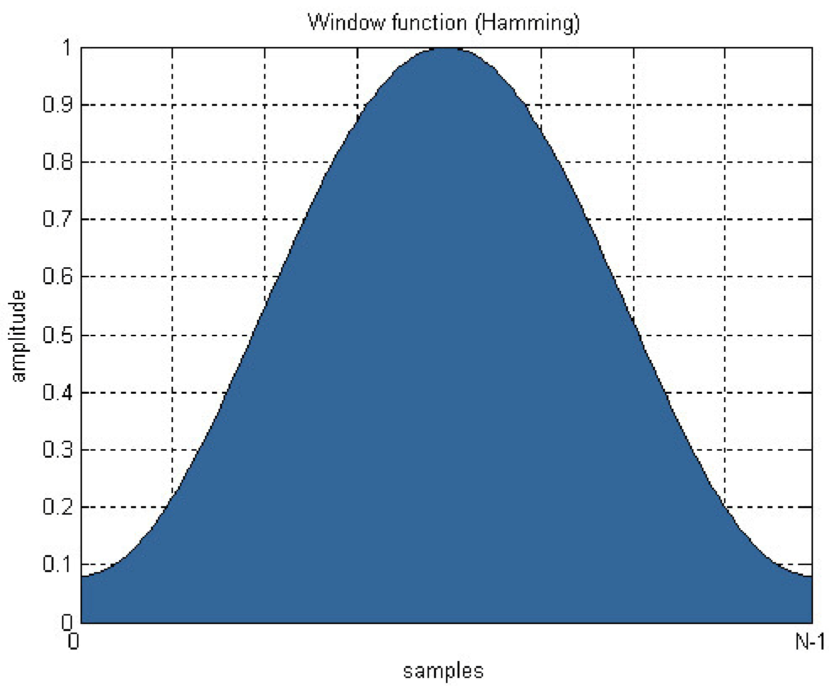
Speech analysis at a glance



1) **Pre-emphasis emphasizes high-frequency signal components (which otherwise account for much less energy than lower frequencies).**

2 a) **Every 10ms, a frame $t \in \{1, \ldots, T\}$ of 25ms width is analyzed (see spectrogram above).**

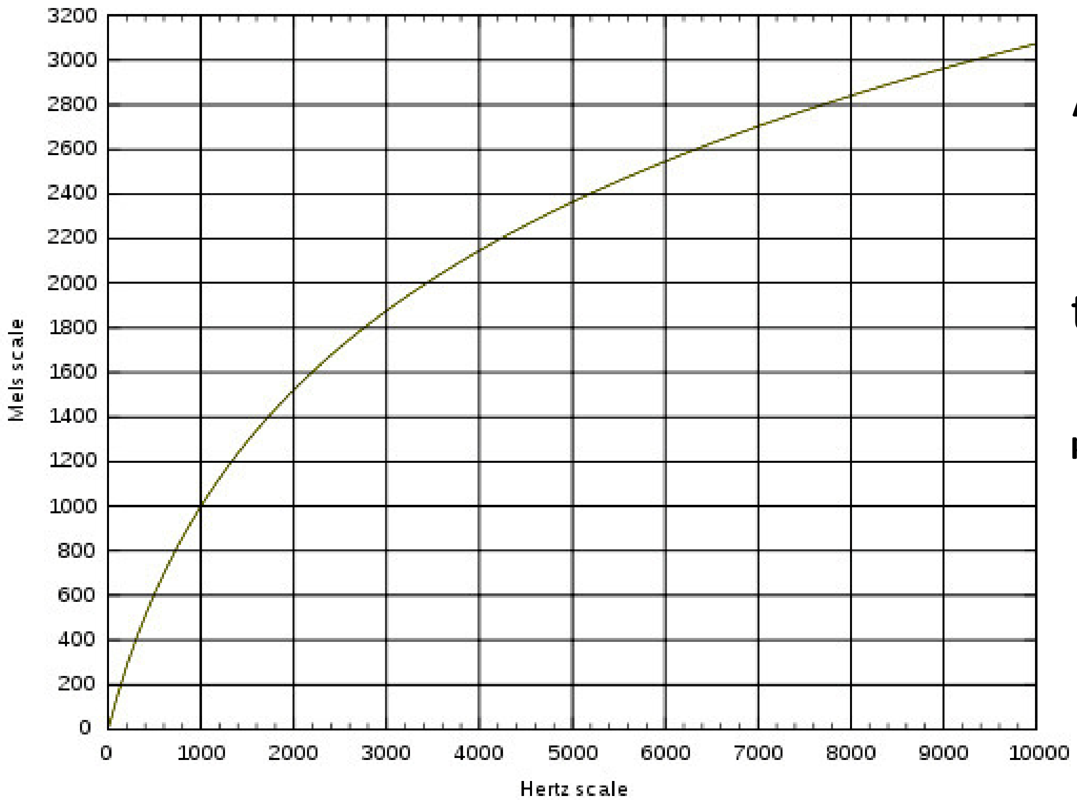
2 b) **This frame is windowed using e.g. a Hamming window:**



