

---

# Probabilistische kontextfreie Grammatiken und Parsing

---

Sebastian Pado

03.02.2004

---

# DasZiel

- Freien Text parsen („Robust Parsing“)
  - Freier Dialog
  - Große Textmengen (Internet)
  
- Dazu notwendig:
  1. Große und flexible Grammatik
  2. Effizientes Parsing-Verfahren

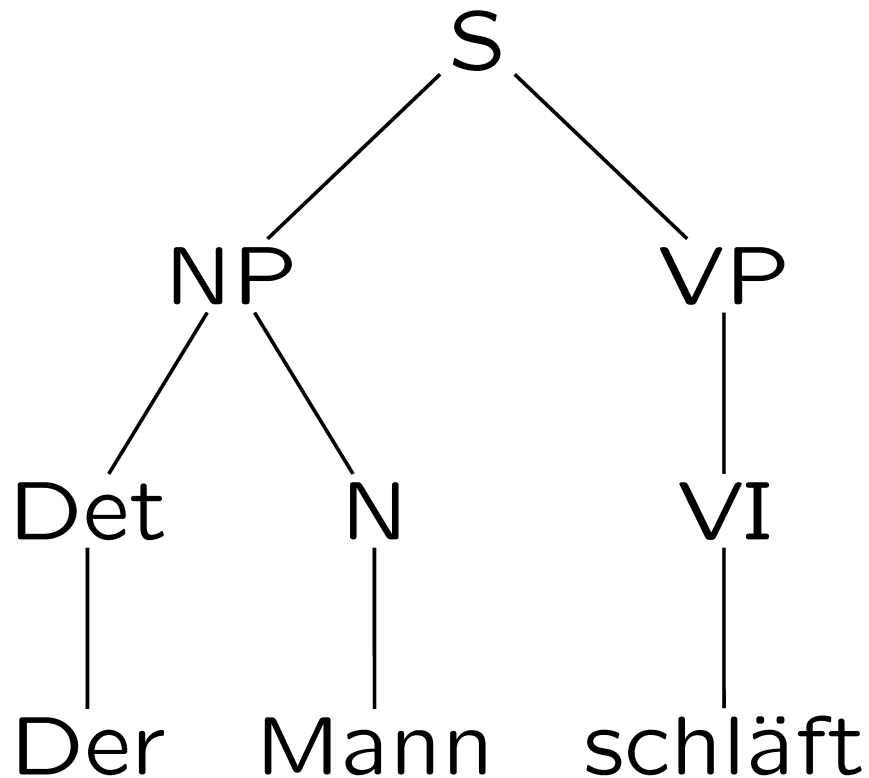
---

# Erstellung einer großen Grammatik

- Von Hand schreiben ☹
- Idee: In kontextfreier Bankbank steckt implizites linguistisches Wissen
- **Naive Extraktion einer Baumbank-Grammatik:**
  - Eine (kontextfreie) Baumbank nehmen
  - Alle Parsebäume in „lokale Bäume“ aufteilen
  - Jeden lokalen Baum als Regel hinzufügen
- Ergebnis: Baumbank-Grammatik

# NaiveBaumbank -Grammatiken

Parsebaum



Abgeleitete Regeln

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $Det \rightarrow Der$   
 $N \rightarrow Mann$   
 $VP \rightarrow VI$   
 $VI \rightarrow schläft$

---

# Grammatikgröße und Mehrdeutigkeit

- Bessere Abdeckung = mehr Mehrdeutigkeit
  - Wenige Regeln: Weniger erkennbar, aber wenig Ambiguität
  - Viele Regeln: Viele Konstruktionen erkennbar, viel Ambiguität
- Baumbank-Grammatiken: sehr viele Regeln
  - Nötig: Strategien für Ambiguität
- Beispiel: Die Frau sieht den Hund und den Mann  
[PP<sub>1</sub> auf dem Hügel] [PP<sub>2</sub> mit dem Teleskop]
  - (mindestens  $3 * 4 = 12$  Lesarten)
  - Sätze mit 40 Wörtern können über 100 000 Lesarten haben

---

# Form von Baumbank - Regeln

Gelernte Regeln sind vom Annotationsschema der annotierten Baumbank abhängig

- Tiefere Bäume: mehr Nonterminale, weniger Regeln pro Nonterminal
- Flachere Bäume: weniger Nonterminale, mehr Regeln pro Nonterminal

Kontextfreie Baumbank-Grammatiken sind **sehr flach** / generieren stark über

# Penn Treebank vs. TIGER

- Intransitive Verben: NP – V (schlafen)
- Transitive Verben: NP – V – NP (sehen)
- Ditransitive Verben: NP – V – NP – NP (geben)

