Summer University

Speech Recognition

01 July – 30 August 2002

Lecture & Computer Experiments

Large Vocabulary Speech Recognition

Andreas Wendemuth

Schedule:
Large Vocabulary Speech Recognition

- Refresh
- Gaussian densities
- What about LVSR?
- Integrating the Language Model
- Search techniques
- Lattice techniques

Modelle mit beschränktem Kontext

$$P(w_t \mid w_1 \ldots w_{t-1}) \approx P(w_t \mid \underbrace{w_{t-n+1} \ldots w_{t-1}}_{(n-1)\text{-Gramm-Kontext}})$$

Unigramm-, Bigramm- und Trigramm-Modelle

$$P_{1G}(\boldsymbol{w}) = \prod_{t=1}^{T} P(w_t)$$

$$P_{2G}(\boldsymbol{w}) = P(w_1) \cdot \prod_{t=2}^{T} P(w_t \mid w_{t-1})$$

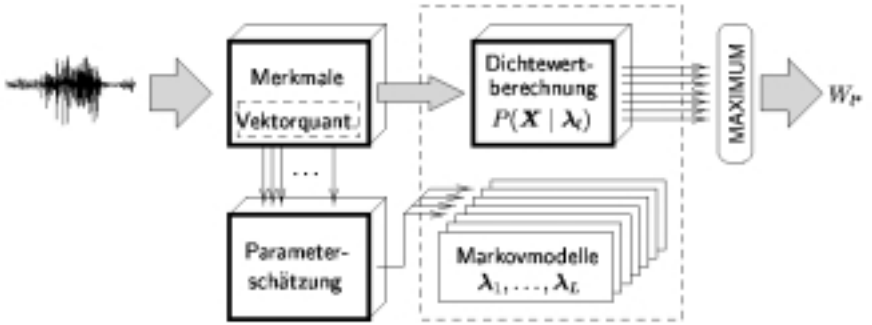
$$P_{3G}(\boldsymbol{w}) = P(w_1) \cdot P(w_2 \mid w_1) \cdot \prod_{t=3}^{T} P(w_t \mid w_{t-2} w_{t-1})$$

n-Gramm-Sprachmodelle

$$P_{nG}(\boldsymbol{w}) = \prod_{t=1}^{T} P(w_t \mid w_{t-n+1} \ldots w_{t-1})$$

mit $w_0, w_{-1}, w_{-2}, \ldots :=$ \$ (Satzanfangsmarkierungen)

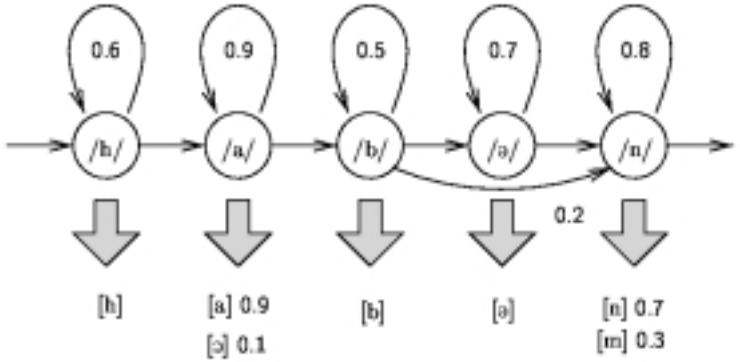
Statistische Wortdekodierung ...



$$l^* = \underset{l}{\operatorname{argmax}}\, P(W_l \mid \boldsymbol{X}) = \underset{l}{\operatorname{argmax}}\, \frac{P(\boldsymbol{X} \mid W_l) \cdot P(W_l)}{P(\boldsymbol{X})}$$

... mit wortbezog. „Markovmodellen“

Hidden Markov Model (HMM)



HIDDEN MARKOVMODELL mit DISKRETER Ausgabeverteilung

$$\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$$

Normierungsbedingungen $(\pi_i \geq 0,\ a_{ij} \geq 0,\ b_{ik} \geq 0)$

$$\sum_{i=1}^{N} \pi_i = 1$$

$$\sum_{j=1}^{N} a_{ij} = 1 \quad (i = 1, \ldots, N)$$

$$\sum_{k=1}^{K} b_{ik} = 1 \quad (i = 1, \ldots, N)$$

3 offene Fragen zum Thema HMM:

- Berechnung der **Produktionswahrscheinlichkeit**

$$P(\boldsymbol{X}|\boldsymbol{\lambda}) = ?$$

- Bestimmung der **wahrscheinlichsten Zustandsfolge**

$$P(\boldsymbol{q}, \boldsymbol{X} \mid \boldsymbol{\lambda}) \overset{q}{\Longrightarrow} \text{MAX}$$

- Schätzung der bestpassenden **Modellparameter**

$$\hat{\boldsymbol{\lambda}} = (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}) \quad \text{mit} \quad P(\boldsymbol{X}|\hat{\boldsymbol{\lambda}}) = \max_{\boldsymbol{\lambda}} P(\boldsymbol{X}|\boldsymbol{\lambda})$$

... und die gesuchte
PRODUKTIONSWAHRSCHEINLICHKEIT

$$P(\boldsymbol{O} \mid \boldsymbol{\lambda}) = \sum_{\boldsymbol{q} \in \mathcal{Q}^T} P(\boldsymbol{O}, \boldsymbol{q} \mid \boldsymbol{\lambda}) = \sum_{\boldsymbol{q} \in \mathcal{Q}^T} \pi_{q_1} b_{q_1}(O_1) \cdot \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(O_t)$$

⇨ ca. $2T \cdot N^T$ Multiplikationen (exponentielle Komplexität!)

•From HMMs, we know that transitions and production probabilities give the total production prob. lined out above.

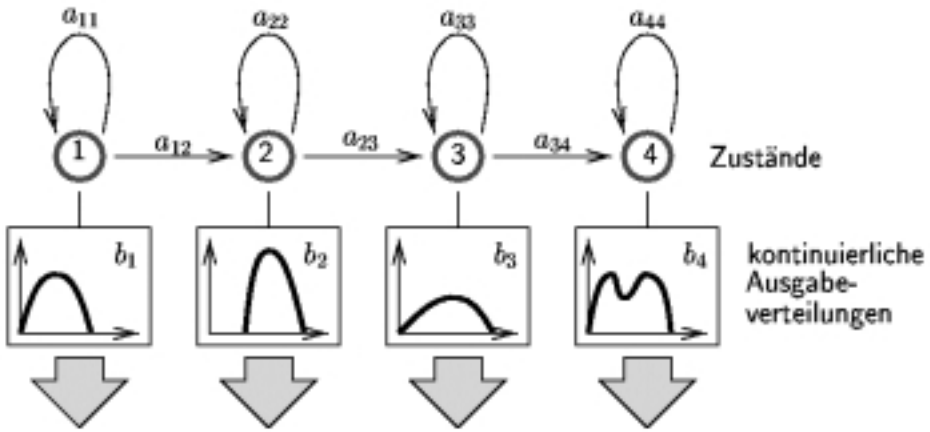
•These can be trained with Baum-Welch algorithm, using forward- and backward probabilities. This is a variant of Estimation-Maximization (EM) Techniques.

