SPEECH RECOGNITION
SS 04

Prof. Dr.-Ing. Hermann Ney

Lehrstuhl für Informatik VI

Computer Science Department

RWTH Aachen, University of Technology

D-52056 Aachen

Contents

0 Computer Science Informatik VI (i6) 3

1 Time Alignment and Isolated Word Recognition 16

2 Statistical Interpretation and Models 109

3 Connected Word Recognition 142

4 Large Vocabulary Speech Recognition 177

5 Search using Lexical Pronunciation Tree 274

6 Word graph method using word pair approximation 323

7 Across-Word Models 360

8 Time Conditioned Search 380

9 Phonetic Decision Trees 402

0 Computer Science Informatik VI (i6)

0.1 Research Topics 4

0.2 Projects 6

0.3 Courses 9

0.4 Textbooks 12

0.1 Research Topics

Method: Stochastic Modelling

- **Modelling Dependencies and Vague Knowledge (contrast: rule-based approach)**
- **Decision Making, in particular in Context**
- **Automatic Learning from Data/Examples**

Applications:

Human Language Technology and Pattern Recognition

- Speech recognition
  - small vocabulary
  - large vocabulary
- Machine translation
- Natural language processing
  - text/document classification
  - information retrieval
  - parsing and syntactic analysis
- Language understanding and dialog systems
- Image recognition
  - object recognition
  - handwriting recognition
- Diagnosis and expert systems
- Other applications:
  - speaker verification and identification
  - fingerprint verification and identification
  - DNA sequence identification
  - gesture recognition
  - lip reading
  - geological analysis
  - high-energy physics: bubble chamber tracks
  - ...

0.2 Projects

- **ARISE (EU):**
  Automatic Railway Information Systems across Europe
  – Speech Recognition and Language Modelling
- **EuTrans II (EU):**
  Translation of Spoken Language
  – Speech Recognition and Translation
- **Institut für deutsche Sprache (IdS):**
  – Language Modelling for Newspapers
- **Audio Document Retrieval (NRW):**
  – Speech Recognition and Information Retrieval
- **Verbmobil II (BMBF):**
  Speech Recognition and Translation for
  Appointment Scheduling and Traveling Information
  – Speech Recognition
  – Speech Translation
  – Prototype Modules

- **Image Object Recognition (RWTH):**
  - **OCR (optical character recognition)**
  - **Medical Images**
- **Advisor (EU):**
  - **Speech Recognition for German Broadcast News**
- **EGYPT follow-up (NSF):**
  - **Basic Algorithms for Statistical Machine Translation**
- **Audio Document Retrieval (NRW ?):**
  - **German Broadcast News: Recognition and Information Retrieval**
- **Bilateral Projects with Companies (including start-ups)**

- German DFG:
  - Improved Acoustic Modelling using Structured Models
- TC-Star (EU):
  - Technology and Corpora for Speech to Speech Translation
  - Applications: Broadcast News and Speeches/Lectures
- Coretex (EU):
  - Improving Core Technology for Speech Recognition
  - Applications: Broadcast News in Several Languages
- LC-Star (EU):
  - Lexical and Corpora Resources for Recognition, Translation and Synthesis
  - Prototype system for machine translation of spoken sentences
- Transtype-2 (EU):
  - Machine translation of written text
  - Application: interactive machine-aided translation
- PF-Star (EU):
  - Machine translation of spoken dialogues
  - Application: tourism and travelling

0.3 Courses

- Lectures (L4) with exercises (E2):
  - SR: Speech Recognition
  - PR1: Pattern Recognition + Neural Nets
  - SI: Signal + Image Processing
  - NLP: Natural Language Processing
- Lectures (L2) with exercises (E1):
  - PR0: 'Man-Machine Interface'
  - PR2: Stochastic Modelling
  - LM: Language Modelling
  - MT: Machine Translation
  - SR2: Speech Recognition II (?)
  - MIP: Medical Image Processing ('Ringvorlesung', each WS)
- Seminars: – Diplom Degree (SS, Block)
  – Doctor Degree (WS+SS)
- Laboratory Courses (WS, Block)
- Study Groups (WS+SS: speech, language, image)

course cycles of (about) two years:

year	term	lectures	
02/03	WS	SR (L4/E2)	–
	SS	SI (L4/E2)	PR2 (L2/E1)
03/04	WS	PR1 (L4/E2)	NLP (L2/E1)
	SS	SR (L4/E2)	–
04/05	WS	NLP (L4/E2)	–
	SS	PR1 (L4/E2)	SI (L4/E2)

- **area of specialization (Vertiefungsgebiet) i6 with the topics:**
  - **Speech + Pattern Recognition**
  - **Speech + Language Processing**
  - **Signal + Image Processing**
  - **...**

  select 12 hours (SWS) out of i6 lectures

- **practical computer science (Prakt. Informatik) (3 areas):**
  recommendation: 12 hours (SWS) out of

  L4: Signal + Image Processing (SI, ...)
  L4: Language (LM, MT, NLP, ...)
  or Speech (SR, LM, ...)
  L4: i6-external lectures:
  - **data bases**
  - **artificial intelligence**
  - **...**

  additional alternatives: on demand

- **Master in Media Informatics or Software Systems Engineering:**
  credit system: examination after each course
- **Technische Informatik (Diplom):**
  oral exam at the end of the lecture (exception)
- **Technischer Redakteur:**
  more or less similar to Diplom degree
- **ERASMUS students of Computer Science:**
  oral exam/colloquium for graded certificate at the end of the lecture

0.4 Textbooks

Textbooks on Speech Recognition:

- **emphasis on signal processing and small-vocabulary recognition:**
  L. Rabiner, B. H. Juang: Fundamentals of Speech Recognition.
  Prentice Hall, Englewood Cliffs, NJ, 1993.
- **emphasis on large vocabulary and language modelling:**
  F. Jelinek: Statistical Methods for Speech Recognition.
  MIT Press, Cambridge, 1997.
- **introduction to both speech and language:**
  D. Jurafsky, J. H. Martin: Speech and Language Processing. Prentice Hall, Englewood Cliffs, NJ, 2000.
- **advanced topics:**
  R. De Mori: Spoken Dialogues with Computers.
  Academic Press, London, 1998

Textbooks on Natural Language Processing (statistical/corpus-based):

- **introduction to both speech and language:**
  D. Jurafsky, J. H. Martin: Speech and Language Processing. Prentice Hall, Englewood Cliffs, NJ, 2000.
- **emphasis on statistical methods for written language:**
  C. D. Manning, H. Schütze: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, 1999.
- **related field: artificial intelligence:**
  S. Russel, P. Norvig: Artificial Intelligence. Prentice Hall, Englewood Cliffs, NJ, 1995 (in particular Chapters 22-25).

Textbooks on Statistical Learning (Pattern Recognition, Neural Networks, Data Mining, ...):

- **best introduction (including modern concepts):**
  R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification. 2nd ed., J. Wiley & Sons, New York, NY, 2001.
- **emphasis on statistical concepts:**
  B. D. Ripley: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge, England, 1996.
- **emphasis on modern statistical concepts:**
  T. Hastie, R. Tibshirani, J. Friedman: The Elements of Statistical Learning: Data Mining, Inference and Predictions. Springer, New York, 2001.
- **emphasis on theory and principles:**
  L. Devroye, J. Györfi, G. Lugosi: A Probabilistic Theory of Pattern Recognition. Springer, New York, 1996.

Textbooks on mathematical methods (vector spaces and matrices, statistics, optimization methods, ...):

- **best overall summary:**
  T. K. Moon, W. C. Stirling: Mathematical Methods and Algorithms for Signal Processing. Prentice Hall, Upper Saddle River, NJ, 2000.
- **introduction to modern statistics:**
  G. Casella, R. L. Berger: Statistical Inference. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1990.
- **good overview of numerical algorithms and implementations:**
  W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery: Numerical Recipes in C. Cambridge Univ. Press, Cambridge, 2nd ed., 1992.

1 Time Alignment and Isolated Word Recognition

1.1 What is speech recognition? 17

1.2 Speech Signal Analysis 34

1.3 Distance Measures 46

1.4 Time Alignment 61

1.5 Dynamic Programming 71

1.6 Isolated Word Recognition 97

1.1 What is speech recognition?

Speech recognition means:

convert the acoustic signal (sound) into a sequence of written words (text)

Related tasks:

- **Speech understanding: generating a semantic representation**
- **Speaker recognition: identifying the person who spoke**
- **Speech detection: separating speech from non-speech**
- **Speech enhancement: improve the intelligibility of a signal**
- **Speech compression: encode speech signal for transmission or storage with a small amount of bits**

Terminology: Speech Recognition vs. Understanding

- **Speech recognition (Spracherkennung)**
  typical application: dictation,
  i.e. speech to text;
  understanding is secondary.

- **Speech (or language) understanding (Sprachverstehen)**
  recognition AND 'logical' understanding:
  - **easy application:**
    Recognize 1 of K voice commands and
    carry them out (e.g. name dialing).
  - **difficult application:**
    Spoken dialogue system with natural language
    input (e.g. travel information)

History of speech and language technology:

- 3000 BC: non-alphabetic writing systems: cuneiform script, hieroglyphs, ...
- 1000 BC: Phoenician and Greek alphabets
- 1450: printing press (J. Gutenberg)
- 1876: telephone (A. Bell)
- 1877: phonograph (drum) (T. Edison)
  gramophone record (disk) (E. Berliner 1888)
- 1906: radio broadcast (R. Fessenden)
- 1935: audio tape (AEG)
- 1968: digital telephony (PCM, in London)
- 1982: (digital) compact disc (Philips/Sony)
- 1980: text processing with computers
- 1995: automatic speech recognition:
  dictation systems, dialogue systems

1 Book ('standard'):

= 400 pages * 50 lines/page * 10 words/line
= 200 000 words
= 2 * 10^6 letters
= 2 Mbyte

Project Thesaurus Linguae Graecae,
UC Irvine, Irvine, CA:

- 'classical' texts: 800 BC – 600 AD
  'Byzantine' texts: 600 AD - 1453 AD
- 73 million running words
  = 730 Mbyte
  = 365 books
  = 1.14 CD-ROM

as comparison:

1 year of Wall Street Journal
= 10 million running words

Development (in research):

1965	first tests
1975	isolated words
	each word spoken in isolation, small vocabulary
1985	isolated: 5000 words
	continuous; 1000 words
1990	continuous: 10 000 words
1995	telephone systems: speaker independent,
	continuous, 3 000 words

Applications:

- command & control
- text dictation ("listening typewriter")
- spoken dialog information systems
  (recognition AND understanding)

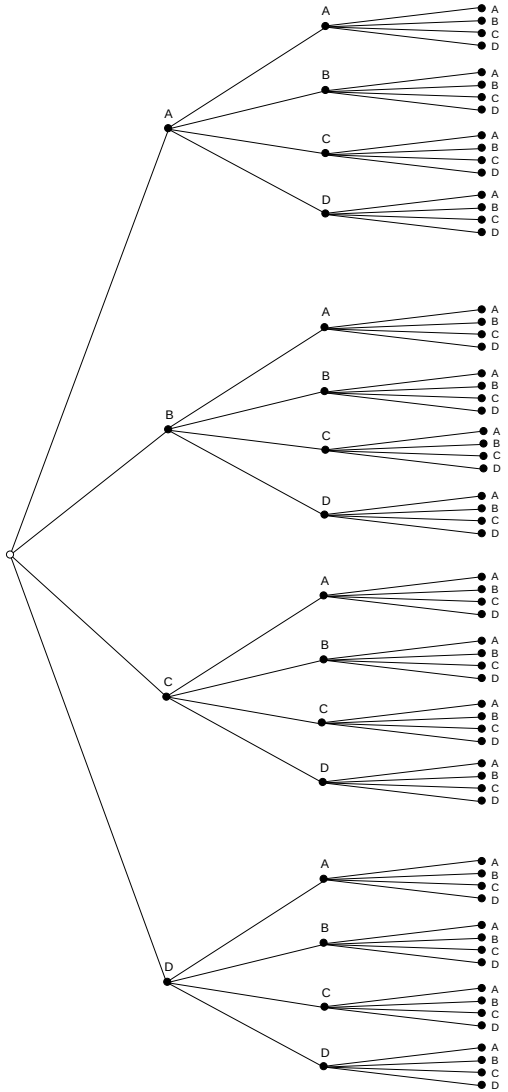
- **Command & Control**
  - **voice dialing**
  - **car radio, navigation system**
  - **data entry, hands-free operation**
- **Text Dictation ("listening typewriter")**
  - **medical reports**
  - **business letters**
- **Spoken dialog information system**
  access to information systems by phone, e.g.:
  - **voice mail system**
  - **telephone directory assistance**
  - **travelling information (trains, flights, hotels, weather,...)**
    (Philips system: 0241/604020)
- **Speech-to-speech translation**
- **Spoken document retrieval**
  Finding audio documents in large archives

Speech recognition is hard due to several factors:

- high variability of the signal
- most decisions are interdependent
  - word and phoneme (sound) boundaries are not visible in the signal
  - high variations in speaking rate
- decision in context:
  recognize whole sentences rather than single words
  $\rightarrow$ large number of classes to be distinguished

Example: text dictation ("listening typewriter")

- vocabulary: 20 000 words
  number of 10-word sentences:
  $20\,000^{10} = 10^{43}$



Sources of variability of the signal:

- from utterance to utterance
- speaking rate can vary drastically
- different speakers
  - male/female, old/young, ...
  - dialect and accent
- recording conditions:
  - environmental noise (office, car, shopping mall, ...)
  - type of microphone
  - channel characteristics (high-quality, telephone, mobile phone, ...)
- spontaneous speech effects:
  - hesitation
  - false starts, ...

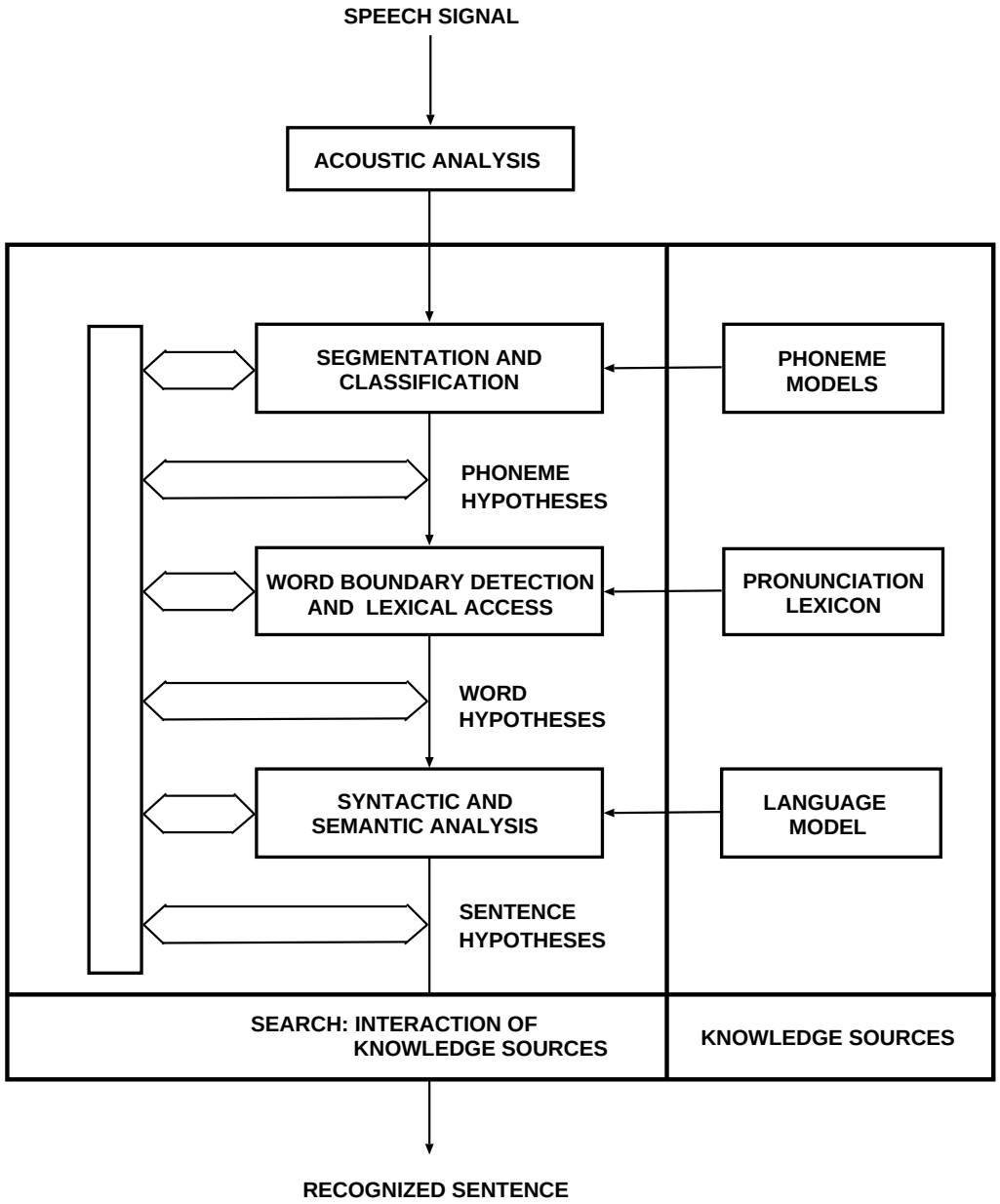
Key Ideas:

- **put all ambiguities in probability distributions (stochastic knowledge sources)**
- **stochastic modelling in speech recognition:**
  - **phoneme (or word) models**
  - **pronunciation lexicon**
  - **language model**
- **training: use data to train the free parameters of the models**
- **leave all the interdependencies and ambiguities to a search process,**
  e.g. 16 values/10 msec = 32 000 values/20 sec:
  - **optimal interaction between all knowledge sources**
  - **(virtually) no local (=intermediate) decisions**
  - **no distinction between statistical and syntactic pattern recognition**

  $\rightarrow$ holistic approach to decision making

contrast: rule-based system (à la Prolog) with hard decisions at intermediate levels

Knowledge Sources and Interactions in Speech Recognition



When comparing speech recognition tasks several points have to be considered:

- speaker dependent or independent
- isolated words or continuous speech
- vocabulary
  - confusibility increases with size of vocabulary
  - closed vocabulary (input is restricted to the fixed vocabulary), or
    open vocabulary ("unknown" words occur)
- quality of speech:
  - planned vs. spontaneous speech
  - cooperative vs. non-cooperative speaker
- recording conditions
  - channel: telephone, mobile phone, ...
  - noise
  - position of microphone: headset, room microphone, ...
- real time operation: yes/no

How to fairly evaluate the performance of a speech recognition system?

- Use only unseen data for evaluation
- Common performance measure:

$$\text{word error rate} = \frac{\text{edit distance}}{\text{spoken words}}$$

  edit distance := minimum number of substitution, deletion and insertion errors

- Comparison of different systems requires standardized speech corpora for training and testing.
- Additional criteria: real time factor, memory requirements, software complexity

Out of vocabulary (OOV) words:

- **words in the testing corpus that are not included in the recognition vocabulary**
- **these words can not be recognized correctly**
- **the OOV rate [%] is a lower bound for the word error rate**
- **every OOV word leads to at least one recognition error, but the average is about 2 errors per OOV word.**

Example from the Verbmobil Corpus play /example-verbmobil-2.wav

Spoken:

> **also ich vielleicht ist grade zu der Zeit die CeBit das wäre vielleicht für uns fachlich auch ganz interessant**

Recognized:

> **also ich vielleicht das grade zu der Zeit die CeBit das wäre vielleicht — uns fachlich auch noch ganz interessant**

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

$$\mathrm{WER} = \frac{1\ \mathrm{deletion} + 1\ \mathrm{insertion} + 1\ \mathrm{substitution}}{19\ \mathrm{spoken\ words}} = 15.8\%$$

SieTill

language:	German	**domain:**	digit strings
recording:	telephone	**vocabulary:**	11 digits (including “zwo”)
examples:	play example-sietill-1.wav	play example-sietill-2.wav	

Arise (Automatic Railway Information System Across Europe)

language:	Dutch	**domain:**	timetable information
recording:	telephone	**vocabulary:**	1000 words
examples:	play example-arise-1.wav		

Wall Street Journal 5k

language:	American English	**domain:**	news paper text
recording:	studio quality, read speech	**vocabulary:**	5000 words
examples:	play example-wsj-1.wav	play example-wsj-2.wav	

Verbmobil 2

language:	**German**
domain:	**appointments, travel information**
recording:	**office environment, high quality and telephone recordings**
	spontaneous speech, various dialects
vocabulary:	**10000 words**
examples:	**play** `example-verbmobil-1.wav` | **play** `example-verbmobil-3.wav`

Hub4 (Broadcast News)

language: American English
domain: TV and radio broadcasts (CNN Headline News, NPR All things considered, ...)
recording: various conditions (studio , interviews, reporters, ...)
vocabulary: 65000 words
examples: `play example-hub4.k`

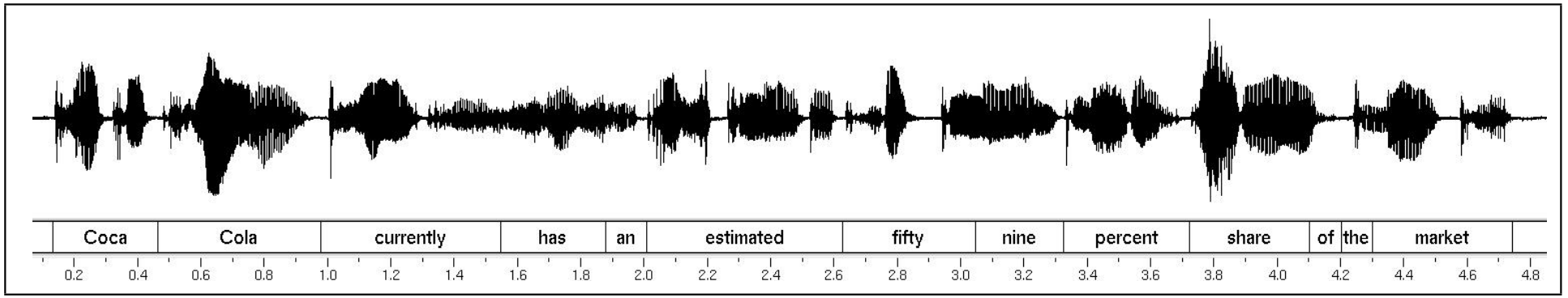
Advisor (Broadcast News)

language: German
domain: TV and radio broadcasts (Report Mainz (SWR), ...)
recording: various conditions (studio , interviews, reporters, ...)
vocabulary: 62000 words
examples: `play example-advisor.k`

1.2 Speech Signal Analysis

The acoustic signal is recorded by a microphone and sampled at a frequency of (say) 16 kHz and converted into a sequence of 16-bit numbers.

An example from the Wall Street Journal corpus: **play** `example-1.wav`

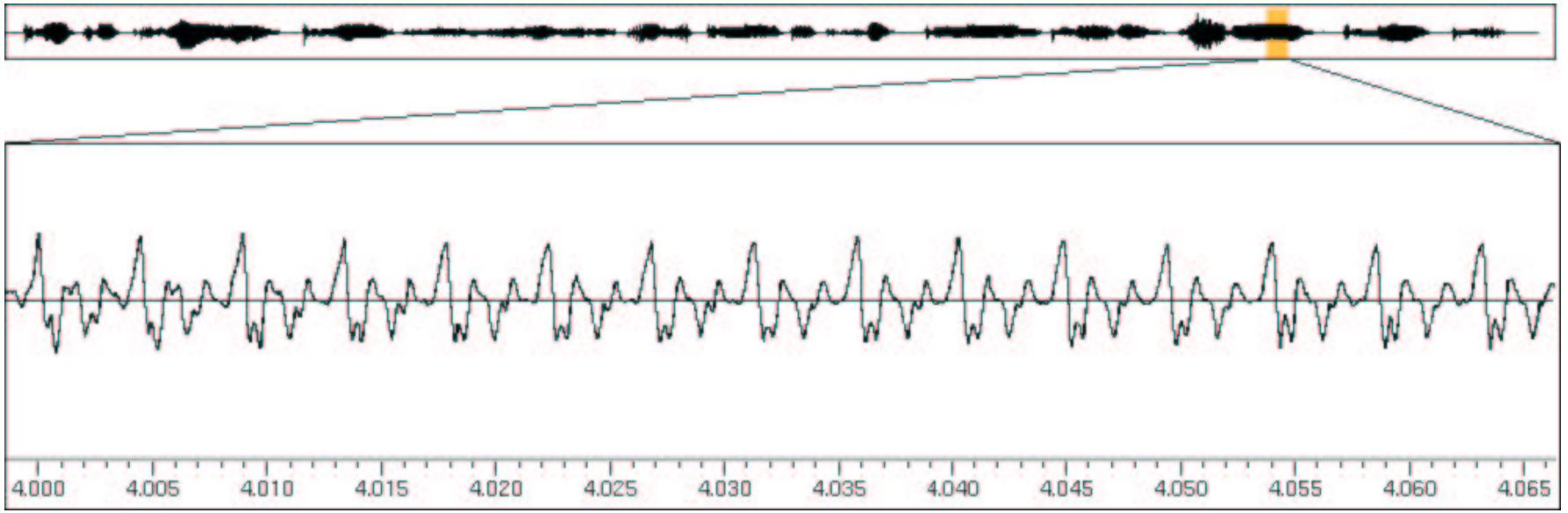


This acoustic waveform shows very little direct cues about what might have been said.

Note that not even the word boundaries are obvious.

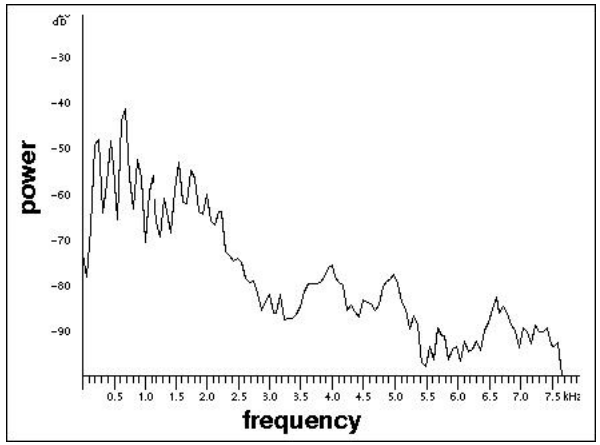
On short time scales (about 10 ms) the speech signal is almost periodic.

This segment from the previous example shows the “A” in “share”:



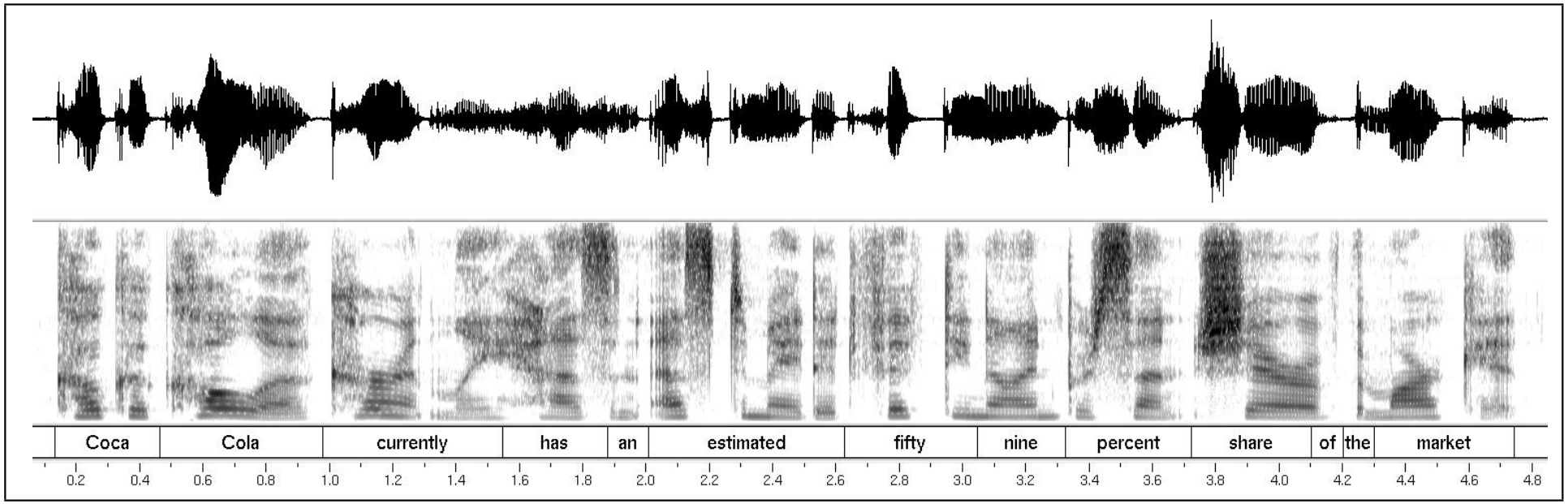
The Fourier transform decomposes the signal into its frequency components.

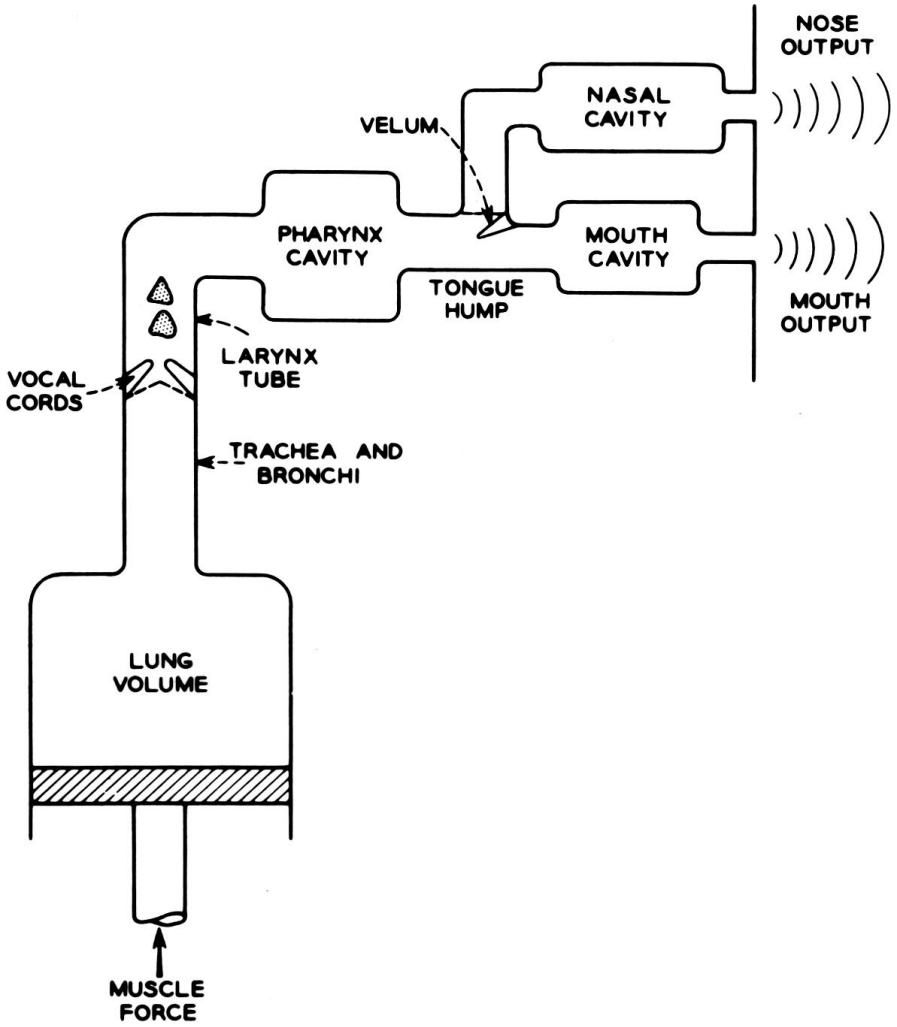
→ power spectrum

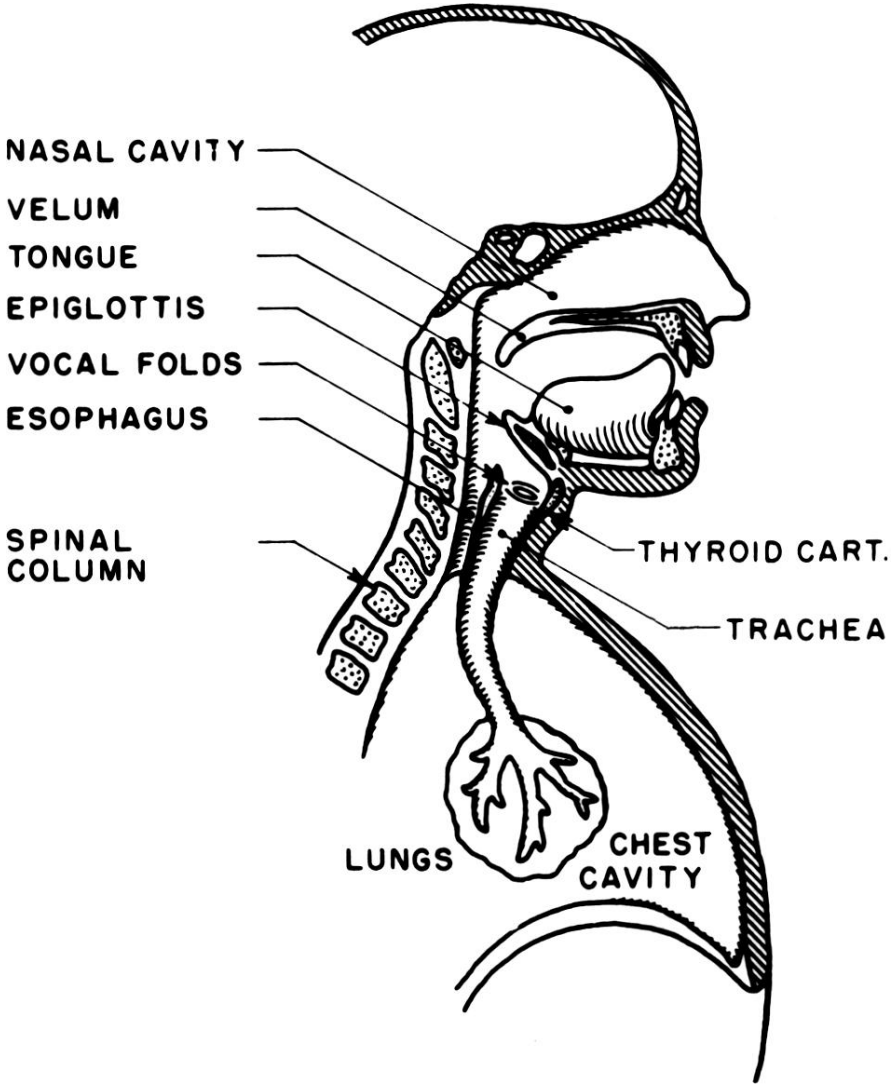


The speech signal is divided into overlapping windows approximately 25ms long and 10ms apart. For each so-called time-frame the power spectrum is calculated.

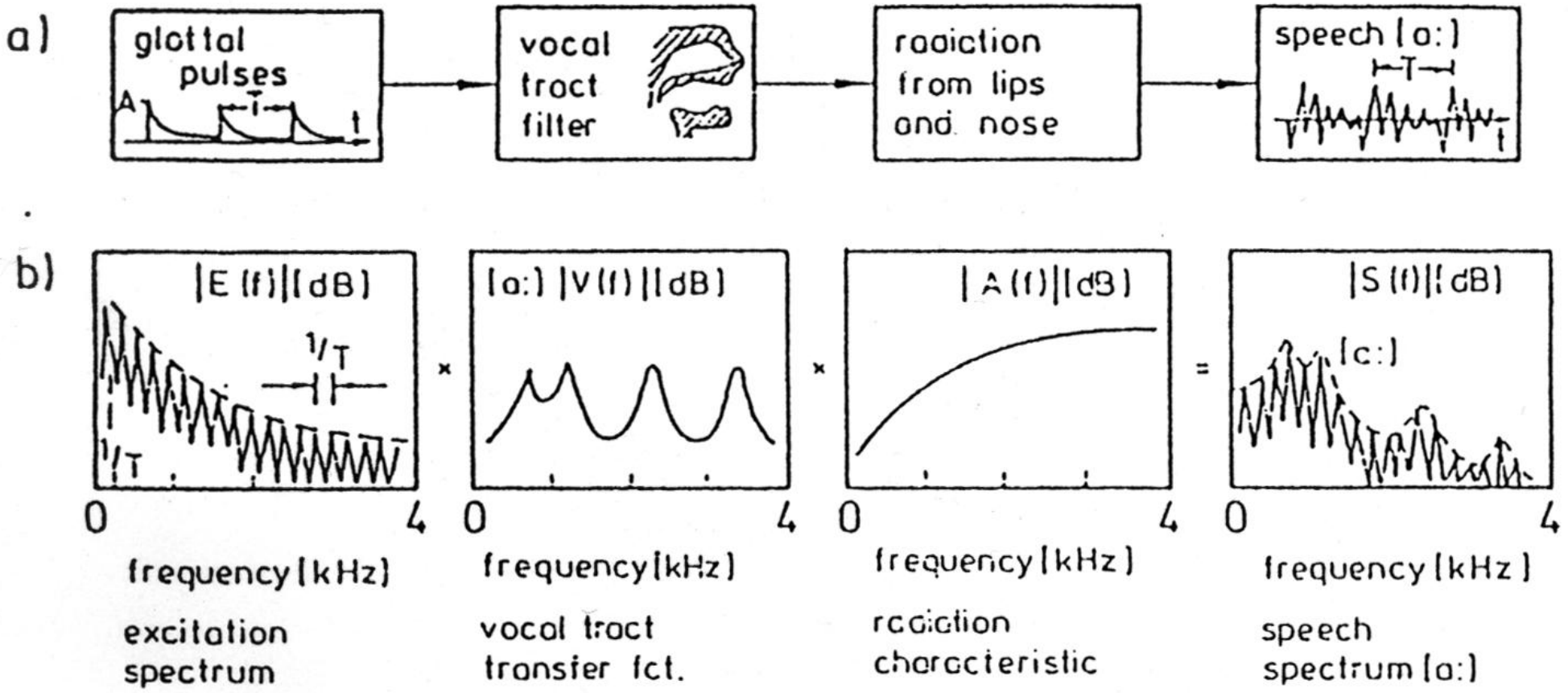
This results in a spectrogram which shows the spectral energy distribution for each time-frame:



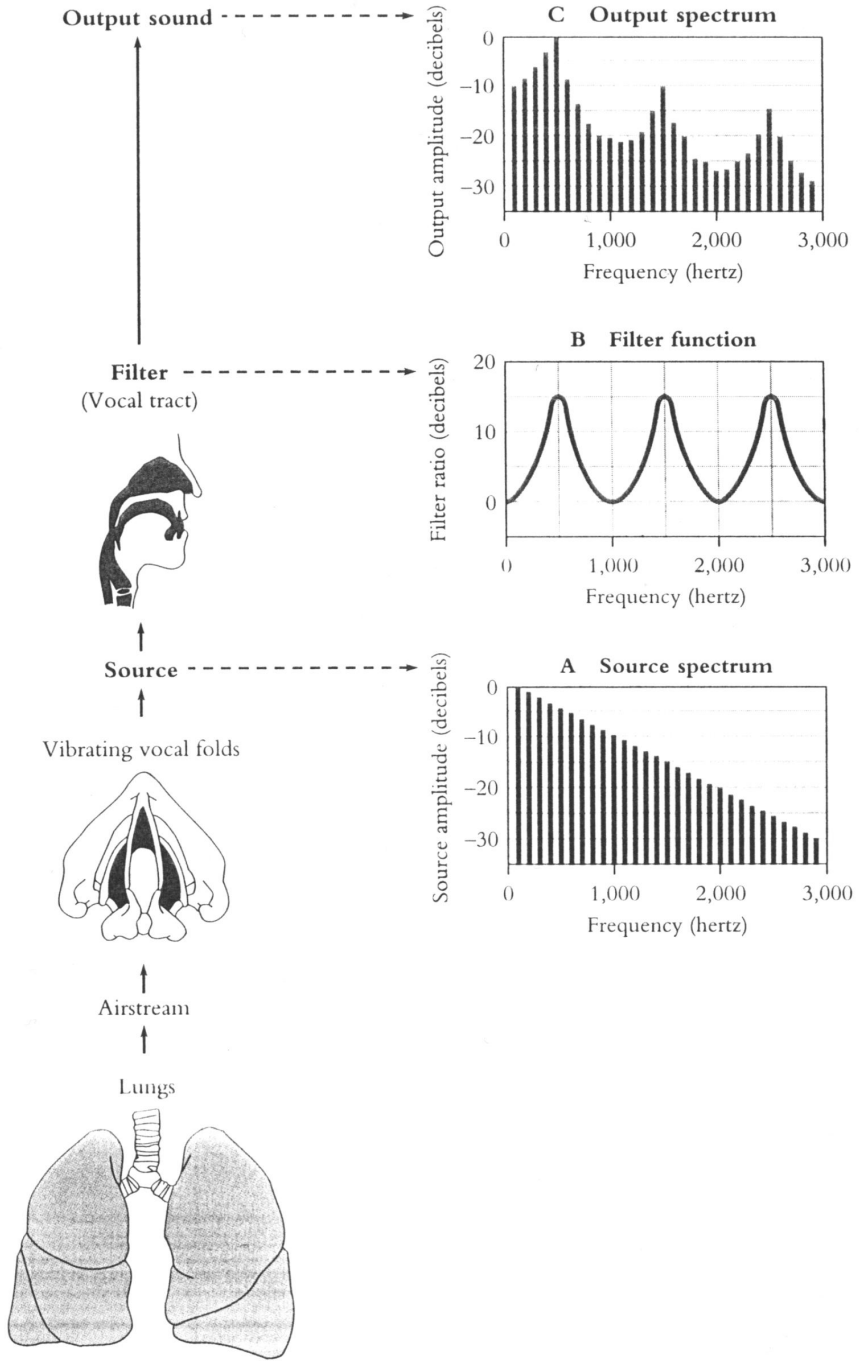




From: B.-H. Juang and L. Rabiner: Fundamentals of speech recognition ©1993 by Prentice-Hall



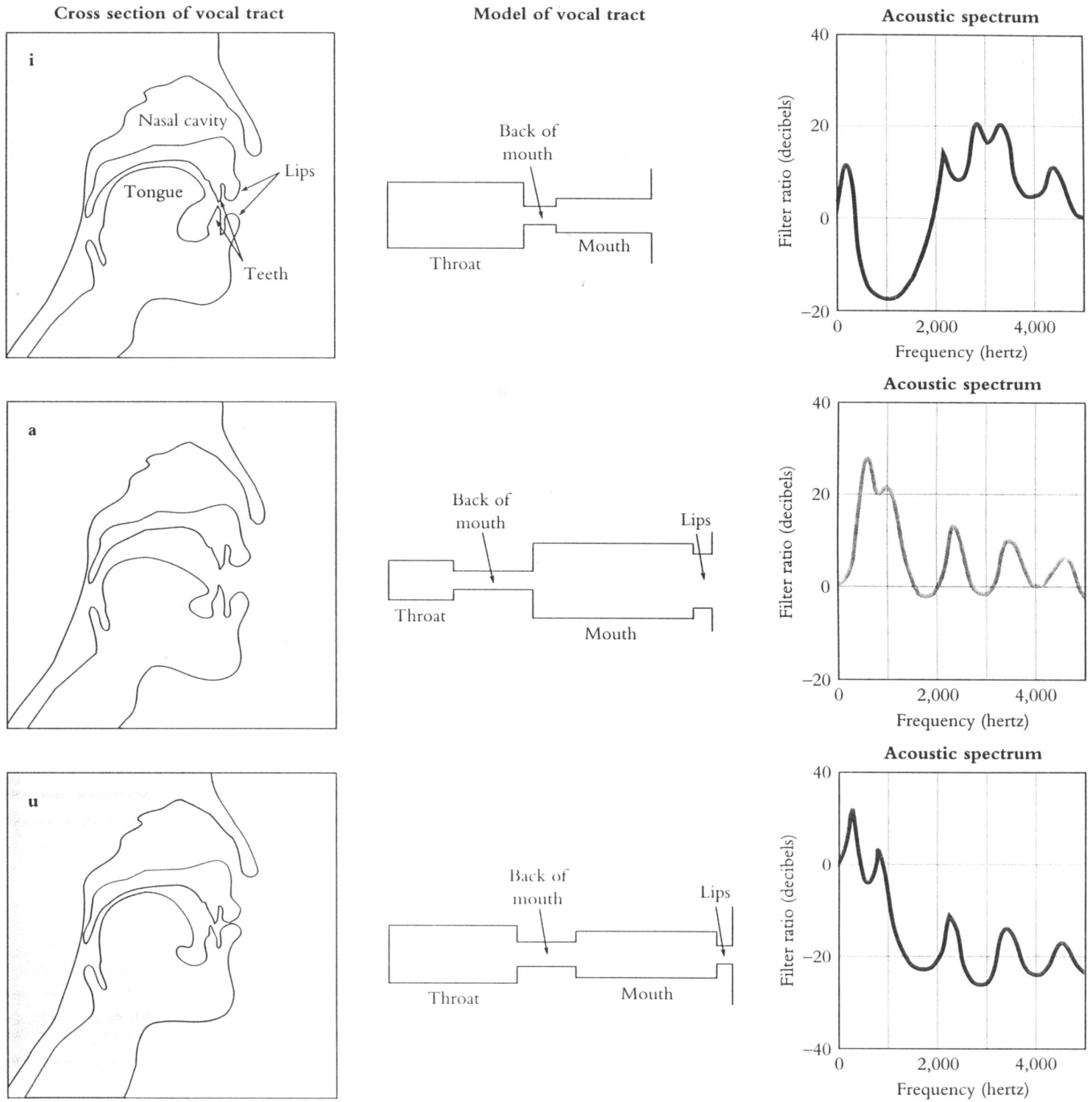
From: B.-H. Juang and L. Rabiner: Fundamentals of speech recognition ©1993 by Prentice-Hall



The lungs supply the power for speaking, the vocal folds act as an oscillator, and the vocal tract is the resonator. According to the source filter theory, the source is the laryngeal tone produced when the vibrating vocal chords interrupt the airstream from the lungs. The acoustic spectrum for a source with a fundamental frequency of 100 Hz (A) shows the amplitude decreasing uniformly with frequency. The vocal tract is a filter that modifies the source spectrum in a predictable way. For the vowel [Λ], for example, the filter function (B) has resonant peaks at 500, 1,500, and 2,500 Hz. When the source passes through this filter, the output spectrum (C) has peaks or formats, at those frequencies.

From: G. A. Miller: The Science of Words

©1996 Scientific American Library



From: G. A. Miller: The Science of Words ©1996 Scientific American Library

The feature extraction component analyzes the speech signal and generates a sequence of real-valued vectors.

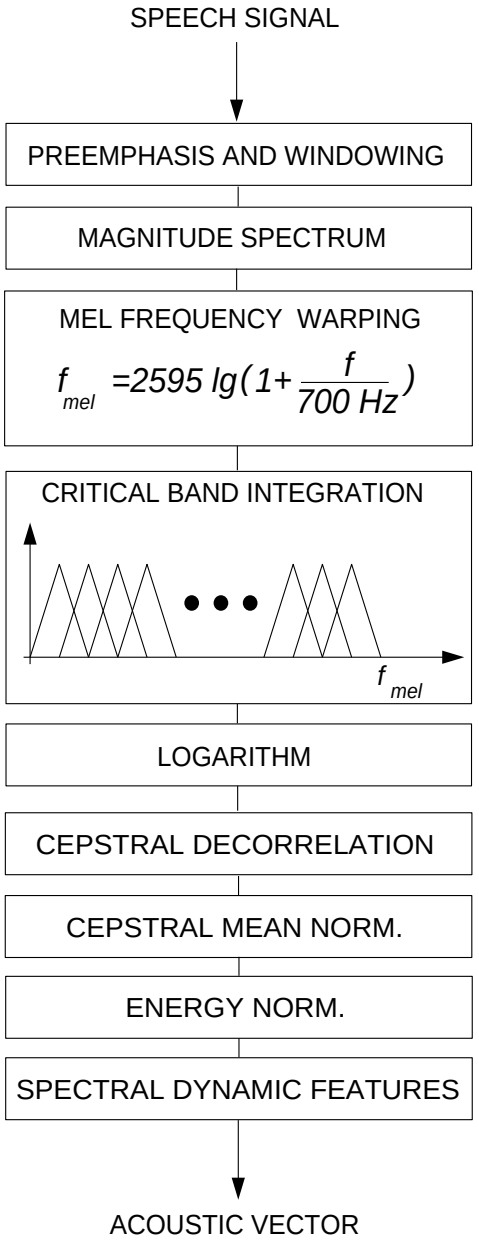
$$x_1^T := x_1, x_2, \ldots, x_T \qquad x_t \in \mathbb{R}^D$$

Goals:

- **similar sound $\Leftrightarrow$ similar vector**
- **suppress portions of the signal irrelevant or disturbing to recognition**
- **optional: adaptation to background noise and/or speaker characteristics**

Typically:

- **vector: 16–50 components**
- **one vector per 10 msec, i.e. 100 vectors per 1 sec of speech**



Starting point: digitized signal = sequence of samples

- Preemphasis:
  high-frequency components in the signal are emphasized (see Fourier analysis)
- Segmenting and Windowing:
  every 10ms-segment $t = 1, ..., T$, a 25ms-wide Hamming window is applied to the (preemphasized) speech samples $s_{nt}, n = 1, ..., N$:

$$x_{nt} = s_{nt} \cdot w_n = s_{nt} \cdot \left[0.54 - 0.46 \cos\left(\frac{2n\pi}{N-1}\right)\right]$$

- Magnitude spectrum: use DFT (discrete Fourier transform) or FFT (fast Fourier transform)

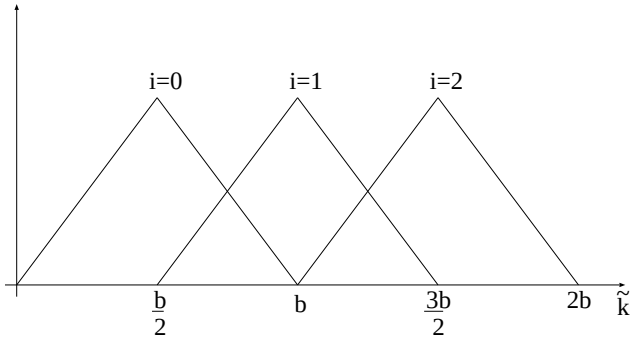
$$X_{kt} = \sum_{n=0}^{N-1} x_{nt} \cdot \exp\left(-j\frac{2\pi}{N} \cdot k \cdot n\right)$$

- **Mel frequency warping: convert frequency f_k to Mel frequency $\tilde{f}_k$ (with the sampling frequency F_S):**

$$\begin{aligned} f_k &= \frac{k}{N} \cdot F_S \\ \tilde{f}_k &= 2595 \cdot \log_{10}\left(1 + \frac{f_k}{700\,Hz}\right) \\ X_{kt} &\rightarrow X_{\tilde{k}t} \end{aligned}$$

- **Critical band integration: sum magnitude of Fourier components in each bandpass (= critical band) i:**

$$Y_{it} = \sum_{\tilde{k}} |X_{\tilde{k}t}| \cdot a_{\tilde{k}i}$$



with the triangular window $a_{\tilde{k}i}$

- **Logarithm: compute the logarithm of the filter bank outputs.**

- Cepstral decorrelation: apply discrete cosine transform to get cepstral coefficients c_{mt}

$$c_{mt} = \sum_{i=0}^{I-1} \cos\left(\frac{\pi \cdot m \cdot (i+0.5)}{I}\right) \cdot \log Y_{it}$$

- Cepstral mean normalization: subtract mean to eliminate unknown transfer function

$$c'_{mt} = c_{mt} - \overline{c_m} = c_{mt} - \frac{1}{T}\sum_{\tau=1}^{T} c_{m\tau}$$

- Energy Normalization: normalize the 0th cepstral coefficient (i.e. the energy)

$$c''_{mt} = \begin{cases} c'_{mt} - \max_{\tau} c'_{m\tau} & m = 0 \\ c'_{mt} & \text{otherwise} \end{cases}$$

- Typically the first 16 cepstral coefficients are used for recognition
- Spectral dynamic features: augment the feature vector c_t'' by it's first $\Delta c_t''$ and second derivatives $\Delta\Delta c_t''$.
- The resulting acoustic feature vector x_t at time frame t then is:

$$x_t := \begin{bmatrix} c_t'' \\ \Delta c_t'' \\ \Delta\Delta c_t'' \end{bmatrix}$$

Recognition is a comparison of acoustic feature vectors.

- It requires a similarity or distance measure
- and temporal synchronisation of corresponding vectors.

1.3 Distance Measures

To measure the dissimilarity of two acoustic vectors $x, y \in \mathbb{R}^D$, we will use a distance measure or metric $d(x, y)$.

A mapping d

$$\begin{aligned} d : \mathbb{R}^D \times \mathbb{R}^D &\rightarrow [0, \infty[\\ (x, y) &\rightarrow d(x, y) \end{aligned}$$

is called a metric (or distance measure) if for $x, y, z \in \mathbb{R}^D$:

a) $d(x, y) = 0$ if and only if $x = y$

b) symmetry: $d(x, y) = d(y, x)$

c) triangle inequality: $d(x, y) \leq d(x, z) + d(z, y)$

In the following, metrics are defined using a norm:

$$d(x,y) = ||x-y||$$

Here as in many other applications, we will use the so-called l_p norm ($p > 1$):

$$\begin{aligned} ||x-y||_p &= \left(\sum_{d=1}^{D} |x_d - y_d|^p \right)^{1/p} \\ &:= d_p(x,y) \end{aligned}$$

special cases which are often used:

- l_1 norm: city-block metric, Manhattan distance, chessboard distance

$$||x-y||_1 = \sum_{d=1}^{D} |x_d - y_d| = d_1(x,y)$$

- l_2 norm: Euclidean distance

$$||x-y||_2 = \sqrt{\sum_{d=1}^{D} (x_d - y_d)^2} = d_2(x,y)$$

  In Gaussian models (see later), the SQUARED Euclidean distance is used.

- l_∞ norm: Tschebyschev (Csebyshev) norm, maximum norm

$$||x-y||_\infty = \max_d |x_d - y_d| = d_\infty(x,y)$$

Depending on the circumstances, we want the distance measure to be invariant under specific transformations:

- shift (or translation) invariance:
  shift or translation:

$$x \rightarrow x' = x + a \qquad a \in \mathbb{R}^D$$

  This property is satisfied by ALL metrics defined by a norm because:

$$\begin{aligned} d(x', y') &:= ||x' - y'|| \\ &= ||(x + a) - (y + a)|| \\ &= ||x - y|| \\ &= d(x, y) \end{aligned}$$

  In other words: the distance does not depend on the choice of the origin of the coordinate system.

- rotation (including reflection) invariance:
  The Euclidean distance ($=l_2$ norm) is invariant under this transformation (see linear algebra).

Another type of invariance is concerned with transformations of the scaling of each vector component $d = 1, ..., D$:

$$x_d \rightarrow x'_d = c_d \cdot x_d \qquad c_d > 0$$

This type of invariance takes into account:

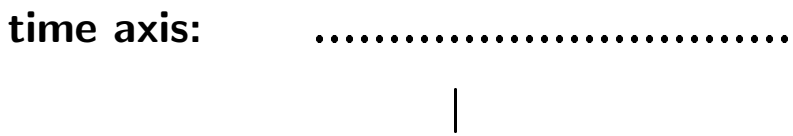
- the dimension of each vector component (=physical measurement),
- the statistical variation of each vector component x_d.

remark: in mathematical expressions, there are constraints on the physical dimensions such as:

- addition: $a + b$ for $a, b \in \mathbb{R}$:
  a and b must have the same physical units.
- exponentiation: e^a for $a \in \mathbb{R}$:
  a must be a pure number, i.e. WITHOUT any dimension.

To introduce scale invariance into the norms,
we take the statistical boundary conditions into account (see later for more details):

- **we isolate a single acoustic event as part of either a word or a sound (phoneme):**



- **we collect acoustic vectors for such an event (by extracting them from training utterances) and consider their statistical variations**

ultimate justification (see later):
statistical point of view

for a specific event, define the distance measure by extending the l_2 norm:

$$d_2(x,\mu) \quad = \quad \sum_{d=1}^{D} \left[\frac{x_d - \mu_d}{\sigma_d} \right]^2$$

with μ_d the empirical mean and σ_d the empirical variance of vector component x_d:

$$\mu_d = \frac{1}{N} \sum_{n=1}^{N} x_{nd} \quad , \quad \sigma_d^2 = \frac{1}{N} \sum_{n=1}^{N} [x_{nd} - \mu_d]^2$$

calculated from a set of training vectors $x_1 \ldots x_n \ldots x_N \in \mathbb{R}^D$

exercise: prove the scale invariance.

for a specific event, define the distance measure by extending the l_1 norm:

$$d_1(x, \mu) = \sum_{d=1}^{D} \left| \frac{x_d - \mu_d}{\sigma_d} \right|$$

with μ_d the empirical mean (median) and σ_d the empirical absolute deviation (sort of square root of the variance) of vector component x_d:

$$\mu_d = \frac{1}{N} \sum_{n=1}^{N} x_{nd} \quad , \quad \sigma_d = \frac{1}{N} \sum_{n=1}^{N} |x_{nd} - \mu_d|$$

calculated from a set of training vectors $x_1 \ldots x_n \ldots x_N \in \mathbb{R}^D$

exercise: prove the scale invariance.

A rather general type of distance measure is introduced as a quadratic form:

- definition: quadratic form for $y, z \in \mathbb{R}^D$:
  for a positive definite matrix $W \in \mathbb{R}^{D \times D}$ we define:

$$d_W(z, y) \quad := \quad (z - y)^T W (z - y)$$

  positive definite matrix W: $y^T W y \geq 0$ for $y \in \mathbb{R}^D$
  exercise: prove the properties of a metric.

- properties:
  by suitably selecting the matrix W, we can achieve:
  – any linear combination of features (=vector components)
  – suitable scaling and weighting of features

In the statistical context, a special type of quadratic form is important, which is referred to as covariance-weighted distance (or Mahalanobis distance). As before, we fix a specific acoustic event:

- training vectors (extracted from utterances):

$$x_1, ..., x_n, ..., x_N \quad x_n \in \mathbb{R}$$

- compute empirical covariance matrix:

$$\Sigma = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^T$$

  with the empirical mean vector (as before):

$$\mu = \frac{1}{N} \sum_{n=1}^{N} x_n$$

- use the inverse covariance matrix Σ^{-1} to define a quadratic form:

$$d_{\Sigma^{-1}}(x, \mu) = (x - \mu)^T \Sigma^{-1} (x - \mu)$$

The covariance-weighted distance measure is invariant
under arbitrary affine invertible transformations ($A \in \mathbb{R}^{D \cdot D}, b \in \mathbb{R}^D$):

$$x \quad \rightarrow \quad \tilde{x} = Ax + b$$

With this transformation, we have:

$$\begin{aligned}
\tilde{\mu} &= \frac{1}{N} \sum_{n=1}^{N} \tilde{x}_n = \frac{1}{N} \sum_{n=1}^{N} [Ax_n + b] \\
&= A\mu + b \\
\\
\tilde{\Sigma} &= \frac{1}{N} \sum_{n=1}^{N} (\tilde{x}_n - \tilde{\mu})(\tilde{x}_n - \tilde{\mu})^T \\
&= \ldots = A\Sigma A^T
\end{aligned}$$

property:

$$d_{\tilde{\Sigma}^{-1}}(\tilde{x}, \tilde{\mu}) \quad = \quad d_{\Sigma^{-1}}(x, \mu)$$

Covariance-Weighted Distance: Proof of Invariance Property

For the proof, we have to verify:

$$\begin{aligned} d_{\tilde{\Sigma}^{-1}}(\tilde{x},\tilde{\mu}) &= (\tilde{x}-\tilde{\mu})^T\tilde{\Sigma}^{-1}(\tilde{x}-\tilde{\mu}) \\ &= \ldots \\ &= (x-\mu)^T\Sigma^{-1}(x-\mu) \\ &= d_{\Sigma^{-1}}(x,\mu) \end{aligned}$$

exercise: work out the details.

remarks (for $x, y \in \mathbb{R}^D$):

- inner (or scalar) product:

$$x^T y \qquad \in \mathbb{R}$$

- l_2 norm and inner product:

$$||x||_2 = \sqrt{x^T x} \qquad \in \mathbb{R}$$

- outer product:

$$x y^T \qquad \in \mathbb{R}^{D \cdot D}$$

- hyperellipsoids described by a quadratic form with matrix W:

$$\{x \in \mathbb{R}^D : x^T W x = c\}$$

  for each constant $c > 0$.

This leads to the so-called Mahalanobis distance:

$$d_{MA}(z,y) = (z-y)^T K^{-1}(z-y)$$

properties:

- invariant with respect to any non singular linear transformation, scale invariant.

the transformation

$$\begin{aligned} y' &= K^{-1/2} y \\ z' &= K^{-1/2} z \end{aligned}$$

diagonalizes the covariance matrix:

$$d_{MA}(z',y') = \sum_{d=1}^{D} \left(\frac{y'_d - z'_d}{\lambda_d} \right)^2$$

$\lambda_d, d = 1, ..., D$ are the (positive) eigenvalues of K.

consider the distance measures for vectors x and y with $x_d, y_d > 0$:

- example A:

$$\begin{aligned} d(x,y) &= \sum_d \left| \log \frac{x_d}{y_d} \right| \\ &= \sum_d |\log x_d - \log y_d| \end{aligned}$$

- example B:

$$d(x,y) = \sum_d x_d \cdot \left| \log \frac{x_d}{y_d} \right|$$

- example C:

$$d(x,y) = \sum_d \left[\sqrt{x_d} - \sqrt{y_d} \right]^2$$

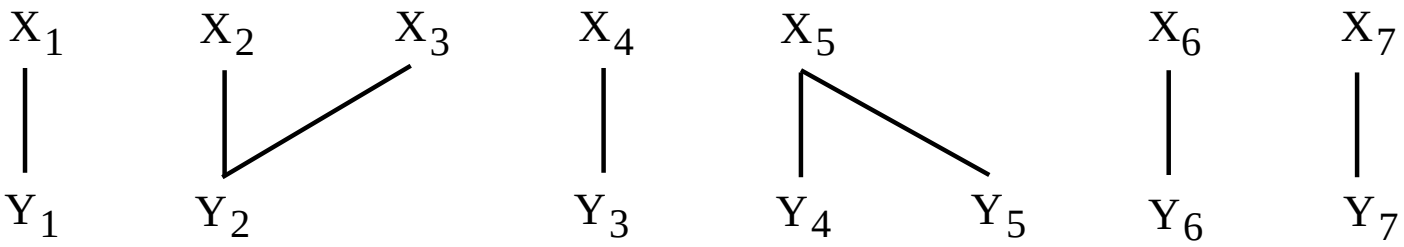
exercise:
check which properties of a metric are satisfied for each example.

1.4 Time Alignment

Given two sequences of acoustic vectors over time axis t and s,

$$\begin{aligned} \boldsymbol{X} &= (x_1 \ldots x_t \ldots x_T), \quad x_t \in \mathbb{R}^D \\ \boldsymbol{Y} &= (y_1 \ldots y_s \ldots y_S), \quad y_s \in \mathbb{R}^D \end{aligned}$$

we want to find a mapping of 'corresponding' vectors. This task is referred to as time alignment or time warping.



Possible methods for time alignment:

- linear time alignment:
  strech or compress one of the two time axes in a linear way, say s:

$$s = t \cdot \frac{S}{T}$$

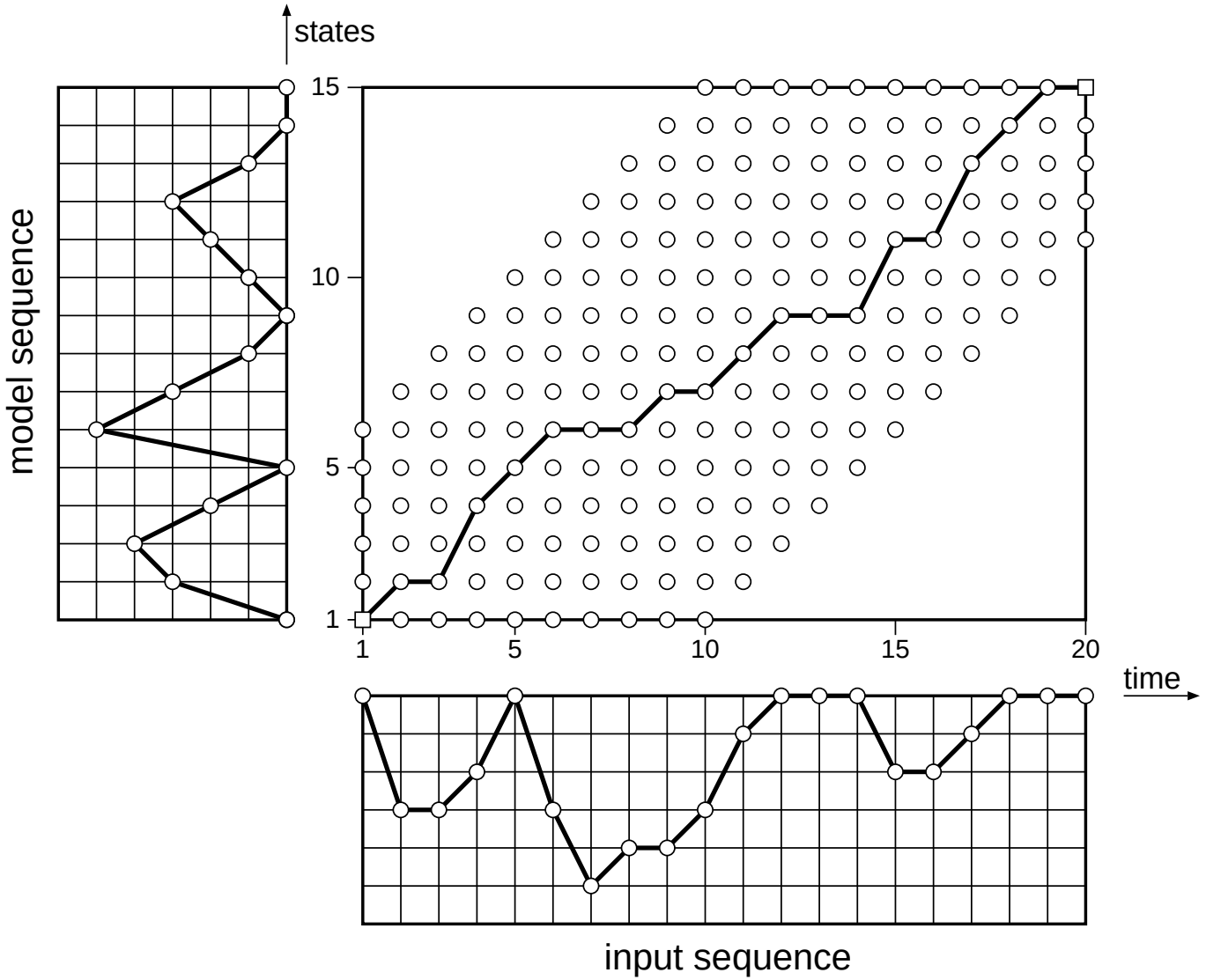
  experimental results: does not work well.