## Penn Treebank

|    |   |         |
|----|---|---------|
| S  | → | NP VP   |
| VP | → | V       |
| VP | → | V NP    |
| VP | → | V NP NP |

## TIGER

|   |   |            |
|---|---|------------|
| S | → | NP V       |
| S | → | NP V NP    |
| S | → | NP V NP NP |

# Probleme von Baumbank - Grammatiken

- Regeln werden automatisch abgeleitet
  - Baumbank-Grammatiken sind sehr groß
    - 19.000 Regeln für 18.000 Sätze NEGRA-Korpus  
Eingeschränkte linguistische Relevanz
  - **Naive** Baumbank-Grammatiken haben kein Konzept von gradueller Grammatikalität (letzte Vorlesung)  
Keine Möglichkeit, von der „besten Analyse zu sprechen“
  - Baumbank-Grammatiken generieren stark über
    - Sehr ambig
      - Schon für Sätze unter 10 Wörtern mehrere 100 Analysen  
Riesiger Suchraum für Parsing



# Probabilistische Grammatiken( PCFGs)

- Wahrscheinlichkeiten für Regeln
  - $S \rightarrow NP VP$  (0.6)
  - $S \rightarrow NP VP PP$  (0.4)
    - „60% aller Sätze haben die Form NP VP, 40% die Form NP VP PP“
- Wahrscheinlichkeiten für Parsebäume
  - Werden aus Regelwahrscheinlichkeiten berechnet
- Vorteile:
  - Graduelle Grammatikalität
    - Beste Analyse = wahrscheinlichster Parsebaum
  - Einschränkung des Suchraumes
    - Verfolge vor allem „wahrscheinliche“ Parsebäume

---

# Schritte für Parsing mit PCFGs

1. Lerne Regeln mit Wahrscheinlichkeiten
2. Berechne Wahrscheinlichkeiten für Bäume aus Regelwahrscheinlichkeiten
3. Finde effizientes Verfahren, um wahrscheinlichste Bäume zu konstruieren

# Schritt1:

## Regelwahrscheinlichkeitenlernen

- **Nicht naive** Baumbank-Grammatik
  - Eine Baumbank nehmen
  - Alle Parsebäume in „lokale Bäume“ aufteilen
  - Jeden neuen lokalen Baum als Regel hinzufügen
  - Regeln zählen und Wahrscheinlichkeiten aus Frequenzen berechnen
    - Für jede Regel R der Form  $LS \rightarrow RS$  berechne bedingte Wahrscheinlichkeit

$$P(RS|LS) = \frac{f(LS \rightarrow RS)}{f(LS)} = \frac{f(LS \rightarrow RS)}{\sum_{RS} f(LS \rightarrow RS)}$$

- Intuition: Anteil der Regel  $LS \rightarrow RS$  an allen Vorkommen von LS (mit beliebigen RS)

# „Echte“ Regelfrequenzen in NEGRA

...

1 NP ADJA \$\*LRB\* \$, ADJA NN

12 NP ADJA \$\*LRB\* NN

1 NP ADJA \$\*LRB\* NN \$\*LRB\* PP

2 NP ADJA \$\*LRB\* NN PP

8 NP ADJA \$, ADJA NN

1 NP ADJA \$, ADJA NN PP

3 NP ADJA \$, AP NN

1 NP ADJA ADJA ADJA NN

1 NP ADJA ADJA CARD

2 NP ADJA ADJA CNP

1 NP ADJA ADJA NN \$\*LRB\* NE

1 NP ADJA ADJA NN \$\*LRB\* NP

4 NP ADJA ADJA NN NP

51 NP ADJA ADJA NN

...

\$.: Komma

ADJA: Attributives Adjektiv

CARD: Kardinalzahl

CNP: Koordinierte NP

\$\*LRB\*: Klammer

NE: Eigennamen

NN: Normales Nomen

NP: Nominalphrase

PP: Präpositionalphrase

Wenn dies alle NP-Regeln wären,  
dann wäre z.B.

$$P(\text{ADJA LRB NN} \mid \text{NP}) = 12/89$$

$$P(\text{ADJA ADJA NN} \mid \text{NN}) = 51/89$$

$$P(\text{ADJA ADJA CARD} \mid \text{NP}) = 1/89$$

# Regelwahrscheinlichkeiten

- Es gilt:  $\sum_{RS} P(LS \rightarrow RS) = 1$ 
  - Alle Wahrscheinlichkeit auf „gesehene“ RS verteilt
  - Modell „muß“ eine der gesehenen RS wählen
  - Zipf-Verteilung: Viele sehr seltene Regeln
    - Viele Regeln nicht gesehen?
      - Smoothing: Verteile ein bißchen Wahrscheinlichkeit auf mögliche, aber ungesehene Regeln