•Recognition (decoding) can be done with Viterbi algorithm.

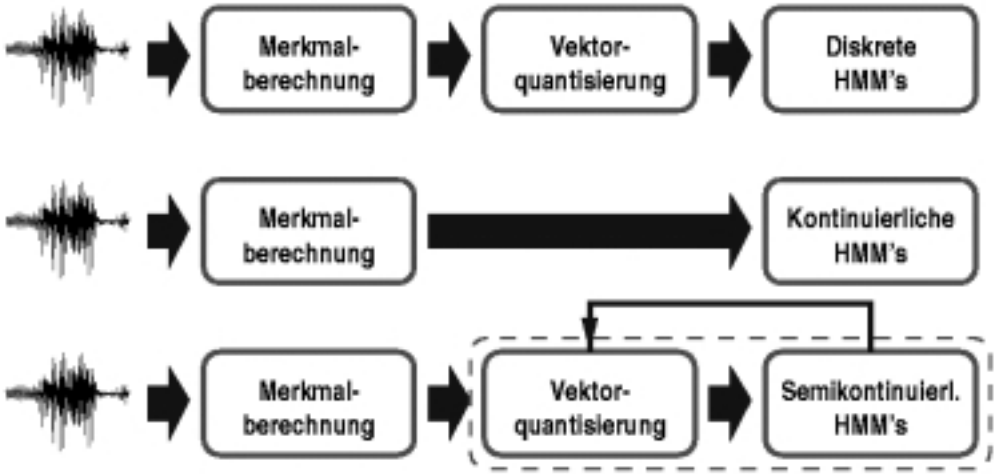
•Now we are going to apply GAUSSIAN probability density functions:

Gaussian densities

HMM's mit kontinuierlichen Ausgabeverteilungen



Architektur von HMM-Spracherkennern:



Baum-Welch-Algorithmus

Berechnung „neuer“ Parameter $\hat{\boldsymbol{\lambda}}$ aus den „alten“ $\boldsymbol{\lambda}$ mit

$$\mathcal{L}_{\text{HMM}}(\hat{\boldsymbol{\lambda}}) \quad \geq \quad \mathcal{L}_{\text{HMM}}(\boldsymbol{\lambda})$$

a posteriori Übergangswahrscheinlichkeiten

für $s_i \to s_j$ zum Zeitpunkt t

$$\xi_t(i,j) := P(q_t = i, q_{t+1} = j \mid \boldsymbol{O}, \boldsymbol{\lambda}) \;=\; \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)}$$

a posteriori Zustandswahrscheinlichkeiten

für s_i zum Zeitpunkt t

$$\gamma_t(i) := P(q_t = i \mid \boldsymbol{O}, \boldsymbol{\lambda}) \;=\; \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)} \;=\; \sum_{j=1}^{N} \xi_t(i,j)$$

Baum-Welch-Formeln für diskretwertige HMMs

$$\hat{\pi}_i \;=\; \gamma_1(i) \;=\; \frac{\alpha_1(i) \beta_1(i)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)}$$

$$\hat{a}_{ij} \;=\; \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \;=\; \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)}$$

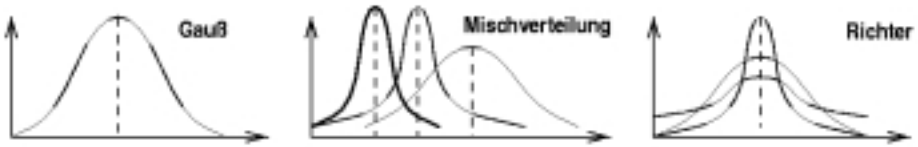
$$\hat{b}_{jk} \;=\; \frac{\sum_{t=1}^{T} \gamma_t(j) \boldsymbol{\chi}_{[O_t = v_k]}}{\sum_{t=1}^{T} \gamma_t(j)} \;=\; \frac{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j) \boldsymbol{\chi}_{[O_t = v_k]}}{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j)}$$

Es sei $\boldsymbol{\chi}_{[\cdot]} = 1$ für zutreffende Aussagen und 0 sonst.

HMM's mit normalverteilter Ausgabe

Ausgabeverhalten des Zustandes s_j

$$b_j(\boldsymbol{x}) \;=\; \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$



Die Dekomposition der **Kullback-Leibler-Statistik** $\mathrm{Q}(\cdot,\cdot)$ ergibt

$$\mathrm{Q}_{\hat{\boldsymbol{b}}_j}(\boldsymbol{\lambda}, \hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j) \;=\; \sum_{t=1}^{T} \gamma_t(j) \log \hat{b}_j(\boldsymbol{x}_t) \;=\; \sum_{t=1}^{T} \gamma_t(j) \log \mathcal{N}(\boldsymbol{x} \mid \hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j)$$

Schätzformeln für die NV-Dichteparameter:

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_j &= \frac{1}{\Sigma_t \gamma_t(j)} \sum_{t=1}^{T} \gamma_t(j) \boldsymbol{x}_t \\
\hat{\boldsymbol{\Sigma}}_j &= \frac{1}{\Sigma_t \gamma_t(j)} \sum_{t=1}^{T} \gamma_t(j) (\boldsymbol{x}_t - \hat{\boldsymbol{\mu}}_j)(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}}_j)^\top \\
&= \frac{1}{\Sigma_t \gamma_t(j)} \sum_{t=1}^{T} \gamma_t(j) \boldsymbol{x}_t \boldsymbol{x}_t^\top \;-\; \hat{\boldsymbol{\mu}}_j \hat{\boldsymbol{\mu}}_j^\top
\end{aligned}$$

Why not word models?

•Statistische Unzuverlässigkeit: Für die Schätzung von Wortmodellen wäre eine Unmenge an Trainingsdaten erforderlich um Ganzwortmodelle zuverlässig zu schätzen. Bei wachsendem Erkennungsvokabular ist beispielsweise für ein Spracherkennungsprogramm bezüglich des Trainingsaufwandes die Grenze des Zumutbaren schon bei kleinerem Vokabular erreicht.

