Rheinisch Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI
Prof. Dr.-Ing. Hermann Ney

Seminar Speech Recognition and Language Processing im SS 2003

Multi-Pass Search

Arne Theres

30.7.2003

Multi-Pass Search

1. Einführung und Motivation
2. Schnittstellen zwischen den Modulen
3. Das akustische Fast Match
4. Architektur zweier Multi-Pass-Systeme
5. Zusammenfassung

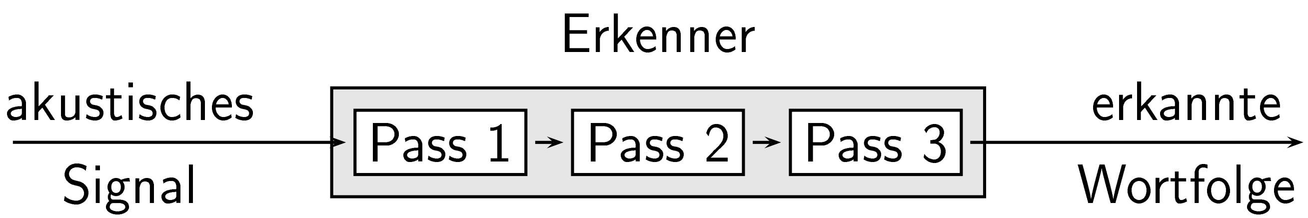


Schaubild der Multi-Pass-Suche

Kapitel 1:

Einführung und Motivation

Probleme der Spracherkennung mit großem Vokabular

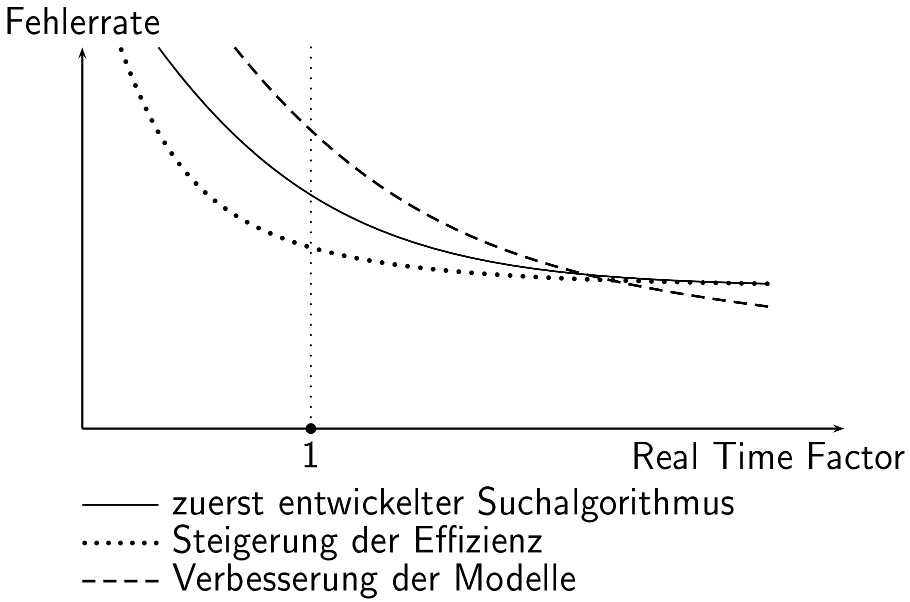
- Suchproblem groß und rechenintensiv
- komplizierte Modelle und Algorithmen

Vorteile mehrphasiger Suchverfahren

- Steigerung der Effizienz
- positiver Nebeneffekt: Modularisierung des Suchproblems

Tradeoff zwischen Fehlerrate und Effizienz

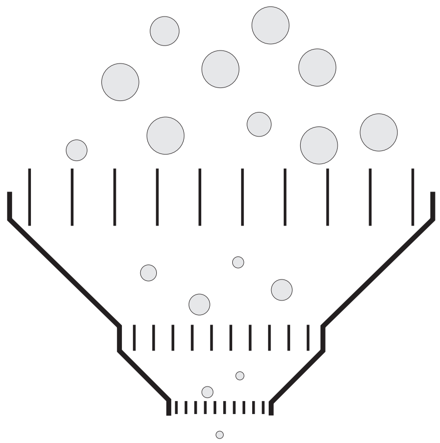
- oftmals: komplexere Modelle $\rightarrow$ schlechtere Effizienz
- schlechtere Effizienz $\rightarrow$ höhere Pruning-Schwelle für gleichen Real Time Factor



- Multi-Pass-Suche kombiniert Vorteile von Effizienz und Genauigkeit

Prinzip der schrittweisen Verfeinerung

- 1. Phase („Hypothesenbildungsphase“): erzeugt eine Vorauswahl für die Hypothesen; grobe, aber effiziente Modelle
- 2. Phase („Testphase“, „Rescoring“): berechnet auszugebende Lösung; detaillierte und rechenaufwändigere Modelle