---

# Schritte für Parsing mit PCFGs

1. Lerne Regeln mit Wahrscheinlichkeiten
2. Berechne Wahrscheinlichkeiten für Bäume aus Regelwahrscheinlichkeiten
3. Finde effizientes Verfahren, um wahrscheinlichste Bäume zu konstruieren

## Schritt2: Baumwahrscheinlichkeiten berechnen

- Baum = Kombination von Regel(anwendungen)

$$P(\text{Baum}) = P(\text{Regel}_1, \text{Regel}_2, \dots, \text{Regel}_n)$$

(gemeinsame Wahrscheinlichkeit)

$$P(\text{Regel}_1, \text{Regel}_2, \text{Regel}_3) = \quad (\text{Kettenregel})$$
$$P(\text{Regel}_1) * P(\text{Regel}_2 | \text{Regel}_1) * P(\text{Regel}_3 | \text{Regel}_2, \text{Regel}_1)$$

- Viel zu kompliziert zum Ausrechnen!

# Unabhängigkeit von Regeln

- Starke Unabhängigkeitsannahme:  
Alle Regeln sind unabhängig
- Erinnerung: Wenn A, B unabhängig,
  - dann beeinflussen sich A und B nicht
  - dann gilt  $P(A|B) = P(A)$  und  $P(B|A) = P(B)$
- $P(\text{Regel}_1, \text{Regel}_2, \text{Regel}_3)$  vereinfacht sich zu  
 $P(R_1) * P(R_2) * P(R_3) = \prod_i P(R_i)$

Einfach zu berechnen – aber ist die Annahme gerechtfertigt?



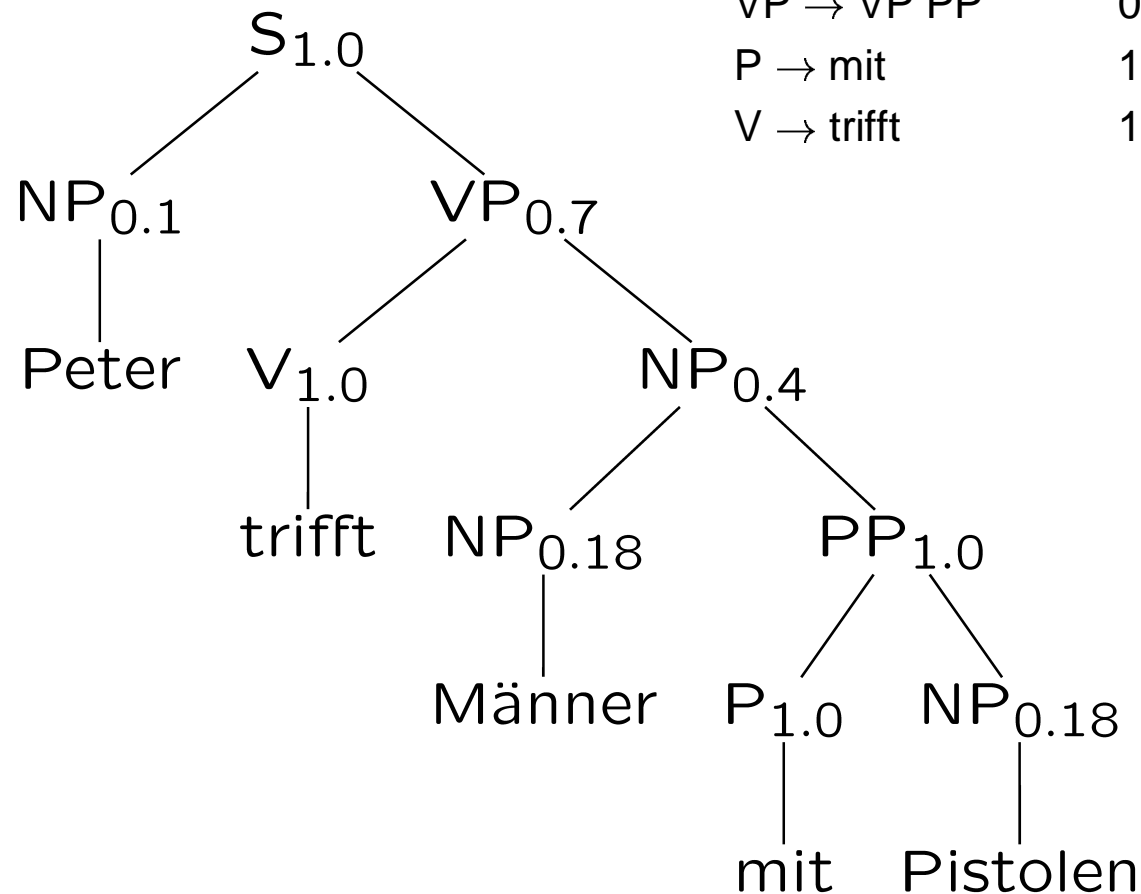
# Interpretation von Top-Down-Parsing als Experiment

