

# Symbolische und statistische Verfahren (CL2)

## 2: Tokenisierung und Satzgrenzenerkennung

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum



# Themenüberblick heute

## Tokenisierung

- 1 Segmentierung in Wörter**
  - 2 Segmentierung in Sätze**
- Konzept “orthographisches Wort”
  - Ambiguität der Interpunktions
  - Satzgrenzenerkennung
  - Strategien
  - Anhang: Spezielle Probleme

# Tokenisierung

- Tokenisierung: Zerlegung eines Textes in einzelne **relevante Segmente** = **Tokens**
- Grundlegender Vorverarbeitungsschritt (Preprocessing) vor der Anwendung komplexerer computerlinguistischer Verfahren
  - wie z.B. Tagging, Parsing, Übersetzung etc.
- Was sind die relevanten Segmente?
  - hängt von der intendierten Anwendung ab ...
  - in der Regel relevant: **Wörter** und **Sätze**
  - dann: Tokens = Wörter

# Tokens = Wörter

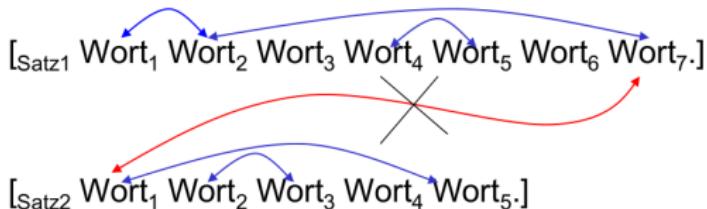
Inwiefern ist das **Wort** eine relevante Einheit für computerlinguistische Verfahren?

- Mehrzahl der Grammatiktheorie: atomare Einheit innerhalb einer syntaktischen Struktur ist das Wort
  - d.h. Parser analysieren in der Regel eine Kette von Wörtern
- Information-Retrieval: Nutzer suchen nach Wörtern
- Wörterbücher: Einträge meist für einzelne Wörter (Lexeme)
- Informationsextraktion: interessante Daten über Personen, Firmen, Daten usw.: oft Suche nach Folgen von Wörtern bestimmter Kategorien

# Sätze/Satzgrenzenerkennung

Inwiefern ist der **Satz** eine relevante Einheit für computerlinguistische Verfahren?

- Sprachliche Kontextabhängigkeit innerhalb und außerhalb eines Satzes unterscheiden sich
  - innerhalb: hauptsächlich syntaktische (und semantische) Kontextabhängigkeit
  - außerhalb: keine syntaktische Kontextabhängigkeit
  - syntaktische Phänomene und Abhängigkeiten sind auf einen (komplexen) Satz beschränkt



# Sätze/Satzgrenzenerkennung

- Formale linguistische Grammatiken: formale Modelle für die unendliche Menge der möglichen Sätze einer natürlichen Sprache
  - d.h. syntaktische Analyse (Parsing) findet traditionell auf der Ebene des Satzes statt (ebenso Übersetzung)
  - ebenso: lokalere und flachere Analysen wie Wortarten-Tagging oder Chunking (= Shallow Parsing)
    - Chunking: flache syntaktische Analyse in Form einzelner **Chunks**
    - Chunks: einfache Phrasen (NC = Noun Chunk, PC, ...) ohne Rekursion

# Sätze/Satzgrenzenerkennung

- Satz: sprachliche Einheit, die oft gerade noch handlich genug für komplexe Analyseschritte ist
- Häufigkeiten verschiedener Satzlängen in einem Zeitungskorpus

Length	Number	%	Cum. %
1–5	1317	3.13	3.13
6–10	3215	7.64	10.77
11–15	5906	14.03	24.80
16–20	7206	17.12	41.92
21–25	7350	17.46	59.38
26–30	6281	14.92	74.30
31–35	4740	11.26	85.56
36–40	2826	6.71	92.26
41–45	1606	3.82	96.10
46–50	858	2.04	98.14
51–100	780	1.85	99.99
101+	6	0.01	100.00

Tabelle 4.3 aus Manning and Schütze (1999, S. 137)