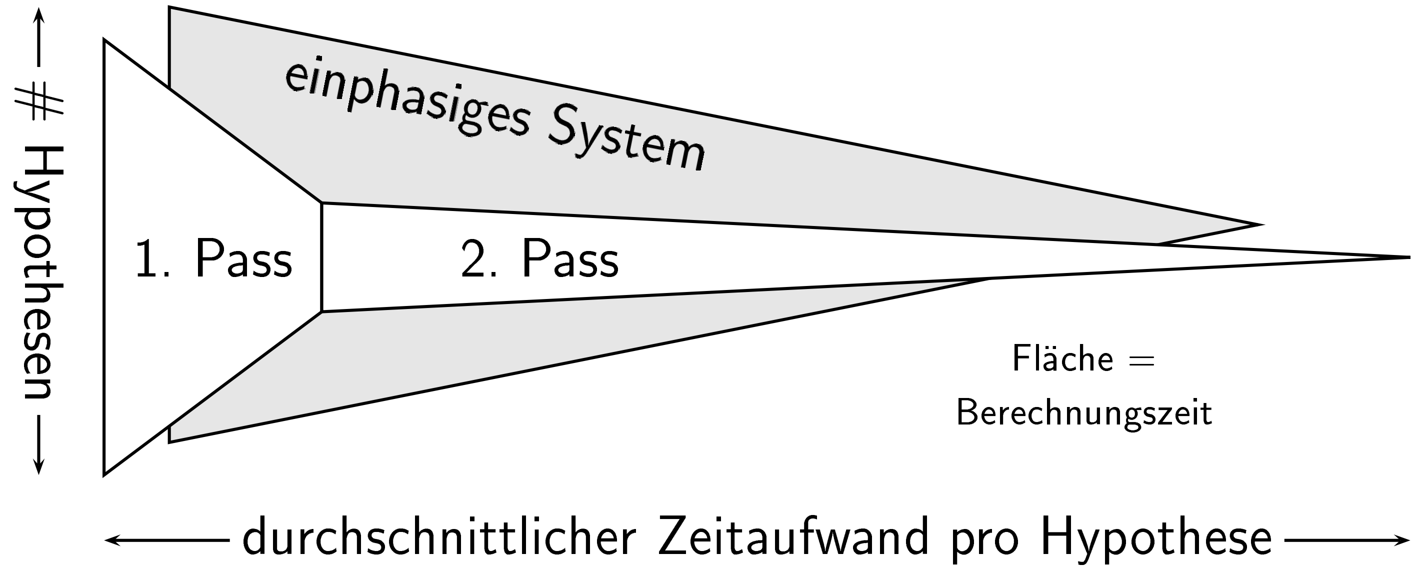
Darstellung der Multi-Pass-Suche als mehrstufiges Filtersystem

Prinzip der schrittweisen Verfeinerung

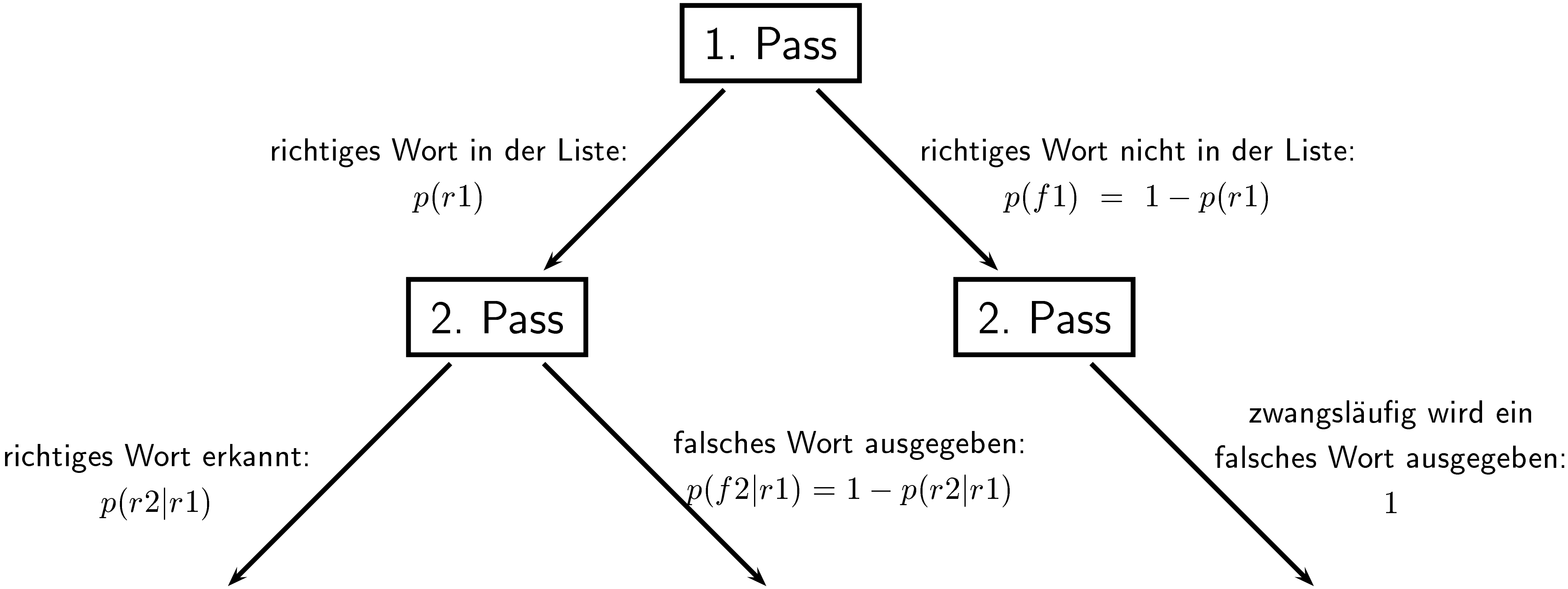
- Vorauswahl muss groß genug sein, um zu garantieren, dass keine (oder nur sehr wenige) Lösungen bezüglich des Testmodells verloren gehen
- Möglichkeiten für die Aufteilung / Art der Hypothesen:
  - Phonemhypothesen: Erkennung mit kontextunabhängigen Phonemen im 1. Pass und mit Triphonen im 2. Pass
  - Worthypothesen: erzeuge zu jedem Timeframe eine Liste der zu startenden Worthypothesen
  - Satzhypothesen: Neubewertung der möglichen Satzhypothesen mit einem besseren Sprachmodell, z.B. einer probabilistischen Grammatik

Effizienzsteigerung bei 2-Pass-Einzelworterkennung

- 1. Pass: soll eine Vorauswahl für zu startende Worthypothesen liefern
  \# Worthypothesen = Vokabulargröße V
  geringer Zeitaufwand pro Hypothese
- 2. Pass: # Worthypothesen << V
  hoher Zeitaufwand pro Hypothese