- Für jedes Nonterminal (jede LS einer Regel) gibt es einen Topf, in dem die RS liegen
- Beginne mit S als LS
  - Ziehe rechte Seite aus dem LS-Topf
    - $\Omega$  sind RS für LS
    - Merke Wahrscheinlichkeit  $P(r) = P(RS|LS)$
  - Für alle Nonterminale in der gezogenen RS, wiederhole Experiment
  - Wahrscheinlichkeit des ganzen Baumes  
 $P(t) = \prod_{r \in t} P(r)$  (Unabhängigkeitsannahme)

# Beispielgrammatik

|                               |     |                                   |      |
|-------------------------------|-----|-----------------------------------|------|
| $S \rightarrow NP VP$         | 1.0 | $NP \rightarrow NP PP$            | 0.4  |
| $PP \rightarrow P NP$         | 1.0 | $NP \rightarrow \text{Peter}$     | 0.1  |
| $VP \rightarrow V NP$         | 0.7 | $NP \rightarrow \text{Pistolen}$  | 0.18 |
| $VP \rightarrow VP PP$        | 0.3 | $NP \rightarrow \text{Sägen}$     | 0.04 |
| $P \rightarrow \text{mit}$    | 1.0 | $NP \rightarrow \text{Männer}$    | 0.18 |
| $V \rightarrow \text{trifft}$ | 1.0 | $NP \rightarrow \text{Teleskope}$ | 0.1  |

# Beispiel1



$S \rightarrow NP VP$

1.0

$NP \rightarrow NP PP$

0.4

$PP \rightarrow P NP$

1.0

$NP \rightarrow \text{Peter}$

0.1

$VP \rightarrow V NP$

0.7

$NP \rightarrow \text{Pistolen}$

0.18

$VP \rightarrow VP PP$

0.3

$NP \rightarrow \text{Sägen}$

0.04

$P \rightarrow \text{mit}$

1.0

$NP \rightarrow \text{Männer}$

0.18

$V \rightarrow \text{trifft}$

1.0

$NP \rightarrow \text{Teleskope}$

0.1

$$\begin{aligned} P(t) &= 1.0 * 0.1 * 0.7 * \\ &\quad 1.0 * 0.4 * 0.18 * \\ &\quad 1.0 * 1.0 * 0.18 \\ &= 0.0009072 \end{aligned}$$