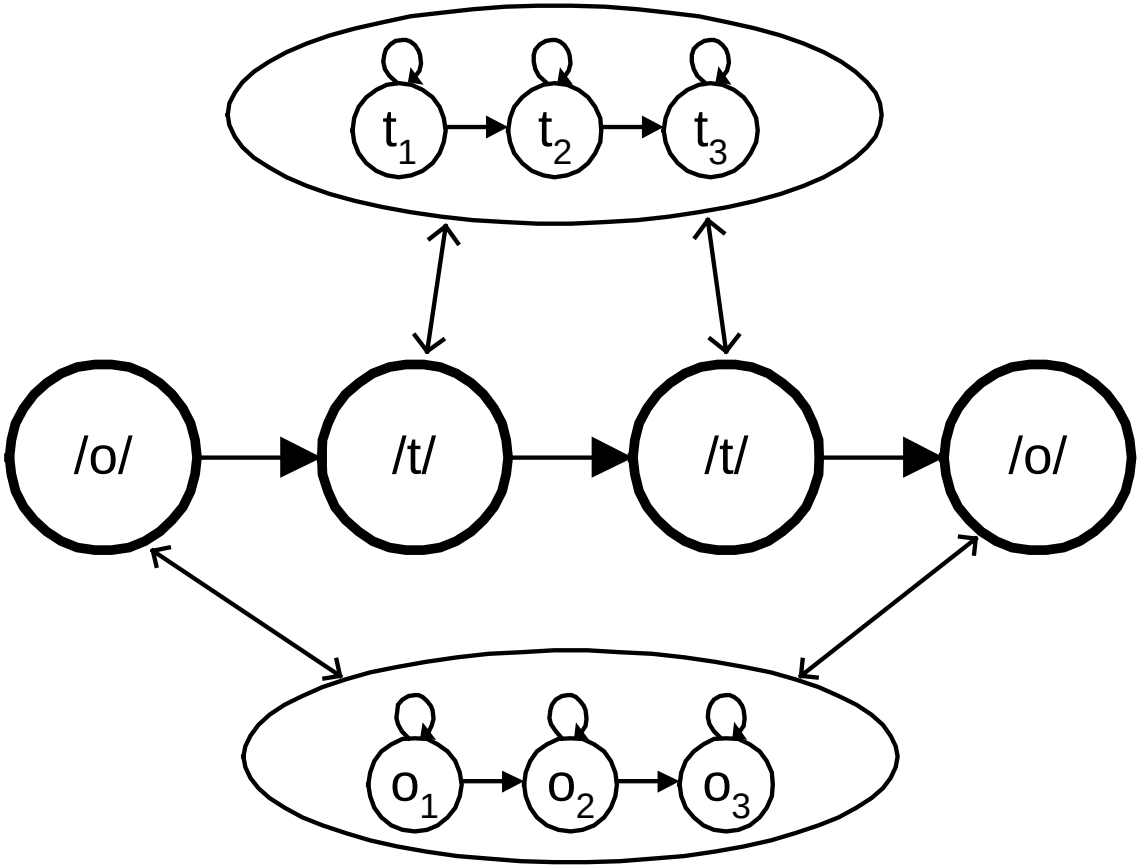
•Rechen- und Speicherbedarf: Für großes Vokabular steigt die Speicher und Rechenaufwand enorm an.

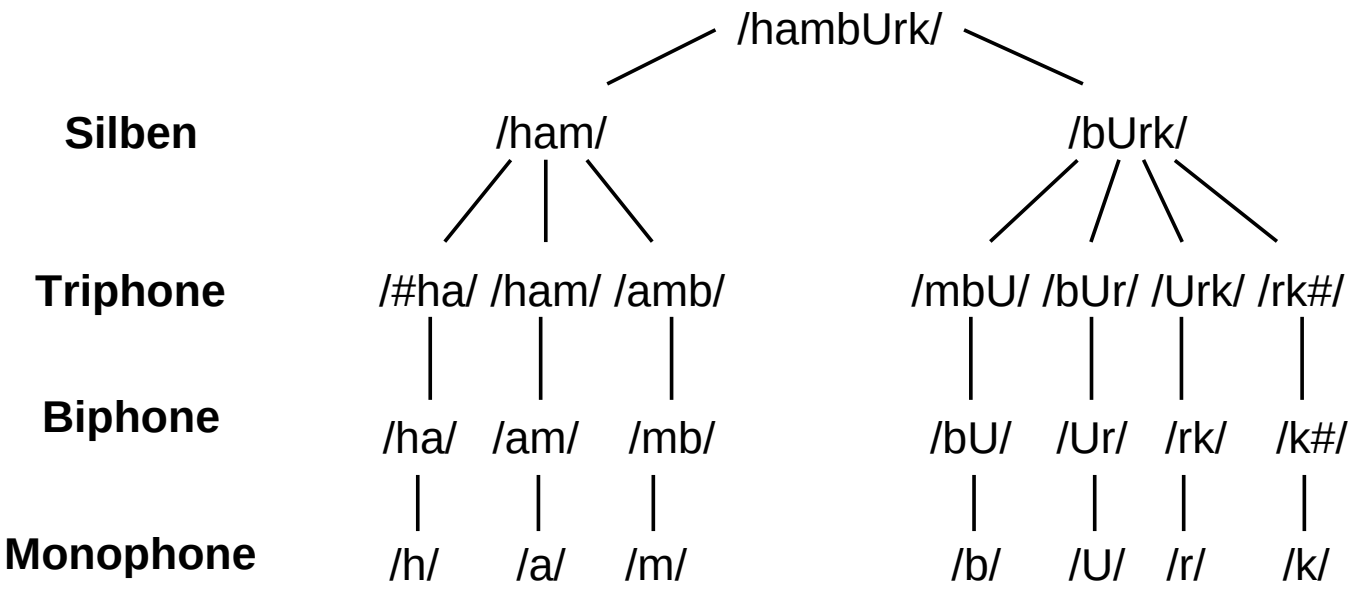
•Geringe Flexibilität: Bei der Verwendung von Wortmodellen ist eine Erkennung von Wörtern außerhalb des Erkennungsvokabulars nicht möglich.

•Geringe Modularität: Ganzwortmodelle berücksichtigen nicht die von der Satzkonstruktion abhängigen Verschleifungen am Wortrand.

Phoneme models!

- **Phonemes are the smallest entities of language which distinguish meaning**
- **Example: [f]ather, [r]ather**
- **There are 50 phonemes /k/,**
  4000 triphones [/a/ /th/ /r/]
  20,000 syllables „rath“,
  200,000 wordforms „fathers“
- **Therefore phonems are for acoustic speech what letters are for written speech.**

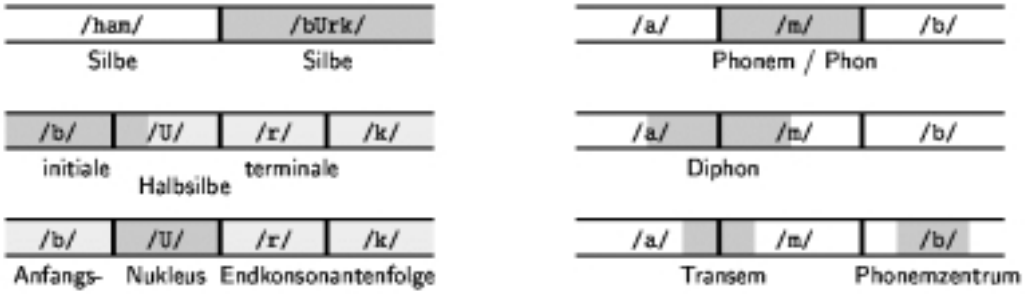

t1
t2
t3
/o/
/t/
/t/
/o/
o1
o2
o3


/hambUrk/
Silben
/ham/
/bUrk/
Triphone
/#ha/ /ham/ /amb/
/mbU/ /bUr/ /Urk/ /rk#/
Biphone
/ha/ /am/ /mb/
/bU/ /Ur/ /rk/ /k#/
Monophone
/h/ /a/ /m/
/b/ /U/ /r/ /k/