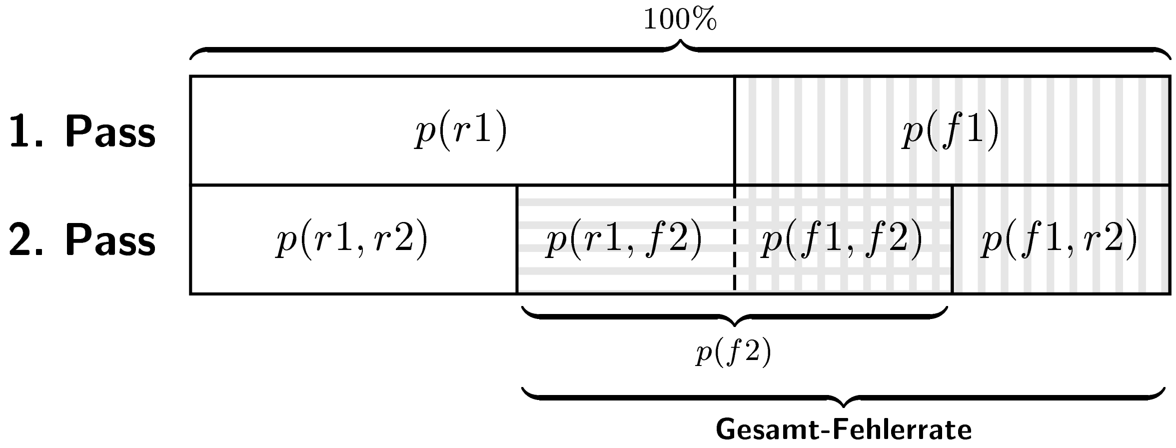


Zusammensetzung der Fehlerrate bei Einzelworterkennung



- Notation: r: richtig, f: falsch; 1: erster Pass, 2: zweiter Pass
- Gesamt-Fehlerrate: $p(r1) \cdot p(f2|r1) + p(f1) \cdot 1 \;=\; p(r1, f2) + p(f1)$

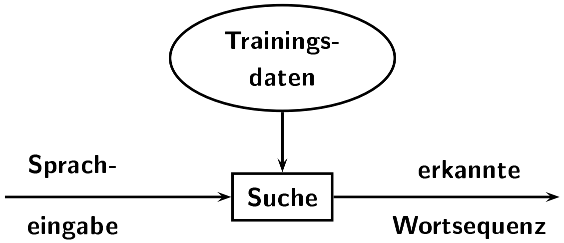
Zusammensetzung der Fehlerrate



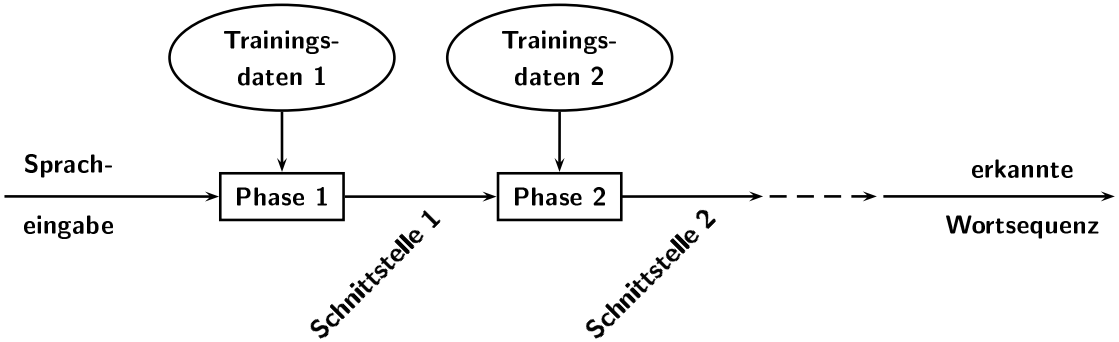
- $p(f1, r2)$: Teil der Fehlerrate, der nur auf den 1. Pass zurückzuführen ist
- *admissibler Algorithmus*: keine zusätzlichen Fehler durch 1. Pass
- Admissibilität nicht praktikabel, da sonst Vorauswahls-Liste zu groß würde

Modularisierung des Suchproblems

1-Pass-Suche:



Multipass-Suche:



- jede Suchphase kann in eigenes Modul eingebettet werden → Suchphasen können unabhängig voneinander programmiert und getestet werden
- Schnittstelle gibt Hypothesen-Vorauswahl weiter

Kapitel 2:

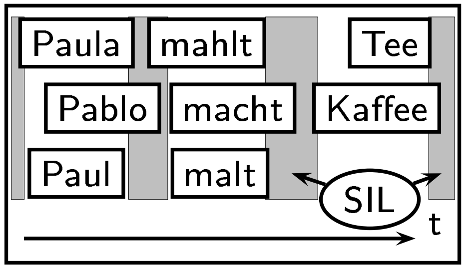
Schnittstellen zwischen den Modulen

Zwei Möglichkeiten für Zwischenrepräsentation von Satzhypothesen

N-BEST-LISTE

1. Paula mahlt Kaffee
2. Paula malt Kaffee
3. Pablo malt Kaffee
4. Paula macht Kaffee
5. Paul mahlt Kaffee

WORTGITTER



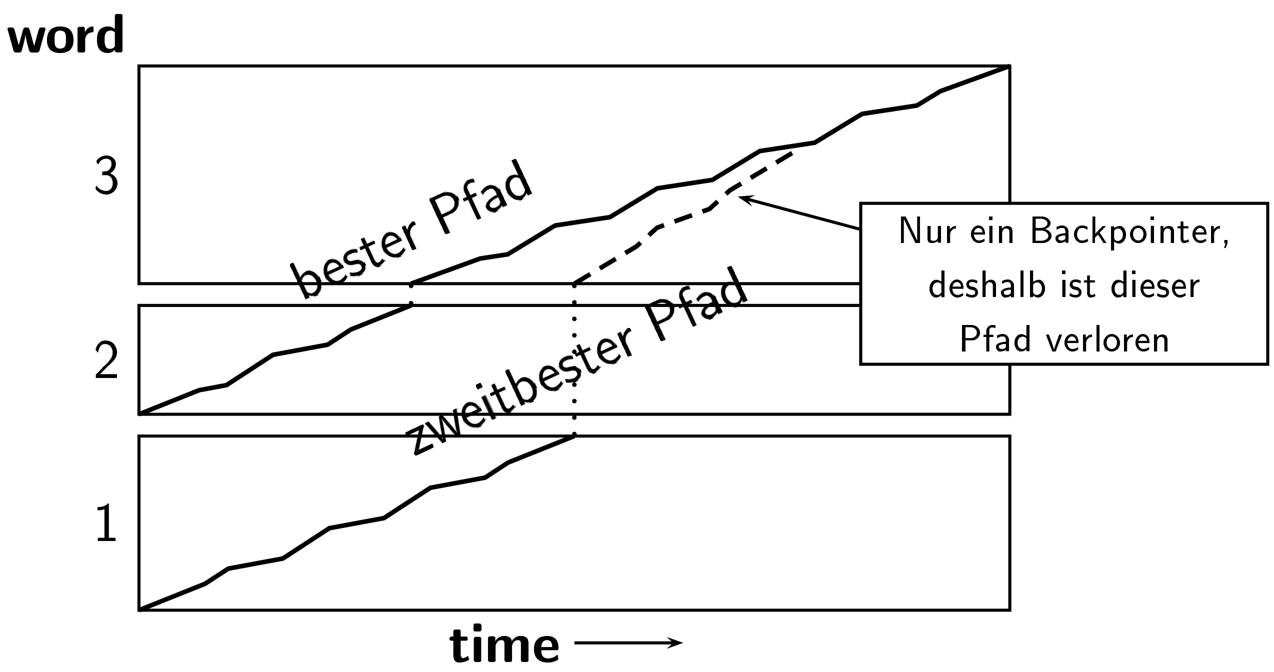
	N-BEST-LISTE	WORTGITTER
Beschreibung:	• Liste der N bestbewerteten Wortketten	• Menge von Wortkanten (w, t_b, t_e, q), wobei w Wortidentität, t_b und t_e Zeitgrenzen und q Bewertung
Vorteile:	• universell anwendbar • ermöglicht einfaches Rescoring	• kompakte Darstellung • Angabe der Zeitgrenzen vorteilhaft für akustisches Rescoring
Nachteile:	• Satzhypothesen unterscheiden sich oft nur paarweise an einer Wortposition $\rightarrow$ hohe Redundanz	• keine Sortierung bzgl. der besten Satzwahrscheinlichkeiten, deshalb umständlicher für manche Rescoring-Algorithmen

Implementation der N-Best-Liste: 3 Ansätze

- **Traceback-Dependent N-Best**:
  - geringe Komplexität (kaum Mehraufwand gegenüber 1-Best)
  - jedoch ungenau: sogar hochbewertete Satzhypothesen können unterbewertet oder ausgelassen werden
- **Sentence-Dependent N-Best**:
  - hohe Komplexität (N-facher Rechenaufwand gegenüber 1-Best)
  - liefert dafür immer die exakte Lösung
- Kompromiss: **Word-Dependent N-Best**

Traceback-Dependent N-Best

- N-faches Tracebackfeld → Rekonstruktion der N besten Sätze
- Schwachstelle: keine Trennung der Hypothesen bezüglich ihrer Historie