# Beispiel2

$S \rightarrow NP VP$

1.0  $NP \rightarrow NP PP$  0.4

$PP \rightarrow P NP$

1.0  $NP \rightarrow Peter$  0.1

$VP \rightarrow V NP$

0.7  $NP \rightarrow Pistolen$  0.18

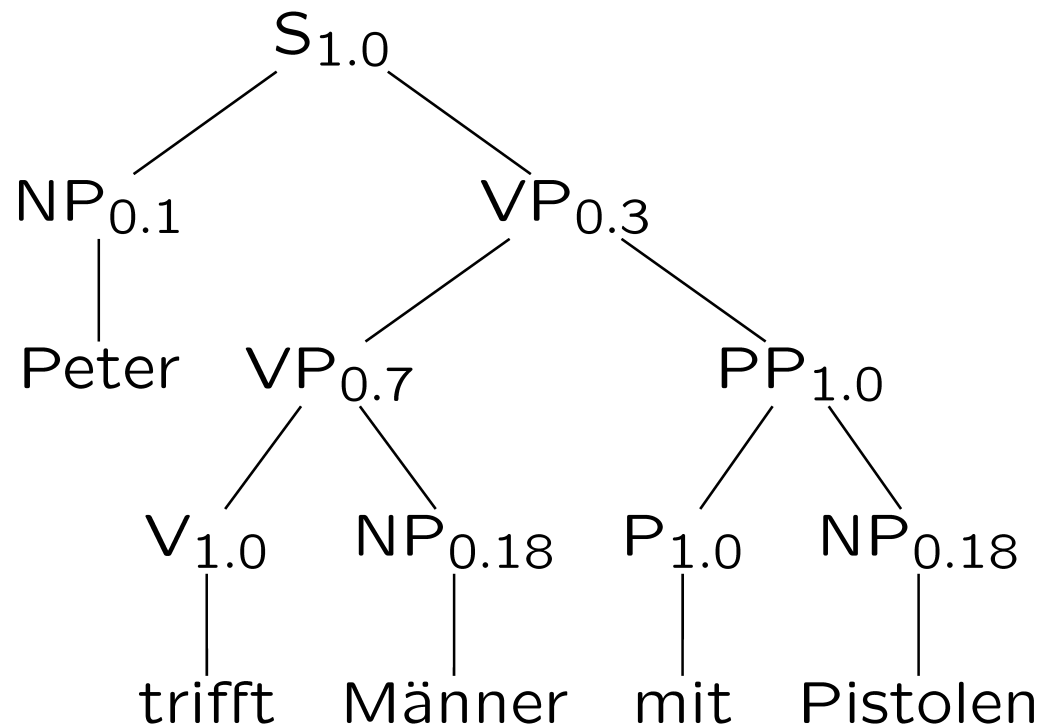
$VP \rightarrow VP PP$

0.3  $NP \rightarrow Sägen$  0.04

$P \rightarrow mit$

1.0  $NP \rightarrow Männer$  0.18

1.0  $NP \rightarrow Teleskope$  0.1



$$\begin{aligned} P(t) &= 1.0 * 0.1 * 0.3 * \\ &\quad 0.7 * 1.0 * 0.18 * \\ &\quad 1.0 * 1.0 * 0.18 \\ &= 0.0006804 \end{aligned}$$

# WiegutsindBaumbank -Grammatiken?

- Standardverfahren:
  - Trainingskorpus (Regelextraktion)
  - Testkorpus (Regelanwendung)
- **Abdeckung:** Kann die Grammatik alles parsen?
  - Wenn Regeln fehlen, können Testsätze nicht geparkt werden
  - **Ergebnis: kein Problem (Übergenerierung)**
- **Korrektheit:** Ist der wahrscheinlichste Baum der gelernten Grammatik (GB) auch der Baum in der Baumbank (BB)?
  - Precision: Anteil der Regeln im GB, die auch im BB vorkommen
  - Recall: Anteil der Regeln im BB, die auch im GB vorkommen

# Korrektheit:Ergebnisse(Englisch)

- Standard-Baumbank-Grammatik mit starker Unabhängigkeitsannahme
  - Penn Treebank

| Satzlänge | Ø Länge | Precision | Recall |
|-----------|---------|-----------|--------|
| 2-16 Test | 11.4    | 85.0      | 87.7   |
| 2-25 Test | 16.3    | 82.0      | 84.0   |
| 2-40 Test | 21.9    | 78.8      | 80.4   |

(Charniak 1993)

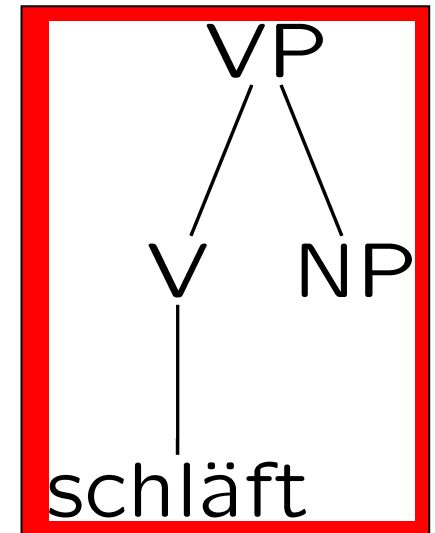
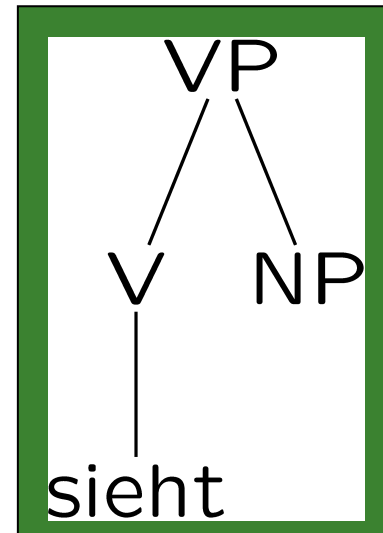
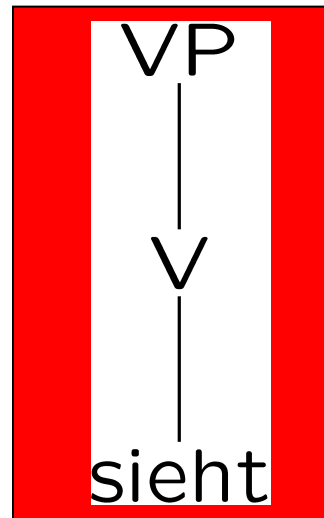
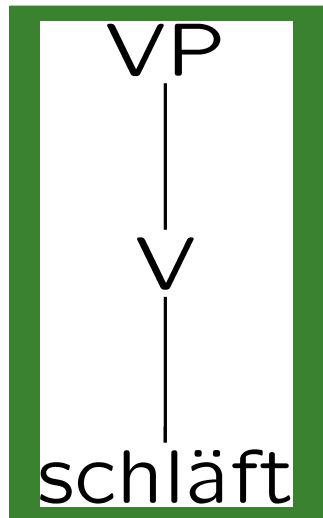
---