Word / Pronunciation Lexicon

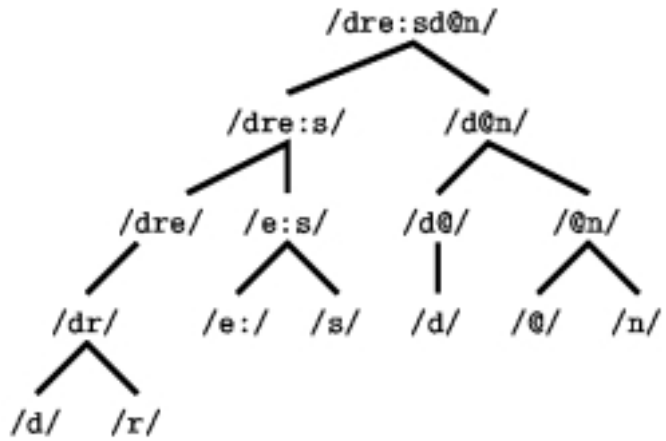
- **Zero Z IH R OW**
- **One W AH N**
- **Two T UW**
- **Three TH R IY**
- **Four F OW R**
- **Five F AY V**
- **Six S IH K S**
- **Seven S EH V AX N**
- **Eight EY T**
- **Nine N AY N**
- **Oh OW**

Phonologisch orientierte Wortuntereinheiten

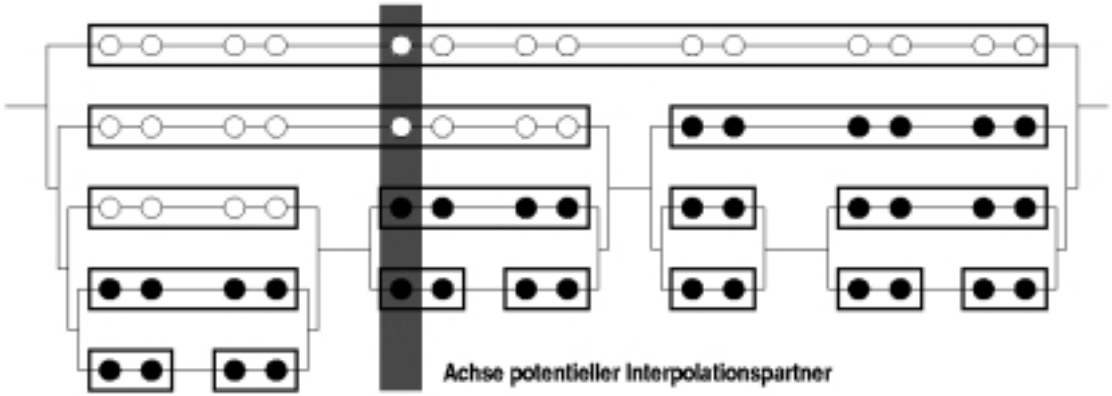


Phone	je nach Differenzierungsgrad 40–200 universelle Einheiten *(+modular, −präzis)*
Phoneme	je nach Sprache 20–60 *(+modular, −präzis)*
Silben	20 000 im Englischen, 100 im Japanischen; Koartikulation primär innerhalb *(+präzis, −modular)*
Halbsilben	800/2560 initiale/terminale im Deutschen *(+trennscharf, ±modular)*
Sylparts	47 AKF, 20 Nuklei, 159 EKF im Deutschen *(guter Kompromiß)*
Diphone	1000–1500 Einheiten (engl./ital.), ungünstige Nahtstellen *(besser: Transeme)*
Doppelhalbsilbe	Silbenkern–Silbenkern, 2 Mill. im Deutschen *(++trennscharf, −−modular, −−robust)*