# Gliederung

## 1 Tokenisierung

- Ambiguität
- Satzgrenzenerkennung

## 2 Disambiguierung ambiger Interpunktionszeichen

- Maschinenlernverfahren

# Beispielkorpus

## Thomas Mann: Der Tod in Venedig

*Ebensweit entfernt vom Banalen wie vom Exzentrischen, war sein Talent geschaffen, den Glauben des breiten Publikums und die bewundernde, fordernde Teilnahme der Wählerischen zugleich zu gewinnen. So, schon als Jüngling von allen Seiten auf die Leistung — und zwar die außerordentliche — verpflichtet, hatte er niemals den Müßiggang, niemals die Fahrlässigkeit der Jugend gekannt. Als er um sein fünfunddreißigstes Jahr in Wien erkrankte, äußerte ein feiner Beobachter über ihn in Gesellschaft: »Sehen Sie, Aschenbach hat von jeher nur so gelebt«—und der Sprecher schloß die Finger seiner Linken fest zur Faust—; »niemals so«—und er ließ die geöffnete Hand bequem von der Lehne des Sessels hängen. Das traf zu; und das Tapfer-Sittliche daran war, daß seine Natur von nichts weniger als robuster Verfassung und zur ständigen Anspannung nur berufen, nicht eigentlich geboren war.*

Quelle: <http://www.gutenberg.org/files/12108/12108-8.txt>

# Tokenisierung

- Als welche Art von Datenstruktur liegt ein solches Textkorpus im “Rohzustand” vor?
  - als Zeichenkette
  - alle Zeichen (inkl. Leerzeichen, Steuerzeichen usw.): kodiert durch einen eindeutigen Zahlenwert
  - es gibt keine fertige Tokenisierung in einzelne Wörter!
- Der Anfang des Beispiel-Korpus in Hexadezimal-Darstellung:
  - hexadezimal: ein Zeichen kodiert eine Folge von 4 Bits
  - Bsp: 41 (hexadez.) = 0100.0001 (binär)

45 62 65 6e 73 6f 77 65 69 74 20 65 6e 74 66 65  
72 6e 74 20 76 6f 6d 20 42 61 6e 61 6c 65 6e 20

Ebensweit entfernt vom Banalen

# Tokenisierung

45 62 65 6e 73 6f 77 65 69 74 20 65 6e 74 66 65  
72 6e 74 20 76 6f 6d 20 42 61 6e 61 6c 65 6e 20

- Was ist ein Wort und was soll daher als ein Token behandelt werden?
- **Orthographische Definition** eines Wortes: *a string of contiguous alphanumeric characters with space on either side* (Kučera and Francis 1967)

# Probleme mit dem “orthographischen Wort”: Tokenisierung an Whitespace

*Als er um sein fünfunddreißigstes Jahr in Wien erkrankte, äußerte ein feiner Beobachter über ihn in Gesellschaft: »Sehen Sie, Aschenbach hat von jeher nur so gelebt«—und der Sprecher schloß die Finger seiner Linken fest zur Faust—; »niemals so«—und er ließ die geöffnete Hand bequem von der Lehne des Sessels hängen.*

# Probleme mit dem orthographischen Wort

- Die eigentlichen Wörter: nicht durch Whitespace von Satzzeichen etc. getrennt
  - *erkrankte,; gelebt<–und; Faust–; hängen.*
- Vorschlag: einfach alle Satzzeichen löschen
  - aber: Satzzeichen vermitteln wichtige Informationen
  - z.B. über Satzgrenzen, wörtliche Rede, Syntax
- Vorschlag: an jedem Satzzeichen eine Tokengrenze annehmen
  - aber: Interpunktionszeichen sind teilweise **ambig**
  - ihre Bedeutung/Funktion ist kontextabhängig
  - einige Satzzeichen treten auch als Bestandteil von Token auf

# Gliederung

## 1 Tokenisierung

- Ambiguität
- Satzgrenzenerkennung

## 2 Disambiguierung ambiger Interpunktionszeichen

- Maschinenlernverfahren

# Ambiguität der Interpunktionszeichen

Welche ambigen Satzzeichen gibt es?



- Viele können ambig sein!
- Hier: Fokus auf den Punkt (“.”)
- Im Anhang: ausführlichere Besprechung weiterer Satzzeichen

# Ambiguität der Interpunktionszeichen: Punkt

- Satzpunkt (d.h. Punkt am Satzende)
- Dezimalpunkt im Englischen
  - *\$15.40*
- Gliederung großer Zahlen im Deutschen
  - *82.459.178*
- Abkürzungspunkt
  - *Auch der Kaffee, bzw. auch z.B. Latte Macchiato*
- Teil von URLs usw.
  - *www.linguistics.rub.de*
- Mehrere Punkte als Ellipsenzeichen (Auslassungszeichen)
  - *“Es ist nicht nötig”, sagte er, “das heißt, ich meine, es ist vielleicht besser, wenn Sie...”* (Rilke, Die Aufzeichnungen des Malte Laurids Brigge)

# Ambiguität der Interpunktionszeichen

Kurzer Blick auf Komma, Doppelpunkt, Bindestrich  
(mehr dazu s. Anhang)

## ■ Komma:

- Satzzeichen, Dezimalzeichen
- als Satzzeichen: Markierung von Aufzählungen, Nebensätzen, Appositionen, ...