$$\begin{aligned} f'_k &= f_k w_k \\ &= f_k \cdot \left(0.54 - 0.46 \cos\left(\frac{2\pi k}{N-1}\right)\right). \end{aligned}$$

3) **DFT** extracts the spectrum $F_{0,t}^{N_0-1}$ from $f'^{N_0-1}_{0,t}$ for every frame t.

4) Warp the spectrum according to the Mel scale (to increase resolution of lower frequency components). Compare our discussions on VTLN for other examples of frequency warping.



Applying the warping function

$$\tilde{f} = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

to $F_{r,t}$ results in $F_{\tilde{r},t}$.

pic:

- source: http://en.wikipedia.org/wiki/File:Mel-Hz_plot.svg
- author: Krishna Vedala
- license: Creative Commons Attribution-Share Alike 3.0 Unported

5) Apply a filter bank summing the magnitude spectrum components (i.e. the absolute values of the spectrum) in bandpass filters (critical bands) windowed with triangular windows:

$$Y_{i,t} = \sum_{\tilde{r}} |F_{\tilde{r},t}| a_{\tilde{r},i} \quad \text{for} \quad i \in \{1, \ldots, I\} \tag{15}$$

with

$$a_{\tilde{r},i} = \begin{cases} b\tilde{r} - i + 1 & : \frac{i-1}{b} \leq \tilde{r} \leq \frac{i}{b} \\ 1 - b\tilde{r} + i & : \frac{i}{b} \leq \tilde{r} \leq \frac{i+1}{b} \\ 0 & : \text{otherwise} \end{cases} \tag{16}$$

Here, b is a constant that depends on the number of desired filter banks I and the number of frequency components N_0:

$$b = \frac{I + 1}{N_0 - 1} \quad (\text{e.g. } N_0 = 100, I = 8). \tag{17}$$

6) Calculate the logarithm of the filter bank output

$$Y'_{i,t} = \log Y_{i,t}. \tag{18}$$

7) Cepstral coefficients are calculated for every frame t using DCT (see also above section on Fourier transform):

$$c_{m,t} = \sum_{i=0}^{I-1} Y'_{i+1,t} \cos\left[\frac{\pi}{I}\left(i + \frac{1}{2}\right) m\right]. \tag{19}$$

8) Mean and energy normalization across all frames:

$$c_{m,t} := c_{m,t} - \frac{1}{T}\sum_{\tau=1}^{T} c_{m,\tau}; \quad c_{0,t} := c_{0,t} - \max_{\tau}\ c_{0,\tau}. \tag{20}$$

9) For composing features vectors, commonly, the first 16 elements are used. Furthermore, to account for signal dynamics, first and second derivatives of the cepstral coefficients are added to produce the final feature vectors:

$$x_t = \begin{bmatrix} c_{1,t}^{16} \\ \Delta c_{1,t}^{16} \\ \Delta\Delta c_{1,t}^{16} \end{bmatrix}. \tag{21}$$