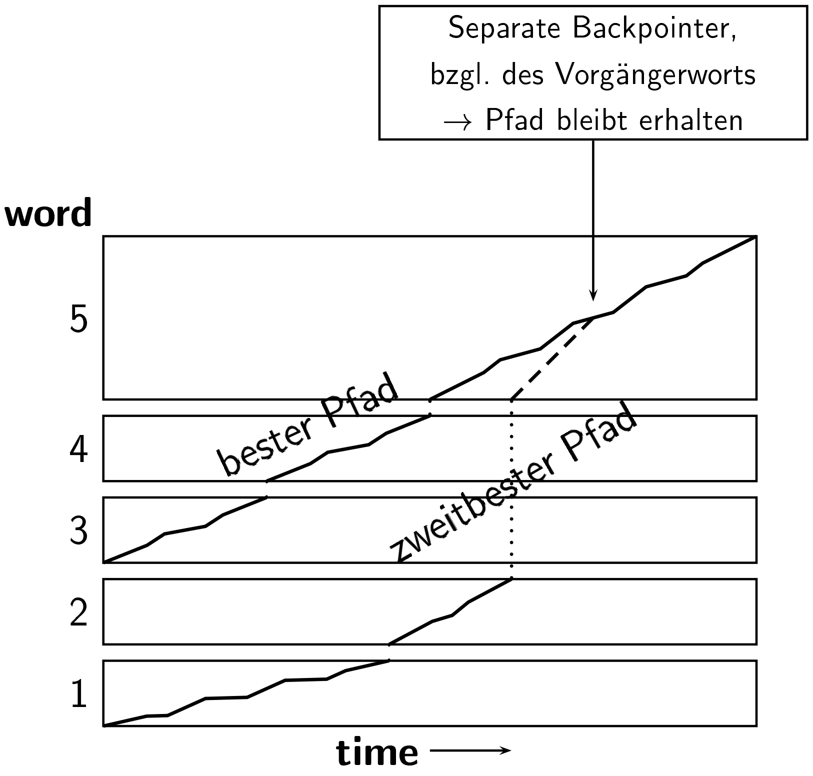


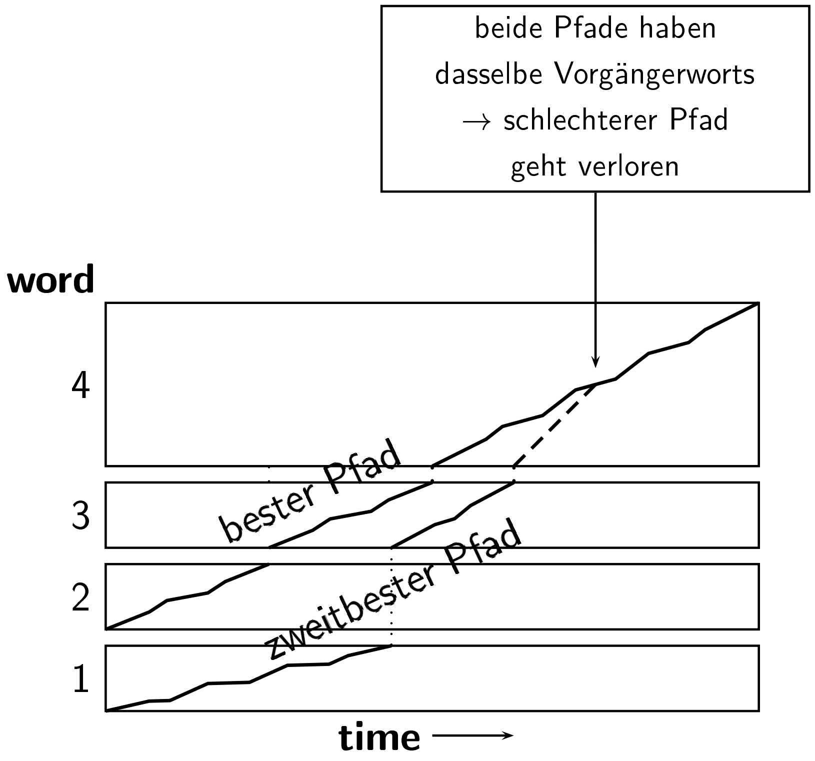
Sentence-Dependent N-Best

- N-faches Tracebackfeld
- Baumkopien bzgl. der *gesamten* Historie, d.h. N-fache Baumkopien
  $\rightarrow$ N-facher Rechenaufwand gegenüber 1-Best
- für jede der N alternativen Historien ein separater Backpointer
  $\rightarrow$ exakte Lösung

Word-Dependent N-Best: benutzt Wortpaar-Approximation

- N-faches Tracebackfeld + Baumkopien bzgl. des Vorgängerworts





Implementation des Wortgitters (Word Lattice)

- Algorithmen zur Generierung der N-Best-Liste übertragbar auf Wortgitter
- Wortgitter ersetzt Tracebackfeld
- nach Wortende: Eintrag einer Wortkante ins Gitter

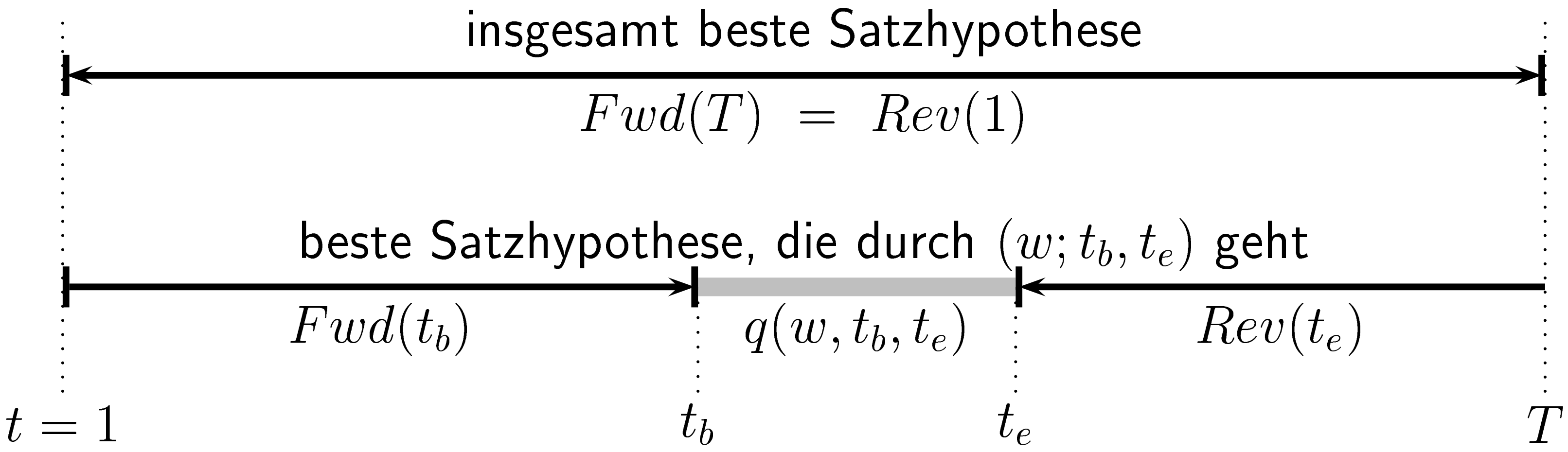
Lattice-Pruning

- findet vor dem Rescoring-Vorgang statt
- Ziel: Größe des Lattice zu reduzieren, ohne Fehlerrate merklich zu erhöhen
- zwei Möglichkeiten: Forward-Pruning (kann schon während der Erkennung vorgenommen werden) und Forward-Backward-Pruning

Forward-Backward-Pruning

- kann erst nach Vorliegen des kompletten Wortgitters erfolgen
- Ausnutzung des zusätzlichen Wissens, dass der Satz bei $t = T$ endet
- Neubewertung einer Wortkante im Gitter im Kontext des gesamten Satzes