- nonlinear time alignment:
  introduce a nonlinear alignment function w:

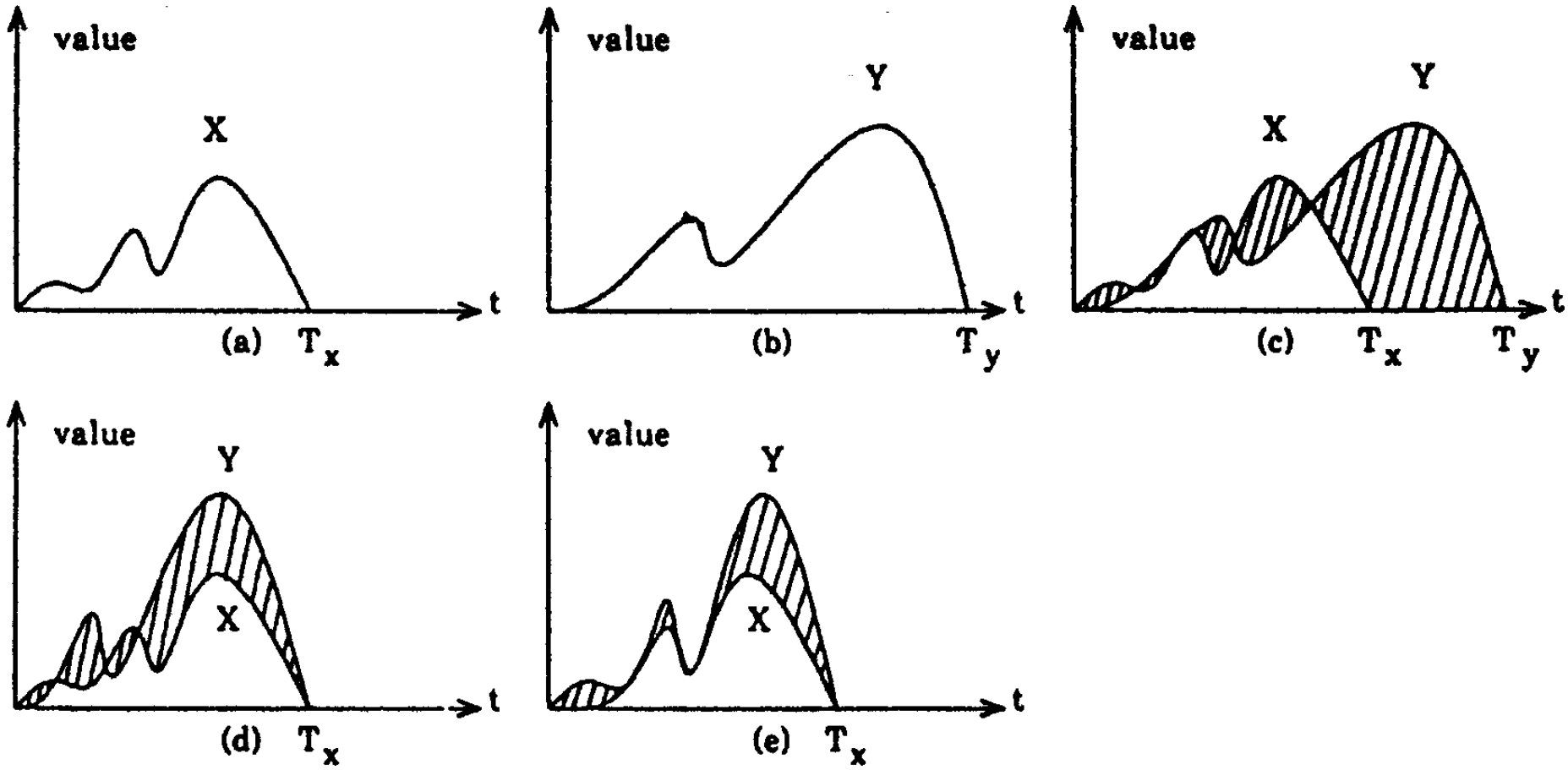
$$\begin{aligned} w &= w(1) \ldots w(l) \ldots w(L) \\ w(l) &= (t(l), s(l)), \quad l = 1, \ldots, L \end{aligned}$$

  The sequence of points $w(l)$, $l = 1 \ldots L$ defines a path in the (t, s)-plane. Each point (t, s) in the plane is referred to as grid point.

Example of a nonlinear time alignment (with time axis s and state axis s):



Examples: (a,b) two vector sequences X and Y, (c) comparison without alignment, (d) linear alignment, (e) nonlinear alignment



- **Time alignment as optimization problem:**

  Goal: determine the optimal path that minimizes the difference $D(X, Y)$ between the vector sequences X, Y:

$$D(X,Y) = \min_{\substack{l \to w(l) \\ L}} \left\{ \sum_{l=1}^{L} d(x_{t(l)}, y_{s(l)}) \right\}$$

  with:

$$w(l) = (t(l), s(l)), \quad l = 1, \ldots, L$$

  i.e.: find the optimum in the set of all allowed paths
  The set of allowed paths will be constrained in the following.

- **Properties of the path (symmetric version):**

  a) **Boundary conditions: start and end points are given:**

$$\begin{aligned} w(1) &= (1,1) \\ w(L) &= (T,S) \end{aligned}$$

  b) **Monotony: the temporal order is preserved:**

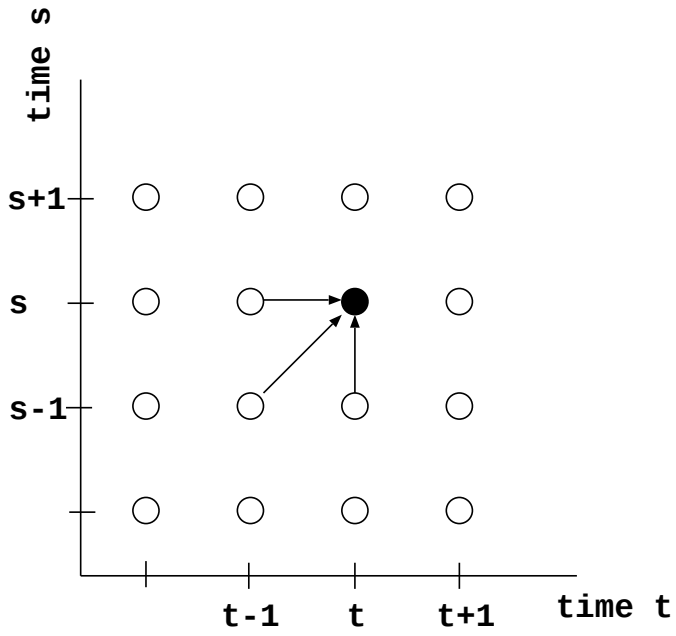
$$\begin{aligned} t(l) &\leq t(l+1) \\ s(l) &\leq s(l+1) \end{aligned}$$

  c) **Continuity: no vector is skipped:**

$$\begin{aligned} t(l) &\leq 1 + t(l-1) \\ s(l) &\leq 1 + s(l-1) \end{aligned}$$

The combination of the local path specifications b) and c) leads to $w(l) = (t, s)$:

$$w(l-1) \quad \in \quad \{(t-1, s), (t-1, s-1), (t, s-1)\}$$



Possible refinements of the time alignment:

- **limit the path slope by introducing a weighting factor**

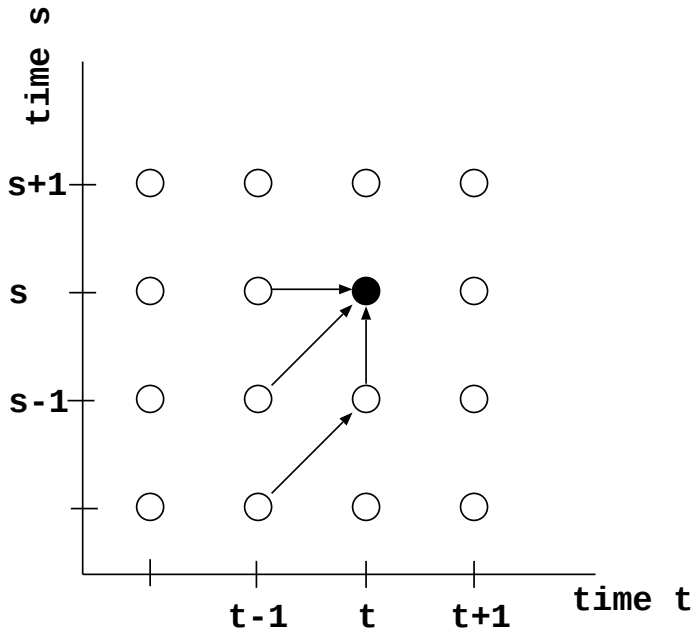
$$A(w(l-1), w(l))$$

to control the time distortion function $w(l)$.

The modified optimization criterion then is:

$$\min_{\substack{l \to w(l) \\ L}} \sum_{l=1}^{L} \left[d(x_{t(l)}, y_{s(l)}) \cdot A(w(l-1), w(l)) \right]$$

- **Nonsymmetric path specifications limiting the path slope smaller than 2.**



$$w(l-1) \quad \in \quad \{(t-1,s),(t-1,s-1),(t-1,s-2) \circ (t,s-1)\}$$

Later: discussion and choice of the refinements using a statistical approach.

Type	Allowable Path Specification
I	$\mathcal{P}_1 \rightarrow (1,0)$ $\mathcal{P}_2 \rightarrow (1,1)$ $\mathcal{P}_3 \rightarrow (0,1)$
II	$\mathcal{P}_1 \rightarrow (1,1)(1,0)$ $\mathcal{P}_2 \rightarrow (1,1)$ $\mathcal{P}_3 \rightarrow (1,1)(0,1)$
III	$\mathcal{P}_1 \rightarrow (2,1)$ $\mathcal{P}_2 \rightarrow (1,1)$ $\mathcal{P}_3 \rightarrow (1,2)$
IV	$\mathcal{P}_1 \rightarrow (1,1)(1,0)$ $\mathcal{P}_2 \rightarrow (1,2)(1,0)$ $\mathcal{P}_3 \rightarrow (1,1)$ $\mathcal{P}_4 \rightarrow (1,2)$
V	$\mathcal{P}_1 \rightarrow (1,1)(1,0)(1,0)$ $\mathcal{P}_2 \rightarrow (1,1)(1,0)$ $\mathcal{P}_3 \rightarrow (1,1)$ $\mathcal{P}_4 \rightarrow (1,1)(0,1)$ $\mathcal{P}_5 \rightarrow (1,1)(0,1)(0,1)$
VI	$\mathcal{P}_1 \rightarrow (1,1)(1,1)(1,0)$ $\mathcal{P}_2 \rightarrow (1,1)$ $\mathcal{P}_3 \rightarrow (1,1)(1,1)(0,1)$
VII	$\mathcal{P}_1 \rightarrow (1,1)(1,0)(1,0)$ $\mathcal{P}_2 \rightarrow (1,2)(1,0)(1,0)$ $\mathcal{P}_3 \rightarrow (1,3)(1,0)(1,0)$ $\mathcal{P}_4 \rightarrow (1,1)(1,0)$ $\mathcal{P}_5 \rightarrow (1,2)(1,0)$ $\mathcal{P}_6 \rightarrow (1,3)(1,0)$ $\mathcal{P}_7 \rightarrow (1,1)$ $\mathcal{P}_8 \rightarrow (1,2)$ $\mathcal{P}_9 \rightarrow (1,3)$
ITAKURA	$\mathcal{P}_1 \rightarrow (1,0)$, consecutive $(1,0)(1,0)$ disallowed $\mathcal{P}_2 \rightarrow (1,1)$ $\mathcal{P}_3 \rightarrow (1,2)$

1.5 Dynamic Programming

Time alignment is a nonlinear optimization problem, which will be solved using 'dynamic programming'.

General concept dynamic programming:

- **decompose problem into subproblems**
- **solve subproblems and store results in tables**
- **combine the results of the subproblems to produce the final result**

Richard Bellman 1957:
– dynamic := sequential over the time axis
– programming := optimization problem with constraints

For the symmetric time alignment case, the optimization criterion is:

$$\min_{\substack{l \to w(l) \\ L}} \left\{ \sum_{l=1}^{L} d(x_{t(l)}, y_{s(l)}) \right\}$$

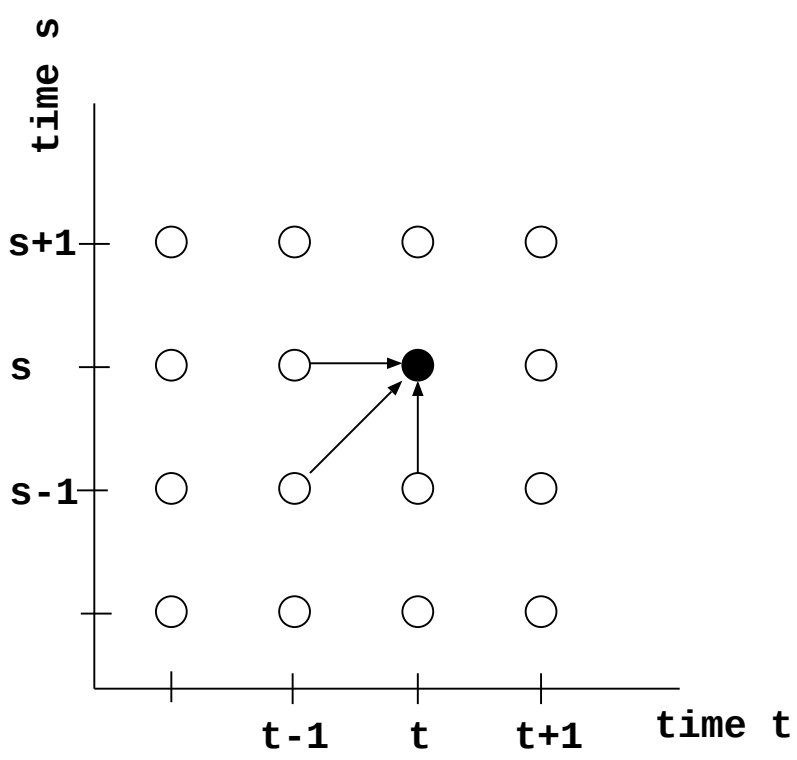
with the path

$$l \to w(l) = (t(l), s(l))$$

and the constraints:

- start: $(t(1), s(1)) = (1, 1)$
- end: $(t(L), s(L)) = (T, S)$
- predecessors of an inner grid point (t, s):

$$(t - \delta_t, s - \delta_s) \quad \text{with} \quad (\delta_t, \delta_s) \in \Delta = \{(1,0), (1,1), (0,1)\}$$



To apply the concept of dynamic programming,
we introduce the partial distance $D(t,s)$ as an auxiliary quantity:

$$D(t,s) := \min_{\substack{l \to (t(l),s(l)) \\ \lambda}} \left\{ \sum_{l=1}^{\lambda} d(x_{t(l)}, y_{s(l)}) : (t(\lambda), s(\lambda)) = (t,s) \right\}$$

$$= \text{minimum accumulated distance between gridpoints} (1,1) \text{ and } (t,s)$$

Decompose the time alignment path:

$$[(1,1) \rightarrow (t,s)] \quad = \quad [(1,1) \rightarrow (t-\delta_t, s-\delta_s)] \circ [(t-\delta_t, s-\delta_s) \rightarrow (t,s)]$$

For the auxiliary quantity, we then have:

$$\begin{aligned}
D(t,s) &= \min_{l \rightarrow (t(l),s(l))} \left\{ d(x_t, y_s) + \sum_{l=1}^{\lambda-1} d(x_{t(l)}, y_{s(l)}) : \right. \\
&\qquad \left. [(t,s) - (t(\lambda-1), s(\lambda-1))] \in \Delta \right\} \\
&= d(x_t, y_s) + \min_{l \rightarrow (t(l),s(l))} \left\{ \sum_{l=1}^{\lambda-1} d(x_{t(l)}, y_{s(l)}) : \right. \\
&\qquad \left. [(t,s) - (t(\lambda-1), s(\lambda-1))] \in \Delta \right\} \\
&= d(x_t, y_s) + \min_{(\delta_t, \delta_s) \in \Delta} \{ D(t-\delta_t, s-\delta_s) \} \\
&= d(x_t, y_s) + \min \{ D(t-1,s), D(t-1,s-1), D(t,s-1) \}
\end{aligned}$$

The dynamic programming recursion for $D(t,s)$ then is:

$$D(t,s) \quad = \quad d(x_t, y_s) + \min\{D(t-1,s), D(t-1,s-1), D(t,s-1)\}$$

with the boundary conditions:

$$\begin{aligned} D(1,1) &= d(x_1, y_1) \\ D(t,s) &= \infty \qquad \text{for} \quad (t,s) \neq (1,1) \end{aligned}$$

and the boundary grid points:

$$\begin{aligned} D(1,s) &= \sum_{\sigma=1}^{s} d(x_1, y_\sigma) \\ D(t,1) &= \sum_{\tau=1}^{t} d(x_\tau, y_1) \end{aligned}$$

Solving the recursion:

- recursive: naive implementation $\rightarrow$ NOT EFFICIENT
- recursive: memoization, storing intermediate results
- iterative: using table and steering the process $\rightarrow$ EFFICIENT

ARRAY D(1:T, 1:S)
for $t = 1, .., T$
  for $s = 1, .., S$

$$D(t,s) = d(x_t, y_s) + \min\{D(t-1,s), D(t-1,s-1), D(t,s-1)\}$$

Memory requirement $(T \cdot S)$, complexity $(T \cdot S)$ operations

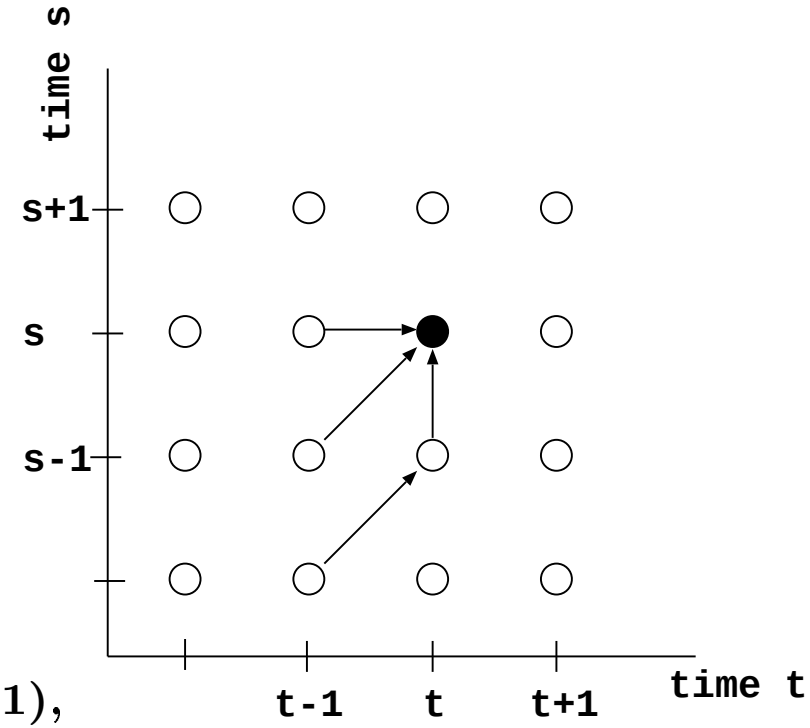
Use backpointer $B(t,s)$ to keep track of the best path:

$$B(t,s) = \underset{(\delta_t,\delta_s)\in\Delta}{\operatorname{argmin}} \{D(t-\delta_t, s-\delta_s)\}$$

non-symmetric case with path slope smaller than 2:

- **recursion:**

$$\begin{aligned} D(t,s) &= d(x_t, y_s) + \\ &\quad \min\{D(t-1,s), D(t-1,s-1), \\ &\quad D(t-1,s-2) + d(x_t, y_{s-1})\} \end{aligned}$$



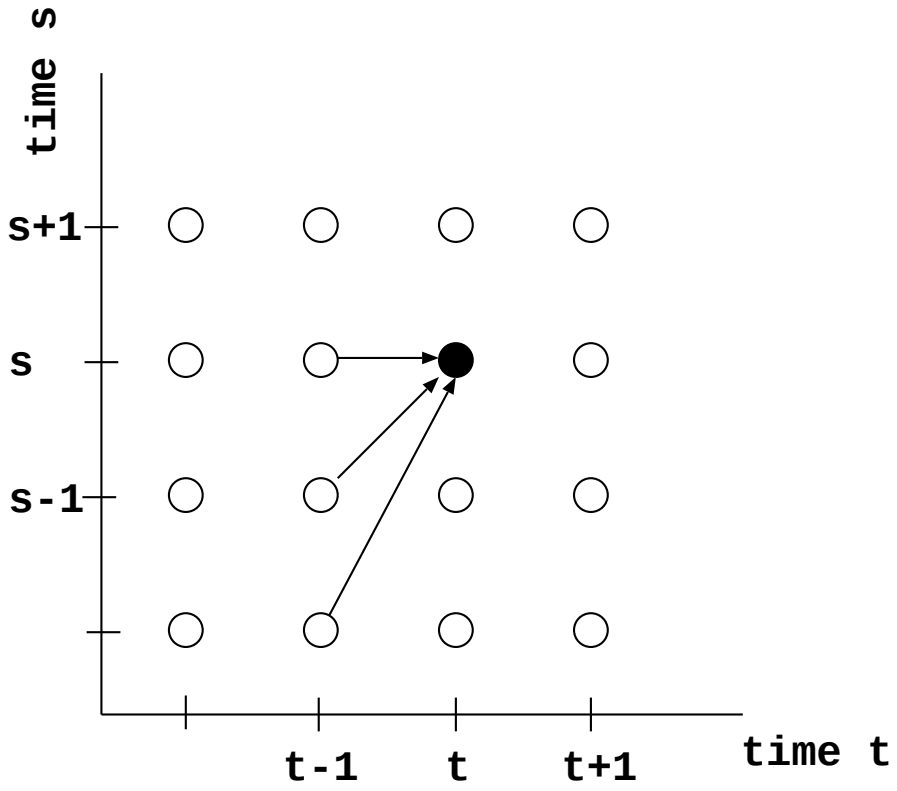
- **model properties:**

  according to the definition it is sufficient to define:

$$t \rightarrow s(t)$$

$$\begin{aligned} B(t,s) &= \operatorname{argmin}\{D(t-1,s), D(t-1,s-1), \\ &\quad D(t-1,s-2) + d(x_t, y_{s-1})\} \end{aligned}$$

(0,1,2)-standard model



(0,1,2)-standard model

the path is:

$$t \quad \rightarrow \quad s(t).$$

the optimization criterion is:

$$\min_{t \rightarrow s(t)} \sum_{t=1}^{T} \left[d(x_t, y_{s(t)}) + \mathcal{T}(s(t) - s(t-1)) \right]$$

with: the time distortion penalty $\mathcal{T}(s(t) - s(t-1))$ for the transition

$$(t-1, s(t-1)) \quad \rightarrow \quad (t, s(t))$$

each acoustic vector in X is only considered once

(0,1,2)-standard model

using dynamic programming we get:

$$
\begin{aligned}
D(t,s) &= \min_{\{\tau \to s(\tau): s(t)=s\}} \sum_{\tau=1}^{t} \left[d(x_\tau, y_{s(\tau)}) + \mathcal{T}(s(\tau) - s(\tau-1)) \right] \\
&= \min_{\{\tau \to s(\tau): s(t)=s\}} \left[d(x_t, y_s) + \sum_{\tau=1}^{t-1} \left[d(x_\tau, y_{s(\tau)}) + \mathcal{T}(s(\tau) - s(\tau-1)) \right] + \right. \\
&\qquad \left. \mathcal{T}(s(t) - s(t-1)) \right] \\
&= d(x_t, y_s) + \min_{\{\tau \to s(\tau): s(t)=s\}} \left[\sum_{\tau=1}^{t-1} \left[d(x_\tau, y_{s(\tau)}) + \mathcal{T}(s(\tau) - s(\tau-1)) \right] + \right. \\
&\qquad \left. \mathcal{T}(s(t) - s(t-1)) \right] \\
&= d(x_t, y_s) + \min_{\delta \in \{0,1,2\}} \{ D(t-1, s-\delta) + \mathcal{T}(\delta) \}
\end{aligned}
$$

(0,1,2)-standard model

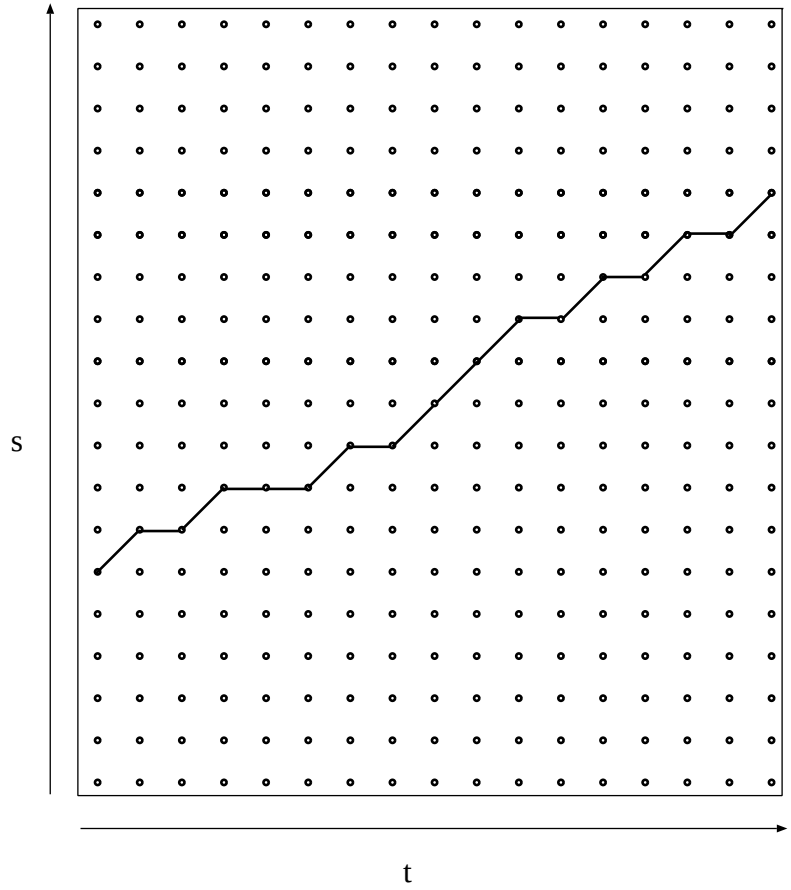
- **recursion:**

$$\begin{aligned} D(t,s) &= \min_{\{\tau \to s(\tau): s(t)=s\}} \sum_{\tau=1}^{t} \left[d(x_\tau, y_{s(\tau)}) + \mathcal{T}(s(\tau) - s(\tau-1)) \right] \\ &= d(x_t, y_s) + \min_{\delta \in \{0,1,2\}} \{ D(t-1, s-\delta) + \mathcal{T}(\delta) \} \end{aligned}$$

- **backpointer:**

$$B(t,s) = \operatorname*{argmin}_{\delta \in \{0,1,2\}} \{ D(t-1, s-\delta) + \mathcal{T}(\delta) \}$$

General case with open starting and ending point.



- **starting point:** $(1, s_{beg}), \quad 1 \leq s_{beg} \leq S$
- **ending point:** $(T, s_{end}), \quad s_{beg} \leq s_{end} \leq S$

Using dynamic programming we get:

- recursion:

$$D(t,s) \quad = \quad d(x_t, y_s) + \min_{\delta \in \{0,1,2\}} \{D(t-1, s-\delta) + \mathcal{T}(\delta)\}$$

  the ending point (T, s_{end}) is found using the optimization:

$$s_{end} = \operatorname*{argmin}_{s=1\ldots S} D(T,s)$$

- Starting point:
  beginning with $B(T, s_{end})$ the path can be traced back to the starting point

Other example:

- Approximative string searching (Levenshtein distance, edit distance)
  motivation:
  - word error rate evaluation
  - typing error tolerant string matching
  - DNA sequencing

  edit distance (Levenshtein distance):
  - deletions
  - insertions
  - substitutions

  boundary conditions:
  - no overlapping
  - monotony of the symbol sequences

Example: A = teacher; B = speaker

alignment:
number of errors: 5

t	e	a	c	h	e	r
|1	|1	|1	|1	|1	|	|
s	p	e	a	k	e	r

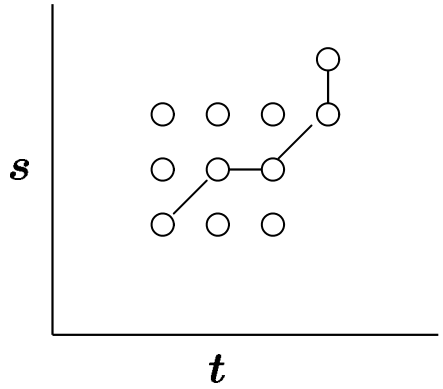
better alignment:
number of errors: 4

t	e	a	c	h	e	r
1	\ 1	\	\ 1	|1	|	|
s	p	e	a	k	e	r

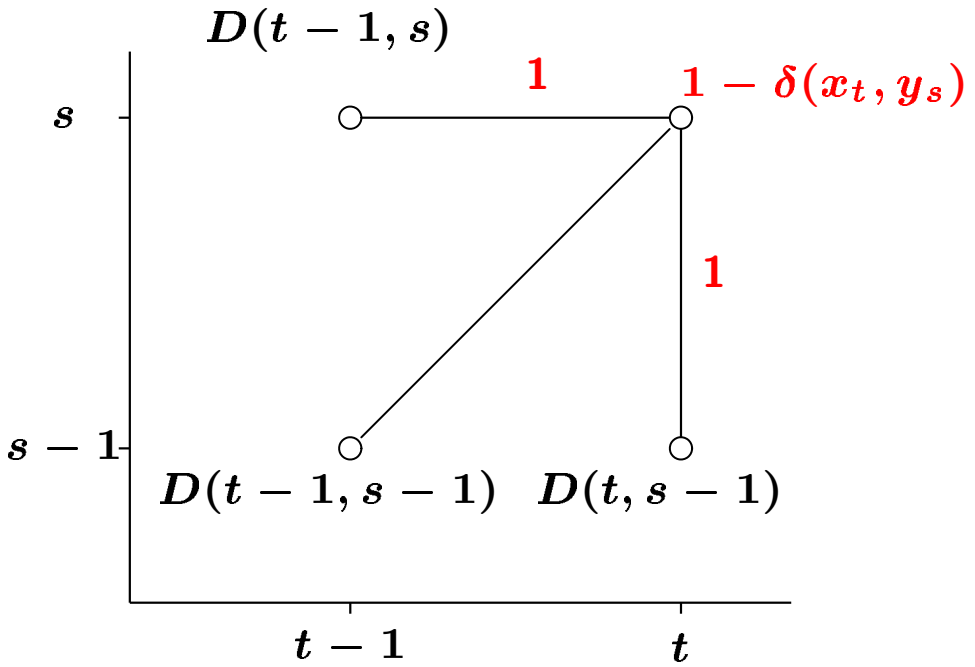
Problem: Determine the alignment with minimal costs.
Deletions, insertions und substitutions shall have equal costs:

$$\begin{aligned} \text{cost} &= \text{number of edit operations} \\ &= \text{deletions} + \text{insertions} + \text{substitutions} \end{aligned}$$

An alignment is a path between $x_1 \ldots x_t \ldots x_T$ and $y_1 \ldots y_s \ldots y_S$.



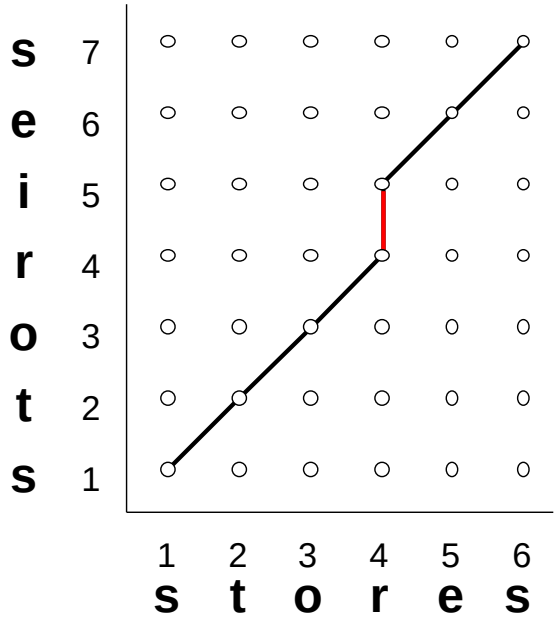
Define auxiliary quantity $D(t, s)$:



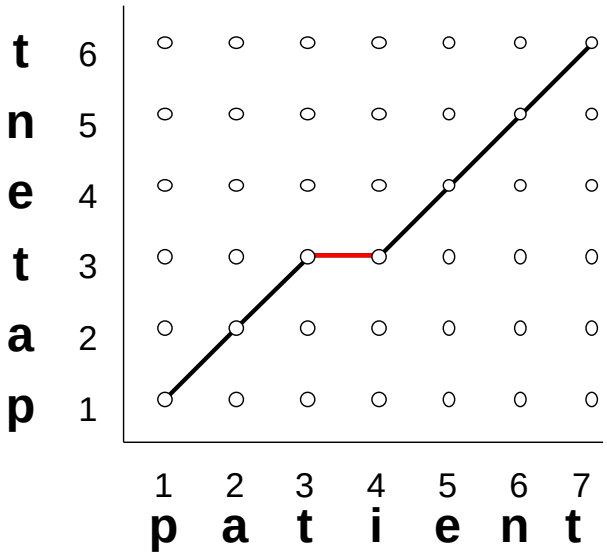
$$D(t,s) = \min\{D(t-1,s)+1,\ D(t,s-1)+1,\ D(t-1,s-1)+(1-\delta(x_t,y_s))\}$$

Examples:

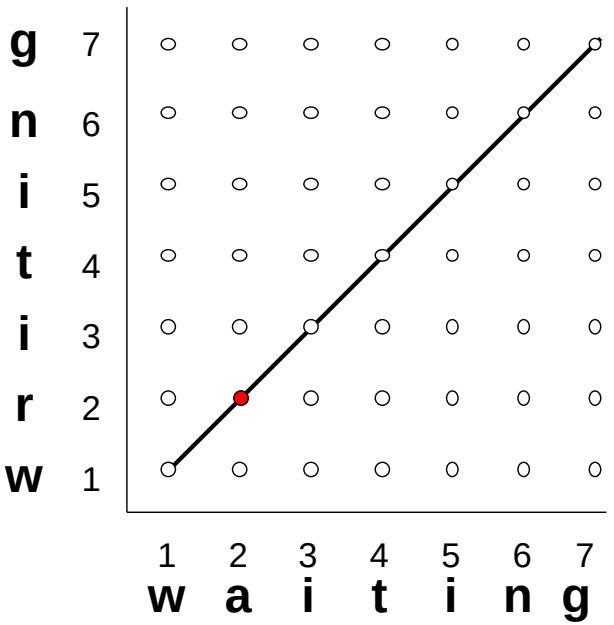
stories - stores (del)



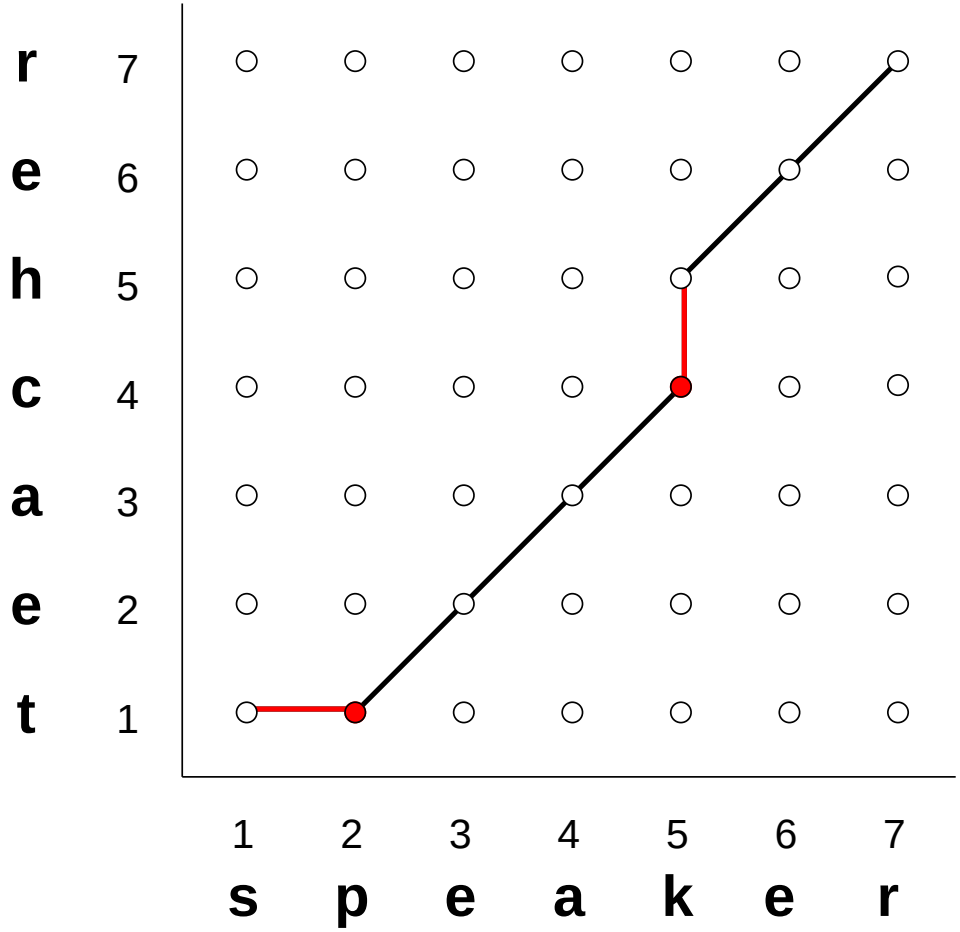
patent - patient (ins)



writing - waiting (sub)



Example: teacher - speaker (1 del + 1 ins + 2 sub)



- **recursion:**

$$\begin{aligned} D(t,s) &= \min\{D(t-1,s)+1, D(t,s-1)+1, \\ &\qquad D(t-1,s-1)+(1-\delta(x_t,y_s))\} \\ &= 1+\min\{D(t-1,s-1)-\delta(x_t,y_s), \\ &\qquad D(t-1,s), D(t,s-1)\} \end{aligned}$$

for $0 < t \leq T, \quad 0 < s \leq S$ with $\delta(x,y) = \begin{cases} 0 & x \neq y \\ 1 & x = y \end{cases}$

initialization: $D(0,0) = 0, \quad D(0,s) = s$ with $s \in [1,S], \quad D(t,0) = t$ with $t \in [1,T]$

resulting complexity:

- time / memory: $(T \cdot S)$ grid points
- without explicitly calculating the alignment path:
  memory: $(\min(T,S))$

Other application examples in computer science:

- **triangulation of polygones**
- **traveling salesman problem**
- **CYK parsing**
- **knapsack problem**
- **bracketing for matrix multiplications**

- **general concept of dynamic programming:**

  main requirement:
  the optimization problem can be split into subproblems:

  approach:

  - **solve subproblems,**
  - **store the solutions to the subproblems,**
  - **and put them together to produce the overall solution.**

  Terminology: (Richard Bellman 1957)
  dynamic := sequential over the time axis
  programming := optimization problem with constraints

  Better name: "sequential optimization"

The following picture shows a system with T steps.
u is an element out of a finite set of discrete decisions.
At every step $t = 1, \ldots, T$ a decision $u(t)$ with a cost is to be taken.
The costs depend on the decisions $u(t)$, the previous decisions $u(t-1)$ and the time t.

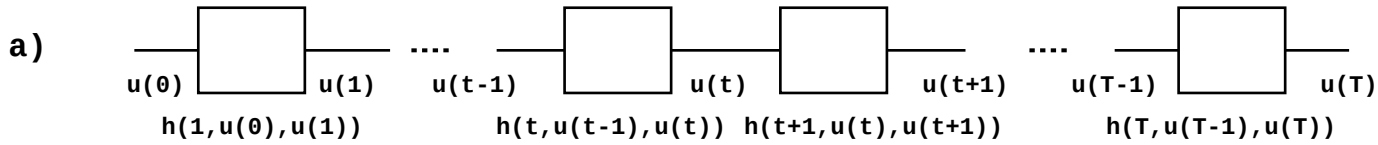
notation for local cost:

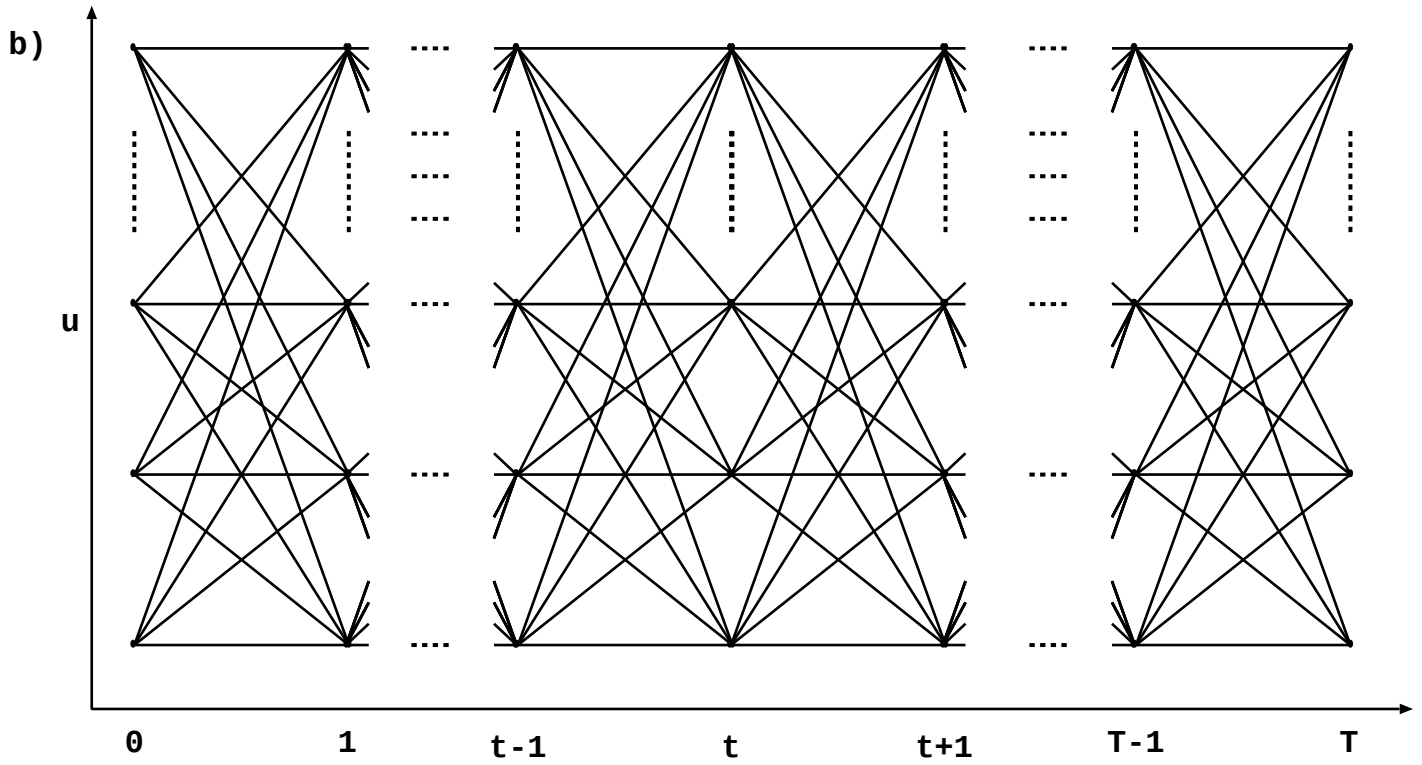
$$h(t, u(t-1), u(t))$$

The sum over all steps $t = 1, \ldots, T$ is the total cost of the decision sequence $u(1), \ldots, u(t), \ldots, u(T)$:

$$\min_{t \to u(t)} \sum_{t=1}^{T} h(t, u(t-1), u(t))$$

General example: system with T steps, costs at each step depend on preceding decisions





We solve the problem by dynamic programming:

- define auxiliary quantity $H(t,u)$:

$$\begin{aligned} H(t,u) &:= \min_{\tau \to u(\tau): u(t)=u} \left\{ \sum_{\tau=1}^{t} h(\tau, u(\tau-1), u(\tau)) \right\} \\ &= \text{minimum costs of partial decision sequence ending in decision } u \text{ at time } t \end{aligned}$$

- recursion:

$$H(t,u) = \min_{u'} \{ H(t-1,u') + h(t,u',u) \}$$

- backpointer:

$$B(t,u) = \operatorname*{argmin}_{u'} \{ H(t-1,u') + h(t,u',u) \}$$

optimal endpoint of the path:

$$\operatorname*{argmin}_{u} H(T,u)$$

remarks:

- **complexity:**

  time: $|\{u\}|^2 \cdot T$

  memory: $2 \cdot |\{u\}| \cdot T$

- **Due to the dependence on the immediately preceding decision, this model is referred to as "first-order-model".**

- **By dynamic programming, the lattice (or trellis) is converted into a tree whose structure is defined by the backpointers.**

How to apply dynamic programming:

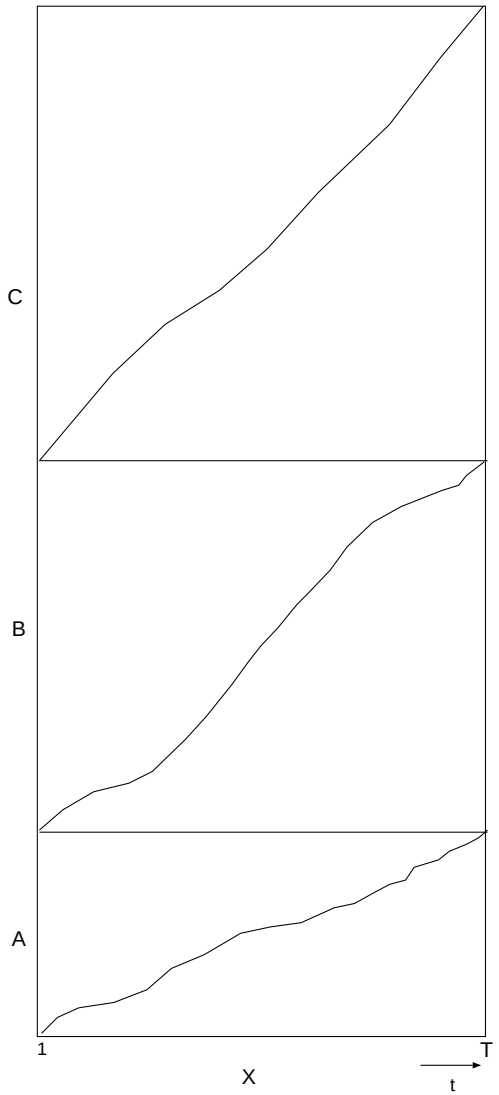
- **Represent problem as network.**

- **Edges in the graph have costs, that do not depend on other edges.**

- **Thus the problem can be divided into subproblems and solved with dynamic programming.**

1.6 Isolated Word Recognition

- **Recognition**
  - **So far: one comparison between two sequences of acoustic vectors.**
  - **Recognition requires comparison of a test pattern with all reference patterns in the vocabulary.**



Definitions:

- **Test pattern: sequence of acoustic feature vectors**

$$\boldsymbol{X} \quad = \quad (x_1 \ldots x_t \ldots x_T), \quad x_t \in \mathbb{R}^D$$

- **Reference pattern:**
  - **each word w consists of $s = 1, \ldots, S(w)$ states**
    these form a normalized time basis
  - **every state s has reference or model parameters:**
    - **prototype or mean vector $\mu_{sw} \in \mathbb{R}^D$**
    - **variance vector $\sigma_{sw}^2 \in \mathbb{R}^D$**
      special choice pooled variance: $\sigma_{sw}^2 = \sigma \in \mathbb{R}^D \quad \forall\, s$

- The distance measure depends on the state s and word w

  e.g. the squared Euclidean distance with variances:

$$d(x_t; s, w) = \sum_{d=1}^{D} \left(\frac{x_{td} - \mu_{swd}}{\sigma_{swd}} \right)^2$$

The recognition consists of two steps:

- Time alignment between test pattern X and every word $w = 1, \ldots, W$ in the vocabulary.
- Choose recognized word w_{opt}:

$$w_{opt} = \underset{w=1,\ldots,W}{\operatorname{argmin}} D(T, S(w), w)$$

For every word w and all gridpoints (t, s) the recursion

$$D(t, s, w) = d(x_t; s, w) + \min_{\delta \in \{0,1,2\}} [D(t-1, s-\delta, w) + \mathcal{T}(\delta)]$$

has to be evaluated.

In the following we use the (0,1,2) standard model that considers each acoustic vector x_t once.

Implementation example in C–code: show /alignment.pdf

- **Training:**
  in training these model parameters have to be estimated:
  - **number of states $S(w)$**
  - **prototype vector μ_{sw}**
  - **variance vector σ^2_{sw}**

- Number of states $S(w)$:
  Use start–stop–detection to determine average
  length of a word $\Rightarrow S(w)$

- Prototype vector μ_{sw}
- Variance vector σ^2_{sw}

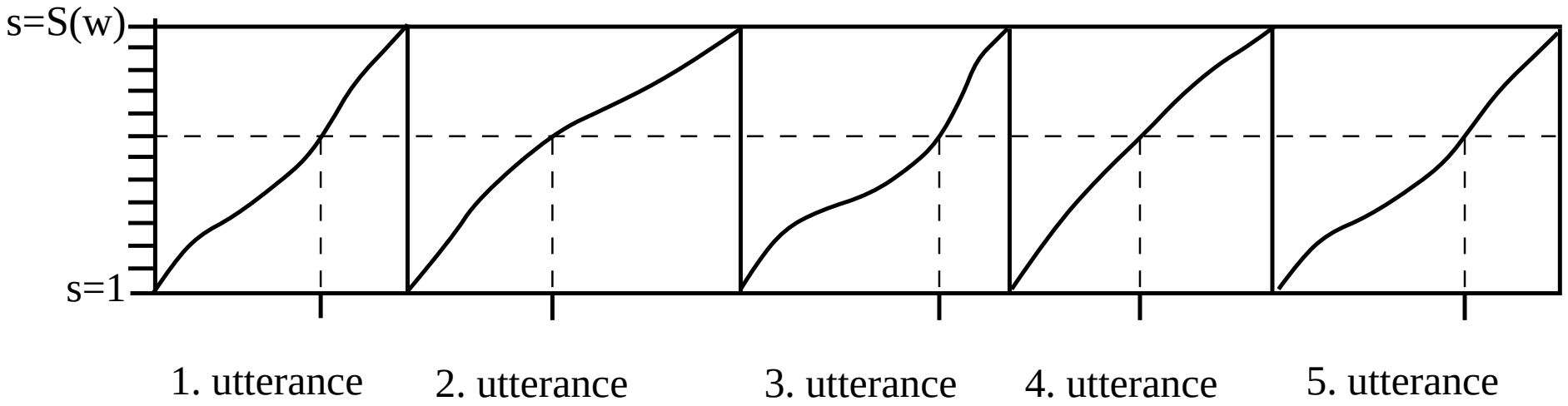
  Iterative training in two steps:

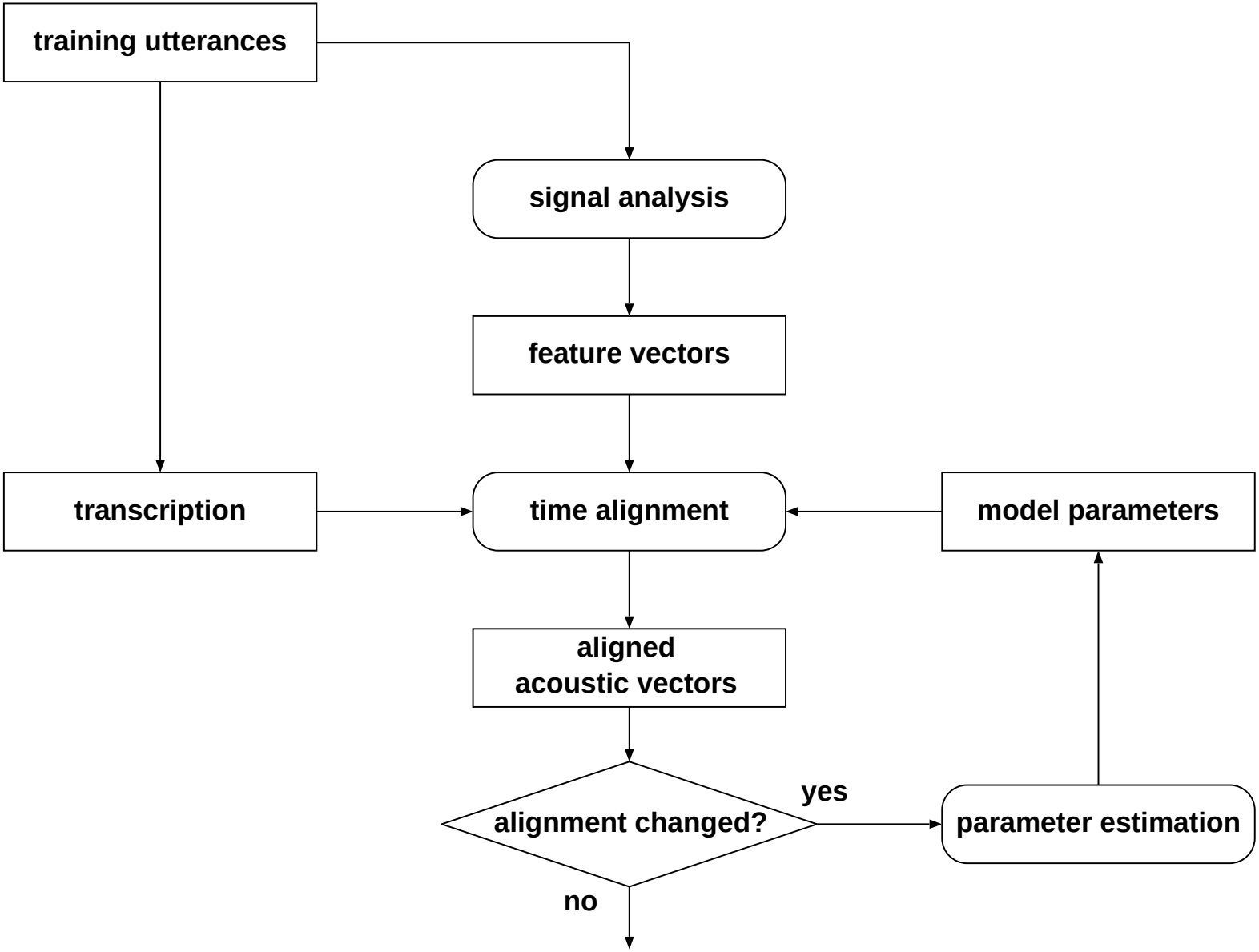
  1. time alignment
  2. parameter estimation
     - collect all observations belonging to s along the path
     - calculate model parameters μ_{sw} and σ^2_{sw} for s

  Note: in the first iteration no model parameters are available
  linear segmentation is used instead of time alignment

Time alignment in training.

All observations for the states s of word w are collected.




training utterances
signal analysis
feature vectors
transcription
time alignment
model parameters
aligned
acoustic vectors
alignment changed?
yes
no
parameter estimation

For optimal performance of the resulting recognizer, the training data should

- capture the variability of the signal as much as possible,
- reflect the application conditions as much as possible.

Therefore:

- collect training data under real world conditions.
- distribute data collection over several weeks or months.
- speaker independence: use many different speakers (> 100)
  cover different ages, genders, dialects, ...
- try to speak "normally": avoid both overarticulation and monotone way of speaking.
- use the same recording conditions (type of microphone,...) as in the test phase
  for telephone: use many different dialed-up lines.
- the collected training data has to be transcribed carefully.

do not use training data for recognition tests.

Preprocessing steps:

a) Long term normalization for spectral vectors x_{td}:
Goal: eliminate variabilities in the acoustic vectors, which can be caused by differences in the transmission channels (telephone, microphone, ...).

In the linear approximation the influence of the acoustic–electric transmission channel can be expressed as an (unknown) transmission function H_d:

$$x'_{td} = H_d \cdot x_{td}$$

with $d = 1, \ldots, D \quad \hat{=}$ Filter bank channels,
$x_{td} \quad \hat{=}$ spectral vector

Define new acoustic vectors y_{td}:

$$y_{td} := \frac{x'_{td}}{< x'_{td} >_t}$$

where $<>_t$ is the temporal mean over the utterance.

y_{td} is independent from H_d and only depends on x_{td}:

$$\begin{aligned} y_{td} &= \frac{x'_{td}}{< x'_{td} >_t} \\ &= \frac{H_d \cdot x_{td}}{H_d \cdot < x_{td} >_t} \\ &= \frac{x_{td}}{< x_{td} >_t} \end{aligned}$$

where H_d is considered independent from t.

The derivation is correct for absolute values as well as squared values.

In the logarithmic domain correspondingly:

$$y_{td} = x'_{td} - < x'_{td} >_t$$

b) **Energy normalization (global):**

$E(t)$: **logarithmic energy**

$$E_{NORM}(t) := E(t) - < E(t) >_t$$

Instead of the mean the maximum can also be used (especially if the amount of silence in the utterance is large):

$$E_{NORM}(t) := E(t) - \max_t(E(t))$$

2 Statistical Interpretation and Models

2.1 Single Densities 110

2.2 Mixture Densities 116

2.3 Hidden Markov Models 121

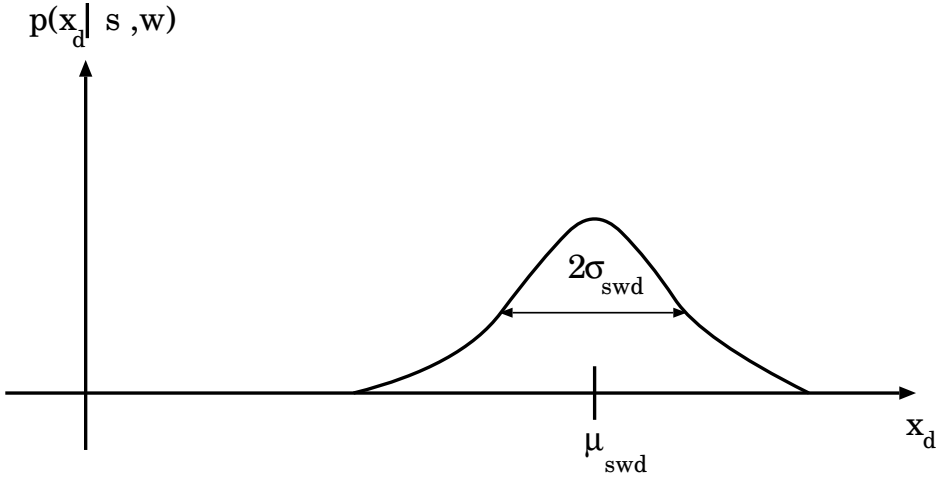
2.4 Bayes Decision Rule 126

2.1 Single Densities

- Gaussian distributions
  - The observations belonging to a state s of word w vary statistically
  - Gaussian distributions can be used to model these variations

$$p(x_d|s,w) = \frac{1}{\sqrt{2\,\pi\,\sigma_{swd}^2}}\, e^{-\frac{1}{2}\left(\frac{x_d - \mu_{swd}}{\sigma_{swd}}\right)^2}$$

x_d component of the feature vector, μ_{swd} mean, σ_{swd}^2 variance



- Multiply all components $d = 1, \ldots, D$ to get overall distribution for the vector $x = [x_1, \ldots, x_d, \ldots, x_D]$

$$p(x|s,w) = \prod_{d=1}^{D} p(x_d|s,w)$$

- When assuming statistical independence of the components

$$\begin{aligned} p(x|s,w) &= \prod_{d=1}^{D} p(x_d|s,w) \\ &= \frac{1}{\prod_{d=1}^{D} \sqrt{2\,\pi\,\sigma_{swd}^2}} \; e^{-\frac{1}{2} \sum_{d=1}^{D} \left(\frac{x_d - \mu_{swd}}{\sigma_{swd}} \right)^2} \end{aligned}$$

- The negative logarithm of $p(x|s,w)$ can be interpreted as distance

$$d(x;s,w) := -\log p(x|s,w)$$
$$= \underbrace{\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - \mu_{swd}}{\sigma_{swd}}\right)^2}_{\text{distance}} + \underbrace{\frac{1}{2}\sum_{d=1}^{D}\log\left(2\,\pi\,\sigma_{swd}^2\right)}_{\text{normalizing factor}}$$

- Often the variances are pooled over s and w

$$\sigma_{swd} = \sigma_d = \text{const}(s,w)$$

σ_d only depends on the vector component and is the same for all sates and words. This results in

$$d(x;s,w) = \frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - \mu_{swd}}{\sigma_d}\right)^2 + \text{const}(s,w)$$

- General Gaussian model
  no independence of the components is assumed

$$p(x|s,w) = \frac{1}{\sqrt{\det\,(2\,\pi\,\Sigma_{sw})}}\; e^{-\frac{1}{2}\Big((x-\mu_{sw})^T \Sigma_{sw}^{-1}(x-\mu_{sw})\Big)}$$

  a diagonal covariance matrix Σ_{sw} leads to the previous case

- Mahalanobis distance
  the negative logarithm of the general Gaussian model

$$\begin{aligned} d(x;s,w) &:= -\log p(x|s,w) \\ &= \underbrace{\frac{1}{2}\Big((x-\mu_{sw})^T \Sigma_{sw}^{-1}(x-\mu_{sw})\Big)}_{\text{distance}} + \underbrace{\log\sqrt{\det\,(2\,\pi\,\Sigma_{sw})}}_{\text{normalizing factor}} \end{aligned}$$

  usually: $\sum_{sw} = \text{const}(s,w)$

- **Laplacian distribution**
  absolute distance rather than squared distance

$$p(x_d|s,w) = \frac{1}{2\,\sigma_{swd}}\, e^{-\left|\frac{x_d - \mu_{swd}}{\sigma_{swd}}\right|}$$

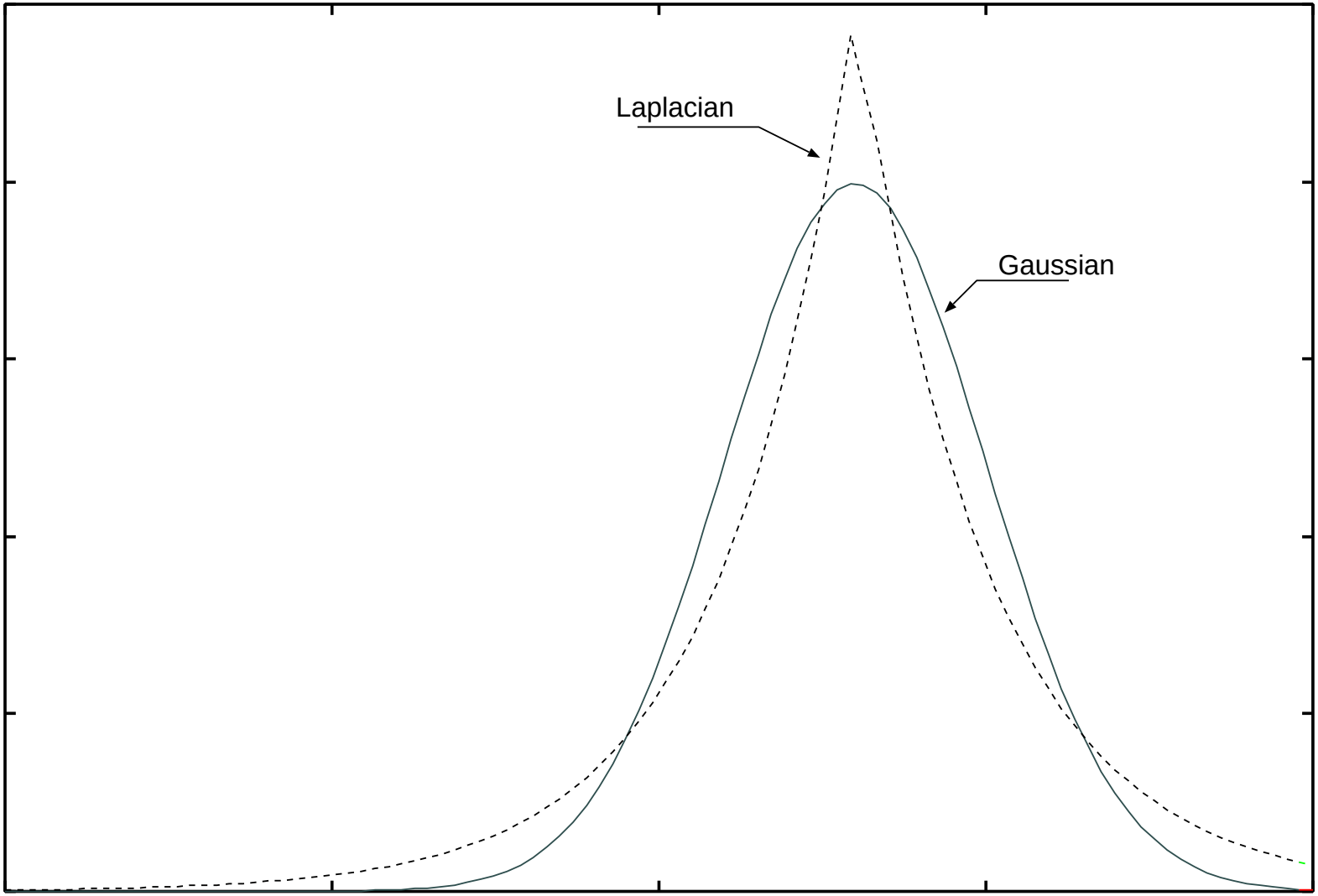
  μ_{swd} **mean (or median),** σ_{swd} **deviation**

- **With statistically independent components**

$$p(x|s,w) = \prod_{d=1}^{D} p(x_d|s,w)$$

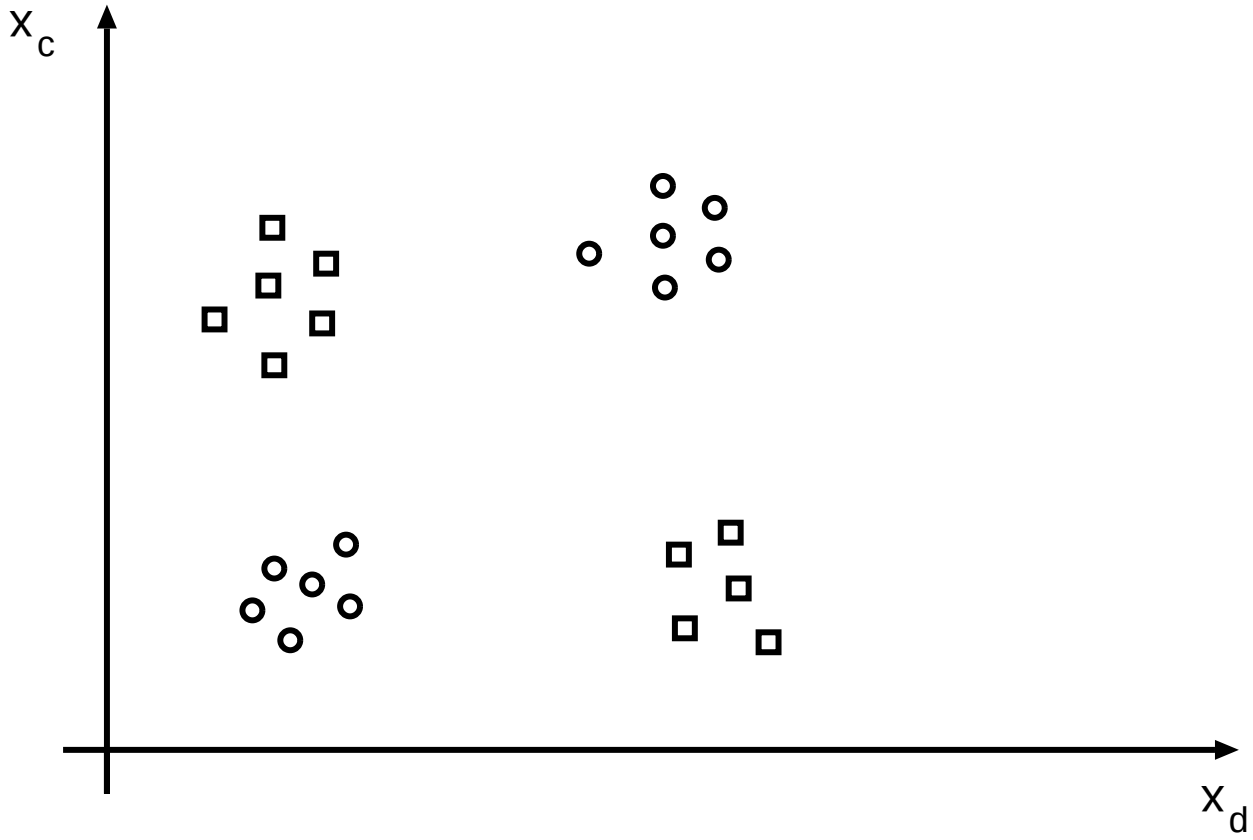
- **Negative logarithm**

$$\begin{aligned} d(x;s,w) &:= -\log p(x|s,w) \\ &= \underbrace{\sum_{d=1}^{D} \left|\frac{x_d - \mu_{swd}}{\sigma_{swd}}\right|}_{\text{distance}} + \underbrace{\sum_{d=1}^{D} \log\left(2\,\sigma_{swd}\right)}_{\text{normalizing factor}} \end{aligned}$$



Comparison: Gaussian and Laplacian distribution with equal deviation

2.2 Mixture Densities



Multimodal distributions

Multimodal distributions are well modelled with mixture densities.
Typical case: weighted sums of Gaussian of Laplacian densities, where each centre is identified with a unimodal density:

$$\begin{aligned} p(x|s,w) &= \sum_{l=1}^{L(s,w)} p(x,l|s,w) \\ p(x,l|s,w) &= p(l|s,w) \cdot p(x|s,w,l) \end{aligned}$$

with the triple $(l,s,w) = (density, state, word)$:

- $L(s,w)$: number of densities depending on state s and word w
- $p(x|s,w)$: multimodal distribution of state s from word w.
- $p(x|s,w,l)$: unimodal distribution for density l of state s from word w.
- $p(l|s,w)$: mixture weight with normalization: $\sum_l p(l|s,w) = 1$.