## ■ Doppelpunkt:

- Satzzeichen, Gliederungszeichen (z.B. Uhrzeit, Fußball-Ergebnis)

## ■ Bindestrich:

- Silbentrennung, Komposita u.a.

# Gliederung

## 1 Tokenisierung

- Ambiguität
- Satzgrenzenerkennung

## 2 Disambiguierung ambiger Interpunktionszeichen

- Maschinenlernverfahren

# Satzgrenzenerkennung

Woran erkennt man Satzgrenzen (in deutschen Texten)?

- Markierung durch spezielle Interpunktionszeichen . ! ? ( ; : )
- ... aber oben schon gesehen: Probleme aufgrund der Ambiguität des Punktes
  - 1 Punkt als Satzgrenzenmarkierer
    - *Das ist ein Satz. Das ist noch einer.*
  - 2 Punkt als Teil eines anderen Tokens
    - *Dies hier ist z.B. auch ein Satz.* (Abkürzungen und Initialen)
    - *Es war am 3. März.* (Ordinalzahlen)
  - 3 ... beides gleichzeitig
    - *Das größte Problem sind Viren, Trojaner, Würmer, usw. Das zweitgrößte Problem sind Spam-E-Mails.*
    - *Hier geht es nicht weiter ...*
    - Achtung: *usw.*: Abkürzungspunkt und Satzendepunkt gleichzeitig (ein Punkt wird haplogatisch eingespart)!

# Wie groß ist das Ambiguitätsproblem?

Table 7  
Statistical Properties of the Test Corpora

Corpus	Tokens	Tokens with Final Periods	Abbr. Tokens	Abbr. Tokens %	Abbr. Types
B. Portuguese	321,032	15,250	481	3.15 %	102
Dutch	340,238	20,075	1,270	6.33 %	141
English – WSJ	469,396	26,980	7,297	27.05 %	196
English – Brown	1,105,348	54,722	5,586	10.21 %	213
English – Poe	324,247	11,247	600	5.33 %	59
Estonian	358,894	25,825	2,517	9.75 %	248
French	369,506	12,890	375	2.91 %	91
German	847,207	38,062	3,603	9.47 %	139
Italian	312,398	11,561	442	3.82 %	156
Norwegian	479,225	28,368	1,882	6.63 %	242
Spanish	352,773	13,015	570	4.38 %	84
Swedish	338,948	19,724	769	3.90 %	100
Turkish	333,451	21,047	598	2.84 %	103

Tabelle aus Kiss and Strunk (2006); vgl. z.B.

- Türkisch: 2,84% aller Punkte sind Abkürzungspunkte
- Wall Street Journal: 27,05%

# Tokenisierung und Satzgrenzenerkennung

- Erkenntnis: Tokenisierung und Satzgrenzenerkennung haben mit den gleichen Problemen zu kämpfen:  
Ambiguität der Satzzeichen
- Daher: Lösungen für beide Aufgaben überschneiden sich
- ... und wir schauen uns beide zusammen an (s. unten)
- Zunächst aber noch einige Aspekte zu Tokenisierung

# Übliches Vorgehen bei der Tokenisierung

## 1 Definition verschiedener **Zeichenklassen** (als RegEx)

### a. **Whitespace**

- Leerzeichen, Tabulator, Zeilenumbruch

### b. **Alphanumerische Zeichen** (Buchstaben und Ziffern)

- A-ZÄÖÜä-zäöüß, 0-9

### c. **Eindeutige Interpunktionszeichen**, die (quasi) *immer* ein eigenständiges Token bilden, d.h. nicht innerhalb von Wörtern/Zahlen vorkommen

- Klammern: **( ) [ ]**

(kommen allerdings in Emoticons vor, z.B.: :-))

- **! ? ;** (kommen selten in Markennamen vor, z.B.: **Yahoo!**)

- Anführungszeichen: „ “ » « “ ”

### d. **Ambige Interpunktionszeichen**, die *auch* als Teil von Token vorkommen → extra Behandlung (s. unten)

- , . : -

## Übliches Vorgehen bei der Tokenisierung (Forts.)

- 2 Evtl. Suche nach bestimmten **Tokenklassen**, z.B. URLs oder Emoticons, und Ausschluss von der weiteren Tokenisierung
- 3 Trennung der Zeichenkette an **klaren Tokengrenzen**, d.h. an Whitespace und an eindeutigen Interpunktionszeichen
- 4 Trennung an **ambigen Interpunktionszeichen** unter *bestimmten Bedingungen* (mehr dazu unten)