# Korrektheit:Ergebnisse(Deutsch)

- Standard-Baumbank-Grammatik mit starker Unabhängigkeitsannahme
  - NEGRA: 72.99 Recall, 70.00 Precision
- Resultate:
  - Lange Sätze sind schwieriger
    - „Small tree bias“
  - Deutsch ist schwieriger
    - Verschiedene Annotationsschemata
      - Deutsch: Parsebäume haben weniger Knoten

# Problem der Unabhängigkeitsannahme

- Erinnerung: Annahme ist, daß alle Regeln voneinander **unabhängig** sind
- Gegenbeispiel: VP
  - Bei transitivem Verb soll  $VP \rightarrow V \ NP$  wahrscheinlicher sein
  - Bei intransitivem Verb soll  $VP \rightarrow V$  wahrscheinlicher sein





# Lexikalisierung

## ■ Lösung: Lexikalisierung

### □ P(Regel) auf **lexikalischen Kopf** konditionieren

■  $P(V \ NP \mid VP, \text{schlafen}) = 0.1$

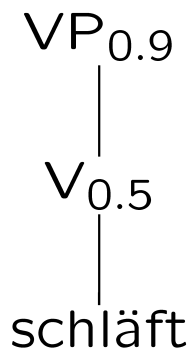
■  $P(V \mid VP, \text{schlafen}) = 0.9$

■  $P(V \ NP \mid VP, \text{sehen}) = 0.99$

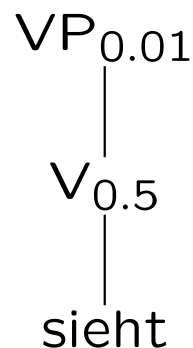
■  $P(V \mid VP, \text{sehen}) = 0.01$

$$P(\text{schlafen} \mid V) = 0.5$$

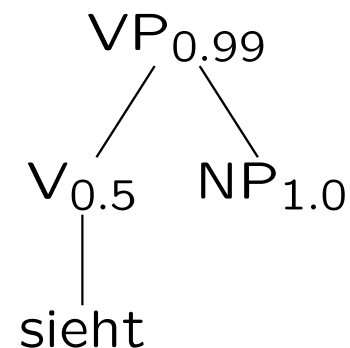
$$P(\text{sehen} \mid V) = 0.5$$



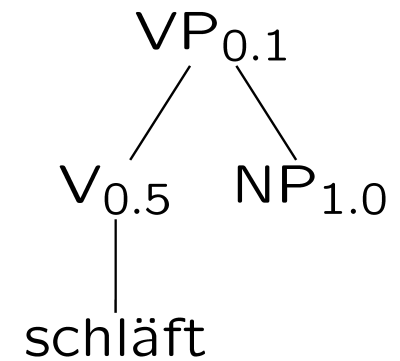
$$P(t) = 0.45$$



$$P(t) = 0.005$$



$$P(t) = 0.495$$



$$P(t) = 0.05$$

# Mehr zur Lexikalisierung

- Lexikalischer Kopf: Verb (VP,S), Nomen (NP), Adjektiv(AP),...
- Berechnung aus Frequenzen:
  - Kopf wird neues konditionierendes Ereignis (Feature)

$$P(RS|LS, K) = \frac{f(LS \rightarrow RS, K)}{f(LS, K)} = \frac{f(LS \rightarrow RS, K)}{\sum_{RS} f(LS \rightarrow RS, K)}$$

- Lexikalisierung hilft deutlich, den besten Parsebaum zu finden (Korrektheit)
  - Aber: Abdeckung wird ein Problem (Sparse Data)
    - Wie behandelt man ungesehene Verben?
    - Typisch: Backoff zu unlexikalisiertem Modell
  - Aber: Noch **sehr viel mehr** Regeln
    - Eine Regel pro Kombination aus unlexikalisierte Regel und Kopfwort
    - Suchraum wird immens groß