Gaussian mixture densities:

$$p(x|s,w) = \sum_l [p(l|s,w) \cdot p(x|s,w,l)]$$

$$p(x|s,w,l) = \frac{1}{\prod_{d=1}^{D} \sqrt{2\,\pi\,\sigma_{lswd}^2}} \; e^{-\frac{1}{2} \sum_{d=1}^{D} \left(\frac{x_d - \mu_{lswd}}{\sigma_{lswd}} \right)^2}$$

In practice, the sum is often approximated by the maximum:

$$p(x|s,w) = \max_l \{p(x,l|s,w)\}$$

$$= \max_l \{p(x|s,w,l) \cdot p(l|s,w)\}$$

Applying the negative logarithm then leads to:

$$-\log p(x|s,w) = \min_l \left\{ \frac{1}{2} \sum_{d=1}^{D} \left(\frac{x_d - \mu_{lswd}}{\sigma_{lswd}} \right)^2 - \log p(l|s,w) + \frac{1}{2} \sum_{d=1}^{D} \log \left(2\,\pi\,\sigma_{lswd}^2\right) \right\}$$

for the moment: for each state s and word w,
the number $L(s,w)$ of component densities is kept constant.

These steps are carried out iteratively:

- time alignment:
  assign each acoustic vector x_t to an index pair (s,w)
  and to a component density l
- estimation ('learning') of the model parameters for each triple (l,s,w):
  - reference vector μ_{lsw}
  - variance vector σ^2_{lsw}
  - mixture weight $p(l|s,w)$ (estimated as relative frequency)

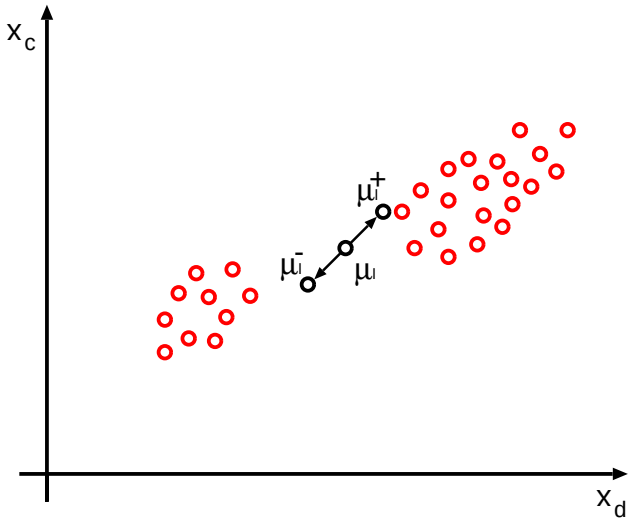
successive splitting of density (l, s, w):

- collect all observations and compute mean vectors μ_{lsw}
- splitting step:
  generate two new mean vectors $\mu^{+}_{lsw}, \mu^{-}_{lsw}$:

$$\begin{aligned} \mu^{+}_{lsw} &= \mu_{lsw} + \epsilon \cdot u \\ \mu^{-}_{lsw} &= \mu_{lsw} - \epsilon \cdot u \end{aligned}$$

  where u is a suitable direction vector, e.g. $u = (1, 1, ..., 1)$
- repeat several times:
  assign observations to nearest mean vector and update estimates: means, variances and mixture weights.

$\Rightarrow$ local optimum for the problem



2.3 Hidden Markov Models

- Idea: Represent each "part" of a word by a state of a (stochastic) finite state machine.
- Mathematical formulation:

  x_1^T temporal sequence of feature vectors

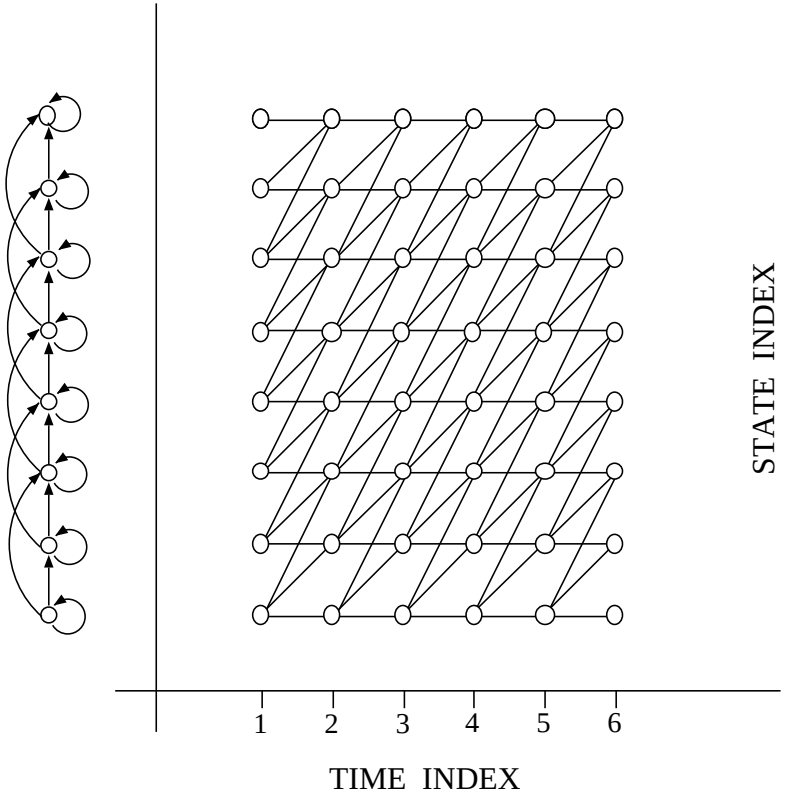
  $x_1, \ldots, x_t, \ldots, x_T$,

  s_1^T temporal sequence of states

  $s_1, \ldots, s_t, \ldots, s_T$.

$$p(x_1^T|w) = \sum_{[s_1^T]} p(x_1^T, s_1^T|w)$$

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



- Model assumptions for a word w:
  - The states $s = 1, \ldots, S(w)$ of a word w are an abstract concept that can not be observed ("hidden"):

$$p(x_t, s_t | x_1^{t-1}, s_1^{t-1}, w) = p(x_t, s_t | s_1^{t-1}, w)$$

  - The dependencies are restricted to the predecessor state s_t "first order", "Markovian".

$$\begin{aligned} p(x_t, s_t | s_1^{t-1}, w) &= p(x_t, s_t | s_{t-1}, w) \\ &= p(s_t | s_{t-1}, w) \cdot p(x_t | s_{t-1}, s_t, w) \\ &= \text{Transition Probability} \cdot \text{Emission Probability} \end{aligned}$$

  - Note:
    . the states correspond to the usual states of a regular grammar the observations are continuous.
    . "stochastic finite state automaton", "stochastic regular grammar", and "Hidden Markov Model" are all equivalent formulations.

The model assumption leads to the expression $p(x_1^T|w)$:

$$\begin{aligned} p(x_1^T|w) &= \sum_{[s_1^T]} \prod_{t=1}^{T} \left[p(s_t|s_{t-1}) \cdot p(x_t|s_{t-1}, s_t, w) \right] \\ &= \sum_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t|s_{t-1}, w) \end{aligned}$$

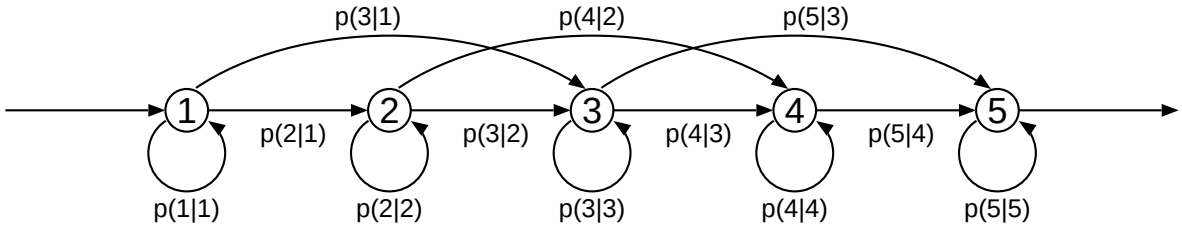
The sum is approximated by the maximum:

$$p(x_1^T|w) = \max_{s_1^T} \left\{ \prod_{t=1}^{T} p(x_t, s_t|s_{t-1}, w) \right\}$$

After applying the negative logarithm the expression is:

$$\min_{s_1^T} \sum_{t=1}^{T} \left\{ -\log p(x_t, s_t|s_{t-1}, w) \right\}$$

- **Special choice:**



  - **Linear sequence of states:** $s = 1, \ldots, S(w)$,
  - **Transition probabilities:**

$$p(s|s', w) = \begin{cases} q(s - s') & : \quad s \in \{s' + 0, s' + 1, s' + 2\} \\ 0 & : \quad otherwise \end{cases}$$

  - **Emission probabilities only depend on the state that is reached:**

$$p(x_t|s', s, w) = p(x_t|s, w)$$

then:

$$-\log p(x_t, s|s', w) = \underbrace{-\log p(x_t|s, w)}_{\text{distance}} \underbrace{-\log q(s - s')}_{\text{time distortion penalty}}$$
$$= d(x_t; s, w) + \mathcal{T}(s - s')$$

For this special model the optimization criterion is:

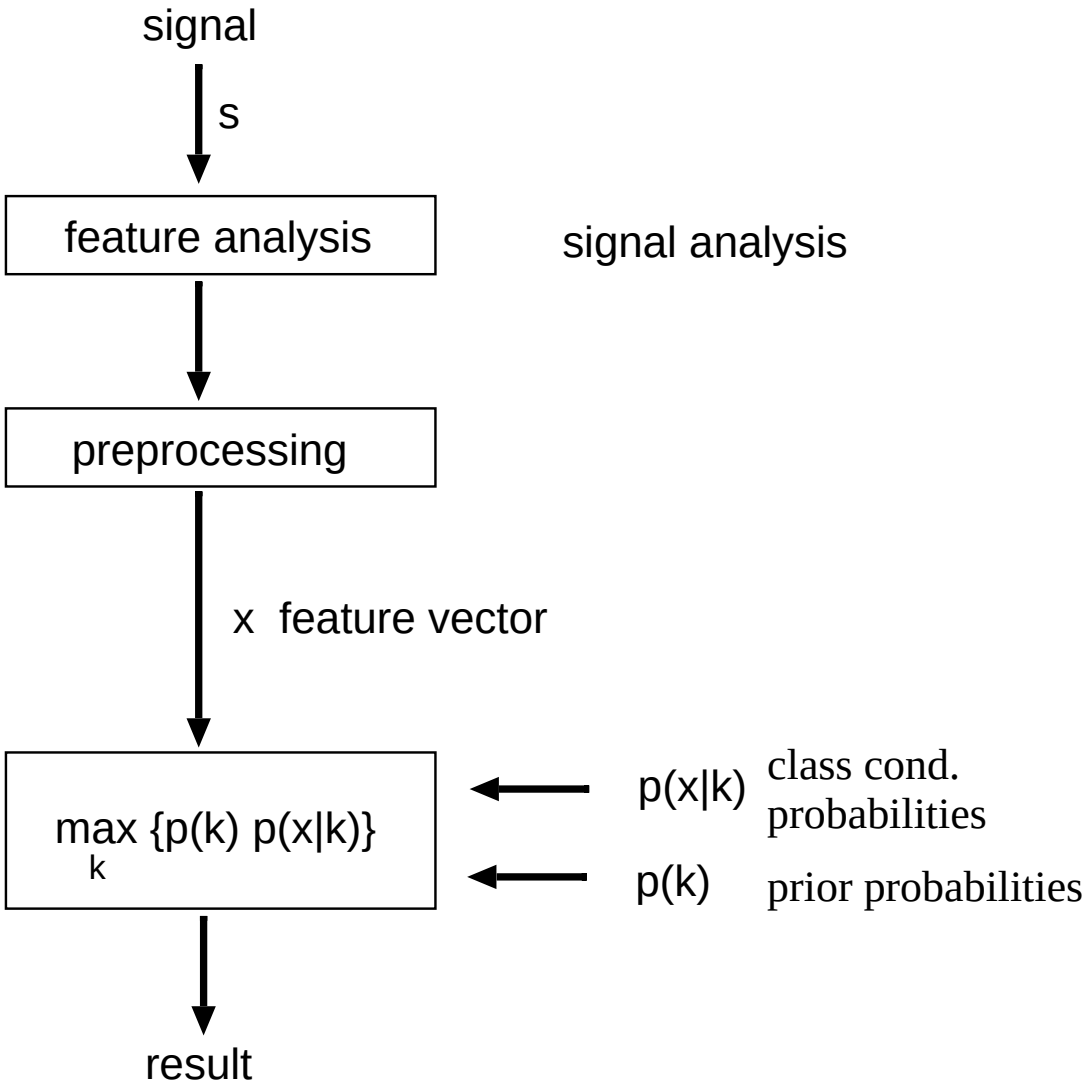
$$\min_{s_1^T} \sum_{t=1}^{T} \{ d(x_t; s_t, w) + \mathcal{T}(s_t - s_{t-1}) \}$$

This optimization criterion is identical to the one of the time alignment problem.
It is the time alignment problem in a statistical formulation.

2.4 Bayes Decision Rule

Requirement for a general pattern recognition system:

Given an observation $x \in X \subseteq \mathbb{R}^D$ find the class $k = 1, \ldots, K$ it belongs to.



- **Probabilities / probability densities**
  - **prior probability (e.g. relative frequencies)**

$$p(k)$$

  - **class conditional probabilities (e.g. Gaussian distributions)**

$$p(x|k)$$

  - **joint probability for $X \times K$**

$$p(x,k) = p(k) \cdot p(x|k)$$

  - **marginal probabilities for X**

$$p(x) = \sum p(x,k)$$

  - **a–posteriori probability**

$$p(k|x) = \frac{p(x,k)}{p(x)}$$

- **Find a decision rule**

$$\begin{aligned} r: \quad & X & \rightarrow \{1, \ldots, K\} \\ & x & \rightarrow r(x) \end{aligned}$$

  that minimizes recognition errors.

- **Bayes decision rule meets this demand**

$$\begin{aligned} r(x) &= \operatorname*{argmax}_k \{p(k|x)\} \\ &= \operatorname*{argmax}_k \{p(x,k)\} \\ &= \operatorname*{argmax}_k \{p(k) \cdot p(k|x)\} \end{aligned}$$

- A general discrimination function $g(x,k)$ can be introduced:

$$r(x) = \operatorname*{argmax}_{k} \{g(x,k)\}$$

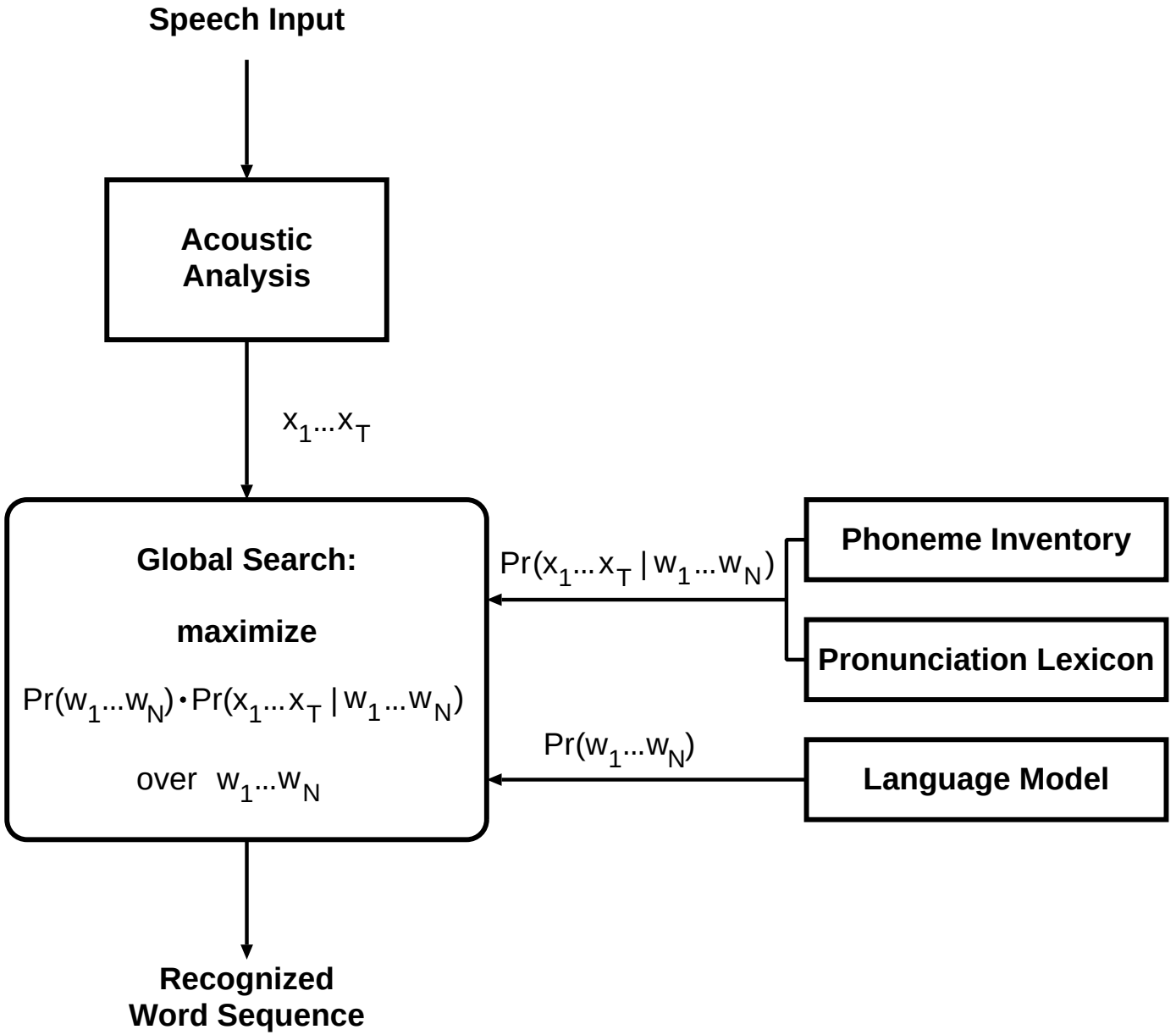
The decision rule is invariant with respect to monotonous transformations, these functions $g(x,k)$ lead to the same recognition results:

$$\begin{aligned}
g(x,k) &= p(k|x) \\
&= \frac{p(k) \cdot p(x|k)}{\sum_{c=1}^{K} p(c) \cdot p(x|c)} \\
g(x,k) &= p(k) \cdot p(x|k) \\
g(x,k) &= \log p(k) + \log p(x|k) \\
g(x,k) &= \log [p(k) \cdot p(x|k)] - \log \sum_{c=1}^{K} [p(c) \cdot p(x|c)]
\end{aligned}$$

- To design a recognition system four crucial problems have to be solved:
  - preprocessing and feature extraction:
    find suited features x for the task
  - modelling:
    specify models and structures for $p(k|x)$ and $p(k)$
  - training:
    learn the parameters from data
  - search:
    find the maximum during recognition

Application in speech recognition:

- class k: word sequence
  $w_1 \ldots w_n \ldots w_N = w_1^N$
- observation x:
  sequence of vectors
  $x = x_1 \ldots x_t \ldots x_T = x_1^T$
- search:
  maximize $Pr(w_1^N | x_1^T)$ over all possible word sequences w_1^N with unknown N.



- Solving the four crucial problems:
  - Feature extraction:
    Mel–frequency cepstral coefficients with first and second derivatives
  - Modelling:
    - Acoustic phonetic modelling: word as sequence of HMM–states

$$Pr(x_1^T|w_1^N) = \sum_{[s_1^T]} Pr\left(x_1^T, s_1^T \middle| w_1^N\right)$$

    - Language model:
      Unigram:

$$Pr(w_1^N) = \prod_{n=1}^{N} p(w_n)$$

      Bigram:

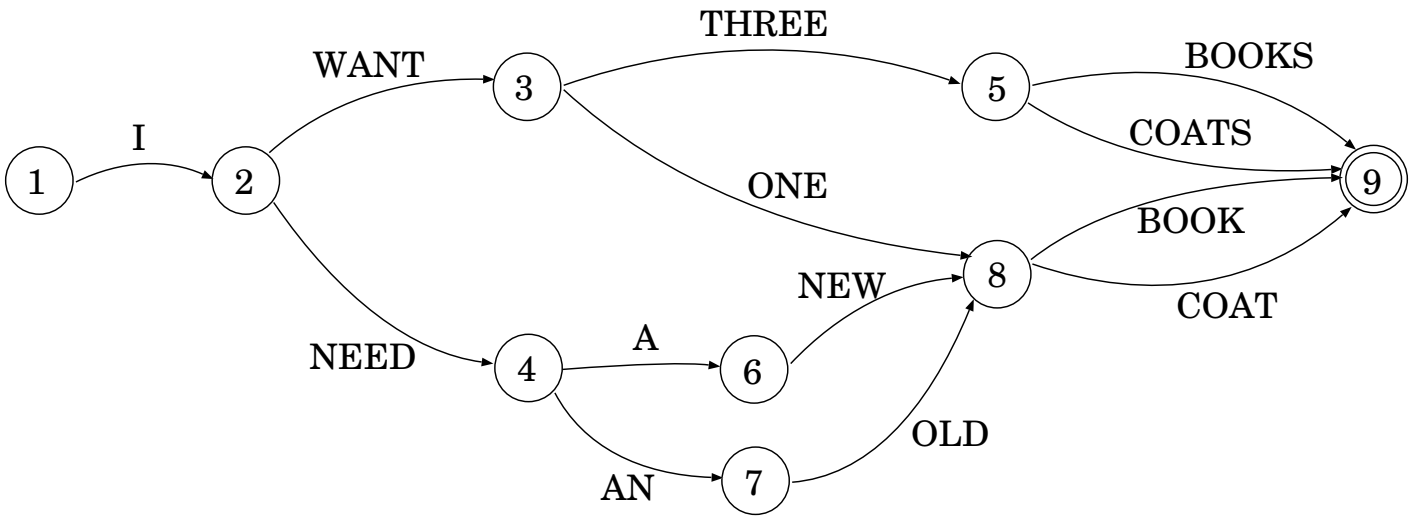
$$Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1})$$

Trigram:

$$Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-2}, w_{n-1})$$

Grammar (finite state network):

e.g. voice commands:



– **Search:**
acoustic model and language model have to be considered

$$\left[w_1^N\right]_{opt} = \underset{w_1^N}{\operatorname{argmax}} \left\{ Pr(w_1^N) \cdot Pr(x_1^T | w_1^N) \right\}$$

$$= \underset{w_1^N}{\operatorname{argmax}} \left\{ Pr(w_1^N) \cdot \sum_{s_1^T} Pr\left(x_1^T, s_1^T \middle| w_1^N\right) \right\}$$

The sum includes all paths that are consistent with the word sequence w_1^N

Using the maximum approximation the sum is replaced by the maximum:

$$\left[w_1^N\right]_{opt} = \underset{w_1^N}{\operatorname{argmax}} \left\{ Pr(w_1^N) \cdot \max_{s_1^T} Pr\left(x_1^T, s_1^T \middle| w_1^N\right) \right\}$$

The evaluation of the equation requires two steps:

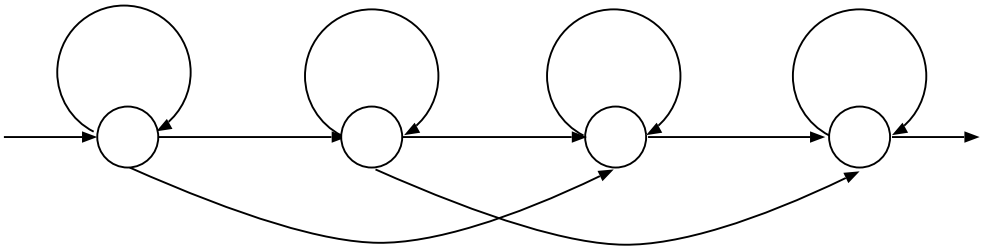
- Time alignment to maximize s_1^T
- Recognition of the best word sequence w_1^N

Note: the number of possible word sequences can be very large e.g.

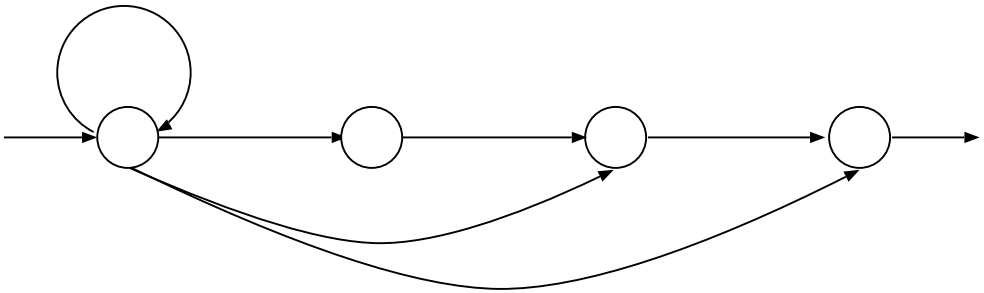
$$\begin{aligned} W &= 10000 \quad \text{words in the vocabulary} \\ N &= 10 \quad \text{typical sentence length} \\ W^N &= 10000^{10} = 10^{40} \quad \text{possible word sequences} \end{aligned}$$

HMM topologies:

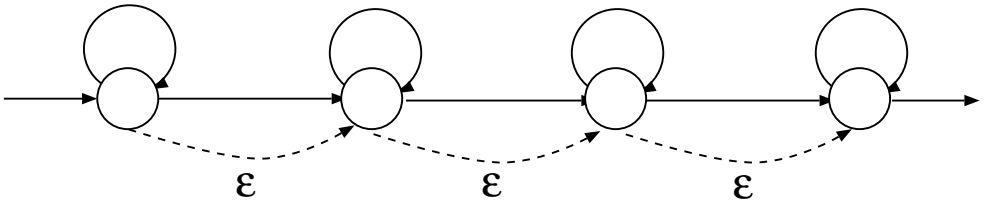
- **(0,1,2)-standard model**



- **Long skips**

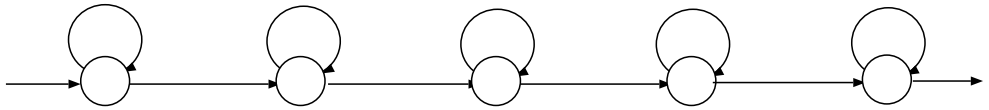


- **Empty transitions (without observations)**

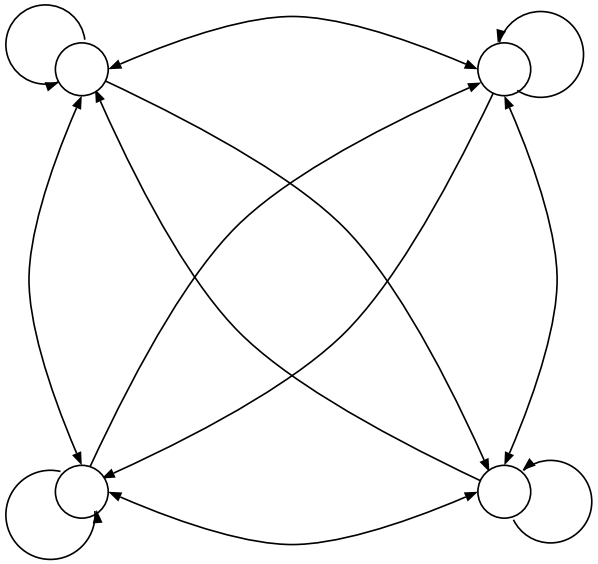


HMM topologies:

- **One state per phoneme (historic approach, not good)**



- **Ergodic model without left to right structure (not successful for recognition so far)**



- **Dynamic information:**

  Idea: do not use the current value x_t alone, consider the whole region around t

$$\cdots y_{t-2}\, y_{t-1}\, y_t\, y_{t+1}\, y_{t+2} \cdots \qquad y_t \in \mathbb{R}^D$$

$$x_t = \begin{bmatrix} y_t \\ y_t - y_{t-\delta} \end{bmatrix}$$

first temporal derivative

or:

$$x_t = \begin{bmatrix} y_t \\ y_t - y_{t-\delta} \\ y_{t-\delta} - 2y_t + y_{t+\delta} \end{bmatrix}$$

first and second temporal derivative

with typically $\delta = 2$

- **Time distortion penalty:**
  - **Model approach: count the transitions in training**

$$p(s'|s) = \frac{N(ss')}{\sum_{s''} N(ss'')} = \frac{N(ss')}{N(s)} ,$$

    were $N(ss')$ is the number of state sequences (ss').
  - **Homogeneous model: all states behave equally**

$$p(s'|s) = \begin{cases} q(s'-s) & s' = s+0,\ s+1,\ s+2 \\ 0 & \text{otherwise} \end{cases}$$

$$q(s'-s) = \frac{N(s'-s)}{\sum_{\delta_s} N(\delta_s)}$$

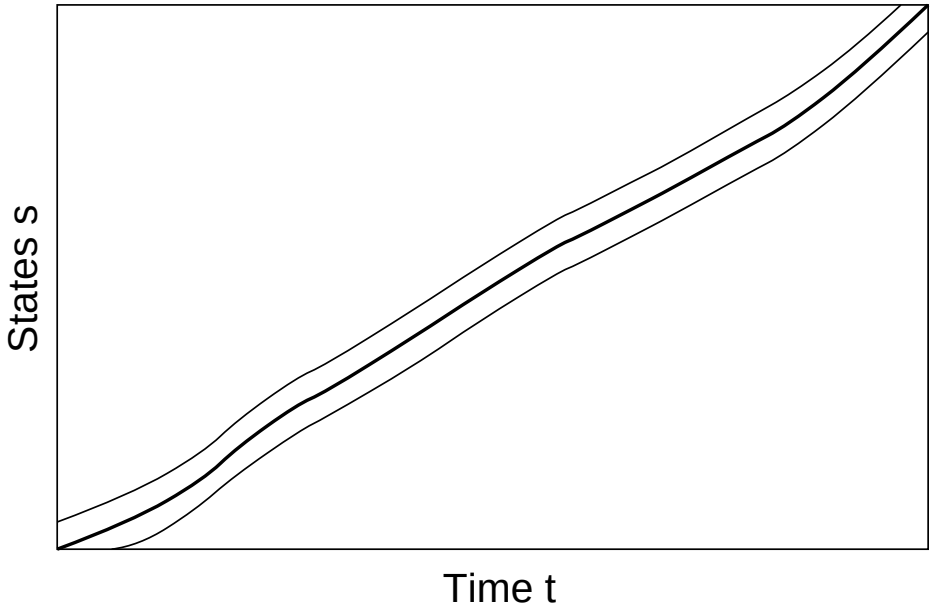
with: $\sum_{\delta_s} N(\delta_s)$ = **sum of the transitions**

- **Empirical approach:**

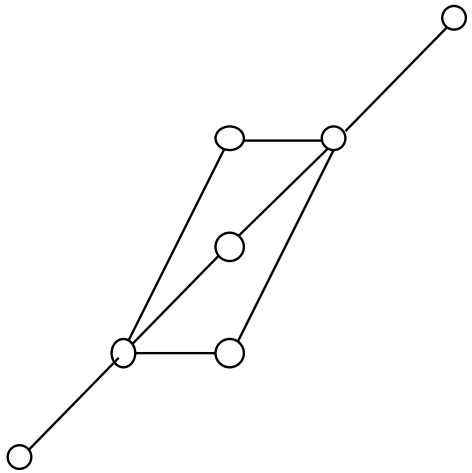
  Time distortion penalty: $\mathcal{T}(s' - s)$ with

  - **normalization: $\mathcal{T}(1) = 0$**
  - **symmetry: $\mathcal{T}(2) = \mathcal{T}(0)$**

  If the paths "neighboring" the best path are considered, $\mathcal{T}(0)$ and $\mathcal{T}(2)$ can be calculated from the average local distance between the best path and its neighbors.



- **Thus in this example:**



  - **local distance $d(x_t; s)$, and**
  - **time distortion penalty $\mathcal{T}(s - s')$**

  will both contribute to the best path.

3 Connected Word Recognition

3.1 Interdependence of the Decisions 143

3.2 Optimization Criterion 146

3.3 Dynamic Programming 151

3.4 Refinement of the DP Search 155

3.5 Implementation 163

3.6 Recognition Results 169

3.1 Interdependence of the Decisions

The boundary between words is not evident in the signal.
(Try to identify word boundaries when listening to a foreign language!)

Idea: build a model for whole sentence by concatenating the corresponding word models.

The search process will now optimize over

- **all sentence lengths (number of words)**
- **all word identities**
- **all possible word boundaries**
- **time alignment**
- **speech / non-speech (silence, noise,...)**
- **language constraints**
  (syntax, semantic,...)

These decisions are not independent:

German: wen ich — wenig
English: I worry — ivory

Bayes decision rule:

$$\left[w_1^N\right]_{opt} = \arg\max_{w_1^N}\left\{Pr(w_1^N)\cdot\sum_{s_1^T} Pr\left(x_1^T, s_1^T \middle| w_1^N\right)\right\}$$

$$\cong \arg\max_{w_1^N}\left\{\underbrace{Pr(w_1^N)}_{\text{language model}} \cdot \max_{s_1^T} \underbrace{Pr\left(x_1^T, s_1^T \middle| w_1^N\right)}_{\text{acoustic model}}\right\}$$

where we have used the so-called maximum approximation.

note:
for the maximum approximation to work, the maximizing arguments rather than the numeric scores must be identical!

- acoustic models: HMM
  - (0,1,2)-standard model for isolated words
    linear sequence of states for $s = 1 \ldots S(w)$ all words w.
  - product of transition and emission probabilities:

$$p(x_t, s|s', w) = p(s|s', w) \cdot p(x_t|s, w)$$

  - build super HMM for each hypothesis w_1^N:

$$Pr\left(x_1^T, s_1^T \middle| w_1^N\right) = \prod_{t=1}^{T} p(x_t, s_t|s_{t-1}, w_1^N)$$

    with observations x_1^T and states s_1^T

- language model:

  as simplification a unigram language model shall be considered here

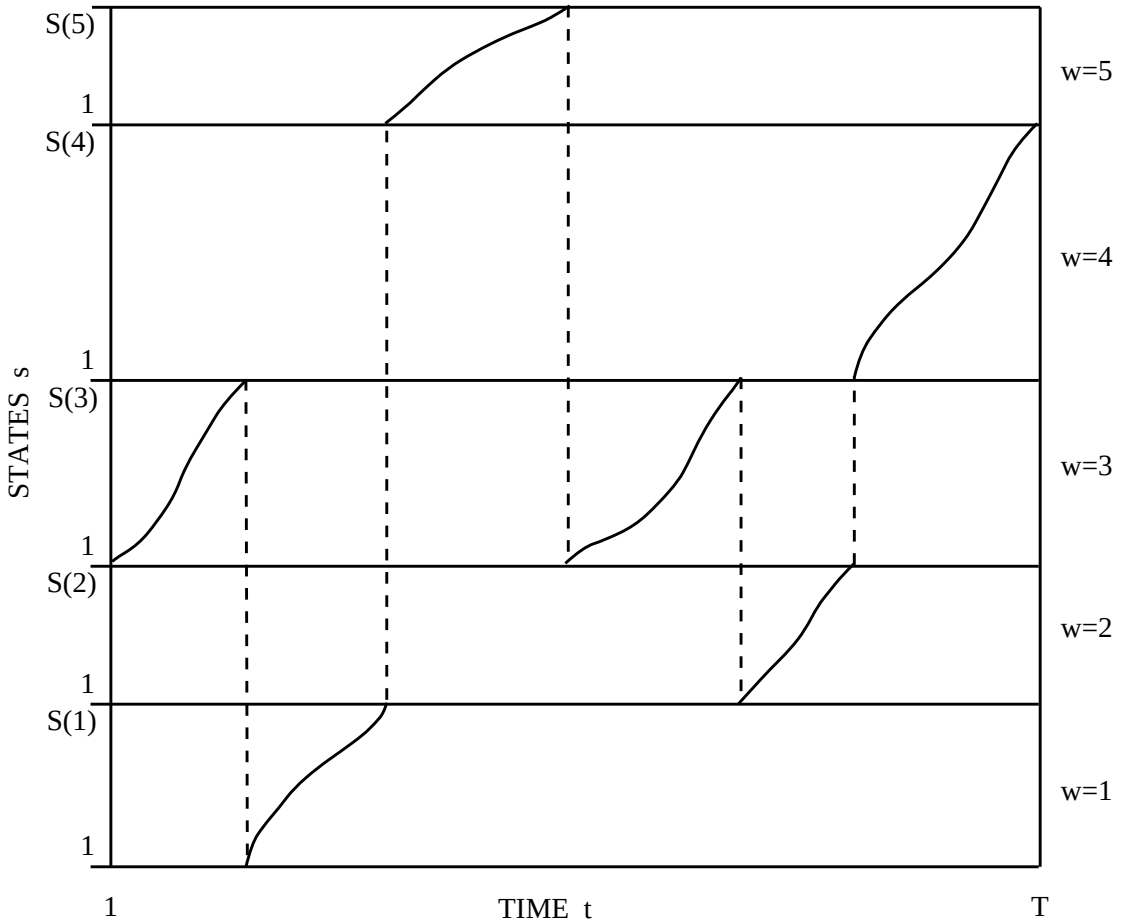
$$Pr(w_1^N) = \prod_{n=1}^{N} p(w_n)$$

3.2 Optimization Criterion

The Bayes decision rule takes into account all interdependencies.

Instead of the sequence $[s_1^T, w_1^N]$ we consider a path that does not only assign t a state s_t but also a word index w_t to every time frame:

$$t \rightarrow [s_t, w_t] \quad .$$



Rewrite the optimization criterion from $[s_1^T, w_1^N]$ to $[s_1^T, w_1^T]$
(with word boundaries $t_0^N, t_0 = 0, t_N = T$):

$$\max_{N, w_1^N} \left\{ Pr(w_1^N) \cdot \max_{[s_1^T]} Pr(x_1^T, s_1^T | w_1^N) \right\} =$$

$$= \max_{N, w_1^N} \left\{ \prod_{n=1}^{N} p(w_n) \cdot \max_{s_1^T} \prod_{t=1}^{T} p(x_t, s_t | s_{t-1}, w_1^N) \right\} \quad \text{with } p(\cdot|\cdot) \text{ of super HMM}$$

$$\left[\text{change path } s_1^T \text{: from super HMM } w_1^N \text{ to single HMM } w_n \right]$$

$$= \max_{N, w_1^N} \max_{t_1^N} \left\{ \prod_{n=1}^{N} \left[p(w_n) \cdot \max_{s_{t_{n-1}+1}^{t_n}} \left(\prod_{t=t_{n-1}+1}^{t_n} p(x_t, s_t | s_{t-1}, w_n) \right) \right] \right\}$$

$$= \max_{s_1^T, w_1^T} \left\{ \prod_{t=1}^{T} \tilde{p}(x_t, s_t | s_{t-1}, w_t) \right\}$$

with a suitable definition of $\tilde{p}(x_t, s_t | s_{t-1}, w_t)$

with the definition for $\tilde{p}(x_t, s|s', w)$
(i.e. with the unigram language model) using the transition and emission probabilities of HMM:

- within words:

$$\tilde{p}(x_t, s|s', w_t) = p(s|s', w) \cdot p(x_t|s, w)$$

- word boundaries:

$$\tilde{p}(x_t, s|s', w_t) = p(w_t) \cdot p(x_t, s|s' = 0, w_t)$$

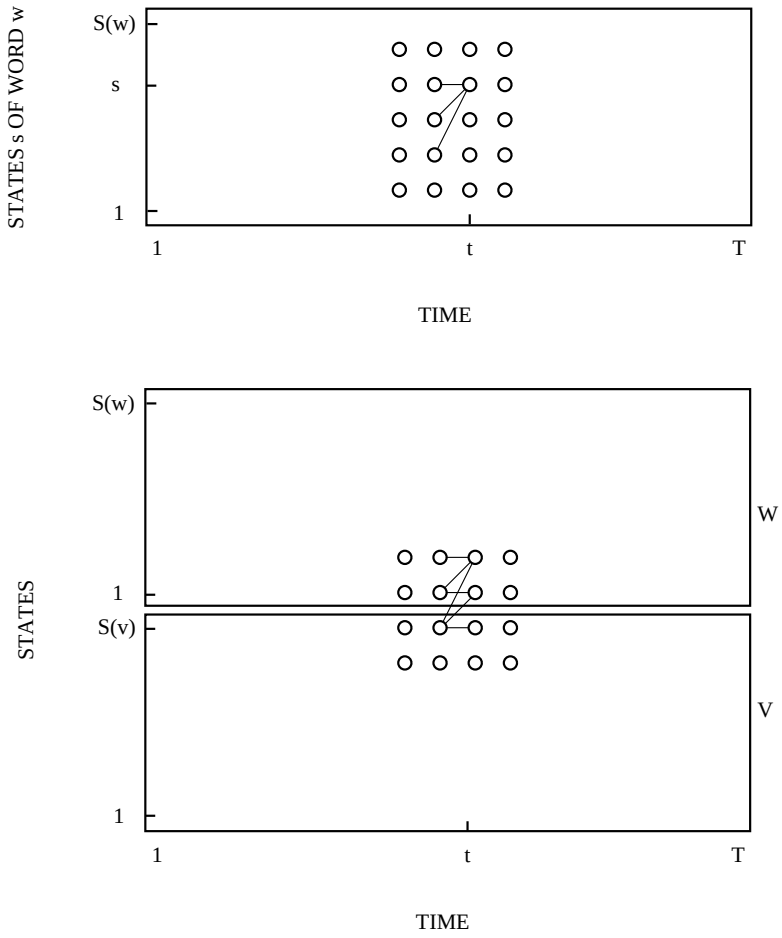
  where the virtual state $s' = 0$ serves as start state for each word HMM.

Transition rules or path constraints:
for a hypothesis (t, s_t, w_t), consider the possible predecessor hypotheses:

- **within a word:**
  as for isolated word recognition

- **at word boundaries:**

$$s_{t-1} = S(w_{t-1})$$
$$s_{t-1} > s_t \in \{1, 2\}$$



Complexity estimation for digit strings (i.e. 10 different words) the average length of a digit shall be 30 acoustic vectors

- Number of possible digit strings:

$$10^7 + 10^6 + \ldots 10 \cong 10^7$$

- Number of possible digit boundaries:

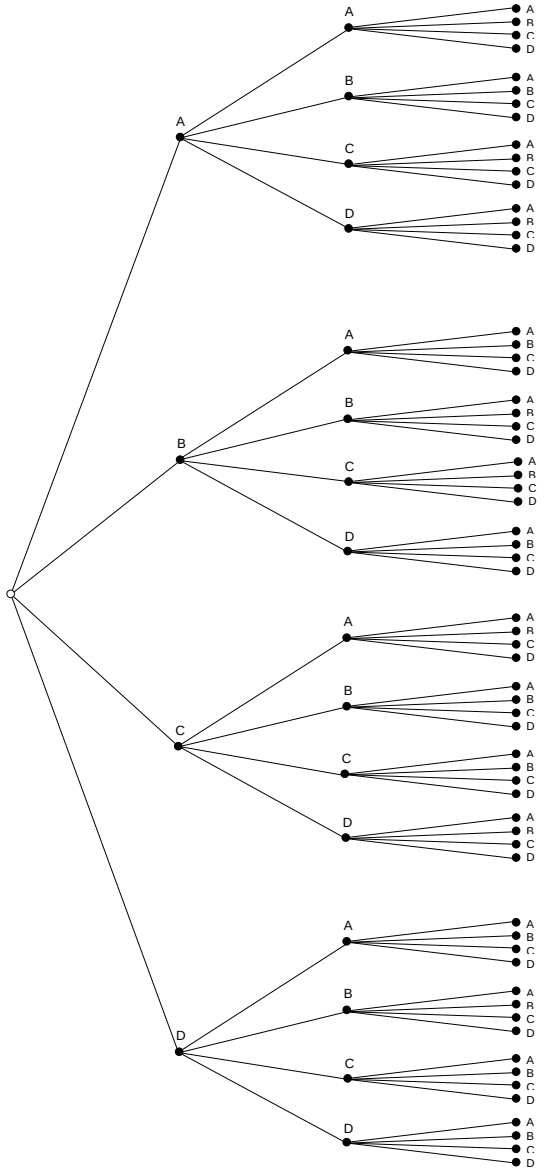
  a maximum of 6 digit boundaries in a sequence of $30 \cdot 7$ acoustic vectors:

$$\Rightarrow \frac{(30 \cdot 7)!}{6! \cdot (30 \cdot 7 - 6)!}$$

- Number of possible paths:

  approx. 3^{30} possible paths within a digit,

$$\Rightarrow \text{approx. } 3^{30} \cdot 7 \cdot 10^7 \cong 1.44 \cdot 10^{22}$$



3.3 Dynamic Programming

For dynamic programming an auxiliary quantity is defined:

$$Q(t,s;w) = \max_{[s_1^t, w_1^t]} \left\{ \prod_{\tau=1}^{t} \tilde{p}(x_\tau, s_\tau | s_{\tau-1}, w_\tau) : (s_t, w_t) = (s, w) \right\}$$

$$= \text{joint probability for the best partial path up to time } t$$

$$\text{ending in state } s \text{ of word } w.$$

There are two different recursion expressions corresponding to the different transition rules:

- within words:

  as for isolated word recognition:

$$Q(t, s; w) = \max_{s'} \{Q(t-1, s'; w) \cdot p(x_t, s|s', w)\}$$

- word boundaries:

  a special state $s = 0$ is introduced as initial state for a new word w:

$$Q(t, s = 0; w) = p(w) \cdot \max_{v} \{Q(t, S(v); v)\}$$

Note that the function $Q(t, s; w)$ takes into account the acoustic as well as the language model (here a unigram).

formulation in terms of distances:
for efficient implementation, the (negative) logarithms are used:

$$D(t,s;w) = \min_{[s_1^t, w_1^t]} \left\{ \sum_{\tau=1}^{t} d(x_\tau, s_\tau | s_{\tau-1}, w_\tau) : (s_t, w_t) = (s, w) \right\}$$

$$\text{with} \quad d(x_t, s_t | s_{t-1}, w_t) = -\log \tilde{p}(x_t, s_t | s_{t-1}, w_t)$$

- within words:

$$D(t,s;w) = \min_{s'} \{ D(t-1, s'; w) + d(x_t, s | s', w) \}$$

- word boundaries:

$$D(t, s=0; w) = -\log p(w) + \min_{v} \{ D(t, S(v); v) \}$$

Carry out dynamic programming by 'filling the table $D(t, s; w)$' :

- outer loop over $t = 1, \ldots, T$:

  hypotheses $Q(t, s; w)$ are propagated time synchronously from left to right.

- if $t = T$ (the end is reached):

  'traceback' the decisions to find the best path $t \rightarrow [s_t, w_t]$ representing the spoken word sentence.
  Start from the 'best word end' at $t = T$

3.4 Refinement of the DP Search

Along with the hypotheses $Q(t, s; w)$, backpointers $B(t, s; w)$ are propagated during dynamic programming:

- Word interior, i.e. $s > 0$:

$$\begin{aligned} Q(t,s;w) &= \max_{s'} \{Q(t-1,s';w) \cdot p(x_t, s|s';w)\} \\ \sigma(t,s;w) &:= \operatorname*{argmax}_{s'} \{Q(t-1,s';w) \cdot p(x_t, s|s';w)\} \\ B(t,s;w) &= B(t-1, \sigma(t,s;w); w) \end{aligned}$$

  The backpointer of the best predecessor hypothesis is passed on.

- Word boundaries, i.e. $s = 0$:

$$\begin{aligned} Q(t,0;w) &= p(w) \cdot \max_{v} \{Q(t, S(v); v)\} \\ B(t,0;w) &= t \end{aligned}$$

  Each hypothesis $(t, s = 0, w)$ is assigned the start time t (or end time of predecessor word).

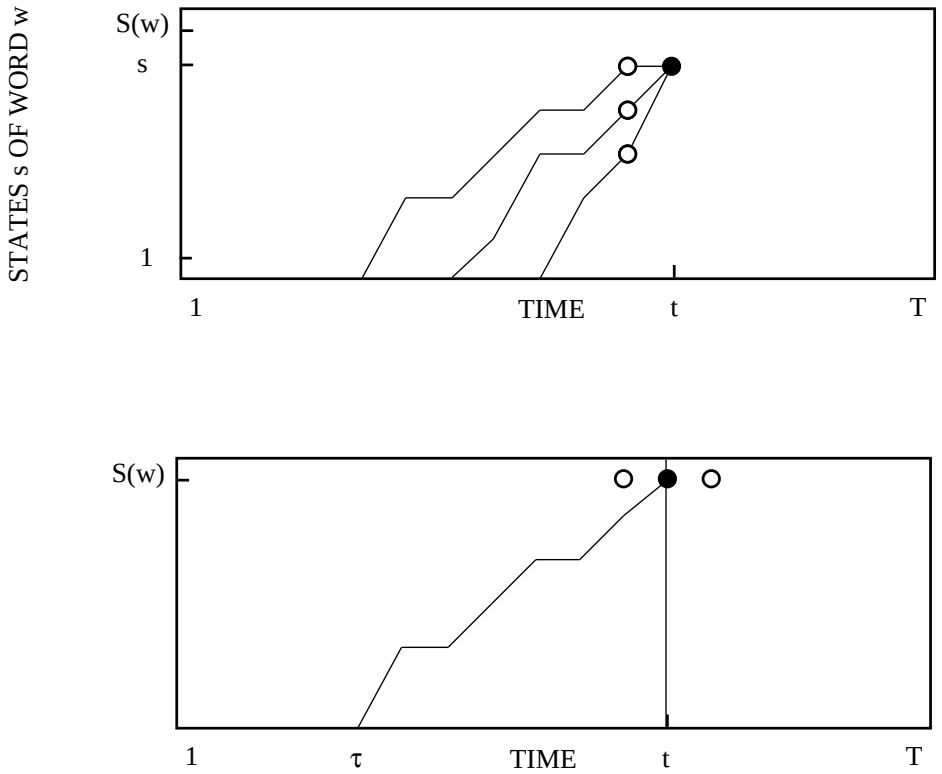
To store the decisions at the word boundaries, we introduce the so-called traceback arrays:

best word end:	$W(t)$	$:=$	$\underset{w}{\operatorname{argmax}} \{Q(t, S(w); w)\}$
best start time:	$B(t)$	$:=$	$B(t, S(W(t)); W(t))$
best word score (optional):	$Q(t)$	$:=$	$\max_{w} \{Q(t, S(w); w)\}$

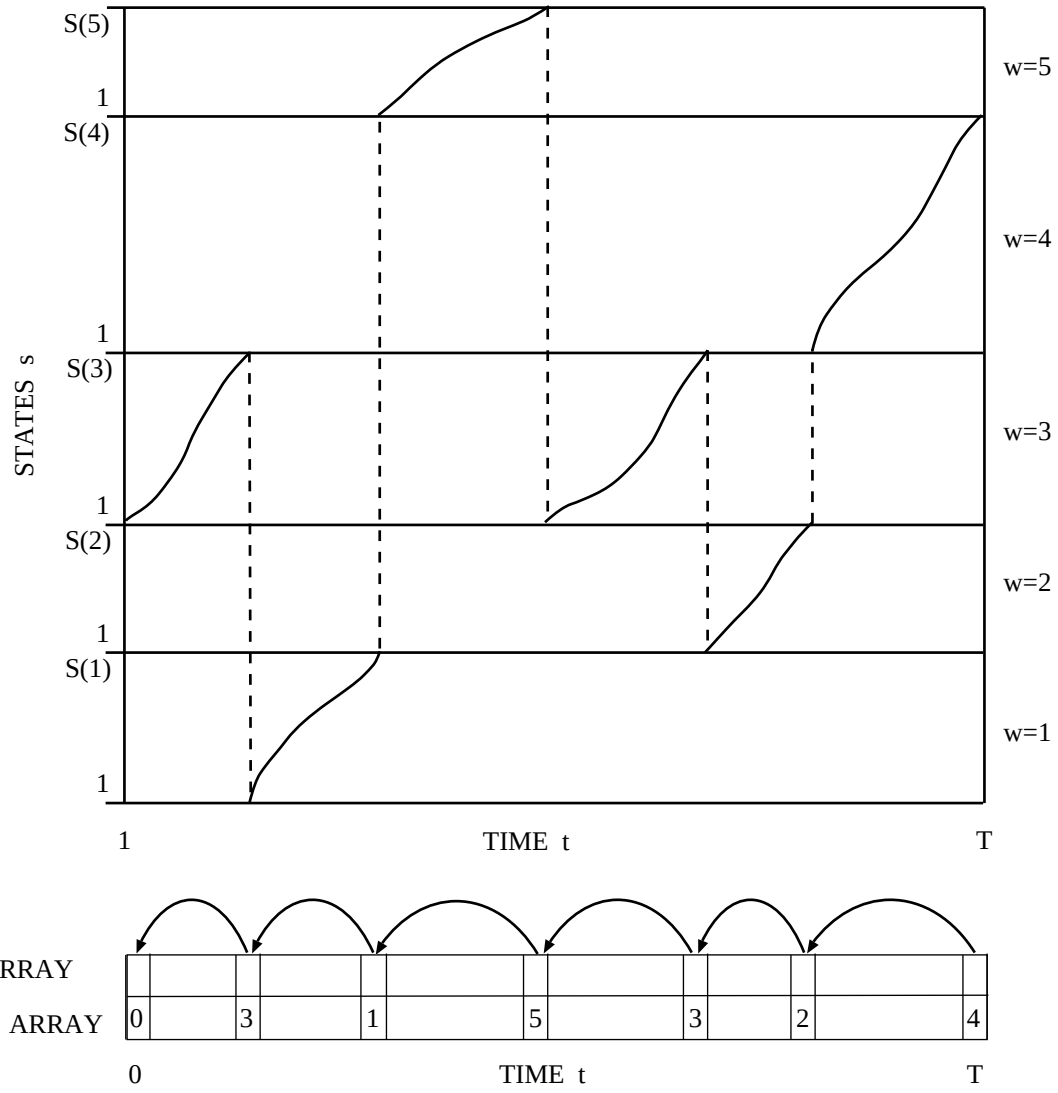
For every time frame, the traceback arrays store the best word end hypothesis, its start time (and optionally the word end score).

backpointer $B(t, s, w)$ of hypothesis (t, s, w):

reports start time of word w



Alignment path and backpointers



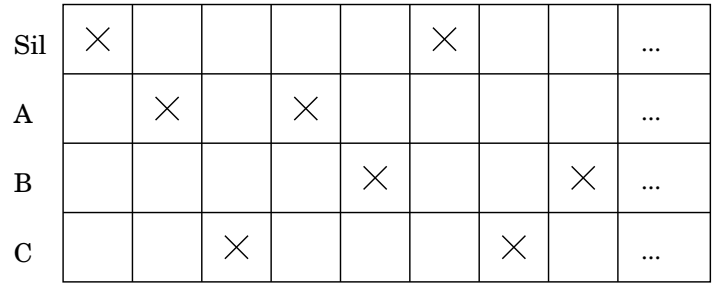
Properties of the traceback arrays:

a) **At every time frame any word can end. Determine the "best" ending word and store it in the traceback arrays.**

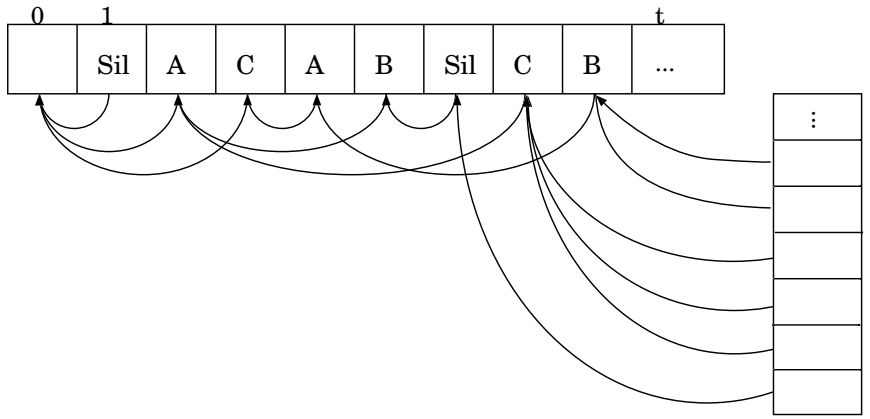
b) **This entry corresponds to the beginning of several new words.**

c) **The traceback arrays describe the hypothesized preceding word sequences as a tree.**
Each entry in the array (node) has only one predecessor.
The root corresponds to the entry at time $t = 0$

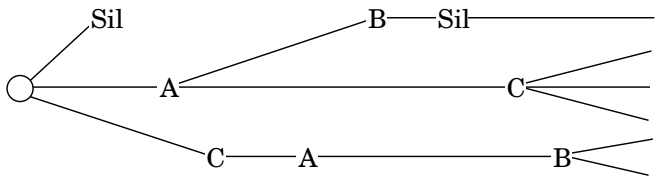
a)



b)



c)



Complexity of dynamic programming:

- **time complexity:**

$$T \cdot W \cdot \overline{S}$$

- **Memory requirement:**

  simple version:

  - **requires a full table $Q(t, s; w)$:**

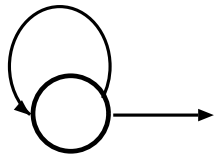
$$T \cdot W \cdot \overline{S}$$

  'refined' version:

  - **(s, w)-columns for $Q(t, s; w)$ and $B(t, s; w)$,**
  - **2 traceback arrays of length T or 3, if $Q(t)$ is stored.**

$$2 \cdot W \cdot \overline{S} + 2 \cdot T$$

- To allow for speech pauses, introduce a single-state silence model Sil:



This model does not affect the 'true words'; it is ignored by the language model:

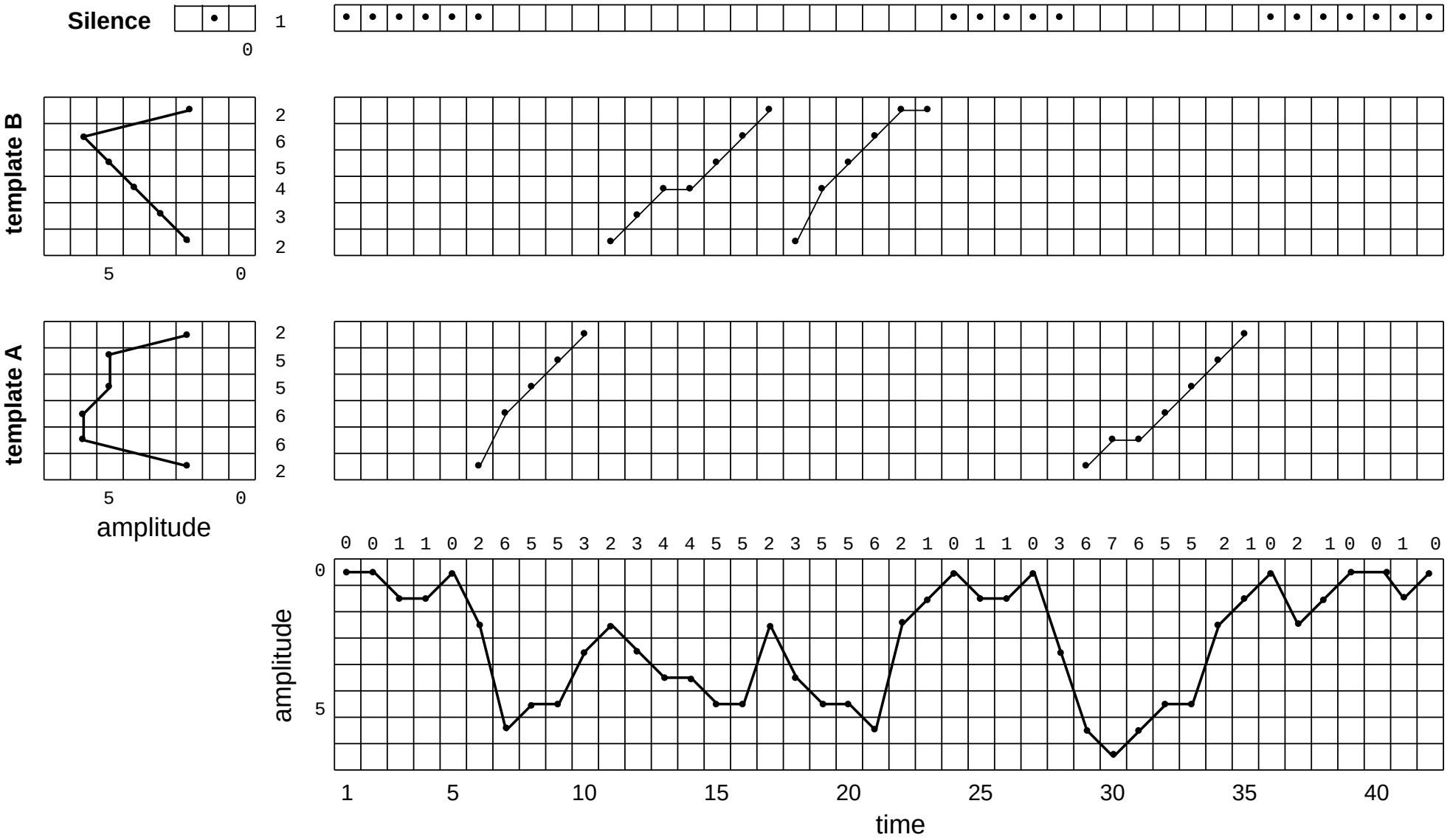
$$\begin{aligned} \text{“}p(w = Sil) &= 1\text{“} \\ \log p(w = Sil) &= 0 \end{aligned}$$

- Language model:
  we may use the same constant probability for each word w:

$$p(w) = const$$

Thus, a cost is introduced for each word hypothesis and long word sequences are penalized. This is desirable to counteract the number of possible word sequences which increases with the number of words.