# Einschub: mögliche Datenstrukturen für tokenisierten Text

- **Interne Repräsentation** (d.h. innerhalb des Programms):
  - eine Liste (Array) von einzelnen Strings (den Tokens)
- **Externe Repräsentation:**
  - die einzelnen Token werden durch ein Leerzeichen voneinander getrennt; in jeder Zeile steht ein Satz (“**one sentence per line**”, \*.1spl, \*.spl)
  - jedes Token steht in einer neuen Zeile; zwischen zwei Sätzen wird eine Leerzeile eingefügt (“**one token/word per line**”, \*.1wpl, \*.wpl)
  - oder: Verwendung einer Markup-Sprache wie XML

Diese Formate stellen das Inputformat für viele weitere Analysetools (Tagger, Chunker, ...) dar

# Übliches Ausgabeformat für tokenisierten Text I

Ebensweit entfernt vom Banalen wie vom Exzentrischen , war sein Talent geschaffen , den Glauben des breiten Publikums und die bewundernde , fordernde Teilnahme der Wählerischen zugleich zu gewinnen .

So , schon als Jüngling von allen Seiten auf die Leistung – und zwar die außerordentliche – verpflichtet , hatte er niemals den Müßiggang , niemals die Fahrlässigkeit der Jugend gekannt .

Als er um sein fünfunddreißigstes Jahr in Wien erkrankte , äußerte ein feiner Beobachter über ihn in Gesellschaft : » Sehen Sie , Aschenbach hat von jeher nur so gelebt « – und der Sprecher schloß die Finger seiner Linken fest zur Faust – ; » niemals so « –und er ließ die geöffnete Hand bequem von der Lehne des Sessels hängen .

Das traf zu ; und das Tapfer-Sittliche daran war , daß seine Natur von nichts weniger als robuster Verfassung und zur ständigen Anspannung nur berufen , nicht eigentlich geboren war .

## Übliches Ausgabeformat für tokenisierten Text II

Ebensweit  
entfernt  
vom  
Banalen  
wie  
vom  
Exzentrischen  
,,  
war  
sein  
Talent  
geschaffen  
,,  
den  
Glauben  
des  
breiten  
Publikums

und  
die  
bewundernde  
,,  
fordernde  
Teilnahme  
der  
Wählerischen  
zugleich  
zu  
gewinnen  
. .  
So  
,,  
schon  
als  
Jüngling

← Leerzeile  
nach Satzgrenze

## Übliches Ausgabeformat für tokenisierten Text III

```
<sent>
  <tok>Ebensweit</tok>
  <tok>entfernt</tok>
  <tok>vom</tok>
  <tok>Banalen</tok>
  <tok>wie</tok>
  <tok>vom</tok>
  <tok>Exzentrischen</tok>
  <tok>, </tok>
  <tok>war</tok>
  <tok>sein</tok>
  <tok>Talent</tok>
  <tok>geschaffen</tok>
  <tok>, </tok>
  <tok>den</tok>
  <tok>Glauben</tok>
  <tok>des</tok>
  <tok>breiten</tok>
```

```
<tok>Publikums</tok>
<tok>und</tok>
<tok>die</tok>
<tok>bewundernde</tok>
<tok>, </tok>
<tok>fordernde</tok>
<tok>Teilnahme</tok>
<tok>der</tok>
<tok>Wählerischen</tok>
<tok>zugleich</tok>
<tok>zu</tok>
<tok>gewinnen</tok>
<tok>. </tok>
</sent>
<sent>
  <tok>So</tok>
  <tok>, </tok>
  <tok>schon</tok>
```

# Gliederung

## 1 Tokenisierung

- Ambiguität
- Satzgrenzenerkennung

## 2 Disambiguierung ambiger Interpunktions

- Maschinenlernverfahren

# Ambige Interpunktionszeichen

- Ambige Tokengrenzen: , . : -
- Welche davon markieren potenziell Satzgrenzen?
  - Punkt (und evtl. Doppelpunkt)
- Funktionen des Punktes (Wdh)
  - Satzende, Zahlengliederung, Abkürzung, URL, Ellipsen
- Welche davon sind “kritisch” für unsere Aufgabe?
  - einfach zu erkennen: Zahlengliederung, URL, Ellipsen
  - schwierig: **Satzende vs. Abkürzung**
- Wie kann man Satzende vs. Abkürzung unterscheiden?



→ linker und rechter Kontext geben Hinweise!