- **In order to compare two feature vectors x and y with the dimensionality D, we need a distance measure (or metric) d with**

$$d : \mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R}^+. \tag{22}$$

- **We call d a metric iff**

  1. $d(x, y) = 0 \quad$ **iff** $\quad x = y$.
  2. $d(x, y) = d(y, x)$ **(symmetry).**
  3. $d(x, y) \leq d(x, z) + d(z, y)$ **(triangle inequality).**

- **Very common is the L_p norm:**

$$||y - x||_p = \sqrt[p]{\sum_{d=1}^{D} |x_d - y_d|^p}. \tag{23}$$

- **Special cases of the L_p norm include**

1. **Manhattan distance**

$$||y - x||_1 = \sum_{d=1}^{D} |x_d - y_d|. \tag{24}$$

2. **Euclidean distance**

$$||y - x||_2 = \sqrt{\sum_{d=1}^{D} (x_d - y_d)^2}. \tag{25}$$

3. **Chebyshev distance**

$$||y - x||_\infty = \max_d \; |x_d - y_d|. \tag{26}$$

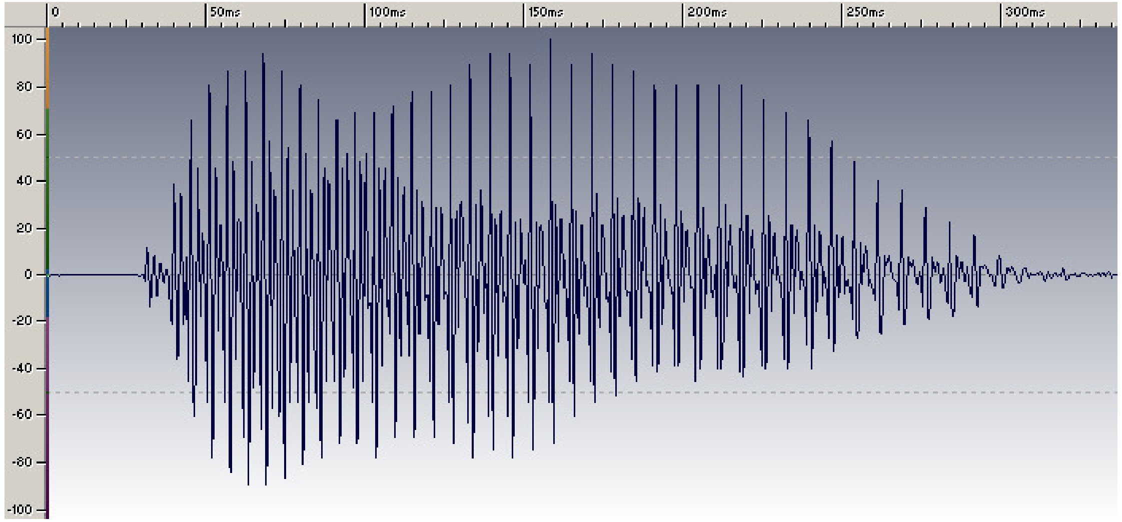
- **We now understand how a speech signal is converted into feature vectors and how such vectors can be compared to each other.**
- **Now, we design a simple speech recognizer for command word recognition as follows:**
  - **At training time, collect M training samples and convert them into feature vector sequences $x_1^{T_1,(1)}, \ldots, x_1^{T_M,(M)}$.**
  - **Annotate each training sample with the respective command w_m for $m \in \{1, \ldots, M\}$.**
  - **At runtime, convert the speech signal to the feature vector sequence x_1^T.**
  - **Now, the recognized command is**

$$\hat{w} := w_{\hat{m}} \text{ with } \hat{m} = \underset{m \in \{1,\ldots,M\}}{\arg\min}\, d'\left(x_1^T, x_1^{T_m,(m)}\right). \tag{27}$$