Example: connected word recognition with two templates



3.5 Implementation

- **Recognition**
- **Training**

acoustic vectors: x_t over time t; local distance: $d(x_t, s, w)$;
time distortion penalty: $T(s - s', w)$

INITIALIZE ARRAYS

FOR EACH TIME FRAME $t = 1, \ldots, T$ OF THE INPUT DO

FOR EACH REFERENCE WORD $w = 1, \ldots, W$ DO

WORD TRANSITION $D(t-1, 0; w) = D(t-1, S(W(t-1)); W(t-1)) - \log p(w)$

$B(t-1, 0; w) = t - 1$

FOR EACH STATE $s = 1, \ldots, S(w)$ OF WORD MODEL w DO

DETERMINE BEST PATH TO GRID POINT $(t, s; w)$:

SCORE: $D(t, s; w) = D(t-1, s''; w) + T(s - s'', w) + d(x; s, w)$

BACKPOINTER: $B(t, s; w) = B(t-1, s''; w)$

KEEP TRACK OF DECISIONS AT POTENTIAL WORD BOUNDARIES:

$w'' = \text{argmin}\{D(t, S(w'), w') : w' = 1, \ldots, W\}$

TRACEBACK ARRAY: START TIME $B(t) = B(t, S(w''); w'')$

WORD $W(t) = w''$

DETERMINE THE WORD BOUNDARIES $T = t(0), t(1), \ldots, t(n), \ldots, t(N) = 0$

AND THE WORDS $w(1), \ldots, w(n), \ldots, w(N)$

START: $n = 0 \quad t(n) = T$

REPEAT TRACEBACK UNTIL WORD BOUNDARY $t = 0$ IS REACHED

$n = n + 1 \quad t(n) = B(t(n-1))$

$w(n) = W(t(n-1))$

Implementation example in C–code: **show** /dynamic-programming.pdf

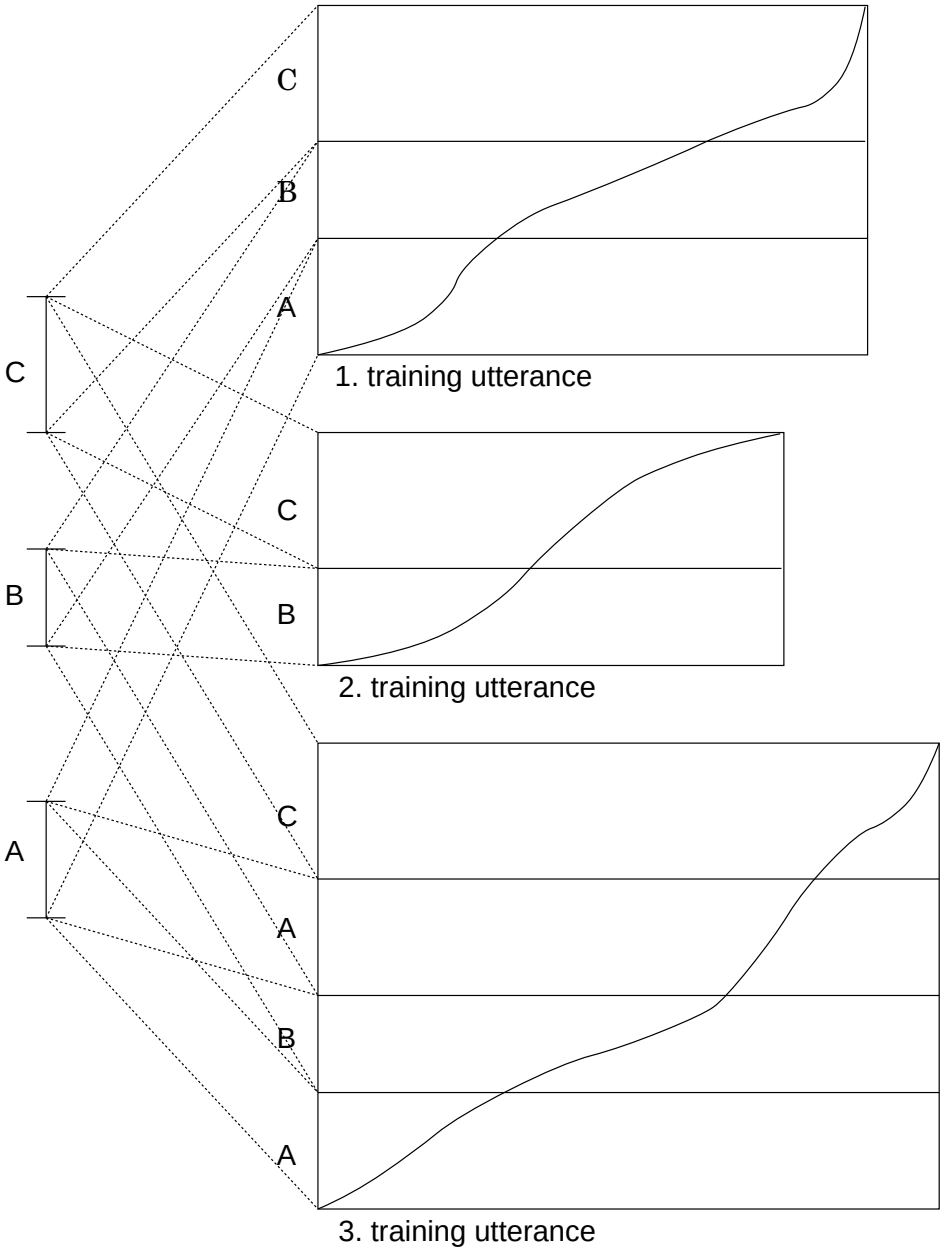
Illustration of Time Alignment in Training

example:

- **vocabulary: word models A, B, C**
- **training data: three utterances**

training procedure:

- **concept: see chapter 2**
- **implementation: here**



LOOP OVER PASSES: `pss = 1, ..., N_pss`

INITIALIZE RUNNING SUMS AND COUNTS FOR ALL MIXTURES AND DENSITIES

FOR EACH SENTENCE `snt = 1, ..., N_snt` **OF TRAINING DATA DO**

CREATE SEQUENCE OF STATES `mix_snt` **OF THE SENTENCE**

TIME ALIGNMENT (linear segmentation in first pass)

FOR EACH TIME FRAME `tim = 1, ..., N_tim` **OF THE SENTENCE DO**

ACCUMULATE SUMS AND COUNTS:

`tim` $\longrightarrow$ `(mix,dns)`

reference vector:	`S_ref[dns] = S_ref[dns] + Vct[tim]`
deviation vector:	`S_dev[mix] = S_dev[mix] + abs(Vct[tim]-ref[dns])`
count of densities:	`C_dns[dns] = C_dns[dns] + 1`
count of mixtures:	`C_mix[mix] = C_mix[mix] + 1`

ESTIMATE PARAMETERS:

reference vector:	`ref[dns] = S_ref[dns] / C_dns[dns]`
deviation vector:	`dev[mix] = S_dev[mix] / C_mix[mix]`
weight:	`weight[dns] = -log ( C_dns[dns] / C_mix[mix] )`

SPLIT DENSITIES (optional):

new reference vectors:	`ref[dns_new] = S_ref[dns] / C_dns[dns] + ϵ`
	`ref[dns_new] = S_ref[dns] / C_dns[dns] - ϵ`
new weight:	`weight [dns_new] = -log ( C_dns[dns] / C_mix[mix] )`

Implementation example in C–code: show /training.pdf

3.6 Recognition Results

Task: digit string recognition

Acoustic analysis (feature extraction):

- **preemphasis: enhancement of high-frequency Fourier components**
- **Hamming window of 15-ms length**
- **mel cepstrum**
- **normalization steps: cepstral mean and energy normalization**
- **acoustic vector (D=48) including derivatives:**
  - **16 position vector (measurement)**
  - **16 first-order derivative**
  - **16 second-order derivative**

HMMs for whole-words:

- **vocabulary: 11 classes (10 digits + "oh"):**
  - **gender dependent models**
  - **for each gender: 357 states + 1 state for silence**
- **Laplacian densities:**
  - **single and mixture distributions**
  - **state dependent deviation vectors**

TI digit string database: speaker–independent

number of	training		testing		total
	male	female	male	female	
speakers	55	57	56	57	225
strings	4235	4388	4311	4389	17323
digits	13915	14414	14159	14424	56912
duration [min]	122	132	125	131	510

17323 strings = 56912 digits = 8.5 h of speech

= 1.25 GByte of sampled speech signal (20 kHz, 16 bit)

= 97.8 MByte of cepstrum vectors

Experiments to test various aspects of the baseline system:

- analysis window: 15 ms
- normalization steps
- acoustic vector
- number of mixture components
- type of training criterion:
  maximum likelihood vs. maximum mutual information:

$$\max_{\vartheta} \left\{ \sum_n \log p(x_n|c_n) \right\} \quad \text{vs.} \quad \max_{\vartheta} \left\{ \sum_n \log p(c_n|x_n) \right\}$$

Window length and preemphasis

window length		sub/del/ins	WER [%]	SER [%]
10 ms	(baseline system)	133/76/36	0.86	2.36
15 ms		132/70/26	0.80	2.24
20 ms		115/63/36	0.75	2.13
25 ms		122/74/64	0.91	2.60
15 ms, no preemphasis		133/73/40	0.86	2.45

Normalization setups

	sub/del/ins	WER [%]	SER [%]
baseline system	132/70/26	0.80	2.24
no mel scale	151/90/27	0.94	2.74
no spectral normalization	167/74/38	0.98	2.61
no energy normalization	129/71/36	0.83	2.31

WER: Word Error Rate, SER: Sentence Error Rate

Acoustic modelling

	sub/del/ins	WER [%]	SER [%]
baseline system (16–16–16)	132/70/26	0.80	2.24
no energy (15-16-16)	170/73/37	0.98	2.61
no 2nd-order deriv. (16-16-0)	132/74/27	0.82	2.28
no deriv. (16-0-0)	317/153/75	1.91	5.62
Laplacian dens., pooled dev. (?)	133/90/30	0.89	2.56
Gaussian densities	156/93/39	1.01	2.69

Number of component densities per mixture (state)

densities per state	sub/del/ins	WER [%]	SER[%]
1	132/70/26	0.80	2.24
2	109/60/26	0.68	1.91
4	94/55/18	0.58	1.69
8	76/46/19	0.49	1.39
16	62/36/12	0.38	1.10
32	53/32/15	0.35	1.01
64	58/17/24	0.35	0.99

two training criteria: ML vs.MMI

- single Gaussian densities
- linear discriminant analysis (LDA)

	sub/del/ins	WER [%]	SER [%]
ML	56/31/120	0.72	2.00
MMI	36/24/75	0.47	1.32

type of training criterion:
ML (maximum likelihood) vs. MMI (maximum mutual information):

$$\max_{\vartheta} \left\{ \sum_n \log\, p(x_n|c_n) \right\} \quad \text{vs.} \quad \max_{\vartheta} \left\{ \sum_n \log\, p(c_n|x_n) \right\}$$

Comparison with results reported in literature

authors	publication	WER [%]	SER [%]	remarks
Doddington	ICASSP–89	0.5	1.5	LDA
Gauvain, Lee	ICASSP–92	0.3	0.9	corr.train.
Haeb–Umbach, Ney	ICASSP–93	0.28	0.84	LDA
Normandin	IEEE Tr. SA'94	0.29	0.89	MMI
Welling, Eiden, Ney	EuroSpeech–95	0.24	0.72	LDA

4 Large Vocabulary Speech Recognition

4.1 Overview: Architecture 178

4.2 Phoneme Models and Subword Units 183

4.3 Language Modelling 205

4.4 Dynamic Programming Beam Search along with Implementation Details 249

4.5 Excursion (for experts): Language Model Factor 264

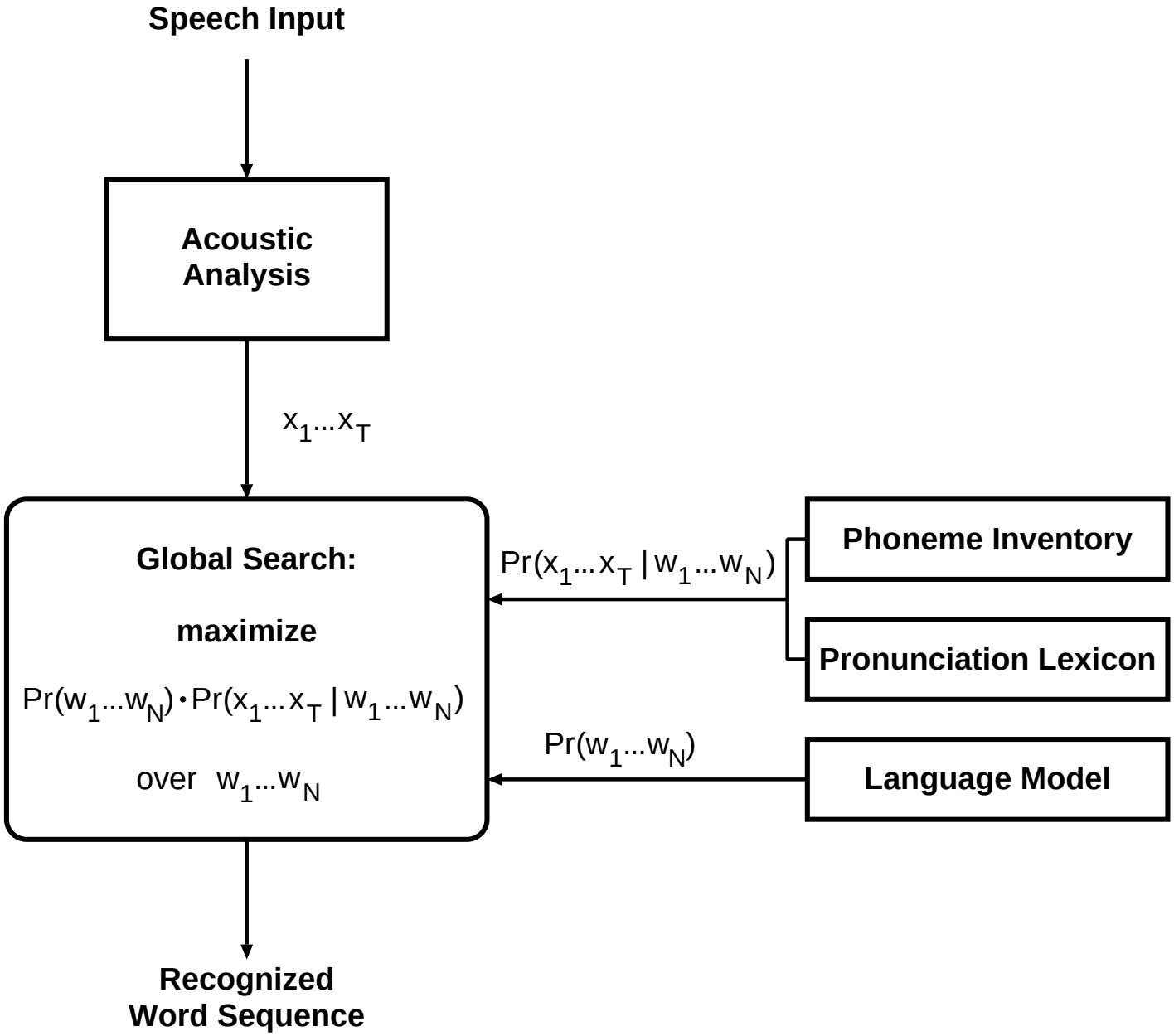
4.6 Excursion (for experts): Length Modelling 269

4.1 Overview: Architecture

starting point: Bayes decision rule

which, under certain conditions, results in a minimum number of recognition errors

more details: see course on pattern recognition


Speech Input
Acoustic Analysis
$x_1 \ldots x_T$
Global Search:
maximize
$Pr(w_1 \ldots w_N) \cdot Pr(x_1 \ldots x_T \mid w_1 \ldots w_N)$
over $w_1 \ldots w_N$
$Pr(x_1 \ldots x_T \mid w_1 \ldots w_N)$
Phoneme Inventory
Pronunciation Lexicon
$Pr(w_1 \ldots w_N)$
Language Model
Recognized Word Sequence

Why does a recognition system make errors?

Reasons from the viewpoint of Bayes' decision rule:

- **incorrect acoustic model:**
  - **poor acoustic analysis**
  - **poor phoneme models**
  - **poor pronunciation model**
- **incorrect language model**
- **incorrect search procedure:**
  the maximum is not found

Importance of integrating higher level knowledge in the search process

Test results on the Wall Street Journal 5k task:

knowledge sources used	perplexity PP	phoneme error rate [%]	word error rate [%]
unconstrained phoneme recognition	–	36.3	—
+ pronunciation lexicon	5000	13.9	40.0
+ LM: unigram	746	8.4	22.9
bigram	107	2.8	6.9
trigram	56	1.9	4.5

Example from the Wall Street Journal 5k task:

LM	recognized	errors
no lexicon	k t k t dh ey d v eh d ey n ey ih z n un k oh sh h ee ey d ih ng n dh uh dh s ey l uh f s ur n d h aa s dh aa t s UH b dh uh b r oh k r ih j y ooh n ih t p p	28
0-gram	h ih t s eh n uh t ur z n ih g oh sh ee ey t ih ng — — s ey l — — s ur t un aa s eh t s aw n t uh b r oh k ur ih j y ooh n ih t s	11
	HIT SENATORS — — NEGOTIATING — SALE — CERTAIN ASSETS ONTO — BROKERAGE UNIT'S	9
1-gram	ih t s s eh n ih t ih z n ih g oh sh ee ey t ih ng — — s ey l — — s ur t un aa s eh t s aw v dh uh b r oh k ur ih j y ooh n ih t	6
	ITS SENATE — IS NEGOTIATING — SALE — CERTAIN ASSETS OF THE BROKERAGE UNIT	5
2-gram	ih t s eh d ih t ih z n ih g oh sh ee ey t ih ng dh uh s ey l aw v s ur t un aa s eh t s aw v dh uh b r oh k ur ih j y ooh n ih t	0
	IT SAID IT IS NEGOTIATING THE SALE OF CERTAIN ASSETS OF THE BROKERAGE UNIT	0

4.2 Phoneme Models and Subword Units

For large vocabularies, it is prohibitive to use whole-word models for each word of the vocabulary:

- There are not enough training samples for each word
- The memory requirements increase linearly with the number of words
  (today: no real problem)

Solution: create word models by concatenating sub-word units,
such as phonemes, context dependent phonemes, demi-syllables, syllables, ...

Advantages:

- Training data is shared between words.
- Words not seen in training (i.e. without training examples) can be recognized by using a pronunciation lexicon.

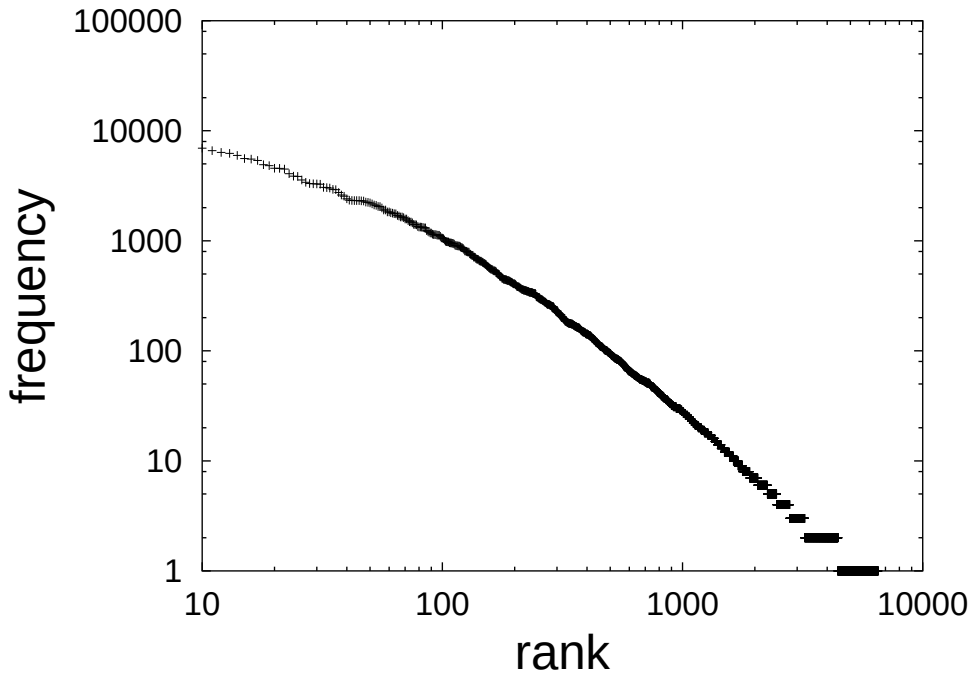
The problem of sparse data is related to "Zipf's law":

> **The frequency $N(w)$ of a word w is (approximately) inversely proportional to some power γ of its rank $r(w)$.**

$$N(w) = \text{const} \cdot r(w)^{-\gamma}$$

Example from the Verbmobil corpus:

rank	word	frequency
1	ich	18648
2	ja	16613
3	das	14288
4	wir	13532
	...	
4440	Abendtermine	1
4441	Aberglaubens	1
	...	
10000	zwingend	0



Distinguish the various levels:
– acoustic realization: acoustic signal
– class of equivalent sounds: phone (allophone, triphone)
– (more) abstract level: phoneme

Speech sounds may be categorized according to different 'features':

for consonants

- **voiced / voiceless**
- **manner of articulation**
  stop, nasal, fricative, approximant
- **place of articulation**
  labial, dental, alveolar, palatal,
  velar, glottal

for vowels

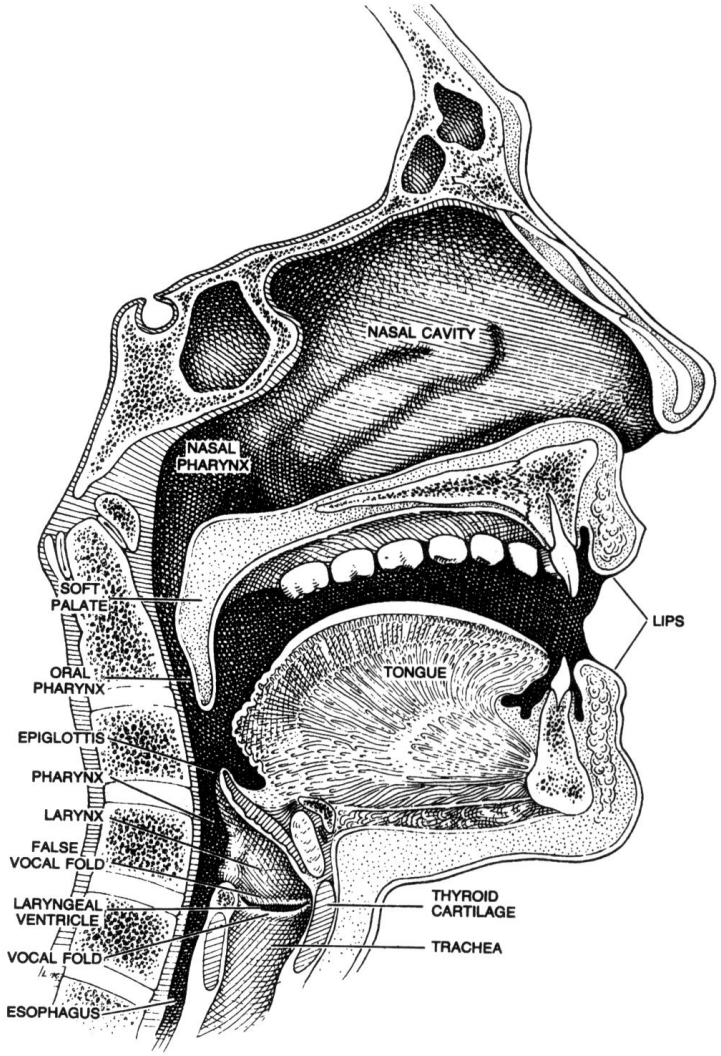
- **position of tongue:**
  high/low, front/back
- **rounded or not**

speech	$\Longleftrightarrow$	temporal sequence of sounds
$\Downarrow$		$\Downarrow$
acoustic signal	$\Longleftrightarrow$	temporal sequence of acoustic vectors, (acoustic realization of the sounds)

Model of speech production:

- **Every sound has a program for the movements of the vocal tract.**
- **Movements of individual sounds merge into one continuous sequence of movements.**
- **Ideal the positioning of the vocal tract is only approximated (depending on the amount of coarticulation).**
  $\Rightarrow$ the real acoustic signal differs from the 'ideal' signal

The Vocal Tract



Drawing by Laszlo Kubinyi ©Scientific American 1977

Criterions for sound classification

- type of articulation (fricative, plosive)
- location of articulation ([p]: labial, [s]: dental)
- consonants and vowels
- voiced and unvoiced sounds
- stationary and non stationary sounds
  (vowels vs. diphtongs, plosives)

Perception of sounds:

- loudness
- tone (smoothed spectrum = formant spectrum)
- unvoiced, voiced (fundamental frequency, pitch)

The pronunciation of a word is usually described in a less detailed way using phonemes.
A phoneme is an abstraction over different phonetic realizations.

Two sounds correspond to different phonemes if they can occur in the same context and distinguish different words.

The phoneme inventory of a language can be inferred from "minimal pairs". A minimal pair is a pair of words whose phonetic transcriptions have an edit distance of one.

Examples of minimal pairs for German:

i: / o:	Kiel	/ Kohl	p / b	packe	/ backe
I / E	fit	/ fett	t / m	Tasse	/ Masse
e: / Y	fehle	/ fülle	k / ts	Kahn	/ Zahn
a: / a	Rate	/ Ratte	f / v	Fall	/ Wall
Y / 9	Hülle	/ Hölle	s / S	Bus	/ Busch
o: / aU	roh	/ rau	s / z	Muße	/ Muse
e: / Ẽ:	Tee	/ Teint	l / –	Klette	/ Kette

Characteristics of the phoneme set:

- **The phoneme set is language specific**

Chinese (? Japanese)	**[l] – [r]**	**one phoneme**
Arabic	**[k_i] – [k_u]**	**different phonemes**

- **Humans are trained to distinguish sounds of specific languages.**
- **The acoustic realizations of phonemes are context dependent (coarticulation):**
  - **static dependencies on surrounding phonemes**
  - **dynamic dependency:**
    temporal overlap of the articulation of subsequent phonemes

Phoneme System for German in SAMPA notation

consonants

plosives

p	Pein	p al n
b	Bein	b al n
t	Teich	t al C
d	Deich	d al C
k	Kunst	k U n s t
g	Gunst	g U n s t

fricatives

f	fast	f a s t
v	was	v a s
s	Tasse	t a s @
z	Hase	h a: z @
S	waschen	v a S @ n
Z	Genie	Z e n i:
C	sicher	z i C 6
j	Jahr	j a: 6
x	Buch	b u: x
h	Hand	h a n t

consonants

sonorants

m	mein	m al n
n	nein	n al n
N	Ding	d I N
l	Leim	l al m
R	Reim	R al m

vowels

“checked” (short) vowels

I	Sitz	z I t s
E	Gesetz	g @ z E t s
a	Satz	z a t s
O	Trotz	t r O t s
U	Schutz	S U t s
Y	hübsch	h Y p S
9	plötzlich	p l 9 t z l I C

vowels

“free” (long) vowels

i:	Lied	l i: t
e:	Beet	b e: t
E:	spät	S p E: t
a:	Tat	t a: t
o:	rot	r o: t
u:	Blut	b l u: t
y:	süß	z y: s
2:	blöd	b l 2: t

diphthongs

aI	Eis	aI s
aU	Haus	h aU s
OY	Kreuz	k r OY t s

“schwa” vowels

@	bitte	b I t @
6	besser	b E s 6

Function of the phonemes:

acoustic signal	**continuous,** **infinite number of realizations**
$\Uparrow$ **(1:∞)**	
(allo-)phones	**discrete sounds, approx. 40 000**
$\Uparrow$ **(1:1000)**	
phonemes	**discrete, alphabet: 40 – 60,** **depending on the language**
$\Updownarrow$ **(1:1)**	
words of the language: **pronunciation and meaning**	**several 100 000 words**

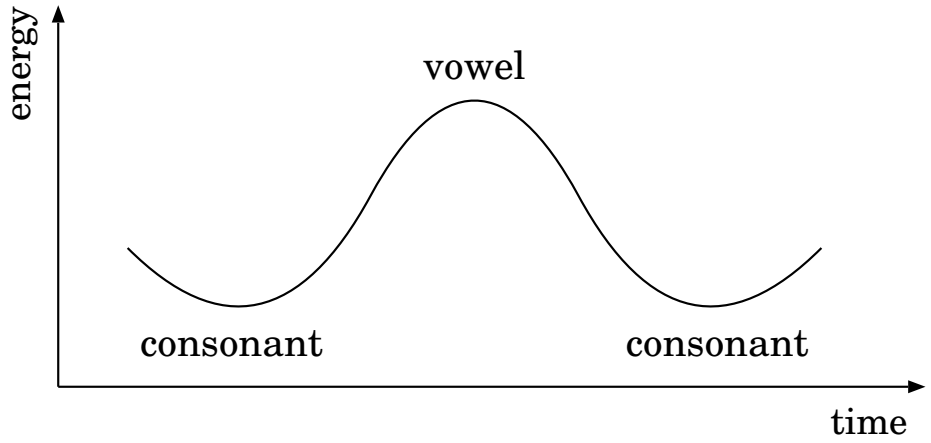
The acoustic realization of phonemes is context dependent.

Context dependent modelling is more accurate:

- **Diphones**

$$\begin{array}{ccccc} A & B & C & D & E \\ \#A \mid & AB \mid & BC \mid & CD \mid & DE \mid E\# \end{array}$$

- **Syllables**
  group of phonemes, standard form consonant–vowel–consonant
  about 20000 syllables for German



- **Demi syllables (syllables split at the vowel)**
- **Consonant clusters**

Example: possible subword units for German

Subword Unit	Number (approx.)	Representation of the acoustic signal
phonemes	50	inaccurate
consonant clusters and vowels	250	... ...
diphones	2450	...
demi-syllables	...	...
syllables	20 000	accurate

note terminology: consonant cluster = consonant sequence

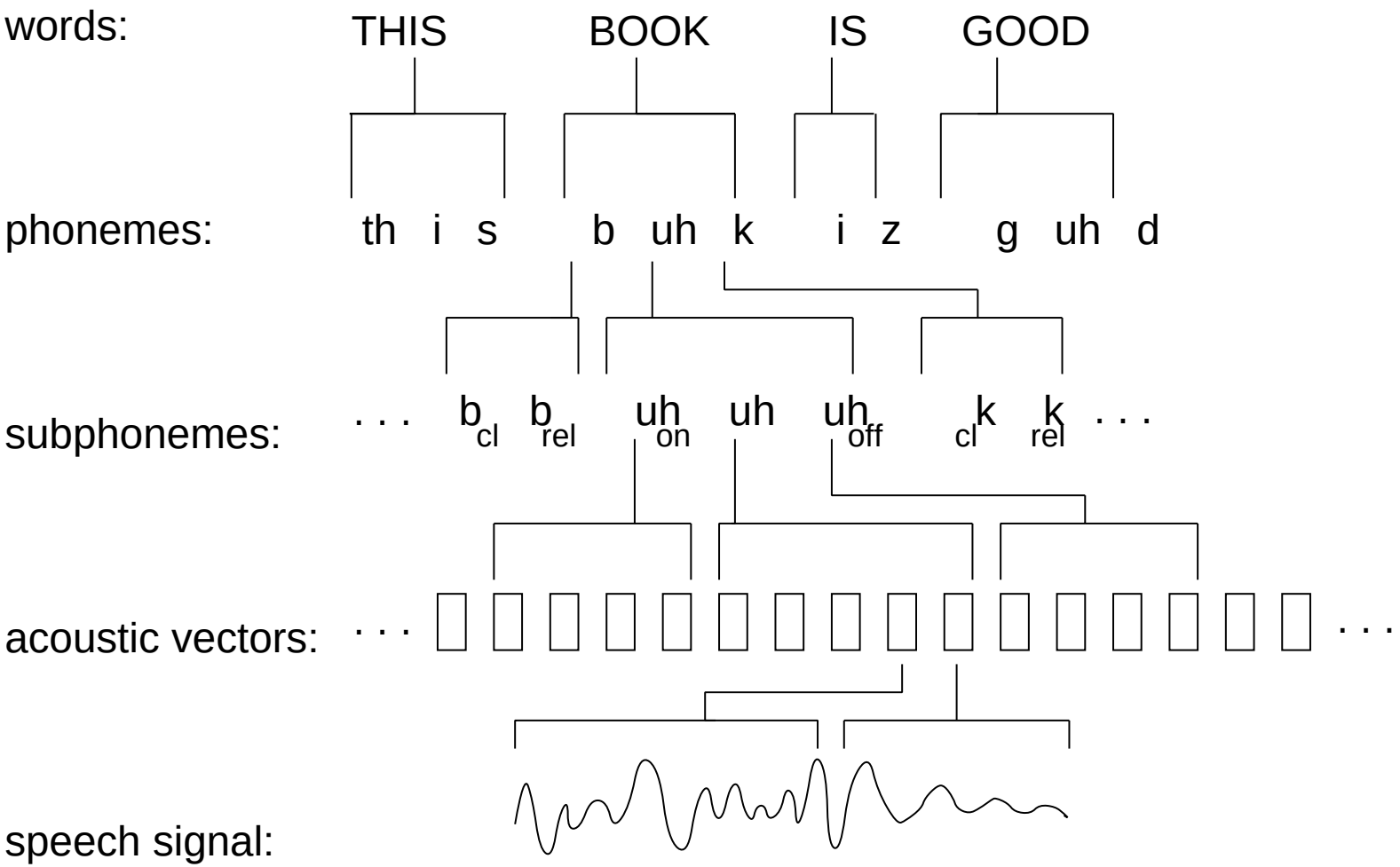
Practical reasons for using subword units in speech recognition:

- **Not enough training data for whole word models.**
- **More observations for subword units (better training).**
- **Vocabulary can be extended without new acoustic training.**
  Specifying the corresponding subword units is sufficient.

Important issues when using subword units:

- **Define and specify subword units.**
- **Map the continuous signal to the discrete sequence of units**
  i.e. specify the units and the pronunciation lexicon.
- **Train the subword units.**
- **Use the subword units and the pronunciation lexicon for recognition.**

Layers of the acoustic modelling



Speech can be modeled on any of these layers

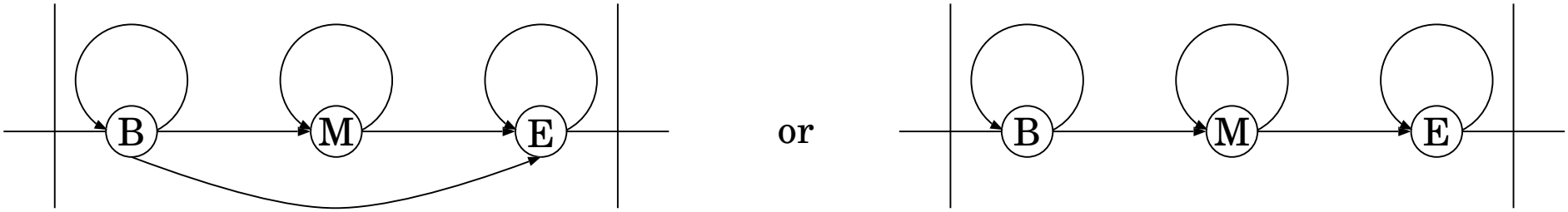
Different HMM topologies for phonemes can be used, define

- **number of states and**
- **allowed transitions.**

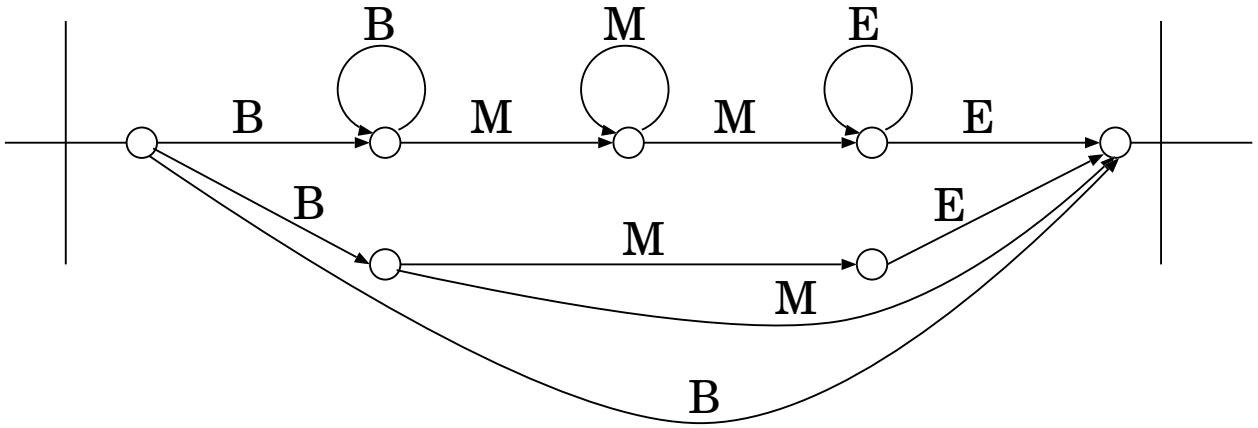
Usually three sub phonemes are used:

Begin – Middle – End

3 state model



"IBM model"



Properties:

- **transition assigned emissions: the emission probability distributions are assigned to the transitions (not to the states).**
- **the number of possible paths is restricted for short vector sequences:**

  1: B

  2: BM

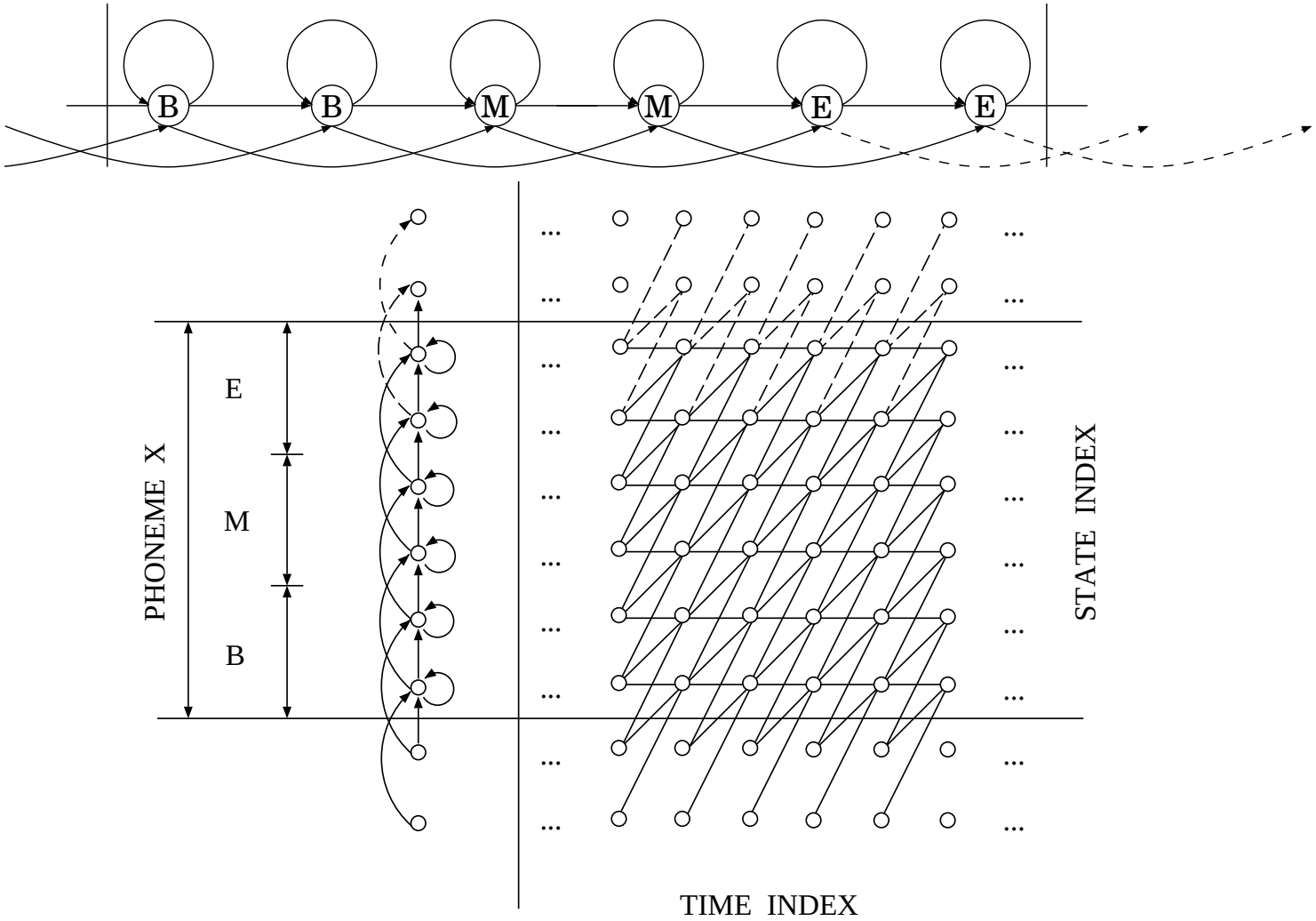
  3: BME

  4: $BMME$

  5: $BBMME$, $BMMEE$, $BMMME$

  6: ...

- **6 state model**



A pronunciation lexicon with phonetic transcriptions is required when using subword units.

Usually phonemes are used as subword units.

Example: English digits

word	phonemes
Zero	Z IH R OW
One	W AH N
Two	T UW
Three	TH R IY
Four	F OW R
Five	F AY V
Six	S IH K S
Seven	S EH V AX N
Eight	EY T
Nine	N AY N
Oh	OW

phoneme	number of occurances
AH	1
AX	1
AY	2
F	2
EH	1
EY	1
IH	2
IY	1
K	1
N	4
OW	3
S	3
R	2
T	2
TH	1
UW	1
V	2
W	1
Z	1

Context dependencies:

- Context independent (real) phoneme models:
  the international phonetic alphabet defines 74 phonemes for English.
  in practical applications about 40–50 phonemes are used typically.

$$\begin{array}{cccc} z & e & r & o \\ Z & IH & R & OW \end{array}$$

- Context dependent phoneme models
  coarticulation is considered

$$\begin{array}{cccc} z & e & r & o \\ {}_{\#}Z_{IH} & {}_{Z}IH_{R} & {}_{IH}R_{OW} & {}_{R}OW_{\#} \end{array}$$

  - "Diphone" ${}_{A}B, B_{C}$: context dependent phoneme in diphone context
  - "Triphone" ${}_{A}B_{C}$: context dependent phoneme in triphone context

Terminology:

- **context independent phonemes ('monophones', more or less the 'real' phonemes as defined in linguistics)**
- **phonemes in (left or right) diphone context ('diphones')**
- **phonemes in triphone context ('triphones')**
- **phonemes in word context ('wordphones')**

The context dependency only determines the labels for emission probabilities which have to be specified for each state of a phoneme model.

The emission probabilities can be trained independent from the phonetic context.

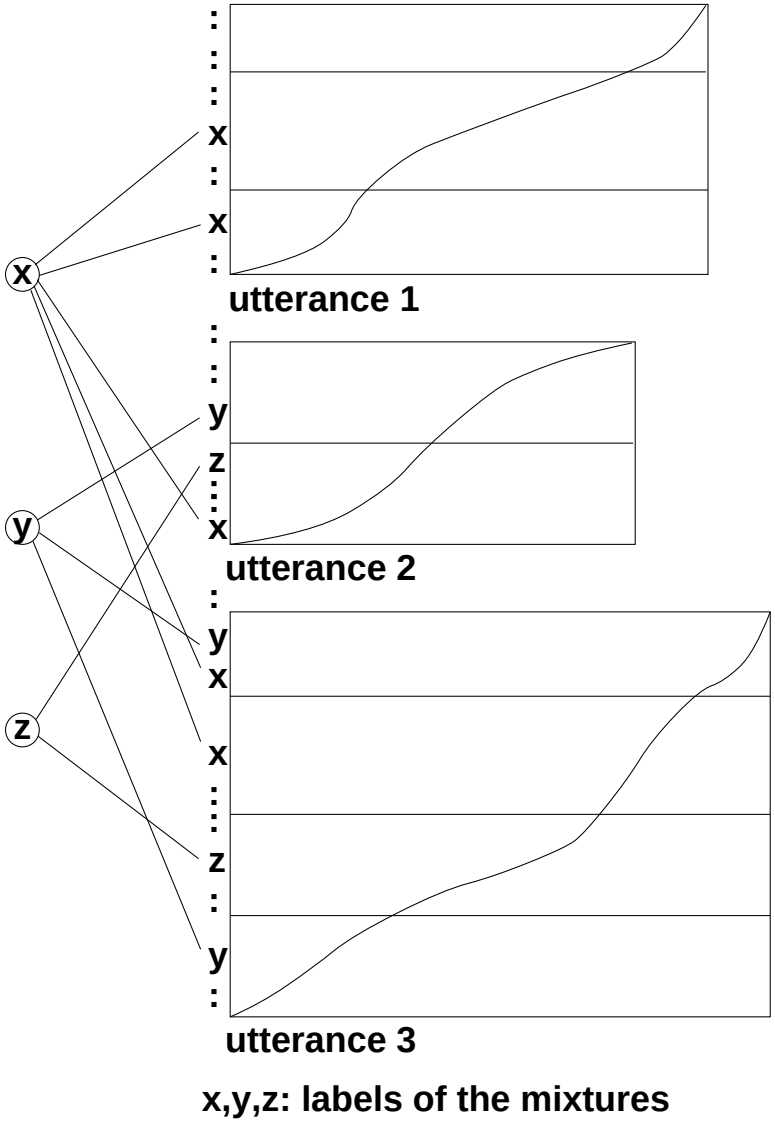
The sequence of states is used for recognition:

- **word: sequence of phoneme models,**
- **phoneme model: sequence of HMM states**

  $\Rightarrow$ **word: sequence of HMM states.**

Training phoneme models:

- The sequence of HMM state indices for a word depend on the phoneme sequence.
- The training procedure corresponds to the one used for word models:
  - Time alignment:
    assign a state to every acoustic vector
  - Parameter estimation:
    . Collect all observations for every mixture m of every phoneme model based on the time alignment.
    . Estimate the model parameters for all densities l of the mixture m:
    - Reference or prototype vector μ_{lm}
    - pooled variance vector σ_m^2
    - mixture weight $p(l|m)$



Practical considerations:

All possible triphones $50^3 = 125000$ are too many!

- **Combine monophones, diphones and triphones:**
  only use diphones and triphones that occur more than e.g. 100 times in the training data.
- **Generalized context dependent phoneme models:**
  use phoneme classes (nasals, fricatives, vowels, stop consonants, ...)
  $_{g(A)}B_{g(C)}$ instead of $_{A}B_{C}$, $g(X)$ phoneme class X belongs to
- **Parameter tying:**
  use clustering or Classification And Regression Trees (CART) to tie "similar" phonemes

$\Rightarrow$ 1000–2000 models that are actually used

4.3 Language Modelling

goal: model syntax and semantics of natural language (spoken or written)

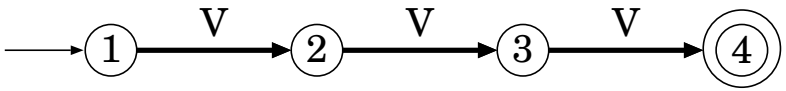
needed in automatic systems
that process speech (= spoken language) or language (= written language):

- **speech recognition**
- **speech and text translation**
- **(spoken and written) language understanding**
- **spoken dialog systems**
- **text summarization**
- **...**

Example: finite state networks for digit strings

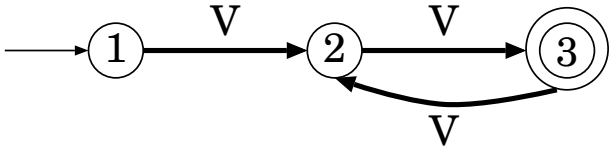
syntactical constraints can be expressed with formal grammars, here represented as networks

- **String of three digits**



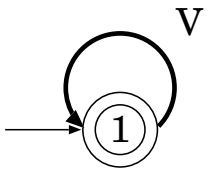
V ε {zero, one, two, three, . . . ,nine}

- **String with even number of digits**



V ε {zero, one, two, three, . . . ,nine}

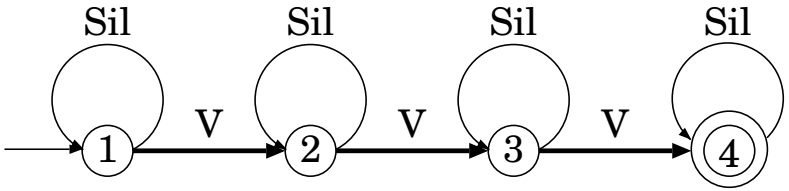
- **Unconstrained digit string**



V ε {zero, one, two, three, . . . ,nine}

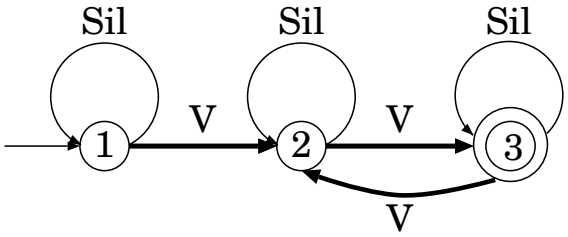
Allow silence between the words:

- **String of three digits**



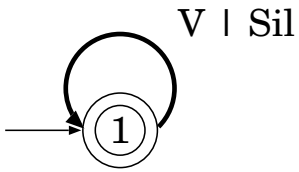
V ε {zero, one, two, three, . . . ,nine}

- **String with even number of digits**



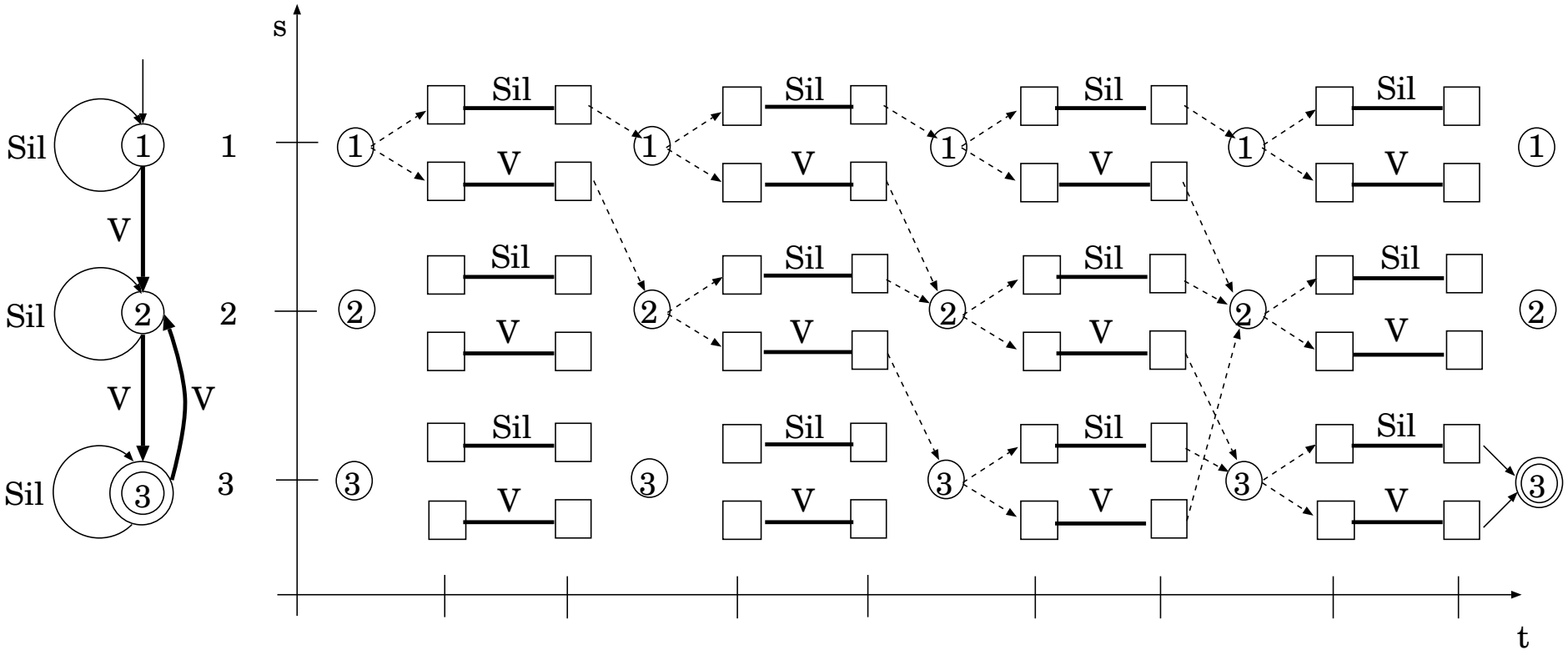
V ε {zero, one, two, three, . . . ,nine}

- **Unconstrained digit string**



V ε {zero, one, two, three, . . . ,nine}

For the recognition the network has to be unfolded along the time axis:



The computational complexity is proportional to the acoustic transitions.

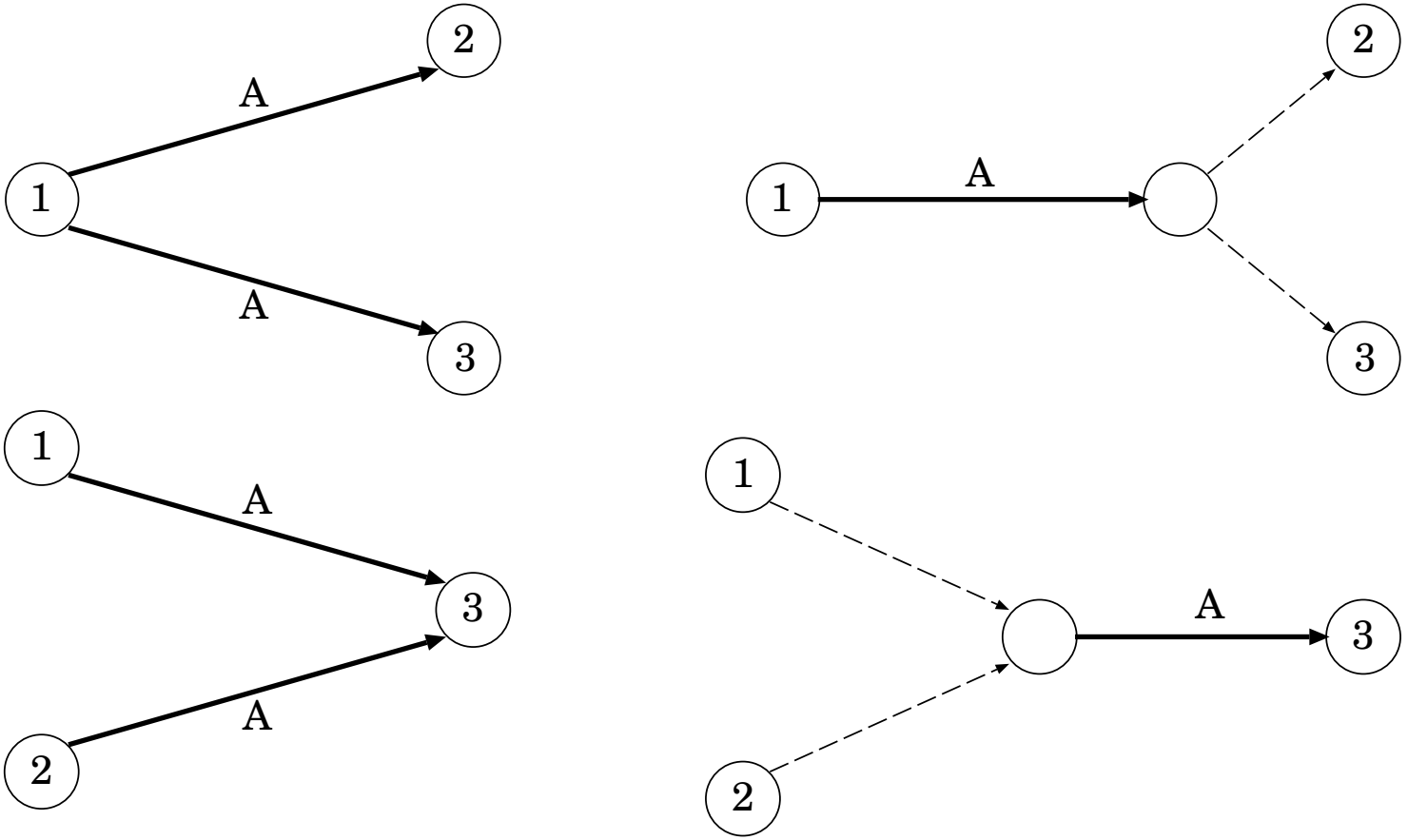
As shown later it is favorable to reduce the number of "real" i.e. acoustic transitions.

As shown in the example a network consists of transitions and nodes.

- **Transitions:**
  correspond to spoken words (including silence)
- **Nodes: Every word has a start and end node, they define the syntactic (linguistic) context of the transition.**

As shown on the next page a word A can occur in four different contexts.

Possible contexts for a word A:



In case a) the automaton is non deterministic.

Bayes decision rule and und perplexity

- Bayes decision rule using maximum approximation:

$$\left[w_1^N\right]_{opt} = \underset{[w_1^N]}{\operatorname{argmax}} \left\{ Pr(w_1^N) \cdot \max_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t | s_{t-1}, w_1^N) \right\}$$

- The perplexity (corpus perplexity / test perplexity) of a language model and a test corpus $[w_1^N]$ is defined as

$$\begin{aligned} PP &= Pr(w_1^N)^{-\frac{1}{N}} \\ &= \left[\prod_{n=1}^{N} Pr(w_n | w_1^{n-1}) \right]^{-\frac{1}{N}} \end{aligned}$$

- The logarithm of the perplexity then is:

$$\begin{aligned}\log PP &= \log\left[Pr(w_1^N)^{-\frac{1}{N}}\right] \\ &= -\frac{1}{N}\sum_{n=1}^{N}\log(Pr(w_n|w_1^{n-1}))\end{aligned}$$

A small perplexity corresponds to strong language model restrictions.

Properties of the perplexity:

- normalization: probability per word
- inverse probability: number of possible choices per word position
- probability zero: infinite penalty

Given constant probabilities (no dependence on the word and between words) the perplexity is:

$$\begin{aligned}Pr(w_1^N) &= \left(\frac{1}{W}\right)^N \\ \Rightarrow \quad PP &= W \\ \text{with } W &= \text{size of the vocabulary}\end{aligned}$$

Language model $Pr(w_1^N)$ in networks

a Deterministic Finite state Automaton (DFA) is defined by a transition function δ

$$\begin{aligned} \mathcal{V} &= \{1, \ldots, V\} && \text{nodes (linguistic contexts)} \\ \mathcal{W} &= \{1, \ldots, W\} && \text{arcs (words including silence)} \\ \delta: & && \mathcal{V} \times \mathcal{W} \rightarrow \mathcal{V} \\ & && (v, w) \rightarrow v' = \delta(v, w) \end{aligned}$$

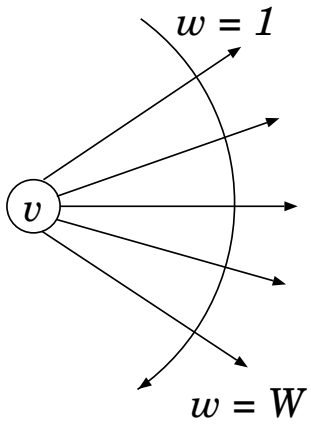
A path in the network defines a word sequence $[w_1^N]$.

For every word w given a node v a probability $p(w|v)$ is defined:

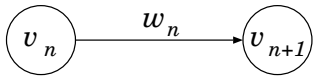
$$p(w|v) = \begin{cases} 0 & \text{if word } w \text{ does not} \\ & \text{leave node } v \\ \leq 1 & \text{else} \end{cases}$$

The sum of the probabilities $p(w|v)$ over all words w is:

$$\sum_{w=1}^{W} p(w|v) = 1$$



Index convention: $v_{n+1} := \delta(v_n, w_n)$



In the general case of Non deterministic Finite state Automata (NFA) the sum over all paths corresponding to a word sequence w_1^N has to be calculated:

$$Pr(w_1^N) = \sum_{v_1^N} \left\{ \prod_{n=1}^{N} p(w_n|v_n) \right\} .$$

Often the maximum approximation is used:

$$\begin{aligned} Pr(w_1^N) &\approx \max_{v_1^N} \left\{ \prod_{n=1}^{N} p(w_n|v_n) \right\} \\ &= \max_{v_1^N} \left\{ \prod_{n=1}^{N} p(w_n|\delta(v_{n-1}, w_{n-1})) \right\} \end{aligned}$$

Note the hierarchical structure of the grammars:

- HMM: acoustic modelling for each word defining the correspondence between word classes and acoustic vectors
- LM: network

Non deterministic and Deterministic Finite state Automata (NFA and DFA):

- **general case NFA:**

$$p(w, v|v') = \underbrace{p(v|v')}_{\text{Transitionprob.}} \cdot \underbrace{p(w|v', v)}_{\text{Emissionprob.}}$$

- **special case DFA: given a pair (v', w) the successor state v is determined by $v = \delta(v', w)$, a different factorization of $p(w, v|v')$ is useful:**

$$p(w, v|v') = p(w|v') \cdot p(v|v', w)$$

with

$$p(v|v', w) = \begin{cases} 1 & v = \delta(v', w) \\ 0 & v \neq \delta(v', w) \end{cases}$$

For an allowed transition $(v', w) \rightarrow v = \delta(v', w)$ the probability is:

$$p(w, v|v') = p(w|v')$$

Dynamic programming recursion for language model networks:

Here the auxiliary quantity $Q_v(t, s; w)$ is defined as:

$$Q_v(t, s; w) := \text{probability of the best path at time } t \text{ leading to the state } s \text{ of word } w \text{ with starting node } v.$$

Note the additional index v.

Dynamic Programming Recursion

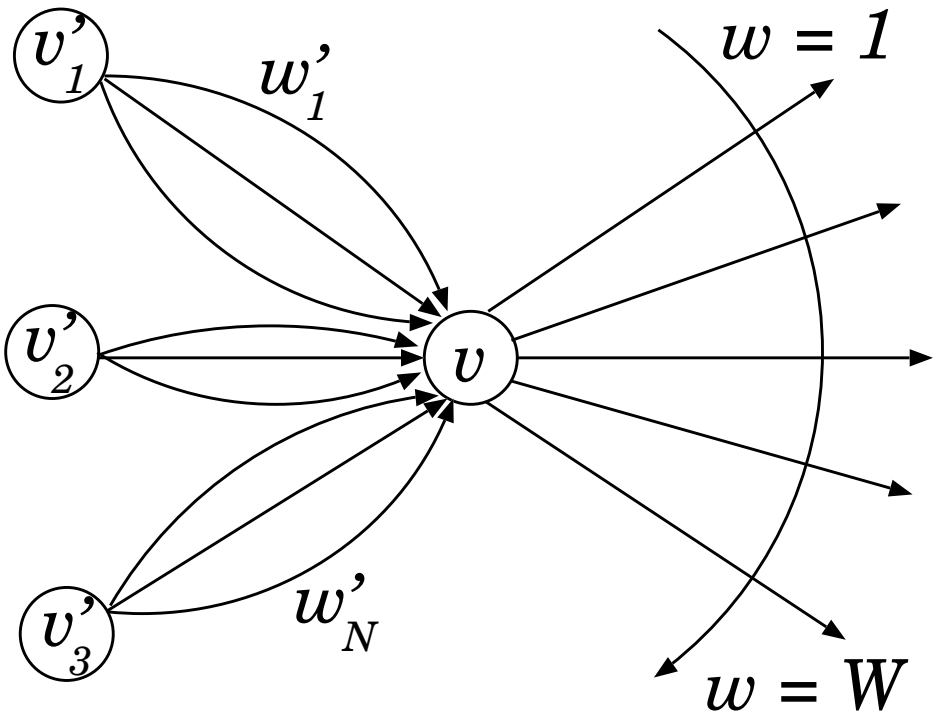
- **Within words: acoustic search**

$$
\begin{aligned}
Q_v(t,s;w) &= \max_{s'}\left\{Q_v(t-1,s';w)\cdot p(x_t,s|s',w)\right\} \\
\sigma_v^{opt}(t,s;w) &:= \operatorname*{argmax}_{s'}\left\{Q_v(t-1,s';w)\cdot p(x_t,s|s',w)\right\} \\
B_v(t,s;w) &= B_v(t-1,\sigma_v^{opt}(t,s;w);w)
\end{aligned}
$$

- **Word boundaries: language model recombination**

$$
\begin{aligned}
Q_v(t-1,0;w) &= \max_{\{v',w':\delta(v',w')=v\}}\left\{Q_{v'}(t-1,S(w');w')\cdot p(w|v)\right\} \\
&= p(w|v)\cdot\max_{\{v',w':\delta(v',w')=v\}}\left\{Q_{v'}(t-1,S(w');w')\right\} \\
B_v(t-1,0,w) &= t-1
\end{aligned}
$$

Word boundaries: language model recombination



The dynamic programming recursion is carried out for every word w and node v. The context defined by v has to be considered in the traceback arrays:

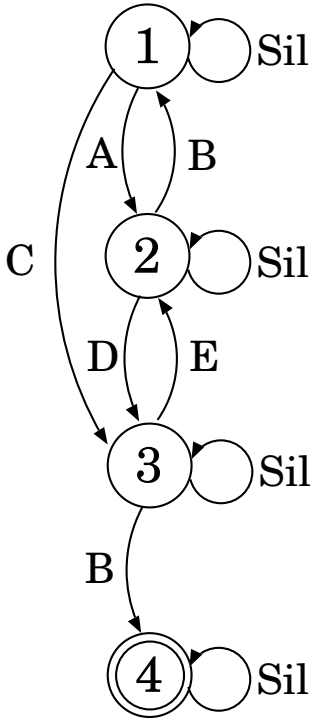
$$\text{Score:} \quad H(v,t) = \max_{\{v',w':\delta(v',w')=v\}} Q_v(t, S(w'); w')$$

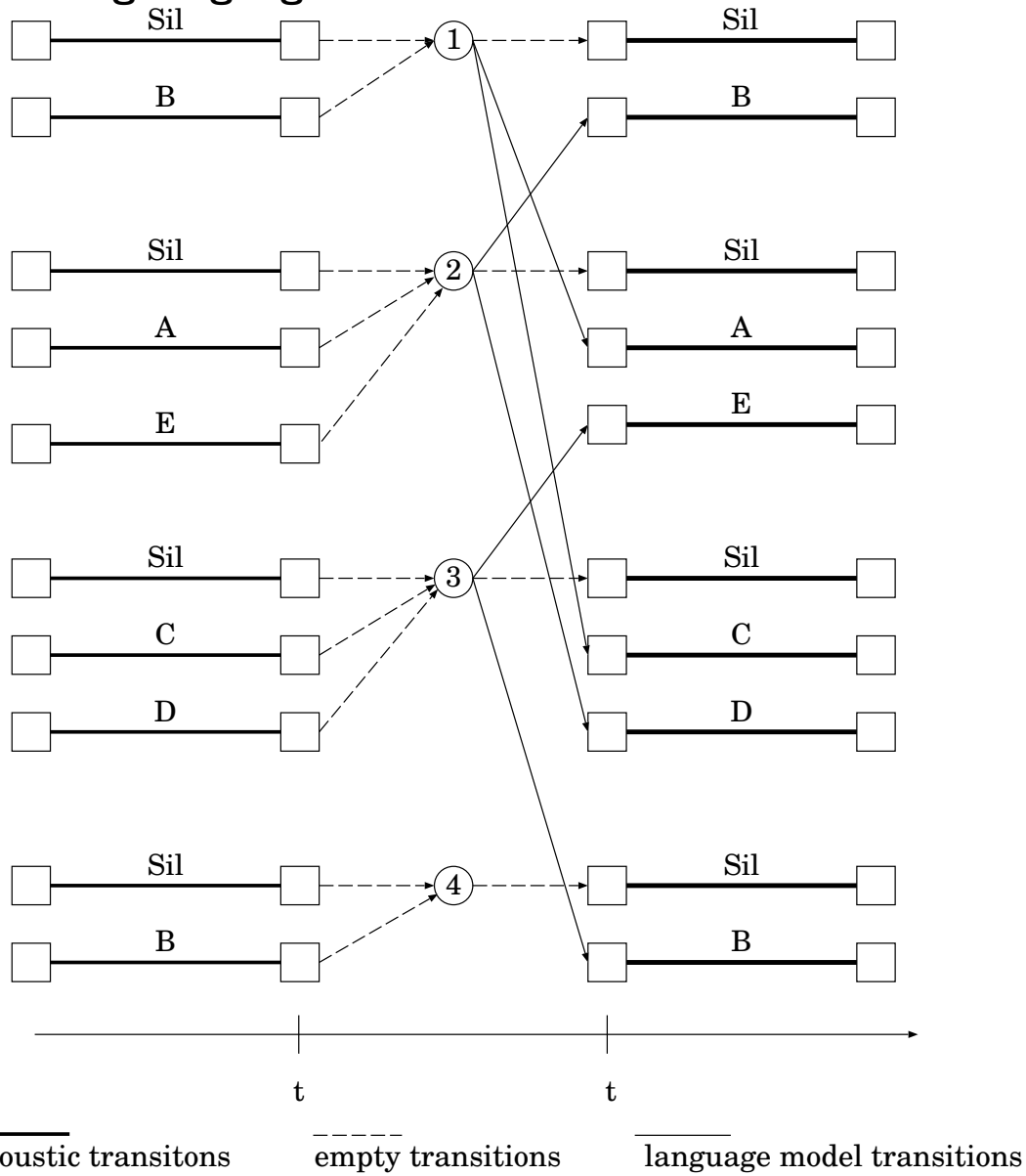
$$\text{Starting node, word:} \quad (V,W)(v,t) = \operatorname*{argmax}_{\{v',w':\delta(v',w')=v\}} Q_v(t, S(w'); w')$$

$$\text{Backpointer:} \quad B(v,t) = B(t, S(W(v,t)); W(v,t))$$

The index pair (predecessor node, word) is stored in the traceback array $(V,W)(v,t)$.
It can be interpreted as linguistic copy of word w in the context v.

Example: language model network and corresponding language model transitions





Factorization without restrictions
($w \in \mathcal{W} \cup \{\$\}$; $\$\hat{=}$sentence end):

$$Pr(w_1^N) = \prod_{n=1}^{N} Pr(w_n|w_1^{n-1})$$

Limit the dependence:

$$\text{Unigram LM :} \quad Pr(w_1^N) = \prod_{n=1}^{N} p(w_n)$$

$$\text{Position Unigram LM :} \quad Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|n)$$

$$\text{Bigram LM :} \quad Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1})$$

$$\text{Trigram LM :} \quad Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-2}, w_{n-1})$$

Training bigram language models:

Count words and word pairs:

$$p(w|v) = \frac{N(v,w)}{N(v)}$$

$N(v,w)$: **word pair count** (v,w) **in the training text**

$N(v)$: **count for word** v

Motivation:

Bigram probability:

$$Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1})$$

maximize log-likelihood function:

$$F = \sum_{n=1}^{N} \log p(w_n|w_{n-1}) \qquad \text{with} \quad \sum_w p(w|v) = 1 \quad \forall \quad v$$

$$F = \sum_{v,w} N(v,w) \log p(w|v) - \sum_v \mu_v \left[\sum_w p(w|v) - 1 \right]$$

Differentiate by $p(w|v)$ and μ_v:

$$\frac{\partial F}{\partial p(w|v)} = \frac{N(v,w)}{p(w|v)} - \mu_v \stackrel{!}{=} 0$$

$$\frac{\partial F}{\partial \mu_v} = \sum_w p(w|v) - 1 \stackrel{!}{=} 0$$

Solve:

$$p(w|v) = \frac{N(v,w)}{\sum_{w'} N(v,w')} = \frac{N(v,w)}{N(v)}$$

Problem:

many pairs (v, w) are not seen in training

$$N(v, w) \quad = \quad 0 \ ,$$

relative frequency is zero.

Discounting: shift probability mass from seen to unseen events

- **Linear discounting:**

$$p(w|v) \quad = \quad \begin{cases} (1 - \lambda) \cdot \dfrac{N(v, w)}{N(v)} & N(v, w) > 0 \\ \lambda \cdot \dfrac{p(w)}{\sum\limits_{w': N(v, w') = 0} p(w')} & N(v, w) = 0 \end{cases}$$

Estimate $0 < \lambda < 1$ by Leaving One Out:
Leave (v, w) out of the corpus
$\rightarrow$ change counts: $N(v, w) \rightarrow N(v, w) - 1$ for $N(v, w) > 1$

$$p_{-1}(w|v) = \begin{cases} (1-\lambda) \cdot \dfrac{N(v,w)-1}{N(v)-1} & N(v,w) > 1 \\ \lambda \cdot \dfrac{p(w)}{\sum\limits_{w':N(v,w')=0} p(w')} & N(v,w) = 1 \end{cases}$$

$$\begin{aligned} F(\lambda) &= \sum_{v,w} N(v,w) \cdot \log p_{-1}(w|v) \\ &= \sum_{v,w:N(v,w)>1} \cdots + \sum_{v,w:N(v,w)=1} \cdots \\ &= (N - n_1) \cdot \log(1-\lambda) + n_1 \cdot \log(\lambda) + const(\lambda) \end{aligned}$$

$$F(\lambda) = (N - n_1) \cdot \log(1 - \lambda) + n_1 \cdot \log(\lambda) + const(\lambda)$$

with
$$n_1 := \sum_{v,w: N(v,w)=1} 1$$
$$\hat{=} \text{ number of bigram singletons}$$
$$N := \text{length of the corpus}$$

Differentiate and set to zero:

$$\lambda = \frac{n_1}{N}$$

- **Absolute discounting**

$$p(w|v) = \begin{cases} \dfrac{N(v,w) - b}{N(v)} & N(v,w) > 0 \\ b \cdot \dfrac{W - W_0(v)}{N(v)} \dfrac{p(w)}{\sum\limits_{w': N(v,w')=0} p(w')} & N(v,w) = 0 \end{cases}$$

W := vocabulary size

$W_0(v)$:= number of words that do not occur as successor of v.

Leaving-One-Out approach with maximum-likelihood estimation:

$$\begin{aligned} F(b) &= n_1 \cdot \log(b) + \sum_{v,w:N(v,w)>1} N(v,w) \log\left(\frac{N(v,w)-1-b}{N(v)-1}\right) \\ &= n_1 \cdot \log(b) + \sum_{r>1} r \cdot n_r \cdot \log(r-1-b) + const(b) \end{aligned}$$

with $n_r := \sum_{v,w:N(v,w)=r} 1$

$\hat{=}$ Number of word pairs seen r times

Differentiate $F(b)$ by b and rewrite:

$$\frac{n_1}{b} - \frac{2n_2}{1-b} = \sum_{r>2} \frac{r n_r}{r-1-b}$$

There is no closed form solution, but the following estimate can be proven:

$$\frac{n_1}{n_1 + 2n_2 + \frac{1}{2}[N - n_1 - 2n_2]} \leq b \leq \frac{n_1}{n_1 + 2n_2}$$

Usually the upper bound is a sufficient estimate

$$b \cong \frac{n_1}{n_1 + 2n_2}$$

For corpora of $10 - 20$ million words and a vocabulary of $10.000 - 20.000$ words $b \approx 0.95$

Result:
A LM where all w_1^N are possible, i.e. $Pr(w_1^N) > 0$.