# Entscheidungsregel für Satzgrenzen

Nimm eine Satzgrenze an:

- ... nach einem nicht-ambigen Satzzeichen ! ?
- ... nach einem Punkt am Wortende, wenn der linke und rechte Kontext auf eine Satzgrenze hindeuten:
  - linker Kontext ist keine Abkürzung
  - rechter Kontext sieht nach Satzanfang aus

Token<sub>-3</sub> Token<sub>-2</sub> Token<sub>-1</sub> • Token<sub>+1</sub> Token<sub>+2</sub> Token<sub>+3</sub>



# Rechter und linker Kontext

Im Folgenden

- Zunächst rechter Kontext: Satzanfang?
- Dann linker Kontext: Abkürzung?

# Woran erkennt man einen Satzanfang?

## Rechter Kontext

- Orthographisches Kriterium: wenn dem Punkt ein großgeschriebenes Wort folgt → Satzgrenze
  - *Das ist ein Satz. Das ist noch einer.* → Satzgrenze
  - *d.h. integrationspolitisch* → keine Satzgrenze
- Aber: in vielen Sprachen gibt es bestimmte Wörter, die auch innerhalb eines Satzes großgeschrieben werden
  - Alle Nomen und das Pronomen *Sie* im Deutschen
    - *12 Mio. Fr. Schulden*
  - Eigennamen und das Pronomen */* im Englischen
    - *Mr. Smith and Mrs. Smith*
- → Rechter Kontext muss disambiguiert werden:  
**Nur wenn dem Punkt ein großgeschriebenes Wort folgt, das innerhalb eines Satzes *normalerweise kleingeschrieben* wird, kann man sicher eine Satzgrenze annehmen**

# Probleme mit dem orth. Kriterium

## Rechter Kontext

Woher weiß man, welche Wörter normalerweise kleingeschrieben werden?

- Rückgriff auf ein fertiges Lexikon
  - Nachteil: unbekannte Wörter im Korpus
- Aufbau eines Lexikons aus dem zu tokenisierenden Korpus selbst (sog. "Korpusfiltermethode")
  - Nachteil: Viele Wörter kommen nur einmal vor (**Hapax Legomena**)
  - Für diese Wörter kann man dann nicht sagen, wie sie sich *normalerweise* innerhalb eines Satzes verhalten

# Probleme mit dem orth. Kriterium

## Rechter Kontext

- Problem: Nicht alle Token im rechten Kontext enthalten Informationen bezüglich Groß- und Kleinschreibung
  - *in Art. 22 Abs. 3*
  - *Ledergerber (sp.) und*
- Daher Formulierung als Ausschlusskriterium:  
**Wenn das Token<sub>+1</sub> hinter einem Punkt eindeutig kleingeschrieben ist, dann markiert der Punkt keine Satzgrenze (und Token<sub>-1</sub> ist eine Abkürzung)**
  - wenn keine eindeutige Kleinschreibung: weitere Heuristiken anwenden

# Abkürzungserkennung

## Linker Kontext

- Viele Abkürzungen enthalten einen finalen Punkt als festen Bestandteil
- Der Punkt ist Teil des Tokens und sollte beim Tokenisieren nicht abgetrennt werden

# Abkürzungserkennung

## Linker Kontext

- Für die Satzgrenzenerkennung relevante Abkürzungsarten
  - Abkürzungen: *amerikan.*, *bzw.*, *Dr.*, *Genf-Str.*, *inkl.*, *vgl.*
  - Initialen: *Frank A. Meyers*
  - Ordinalzahlen: *16. Semester*, *Papst Johannes Paul II.*
- Für die Satzgrenzenerkennung irrelevante Arten von Abkürzungen (ohne abschließenden Punkt)
  - Akronyme (Initialwörter): *ARD*, *NATO*, *NZZ*, *SPD*, *UvD*
  - Kurzwörter: *Ersti*, *Kripo*
  - Abkürzungen ohne Punkt: *BAföG*, *BArtSchV*, *GmbH*, *km*

# Erkennungsmerkmale von Abkürzungen

## Linker Kontext

- Feste Eigenschaften von Abkürzungstypen
  - in der Regel kurz
  - immer ein abschließender Punkt
  - können intern weitere Punkte enthalten
    - *d.h., u.a., F.D.P., a.D., z.B.*
  - enthalten oft für normale Wörter untypische Buchstabenfolgen
    - *Nkm., Mrd., bzw., ldu., usf.*
  - insbesondere: oft kein Vokal!