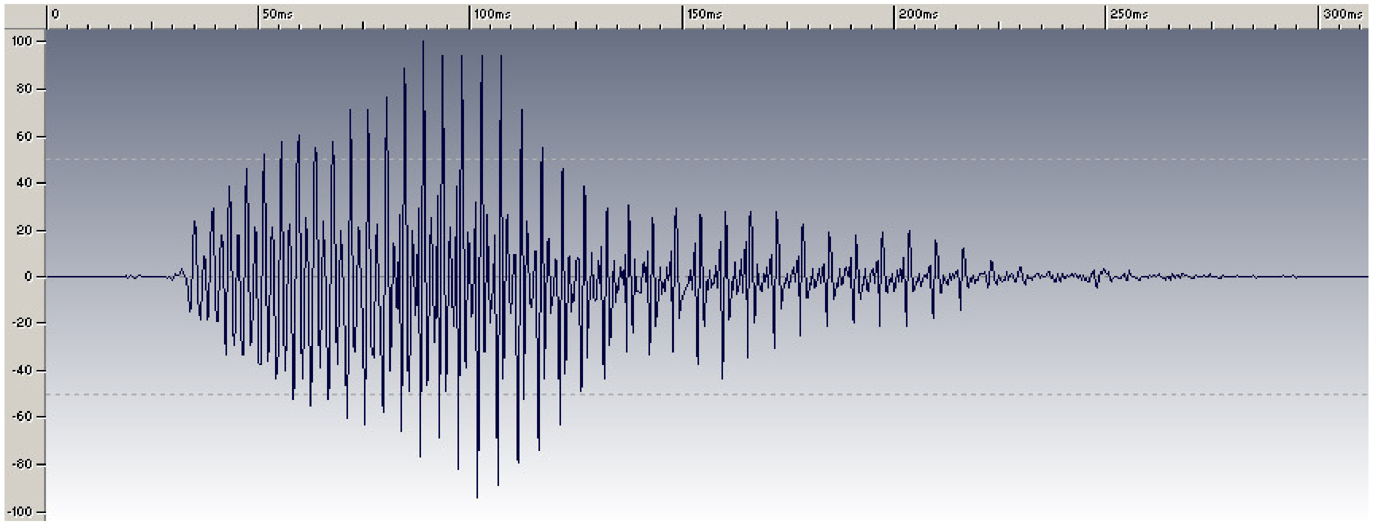
- **Only missing piece in this algorithm is the measure $d'(x_1^T, y_1^U)$ which is to evaluate the distance between the two vector sequences of generally different length and timing patterns.**
- **This is somewhat similar to our considerations on matching two word sequences to measure the Levenshtein distance.**
- **Indeed, d' can be calculated by allowing for vectors to be skipped in x_1^T or y_1^U to best match them to each other (deletions and insertions).**
- **The notion of a substitution changes in that there are, generally, no identical vectors anymore. Instead, every diagonal produces a non-zero penalty. using dynamic programming as well.**