Ideal case:

typical word sequence w_1^N:	high probability $Pr(w_1^N)$
possible word sequence w_1^N:	low probability $Pr(w_1^N)$
un typical word sequence w_1^N:	very low probability $Pr(w_1^N)$

Using m–grams homophones (phonetically equal words with different spellings) can be distinguished:

Examples from the IBM TANGORA system:

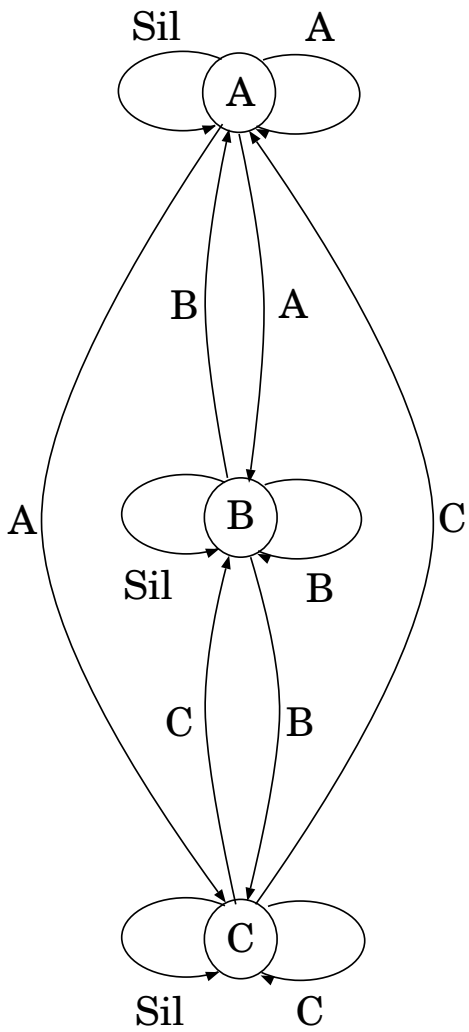
- **To, too, two:**
  Twenty–two people are too many to be put in this room.
- **Right, Wright, write:**
  Please write to Mrs. Wright right away.

A bigram for the three words A, B, and C represented as network.

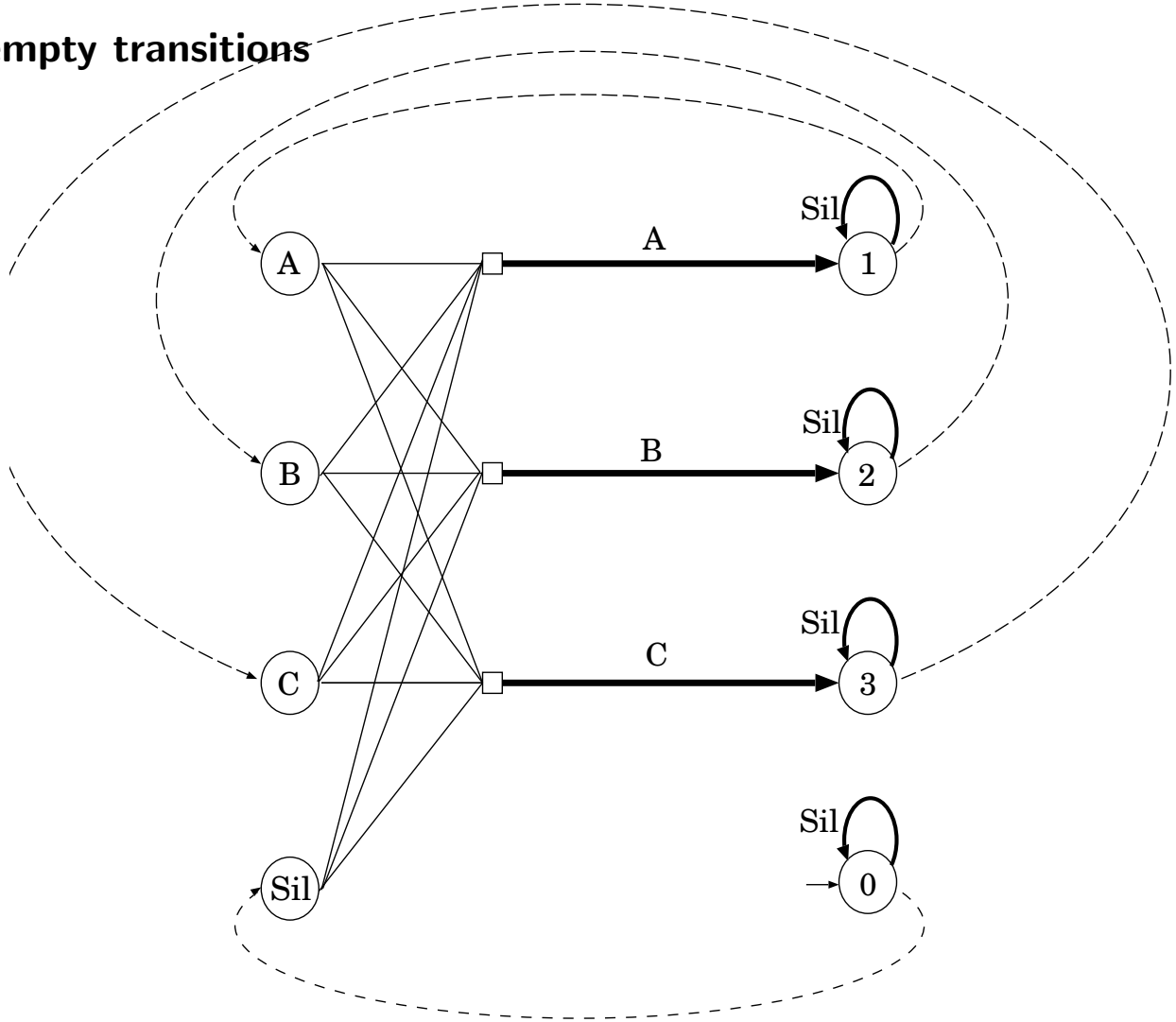
With W words the network has W^2 arcs plus W arcs for silence.

Problem: the computational complexity rises like W^2

Introducing empty transitions can help.

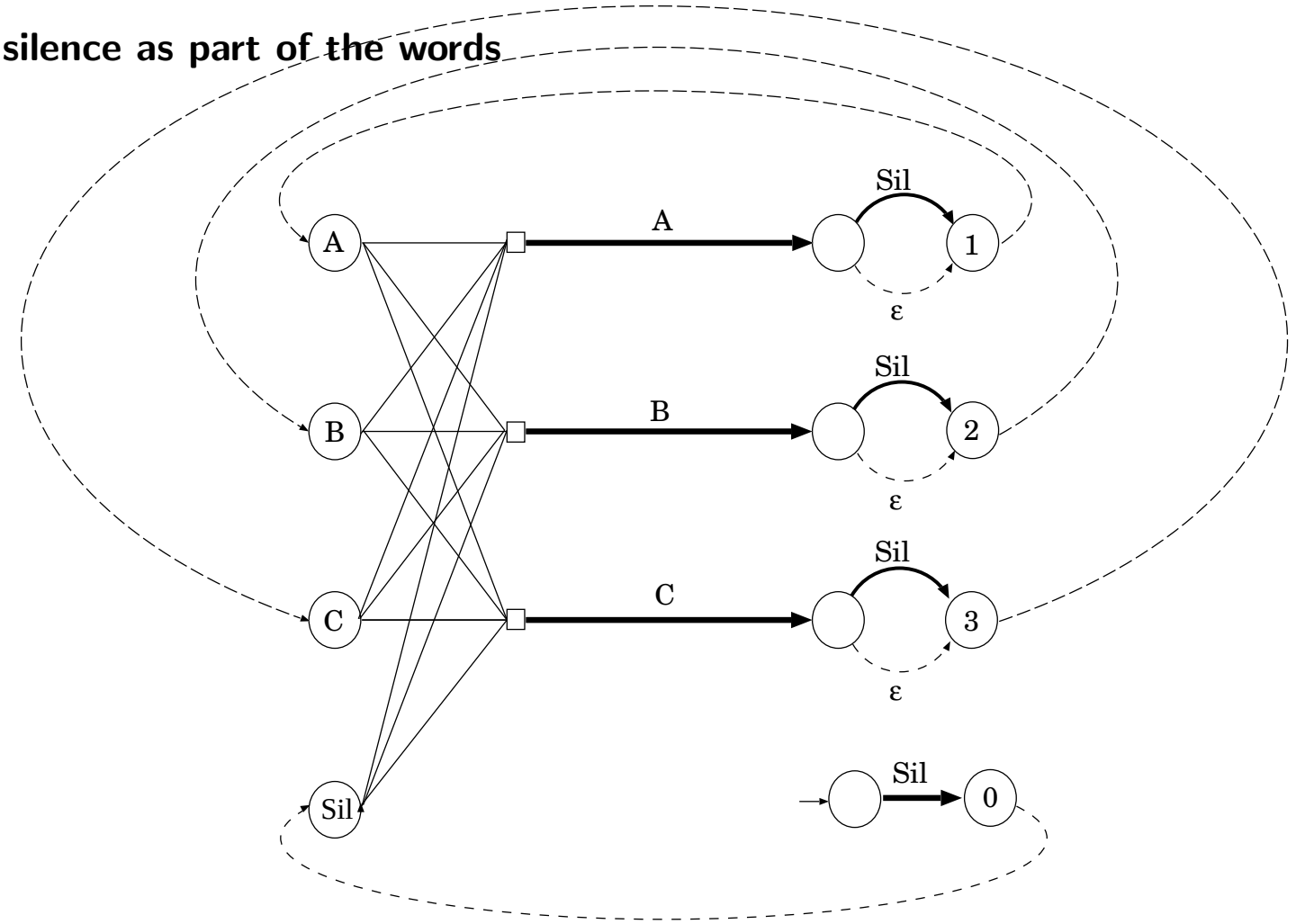


Bigram LM with empty transitions



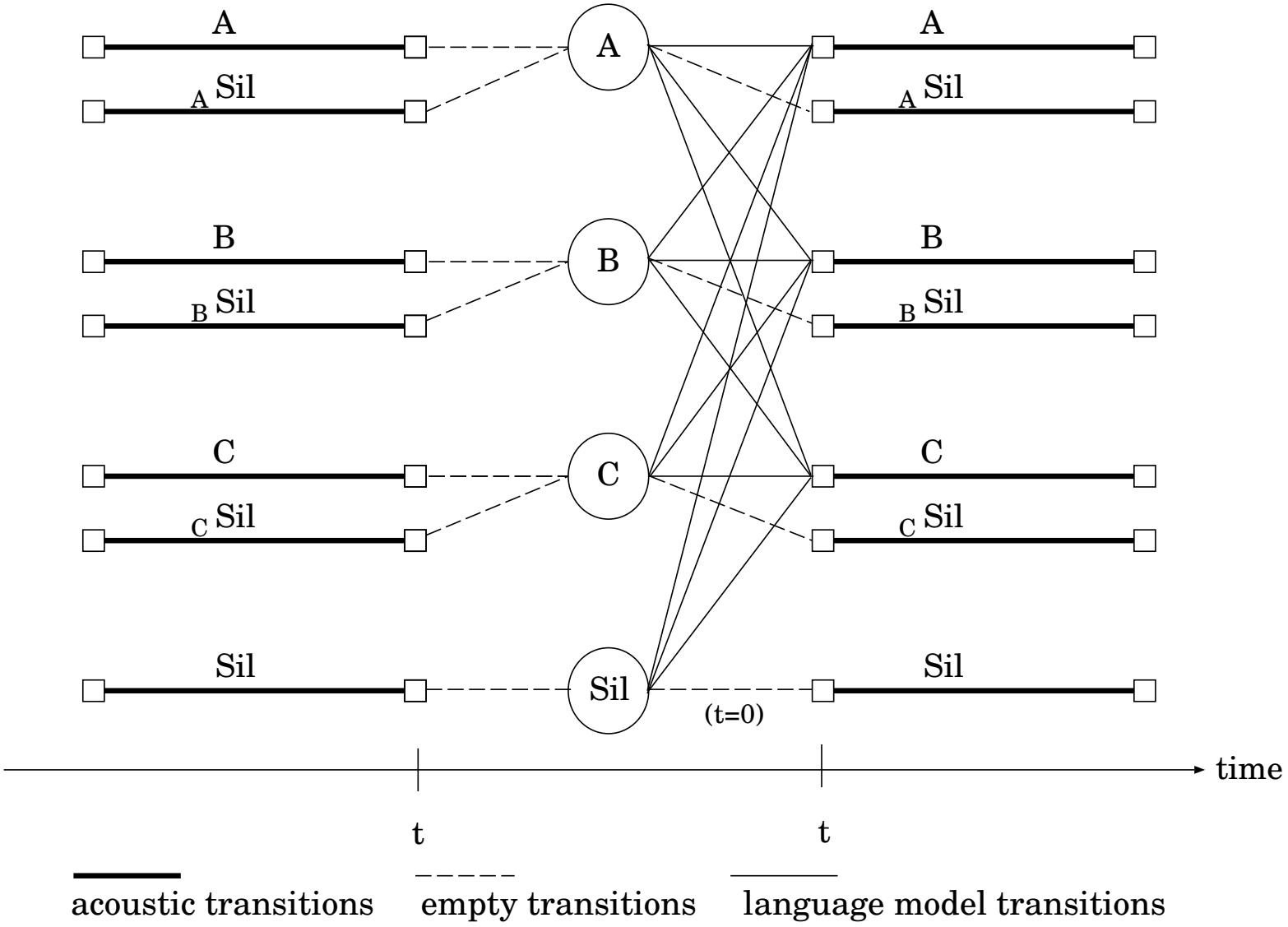
Start node: 0 End node: 1,2,3

Bigram LM: silence as part of the words

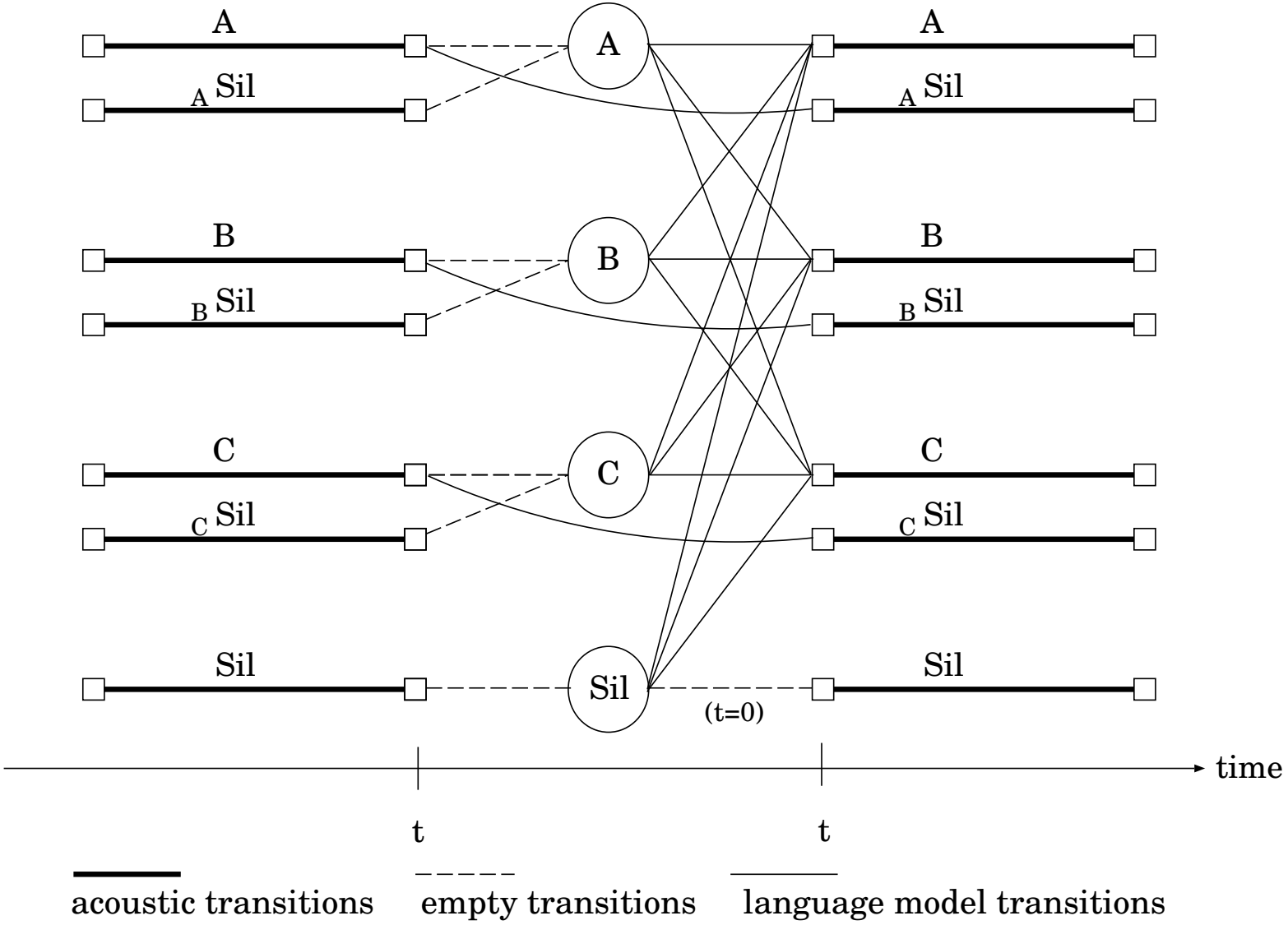


Start node: 0 End node: 1,2,3

Unfolding the bigram over the time:



Unfolding the bigram over the time (silence as part of the words):



Bigram LM in recognition:

The network has the following transitions:

Sil	(silence at the beginning of the sentence)
A	word A
B	word B
:	
:	
${}_A Sil$	(silence after word A)
${}_B Sil$	(silence after word B)
:	

We augment the vocabulary w as follows:

$$w \in \{Sil,\ A,\ B,\ \ldots,\ {}_A Sil,\ {}_B Sil,\ \ldots\}$$

The auxiliary quantity $Q(t, s, w)$ for dynamic programming is defined as:

$Q(t, s, w)$:= probability of the best partial path at time t leading to state s of word w.

The recursion then is:

- Within words:

$$Q(t, s; w) = \max_{s'}\{Q(t-1, s'; w) \cdot p(x_t, s|s', w)\}$$

- Word boundaries:

$$Q(t-1, 0; w) = \max_{v}\{Q(t-1, S(v); v) \cdot p(w|v)\}$$

(The special handling of the silence word is not expressed in the equations)

In principle the probability $p(w|v)$ is the LM,
but silence transitions require special interpretation:

Transition			LM Probability $p(w\|v)$
A	–	B	$p(B\|A)$
A	–	${}_A Sil$	1
${}_A Sil$	–	B	$p(B\|A)$
${}_A Sil$	–	${}_A Sil$	1
Sil	–	B	$p(B)$: unigram
${}_A Sil$	–	${}_B Sil$	0 : not possible

Traceback arrays:

it is easiest to store the decisions about starting words w at time t:

$$\begin{aligned}
\textbf{score:} \quad & H(w,t) &=& \max_{v} \{Q(t, S(v); v) \cdot p(w|v)\} \\
\textbf{predecessor:} \quad & V(w,t) &=& \operatorname*{argmax}_{v} \{Q(t, S(v); v) \cdot p(w|v)\} \\
\textbf{backpointer:} \quad & B(w,t) &=& B(t, S(V(w,t)); V(w,t))
\end{aligned}$$

Remarks:

- Due to the regular structure of the bigram it is sufficient to store either the LM nodes or the predecessor words in the traceback arrays.
- The traceback at the end of the sentence has to start at the word ends not the word beginnings.
- The real implementation is different to optimize memory efficiency:
  - traceback arrays with one index instead of a pair (w, t)
  - when using beam search (following section) the number of word ends reached is smaller, instead of storing word beginnings it is more efficient to store word ends.

Trigram language model:

The trigram language model probability is given by:

$$Pr(w_n|w_1^{n-1}) \quad = \quad p(w_n|w_{n-2}, w_{n-1})$$

Notation: $(u, v, w) \; = \; (w_{n-2}, w_{n-1}, w_n)$

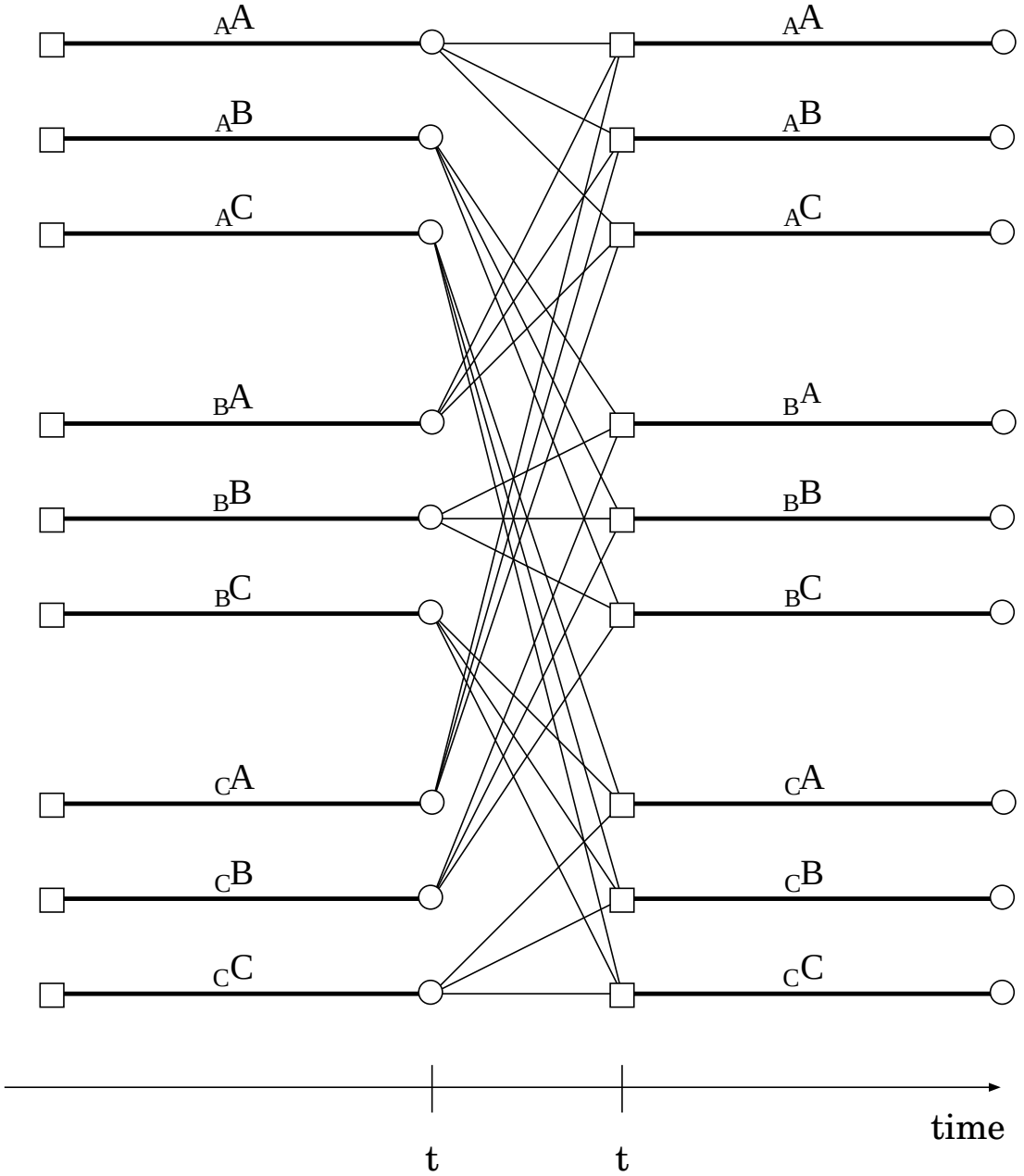
u, v are the predecessor words of w, these have to be considered in the LM recombination. The auxiliary quantity for dynamic programming is defined as:

$Q_v(t, s; w) :=$ **probability of the best path at time t leading to the state s of word w with predecessor word v.**

- **For each word w a copy for every predecessor word v has to be made.**
- **The cost of an arc only depend on the arc itself. This allows the practical implementation of dynamic programming.**

Trigram

LM recombination



The dynamic programming recursion is:

- within words:

$$Q_v(t,s;w) \quad = \quad \max_{s'}\{Q_v(t-1,s';w) \cdot p(x_t,s|s',w)\}$$

- word boundaries:

$$Q_v(t-1,0;w) \quad = \quad \max_{u}\{Q_u(t-1,S(v);v) \cdot p(w|u,v)\}$$

Traceback arrays (at word beginnings):

$$\begin{aligned}
\text{score:} \quad & H(v,w,t) &=& \max_{u}\{Q_u(t,S(v);v) \cdot p(w|u,v)\} \\
\text{predecessor:} \quad & U(v,w,t) &=& \operatorname*{argmax}_{u}\{Q_u(t,S(v);v) \cdot p(w|u,v)\} \\
\text{backpointer:} \quad & B(v,w,t) &=& B_{U(v,w,t)}(t,S(v);v)
\end{aligned}$$

Silence: in principle, silence is treated as in the bigram LM,
the implementation is more complex.

Traceback implementation (principle):

word string:	$w_1, \ldots, w_n, \ldots, w_N$
with word boundaries:	$t_1, \ldots, t_n, \ldots, t_N$
sentence end symbol:	\$ ($\hat{=}$ Sil)

initialization ("choose arbitrary N"): best word end

$$
\begin{aligned}
(w_{N-1}, w_N) &= \operatorname*{argmax}_{v,w} \{Q_v(T, S(w), w) \cdot p(\$|v, w)\} \\
t_N &= T \\
t_{N-1} &= B_{w_{N-1}}(T, S(w_N), w_N)
\end{aligned}
$$

loop: for $n = N-2, N-3, \ldots$ to $t_0 = 0$ do

{

$$
\begin{aligned}
w_{n-2} &= U(w_{n-1}, w_n, t_{n-1}) \\
t_{n-2} &= B(w_{n-1}, w_n, t_{n-1})
\end{aligned}
$$

}

Time complexity for full DP search (later: DP beam search):

language model type	acoustic search comparisons (= 3· HMM states)	language model comparisons
unigram	$3 \cdot T \cdot [W \cdot S + 1]$	$T \cdot [W + 1]$
position unigram	$3 \cdot T \cdot K \cdot [W \cdot S + 1]$	$T \cdot K \cdot [W + 1]$
bigram	$3 \cdot T \cdot [W \cdot S + W]$	$T \cdot W \cdot [W + 1]$
trigram	$3 \cdot T \cdot W \cdot [W \cdot S + W]$	$T \cdot W \cdot [W^2 + 1]$

symbols:

- T = number of time frames of test utterance
- W = nunber of acoustic reference words
- S = average number of states per (acoustic) word
- K = number of positions in position unigram
- silence model: 1 state

Memory requirements:

- acoustic search: one column for backpointer and score
- language model recombination: traceback arrays with one entry for each LM node

LM type	acoustic search	language model
unigram	$2 \cdot [W \cdot S + 1]$	$2 \cdot T$
position unigram	$2 \cdot K \cdot [W \cdot S + 1]$	$2 \cdot T \cdot K$
bigram	$2 \cdot [W \cdot S + W]$	$2 \cdot T \cdot [W + 1]$
trigram	$2 \cdot W \cdot [W \cdot S + \ldots]$	$2 \cdot T \cdot [W^2 + \ldots]$

4.4 Dynamic Programming Beam Search along with Implementation Details

The search consists of principal components:

- language model recombination: word boundaries
- acoustic search: word interior
- bookkeeping: decisions about word (and boundary) hypotheses
- traceback: construct best scoring word sequence

Modifications for large vocabulary systems as opposed to digit string recognition:

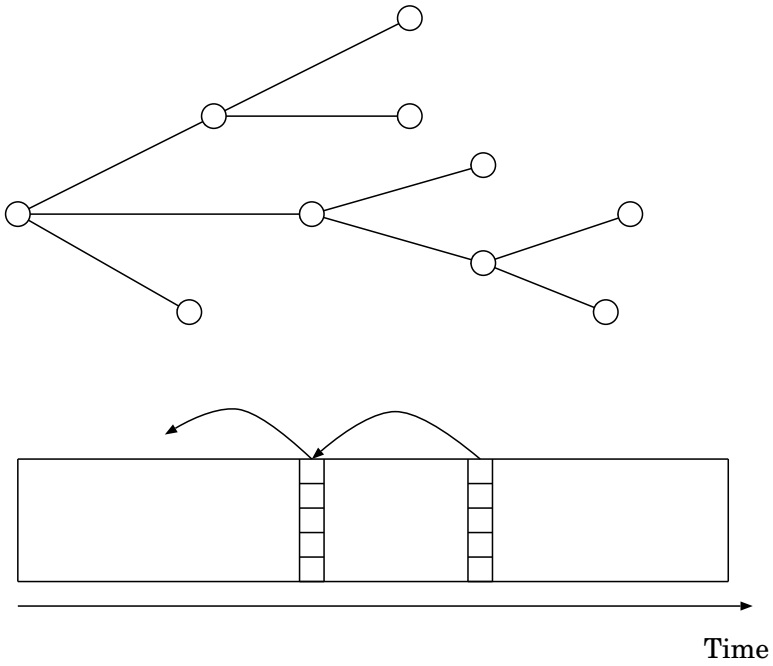
- limit the search space by beam search
- modified bookkeeping for active hypotheses (due to beam search)
- modified bookkeeping for traceback arrays (due to beam search)
- garbage collection for traceback arrays (due to beam search)

Traceback arrays

- Use one index:
  - less memory (exhaustive search)
  - beam search, only few word ends are reached
- Bookkeeping is possible at these stages:
  - at the word ends
  - at the LM nodes (most efficient, smallest number of hypotheses)
  - at the word beginnings
- Organization:
  - so far: one element in traceback array per time frame
  - now: backpointer does not point at the time frame the predecessor word ended, it points at the array element with the corresponding information.

Reminder:

The entries of the traceback array define the nodes of a tree:



- **Garbage collection (beam search): hypotheses can be pruned, array entries no backpointer points at are labeled as free.**
- **Partial traceback: if all backpointers of active hypotheses point at one entry in the traceback array the decision before this entry is determined.**
- **Experimental experience (beam search): delay depends on the task, 1–2 words when using partial traceback.**

Beam search:

- **suboptimal heuristic approach: give up global optimum.**
- **time synchronous search: remaining cost of the path is constant for all hypotheses.**
- **baseline method for pruning:**
  discard unlikely hypotheses at every time frame t:
  - **Acoustic pruning:**
    retains state hypotheses whose scores are close to the score of the best state hypothesis:

$$Q_{AC}(t) := \max_{(v,s)} \{ Q_v(t,s) \} \ ,$$

    prune state hypothesis $(s,t;v)$ iff:

$$Q_v(t,s) < f_{AC} \cdot Q_{AC}(t)$$

- additional pruning steps:
  - Language model pruning:
    retains tree start-up hypotheses whose score is close to the score tree start-up hypothesis:

$$Q_{LM}(t) := \max_{v} \{ Q_v(t, s=0) \}$$

    prune tree start-up hypothesis iff:

$$Q_v(t, s=0) < f_{LM} \cdot Q_{LM}(t) \ ,$$

  - histogram pruning:
    limits the number of surviving state hypotheses to a maximum number (*MaxHyp*).

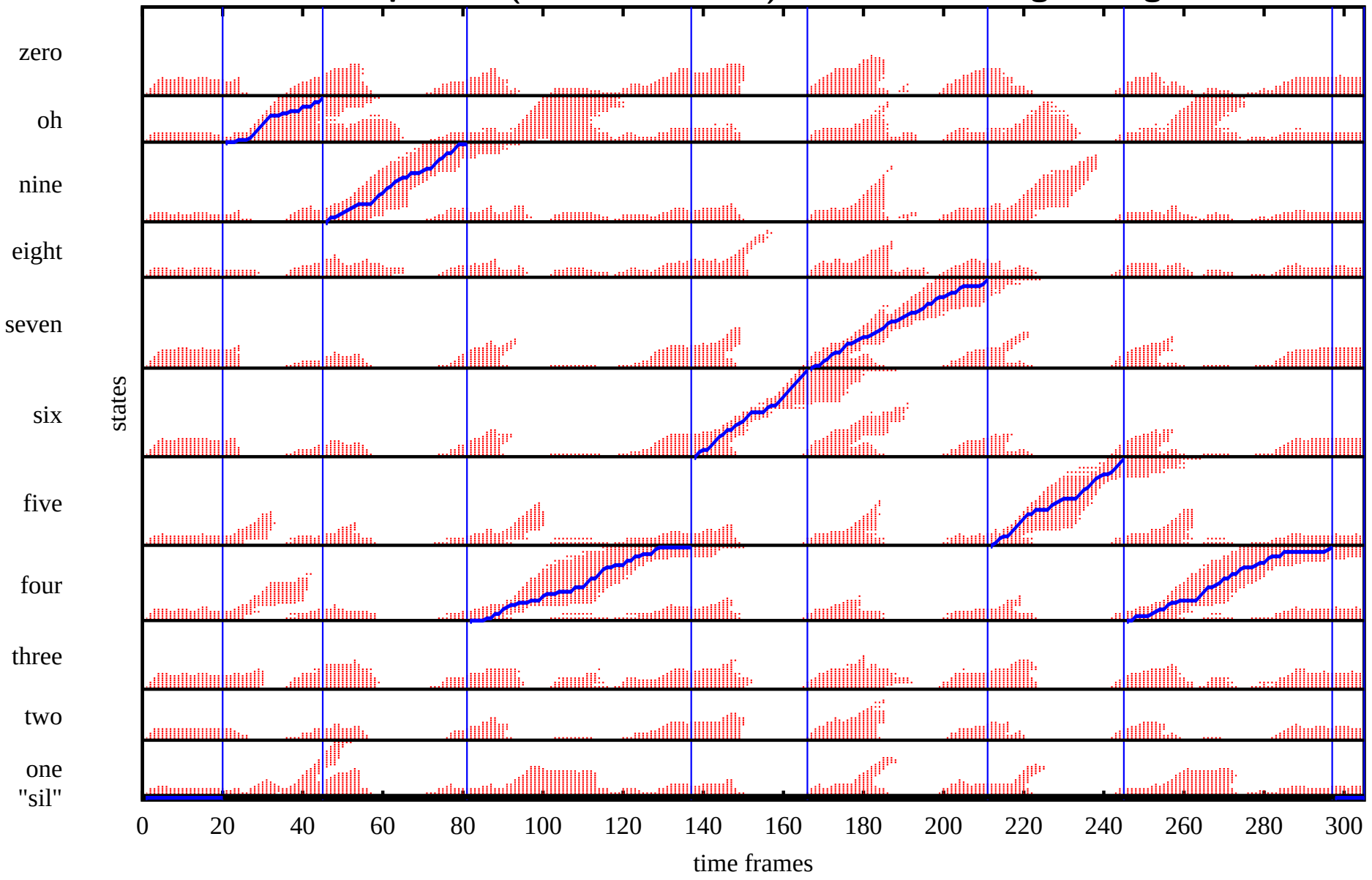
Using pruning techniques can lead to search errors inducing recognition errors.

remember: possible reasons for recognition errors:

- **shortcomings of the acoustic models**
- **shortcomings of the language models**
- **search errors (when using beam search or other heuristic methods)**

In general better acoustic models and language models focus the search space.

Illustration of the search process (DP beam search) for connected digit recognition



Example of the dependency between the search space and the word error rate (WER):
WSJ task, vocabulary size = 20000 words, bigram LM PP = 200

AcuThr k	States (average)	WER [%]
50	252	45.6
60	677	28.3
65	1068	24.2
75	2396	20.6
100	12908	18.4
110	21894	18.3
120	32538	18.2
130	43862	18.2

AcuThr: acoustic pruning threshold

States: average number of state hypotheses in HMM after pruning

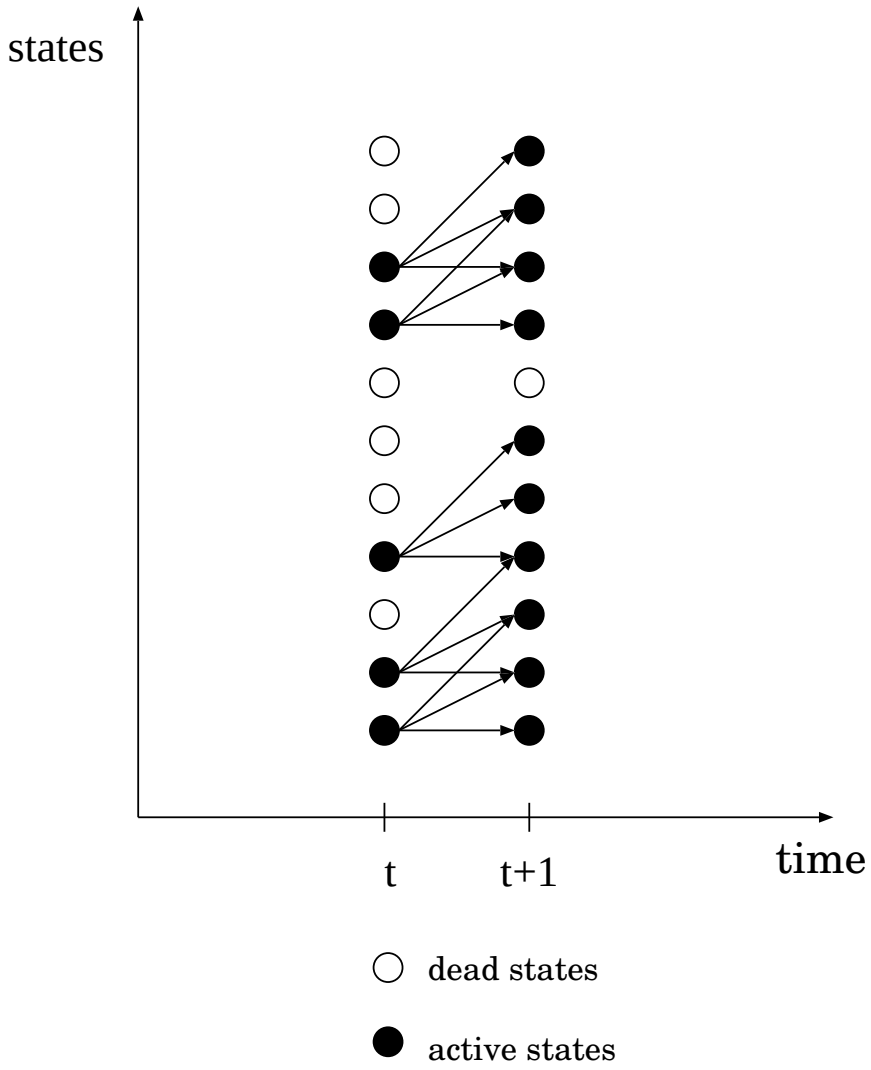
goal: complexity should be linear in the number of active hypotheses.

Efficient expansion of the hypotheses from t to $t+1$ requires these operations:

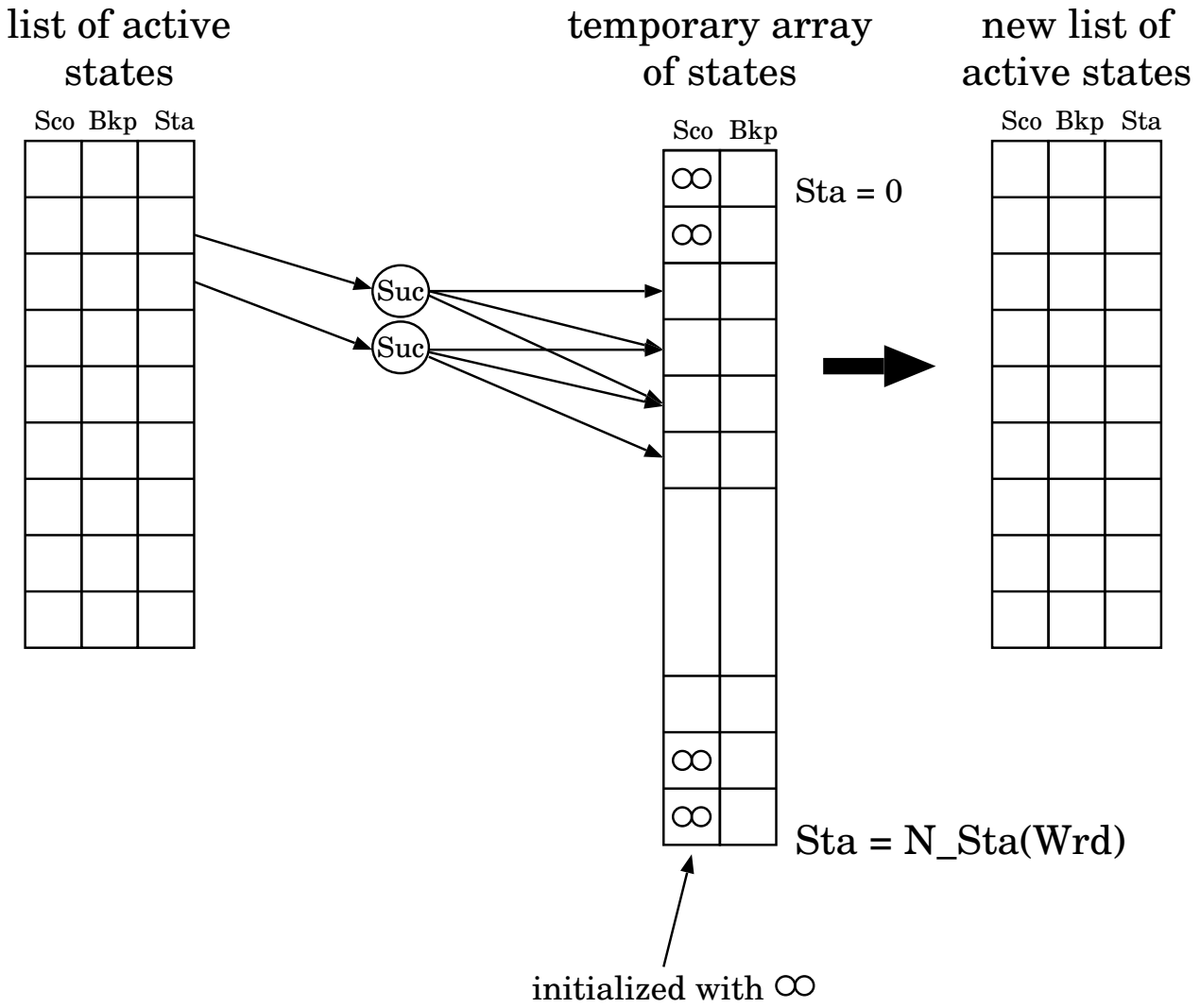
- **search (x, S)**
- **insert (x, S)**
- **initialize (S)**
- **enumerate (S)**

Compare methods for set representation:

- **dictionary operations**
- **array representation of sets**
- **inverted lists and bit vectors**



Extend State Hypotheses: Time Alignment



List of state hypotheses:

From the temporary array, a new list of state hypotheses is generated.

Extend State Hypotheses: Global Organization

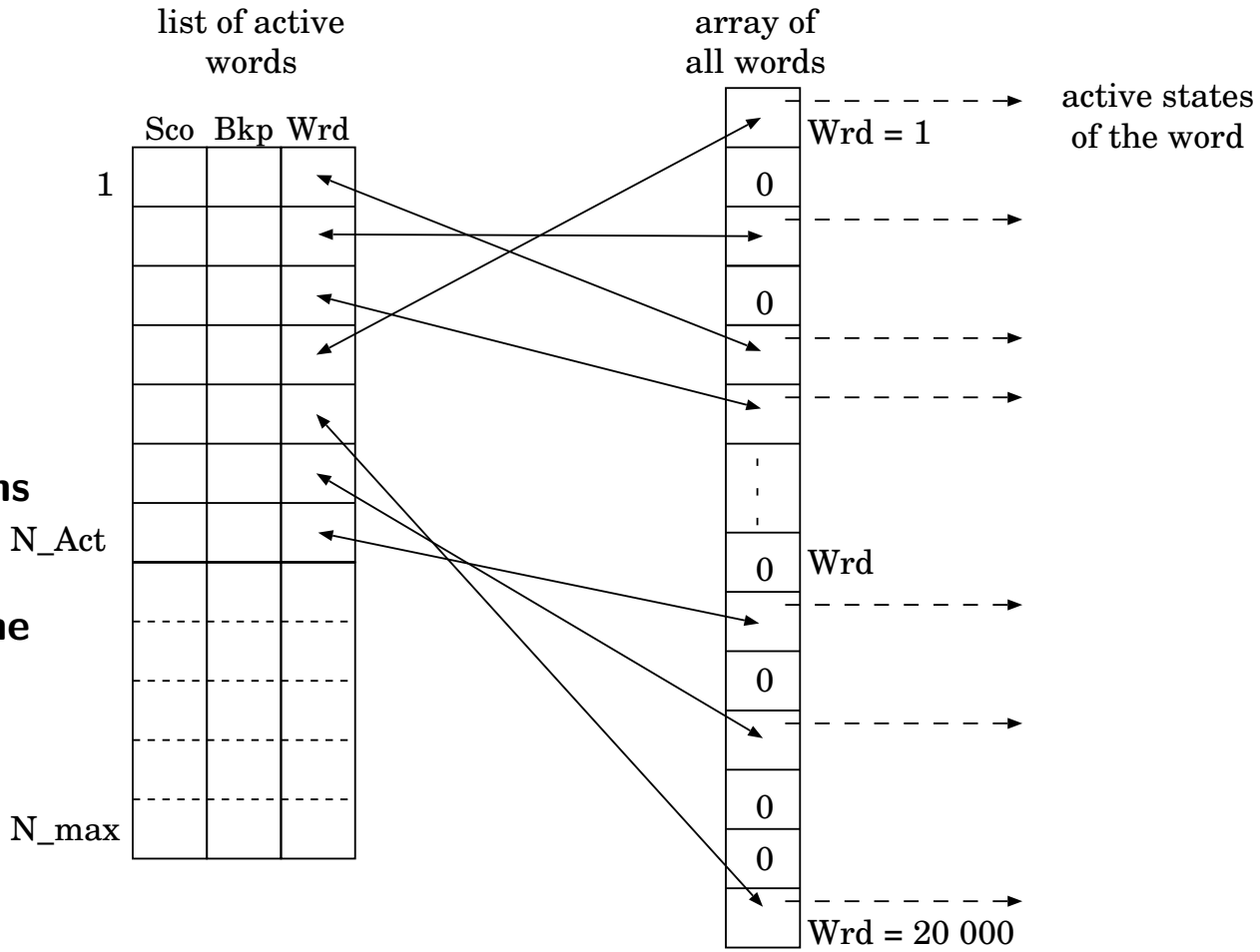
Extension uses two lists:

- **active words**
  {score, backpointer, word}
- **active states for every word**
  {score, backpointer, state}

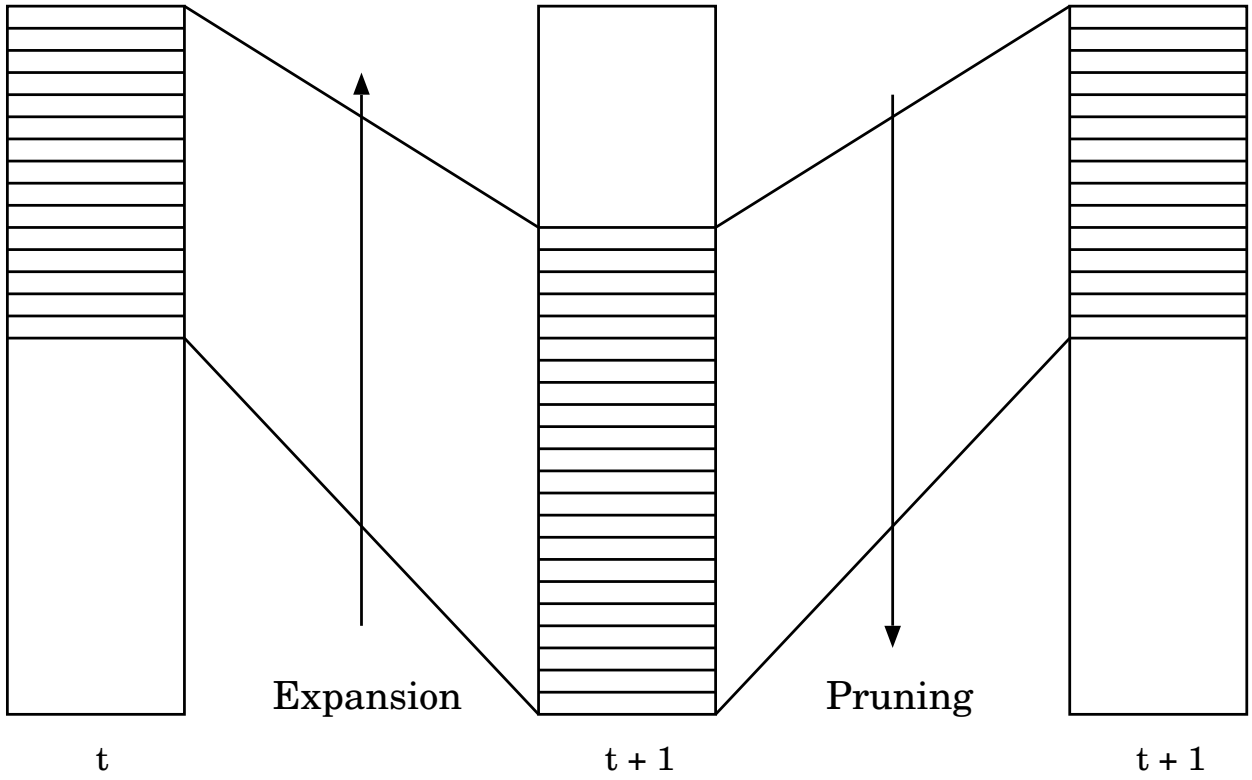
In the example:

- **The list of active words contains the score and the backpointer.**
- **The array of words points at the active states of the word.**

Example: word hypotheses



Update State Hypotheses: Global Organization

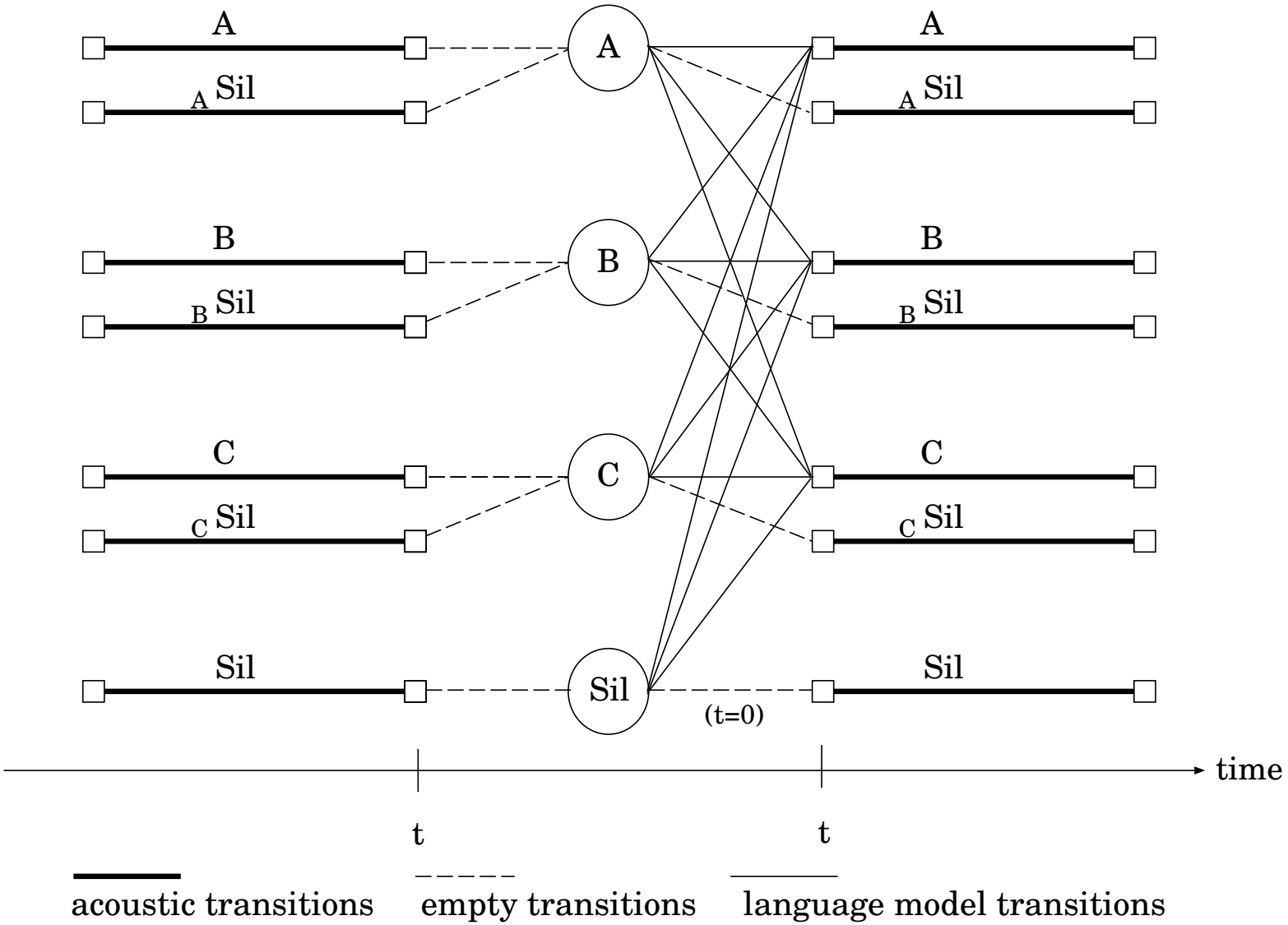


update state hypotheses (in acoustic search):

- **first step (bottom to top):**
  extend hypotheses from t to $t+1$
- **second step (top to bottom):**
  prune hypotheses

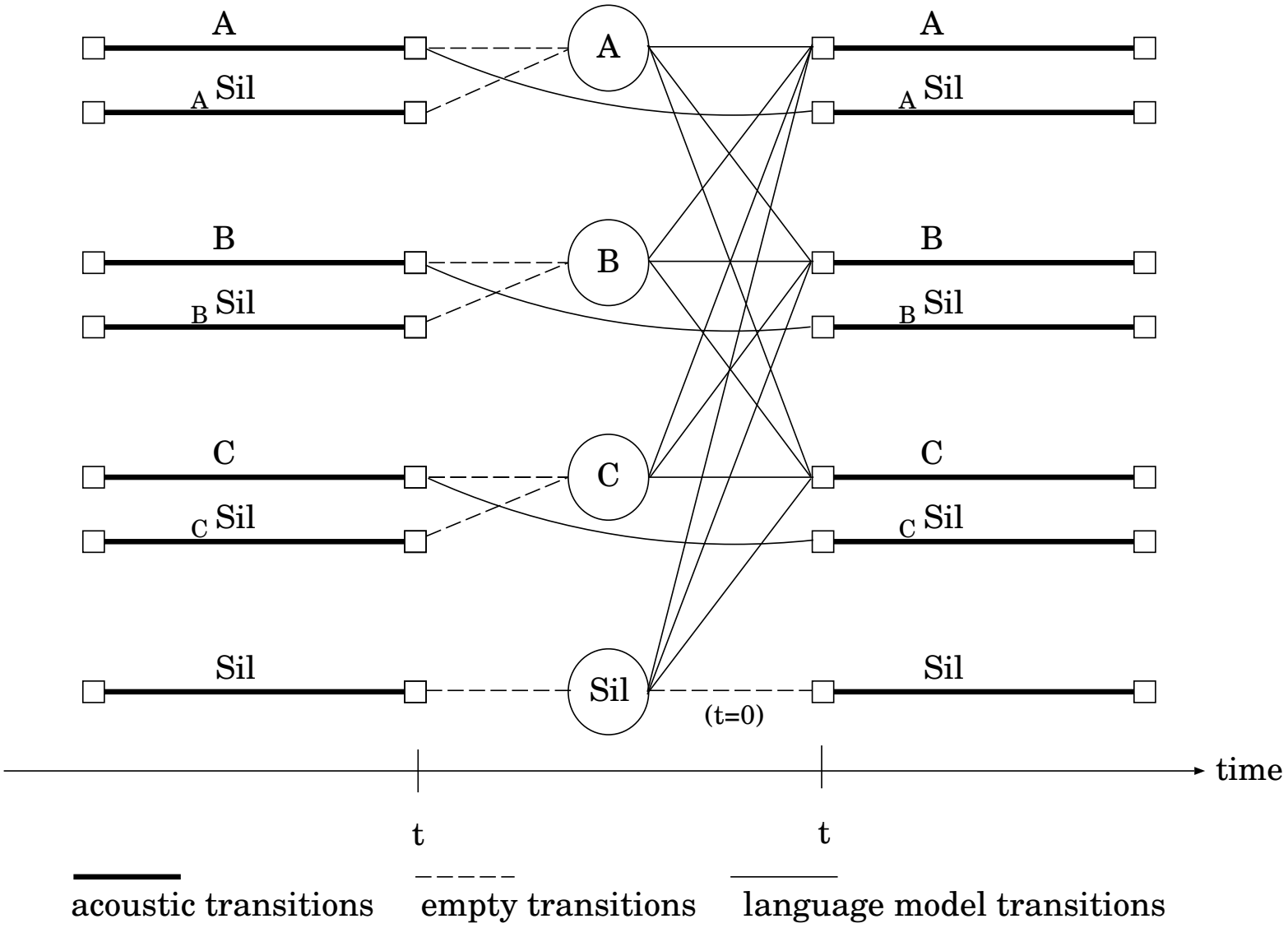
Example: Implementation of DP Beam Search for Bigram LM

consider:



Example: Implementation of DP Beam Search for Bigram LM

consider:



The following example shows a "linear search" with a lexicon represented as linear word list.

The important routines are:

- `MergeSilenceToBigramNodes` :
  recombination of a word and its silence copies (w und ${}_w Sil$).
- `BigramRecombination` :
  bigram recombination in the language model
- `AcoustSearch_LinearLexicon` :
  expands the acoustic hypotheses one time frame ($t \rightarrow t+1$).
- `BookKeeping` :
  puts the entries in the traceback arrays..
- `PurgeStorage` :
  garbage collection for traceback arrays.
- `TracebackWordSeq` :
  trace back recognized word sequence through the traceback array.

Implementation example in C code: show `/linear-search.pdf`

4.5 Excursion (for experts): Language Model Factor

To achieve high performance,
it is very important to give the language model $Pr(w_1^N)$ much more weight than the acoustic model $Pr(x_1^T|w_1^N)$.

Why?

Starting point:
Bayes decision rule with TRUE models $Pr(w_1^N)$ and $Pr(x_1^T|w_1^N)$:

$$\arg\max_{w_1^N} \left\{ Pr(w_1^N) \cdot Pr(x_1^T|w_1^N) \right\}$$

In training, we compute an ESTIMATE of the TRUE models:

$$\begin{aligned} Pr(w_1^N) &\rightarrow p(w_1^N) \\ Pr(x_1^T|w_1^N) &\rightarrow p(x_1^T|w_1^N) \end{aligned}$$

The shapes (i.e. the weights) of the model distributions are changed by exponentiation with exponents α and β:

$$\begin{aligned} p(w_1^N) &\rightarrow p^{\alpha}(w_1^N) \\ p(x_1^T|w_1^N) &\rightarrow p^{\beta}(x_1^T|w_1^N) \end{aligned}$$

Instead of re-normalizing each individual model separately,
we re-normalize by defining the following posterior probability:

$$p(w_1^N|x_1^T) = \frac{p^\alpha(w_1^N) \cdot p^\beta(x_1^T|w_1^N)}{\sum_{\tilde{w}_1^N} p^\alpha(\tilde{w}_1^N) \cdot p^\beta(x_1^T|\tilde{w}_1^N)} = \frac{p^\alpha(w_1^N) \cdot p^\beta(x_1^T|w_1^N)}{\text{const}(w_1^N)}$$

resulting decision rule:

$$\begin{aligned}
\arg\max_{w_1^N} \left\{ p(w_1^N|x_1^T) \right\} &= \operatorname*{argmax}_{w_1^N} \left\{ \frac{p^\alpha(w_1^N) \cdot p^\beta(x_1^T|w_1^N)}{\text{const}(w_1^N)} \right\} \\
&= \operatorname*{argmax}_{w_1^N} \left\{ p^\alpha(w_1^N) \cdot p^\beta(x_1^T|w_1^N) \right\} \\
&= \operatorname*{argmax}_{w_1^N} \left\{ \log \left[p^\alpha(w_1^N) \cdot p^\beta(x_1^T|w_1^N) \right] \right\} \\
&= \operatorname*{argmax}_{w_1^N} \left\{ \alpha \log p(w_1^N) + \beta \log p(x_1^T|w_1^N) \right\} \\
&= \operatorname*{argmax}_{w_1^N} \left\{ \frac{\alpha}{\beta} \log p(w_1^N) + \log p(x_1^T|w_1^N) \right\}
\end{aligned}$$

The factor α/β is referred to as language model factor (e.g. $\approx 10 - 15$).

consider the posterior probability
with suitable word dependent exponents $\beta(w)$:

$$p(w_1^N|x_1^T) = \frac{\prod_n \left[p^\alpha(w_n|w_1^{n-1}) \cdot p^{\beta(w_n)}('x'_n|w_n)\right]}{\sum_{\tilde{w}_1^N} \prod_n \left[p^\alpha(\tilde{w}_n|\tilde{w}_1^{n-1}) \cdot p^{\beta(\tilde{w}_n)}('x'_n|\tilde{w}_n)\right]}$$

resulting decision rule:

$$\begin{aligned}
\arg\max_{w_1^N}\left\{p(w_1^N|x_1^T)\right\} &= \operatorname*{argmax}_{w_1^N}\left\{\prod_n \left[p^\alpha(w_n|w_1^{n-1}) \cdot p^{\beta(w_n)}('x'_n|w_n)\right]\right\} \\
&= \operatorname*{argmax}_{w_1^N}\left\{\sum_n \left[\alpha \log p(w_n|w_1^{n-1}) + \beta(w_n) \log p('x'_n|w_n)\right]\right\} \\
&= \operatorname*{argmax}_{w_1^N}\left\{\sum_n \left[\log p(w_n|w_1^{n-1}) + \frac{\beta(w_n)}{\alpha} \log p('x'_n|w_n)\right]\right\}
\end{aligned}$$

effect: word dependent scale factors $\beta(w)/\alpha$
training: like maximum entropy training

apply scale exponent to each knowledge sources:

language, transition and emission probabilities

$$p(w_1^N|x_1^T) = \frac{\prod_{n=1}^{N} p^{\alpha}(w_n|w_{n-2}^{n-1}) \cdot \max_{s_1^T} \prod_{t=1}^{T} \left[p^{\beta}(s_t|s_{t-1}, w_1^N) \cdot p^{\gamma}(x_t|s_t, w_1^N)\right]}{\sum_{\tilde{w}_1^N} \prod_{n=1}^{N} p^{\alpha}(\tilde{w}_n|\tilde{w}_{n-2}^{n-1}) \cdot \max_{s_1^T} \prod_{t=1}^{T} \left[p^{\beta}(s_t|s_{t-1}, \tilde{w}_1^N) \cdot p^{\gamma}(x_t|s_t, \tilde{w}_1^N)\right]}$$

resulting Bayes decision rule:

$$\begin{aligned}
&\arg\max_{w_1^N} \left\{p(w_1^N|x_1^T)\right\} = \\
&= \arg\max_{w_1^N} \left\{\alpha \sum_{n=1}^{N} \log\, p(w_n|w_{n-2}^{n-1}) \right. \\
&\qquad \left. + \max_{s_1^T} \sum_{t=1}^{T} \left[\beta\, \log\, p(s_t|s_{t-1}, w_1^N) + \gamma\, \log\, p(x_t|s_t, w_1^N)\right]\right\}
\end{aligned}$$

4.6 Excursion (for experts): Length Modelling

refined approach:
for x_1^T and w_1^N, the lengths T and N are random variables themselves.

explicit length models:

- language model: $p(N, w_1^N)$
  check normalization:

$$
\begin{aligned}
p(N, w_1^N) &= p(N) \cdot p(w_1^N|N) \\
\sum_{N, w_1^N} p(N, w_1^N) &= \sum_N p(N) \sum_{w_1^N} p(w_1^N|N) \\
&= \sum_N p(N) \cdot \sum_{w_1^N} \prod_{n=1}^{N} p(w_n|w_1^{n-1}|N) \\
&= \sum_N p(N) \cdot \prod_{n=1}^{N} \sum_{w_n} p(w_n|w_1^{n-1}|N) \\
&= \sum_N p(N) \cdot 1 \; = \; 1
\end{aligned}
$$

- **acoustic model:** $p(T, x_1^T | w_1^N)$
  check normalization (exercise)

- language model:

$$p(N, w_1^N) = p(N) \cdot p(w_1^N|N)$$

model assumptions:

$$= p(N) \cdot \prod_{n=1}^{N} p(w_n|w_{n-2}^{n-1}, N)$$

- acoustic model with word boundaries t_1^N (with $t_0 = 0, t_N = T$):

$$p(T, x_1^T|w_1^N) = \sum_{t_1^{N-1}} p(t_1^N, x_1^T|w_1^N)$$

$$p(t_1^N, x_1^T|w_1^N) = p(t_1^N|w_1^N) \cdot p(x_1^T|t_1^N, w_1^N)$$

model assumptions:

$$= \prod_{n=1}^{N} \left[p(t_n|t_{n-1}, w_n) \cdot p(x_{t_{n-1}+1}^{t_n}|w_n, t_{n-1}^n) \right]$$

- **language model:**

$$\begin{aligned} p(N, w_1^N) &= p(N) \cdot p(w_1^N|N) \\ &\quad \text{model assumptions:} \\ &= p(N) \cdot \prod_{n=1}^{N} p(w_n|w_{n-2}^{n-1}, N) \end{aligned}$$

- **acoustic model with word boundaries t_1^N (with $t_0 = 0, t_N = T$):**

$$p(T, x_1^T|w_1^N) = \sum_{t_1^{N-1}} p(t_1^N, x_1^T|w_1^N)$$

$$\begin{aligned} p(t_1^N, x_1^T|w_1^N) &= p(t_1^N|w_1^N) \cdot p(x_1^T|t_1^N, w_1^N) \\ &\quad \text{model assumptions:} \\ &= \prod_{n=1}^{N} \left[p(t_n|t_{n-1}, w_n) \cdot p(x_{t_{n-1}+1}^{t_n}|w_n, t_{n-1}^n) \right] \end{aligned}$$

optimization criterion (maximum approximation)
using trigram LM $p(w_n|w_{w-2}^{n-1}, N)$ and word segmentation t_1^N (with $t_0 = 0, t_N = T$):

$$\max_N \left\{ p(N) \cdot \max_{w_1^N, t_1^N} \prod_{n=1}^{N} \left[p(w_n|w_{w-2}^{n-1}, N) \cdot p(t_n|t_{n-1}, w_n) \cdot p(x_{t_{n-1}+1}^{t_n}|w_n, t_{n-1}^n) \right] \right\}$$

with the length models:

- length dependencies in language models:
  $p(N)$ and $p(w_n|w_{w-2}^{n-1}, N)$
- duration models of acoustic models:
  $p(t_n|t_{n-1}, w_n)$

experimental results: rarely tested and no significant improvements

5 Search using Lexical Pronunciation Tree

5.1 Review: Linear Search 275

5.2 Tree Search 281

5.3 Dynamic Programming 288

5.4 Pruning Methods 292

5.5 Language Model Look-Ahead 299

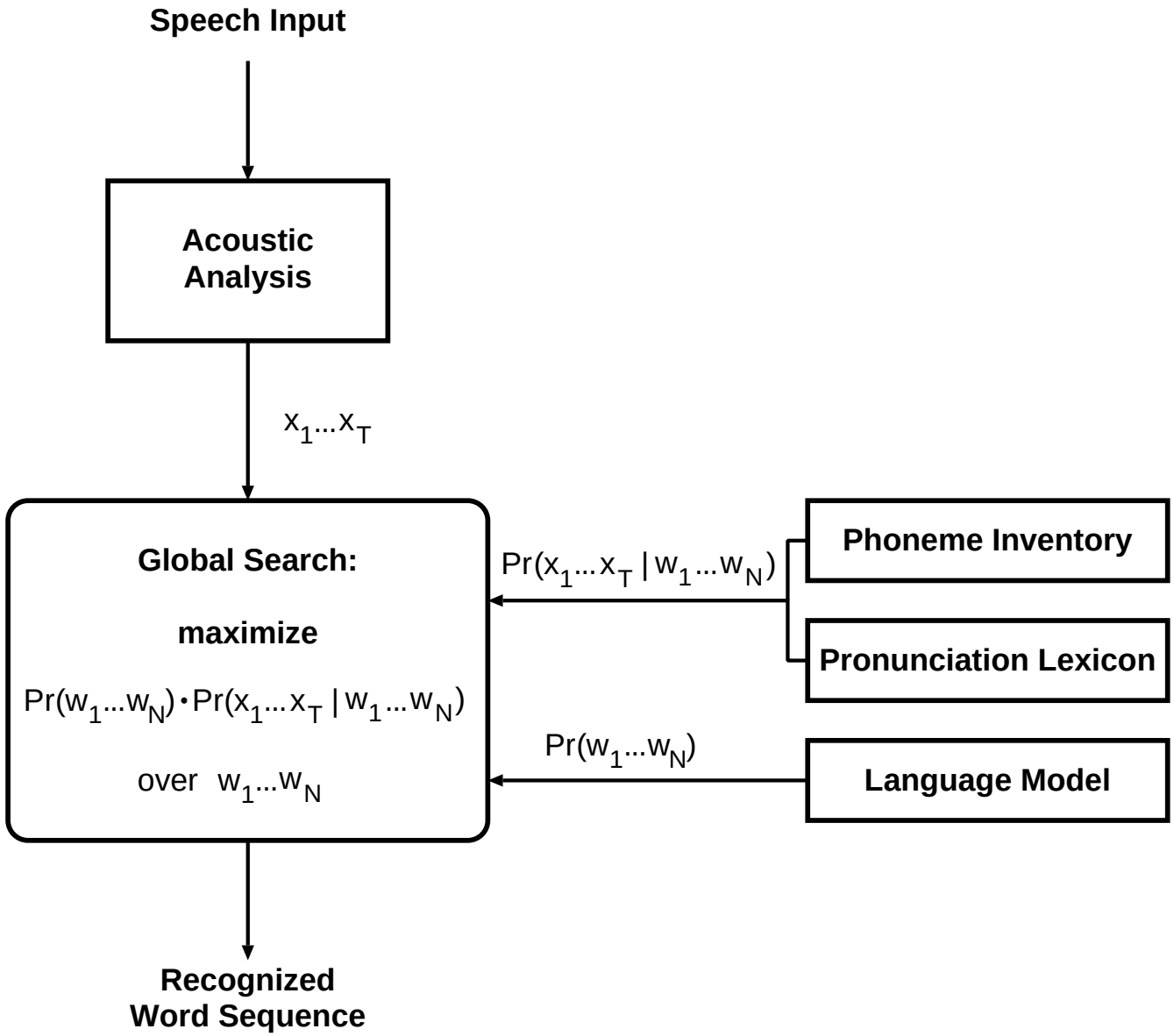
5.6 Phoneme Look-Ahead 305

5.7 Subtree Dominance Recombination (SDR) 314

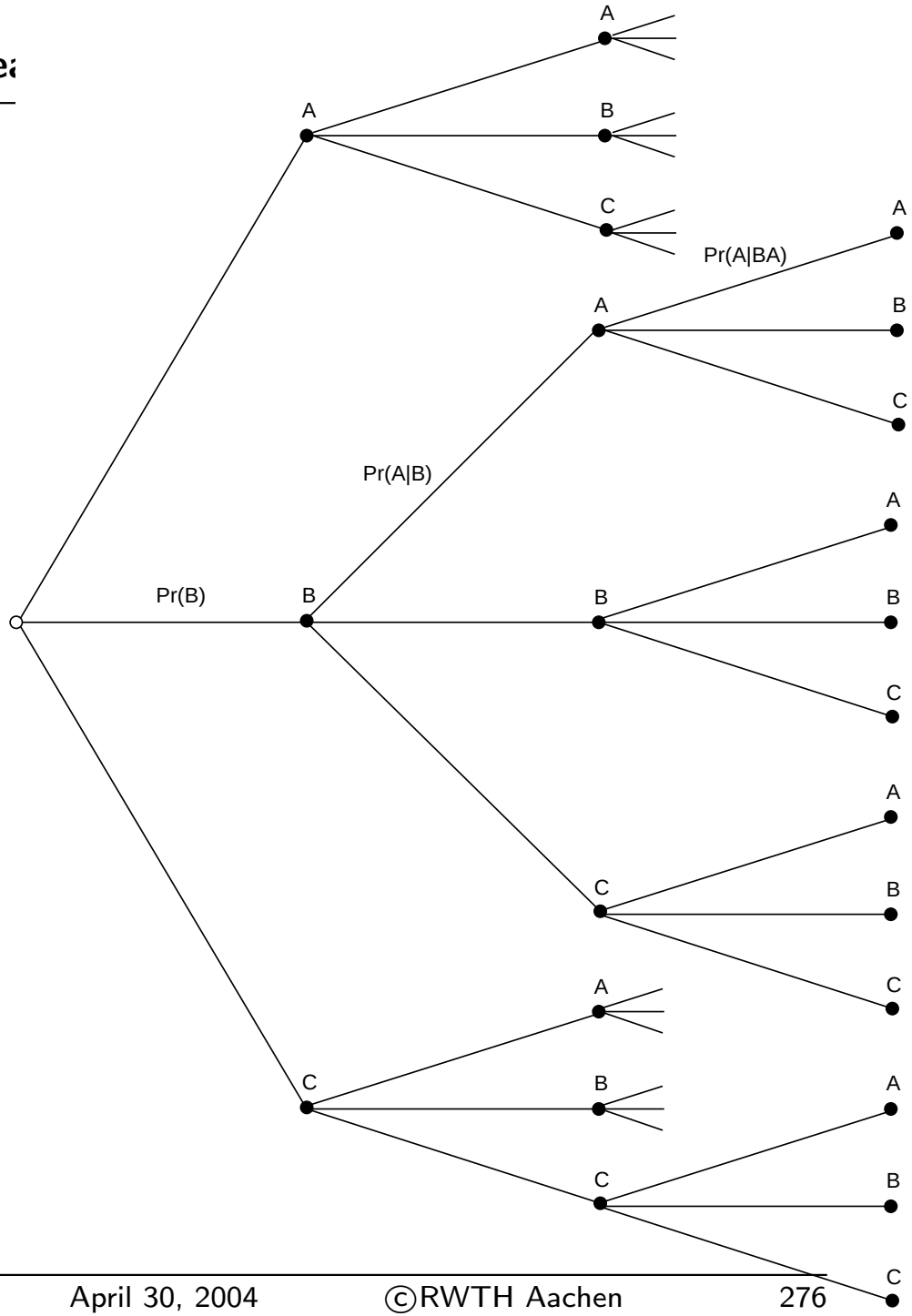
5.8 Implementation 319

5.1 Review: Linear Search

Bayes decision rule:



The search process has to find the spoken word sequence in the set of possible word sequences. The set of possible word sequences can be represented as a tree.