Hierarchische Wortrepräsentation



Hierarchische Modellstruktur

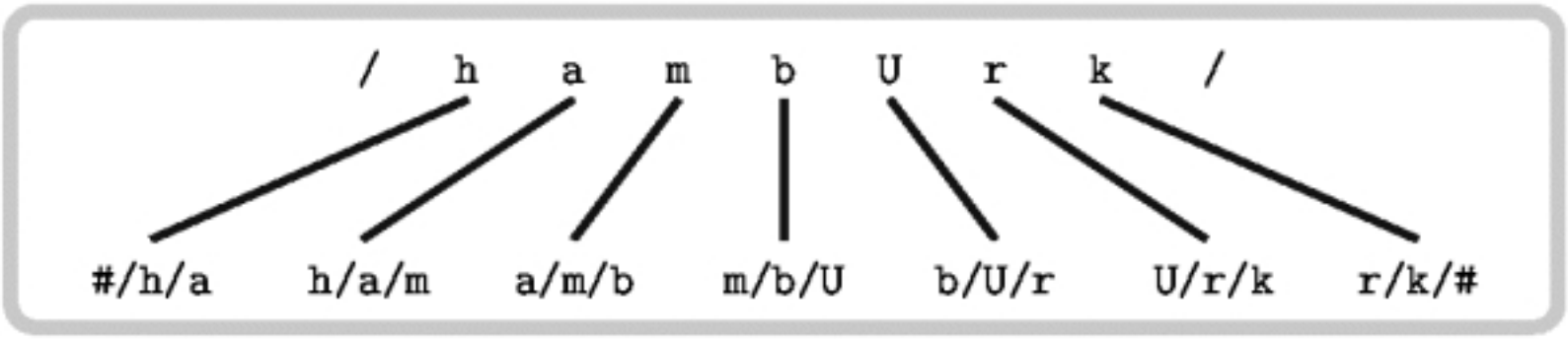


Schreibweise für **Triphone**

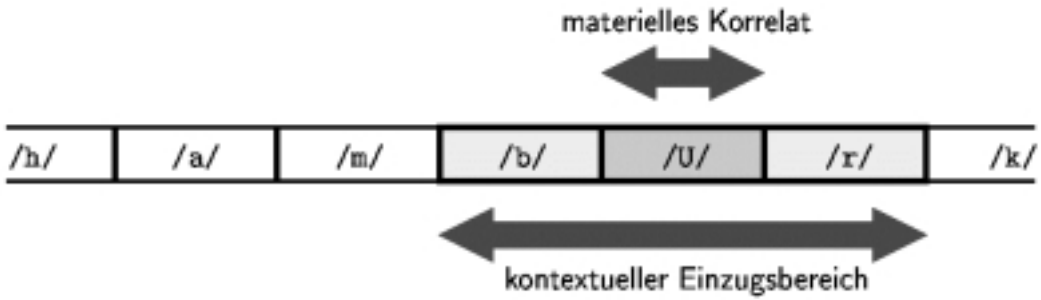
r in hambUrk ⇨ U/r/k

und für **rechte / linke Biphone** und **Monophone**

r ⟶ /r/k r ⟶ U/r/ r ⟶ /r/



Triphonmodelle

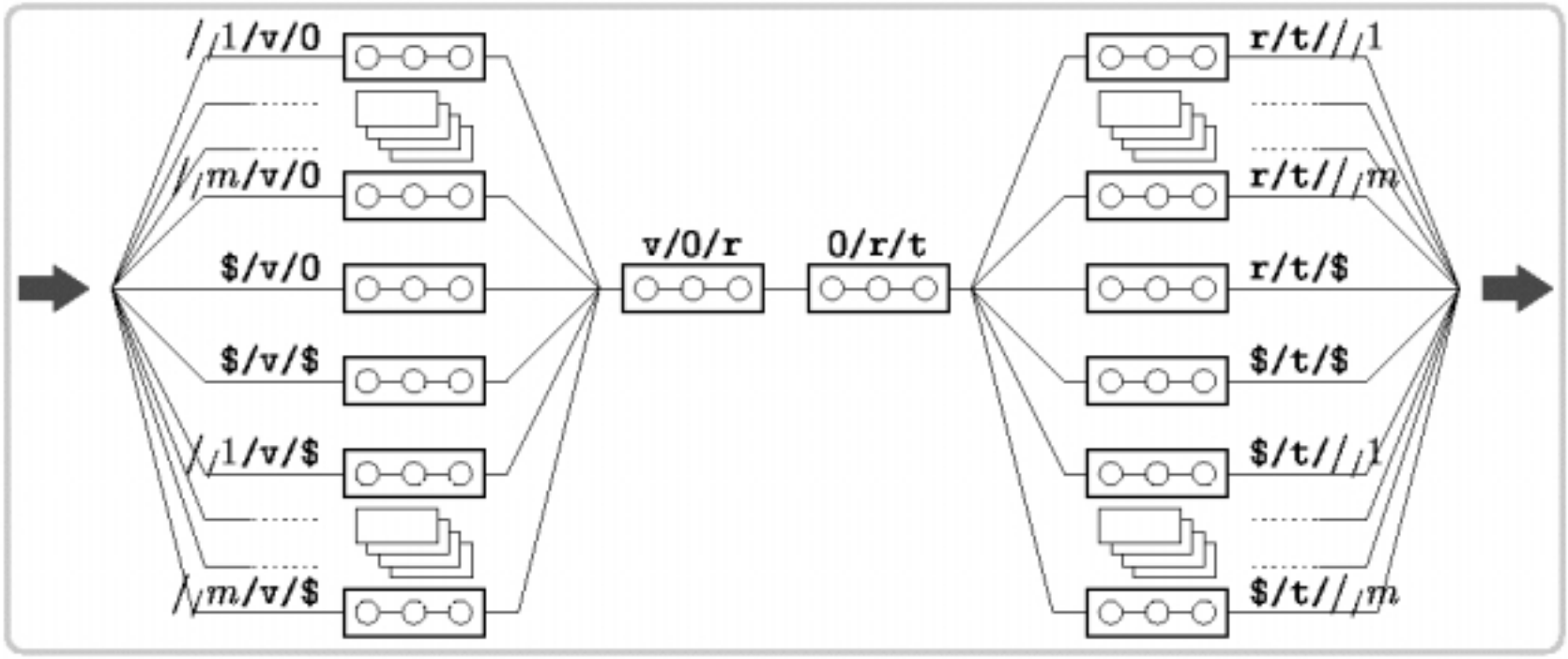


Training von Triphon-HMM's

1. Initialisierung / Training gewöhnlicher **Monophonmodelle**
2. Training der **Biphonmodelle** (⇦ Monophonparameter)
3. Training der **Triphonmodelle** (⇦ Biphonparameter)
4. Konstruktion der Wortmodelle des Erkennungsvokabulars:
   - ***Rückgriffstrategie***:
     $\boxed{\lambda(\mathtt{b/U/r})} \rightsquigarrow \boxed{\lambda(\mathtt{/U/r})} \rightsquigarrow \boxed{\lambda(\mathtt{b/U/})} \rightsquigarrow \boxed{\lambda(\mathtt{/U/})}$
   - ***Interpolation***: Gewichte *uniform*, *heuristisch*, nach *Kopfzahl* oder EM-optimiert

⇨ **wortabhängige** Phonmodelle für häufige *Funktionswörter*
(stellen 3% des Vokabulars, 30% der Vorkommen und 60–75% der Fehler)

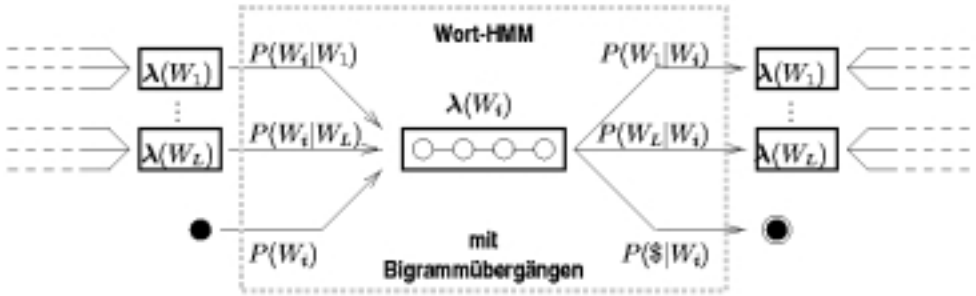
⇨ **prosodische Faktoren**
Akzentmarken; Silben-, Wort- und Morphemgrenzen



Kontextbedingt verästeltes initiales/terminales Triphonmodell des Wortes `vOrt` („*Wort*“)

Verbundworterkennung

mit wortbezogener Bigrammgrammatik



- L Wortmodelle
  (wie ein Einzel- oder grammatikfreier Verbundworterkenner)
- L^2 Wort-HMM-Übergangskanten mit Bigrammwahrscheinlichkeiten

Wann „realisiert“ ein HMM-Netz eine Grammatik?

$$P(\boldsymbol{X}, \boldsymbol{w} \mid \boldsymbol{\lambda}) \;=\; P(\boldsymbol{X} \mid \boldsymbol{\lambda}(\boldsymbol{w})) \cdot P(\boldsymbol{w})$$

mit

$$P(\boldsymbol{X}, \boldsymbol{w} \mid \boldsymbol{\lambda}) := \sum_{\boldsymbol{q} \in \mathcal{Q}^T} P(\boldsymbol{X}, \boldsymbol{q} \mid \boldsymbol{\lambda})$$

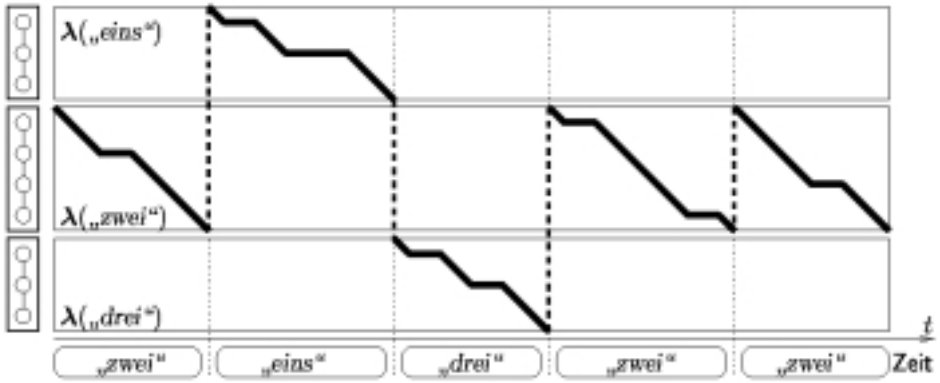
(Summation über alle $\boldsymbol{q}$ mit $w(\boldsymbol{q}) = \boldsymbol{w}$)