Dynamic time warping: example

to be recognized

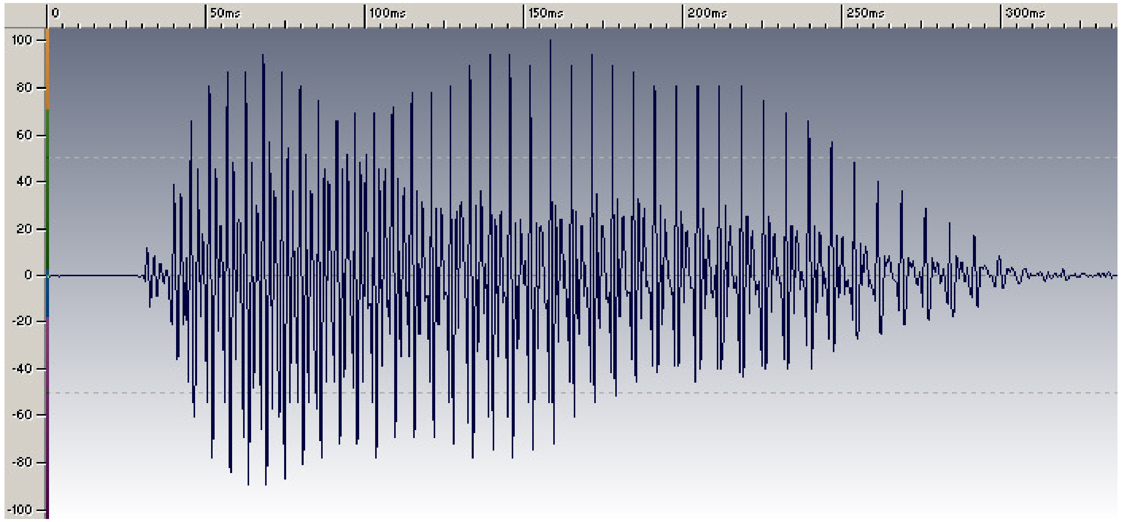


training sample I
“bill”

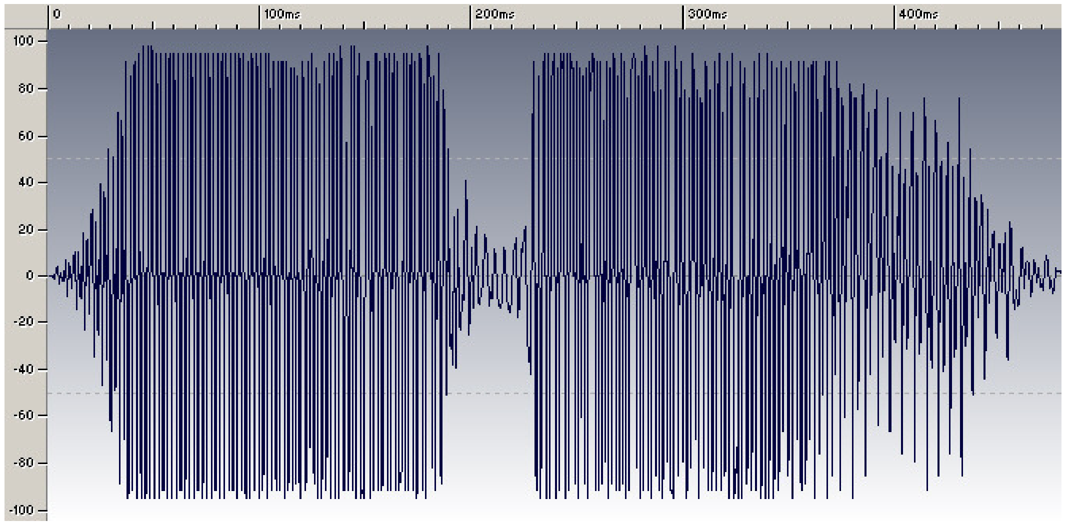


Dynamic time warping: example (cont.)

to be recognized



training sample II
"orders"



to be recognized

$$x_1^6 = \begin{bmatrix} -20 & 0 & -8 & -58 & -126 & -239 \\ -61 & 38 & 53 & 36 & 2 & -69 \\ -0 & -0 & 0 & -0 & -0 & -0 \\ -61 & 38 & 53 & 36 & 2 & -69 \end{bmatrix}$$

training sample I
“bill”

$$x_1^{5,(1)} = \begin{bmatrix} -154 & 0 & -195 & -283 & -327 \\ 10 & 105 & 7 & -47 & -75 \\ 0 & 0 & -0 & -0 & -0 \\ 10 & 105 & 7 & -47 & -75 \end{bmatrix}$$

Dynamic time warping: example (cont.)

to be recognized

$$x_1^6 = \begin{bmatrix} -20 & 0 & -8 & -58 & -126 & -239 \\ -61 & 38 & 53 & 36 & 2 & -69 \\ -0 & -0 & 0 & -0 & -0 & -0 \\ -61 & 38 & 53 & 36 & 2 & -69 \end{bmatrix}$$

training sample II "orders"

$$x_1^{7,(2)} = \begin{bmatrix} -480 & -2 & -14 & -380 & 0 & -247 & -372 \\ -57 & 109 & 56 & -93 & 53 & -14 & -53 \\ -2 & -2 & 1 & -1 & 6 & -2 & 1 \\ -57 & 109 & 56 & -93 & 53 & -14 & -53 \end{bmatrix}$$