# Erkennungsmerkmale von Abkürzungen

## Linker Kontext

- Kontext einzelner Abkürzungstoken
  - auch innerhalb eines Satzes (z.B. vor einem kleingeschriebenen Wort) mit abschließendem Punkt
    - *z.B. vor einem kleingeschriebenen Wort*
  - bilden manchmal Kollokationen (feste Verbindungen) mit folgenden Wörtern
    - *28. Oktober, 19. Jahrhunderts*
    - *James D. Watson, John F. Kennedy*

# 1. Token-basierte Klassifikation

## Linker Kontext

### ■ Token-basiert:

- für jedes Token mit abschließendem Punkt wird einzeln entschieden, ob es sich um eine Abkürzung handelt oder nicht
  - Evidenz kommt nur aus dem jeweiligen lokalen Kontext
    - z.B. Einsatz von RegEx für Abkürzungen und Satzgrenzen
  - System baut keine abstrakte Repräsentation auf
  - nur ein Durchgang durch das Korpus nötig
- Meist: mit Einsatz von Abkürzungslisten

## 2. Typen-basierte Klassifikation

### Linker Kontext

#### ■ Typen-basiert:

- es wird für jeden Worttyp entschieden, ob es sich um eine Abkürzung handelt oder nicht
- Evidenz wird aus allen Kontexten, in denen dieser Typ vorkommt, aggregiert
- die Klassifikation wird dann auf alle einzelnen Instanzen (Token) des Typs angewandt
- System baut eine abstrakte Repräsentation auf (z.B. Lexika mit Abkürzungs- und Nichtabkürzungstypen)
- Es sind in der Regel zwei Durchgänge durch das Korpus nötig:

- 1 Datensammlung
- 2 Klassifikation

# Spezialprobleme

- Im Anhang einige weitere Folien zu Spezialproblemen
  - bei der Satzgrenzenerkennung
  - bei der Tokenisierung

# Gliederung

## 1 Tokenisierung

- Ambiguität
- Satzgrenzenerkennung

## 2 Disambiguierung ambiger Interpunktions

- Maschinenlernverfahren

# Maschinenlernverfahren

- Problem vieler Ansätze: involvieren sprachabhängige Information
  - z.B. Abkürzungslisten, reguläre Ausdrücke
- Allgemeinere Lösung: Maschinenlernverfahren
  - engl.: Machine Learning
- **Überwachte** (engl. **supervised**) Maschinenlernverfahren: lernen Klassifikationsregeln oder für die Klassifikation relevante Merkmale aus **annotierten** Trainingsdaten, d.h. sie sind auf korrekt klassifizierte Beispiele angewiesen
- **Unüberwachte** (engl. **unsupervised**) Verfahren: lernen Klassifikationsregeln aus **unannotierten** Daten, d.h. sie sind nicht auf korrekt klassifizierte Trainingsbeispiele angewiesen
  - Beispiel: Überwachte Satzgrenzenerkennung  
→ Trainingskorpus mit handannotierten Satzgrenzen

# Überwachte Satzgrenzenerkennung: MxTerminator

Reynar und Ratnaparkhi 1997

- Trainingsdaten: in Sätze eingeteiltes Korpus
- "Kandidaten": Wortformen der Form "Präfix.Suffix" (Suffix kann leer sein)
- Aufgabe: Klassifikation: gegeben ein Kandidat, berechne die Wahrscheinlichkeit dafür, dass der Punkt innerhalb des Kandidaten eine Satzgrenze ist, gegeben den lokalen Kontext
- Gelernte Merkmale/Parameter:
  - Präfix und Suffix
  - sind Präfix und Suffix bekannte Abkürzungen?
  - Worttypen an den Positionen  $\text{Kandidat}_{-1}$ ,  $\text{Kandidat}_{+1}$
  - ist  $\text{Kandidat}_{-1}$  oder  $\text{Kandidat}_{+1}$  eine bekannte Abkürzung?
- Zahl benötigter Trainingsbeispiele: mehrere tausend

# Überwachte Satzgrenzenerkennung: SATZ

(Palmer und Hearst 1997)

- Trainingsdaten: in Sätze eingeteiltes Korpus, Tag-Lexikon (Worttypen und ihre Wortarten), Abkürzungsliste
- Gelernte Merkmale:
  - mögliche Tags (Wortarten) der drei Token vor und nach der potentiellen Satzgrenze

<i>at</i>	<i>the</i>	<i>plant</i>	.	<i>He</i>	<i>had</i>	<i>thought</i>
PREP(1.0)	ART(1.0)	N(0.8)/V(0.2)	.	PRON(1.0)	V(1.0)	N(0.1)/V(0.9)