Viterbi-Dekodierung

- Viterbi-Algorithmus auf dem HMM-Netzwerk
- Wahrscheinlichste Zustandsfolge $\boldsymbol{q}^*$
- Resultat ist $\boldsymbol{w}^* = w(\boldsymbol{q}^*)$

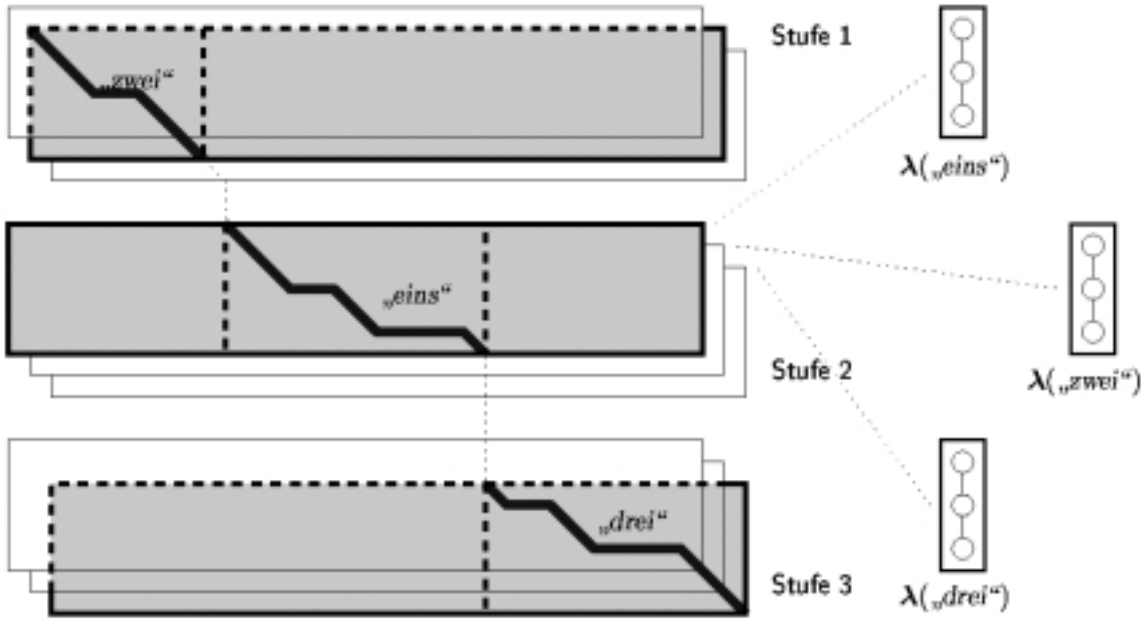
⇨ $\boldsymbol{w}^*$ ist optimale Segmentierung bzgl. $P^*(\boldsymbol{X}|\boldsymbol{\lambda}(\boldsymbol{w}))$

Einstufige Verbundwortdekodierung



- *Viterbi-Algorithmus auf dem Rückkopplungsmodell*
- „kurze“ Sprünge innerhalb eines Wortmodells
- „weitere“ Sprünge zwischen zwei Wortmodellen
- Rechenaufwand $T \cdot L \cdot \Sigma_l N_l$

Mehrstufige Verbundwortdekodierung



- *Viterbi-Algorithmus auf dem m-Wörter-Modell*
  - Keine echten Zyklen im HMM-Netz
  - ⇨ *synchrone* wie auch *asynchrone* $\vartheta_l(i)$-Berechnung möglich
  - Rechenaufwand $m \cdot T \cdot L \cdot \Sigma_l N_l$

Vorwärtsdekodierung

Näherungsweise Wortsegmentierung bzgl. $P(\boldsymbol{X}|\boldsymbol{\lambda}(\boldsymbol{w}))$

- α-Berechnung (Summe) *innerhalb* der Wortmodelle
- ϑ-Berechnung (Max) *zwischen* den Wortmodellen

- **Initialisierung**: Setze für alle $j = 1, \ldots, N$

$$\vartheta_1(j) = \pi_j b_j(\boldsymbol{x}_1) \quad \text{und} \quad \psi_1(j) = 0$$

- **Rekursion**: Für alle $j = 1, \ldots, N$ setze

$$\psi_t(j) \;=\; \operatorname*{argmax}_i \vartheta_{t-1}(i) a_{ij}$$

sowie

$$\vartheta_t(j) \;=\; \begin{cases} \max\limits_i \; (\vartheta_{t-1}(i) a_{ij}) \cdot b_j(\boldsymbol{x}_t) & \text{falls } s_j \text{ Wortanfangszustand ist} \\ \sum\limits_i \; (\vartheta_{t-1}(i) a_{ij}) \cdot b_j(\boldsymbol{x}_t) & \text{für alle sonstigen } s_j \end{cases}$$

- **Terminierung**: Setze

$$P^*(\boldsymbol{X} \mid \boldsymbol{\lambda}) = \vartheta_T(N) \quad \text{und} \quad q_T^* = \vartheta_T(N)$$

- **Rückverfolgung**: Für $t = t - 1, \ldots, 1$ setze

$$q_t^* \;=\; \psi_{t+1}(q_{t+1}^*)$$

- **Lösungswortkette**: Setze $\boldsymbol{w}^* \;=\; w(\boldsymbol{q}^*)$.

Language Model Recombination:

- Scores: Log P(A,W) = log P(A|W) + log P(W)
- But the word chain probability P(W) is only known after closure of the history
- Therefore, keep copies of (acoustic) history edges until new word transition
- Then, compute partial best score and close all sub-optimal histories which are the most in the past
- Example: current word: „is“, LM: trigram, 4 edges so far:

  The tree is, A tree is, The three is, <start> Three is
- Best score: The tree is -> close „The“
- Next word investigated: „green“. New hypotheses are:

  Tree is green, three is green

Strahlsuche *(„beam search“)*

- Verfolgt kleine Schar *wahrscheinlichster* aktueller Zustände
- Anzahl der Kandidaten (Strahlbreite) ist *adaptiv*
- Reformulierung des Viterbi-Algorithmus notwendig

Operationen des VA beim Fortschreiten $t \rightarrow t+1$:

- ▷ für alle j: $\vartheta_{t+1}(j) \leftarrow 0$
- ▷ für alle i, j: $\vartheta_{t+1}(j) \leftarrow \max \left\{ \begin{array}{c} \vartheta_{t+1}(j) \\ \vartheta_t(i) \cdot a_{ij} \cdot b_j(\boldsymbol{x}_{t+1}) \end{array} \right\}$

Jede der N^2 Maximumop's erübrigt sich, wenn einer der 3 Faktoren $= 0$ wird!