training sample I
"bill"

$$d(x_t^{(1)}, x_u) = \begin{bmatrix} 114 & 313 & 97 & 79 & 122 \\ 159 & 95 & 200 & 307 & 364 \\ 158 & 74 & 198 & 309 & 367 \\ 103 & 114 & 143 & 254 & 311 \\ 30 & 193 & 69 & 172 & 229 \\ 140 & 343 & 116 & 54 & 88 \end{bmatrix}$$

training sample II
"orders"

$$d(x_t^{(2)}, x_u) = \begin{bmatrix} 273 & 316 & 254 & 179 & 262 & 78 & 165 \\ 498 & 100 & 29 & 423 & 22 & 258 & 394 \\ 497 & 79 & 7 & 425 & 10 & 257 & 394 \\ 442 & 117 & 52 & 370 & 63 & 202 & 338 \\ 364 & 196 & 136 & 287 & 145 & 123 & 258 \\ 242 & 346 & 286 & 145 & 295 & 78 & 135 \end{bmatrix}$$

- **Similarly to the Levenshtein distance, dynamic time warping requires the search for the grid path inducing minimal costs:**

$$d' = l(T, U). \tag{28}$$

- **Boundary conditions are defined to make sure, matrix edges are never crossed:**

$$l(0,0) = 0; \quad l(t,0) = \infty \text{ for } t \in \{1, \ldots, T\}; \quad l(0,u) = \infty \text{ for } u \in \{1, \ldots, U\}.$$

- **For every grid node, the path with the minimum cost ending in this node is determined.**

- **In the case of symmetric time alignment, the same neighors as for the determination of the Levenshtein distance are considered:**

$$l(t,u) = d(x_t, y_u) + \min\left(\{l(t, u-1), l(t-1, u), l(t-1, u-1)\}\right) \tag{29}$$

- **The standard 0,1,2-model forces the optimal path to be of length T, i.e., d' will have the same number of addends for all $m \in \{1, \ldots, M\}$ resulting in a fair comparison:**

$$l(t,u) = d(x_t, y_u) + \min\left(\{l(t-1, u), l(t-1, u-1), l(t-1, u-2)\}\right) \tag{30}$$

Dynamic time warping: example (cont.)

training sample I
"bill"

$$l^{(1)}(t,u) = \begin{bmatrix} 114 & 427 & 524 & 603 & 725 \\ 273 & 209 & 409 & 716 & 967 \\ 431 & 283 & 407 & 716 & 1083 \\ 534 & 397 & 426 & 661 & 972 \\ 564 & 590 & 466 & 598 & 827 \\ 704 & 907 & 582 & 520 & 608 \end{bmatrix}$$

training sample II
"orders"

$$l^{(2)}(t,u) = \begin{bmatrix} 273 & 589 & 843 & 1022 & 1284 & 1362 & 1527 \\ 771 & 373 & 402 & 825 & 847 & 1105 & 1499 \\ 1268 & 452 & 380 & 805 & 815 & 1072 & 1466 \\ 1710 & 569 & 432 & 750 & 813 & 1015 & 1353 \\ 2074 & 765 & 568 & 719 & 864 & 936 & 1194 \\ 2316 & 1111 & 854 & 713 & 1008 & 942 & 1071 \end{bmatrix}$$

Outline

- **introduction**
- **speech recognition**
  - **introduction**
  - **evaluation/Levenshtein distance/dynamic programming**
  - **Fourier transform**
  - **speech analysis**
  - **statistical models**
- **speech synthesis**
- **voice conversion**

- **The DTW approach requires a comparison between the utterance to be recognized with every sample in the database.**
- **This is not feasible when the database becomes very large (modern speech recognizers are trained using thousands of hours of speech data).**
- **A solution to this problem is to train statistical models describing the essential information contained in the training data.**
- **Advantage of this approach is that**
  - **the models require far less storage than the training database,**
  - **the determination of the optimal class (i.e. word or word sequences) is computationally very cheap and does not depended on the amount of training data.**