- ist Token<sub>-1</sub> eine bekannte Abkürzung?
- ist Token<sub>+1</sub> groß- oder kleingeschrieben?
- Klassifiziert potentielle Satzgrenzen an Hand des lokalen syntaktischen Kontexts (Wortartentags)
- Zahl benötigter Trainingsbeispiele: mehrere hundert

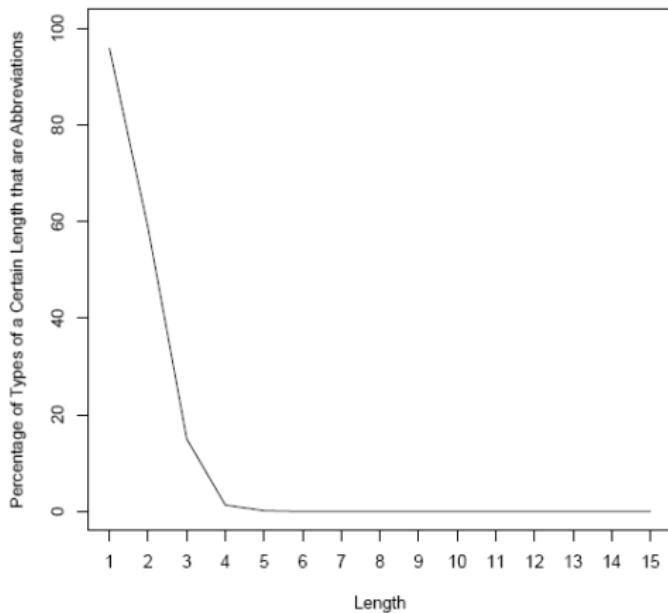
# Ein hybrides System: Punkt

(Kiss and Strunk 2006)

- Erste Stufe: **typen-basierte statistische Abkürzungserkennung**
  - Abkürzungen als **Kollokation** zwischen der eigentlichen Abkürzung und dem abschließenden Punkt (s. nächste Sitzung)
  - Länge des Kandidaten (s.u.)
  - Zahl der internen Punkte
- Alle Punkte nach Nichtabkürzungen sind Satzgrenzen
- Zweite Stufe: **token-basierte Satzgrenzenerkennung**
  - Finden von Abkürzungen und Ellipsen (...) am Satzende
  - Finden von Initialen und Ordinalzahlen
  - Benutzung orthographischer Informationen
- Orthographische Evidenz sekundär → daher auch für Texte ohne Groß- und Kleinschreibung geeignet

# Länge als probabilistisches Kriterium bei der Abkürzungserkennung

aus Kiss and Strunk (2006)



# Wie gut sind Satzgrenzenerkennner?

System	Fehlerrate (Brown Corpus)	Fehlerrate (WSJ Corpus)
Grefenstette & Tapanainen (1994)	0,93 %	---
Mikheev (2002)	0,28 %	0,45 %
MxTerminator (Reynar & Ratnaparkhi 1997)	2,50 %	2,00 %
Punkt (Kiss & Strunk 2006)	1,02 %	1,65 %
Riley (1989)	0,20 %	---
SATZ (Palmer & Hearst 1997)	---	1,00 %

# Accuracy und Fehlerrate (Error)

## ■ Fehlerrate (Error)

$$\text{error} = \frac{\text{fehlerhaft klassifizierte Einheiten}}{\text{Gesamtzahl der klassifizierten Einheiten}}$$

## ■ Genauigkeit (Accuracy)

$$\text{accuracy} = \frac{\text{korrekt klassifizierte Einheiten}}{\text{Gesamtzahl der klassifizierten Einheiten}}$$

- Meist in Prozent angegeben

# Evaluation — Baseline und Upper bound

- Vergleich verschiedener Konkurrenzsysteme (auf den gleichen oder vergleichbaren Daten)
- Einschätzung der Schwierigkeit einer Aufgabe
  - 1 **Absolute Baseline**: die denkbar einfachste Herangehensweise an das Problem
    - Beispiel Satzgrenzenerkennung:  
*Alle Punkte markieren Satzgrenzen.*  
(Fehlerrate 9,5% NZZ-Korpus)
  - 2 **Baseline**: ein simpler Referenzalgorithmus, den alle guten Systeme übertreffen müssen
    - Beispiel Satzgrenzenerkennung:  
*Alle Punkte markieren Satzgrenzen, es sei denn das folgende Token wird kleingeschrieben.*  
(Fehlerrate 6,23% NZZ-Korpus)
  - 3 **Upper bound**: die bestmögliche Lösung eines Problems
    - Wie gut können menschliche Experten das Problem lösen?
    - Inwieweit sind sich mehrere menschliche Experten einig?