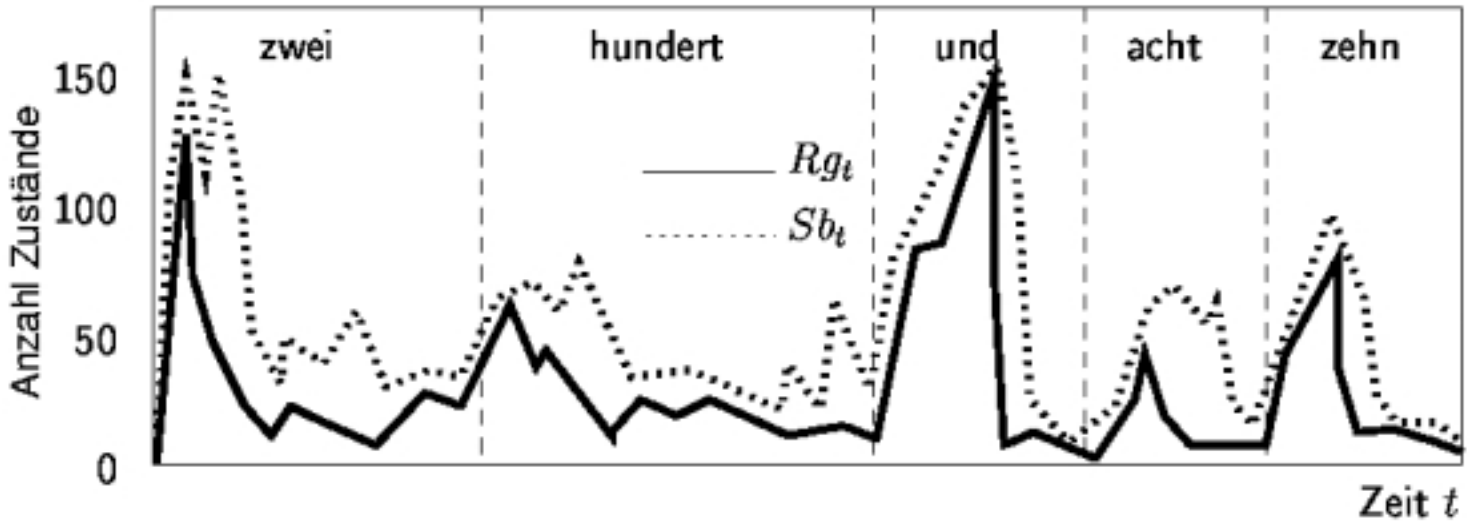
Menge der aktiven Zustände

$$\mathcal{O}_t = \{i \mid \vartheta_t(i) \neq 0\}$$

Beschneidungsstrategie ($B_0 > 0$ sehr klein):

$$\mathcal{O}_t = \{i \mid \vartheta_t(i) \geq B_0 \cdot \Lambda_t\} \quad \text{mit} \quad \Lambda_t = \max_j \vartheta_t(j)$$

Strahlbreite & Rang der wahrscheinl. Zustandsfolge



Die n besten Wortketten

(Näherungsweise) Berechnung mit n-Best-Viterbi-Algorithmen (NVA)

- ***Zustandsbezogener NVA***
  hält in jedem Gitterpunkt (t, j) die n besten Kandidaten in bewertungssortierter Liste $\mathcal{D}_t(j)$ und berechnet

$$\vartheta_t^{(k)}(j) \;=\; \max^{(k)} \underbrace{\left\{\vartheta_{t-1}^{(l)}(i) \cdot a_{ij} \cdot b_j(\boldsymbol{x}_t) \mid 1 \leq i \leq N,\; 1 \leq l \leq n\right\}}_{\mathcal{D}_t(j)}$$

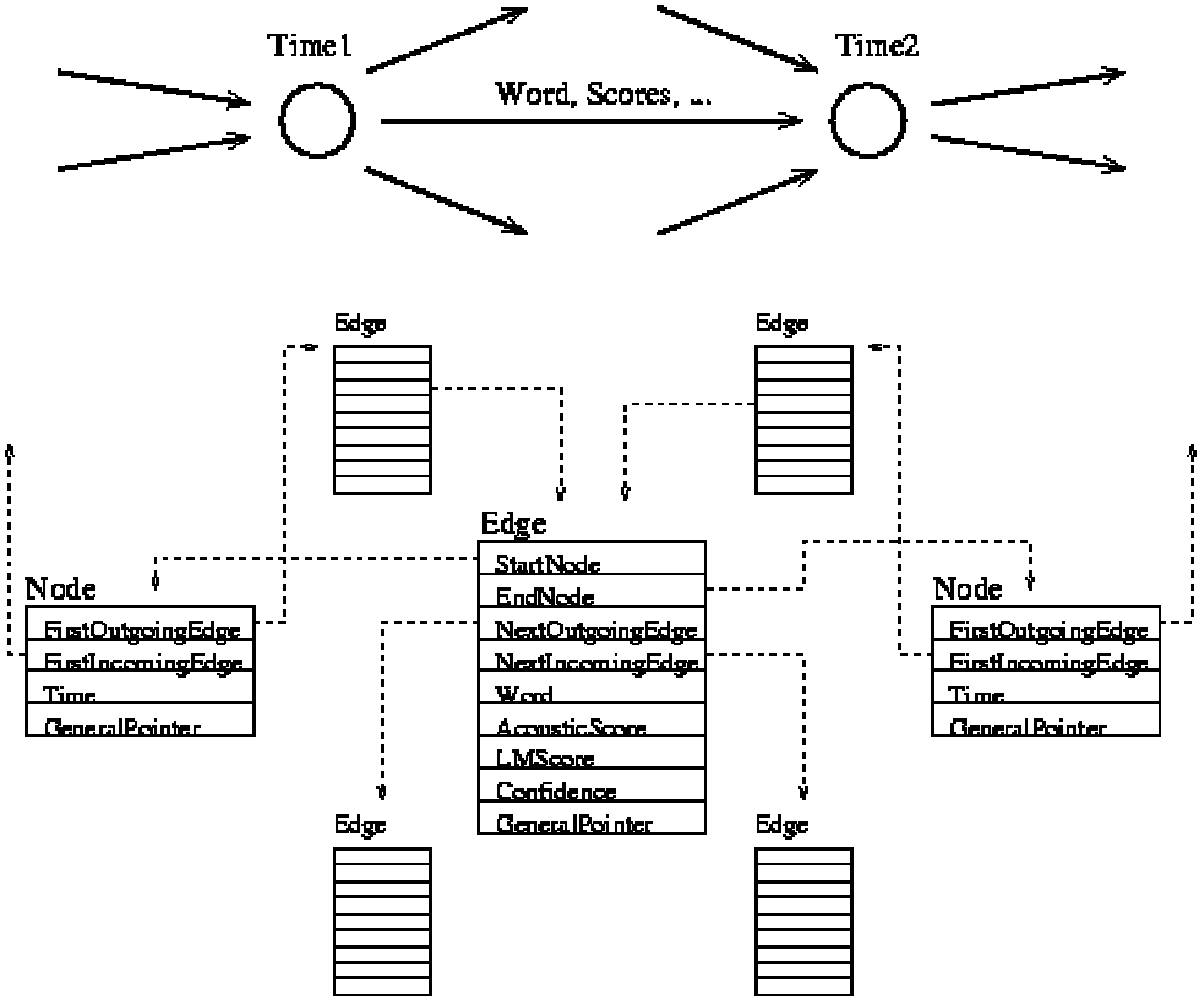
- ***Satzbezogener NVA***
  rekombiniert konkurrierende Kandidaten für gleiche Wortfolgen