- **The goal is determine the optimal word sequence given a feature vector sequence.**
- **This can be done by maximizing the conditional probability:**

$$\begin{aligned} \hat{w}_1^N &= \operatorname*{arg\,max}_{w_1^N} \; p(w_1^N|x_1^T) \\ &= \operatorname*{arg\,max}_{w_1^N} \; \frac{p(w_1^N)p(x_1^T|w_1^N)}{p(x_1^T)} \\ &= \operatorname*{arg\,max}_{w_1^N} \; \underbrace{p(w_1^N)}_{\text{language model}} \; \underbrace{p(x_1^T|w_1^N)}_{\text{acoustic model}} \end{aligned} \tag{31}$$

- **A statistical language model (SLM) assigns a probability to a sequence of words $p(w_1^M)$.**
- **It is a crucial component in many speech and spoken language processing disciplines such as**
  - **speech recognition,**
  - **spoken language understanding,**
  - **machine translation,**
  - **syntactic tagging and parsing.**
- **Due to data sparseness, context is taken into account in a varying degree (unigram SLM, bigram SLM, trigram SLM, $n$gram SLM).**

- **When we do not apply any model assumptions, i.e., the language model takes arbitrary context into account, we have**

$$\begin{aligned} p(w_1^M) &= p(w_1^M) \cdot \frac{p(w_1^{M-1})}{p(w_1^{M-1})} \cdot \frac{p(w_1^{M-2})}{p(w_1^{M-2})} \cdots \frac{p(w_1^2)}{p(w_1^2)} \cdot \frac{p(w_1)}{p(w_1)} \\ &= \frac{p(w_1^M)}{p(w_1^{M-1})} \cdot \frac{p(w_1^{M-1})}{p(w_1^{M-2})} \cdots \frac{p(w_1^2)}{p(w_1)} \cdot p(w_1) \\ &= p(w_M|w_1^{M-1}) \cdot p(w_{M-1}|w_1^{M-2}) \cdots p(w_2|w_1) \cdot p(w_1) \\ &= \prod_{m=1}^{M} p(w_m|w_1^{m-1}) \end{aligned} \tag{32}$$

Unigram SLM

- **A unigram SLM takes no context knowledge into consideration.**
- **That is, the probability of a word w_m is independent of its predecessors (w_{m-1}, etc.) and successors (w_{m+1}, etc.).**
- **Repectively, we have**

$$p(w_1^M) = \prod_{m=1}^{M} p(w_m) \quad (33)$$

Bigram SLM

- **A bigram SLM takes knowledge about a word's predecessor into consideration.**
- **That is, the probability of a word w_m depends on its single predecessor w_{m-1}.**
- **Repectively, we have**

$$p(w_1^M) = p(w_1) \prod_{m=2}^{M} p(w_m|w_{m-1}) \quad (34)$$

Trigram SLM

- **A trigram SLM takes knowledge about a word's two closest predecessors into consideration.**
- **That is, the probability of a word w_m depends on the predecessors w_{m-1} and w_{m-2}.**
- **Repectively, we have**

$$p(w_1^M) = p(w_1)p(w_2|w_1)\prod_{m=3}^{M} p(w_m|w_{m-2}, w_{m-1}) \qquad (35)$$

Smoothing

- In an n-gram model, the probabilities $p(w_m|w_{m-n+1},\dots,w_{m-1})$ are estimated based on counts of the word sequence w_{m-n+1}^m occurring in a training corpus.
- The larger the n-gram order, the more likely it is that these counts happen to be zero making the total probability $p(w_1^M)$ equal to zero.
- The fact that we have no encountered certain (combinations of) words does not mean they do not exist, it may just be that our training corpus is too sparse.
- A technique to overcome this effect is smoothing which discounts some of the probability mass of observed word sequences and assigns it to unobserved ones.
- Popular smoothing techniques include
  - absolute discounting ($n$grams)
  - leaving-one-out ($n$grams)