Forward-Backward-Pruning

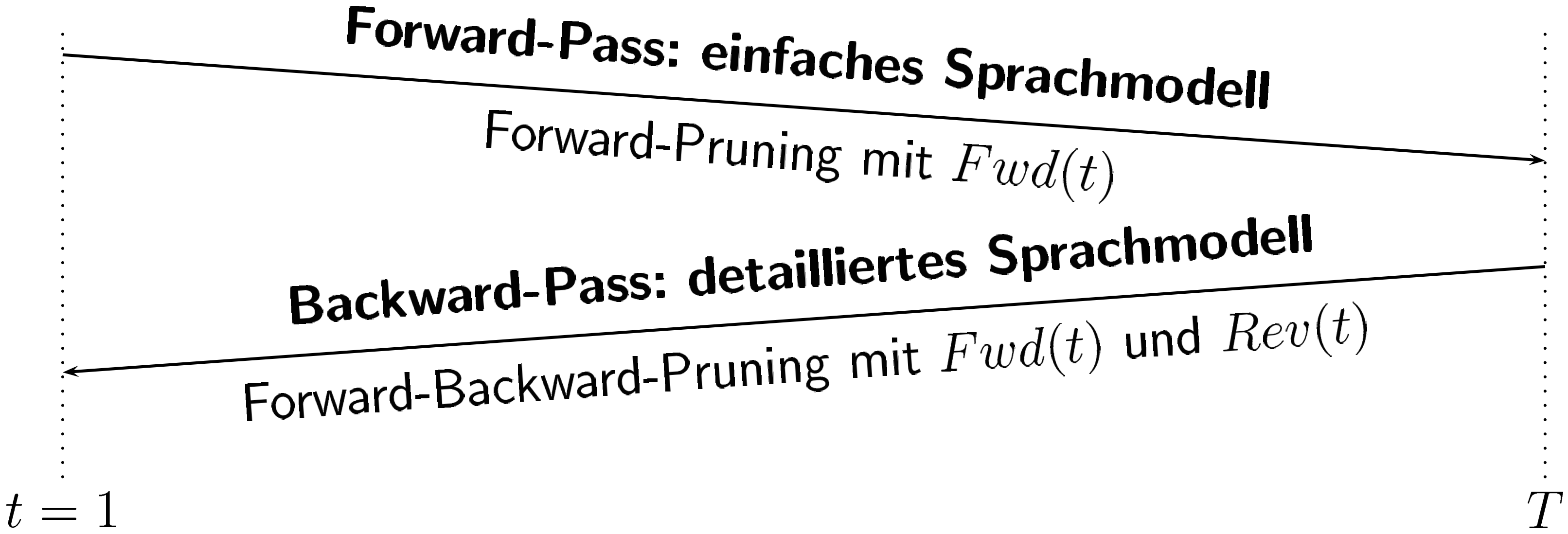


- Worthypothese $(w; t_b, t_e)$ mit W'keit $q(w; t_b, t_e)$ überlebt Pruning, wenn

$$\frac{Fwd(t_b) \cdot q(w; t_b, t_e) \cdot Rev(t_e)}{Fwd(T)} \; > \; f_{lat\_fb}$$

- $f_{lat\_fb} < 1$ Forward-Backward-Pruningschwelle

Forward-Backward-Search



- zweiphasiges Suchverfahren mit Wortgitter als Zwischenrepräsentation
- in der Rückwärtssuche starte zum Zeitpunkt $t = t_e$ nur Wörter w, die als Kandidaten $(w; t_b, t_e)$ im Gitter stehen
- Rückwärtslauf deshalb sehr schnell (ca. 20 ms)
- Rückwärtslauf berechnet exakte Startzeiten, benutzt detailliertere Modelle

Wortgitter vs. N-Best

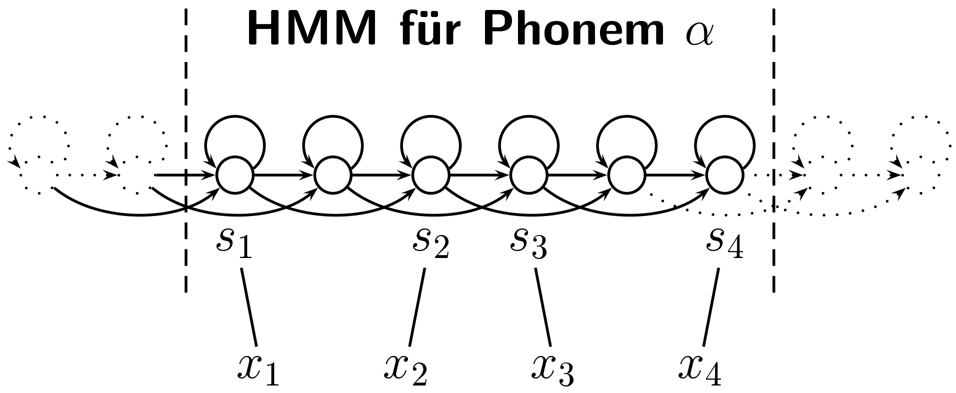
- Wortgitter kompakter
- viele Rescoring-Algorithmen auf Wortgitter effizienter durchführbar
- bei manchen Algorithmen jedoch keine wesentlich effizientere Lösung auf dem Wortgitter bekannt
- allerdings: Wortgitter kann in N-Best-Liste konvertiert werden; umgekehrte Konvertierung nicht möglich
- Fazit: Wortgitter in jeder Hinsicht überlegen

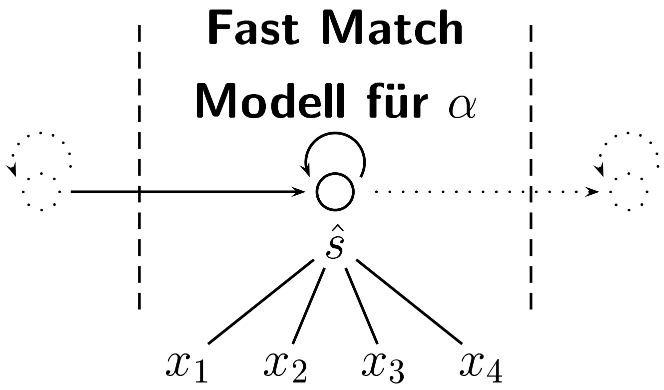
Kapitel 3:

Das akustische Fast Match

Das akustische Fast-Match

- Vereinfachung des detaillierten HMM-Phonemmodells
- ermöglicht grobe, aber effiziente Erkennung $\rightarrow$ nur geeignet für 1. Pass
- Fast-Match-Wahrscheinlichkeit als obere Schranke für detaillierte W'keit