Maximize over all possible wordsequences:

$$\left[w_1^N\right]_{opt} = \underset{w_1^N}{\text{argmax}} \left\{ Pr(w_1^N) \cdot \sum_{s_1^T} Pr(x_1^T, s_1^T | w_1^N) \right\}$$

typical assumptions:

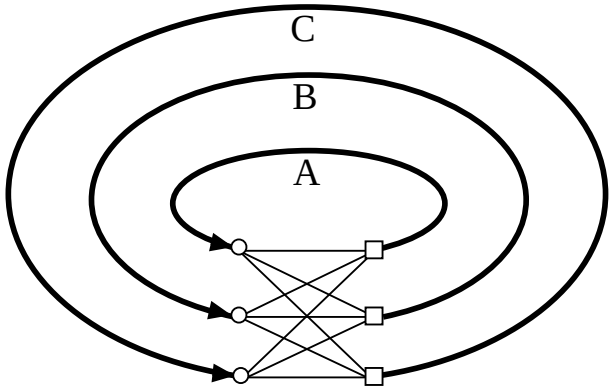
- **Maximum approximation:**

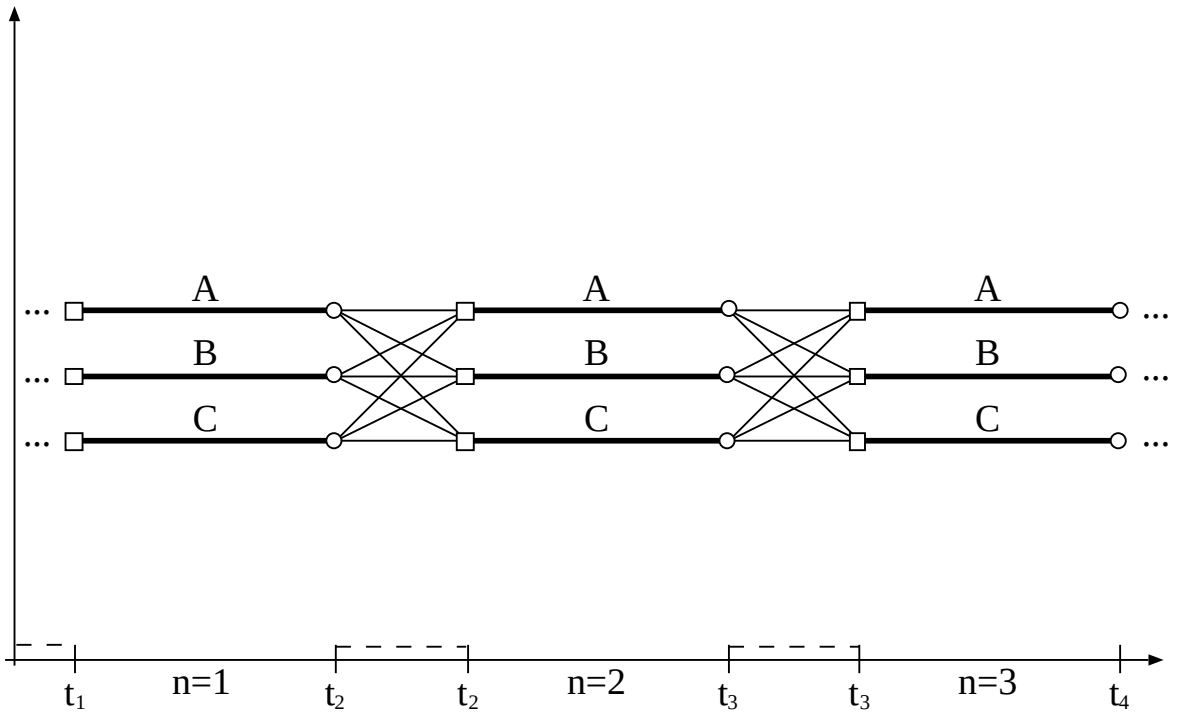
$$\left[w_1^N\right]_{opt} = \underset{w_1^N}{\text{argmax}} \left\{ Pr(w_1^N) \cdot \max_{s_1^T} Pr(x_1^T, s_1^T | w_1^N) \right\} \quad .$$

Note: only consider state sequences s_1^T that are consistent with w_1^N.

assumptions used:

- **Bigram language model:** $p(w_n|w_{n-1})$ **or** $p(w|v)$.



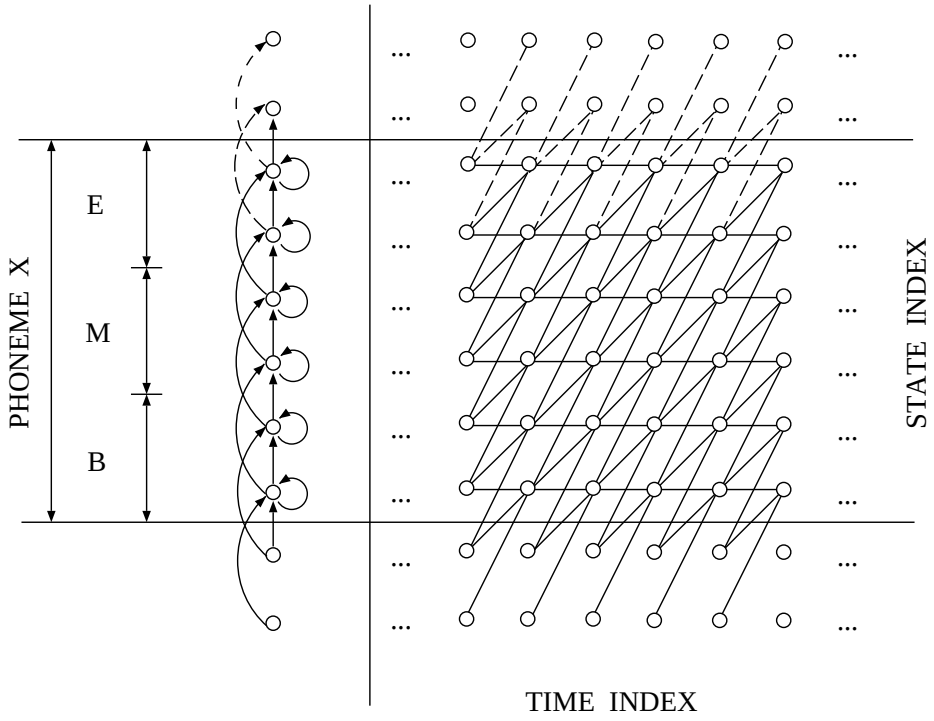


When recognizing continuous speech, language model recombination is carried out every 10ms because the word boundaries are not known.

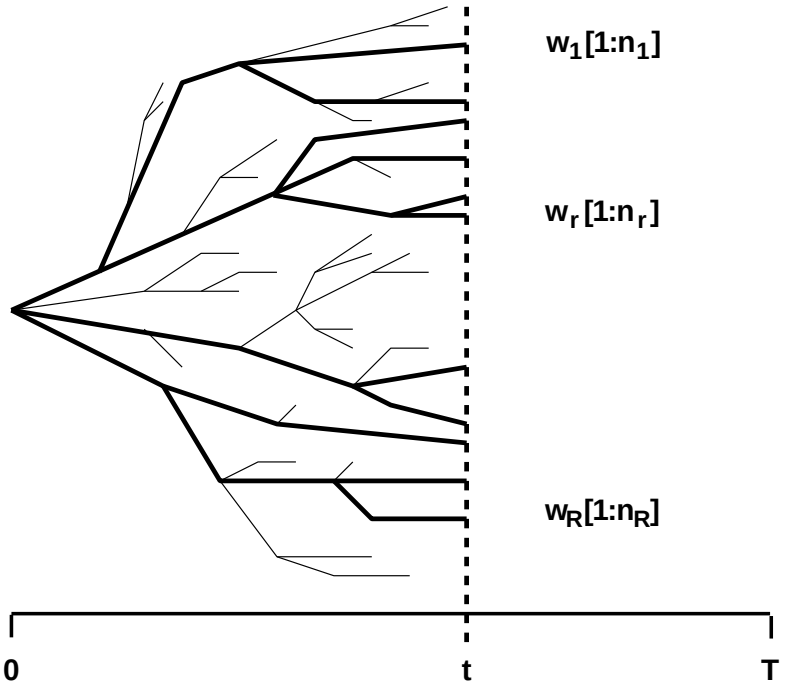
Words are represented as phoneme sequences:

- **Typically about 40 phonemes (German, French, English)**
- **A pronunciation lexicon gives the transcription.**
- **Words are modelled as concatenation of phoneme models.**
- **Automatic training.**
- **Context dependent phonemes:**
  - **diphones**
  - **triphones**

  i.e. one phoneme in the respective context (not two or three phonemes).



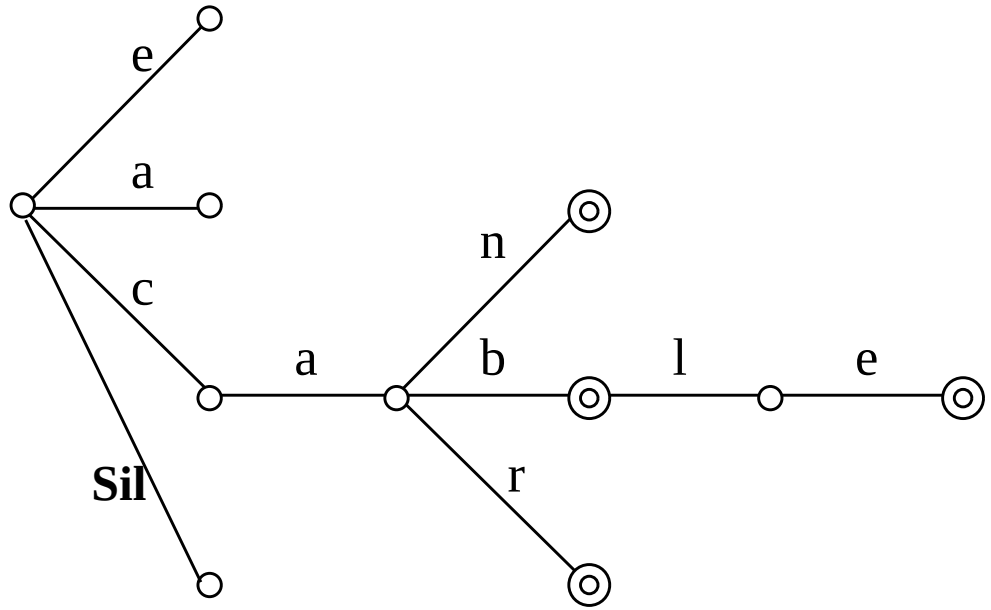
Dynamic programming with beam search:

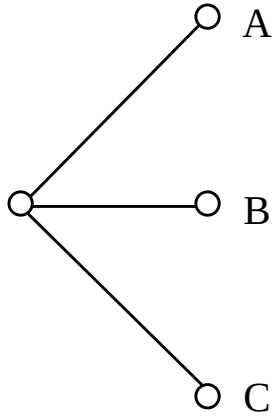


- left-to-right time synchronous evaluation: direct comparison of scores
- beam search:
  - find best hypothesis at time t
  - prune unlikely hypotheses
- experiments: fraction of the potential space (depends on perplexity, acoustic models,...)
- implementation: 'dynamic' search space

5.2 Tree Search

- Every word end hypothesis activates the initial states of all following words.
- Experiments show that in the case of large vocabulary recognition (20000 word vocabulary) 90% of the search effort is in the first two phonemes.
- Solution: represent the vocabulary as a lexical prefix tree.

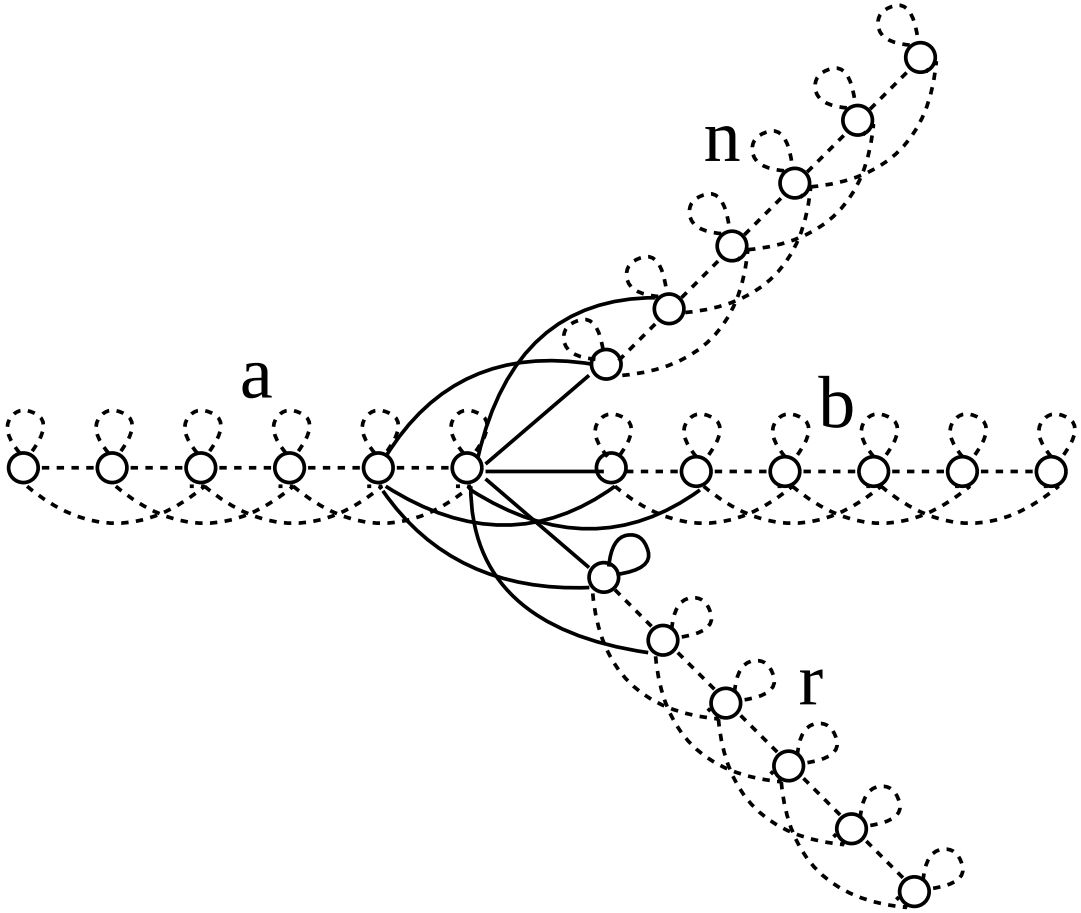




Example with letters instead of phonemes.

The phonemes (triphones) at the arcs of the lexical tree are modeled as usual with 6-state HMMs.

At the tree nodes the following transitions are possible:



The following table shows some statistics for the tree lexicon for the so–led Wall Street Journal task (19 978 words):

- 44 context independent (CI) phoneme models
- 4688 context dependent (CD) phoneme models
- a word: typically 6 phonemes

$\Rightarrow$ compression factor: CI: $\frac{6 \cdot 19\,978}{44587} \approx 2.7$

CD: $\frac{6 \cdot 19\,978}{63155} \approx 1.9$

terminology: lexicon: linear vs. tree
search: linear vs. tree

phoneme generation	number of word ends	number of arcs CD tree	number of arcs CI tree
1	10	544	45
2	231	3626	652
3	1476	9335	3799
4	2959	12180	8245
5	3791	11653	9432
6	3576	9402	7932
7	2845	6829	5895
8	2240	4563	4025
9	1414	2632	2341
10	778	1359	1223
...	...	...	...
16	7	8	8
17	1	1	1
sum	19978	63155	44587

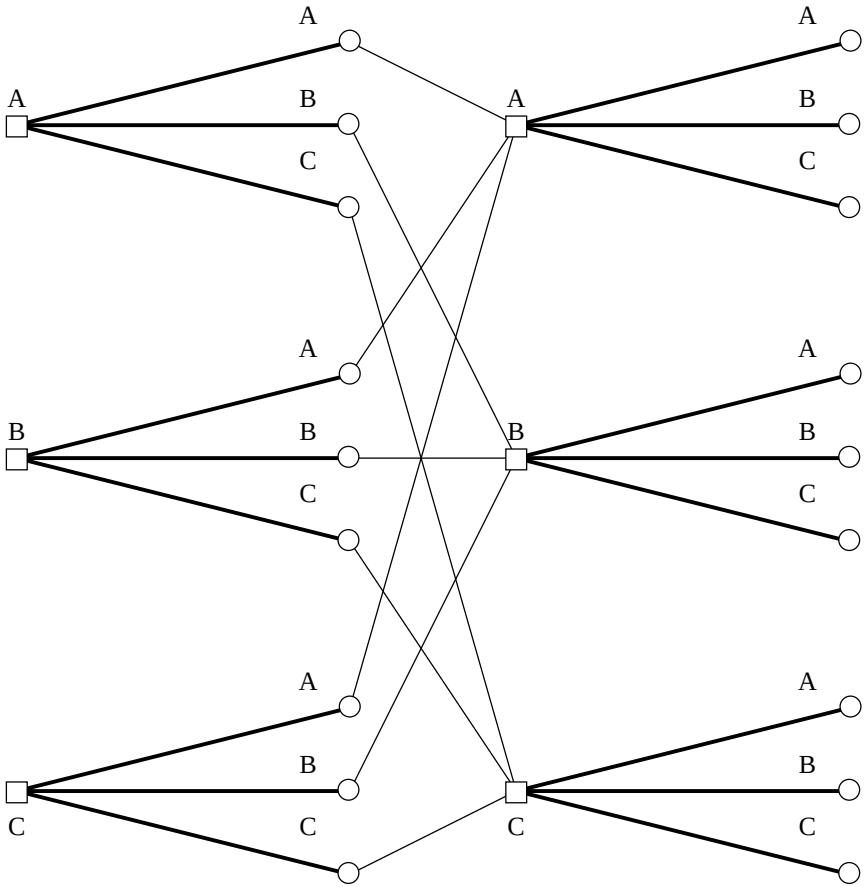
The average compression factor of a tree lexicon compared to a linear lexicon is about 2 to 3.

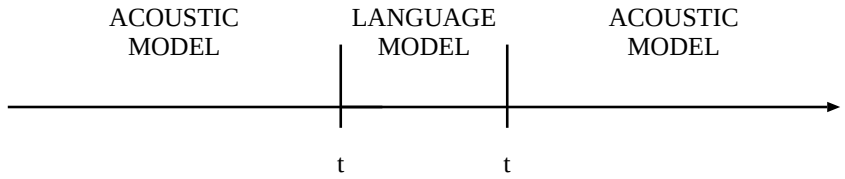
Since the first phonemes require the most search effort the search space reduction using beam search is much larger (a factor of 15 was determined in experiments on the WSJ corpus with a vocabulary of 5000 words).

A fundamental problem: the word identity is not known till a word end node is reached. This requires a different language model recombination:

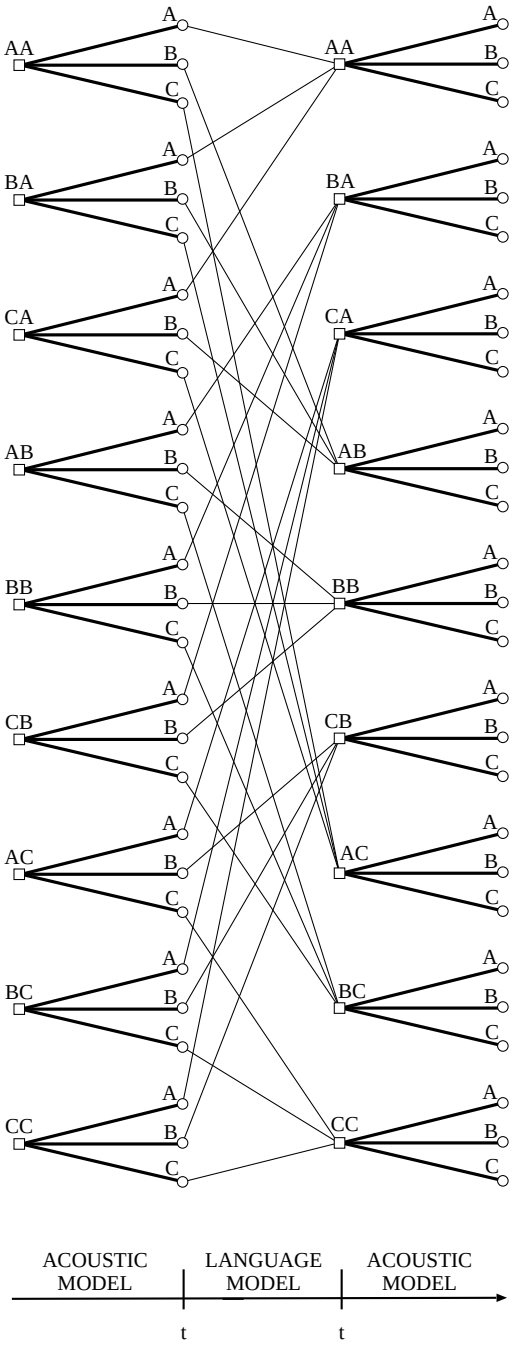
Use a separate tree for every predecessor word (only a few trees are active due to beam search).

Bigram language model recombination for a three word vocabulary:





Trigram language model recombination for a three word vocabulary:



5.3 Dynamic Programming

In the previous picture every arc has costs that only depend on the arc itself, not on the history of the percceding path, so dynamic programming can be used:

Indices: t time, v predecessor word, w word index (defined only at word end states), s state, S_v final state of words v in the tree

Auxiliary quantity:

$Q_v(t,s)$:= probability of the best partial path producing the acoustic vectors $x_1,\ldots,x_t$ and ending in state s of the tree lexicon with v as predecessor word.

Within a word (a tree):

$$Q_v(t,s) \quad = \quad \max_{\sigma} \{p(x_t, s|\sigma) \cdot Q_v(t-1,\sigma)\}$$

Maximize σ over the best predecessor state. At phoneme boundaries the complex transitions have to be considered.

After finishing the computation for the states within a word the recombination is carried out at the word boundaries:

$$Q_v(t,0) \quad = \quad \max_{u} \{p(v|u) \cdot Q_u(t,S_v)\} \quad ,$$

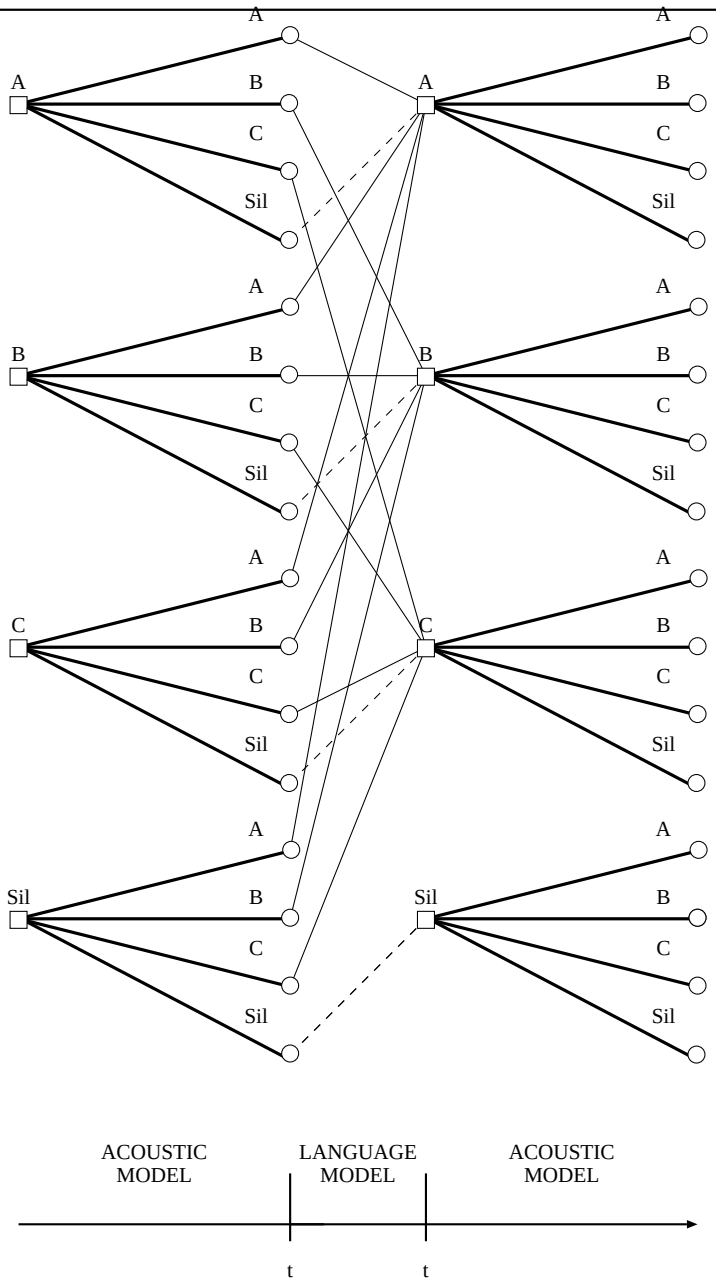
were $s = 0$ is a virtual starting state and S_v the final state of word v.

The silence model requires a special handling:

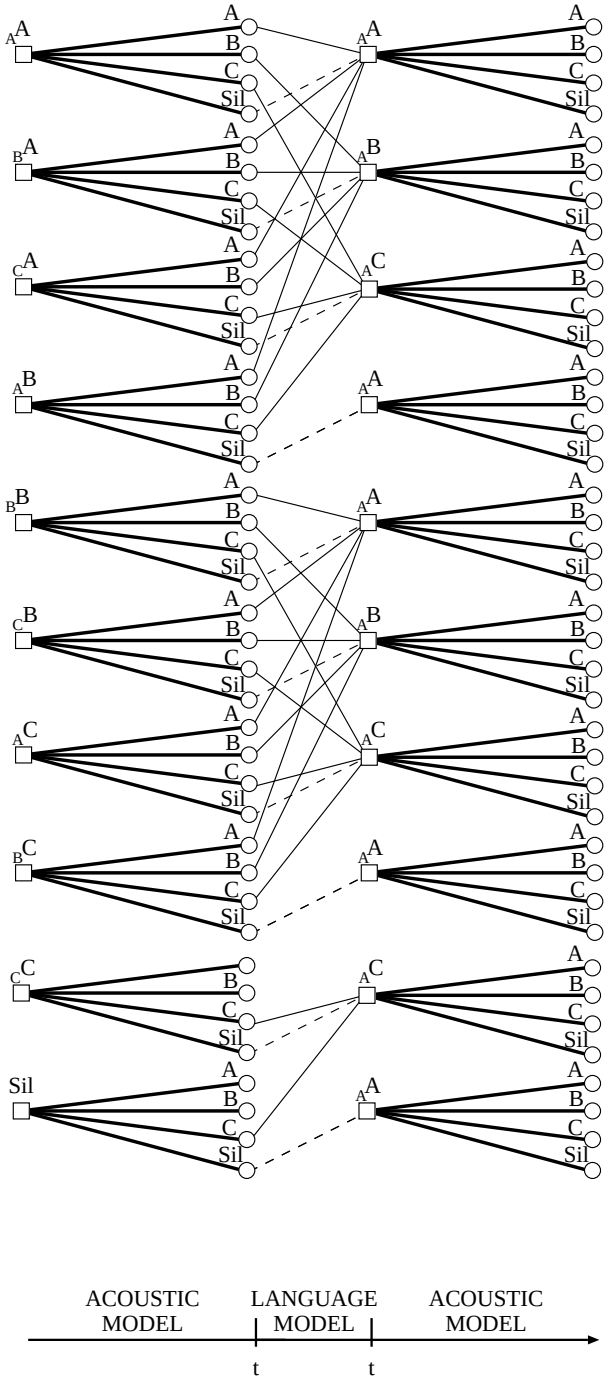
- **silence has to be allowed between words**
- **the word before silence has to be stored**

Solution:

- **a silence arc is added to every tree**
- **during language model recombination only the root of the same tree can be reached from the silence node.**



The silence handling for a trigram language model



5.4 Pruning Methods

- **Acoustic pruning**
  after acoustic recombination:

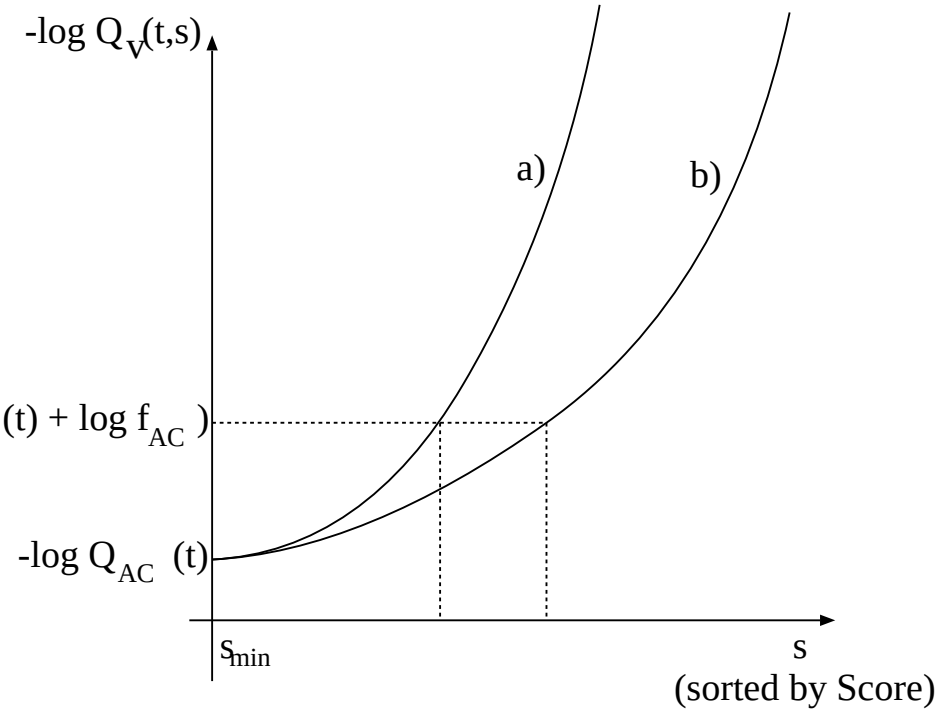
  1. **determine the score of the best hypothesis at time t:**

$$Q_{AC}(t) := \max_{(v,s)} Q_v(t,s).$$

  2. **prune, eliminate a hypothesis $(t, s; v)$ if**

$$Q_v(t,s) < f_{AC} \cdot Q_{AC}(t).$$

  $f_{AC} < 1$ is the acoustic pruning threshold.



effect of acoustic pruning:
a) narrow minimum of the path scores
$\rightarrow$ only a few hypotheses survive the pruning;
b) broad minimum $\rightarrow$ many hypotheses survive.

- **Language model pruning (tree start pruning)**
  prune at the tree roots directly after language model recombination:
  - **determine the score of the best starting tree hypothesis t:**

$$Q_{LM}(t) := \max_{v} Q_v(t, s = 0).$$

  - **prune a tree hypothesis $(t, s = 0; v)$ if:**

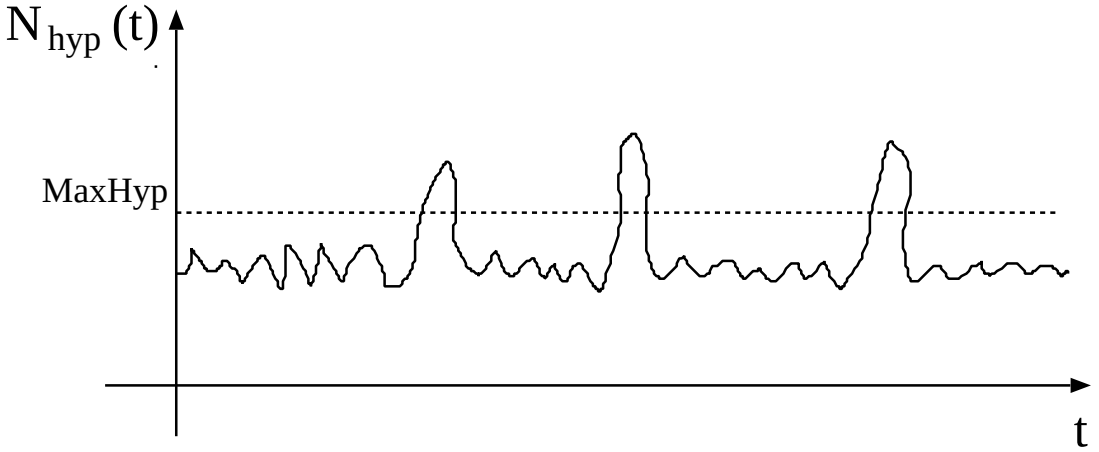
$$Q_v(t, s = 0) < f_{LM} \cdot Q_{LM}(t) \quad .$$

  $f_{LM} < 1$ is the language model pruning threshold.

- **Histogram pruning**
  after acoustic and language model pruning
  if the number of active hypotheses $N(t)$ at time t is larger than a value MaxHyp:

$$N(t) > MaxHyp \quad ,$$

  only retain the MaxHyp best
  hypotheses (using bucket sort). Temporary peaks in the number of hypotheses are eliminated.



Experimental result:

Acoustic pruning is by far the most important method. The others eliminate temporary peaks in the number of hypotheses.

Experimental results on the WSJ corpus:

- Training corpus: WSJ0,1;
- Test corpus: NAB '94 H1 development set;
- speaker independent;
- word conditioned tree copies;
- vocabulary:
  - $W = 20\,000$ words;
  - tree: 63 155 arcs (4 688 context dependent phoneme models)
  - 6 states per phoneme
- bigram language model perplexity 215
- language model pruning threshold 65 000
- $MaxHyp$: 100 000

Experimental Results: Effect of Acoustic Pruning

The acoustic pruning threshold was varied. The number of active states, arcs, trees, and words was measured along with the word error rate (WER).

threshold	average number of active (after pruning)				WER[%]
	states	arcs	trees	word ends	
50 k	252	80	4	9	45.6
60 k	677	213	6	17	28.3
65 k	1068	332	9	24	24.2
75 k	2396	703	15	49	20.6
100 k	12908	3554	38	238	18.4
120 k	32538	8838	59	564	18.2
130 k	43862	11784	67	745	18.2
170 k	79836	20649	87	1363	18.1

The word error rate reaches saturation at about 10000 active phoneme arcs (40000 states).

The potential search space consists of $20000 \cdot 63000 \cdot 6$ states.

Experimental Results: Effect of the Language Model

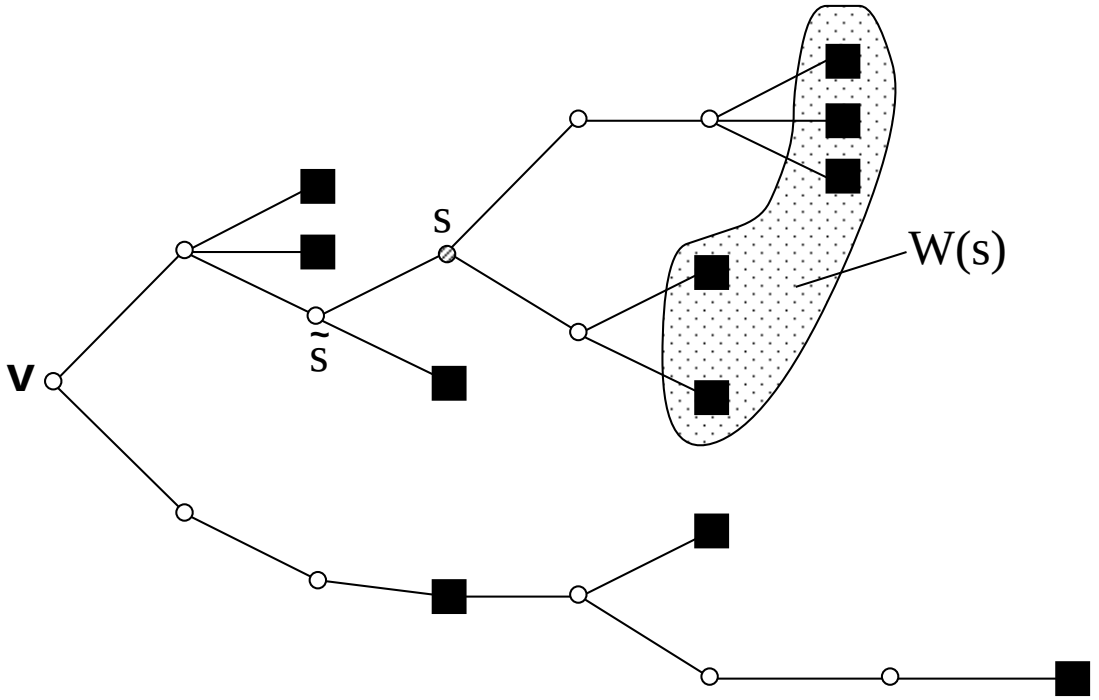
- Corpus: WSJ (Wall Street Journal) Nov '92, development and eval
- 5k vocabulary
- 1.44 h speech data (516984 time frames)
- 18 speakers, 740 sentences, 12137 spoken words
- 2001 mixtures, 96277 densities
- pooled variance vector
- feature vector with 33 components
- LDA (linear discriminant analysis) and CART (phonetic decision tree)
- pruning parameter was optimized for trigram LM
  and kept constant for bi-, uni- und zerogram LM
- LM look-ahead:
  - zerogram-LM: no LM look-ahead
  - unigram-LM: unigram look-ahead
  - bigram-LM: bigram look-ahead
  - trigram-LM: bigram look-ahead
- real time factor (RTF) measured on a 500 Mhz Alpha-PC with 512 MB memory

lang. model	perp.	states after expansion	search space (after prun.) states	arcs	trees	error rates [%] del	ins	WER	RTF
zerogram	4988	10702	7537	1825	1	14.6	0.1	40.0	13.3
unigram	746	6521	3880	958	1	4.6	1.1	22.9	11.0
bigram	107	33775	6472	1835	36	1.4	0.5	6.9	12.2
trigram	56	55769	8013	2333	67	0.7	0.4	4.5	14.9

5.5 Language Model Look-Ahead

- **Language model look-ahead**
  Idea: use the knowledge from the language model as early as possible in the search process.
  From a certain state (or arc) only a small subset of leaves i.e. word ends can be reached.

$W(s)$:= **set of word ends that can be reached from state** s.



Set of word ends $W(s)$ **that can be reached from state** s .

Only a few language model probabilities are possible.
Define:

$$\pi_v(s) := \max_{w \in W(s)} p(w|v) \quad .$$

For acoustic pruning these values are combined with the original scores $Q_v(t,s)$:

$$\tilde{Q}_v(t,s) := \pi_v(s) \cdot Q_v(t,s)$$

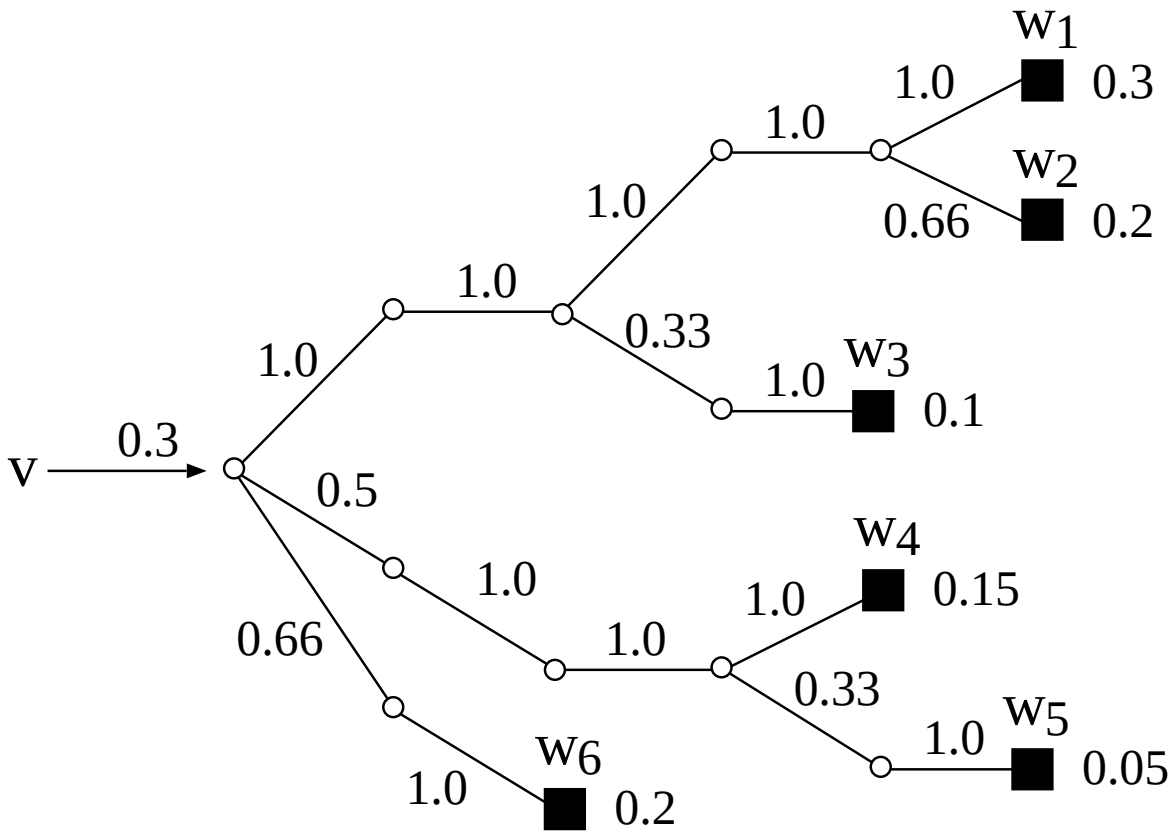
and acoustic pruning is applied to $\tilde{Q}_v(t,s)$ instead of $Q_v(t,s)$.

Problem:

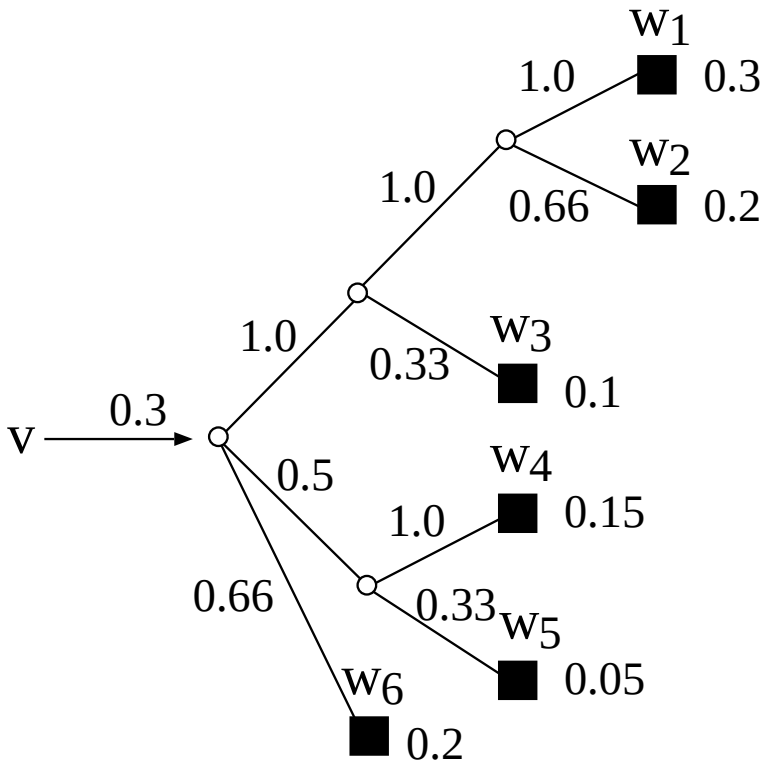
- vocabulary W=20000
- 4688 context dependent phoneme models
- lexikal prefix tree with 63000 phoneme arcs
- $20000 \cdot 63000$ factorized bigram language model probabilities
  $\Rightarrow$ huge look-up table

Solution:

- **Path compression:**
  - **upper limit: W words $\Rightarrow 2 \cdot W$ tree nodes**
  - **compressed LM look-ahead tree**
  - **optional: only consider the first 2-4 arc generations of the LM look-ahead tree**
- **LM tree factorization on demand**



LM tree before compression



LM tree after compression

To reduce memory and computation a unigram language model can be used instead of the bigram model to calculate the look-ahead.

$$\pi_v(s) := \max_{w \in W(s)} p(w)$$

Then only one value has to be stored for for every node.

Experimental result: NAB'94 H1 development set

- 20 speakers (10 men und 10 woman)
- 310 utterances with 7387 words,
  (199 out of vocabulary words)
- RTF (real time factor) measured on a SGI R5000

LM look-ahead	LM look-ahead tree gen.	arcs	trees	search space states	arcs	trees	recog. errors [%] DEL / INS	WER	RTF
no	–	–	–	65568	16932	26	2.4 / 2.5	16.3	110.0
	–	–	–	50020	13034	20	2.5 / 2.5	16.6	91.4
unigram	17	63155	1	16960	4641	32	2.5 / 2.5	16.4	68.1
($PP = 972.6$)	17	63155	1	9443	2599	22	2.6 / 2.5	16.8	54.4
bigram	17	29270	300	3312	935	13	2.5 / 2.6	16.5	32.9
($PP = 198.4$)	3	12002	300	3277	924	13	2.4 / 2.6	16.5	31.4
	2	4097	300	3611	1012	12	2.4 / 2.6	16.9	32.2
	1	544	300	5786	1643	11	2.6 / 2.8	17.0	36.2

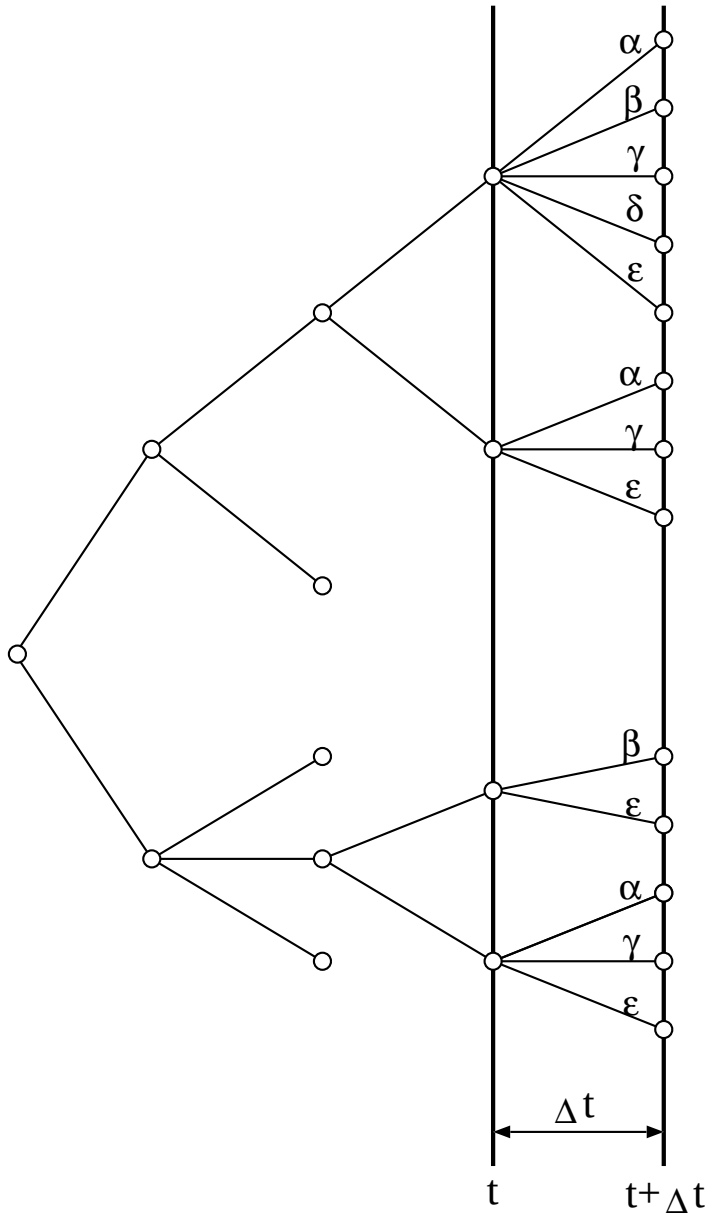
5.6 Phoneme Look-Ahead

Phoneme look-ahead:

look ahead into the 'future'

by one phoneme (CD, CI)

for the acoustic data



definitions and notation:

- α: **a phoneme arc in the lexical prefix tree that shall be started.**
- $\tilde{\alpha}$: **predecessor arc of α. One of its the final states has been reached during search.**
- $\hat{q}(\alpha; t, \Delta t)$: **probability that α produces the vectors $x_{t+1}, ..., x_{t+\Delta t}$.**
  The look-ahead time Δt is constant and approximately equals the average phoneme length:

$$\hat{q}(\alpha; t, \Delta t) \quad := \quad \max_{[s_{t+1}^{t+\Delta t}]} p(x_{t+1}^{t+\Delta t}, s_{t+1}^{t+\Delta t} | \alpha).$$

Pruning strategy:

- **determine the best state hypothesis $\tilde{Q}_0(t)$:**

$$\begin{aligned} \hat{Q}_v(t,\alpha) &:= \hat{q}(\alpha;t,\Delta t) \cdot Q_v(t,S_{\tilde{\alpha}}) \\ \tilde{Q}_0(t) &:= \max_{(v,\alpha)}\{\hat{Q}_v(t,\alpha)\} \end{aligned}$$

- **prune the arc hypothesis (α, v) at time t, if:**

$$\hat{Q}_v(t,\alpha) < f_{PA} \cdot \tilde{Q}_0(t)$$

- **approximate $\tilde{Q}_0(t)$:**

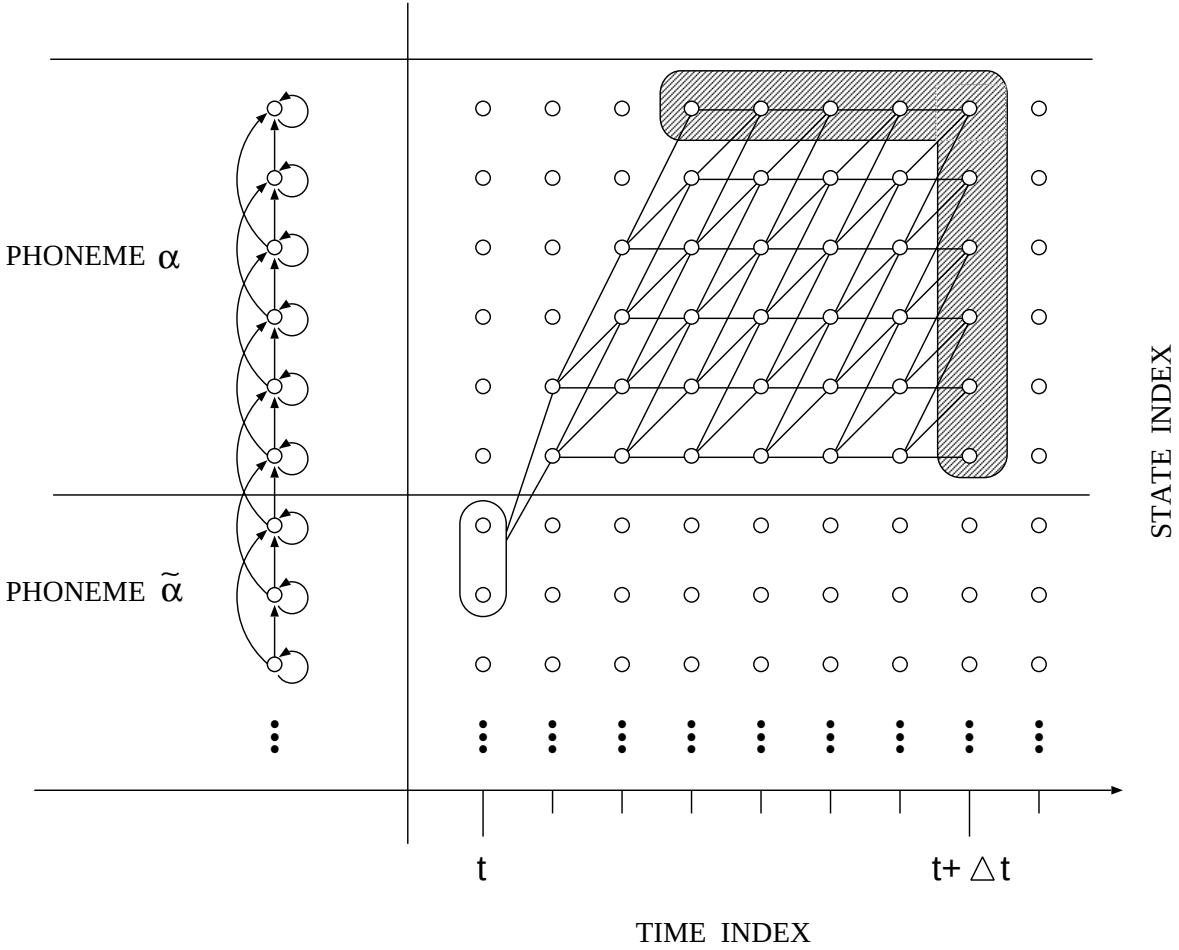
$$\tilde{Q}_0(t) \tilde{=} \max_{\alpha}\{\hat{q}(\alpha;t,\Delta t)\} \cdot \max_{(v,\beta)}\{Q_v(t,S_\beta)\}$$

Calculating the look-ahead probability $\hat{q}(\alpha; t, \Delta t)$:

- **Time alignment for every hypothesized phoneme α:**

$$\Phi_\alpha(\tau, s; t) := \text{probability that the phoneme } \alpha \text{ with the states } 1, ..., s \text{ generates the vectors } x_{t+1}, ..., x_{t+\tau}.$$

- **Hidden Markov model with $S = 6$ states:**



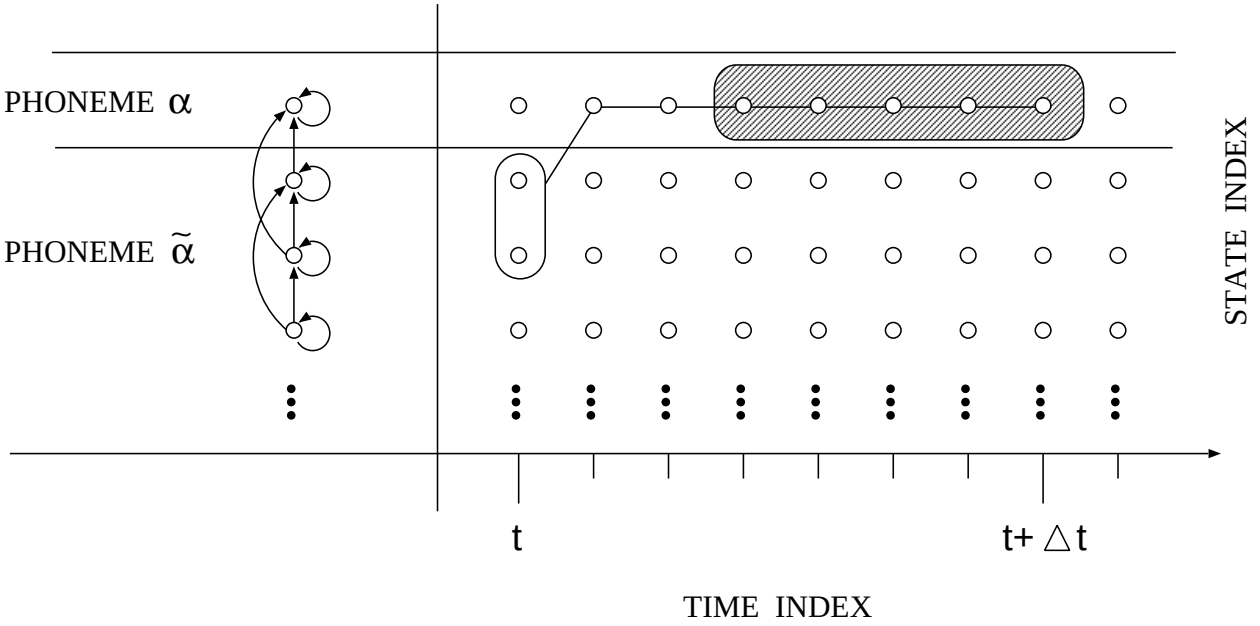
- **Phoneme look-ahead probability:**

$$\hat{q}(\alpha; t, \Delta t) := \max\left\{ \max_{s}\{\Phi_\alpha(\Delta t, s; t)\}, \max_{\tau}\{\Phi_\alpha(\tau, S; t)^{\Delta t/\tau}\}\right\}$$

Efficient calculation of the phoneme look-ahead probability:

$\Rightarrow$ approximations:

- **CI phoneme models instead of CD models (44 CI vs. 4688 CD models)**
- **smaller number of densities**
- **calculation of the look-ahead every second time frame**
- **collapsed one-state HMM.**



Collapsed Hidden Markov Model with one state

Test conditions: NAB'94-H1-Development $20k$

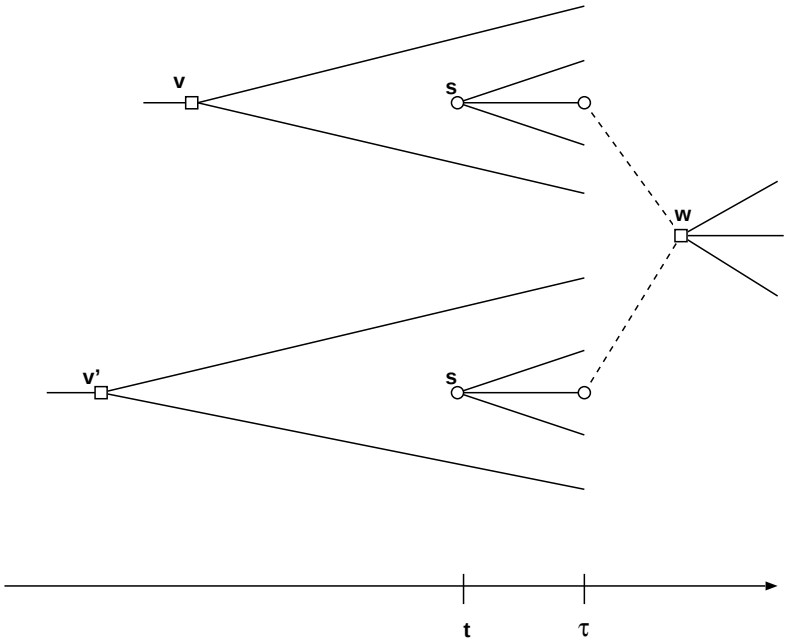
- acoustic search ('detailed search'):
  - 4 688 CD phoneme models
  - 290 000 Laplacian densities per gender
- phoneme Look-Ahead
  - 43 CI phoneme models + 1 silence model
  - $\Delta t = 6$ Time Frames
  - 6 state HMM (497 / 1 926 Laplacian mixtures)
  - 1 state HMM (175 / 675 Laplacian mixtures)
- bigram LM look-ahead
- RTF (real time factor) measured on a SGI R5000

Experimental results:

phoneme look-ahead		search space			recognition errors [%]		
model	densities	states	arcs	trees	DEL / INS	WER	RTF
no	-	3312	935	13	2.5 / 2.6	16.5	32.9
1-state HMM	175	1901	452	10	2.5 / 2.7	16.6	17.9
	175	1551	359	9	2.5 / 2.7	16.7	15.4
	675	1808	434	11	2.5 / 2.6	16.6	16.4
	675	1508	354	10	2.5 / 2.6	16.8	14.2
6-state HMM	497	2472	605	12	2.5 / 2.6	16.5	19.4
	497	1671	379	10	2.4 / 2.7	16.8	16.4
	1926	2160	516	11	2.5 / 2.6	16.5	18.0
	1926	1784	413	11	2.5 / 2.7	16.6	16.6

5.7 Subtree Dominance Recombination (SDR)

Consider a hypothesis $(s, t; v)$ (bigram LM):



Let

$$h((t,s) \rightarrow (\tau, S_w)) \quad := \quad \max_{s_t^\tau} \{Pr(x_t^\tau, s_t^\tau | _) : s_t = s, s_\tau = S_w\}$$

be the acoustic score for the best path from a state s at time t to a final state S_w at time τ. $h((t,s) \rightarrow (\tau, S_w))$ does not depend on the preceding word v.

Because of the recombination on word level we can eliminate or recombine the hypotheses $(s,t;v)$ if:

$$\begin{aligned} \forall w \in W(s): \quad p(w|v) \cdot Q_v(t,s) \cdot h(\ldots) \quad &< \quad \max_{v' \neq v} \{p(w|v') \cdot Q_{v'}(t,s) \cdot h(\ldots)\} \\ &\Updownarrow \\ p(w|v) \cdot Q_v(t,s) \quad &< \quad \max_{v' \neq v} \{p(w|v') \cdot Q_{v'}(t,s)\} \end{aligned}$$

$W(s) :=$ set of leaves i.e. words that can be reached from s.

The partial tree (s, t, v_0) with

$$v_0 = \text{argmax}_{v'} \{p(w|v') \cdot Q_{v'}(t, s)\}$$

dominates the rivaling hypotheses $(s, t; v)$

Because of the complex (w, v) dependency this equation is not very useful.
Simplify both sides by approximation using an upper bound for the left side and a lower bound for the right side:

$$\max_{w \in W(s)} p(w|v) \cdot Q_v(t, s) \quad < \quad \max_{v' \neq v} \left\{ \min_{w \in W(s)} p(w|v') \cdot Q_{v'}(t, s) \right\}$$

Especially at the beginning of a tree $(s = 0)$

$$\max_{w} p(w|v) \cdot Q_v(t, s = 0) \quad < \quad \max_{v'} \left\{ \min_{w} p(w|v') \cdot Q_{v'}(t, s = 0) \right\}$$

Note:

a) **easy implementation:**
only two arrays with index v:

$$\pi_{max}(v,s) := \max_{w \in W(s)} p(w|v)$$
$$\pi_{min}(v,s) := \min_{w \in W(s)} p(w|v)$$

b) **similarity to LM look-ahead**

c) **acoustic pruning is sufficient for an efficient focusing search,**
LM pruning is not needed.