# WeitereProbleme

- Laut Unabhängigkeitsannahme sind Subjekt-NPs und Objekt-NPs gleich
  - Regeln mit linker Seite NP „kennen“ ihre Position nicht

Wird von den Daten (Englisch) nicht bestätigt!

| Regel        | % als Subjekt | % als Objekt |
|--------------|---------------|--------------|
| NP → PRP     | 13.7%         | 2.1%         |
| NP → NP PP   | 5.6%          | 14.1%        |
| NP → NP SBAR | 0.5%          | 2.6%         |
| NP → DT NN   | 5.6%          | 4.6%         |

(Manning and Schütze 1999)

# Geschichte( History)

- Lösung: Knoten kennen ihre „Geschichte“
  - $P(\text{Regel})$  auf „Großvater“ konditionieren (S oder VP)

| Regel                                  | % als Subjekt | % als Objekt |   |       |
|--|---------------|--------------|---|-------|
| $\text{NP} \rightarrow \text{PRP}$     | 13.7%         | 2.1%         | $P(\text{PRP} \mid \text{NP}, \text{S}) =$      | 0.137 |
|  |               |              | $P(\text{PRP} \mid \text{NP}, \text{VP}) =$     | 0.021 |
| $\text{NP} \rightarrow \text{NP PP}$   | 5.6%          | 14.1%        | $P(\text{NP PP} \mid \text{NP}, \text{S}) =$    | 0.056 |
|  |               |              | $P(\text{NP PP} \mid \text{NP}, \text{VP}) =$   | 0.141 |
| $\text{NP} \rightarrow \text{NP SBAR}$ | 0.5%          | 2.6%         | $P(\text{NP SBAR} \mid \text{NP}, \text{S}) =$  | 0.005 |
|  |               |              | $P(\text{NP SBAR} \mid \text{NP}, \text{VP}) =$ | 0.026 |
| $\text{NP} \rightarrow \text{DET NN}$  | 5.6%          | 4.6%         | $P(\text{DET NN} \mid \text{NP}, \text{S}) =$   | 0.056 |
|  |               |              | $P(\text{DET NN} \mid \text{NP}, \text{VP}) =$  | 0.046 |

# MehrüberGeschichte

- Mehr Regeln
  - $P(RS \mid LS, GV)$ : Eine Regel pro Kombination aus unlexikalisierte Regel und Großvaterkategorie
  - Nicht so schlimm wie bei Lexikalisierung
    - Beispiel NP: Regelzahl verdoppelt sich (NP in VP und NP in S)
    - Sparse data-Problem nicht ganz so dramatisch
- Hilft auch bei anderen Unterscheidungen
  - Direktes und indirektes Objekt
  - Hauptsatz und Nebensatz
- Schwierig bei flachen Annotationsschemata
  - Wenn es keinen VP-Knoten gibt...?
    - Mögliche Lösung: Schwesterknoten anschauen

# EinModellausderPraxis

- Benutzt Geschichte und Lexikalisierung
  - Großvaterknoten l
  - Kopf h
  - Modell mit allen Features ist am genauesten, aber keine gute Abdeckung
  - Einfachere Modelle sind dümmer, aber robuster
    - Die Koeffizienten  $\lambda_i$  werden durch Optimierung festgelegt

$$P(RS|LS, h, l) = \lambda_1 p(RS|LS, h, l) + \lambda_2 p(RS|LS, h) + \lambda_3 p(RS|LS, l) + \lambda_4 p(RS|LS)$$

(Charniak 1996)

---

# Ergebnisse: State of the art (Modelle mit Lexikalisierung, Geschichte und Tricks)

- Englisch: Recall 90%, Precision 90%
  - Penn Treebank
  - Vergleich: naives Modell Recall 79%, Precision 80%
- Deutsch: Recall 81%, Precision 77%
  - NEGRA
  - Vergleich: naives Modell Recall 73%, Precision 70%
- Erinnerung: Unterschiedliche Annotationsschemata
  - Unterschiedliche Knotenzahlen
  - Unterschiedliche Wahrscheinlichkeitsmodelle

Oder doch Problem mit kontextfreien Grammatiken?