HMM-Wahrscheinlichkeit $D(x_1^T|\alpha)$ eines Phonems α

$$D(x_1^T|\alpha) \;:=\; \max_{s_1^T} \prod_{t=1}^{T} \underbrace{p(s_t|s_{t-1},\alpha)}_{\text{Transitionsw'keit}} \cdot \underbrace{p(x_t|s_t,\alpha)}_{\text{Emissionsw'keit}}$$

Fast-Match-Wahrscheinlichkeit $F(x_1^T|\alpha)$ eines Phonems α

$$F(x_1^T|\alpha) \;:=\; p(\hat{s}|\alpha) \cdot \prod_{t=1}^{T} p(x_t|\hat{s},\alpha)$$

ergibt sich durch 2 Abschätzungen nach oben:

1. Transitionswahrscheinlichkeit: wähle besten Pfad

$$p(\hat{s}|\alpha) \;:=\; \max_{T} \max_{s_1^T} \prod_{t=1}^{T} p(s_t|s_{t-1},\alpha)$$

2. Emissionswahrscheinlichkeit: maximiere über alle Zustände

$$p(x_t|\hat{s},\alpha) \;:=\; \max_{s\in\{1 \ldots S(\alpha)\}} p(x_t|s,\alpha)$$

Pseudocode für die beiden Phonemmodelle

```
procedure DetailedMatch(x_1 ... x_T , α):D(x_1^T|α)
begin
  for t := 1 to T do
      for s := 1 to S do
          q(t,s) := max_{Δs∈{0,1,2}} {q(t−1, s−Δs)
                        · p(s|s−Δs, α) · p(x_t|s_t, α)}
      end
  end
  return D(x_1^T|α) := q(T,S)
end
```

```
procedure FastMatch(x_1 ... x_T , α):F(x_1^T|α)
begin
  for t := 1 to T do
      q(t) := q(t−1) · p(x_t|ŝ, α)
  end
  return F(x_1^T|α) := q(T) · p(ŝ|α)
end
```

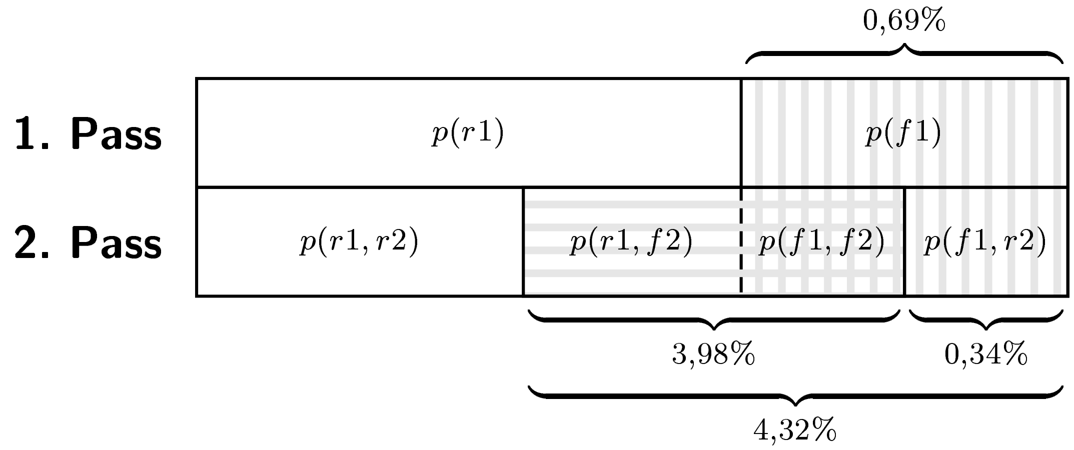
- detailliertes HMM-Modell: berechnet $T \cdot S$ Emissionsw'keiten
- Fast-Match-Modell: berechnet nur T Emissionsw'keiten

Anmerkungen zum akustischen Fast-Match

- Bewertung unabhängig von der Reihenfolge der x_t
- i6-Spracherkenner: in der Baumsuche mit Triphonen wird mit einem integrierten kontextunabhängigen Fast-Match ein Pruning-Lookahead von 60ms erzeugt
- IBM: Lookahead um ein ganzes Wort
  $\rightarrow$ N-Best-Liste der besten Wörter, die im nächsten Pass mit einem detaillierten Phonemmodell neubewertet werden
- bei Lookahead mit variabler Größe des Vorschaufensters: Normalisierung der Bewertungen erforderlich!

Ergebnisse von IBM

- Zusammensetzung der Fehlerrate im IBM-Erkenner:



- Rang der korrekten Wörter in der N-Best-Liste:

	∅-Länge der Liste	∅-Rang des korrekten Worts	Prozentsatz der Worte mit Rang $= 1$	> 1	> 10	> 100
AC	1736	14.61	46.8	53.2	15.53	2.92
AC+LM	47.95	1.88	86.39	13.61	2.84	0.69

Kapitel 4:

Architektur zweier Multi-Pass-Systems

Architektur des BBN-Biblos-Systems

- akustisches Rescoring in sieben (!) Phasen mit Sprecheradaption

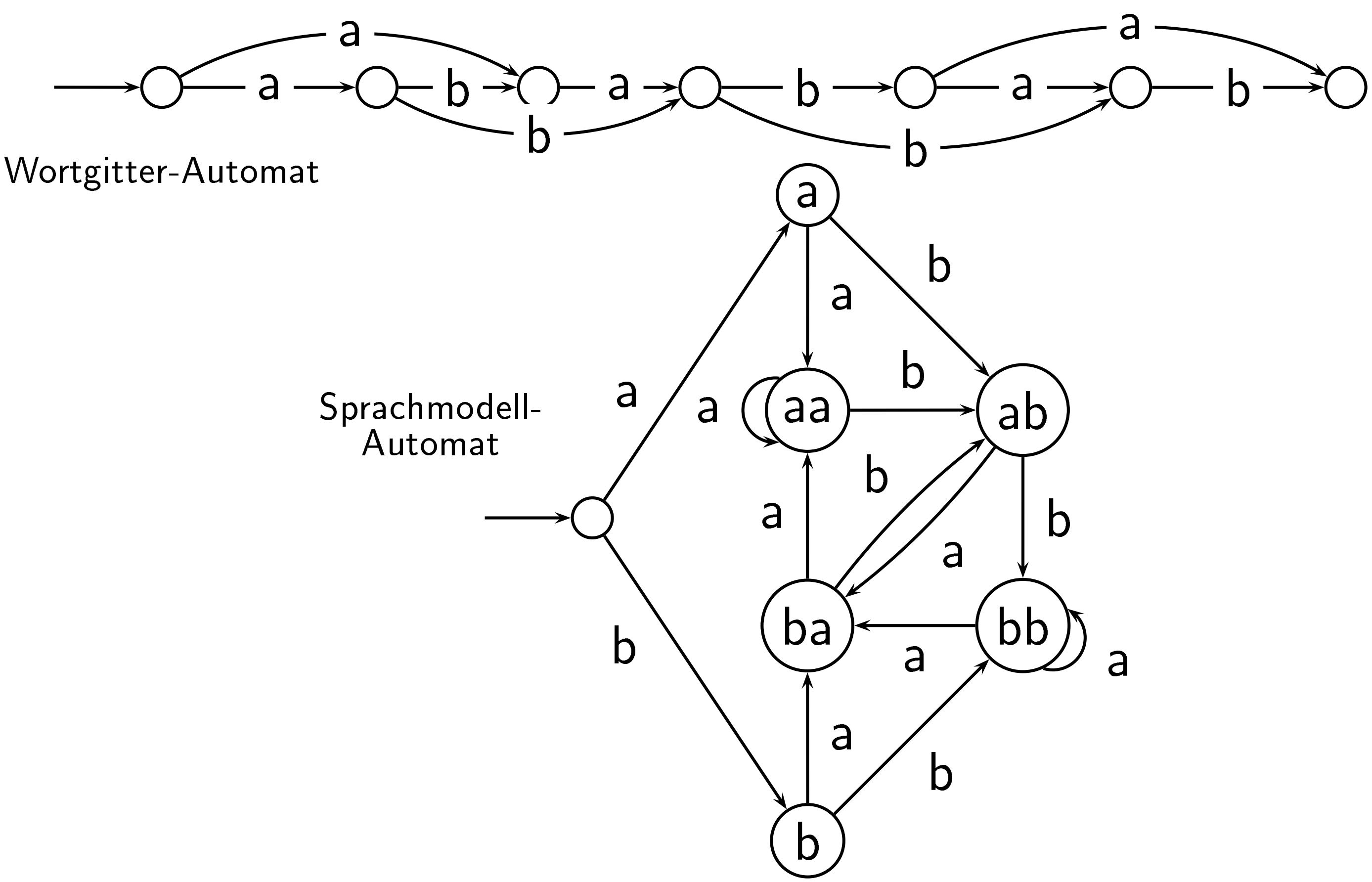
1. Forward-Pass mit akustischem Fast Match; Bigramm-Sprachmodell; Zwischenrepräsentation: Wortgitter
2. Backward-Pass; keine Acrossword-Modelle und keine kontextabhängigen Phonemmodelle; Trigramm-Sprachmodell; Zwischenrepräsentation: Wortgitter
3. Forward-Pass; Modelle wie im 2. Pass; 2. und 3. bilden eine Backward-Forward-Suche; Zwischenrepräsentation: Wortgitter
4. Forward-Pass; diesmal mit Acrossword-Modellen und kontextabhängigen Phonemmodellen (Phonemmodelle mit State-Tying mittels CART); weiterhin Trigramm-Modell; erzeugte Zwischenrepräsentation: N-Best-Liste mit N=100
5. akustisches Rescoring mit sprecherabhängigen Modellen (mittels MLLR); Part-Of-Speech (POS) - geglättetes Trigramm-Modell

6. + 7. ähnlich zu 5.

AT&T: 6-gramm-Sprachmodell-Rescoring

1. Erzeugung eines Phonemgitters mit kontextabhängigen Phonemen mittels Transducer-Viterbi-Suche
2. Konvertierung des Phonemgitters in ein Wortgitter; Bigramm-Sprachmodell mit 3,6 Mio. Bigrammen
3. Forward-Backward-Pruning des gesamten Wortgitters
4. Rescoring des Wortgitters mit einem 6-gramm-Sprachmodell mit 20 Mio. m-Grammen

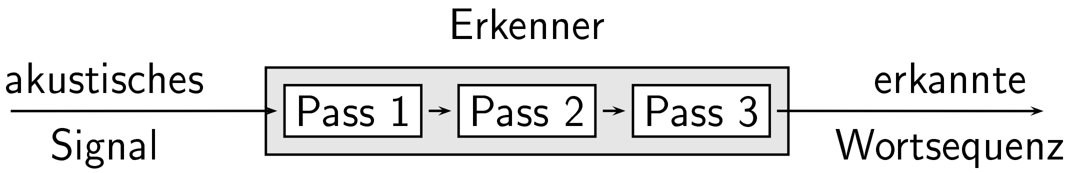
Produktautomaten-Modell für das Sprachmodell-Rescoring



Kapitel 5:

Zusammenfassung

Zusammenfassung



- Multi-Pass-Systeme = mehrstufiges Filtersystem für Hypothesen
- höhere Effizienz durch grobe Vorauswahl (z.B. Fast Match)
- Zwei Schnittstellen: N-Best-Listen und Wortgitter (Lattices)
- mehrere Möglichkeiten zur Erzeugung von N-Best-Listen:
  Sentence-, Traceback- und Word-Dependent N-Best
- Nachteil von Multi-Pass-Systemen: Verzögerung der Ausgabe
- möglicher Ausweg: Rescoring eines Satzes schon während dessen Erkennung