Experimental results:

- NAB'94 H1 development corpus
- vocabulary: 20000 words
- bigram language model
- language model pruning threshold $f_{LM} = \infty$
- RTF (real time factor) measured on an SGI R5000

bigram-look-ahead	MaxHyp	search space			error rate [%]		RTF
		states	arcs	trees	DEL / INS	WER	
without SDR	150 000	3312	935	13	2.5 / 2.6	16.5	32.9
with SDR	∞	3496	887	24	2.4 / 2.6	16.5	39.3
	150 000	3093	859	12	2.4 / 2.6	16.5	29.0
	10 000	2594	743	12	2.4 / 2.6	16.5	25.9
	5 000	2091	606	11	2.4 / 2.6	16.6	25.6

5.8 Implementation

Implementation in C-Code:

- **bigram LM**
- **bookkeeping: at the beginning of each tree;**
- **dynamic generation of tree copies;**
- **"direct memory access" for trees and phoneme arcs, (compare to corresponding methods in linear search);**
- **Language model look-ahead;**
- **NO phoneme look-ahead;**
- **NO subtree dominance recombination;**

memory requirements
(only for the search - with pruning, including unigram LM look-ahead)

- **states :**
  800 000 hypotheses before pruning
  12 Byte each (score, backpointer, state index)
  $\Rightarrow$ 9.6 MByte.

- **arcs:**
  a) total list: 200 000 hypotheses
  8 Byte each (identifier and pointer)
  $\Rightarrow$ 1.6 MByte
  b) temporary list: 70 000 hypotheses with 16 Byte each (4 values)
  $\Rightarrow$ 1.0 MByte

- **bookkeeping array:**
  200 000 entries 20 Byte each (5 values: word, score, time, time stamp, backpointer)
  $\Rightarrow$ 4 MByte

- **Total sum: $\approx$ 16 MByte**

- proceed over time t from left to right
  - ACOUSTIC LEVEL: process state hypotheses
    - initialization: $Q_v(t-1,0) = H(v;t-1)$
    - time alignment: $Q_v(t,s)$ using DP
    - propagate back pointers in time alignment
    - prune unlikely hypotheses
    - purge bookkeeping lists
  - WORD PAIR LEVEL: process word end hypotheses
    - single best: for each pair $(w;t)$ do
      - $H(w;t) = \max_v [p(w|v)\ Q_v(t, S(w))]$
      - $v_0(w;t) = \arg\max_v [p(w|v)\ Q_v(t, S(w))]$
      - store best predecessor $v_0 := v_0(w;t)$
      - store best boundary $\tau_0 := \tau(t; v_0, w)$
- traceback

Implementation example in C–code: show /tree-search.pdf

Tigram LM:

- proceed over time t from left to right
  - ACOUSTIC LEVEL: process states of lexical trees
    - initialization: $Q_{uv}(t-1,0) = H(u,v;t-1)$
      $B_{uv}(t-1,0) = t-1$
    - time alignment: $Q_{uv}(t,s)$ using DP
    - propagate back pointers $B_{uv}(t,s)$
    - prune unlikely hypotheses
    - purge bookkeeping lists
  - WORD PAIR LEVEL: process word ends
    - for each triple $(v,w;t)$ do
      - $H(v,w;t) = \max_u \{ p(w|u,v)\, Q_{uv}(t,S_w) \}$
        $u_0(v,w;t) = \arg\max_u \{ p(w|u,v)\, Q_{uv}(t,S_w) \}$
      - store best predecessor $u_0 := u_0(v,w;t)$
      - store best boundary $\tau_0 := B_{u_0 v}(t,S_w)$

6 Word graph method using word pair approximation

6.1 Why Word Graphs (Lattices) ? 324

6.2 Word Boundary Function 331

6.3 Recombination using the Word Boundary Function 333

6.4 Word Pair Approximation 335

6.5 Algorithm for Word Graph Construction 337

6.6 Algorithm for Word Graph Rescoring 340

6.7 Two Pass vs. Single Pass Method 343

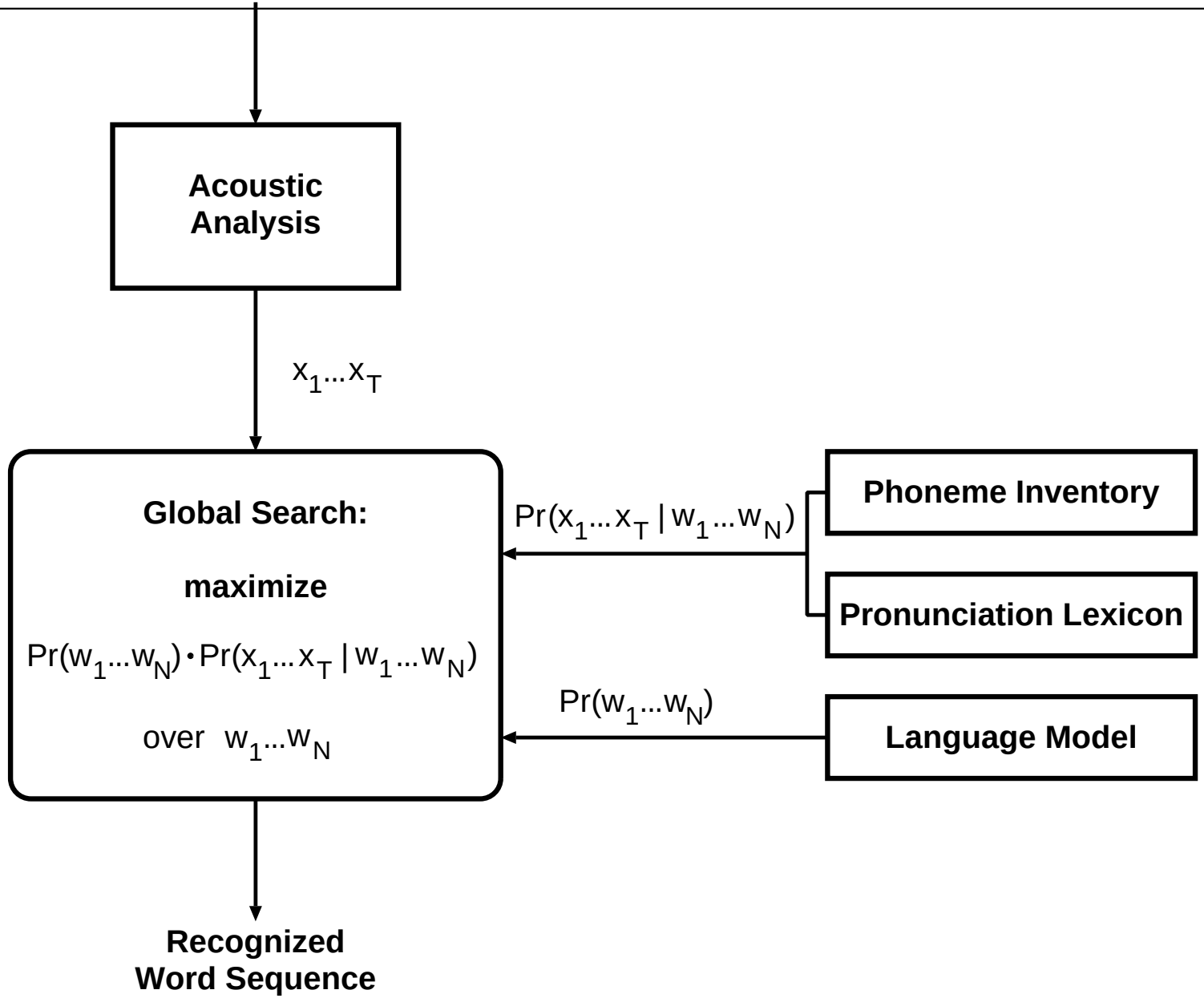
6.8 Lattice Pruning 344

6.9 Experimental Results 346

6.1 Why Word Graphs (Lattices) ?

Integration of Language Models into Bayes Architecture

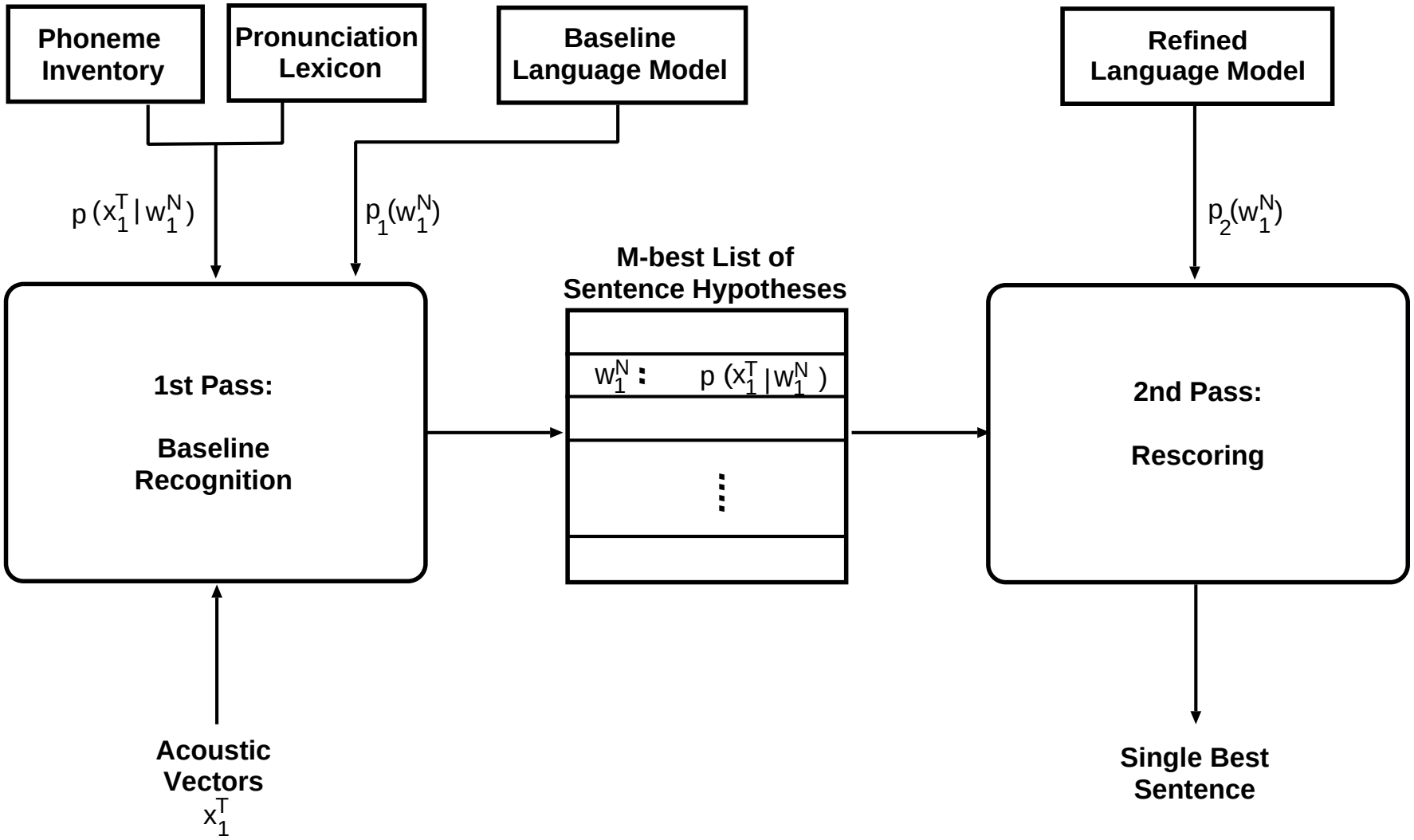
- **One-Pass Approach: Single-Best Sentence**
  (also called: Single-Pass Approach)
- **Two-Pass Approach: M-Best List of Sentences or Word Graph**
  (extension: Multi-Pass Approach)


Acoustic Analysis
$x_1 \ldots x_T$
Global Search:
maximize
$Pr(w_1 \ldots w_N) \cdot Pr(x_1 \ldots x_T \mid w_1 \ldots w_N)$
over $w_1 \ldots w_N$
$Pr(x_1 \ldots x_T \mid w_1 \ldots w_N)$
Phoneme Inventory
Pronunciation Lexicon
$Pr(w_1 \ldots w_N)$
Language Model
Recognized Word Sequence

how to apply a refined language model ?

- **One-Pass Approach: Single-Best Sentence**
  perform the FULL recognition process
- **Two-Pass Approach: Word Graph or M-Best List**
  - **first pass: use baseline language model, e.g. trigram language model**
    output (interface): word graph or M-best list
  - **second pass: rescore output with refined language model, e.g. context-free grammar**

Search: Two-Pass Approach



Search: M-best List of Sentences and Word Graph

M-best list of sentences:
– list $\mathcal{L}(x_1^T)$ of M best sentences ($M = 100, ..., 1000$)
– for each sentence w_1^N, report acoustic probability $Pr(x_1^T|w_1^N)$

The M-best list $\mathcal{L}(x_1^T)$ can be considered as a filter:

. First pass: generate list $\mathcal{L}(x_1^T) \subset w_1^N$

. Second pass: find the maximum in this subset of word sequences with refined models

$$\underset{\tilde{w}_1^N \in \mathcal{L}(x_1^T)}{\operatorname{argmax}} \left\{ Pr(\tilde{w}_1^N) Pr(x_1^T|\tilde{w}_1^N) \right\}$$

e.g. more complex language models or refined acoustic models like across word models

More compact alternative to M-best list: word graph

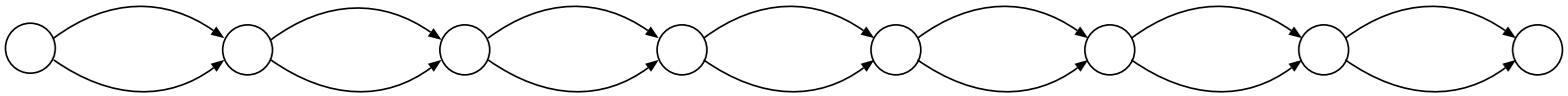
attach to each word graph arc:

– word identity w

– beginning time t_b and end time t_e

– acoustic probability $p(x_{t_b}^{t_e}|w)$ of acoustic vectors $x_{t_b}, ..., x_{t_e}$

example: simple word graph

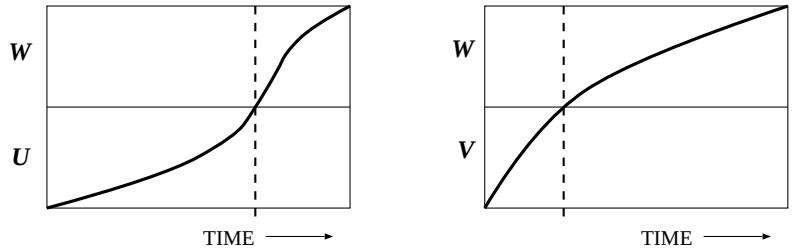


Summary:

- separation of the two levels:
  - acoustic recognition
  - postprocessing: complex language model or understanding
- methods:
  - list of M best sentences
  - word graph: network
- example: 2 word hypotheses per position in a 10-word sentence
  - M-best list: $2^{10} = 1024$ sentence hypotheses
  - word graph: $2 \cdot 10 = 20$ arcs



- problem for word graph: word boundaries may vary



6.2 Word Boundary Function

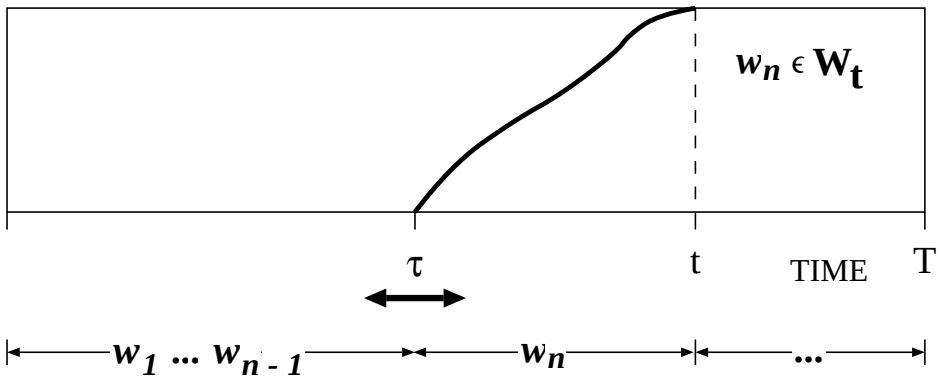
- starting point: similar to time conditioned search
- definitions:

  $h(w;\tau,t) := Pr(x_{\tau+1}^t|w) =$ probability that word w produces the acoustic vectors $x_{\tau+1}...x_t$.

  $G(w_1^n;t) := Pr(w_1^n) \cdot Pr(x_1^t|w_1^n) =$ (joint) probability of generating the acoustic vectors $x_1...x_t$ and a word sequence w_1^n with ending time t.

- decomposition:

$$\underbrace{x_1,\cdots,\cdots,x_\tau}_{G(w_1^{n-1};\tau)} \quad \underbrace{x_{\tau+1},\cdots,x_t}_{h(w_n;\tau,t)} \quad \underbrace{x_{t+1},\cdots,\cdots,x_T}_{\cdots}$$



- optimization over word boundary τ:

$$\begin{aligned} G(w_1^n; t) &= \max_{\tau}\{Pr(w_n|w_1^{n-1}) \cdot G(w_1^{n-1}; \tau) \cdot h(w_n; \tau, t)\} \\ &= Pr(w_n|w_1^{n-1}) \cdot \max_{\tau}\{G(w_1^{n-1}; \tau) \cdot h(w_n; \tau, t)\} \end{aligned}$$

- define word boundary function:

$$\tau(t; w_1^n) := \arg\max_{\tau}\{G(w_1^{n-1}; \tau) \cdot h(w_n; \tau, t)\}$$

6.3 Recombination using the Word Boundary Function

Consider an m-gram language model:

$$p(w_n|w_{n-m+1}^{n-1}) \quad \text{or} \quad p(u_m|u_1^{m-1}) \quad .$$

To apply dynamic programming it is sufficient to consider only the last $m-1$ words $u_2, \ldots, u_{m-1}$.

Define an auxiliary quantity: conditional probability for the generation of the acoustic vectors $x_1 \ldots x_t$ and the a word sequence that ends with u_2^m at time t:

$$H(u_2^m; t) := \max_{w_1^{n-m+1}} \left[Pr(w_1^n) \cdot Pr(x_1^t|w_1^n) : w_{n-m+2}^n = u_2^m \right]$$

Dynamic programming optimizes the word u_1 and the word boundary τ:

$$H(u_2^m; t) = \max_{u_1} \left[p(u_m|u_1^{m-1}) \cdot \max_{\tau} \left\{ H(u_1^{m-1}; \tau) \cdot h(u_m; \tau, t) \right\} \right] .$$

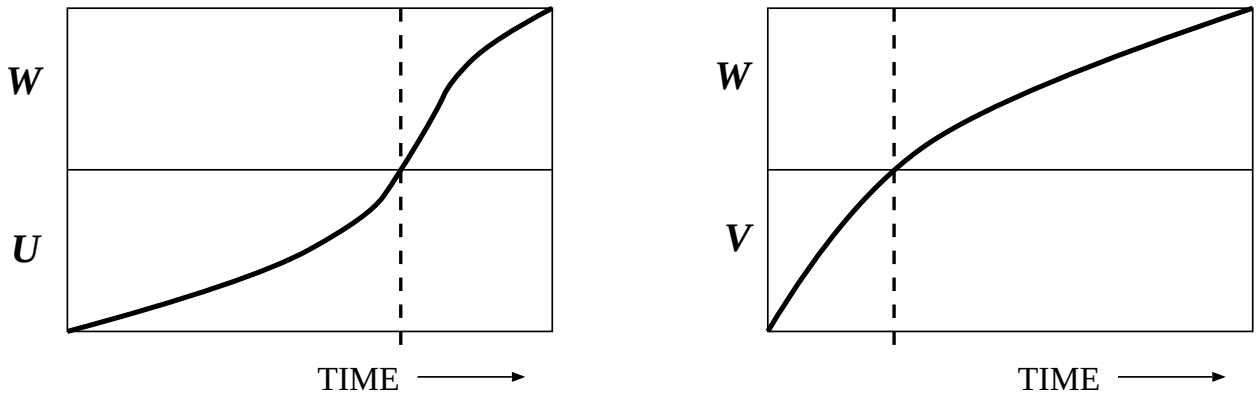
Maximum approximation is used here to avoid the sum over all word boundaries τ.

The definition of the word boundary function

$$\tau(t; u_1^m) := \underset{\tau}{\operatorname{argmax}} \left\{ H(u_1^{m-1}; \tau) \cdot h(u_m; \tau, t) \right\}$$

leads to:

$$H(u_2^m; t) = \max_{u_1} \left[p(u_m | u_1^{m-1}) \cdot H(u_1^{m-1}; \tau(t; u_1^m)) \cdot h(u_m; \tau(t; u_1^m), t) \right] .$$



The word boundary depends on the identity of the surrounding words.

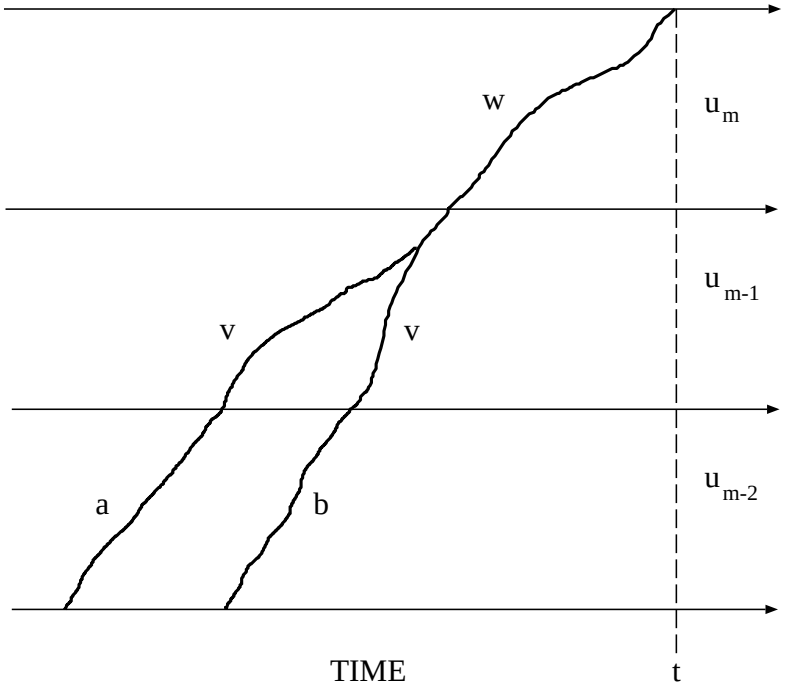
6.4 Word Pair Approximation

remember: τ = start time of hypothesis (t, w)

ASSUMPTION:
τ depends only on the predecessor word v
(in addition to (t, w)):

$$\tau = \tau(t; v, w)$$

Experiments show:
the approximation is correct if the predecessor word is sufficiently long.

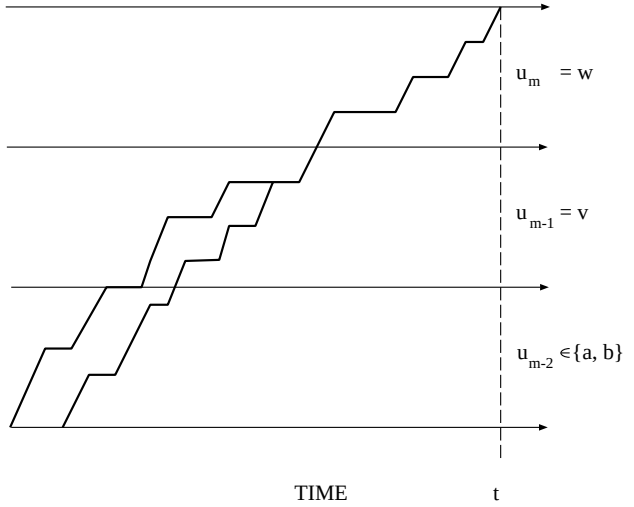


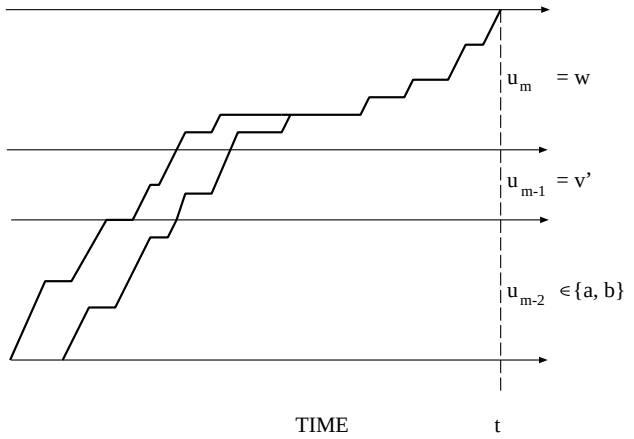
Word Pair Approximation: Illustration

Two examples of word pair approximation:

good example: the predecessor word $u_{m-1} := v$ of word $u_m := w$ is sufficiently long,

bad example: the predecessor word $u_{m-1} := v$ of word $u_m := w$ is too short.





6.5 Algorithm for Word Graph Construction

In words:

- **for each time t, consider all word pairs (v, w) using beam search**
- **for each triple $(t; v, w)$, keep track of:**
  - **the word boundary $\tau(t; v, w)$**
  - **the word score $h(w; \tau(t; v, w), t)$**
- **end of sentence: traceback**

- proceed over time t from left to right
  - **ACOUSTIC LEVEL: process states of lexical trees**
    - initialization: $Q_v(t-1,0) = H(v;t-1)$
      $B_v(t-1,0) = t-1$
    - time alignment: $Q_v(t,s)$ using DP
    - propagate back pointers $B_v(t,s)$
    - prune unlikely hypotheses
    - purge bookkeeping lists
  - **WORD PAIR LEVEL: process word ends**
    - single best: for each pair $(w;t)$ do
      - $H(w;t) = \max_v \{ p(w|v)\, Q_v(t,S_w) \}$
        $v_0(w;t) = \arg\max_v \{ p(w|v)\, Q_v(t,S_w) \}$
      - store best predecessor $v_0 := v_0(w;t)$
      - store best boundary $\tau_0 := B_{v_0}(t,S_w)$
    - word graph: for each triple $(t;v,w)$ store
      - word boundary $\tau(t;v,w) := B_v(t,S_w)$
      - word score $h(w;\tau,t) := Q_v(t,S_w)/H(v;\tau)$
  - **PHRASE LEVEL search (optional)**

Depending on how they are constructed different types of word graphs can be defined:

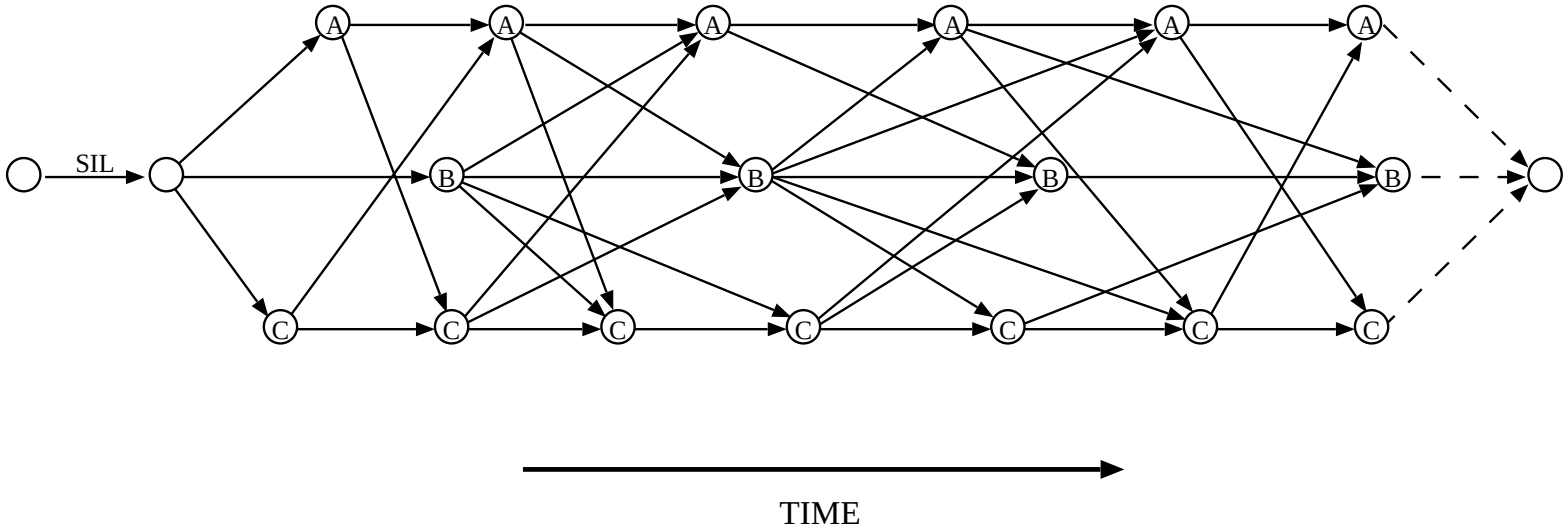
- Word pair approximation $\Rightarrow$ word/predecessor conditioned word graphs
  - The word sequence implies a sequence of word boundaries (unique mapping).
  - Each word sequence occurs only once.
- Other methods $\Rightarrow$ time conditioned word graphs
  - A word sequence might occur more than once with different word boundary times.

Depending on what is to be done with the graph the one or the other method might be suitable.

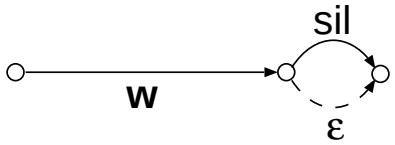
Can it be guaranteed that the M-best sentences are in the word graph?

- It is highly probable, but
- there is no formal proof.

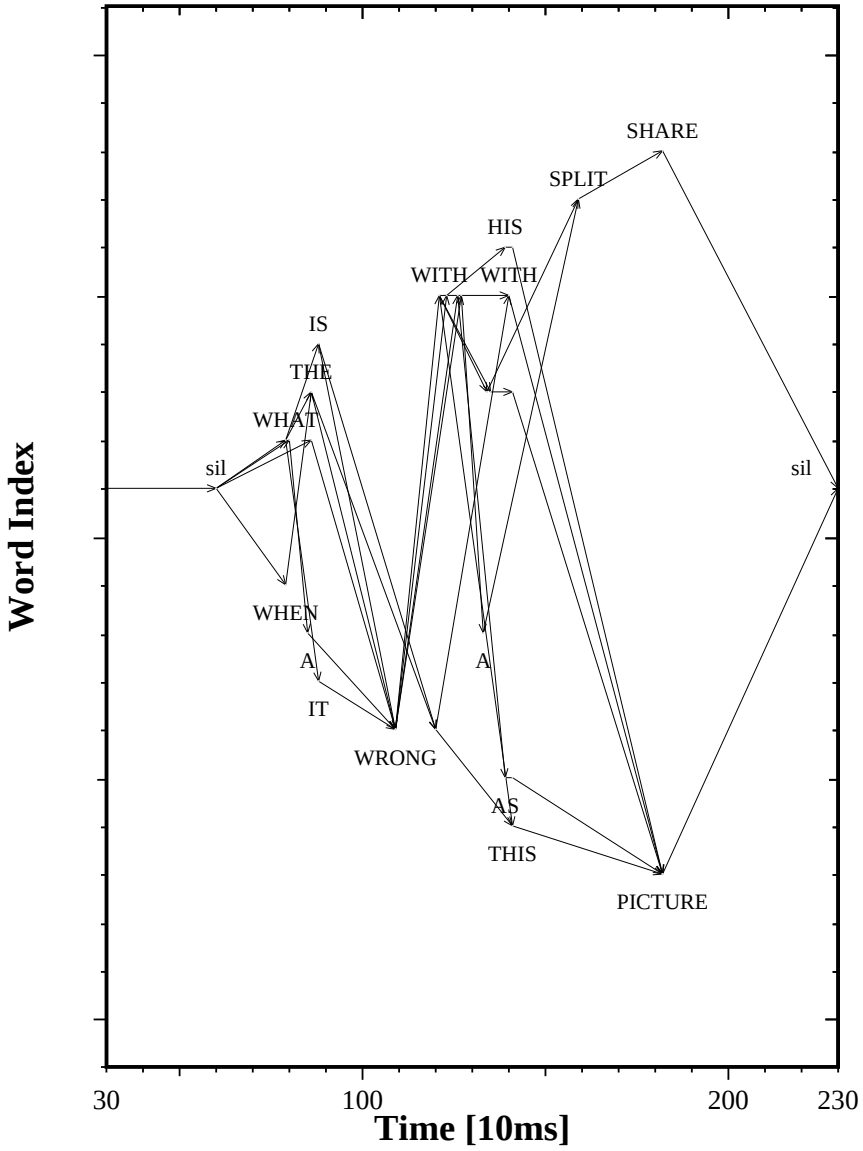
6.6 Algorithm for Word Graph Rescoring



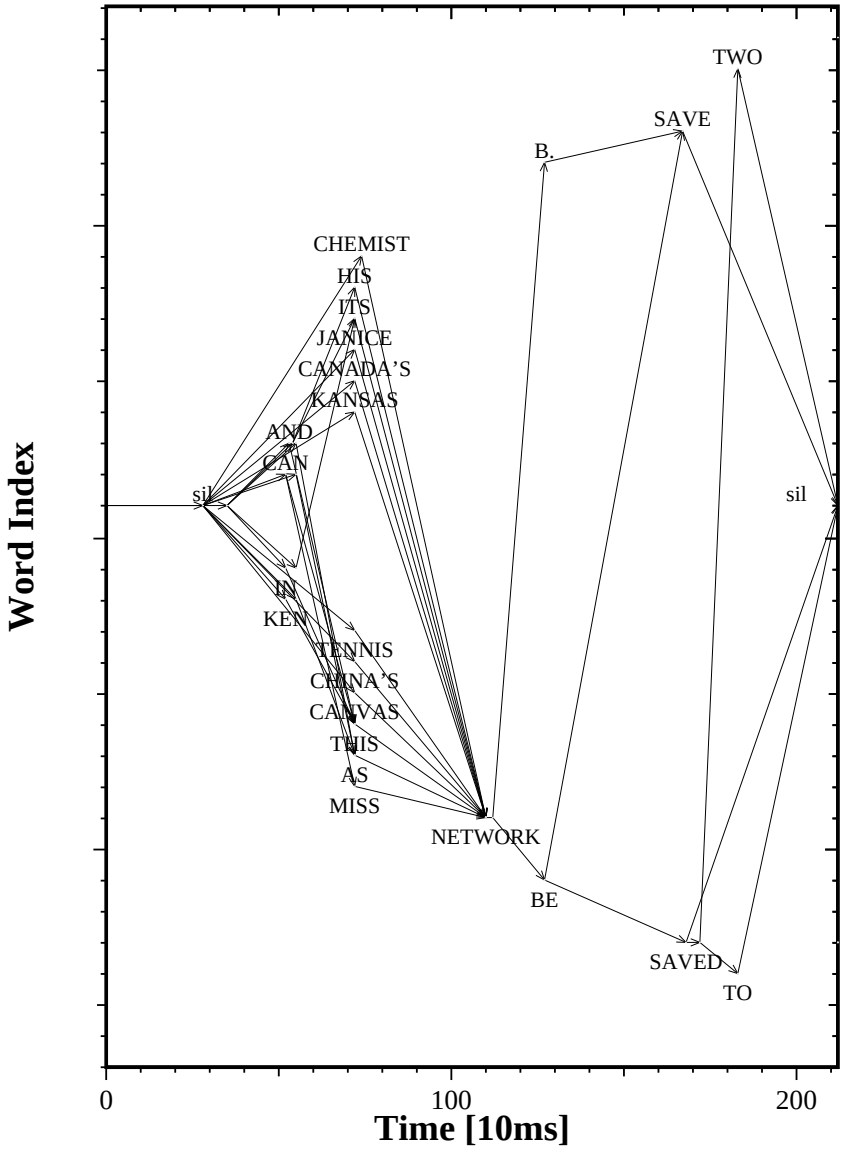
- **incoming arcs: maximum = vocabulary size**
- **outgoing arcs: no maximum**
- **silence is included as integrated part of the words**
- **search: DP through graph**



spoken: what is wrong with this picture?



spoken: can this network be saved?



Word Graph Rescoring Algorithm (2nd pass):

input:	word graph set of word boundaries $\tau(t; u_1^m)$ set of word scores $h(u_m; \tau, t)$
proceed over time t from left to right	
	process each word pair u_{m-1}^m in the graph
	get word boundaries $\tau(t; u_1^m)$ and scores $h(u_m; \tau, t)$
	process all word sequences u_1^m
	$\widehat{H}(u_1^m; t) := p(u_m \mid u_1^{m-1}) \cdot H(u_1^{m-1}; \tau) \cdot h(u_m; \tau, t)$
	$H(u_2^m; t) = \max_{u_1} \widehat{H}(u_1^m; t)$ $B(u_2^m; t) = \arg\max_{u_1} \widehat{H}(u_1^m; t)$
traceback: use back pointers $\{B(u_2^m; t)\}$	

Note: the language model used here is different from the one used to construct the word graph.

6.7 Two Pass vs. Single Pass Method

(A) two pass search (word conditioned bigram search with word graph construction and subsequent word graph rescoring with a trigram).

(B) single pass time conditioned trigram search (chapter 8)

- NAB H1 development corpus; 64k vocabulary
- 310 test sentences, 7387 words
- Bigram LM with a perpexity of 237
- Trigram LM with a perpexity of 172

Number of:	sentences	word errors
		two pass (A) / single pass (B)
identical score	220	452 / 452
(A) is better than (B)	1	2 / 2
(B) is better than (A)	89	439 / 423
total	310	893 / 877

6.8 Lattice Pruning

Pruning methods can also be applied on word graphs.

- Forward Pruning:
  - Word end pruning retains word hypothesis $(v, w; t)$ whose scores are relatively close to score of the best word hypothesis at time t:

$$H_{max}(t) \quad = \quad \max_{w} \{ H(w;t) \}$$

with

$$H(w;t) \quad = \quad \max_{v} \{ p(w|v) \cdot H(v;\tau(t;v,w)) \cdot h(w;\tau(t;v,w),t) \} \ ,$$

prune hypothesis $(v, w; t)$ iff:

$$p(w|v) \cdot H(v;\tau(t;v,w)) \cdot h(w;\tau(t;v,w),t) < f_{LAT} \cdot H_{max}(t) \quad .$$

$f_{LAT} < 1$ is the lattice pruning threshold.

  - Histogram pruning limits the number of surviving word hypotheses to a maximum number (typically 1000).

- **Forward-Backward Pruning:**
  for every edge $(w;\tau,t)$ in the word graph the score $Q(w;\tau,t)$ of the best path trough the word graph including this arc is calculated.
  The arc is pruned if:

$$Q(w;\tau,t) \quad < \quad f_{LAT\text{-}FB} \cdot \max_{(w;\tau,t)} \{Q(w;\tau,t)\}$$

  $f_{LAT\text{-}FB} < 1$ is the forward-backward pruning threshold.

  To calculate $Q(w;\tau,t)$ two dynamic programming search passes on the word graph are needed:

  1. forward pass from $t = 1$ to $t = T$
  2. backward pass from $t = T$ to $t = 1$

  $Q(w;\tau,t)$ is the sum of the forward and backward scores.

6.9 Experimental Results

Experimental results NAB'94 (ARPA Benchmark 1994)

- **NAB'94 (development 20k)**
- **W = 20 000 word vocabulary**
- **199 Out of vocabulary words (2.7%)**
  - **20 speakers (10 female, 10 male)**
  - **310 sentences (155 per gender)**
  - **7387 spoken words**
- **recognizer setup**
  - **2001 context dependent phoneme models (gender independent)**
  - **216000 densities**
  - **trigram language model: perplexity 121.8**
  - **single best word error rate: 14.5 %**

GER[%]: graph error rate. Error rate of the hypothesis in the word graph that has the smallest Levenshtein distance to the spoken word sequence.

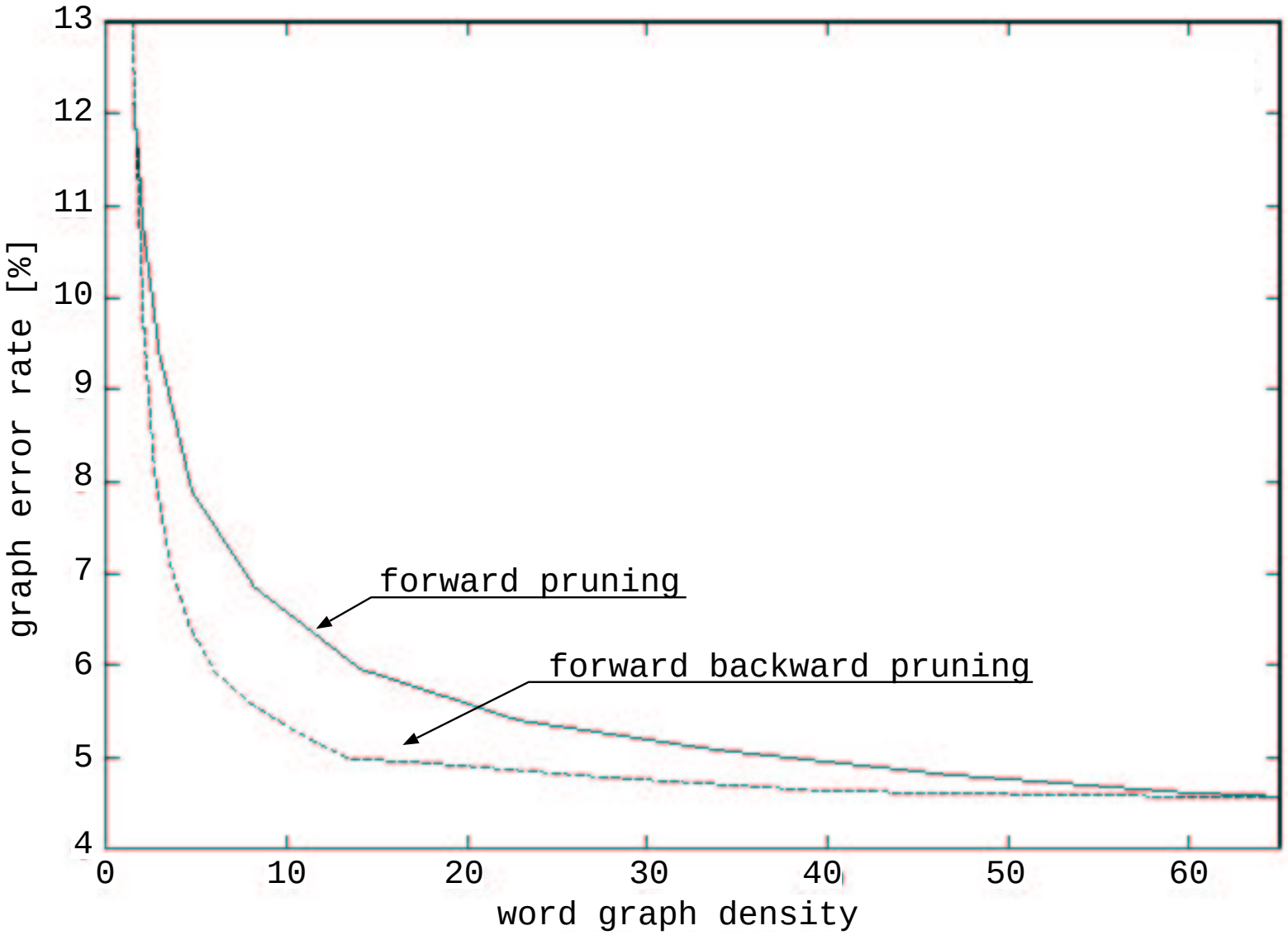
Forward Pruning

$-logf_{Lat}$	per spoken word: arcs	nodes	word boundaries	GER [%]
500.00	105.07	45.70	10.54	4.44
300.00	99.74	45.12	10.20	4.44
250.00	89.11	43.69	9.50	4.45
200.00	71.87	40.85	8.39	4.51
180.00	59.31	35.55	7.42	4.62
160.00	46.52	29.34	6.35	4.81
140.00	33.74	22.40	5.20	5.08
120.00	22.86	15.96	4.20	5.40
100.00	14.07	10.31	3.29	5.96
80.00	8.23	6.36	2.59	6.85
60.00	4.78	3.89	2.10	7.89
40.00	2.97	2.55	1.76	9.39
20.00	2.05	1.85	1.54	10.79
10.00	1.77	1.65	1.45	11.63
5.00	1.64	1.56	1.40	11.90
1.00	1.54	1.48	1.36	12.12

Forward-Backward Pruning

$F_{Lat\text{-}fb}$	per spoken words			GER [%]
	arcs	nodes	word boundaries	
750.00	97.07	43.11	9.96	4.51
500.00	82.47	39.92	9.96	4.53
300.00	39.98	23.52	7.34	4.63
250.00	27.55	17.21	6.00	4.78
200.00	16.95	11.26	4.58	4.95
180.00	13.45	9.17	4.03	4.98
160.00	10.51	7.36	3.54	5.28
140.00	8.03	5.78	3.08	5.58
120.00	6.06	4.49	2.66	5.92
100.00	4.58	3.51	2.32	6.44
80.00	3.48	2.77	2.03	7.16
60.00	2.68	2.23	1.79	8.11
40.00	2.09	1.82	1.59	9.67
20.00	1.65	1.51	1.42	11.87
10.00	1.45	1.38	1.33	13.04
5.00	1.35	1.31	1.29	13.75
1.00	1.25	1.25	1.24	14.34

Word Graph Method: Experimental Results



Test: NAB'94 – H1 Development

- vocabulary: 20000 words
- 10 male and 10 female speakers
- total of 310 sentences = 7387 spoken words
- unknown words: 199 (out of vocabulary rate: 2.7 %)
- test set perplexities:

  bigram: $PP_{bi} = 198.1$

  trigram: $PP_{tri} = 130.2$

Acoustic Modelling:

- trained on: WSJ0 and WSJ1 (284 speakers; 80 h of speech)
- phoneme models: 4688 (43+1 monophones, 557 diphones, 4087 triphones)
- tied states: 4623 (6-state HMM)
- densities: 290 000 per gender

WORD GRAPH METHOD: EXPERIMENTAL RESULTS

Search space:

- **search space: 27672 states, 7674 arcs, 115 trees**

Results: 2.7% out of vocabulary words

WGD: word graph density (word arcs per spoken word)

NGD: node graph density (nodes per spoken word)

BGD: boundary graph density
(different word boundaries per spoken word)

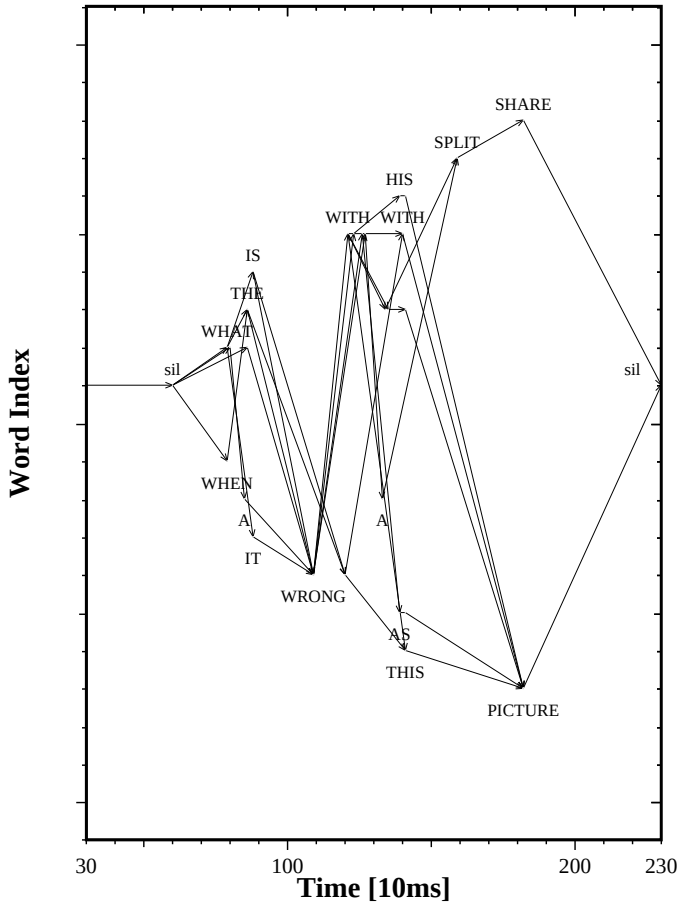
GER[%]: word graph error rate ('sub,del,ins')

WER[%]: word recognition error rate
(trigram: $PP_{tri} = 130.2$)

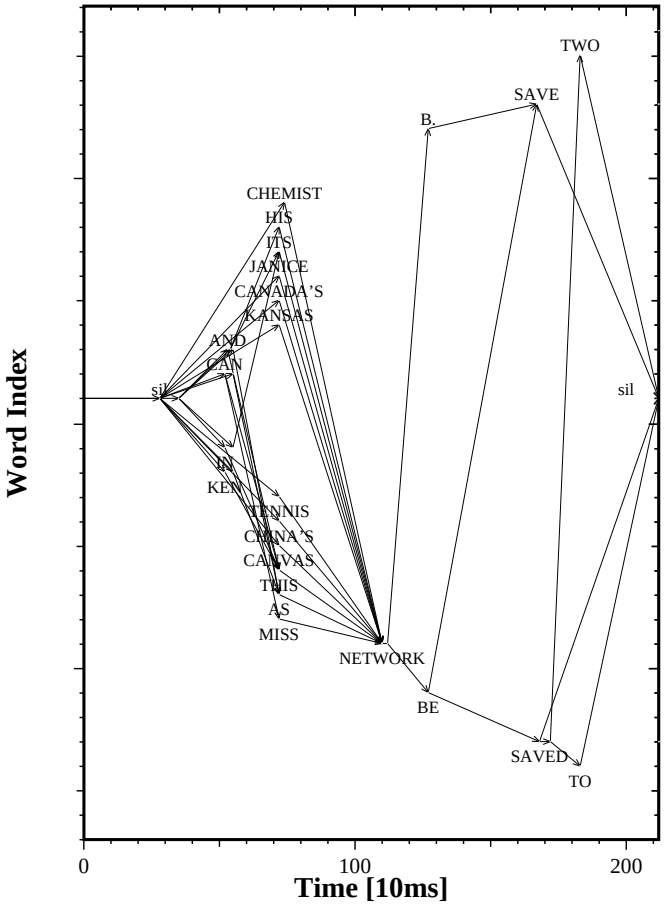
Word Graph Method: Experimental Results

f_{Lat}	Graph density			Graph word error rate		Recognition word error rate	
	WGD	NGD	BGD	del - ins	GER[%]	del - ins	WER[%]
300	1476	181.80	19.67	9 - 38	4.2	120 - 202	14.3
150	1415	175.93	19.25	9 - 38	4.2	120 - 202	14.3
100	684	104.22	14.50	13 - 38	4.3	120 - 202	14.3
90	460	77.35	12.31	12 - 38	4.3	120 - 202	14.3
80	269	51.75	9.84	15 - 44	4.5	120 - 202	14.3
70	137	31.43	7.42	19 - 51	4.8	120 - 202	14.3
60	60	17.20	5.33	23 - 56	5.1	122 - 201	14.3
50	25	8.98	3.77	40 - 64	5.8	124 - 200	14.3
40	10	4.79	2.66	50 - 81	6.8	122 - 202	14.3
30	4	2.75	2.01	84 - 94	8.4	135 - 191	14.6
20	2	1.83	1.60	113 - 129	10.6	147 - 184	14.8
10	1	1.41	1.36	156 - 160	13.8	173 - 180	15.6
5	1	1.29	1.28	168 - 177	15.3	181 - 190	16.3
1	1	1.25	1.24	167 - 192	16.3	167 - 195	16.4

spoken: what is wrong with this picture?



spoken: can this network be saved?



Recognition Results: Word Graph vs. Integrated Method

Test Conditions:

- **NAB'94 – H1 dev. set:**
  310 sentences = 7387 words

Number of:	sentences	word errors
		integrated / word graph
with identical score	277	898 / 898
with better score	32	121 / 146
with worse score	1	11 / 11
Total	310	1030 / 1055

Recognition Results: Word Graph vs. Integrated Method

Examples of spoken test sentences (SNT):

- **integrated search method (ISM)**
- **word graph method (WGM)**

SNT:	... A FEW MUTUAL FUND INVESTORS *INTO* MODELS OF ...
WGM:	... A FEW MUTUAL FUND INVESTORS *IN TWO* MODELS OF ...
ISM:	... A FEW MUTUAL FUND INVESTORS *INTO* MODELS OF ...
SNT:	... WHEN YOU BUY A MUTUAL FUND FROM *YOUR* BROKER
WGM:	... WHEN YOU BUY A MUTUAL FUND FROM *A* BROKER
ISM:	... WHEN YOU BUY A MUTUAL FUND FROM *YOUR* BROKER
SNT:	I ALMOST WISH I HADN'T SEEN *THIS* PART
WGM:	I ALMOST WISH I HADN'T SEEN *AS* PART
ISM:	I ALMOST WISH I HADN'T SEEN *THIS* PART

Cache (unigram/bigram) model:

- language model which is difficult to handle in an integrated search strategy
- 'cache' was updated after each sentence boundary (not after each word!)
- two modes: supervised/unsupervised

Language	model	PP	del - ins	WER [%]
Bigram:	without cache	198.1	179 - 195	16.5
	with cache: unsupervised	178.0	190 - 179	16.3
	supervised	171.2	194 - 179	16.0
Trigram:	without cache	130.2	120 - 202	14.3
	with cache: unsupervised	117.1	127 - 182	14.0
	supervised	113.1	128 - 194	13.8

Search effort: Word Graph vs. Integrated Method

Test Conditions:

- **subset of NAB'94 H1 development set:**
  = 10 male and 10 female speakers,
  = 20 sentences = 519 spoken words
  (with 21 out of vocabulary words)
  = 211 sec
- **perplexities:**
  $PP_{bi} = 201.3$, $PP_{tri} = 134.1$
- **SGI workstation (Indy R4400, SpecInt'92: 94)**

Search effort: Word Graph vs. Integrated Method

	Word Graph Method		Integrated Method	
Search space: states/arcs/trees	16274 / 4516 / 41		32291 / 9413 / 108	
Time	**[sec]**	**[%]**	**[sec]**	**[%]**
Log-likelihood computation	14830	78.2	15174	67.2
Acoustic search	3603	19.0	6576	29.1
Lang. model recombination	139	0.7	621	2.7
Word graph rescoring	210	1.1	–	–
Other operations	177	1.0	208	1.0
Overall recognition	18959	100.0	22579	100.0
Real time factor	90	-	107	-

Conclusions: Word Graph Method

- specification of the word graph problem
- word pair approximation
- efficient method by extending the word conditioned one-pass algorithm
- modification: bookkeeping only
- experimental results for the 20,000 WSJ/NAB task: 4 to 25 word arcs per spoken word
- comparison with integrated search: negligible degradation
- note: only 'forward' pruning using bigram LM

7 Across-Word Models

7.1 Motivation for Across-Word Models 361
7.2 Implementation of Across-Word Models 364
7.3 Extension of the Tree Lexicon 365
7.4 Extension of the Language Model Recombination 367
7.5 Activating the First Phoneme Generation 370
7.6 Recombination after the First Phoneme Generation 372
7.7 Dynamic Programming Recursion 374
7.8 Extended Language Model Look-Ahead 376
7.9 Experimental Results 378

7.1 Motivation for Across-Word Models

- The words of the vocabulary are modeled with phonemes (example: Airline ae r l ah ih n).
- The acoustic realization depends on the context of a phoneme (coarticulation).
- This variability is modeled with triphones, i.e. phonemes in a triphone context.
- A model for phoneme a with predecessor x and successor y ($_x a_y$) is different from the model of a with predecessor w and successor z ($_w a_z$).

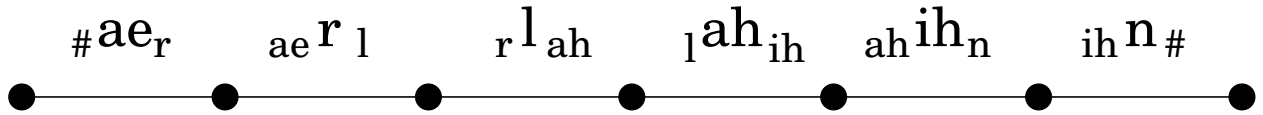
- The models for words consist of concatenated phoneme models.
  When using triphones two principle ways of modelling are possible:

  - Triphones are only used within words. At word boundaries the so called "word boundary context" is used (conventional within word modelling).
    . Advantage: easy implementation of the modelling; word models can be taken directly from the lexicon.
    . Disadvantage: coarticulation at word boundaries is not modelled well.

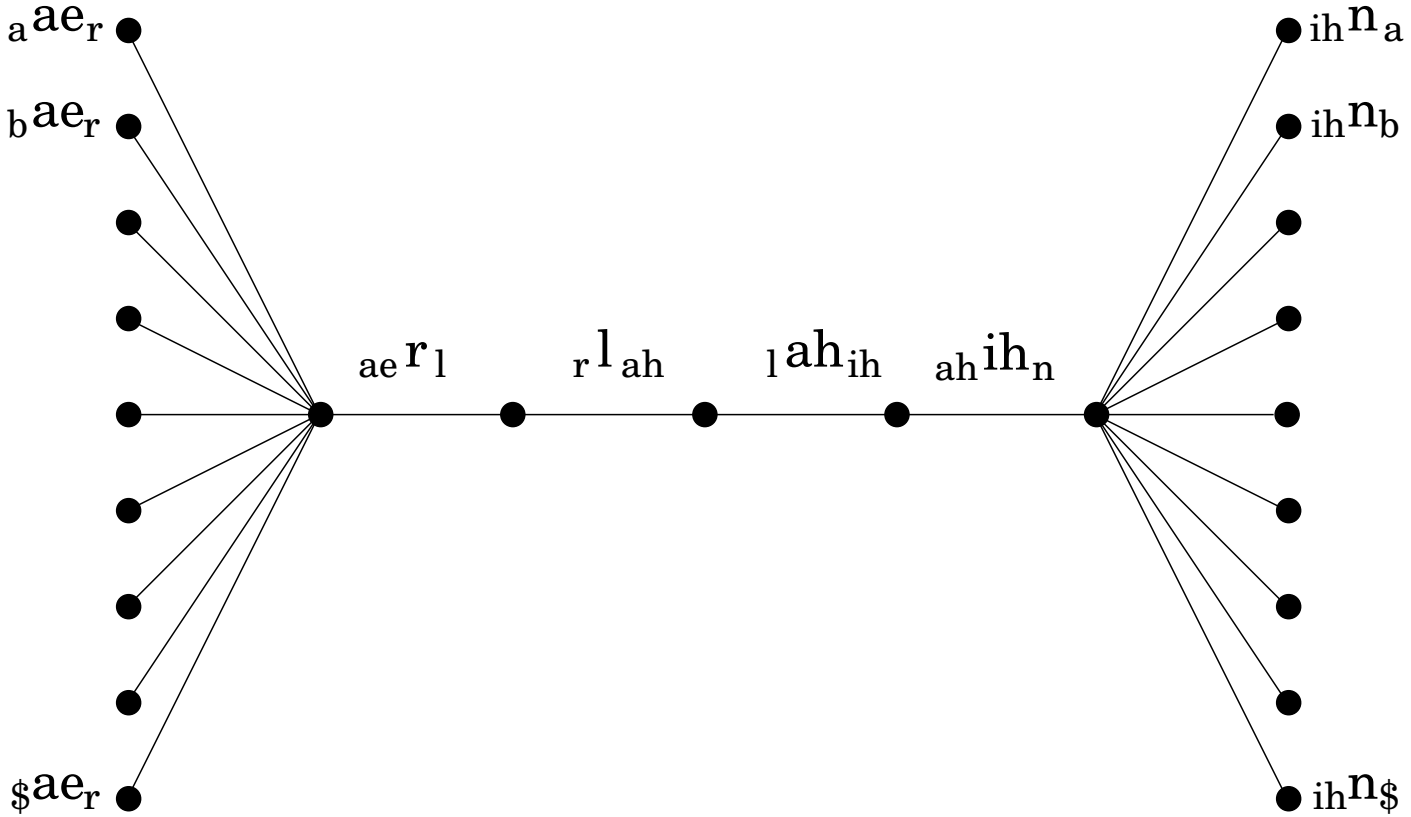
  - Triphones are also used at word boundaries (across-word modelling)
    . Advantage: coarticulation at word boundaries is taken into consideration.
    . Disadvantage: more search effort is required. The word models depend on the predecessor and successor words.

- **Example for the word:** **Airline** **ae r l ah ih n**

- **Transcription with context dependencies only within the word:**



- **Transcription with context dependent models within the word and at the boundaries:**

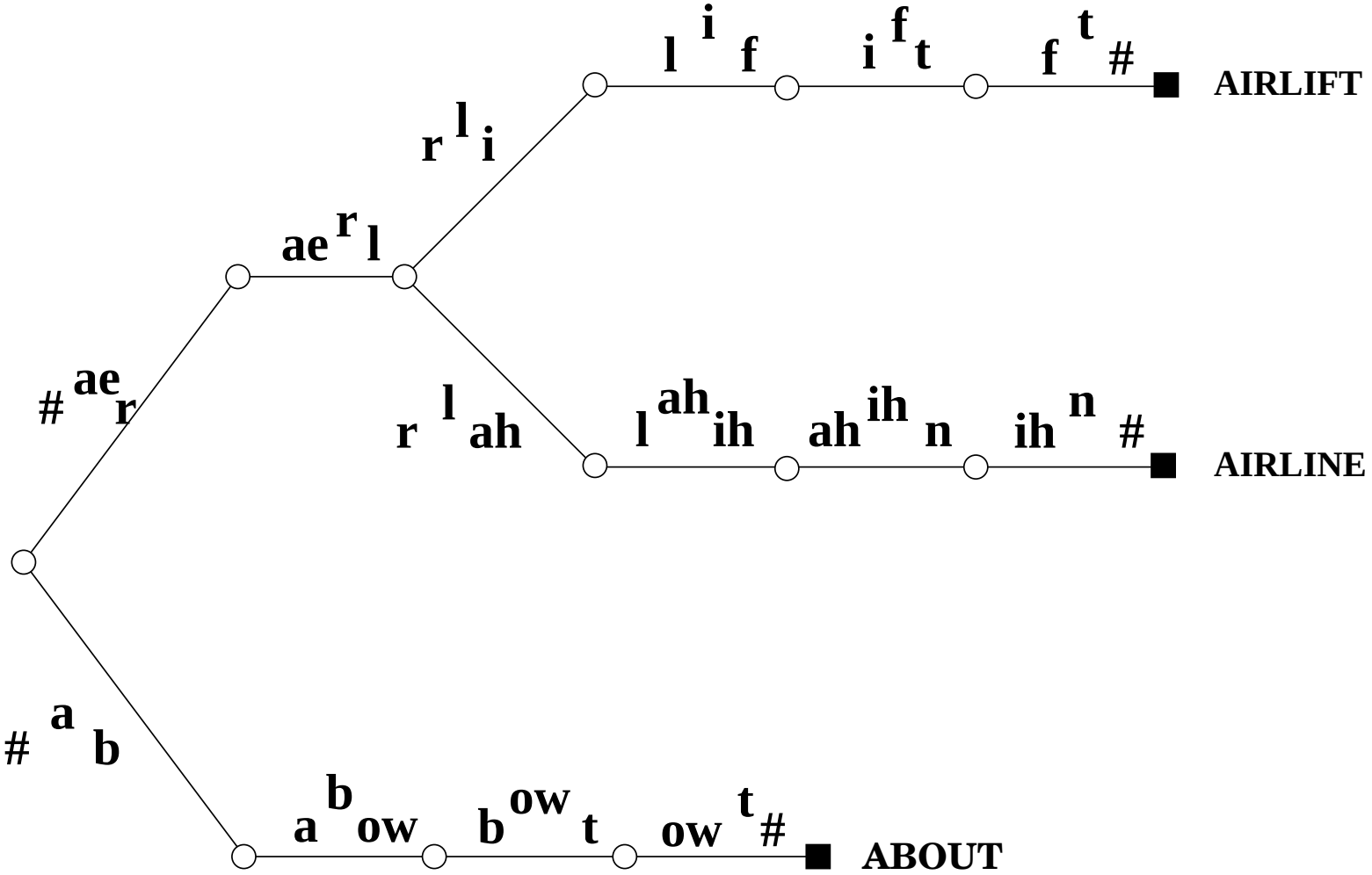


7.2 Implementation of Across-Word Models

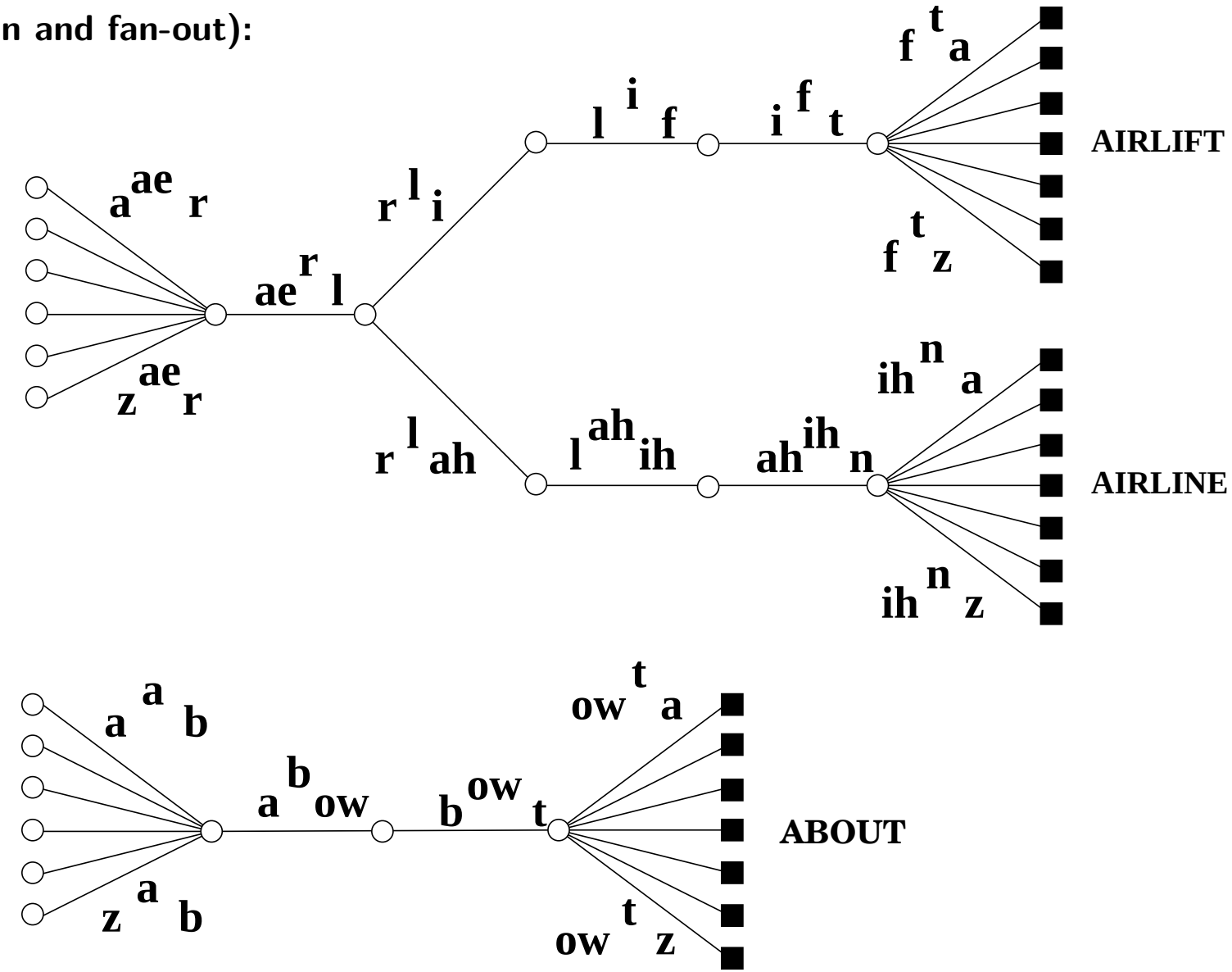
- Starting point: word conditioned tree search, context dependent models only within words.
- Two principle possibilities of using across-word models:
  1. multi pass search:
     - generate a word graph without across-word models
     - extract the N best word sequences from the graph
     - rescore the N best hypothesized sentences with across-word models
  2. single pass search:
     - extend word conditioned tree search to allow processing of context dependencies beyond word boundaries
       . extend the tree lexicon
       . extend the language model recombination
- In the following the single pass search strategy will be described.