- **This technique copes with sparseness of $n$gram counts.**
- **Due to the exponential explosion of possible $n$grams with growing order n, data gets sparser and sparser as well.**
- **Consequently, an approach is to back off the $n$gram order in case of zero probabilities.**
- **Absolute discounting discounts non-zero probabilities by an absolute value β_μ and redistributes the probability mass to unseen events backing off by one $n$gram order:**

$$p'(w_m|w_{m-\mu+1}^{m-1}) = \frac{1}{F}\begin{cases} p(w_m|w_{m-\mu+1}^{m-1}) - \beta_\mu & \text{for } p(w_m|w_{m-\mu+1}^{m-1}) > 0 \\ \beta_\mu p'(w_m|w_{m-\mu+2}^{m-1}) & \text{otherwise} \end{cases} \qquad (36)$$

with the normalization constant F.

- **β_μ can be determined based on heuristics or trained on a development set.**

- **In contrast to absolute discounting, this technique linearly combines the probabilities of all $n$gram counts down to the unigram for the given history:**

$$p'(w_m|w_{m-\mu+1}^{m-1}) = \sum_{\mu=1}^{m} \lambda_\mu p(w_m|w_{m-\mu+1}^{m-1}) \quad \textbf{with} \quad \sum_{\mu=1}^{m} \lambda_\mu = 1. \qquad (37)$$

- **Again, λ_μ can be determined based on heuristics (e.g. by discounting non-zero counts of observed events by one–"leaving one out") or trained on a development set.**

- **Now, we want to investigate how we can generate the acoustic model $p(x_1^T|w_1^N)$.**
- **An idea is to represent the acoustic atomic parts in the word sequence (sounds, phonemes) by states of a (stochastic) finite state machine.**
- **That is, for each feature vector x_t we have a corresponding state s_t which is, however, unknown (hidden).**
- **Invoking all possible hidden state sequences s_1^T, the sought-after model can be written as**

$$p(x_1^T|w_1^N) = \sum_{s_1^T} p(x_1^T, s_1^T|w_1^N). \tag{38}$$

with

$$p(x_1^T, s_1^T|w_1^N) = \prod_{t=1}^{T} p(x_t, s_t|x_1^{t-1}, s_1^{t-1}, w_1^N) \tag{39}$$

(see M-gram model for a prove of a similar equivalence).

- **To render the model tractable, as in the case of the language model, several simplifications are assumed.**
- **It is assumed that the hidden phonetic states model the time dependence sufficiently, so, the additional time dependence between the feature vectors is dropped:**

$$p(x_t, s_t | x_1^{t-1}, s_1^{t-1}, w_1^N) = p(x_t, s_t | s_1^{t-1}, w_1^N). \quad (40)$$

- **Furthermore, time dependence is restricted to the predecessor state:**

$$\begin{aligned} p(x_t, s_t | s_1^{t-1}, w_1^N) &= p(x_t, s_t | s_{t-1}, w_1^N) \\ &= \frac{p(x_t, s_{t-1}^t, w_1^N)}{p(s_{t-1}, w_1^N)} \cdot \frac{p(s_{t-1}^t, w_1^N)}{p(s_{t-1}^t, w_1^N)} \\ &= \frac{p(x_t, s_{t-1}^t, w_1^N)}{p(s_{t-1}^t, w_1^N)} \cdot \frac{p(s_{t-1}^t, w_1^N)}{p(s_{t-1}, w_1^N)} \\ &= \underbrace{p(x_t | s_{t-1}^t, w_1^N)}_{\text{emmission probability}} \cdot \underbrace{p(s_t | s_{t-1}, w_1^N)}_{\text{transition probability}} \quad (41) \end{aligned}$$

Notes for the presentations

- **important dates:**

proposal due	**April 29**
presentations	**May 15**

- **Generally, proposals are expected to cover one of the areas discussed at the very beginning of the lecture excluding the highlighted ones.**
- **Please submit your proposals to all of the following e-mail addresses:**

  `david@suendermann.com`

  `david@speechcycle.com`

  `suendermann@dhbw-stuttgart.de`

- **Presentations are to be in English and have a duration of 20 minutes.**