- ***Gitterbezogener NVA***
  keine Listen im Wortinneren, nur das *dichte Wortgitter*

$$\{\langle w,\; \vartheta_t(w),\; \tau_t(w)\rangle \mid t = 1, \ldots, T,\; w \in \mathcal{W}\}$$

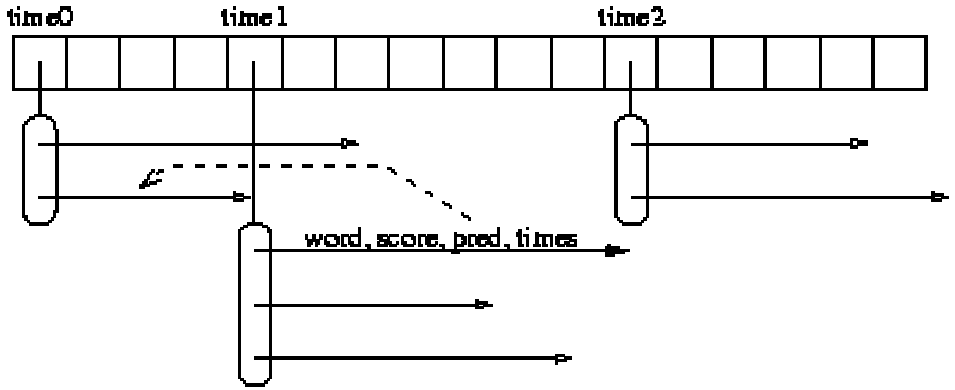
Lattices:

- Use rather general LMs (Shakespeare bigram) and keep many edges (dense lattice)
- Write all edges with acoustic and Lm Score into „lattice“ = mostly time organised graph.
- On dense lattice, do more specific „rescoring“: examples:
  - More specific LMs (Hamlet trigram)
  - Silence penalties
  - Word transition penalties
  - LM penalties

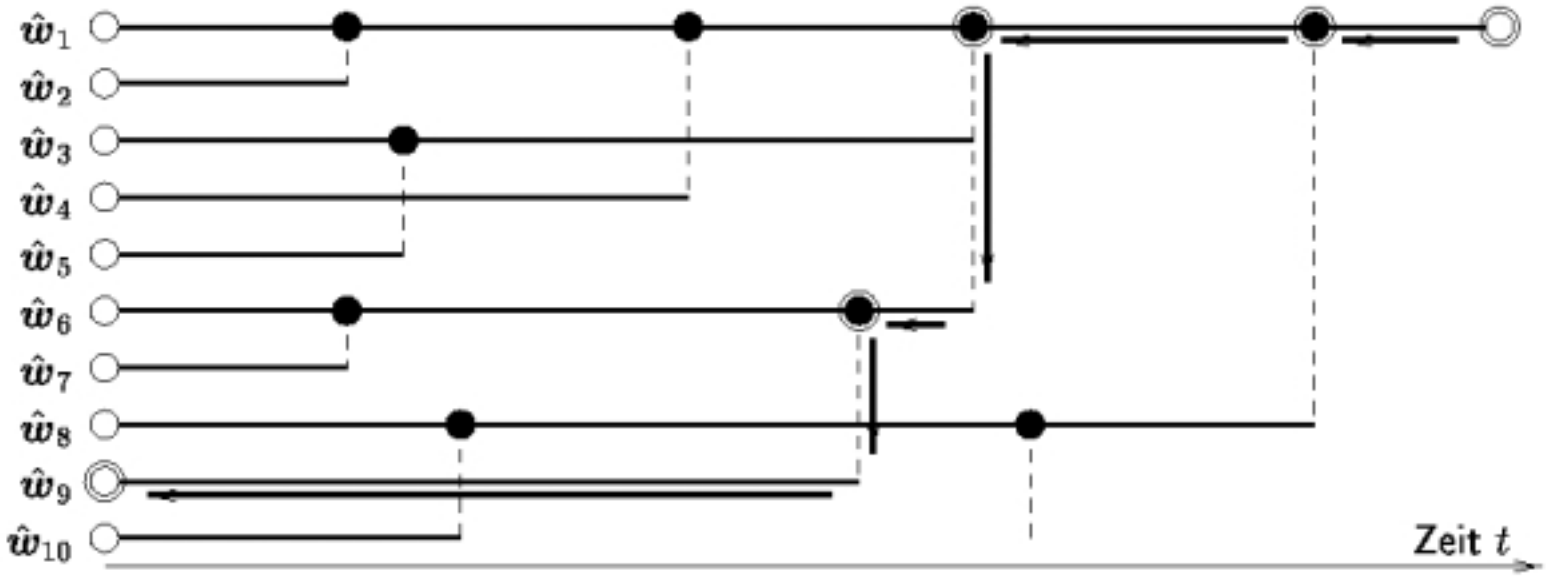

Time1
Time2
Word, Scores, ...
Edge
Edge
Edge
StartNode
EndNode
NextOutgoingEdge
NextIncomingEdge
Word
AcousticScore
LMScore
Confidence
GeneralPointer
Node
FirstOutgoingEdge
FirstIncomingEdge
Time
GeneralPointer
Node
FirstOutgoingEdge
FirstIncomingEdge
Time
GeneralPointer
Edge
Edge

Example of new graph oriented lattice:

Example of old edge oriented lattice:



Rekonstruktion der 10 besten Ketten aus dem dichtem Wortgitter



Lattice:

Header

N=24 L=39

Node definitions

I=0 t=0.00

I=1 t=0.25

I=2 t=0.26

I=3 t=0.61

Link definitions.

J=0	S=0	E=1	W=!ENTER	v=0	a=-1432.27	l=0.00
J=1	S=0	E=2	W=!ENTER	v=0	a=-1500.93	l=0.00
J=2	S=0	E=3	W=!ENTER	v=0	a=-3759.32	l=0.00
J=3	S=0	E=4	W=!ENTER	v=0	a=-3829.60	l=0.00

How to:

Start	End	cumulatexd probability			
0	1	-1432,27-0	=	-1432,27	
0	2	-1500,93-0	=	-1500,93	
1	5	(-2434,05-87,29)-1432,27	=	-3953,61	
2	5	(-2431,55-87,29) -1500,93	=		

Result

4	!ENTER
18	IT

Total score: -20218.25

Today:
Large Vocabulary Speech Recognition

- Refresh
- Gaussian densities
- What about LVSR?
- Integrating the Language Model
- Search techniques
- Lattice techniques