---

---

# Schritte für Parsing mit PCFGs

1. Lerne Regeln mit Wahrscheinlichkeiten
2. Berechne Wahrscheinlichkeiten für Bäume aus Regelwahrscheinlichkeiten
3. Finde effizientes Verfahren, um wahrscheinlichste Bäume zu konstruieren



---

# Herkömmliches Parsing

- Top-Down: Fange an mit S, und wende **alle anwendbaren** Regeln an
  - Beispiel: Earley-Algorithmus
- Bottom-Up: Fange an mit Worten, kombiniere mit **allen anwendbaren** Regeln

## Erschöpfendes (Exhaustives) Parsing

---

# Parsing mit Baumbank - Grammatiken

- Viele Regeln

- Exhaustives Parsing nicht beherrschbar für Sätze mit mehr als 10 Wörtern

- Hoffnung: Zipf-Verteilung für Parsebäume

- Es gibt wenig „vernünftige“ Parsebäume
  - „Vernünftige“ Parsebäume sind viel wahrscheinlicher als (fast) alle anderen

Aufgabe: indentifiziere (konstruiere) die kleine Anzahl sehr wahrscheinlicher Parsebäume

# Folgedemwahrscheinlichen Parsebaum?

- Beginne irgendwo
- Finde den lokal wahrscheinlichsten Parsebaum
- Finde wahrscheinlichste Erweiterung
- Steht am Ende der global wahrscheinlichste Parsebaum?
  - Nein!
  - Holzweg- (Garden path) Sätze:
    - [Peter, der viel redet,] mag ich nicht.
    - [The horse [raced past the barn]] fell.

# Behandlung von lokalen Ambiguitäten

- Deterministische Parsingstrategie schlägt i.A. fehl
  - Wegen lokaler Ambiguitäten muß der lokal beste Baum nicht Teil des global besten Baumes sein
  - Mögliche Folgen:
    - Resultierender Baum ist nicht der wahrscheinlichste
    - Parsing schlägt komplett fehl („harter Holzwegsatz“)
- Durch geschickte Formulierung der Grammatik ist deterministisches Parsing manchmal möglich
  - In diesem Fall liefert Viterbi-Algorithmus garantiert besten Baum
- Oft nicht möglich
  - Näherungslösung: **Beam Search (Strahlsuche)**

---

# Beam Search

- Liste von halbfertigen Parsebäumen (Hypothesen mit Wahrscheinlichkeiten)
- Jede Hypothese um jede anwendbare Regel erweitern
  - Wahrscheinlichkeiten für erweiterte Hypothesen ausrechnen
- Die  $n$  besten Hypothesen behalten, die anderen vergessen
- Wiederholen, bis  $n$  Hypothesen für ganzen Satz

---

# Beam Search

- $n = 1$ : naive „Bestensuche“
- $n = \infty$ : exhaustives Parsing
- Gefahr: für kleines  $n$  kann der global beste Parsebaum „aus dem Strahl fallen“
  - Wie groß muß  $n$  sein?

# Beam Search beim Parsen

(Ratnaparkhi 1999)

| n  | Zeit (s) | Precision | Recall |
|----|----------|-----------|--------|
| 20 | 2.07     | 87.9      | 87.1   |
| 15 | 1.58     | 87.7      | 86.9   |
| 10 | 1.07     | 87.7      | 86.9   |
| 7  | 0.76     | 87.4      | 86.6   |
| 5  | 0.56     | 87.3      | 86.8   |
| 2  | 0.35     | 85.1      | 86.1   |
| 1  | 0.14     | 82.4      | 83.4   |

---

# Beam Search in der Praxis

- Optimales  $n$  muß empirisch ermittelt werden
- Funktioniert ziemlich gut
  - Wegen Zipf-Verteilung ist global bester Parsebaum ist fast immer in der Spitzengruppe
- Beam Search nur Methode, um Suchraum einzuschränken
  - Muß mit einer Suchstrategie (bottom-up, top-down, etc.) kombiniert werden



---

# Zusammenfassung

- Kontextfreie Grammatiken können aus Baumbanken gelernt werden
  - Viele Regeln
  - Kein Problem mit Abdeckung
  - Hohe Ambiguität
- Probabilistische kontextfreie Grammatiken
  - Möglichkeit, über „besten Baum“ zu sprechen
  - Regelwahrscheinlichkeiten aus Baumbank ablesen

# Zusammenfassung

- Was ist die Wahrscheinlichkeit eines Baumes?
  - Naiv: Starke Unabhängigkeitsannahme
    - Jede Regel ist unabhängig
  - Lexikalisierung
    - $P(\text{VP} \rightarrow \text{V NP} | \text{Trans. V}) > P(\text{VP} \rightarrow \text{V NP} | \text{Intrans. V})$
  - Geschichte
    - $P(\text{NP} \rightarrow \text{PRP} | \text{S}) > P(\text{NP} \rightarrow \text{PRP} | \text{VP})$
- Wie finde ich den besten Baum?
  - Beam search