7.3 Extension of the Tree Lexicon

- The conventional tree lexicon only uses within-word models.
  At word boundaries the "word boundary context" is used:



- To model coarticulation across word boundaries the word beginnings and ends are expanded (fan-in and fan-out):



7.4 Extension of the Language Model Recombination

Review: language model recombination for within-word modelling with a bigram language model

Notation:

A, E, I : **ending resp. beginning words**

y : **phoneme of the last generation**

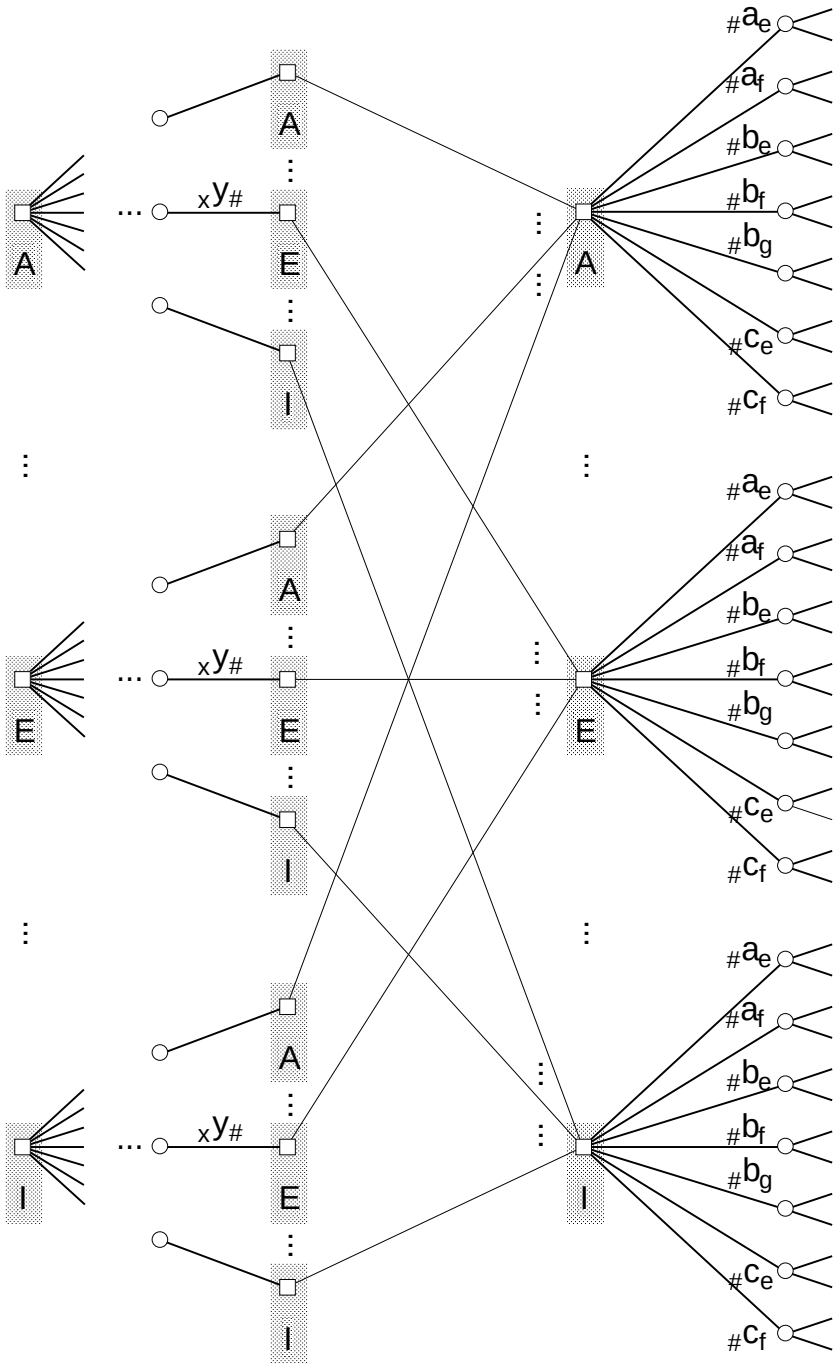
x : **phoneme of the second to last generation**

a, b, c : **phonemes of the first generation**

e, f, g : **phonemes of the second generation**

$\#$: **general word boundary context**

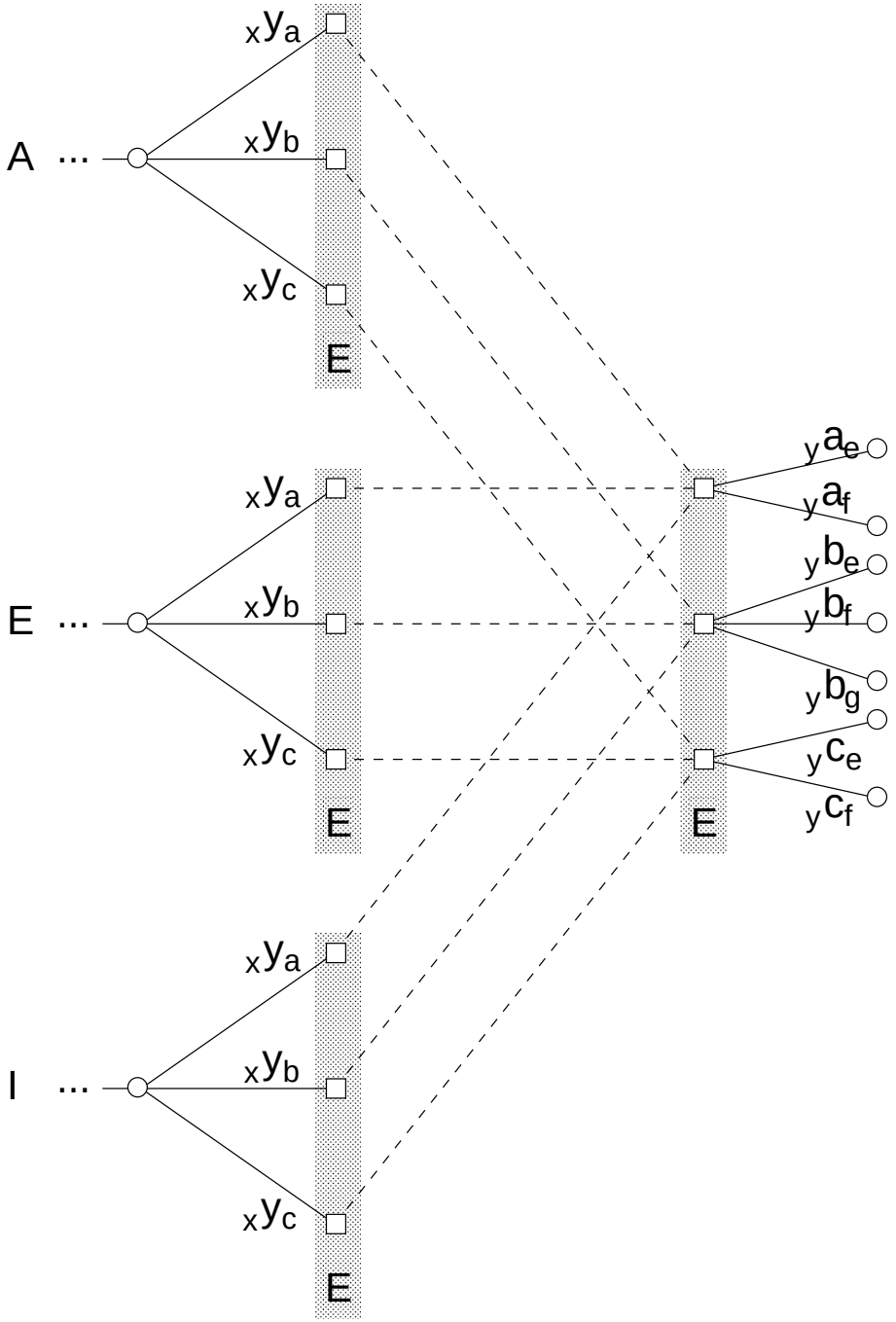
\$: **silence context, for word transitions with silence between the words**



When using across-word models:

- Word endings consist of more than one arc in the lexical prefix tree.
- For every possible right across-word context there is an own arc.
- The hypothesis of a word end is combined with the hypothesis of a successor word.
- The first phoneme of the successor is determined at the ending of a word.
- The hypothesized context influences the future search space.
- Language model recombination can only combine word end hypotheses with the same right across-word context.

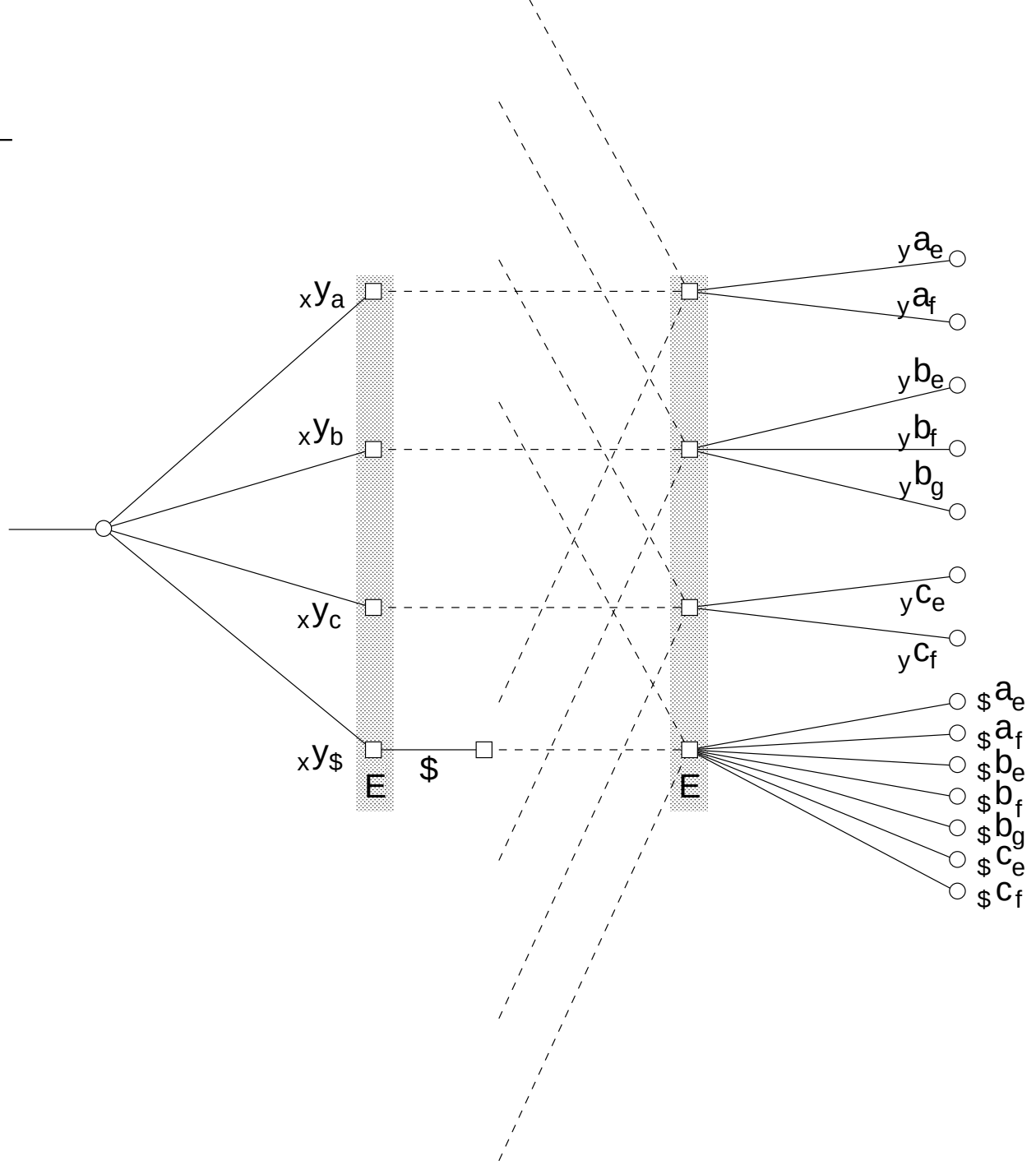
The following picture only shows the ending and beginning of word E with the last phoneme y.


A ...
E ...
I ...
xya
xyb
xyc
E
yae
yaf
ybe
ybf
ybg
yce
ycf

7.5 Activating the First Phoneme Generation

At the root of a tree the set of arcs that will be activated depends on:

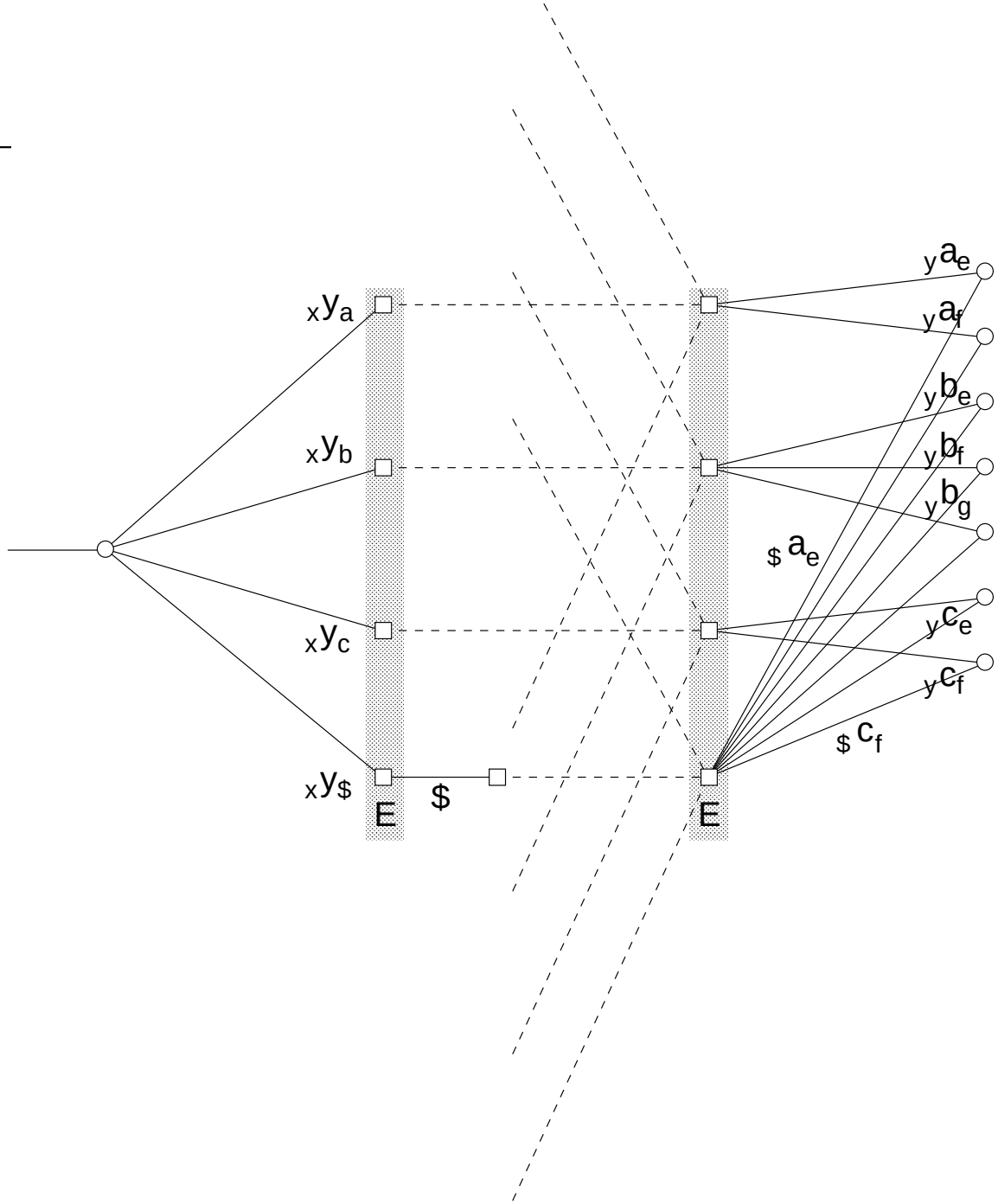
- **the last phoneme of the predecessor word (left context of the new arc)**
- **the right context with which the predecessor word has ended (central phoneme of the new arc).**
- **special case silence ($): it is assumed that the pause between the ending and the starting word is long, so no coarticulation occurs. Activate all arcs of the first phoneme generation with silence as left context.**


$_xy_a$
$_xy_b$
$_xy_c$
$_xy_\$$
E
$
E
$_ya_e$
$_ya_f$
$_yb_e$
$_yb_f$
$_yb_g$
$_yc_e$
$_yc_f$
$\$a_e$
$\$a_f$
$\$b_e$
$\$b_f$
$\$b_g$
$\$c_e$
$\$c_f$

7.6 Recombination after the First Phoneme Generation

The across-word context of the ending word only influences the first phoneme of the successor.

It is possible to recombine the arcs of the non-coarticulation transition with those of the coarticulation transition.


$_{x}y_{a}$
$_{x}y_{b}$
$_{x}y_{c}$
$_{x}y_{\$}$
E
$
E
$_{y}a_{e}$
$_{y}a_{f}$
$_{y}b_{e}$
$_{y}b_{f}$
$_{y}b_{g}$
$_{\$}a_{e}$
$_{y}c_{e}$
$_{y}c_{f}$
$_{\$}c_{f}$

7.7 Dynamic Programming Recursion

- **Within-word modelling:**

  1. **within words:**

$$Q_v(t,s) \quad = \quad \max_{\sigma} \left\{ p(x_t, s|\sigma) \cdot Q_v(t-1,\sigma) \right\}$$

  2. **word boundaries:**

$$Q_v(t,s=0) \quad = \quad \max_{u} \left\{ p(v|u) \cdot Q_u(t,S_v) \right\}$$

- **Across-word modelling:**

  1. **within words (as before):**

$$Q_v(t,s) \quad = \quad \max_{\sigma} \{p(x_t, s|\sigma) \cdot Q_v(t-1,\sigma)\}$$

  2. **word boundaries:**

     The right across-word context has to be considered during language model recombination of the ending word v.

$$Q_v(t, s_0(\chi)) \quad = \quad \max_{u} \{p(v|u) \cdot Q_u(t, S_{(v,\chi)})\}$$

with χ := **right context of the predecessor word** v

$S_{(v,\chi)}$:= **last state in the fan-out arc of word** v **with right context** χ

$s_0(\chi)$:= **virtual starting state of the tree that follows the word end hypothesis** v **with context** χ**. There is only one virtual starting state when using within-word models, for across-word modelling a different virtual starting state** ($s_0(\chi)$) **for every right context** χ **of the predecessor word** v **has to be considered.**

7.8 Extended Language Model Look-Ahead

- The hypothesis of a word end with a certain right across-word context restricts the set of possible successor words to the set of words which begin with the phoneme that was hypothesized as right context.

  This restricts the set of possible language model probabilities and allows additional pruning:

$$W(\chi) := \text{set of words, with } \chi \text{ as first phoneme.}$$

  For every ending word v and every possible right context χ the maximal language model probability at the end of a (yet unknown) word w following v can be calculated.

$$\pi(v, \chi) \quad := \quad \max_{w \in W(\chi)} p(w|v)$$

- This maximal probability can be considered as soon, as the right context χ_r at the end of the predecessor word v is hypothesized.
  Let $\Sigma(v, \chi)$ be the set of states in a fan-out arc of an ending word hypothesis v, with right context χ.
  The score $Q_u^{FO}(t, \tilde{s})$ for a state $\tilde{s} \in \Sigma(v, \chi)$ within a fan-out arc of word v, with right context χ in the tree of the predecessor word u then is:

$$Q_u^{FO}(t, \tilde{s}) \quad := \quad \pi(v, \chi) \cdot Q_u(t, \tilde{s})$$

  A state in the arc of an ending word is pruned if:

$$Q_u^{FO}(t, \tilde{s}) \quad < \quad f_{AC,FO} \cdot Q^{FO}(t)$$

  with

$$Q^{FO}(t) \quad := \quad \max_{u,v,\chi} \left\{ Q_u^{FO}(t, \tilde{s}) : \tilde{s} \in \Sigma(v, \chi) \right\}$$

  $0 < f_{AC,FO} < 1$ is the fan-out pruning threshold.

7.9 Experimental Results

- Verbmobil eval'96, 5k vocabulary
- 35 speakers, 305 utterances, 5421 words
- within-word models (gender independent):
  2001 mixtures, 195451 densities
- across-word models (gender independent):
  3001 mixtures, 209484 densities
- trigram perplexity 36.4
- WITHOUT extended language model look-ahead
- RTF (real time factor) on a 500 Mhz AlphaPC

	search space			error rate [%]			
models	states	arcs	trees	del.	ins.	WER	RTF
within-word	31040	8677	135	3.3	3.2	17.4	46.48
across-word	85729	21932	49	3.7	2.3	15.8	64.04

- NAB'94 H1 development-set, 20k vocabulary
- 20 speaker (10 men and 10 woman), 310 utterances with 7387 words (199 out of vocabulary words)
- within-word models:
  - female: 2001 mixtures, 279079 densities
  - male: 2001 mixtures, 269141 densities
- across-word models:
  - female: 4001 mixtures, 281022 densities
  - male: 4001 mixtures, 354607 densities
- trigram perplexity 121.8
- WITH extended language model look-ahead
- RTF (real time factor) on a 533 Mhz AlphaPC

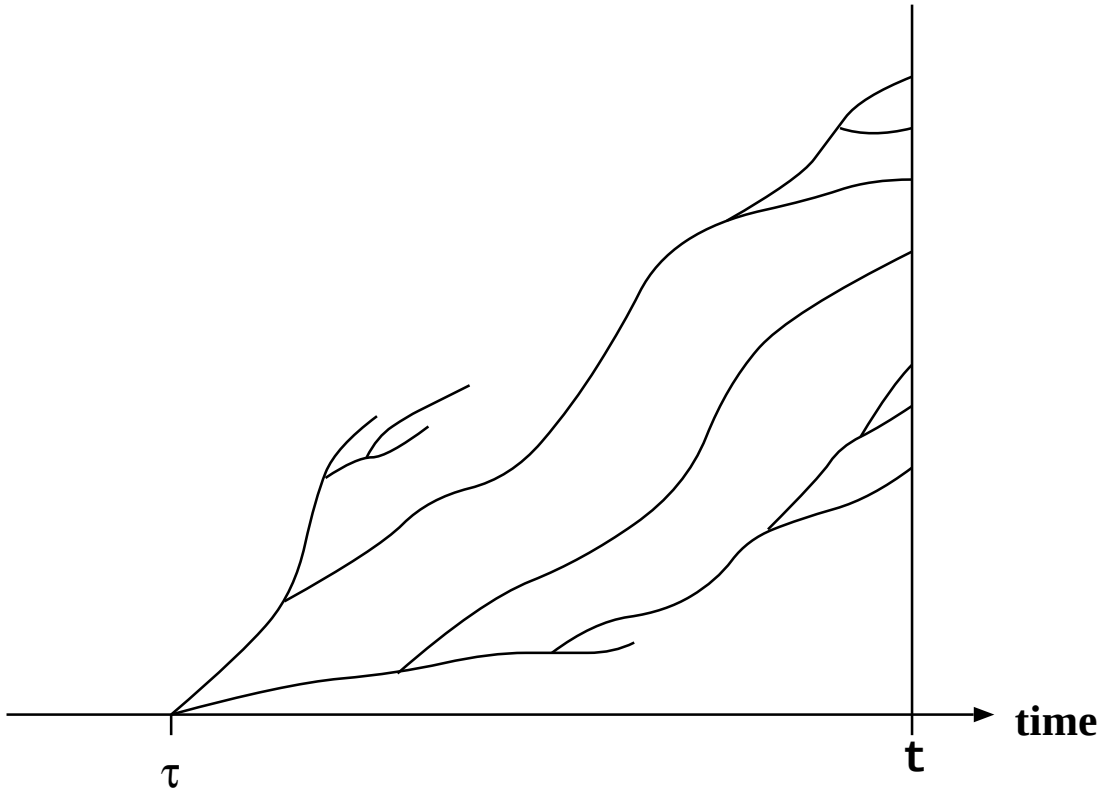
	search space			error rate [%]			
models	states	arcs	trees	del.	ins.	WER	RTF
within-word	19599	5821	136	1.5	2.6	12.9	37.9
across-word	17811	2822	37	1.3	2.4	11.7	36.8

8 Time Conditioned Search

8.1 Idea 381
8.2 Time Conditioned Search: Idea 382
8.3 m-gram language models 385
8.4 Implementation 388
8.5 Extension to Trigram Language Models 392
8.6 Equivalence of Word Conditioned and Time Conditioned Search Organization . . 397

8.1 Idea

At every time τ a separate copy of the lexical prefix tree (time conditioned tree copy) with the hypothesis $Q_\tau(t, s)$ is started.



Possible advantage: using beam search only a few time conditioned tree copies are active (experiments will show 40-50)

8.2 Time Conditioned Search: Idea

- time synchronous evaluation of the Bayes' decision rule as before:

$$Pr(w_1 \ldots w_n) \cdot Pr(x_1 \ldots x_T | w_1 \ldots w_N)$$

- definition

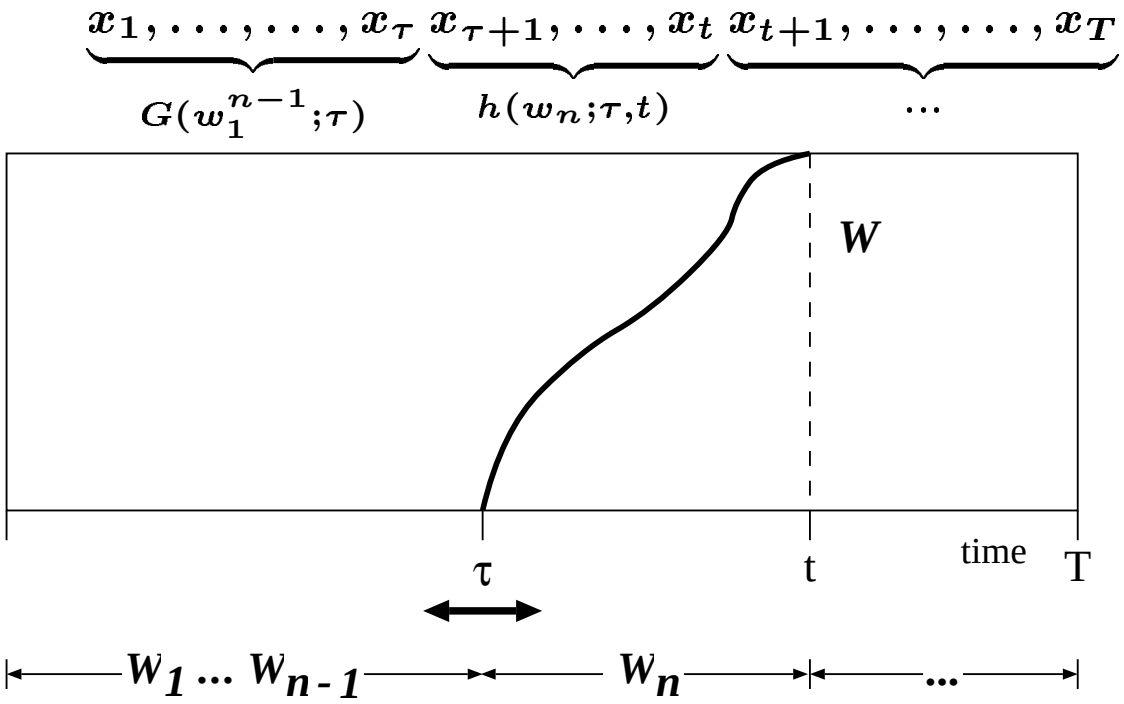
$h(w; \tau, t) := Pr(x_{\tau+1}^t | w)$

$\rightarrow$ probability that word w generated the acoustic vectors $x_{\tau+1}, \ldots, x_t$

$G(w_1^n; t) := Pr(w_1^n) \cdot Pr(x_1^t | w_1^n)$

$\rightarrow$ conditional probability for the generation of the acoustic vectors $x_1, \ldots, x_t$ and a word sequence w_1^n that ends at time t.

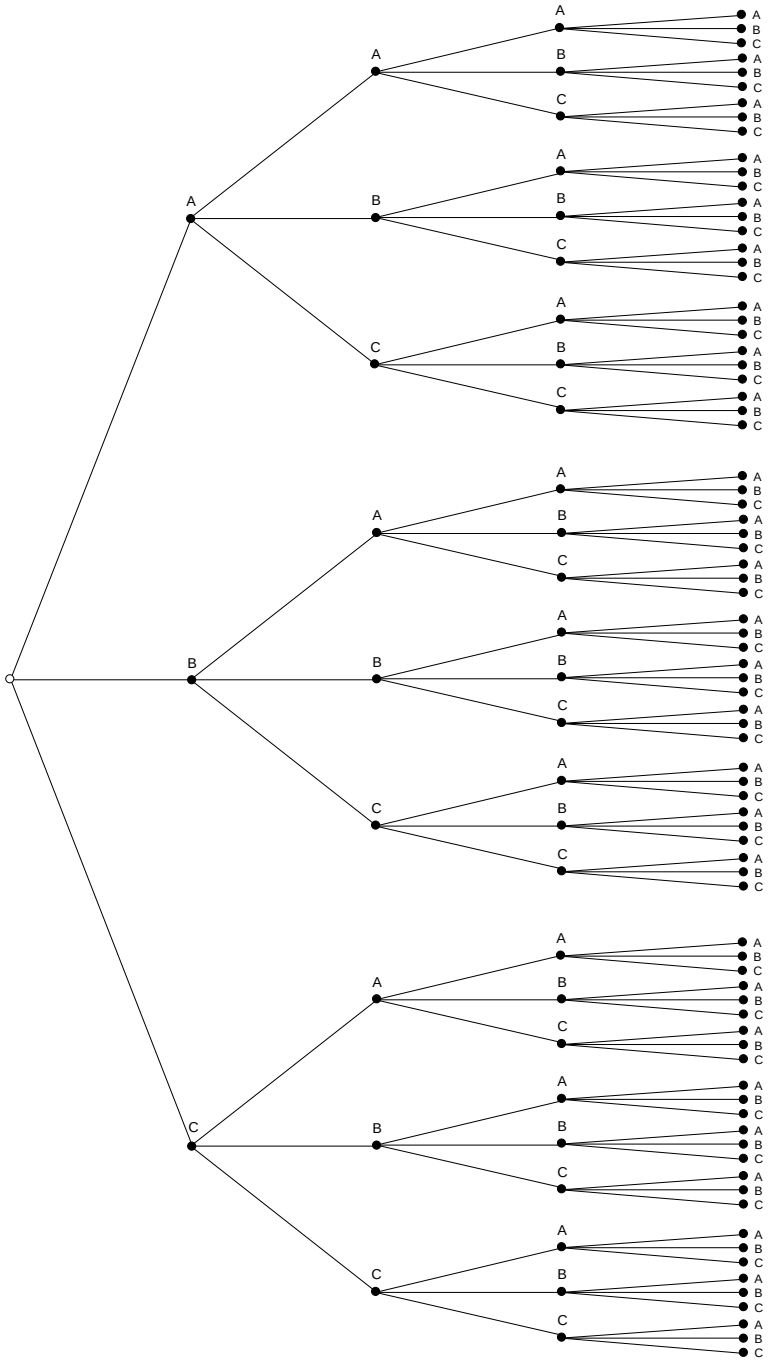
- **decomposition**



- **optimize over word boundary τ**

$$\begin{aligned} G(w_1^n;t) &= \max_\tau \{Pr(w_n|w_1^{n-1}) \cdot G(w_1^{n-1};\tau) \cdot h(w_n;\tau,t)\} \\ &= Pr(w_n|w_1^{n-1}) \cdot \max_\tau \{G(w_1^{n-1};\tau) \cdot h(w_n;\tau,t)\} \end{aligned}$$

- **so far: no optimization of the word boundaries:**
  organization of word successor hypotheses as tree.


A
B
C
A
B
C
A
B
C

8.3 m-gram language models

Consider the m-gram language model $p(u_m|u_1^{m-1})$

- distinguish word sequences by their last $(m-1)$ words u_1^{m-1}
- definition

  $$H(u_2^m;t) = \max_{w_1^m}\{Pr(w_1^n)\cdot Pr(x_1^t|w_1^n) : w_{n-m+2}^n = u_2^m\}$$

  conditional probability for an acoustic vector sequence $x_1 \ldots x_t$ and a word sequence that ends at time t with word u_2^m.

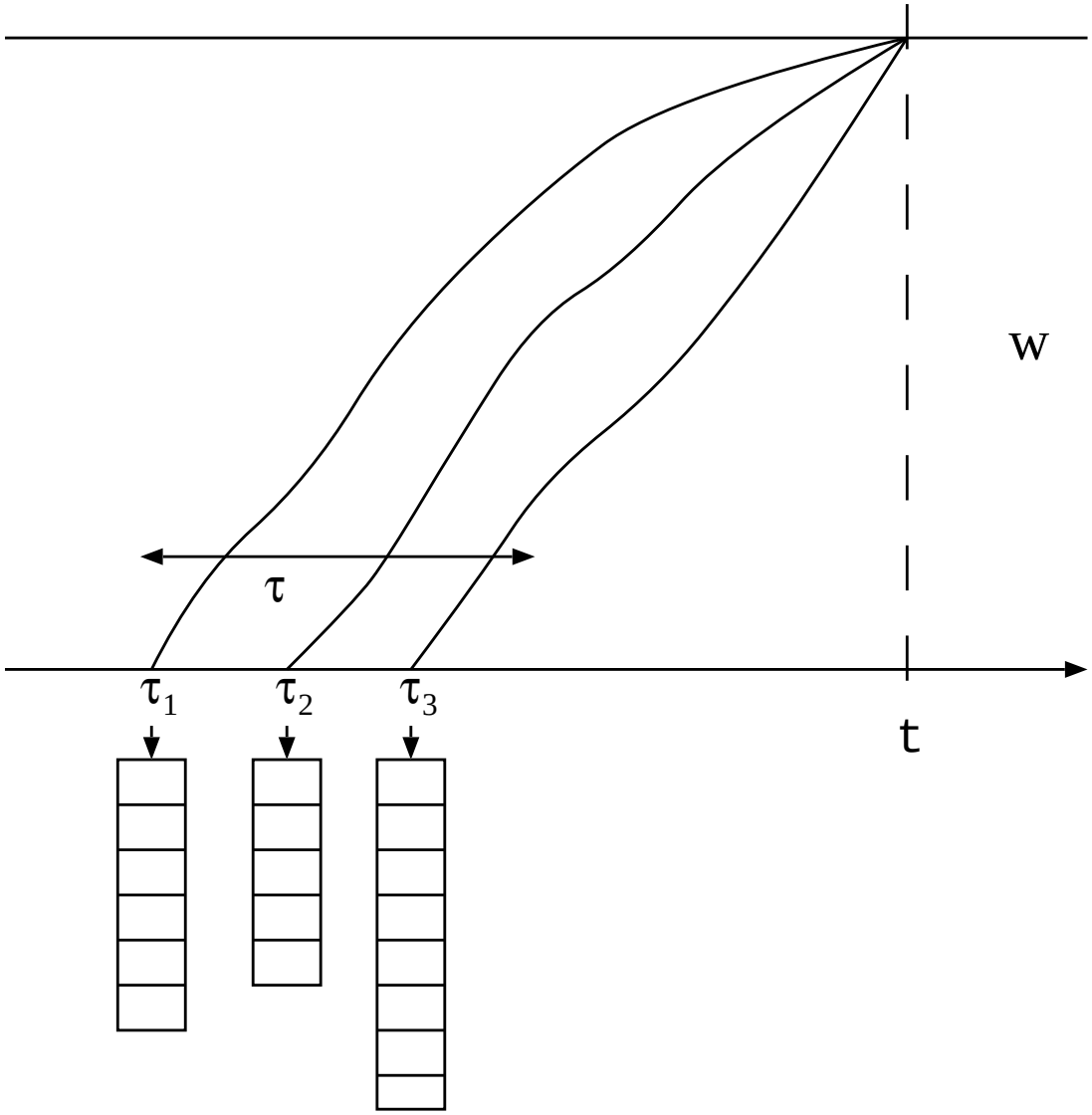
- optimize word ends (DP)

$$H(u_2^m; t) = \max_{u_1}\{p(u_m|u_1^{m-1}) \cdot \max_{\tau}\{H(u_1^{m-1}; \tau) \cdot h(u_m; \tau, t)\}\}$$

- with a bigram language model $p(w|v)$:

$$H(w; t) = \max_{v}\{p(w|v) \cdot \max_{\tau}\{H(v; \tau) \cdot h(w; \tau, t)\}\}$$

Consider the hypothesis that word w ends at time t.



8.4 Implementation

Principle:

- time synchronous evaluation of the hypotheses (as before)
- lexical tree
- start a separate lexical tree at every time frame
- define:

  $Q_\tau(t,s)$ = probability of the best partial path that generates the acoustic vectors $x_1 \ldots x_t$ up to a state s at time t of a lexical prefix tree with start time τ.

- DP recursion for bigram language models

  initialization (best predecessor hypothesis)

$$Q_{t-1}(t-1,s) = \begin{cases} \max_u \{H(u;t-1)\} & : \quad s=0 \\ 0.0 & : \quad s>0 \end{cases}$$

  within words:

$$Q_\tau(t,s) = \max_\sigma \{p(x_t, s|\sigma) \cdot Q_\tau(t-1,\sigma)\}$$

  word boundaries:

$$\begin{aligned} H(w;t) &= \max_v \{p(w|v) \cdot \max_\tau \{H(v;\tau) \cdot \frac{Q_\tau(t,S_w)}{\max_u H(u;\tau)}\}\} \\ &= \max_v \{p(w|v) \cdot \max_\tau \{H(v;\tau) \cdot h(w;\tau,t)\}\} \end{aligned}$$

Properties of the time conditioned search:

- **time synchronous**
- **store the score $H(v;\tau)$ for corrections, modifications of word boundary optimization**
- **separate word boundary optimization over start time and predecessor word**
- **no backpointers**

proceed over time t from left to right

ACOUSTIC LEVEL: process hypotheses $Q_\tau(t,s)$

- initialization:

$$Q_{t-1}(t-1,s) = \begin{cases} \max_u H(u;t-1) & \text{if } s=0 \\ 0.0 & \text{if } s>0 \end{cases}$$

- time alignment: $Q_\tau(t,s)$ using DP recursion

- prune unlikely hypotheses

- purge bookkeeping lists

WORD PAIR LEVEL: process hypotheses $Q_\tau(t,S_w)$

for each pair $(w;t)$ do

- word score: $h(w;\tau,t) = Q_\tau(t,S_w) / \max_u H(u;\tau)$

- best predecessor and best boundary:

$$[v_0,\tau_0](w,t) := \arg\max_{v,\tau} \{p(w|v)\ h(w;\tau,t)\ H(v;\tau)\}$$

- store $[v_0,\tau_0](w,t)$

- tree start-up score:

$$H(w;t) = p(w|v_0(.))\ \ h(w;\tau_0(.),t)\ \ H(v_0(.);\tau_0(.))$$

8.5 Extension to Trigram Language Models

- trigram language model $p(w|uv)$:

$$H(v,w;t) = \max_{u}\{p(w|uv) \cdot \max_{\tau}\{H(u,v;\tau) \cdot h(w;\tau,t)\}\}$$

- DP recursion

  initialization (best word end hypothesis)

$$Q_{t-1}(t-1,s) = \begin{cases} \max\limits_{(u,v)}\{H(u,v;t-1)\} & : \quad s = 0 \\ 0.0 & : \quad s > 0 \end{cases}$$

within words:

$$Q_\tau(t,s) = \max_\sigma \{p(x_t, s|\sigma) \cdot Q_\tau(t-1,\sigma)\}$$

word boundaries:

$$H(v,w;t) = \max_u \left\{ p(w|uv) \cdot \max_\tau \left\{ H(u,v;t) \cdot \frac{Q_\tau(t,S_w)}{\max\limits_{(u',v')} H(u',v';\tau)} \right\} \right\}$$

Test: NAB'94 – H1 Development

- vocabulary: 20000 words
- 10 male and 10 female speakers
- total of 310 sentences = 7387 spoken words
- unknown words: 199 (out of vocabulary rate: 2.7 %)
- test set perplexities:

  bigram: $PP_{bi} = 198.1$

  trigram: $PP_{tri} = 130.2$

Acoustic Modelling:

- trained on: WSJ0 and WSJ1 (284 speakers; 80 h of speech)
- phoneme models: 4688 (43+1 monophones, 557 diphones, 4087 triphones)
- tied states: 4623 (6-state HMM)
- densities: 290 000 per gender

Word conditioned tree search versus time conditioned tree search

word conditioned tree search:

LM	average number of					del - ins	WER[%]
	States	Arcs	Trees	Words	LM Rec		
Bigr.	5022	1416	20	86	86	2.5 - 2.7	17.1
PP:	9335	2593	28	156	156	2.5 - 2.7	16.8
198	26493	7222	45	540	540	2.4 - 2.6	16.5
	39146	10528	53	610	610	2.4 - 2.6	16.4
	52786	13968	60	560	560	2.4 - 2.6	16.4
Trigr.	4714	1392	29	80	80	1.7 - 2.9	14.8
PP:	18734	5430	70	294	294	1.6 - 2.8	14.2
122	48940	13877	112	702	702	1.6 - 2.8	14.1
	67541	18764	130	953	953	1.6 - 2.7	14.0
	86772	23688	145	1223	1223	1.6 - 2.7	13.9

time conditioned tree search:

LM	average number of					del - ins	WER[%]
	States	Arcs	Trees	Words	LM Rec		
Bigr.	6175	1781	27	117	1539	2.4 - 2.9	17.3
PP:	11280	3101	31	197	2976	2.4 - 2.7	16.7
198	30019	7998	37	520	8761	2.4 - 2.6	16.5
	44289	11679	41	777	13162	2.4 - 2.6	16.5
	60692	15877	43	1075	18081	2.4 - 2.6	16.4
Trigr.	8573	2459	28	136	1477	1.6 - 2.9	14.5
PP:	15103	4194	31	234	2806	1.6 - 2.8	14.1
122	24914	6780	34	388	4858	1.6 - 2.7	14.0
	38323	10299	38	606	7663	1.6 - 2.7	14.0
	71757	18953	42	1202	14293	1.6 - 2.7	14.0

From: S. Ortmanns, H. Ney, F. Seide, I. Lindam, "A Comparison of Time Conditioned and Word Conditioned Search Techniques for Large Vocabulary Speech Recognition," Proc. Int. Conf. on Spoken Language Processing, Philadelphia, PA, pp. 2091-2094, October 1996.

8.6 Equivalence of Word Conditioned and Time Conditioned Search Organization

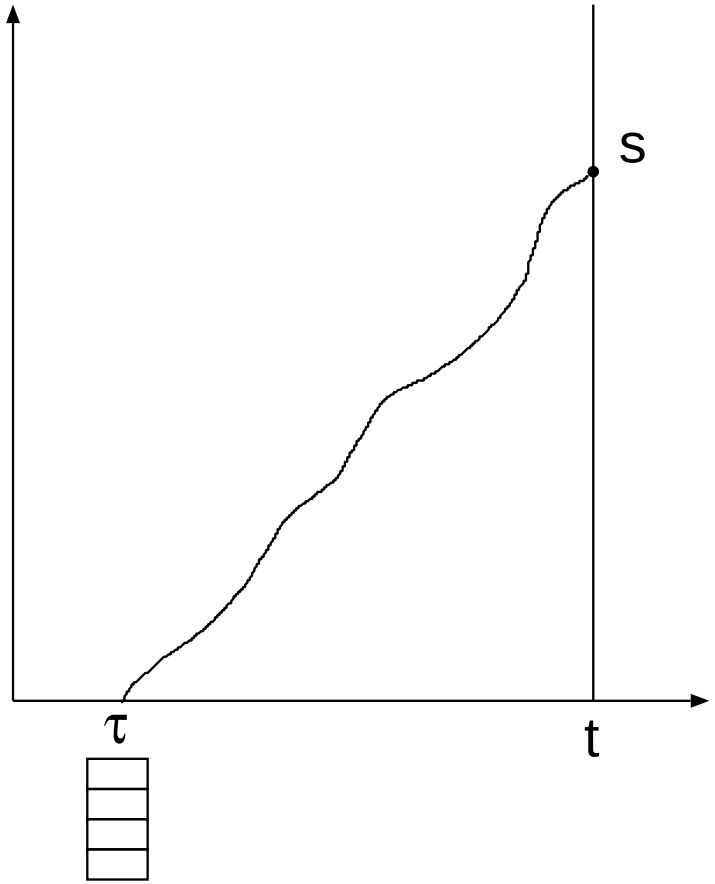
For time conditioned search define:

$$Q_\tau(t,s) \quad := \quad \max_{w_1^{n-1}} \left\{ Pr(w_1^{n-1}) \cdot \max_{s_1^t} \left\{ Pr(x_1^t, s_1^t | w_1^{n-1}, \text{--}) \right\} : s_\tau = S_{w_{n-1}}; s_t = s \right\}$$

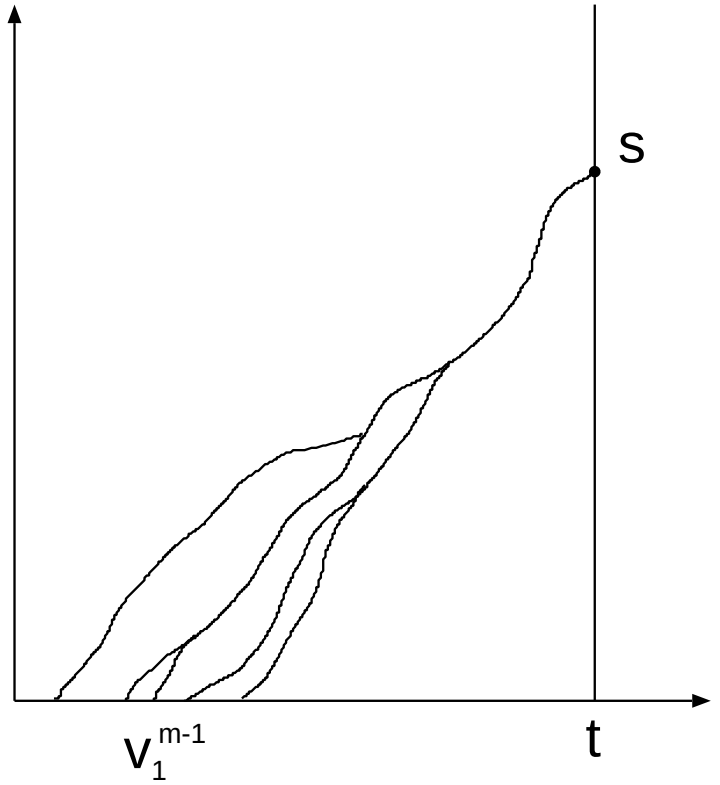
Review: word conditioned tree search for m-gram language models $p(v_m|v_1^{m-1})$:

$$Q_{v_1^{m-1}}(t,s) \quad := \quad \max_{w_1^{n-1}} \left\{ Pr(w_1^{n-1}) \cdot \max_{s_1^t} \left\{ Pr(x_1^t, s_1^t | w_1^{n-1}) \right\} : w_{n-m+1}^{n-1} = v_1^{m-1}; s_t = s \right\}$$

time conditioned search



word conditioned search



The path with the history v_1^{m-1} that ends at time t in state s, must have reached state $S_{v_{m-1}}$ once at time τ: $s_\tau := S_{v_{m-1}}$

This path is split in two partial paths at τ, were τ has to be optimized:

$$
\begin{aligned}
& Q_{v_1^{m-1}}(t,s) = \\
&= \max_{\tau} \left[\max_{w_1^{n-1}} \left\{ Pr(w_1^{n-1}) \cdot \max_{s_1^\tau} \left\{ Pr(x_1^\tau, s_1^\tau | w_1^{n-1}) : w_{n-m+1}^{n-1} = v_1^{m-1}; s_\tau = S_{v_{m-1}} \right\} \right\} \right. \\
&\qquad \left. \cdot \max_{s_{\tau+1}^t} \left\{ Pr(x_{\tau+1}^t, s_{\tau+1}^t | \text{--}) : s_t = s \right\} \right] \\
&= \max_{\tau} \left[H(v_1^{m-1}; \tau) \cdot \max_{s_{\tau+1}^t} \left\{ Pr(x_{\tau+1}^t, s_{\tau+1}^t | \text{--}) : s_t = s \right\} \right] \\
&= \max_{\tau} \left[H(v_1^{m-1}; \tau) \cdot \frac{Q_\tau(t,s)}{\max_{u_1^{m-1}} H(u_1^{m-1}, \tau)} \right]
\end{aligned}
$$

$$Q_{v_1^{m-1}}(t,s) = \max_{\tau}\left[\frac{H(v_1^{m-1};\tau)}{\max_{u_1^{m-1}} H(u_1^{m-1},\tau)} \cdot Q_{\tau}(t,s)\right]$$

$\forall s$ **with** $s \neq 0$**.**

especially for a word boundary $s = S_{v_m}$ **we get:**

$$Q_{v_1^{m-1}}(t,S_{v_m}) \quad = \quad \max_{\tau}\left[H(v_1^{m-1};\tau) \cdot h(v_m;\tau,t)\right]$$

Word conditioned search with recombination for $s = 0$:

$$\begin{aligned} Q_{v_2^m}(t, s=0) &= \max_{v_1} \left\{ p(v_m|v_1^{m-1}) \cdot Q_{v_1^{m-1}}(t, S_{v_m}) \right\} \\ &= \max_{v_1} \left\{ p(v_m|v_1^{m-1}) \cdot \max_{\tau} \left\{ H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t) \right\} \right\} \\ &= H(v_2^m; t) \end{aligned}$$

It is possible to convert scores of the time conditioned search into scores of the word conditioned search.

The opposite conversion is usually impossible because of the optimization over τ which is done implicitly in word conditioned search.

$\rightarrow$ time conditioned search is "less compact".

9 Phonetic Decision Trees

9.1 Motivation 403
9.2 Training Principle 408
9.3 Leaving One Out 419

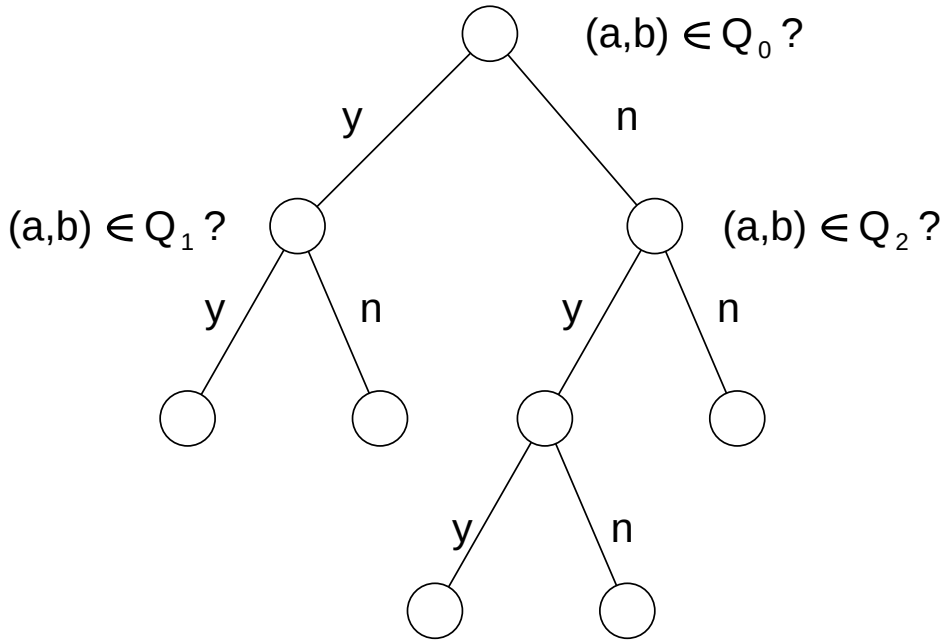
9.1 Motivation

Classification and Regression Trees (CART)

- Used in the acoustic modelling of phonemes
- 50 phonemes $\Rightarrow 50^3 = 125000$ possible phonemes in triphone context ("triphones")
- Problem:
  - too many triphones to be trained reliably
  - many triphones are not seen in training
  - considering accros-word contexts this effect increases
- Solution:
  - tie parameters of similar triphones
  - a decision tree determines similarity

A phoneme X has 2500 possible triphone contexts ${}_aX_b$.

Phonetic decision tree for a phoneme X:

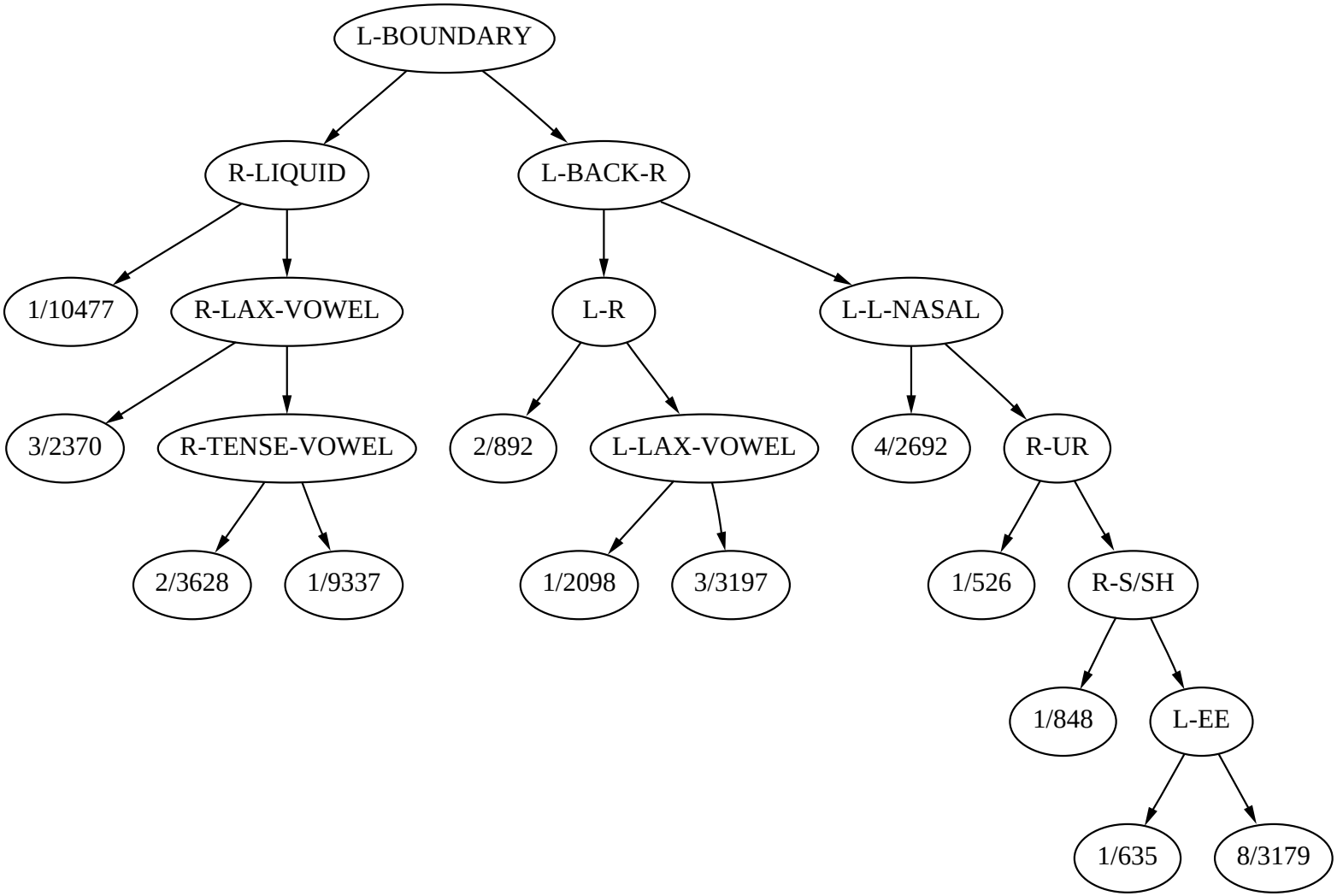


A path trough the decision tree is defined by the answers to phonetic questions $Q_0, Q_1, Q_2, \ldots$

e.g.: - "is the left context a fricative?"

- "is the right context a plosive?"

Example


L-BOUNDARY
R-LIQUID
L-BACK-R
1/10477
R-LAX-VOWEL
L-R
L-L-NASAL
3/2370
R-TENSE-VOWEL
2/892
L-LAX-VOWEL
4/2692
R-UR
2/3628
1/9337
1/2098
3/3197
1/526
R-S/SH
1/848
L-EE
1/635
8/3179

Properties of the tree:

- **Every leaf of the tree stands for a generalized phonetic context and has a corresponding HMM emission probability.**
- **An adequate generalization for triphones not seen in training can be expected.**

General application for CART:

given the two variables

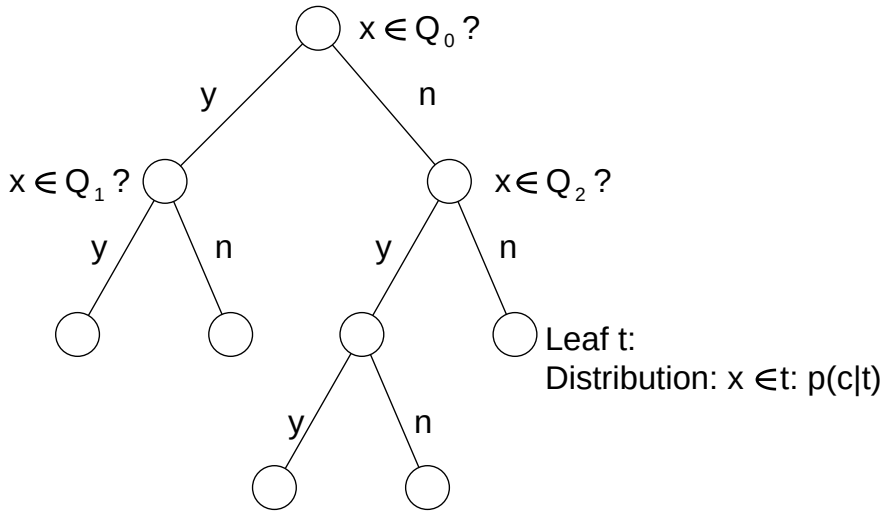
$c =$ **class index**

$x \in \mathbb{R}^D$**, or discrete observation**

model the conditional probability

$p(c|x)$

$\sum_c p(c|x) = 1$



"Classification tree" vs. "estimation tree":
an estimation tree models the conditional probability without classification.

9.2 Training Principle

Given the training data

$$[x_n, y_n], \qquad n = 1, \ldots, N;$$

with $x \rightarrow$ **independent variable**

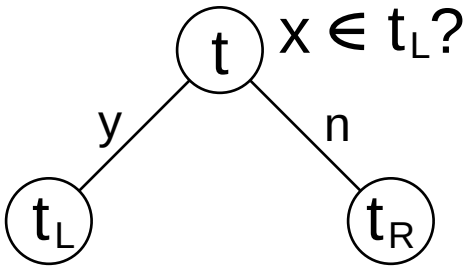
$y \rightarrow$ **dependent variable**

two subsets of $\{x\}$ **are considered**

$$t, t_L, t_R \subset \{x\}$$

Define a tree by binary splitting of a node or subtree:

$$t = t_L \cup t_R, \quad t_L \cap t_R = \emptyset.$$



Define:

- A "score" $g(y_n|t)$ for every observation (x_n, y_n) with $x_n \in t$;
- A score for the node t:

$$G(t) := \sum_{n:x_n \in t} g(y_n|t).$$

The score function $g(y_n|t)$ shall be additive.

Note the change in the score when splitting t in the subsets t_L and t_R:

$$\Delta G(t_L|t) \quad = \quad G(t) - G(t_L) - G(t_R)$$

best split t_L for given t:

$$\max_{t_L} \Delta G(t_L|t)$$

Use the log-likelihood (log-probability) criterion for $G(t)$
(θ represents the parameter of the distribution):

$$\begin{aligned} g(y_n|t) &:= \log p_\theta(y_n|t) \\ G(t) &:= \max_\theta \sum_{n:x_n \in t} \log p_\theta(y_n|t) \end{aligned}$$

$G(t_L), G(t_R)$ correspondingly.

There are two different ways of optimizing:

- learn the best parameters $\hat{\theta}$ for a hypothetic split t_L at a node t.
- choose optimal split

Thus:

$$\begin{aligned} \hat{\theta} &= \hat{\theta}\{(x_n, y_n) : x_n \in t_L; n = 1, \ldots, N\} \\ G(t_L) &= \sum_{n:x_n \in t_L} \log p_{\hat{\theta}}(y_n|t_L) \end{aligned}$$

y with DISCRETE values:
the parameters θ are the distribution $p(y|t)$ itself (non parametric model)

Then:

$$\sum_{n:x_n \in t} \log p(y_n|t) = \sum_y N(t,y) \cdot \log p(y|t) - \lambda \left[\sum_y p(y|t) - 1 \right]$$

$$\begin{aligned} \frac{\partial}{\partial p(y|t)} &: \frac{N(t,y)}{p(y|t)} - \lambda = 0 \\ \frac{\partial}{\partial \lambda} &: \sum_y p(y|t) - 1 = 0 \end{aligned}$$

$$\Rightarrow \hat{\theta} \equiv \hat{p}(y|t) = \frac{N(t,y)}{N(t)}$$

with the counts: $N(t,y), N(t)$

For the optimum:

$$
\begin{aligned}
G(t) &= \sum_{n:x_n \in t} \log p_{\hat{\theta}}(y_n|t) \\
&= \sum_{n:x_n \in t} \log \frac{N(t_L, y)}{N(t)} \\
&= \sum_{y} N(t, y) \cdot \log \frac{N(t_L, y)}{N(t)} \\
&= N(t) \sum_{y} \hat{p}(y|t) \cdot \log \hat{p}(y|t)
\end{aligned}
$$

entropy

y with CONTINUOUS values, especially Gaussian distribution:

$$p_\theta(y|t) \quad = \quad \mathcal{N}(y|\mu_t, \Sigma_t)$$

$$\mathcal{N}(y|\mu_t, \Sigma_t) = \frac{1}{\sqrt{\det(2\pi\Sigma_t)}} \cdot \exp\left[-\frac{1}{2}(y-\mu_t)^T \Sigma_t^{-1}(y-\mu_t)\right]$$

$$\begin{aligned} G(t) \quad &= \quad \sum_{n:x_n \in t} \log \mathcal{N}(y_n|\hat{\mu}_t, \hat{\Sigma}_t) \\ &= \quad -\frac{N(t)}{2}\log(2\pi)^D - \frac{N(t)}{2}\log\left[\det\hat{\Sigma}_t\right] - \frac{1}{2}\sum_{n:x_n \in t}(y_n-\hat{\mu}_t)^T\hat{\Sigma}_t^{-1}(y_n-\hat{\mu}_t) \end{aligned}$$

with

$$N(t) \quad := \quad \sum_{n:x_n \in t} 1$$

Maximum-likelihood estimation for $\hat{\mu}_t$ and $\hat{\Sigma}_t$:

$$\hat{\mu}_t = \frac{1}{N(t)} \sum_{n:x_n \in t} y_n$$

$$\hat{\Sigma}_t = \frac{1}{N(t)} \sum_{n:x_n \in t} (y_n - \hat{\mu}_t)(y_n - \hat{\mu}_t)^T$$

Using a diagonal covariance matrix:

$$\hat{\Sigma}_t = \begin{bmatrix} \hat{\sigma}_{t1}^2 & & & 0 \\ & \hat{\sigma}_{t2}^2 & & \\ & & \ddots & \\ 0 & & & \hat{\sigma}_{tD}^2 \end{bmatrix}$$

$$\hat{\sigma}_{td}^2 = \frac{1}{N(t)} \sum_{n:x_n \in t} (y_{nd} - \hat{\mu}_{td})^2$$

$$\begin{aligned} \sum_{n:x_n \in t} (y_n - \hat{\mu}_t)^T \hat{\Sigma}_t^{-1} (y_n - \hat{\mu}_t) &= \sum_{n:x_n \in t} \sum_d \left(\frac{y_{nd} - \hat{\mu}_{td}}{\hat{\sigma}_{td}} \right)^2 \\ &= \sum_d \frac{1}{\hat{\sigma}_{td}^2} \cdot \sum_{n:x_n \in t} (y_{nd} - \hat{\mu}_{td})^2 \\ &= N(t) \cdot D \end{aligned}$$

General case, full covariance matrix (the index t of $\hat{\mu}_t$ and $\hat{\Sigma}_t$ is dropped here for simplification)

$$
\begin{aligned}
z_n &:= y_n - \hat{\mu} \\
\hat{\Sigma} &:= \frac{1}{N(t)} \sum_{n:x_n \in t}^{N} z_n \cdot z_n^T
\end{aligned}
$$

$$
\begin{aligned}
\sum_{n:x_n \in t} z_n^T \hat{\Sigma}^{-1} z_n &= \sum_{n} \sum_{i=1;j=1}^{D} z_{ni} \left(\hat{\Sigma}^{-1}\right)_{ij} z_{nj} \\
&= \sum_{ij} \left[\sum_{n:x_n \in t} z_{ni} z_{nj} \right] \left(\hat{\Sigma}^{-1}\right)_{ij} \\
&= N(t) \cdot \sum_{ij} \hat{\Sigma}_{ij} \left(\hat{\Sigma}^{-1}\right)_{ij} \\
&= N(t) \cdot \sum_{j} \sum_{i} \hat{\Sigma}_{ji} \left(\hat{\Sigma}^{-1}\right)_{ij} \\
&= N(t) \cdot \sum_{j=1}^{D} \delta_{jj} \quad = \quad N(t) \cdot D
\end{aligned}
$$

Thus:

$$\sum_{n:x_n \in t} \log \mathcal{N}(y_n|\hat{\mu}_t, \hat{\Sigma}_t) = -\frac{N(t)}{2} \log \det \left(2\pi e \hat{\Sigma}_t\right)$$

$$= -\frac{N(t)}{2} \left[D \log(2\pi) + D + \log(\det \hat{\Sigma}_t)\right]$$

The improvement of the log-likelihood score by splitting $t = t_L \cup t_R; t_L \cap t_R = \emptyset$ is:

$$\begin{aligned}
\Delta G(t) &= G(t) - G(t_L) - G(t_R) \\
&= \ldots \\
&= -\frac{N(t_L)}{2} \log \left[\frac{\det \hat{\Sigma}_{t_L}}{\det \hat{\Sigma}_t} \right] - \frac{N(t_R)}{2} \log \left[\frac{\det \hat{\Sigma}_{t_R}}{\det \hat{\Sigma}_t} \right]
\end{aligned}$$

For a diagonal covariance matrix:

$$\begin{aligned}
\log \det \hat{\Sigma}_t &= \log \prod_d \hat{\sigma}_{td}^2 \\
&= \sum_{d=1}^{D} \log \hat{\sigma}_{td}^2
\end{aligned}$$

9.3 Leaving One Out

So far without leaving one out:

$$\max_{\theta} \sum_{n} \log p_{\theta}(y_n) \Rightarrow \hat{\theta} = \hat{\theta}(y_1^N)$$

The optimal value is put in $\sum_n \log p_{\hat{\theta}(y_1^N)}(y_n)$ so every observation is considered twice:

1. **to determine $\hat{\theta}$**
2. **to determine how good the model $p_{\hat{\theta}}(y)$ explains the observations $y_1, \ldots, y_N$.**

With leaving one out:

Idea: take y_n out of the training set when evaluating $p_{\hat{\theta}}(y_n)$

Use $\hat{\theta}(y_1^N \backslash y_n)$ instead of $\hat{\theta}(y_1^N)$ for the leaving one out score evaluation:

$$\sum_n \log p_{\hat{\theta}(y_1^N \backslash y_n)}(y_n)$$

For Gaussian distributions the calculation of $\hat{\theta}(y_1^N \backslash y_n)$ is easy:

$$\begin{aligned}
\hat{\mu}_n &:= \frac{1}{N-1} \sum_{m \neq n} y_m \\
\hat{\Sigma}_n &:= \frac{1}{N-1} \sum_{m \neq n} (y_m - \hat{\mu}_n)(y_m - \hat{\mu}_n)^t
\end{aligned}$$