# Weitere Evaluationskriterien

- Accuracy und Error sind wichtig, aber nicht alles!
- Weitere wichtige Kriterien:
  - **Aufwand** (welche manuelle Arbeit ist nötig?)
    - Zahl der Trainingsinstanzen
    - Zusätzliche Ressourcen
    - Zeitaufwand bei manueller Regelerstellung
  - **Sprachabhängigkeit** vs. Sprachunabhängigkeit
    - Funktioniert das System für alle Sprachen?
    - Kann es einfach an neue Sprachen angepasst werden?
  - **Domänenabhängigkeit** vs. Domänenunabhängigkeit
    - Funktioniert das System gleich gut für verschiedene Textsorten und Domänen? Z.B. Zeitung vs. Roman, Sportreport vs. medizinischer Aufsatz
  - **Effizienz**
    - Geschwindigkeit
    - Speicherbedarf

# Produktivität von Abkürzungen

Könnte man nicht einfach einmal eine lange Liste von Abkürzungen für verschiedene Sprachen erstellen?

- Typische Abkürzungen unterscheiden sich von Textdomäne zu Textdomäne (z.B. medizinische, juristische Abkürzungen usw.)
  - Namen für Spezies in der Biologie ([http://de.wikipedia.org/wiki/Liste\\_der\\_Abk%C3%BCrzungen\\_\(Biologen\)](http://de.wikipedia.org/wiki/Liste_der_Abk%C3%BCrzungen_(Biologen)))
  - Juristische Abkürzungen (<http://www.juristische-abkuerzungen.de>)
- Es können immer wieder neue Abkürzungen gebildet werden (Abkürzungen sind produktiv!)
  - *Seite 69f.* (beliebige Zahlen)
  - *Genf-Str.* (beliebige Straßen)
  - *Apostelgesch.* (ad-hoc-Abkürzungen aus einem Lexikon)

→ Es würde einen sehr großen Aufwand bedeuten, solche Abkürzungslisten von Hand aufzubauen und zu pflegen

# Produktivität von Abkürzungen

## Experiment aus Kiss and Strunk (2006)

- Liste aller Abkürzungen aus Muret-Sanders Deutsch-Englisch-Wörterbuch
  - 1.537 verschiedene Abkürzungstypen für Englisch
- Liste aller Abkürzungen aus dem neuesten DUDEN
  - 769 verschiedene Abkürzungstypen für Deutsch
- Fehlerrate bei der Abkürzungserkennung
  - 1 Punkt-System, keine Abk.-Liste
  - 2 Punkt-System + Abk.-Liste
  - 3 Nur Abk.-Liste

# Produktivität von Abkürzungen

## Experiment aus Kiss and Strunk (2006)

Corpus	Punkt (keine Abk.-Liste)	Punkt (mit Abk.-Liste)	Nur Abk.-Liste
WSJ	0,71 %	0,63 %	4,57 %
Brown	0,82 %	0,70 %	2,32 %
NZZ	0,26 %	0,23 %	2,48 %

# Tokenisierungsregeln

- Entscheidungen während der Tokenisierung haben Konsequenzen für alle nachfolgenden Analyseschritte
- Daher wichtig: Tokenisierungsregeln und -entscheidungen immer ausreichend dokumentieren!
- Konsistente Tokenisierung
- ... nur dann kann das tokenisierte Korpus ohne größere Überraschungen von anderen weiterverwendet werden
- Beispiel für Tokenisierungsregeln: **AMALGAM Project:**  
[http://www.scs.leeds.ac.uk/ccalas/amalgam/  
amalgtag2.html](http://www.scs.leeds.ac.uk/ccalas/amalgam/amalgtag2.html)

# Zusammenfassung I

- Tokenisierung bezeichnet die Segmentierung einer Zeichenkette (eines Textes) in relevante Einheiten (in der Regel Wörter, Interpunktionszeichen und Sätze) für die computerlinguistische Weiterverarbeitung
- Whitespace: wichtigstes Kriterium für die Definition orthographischer Wörter
- Probleme: Ambiguität, Kontextabhängigkeit, Sprach- und Domänenabhängigkeit
- Satzgrenzenerkennung: Evidenz aus dem linken und rechten Kontext
- linker Kontext: relevant v.a. die Erkennung von Abkürzungen
  - dafür gibt es token- und typenbasierte Ansätze

## Zusammenfassung II

- rechter Kontext: Groß-/Kleinschreibung
- Überwachte vs. unüberwachte Lernverfahren
  - überwachte Lernverfahren benötigen annotierte Trainingsdaten
- Evaluation von Systemen:
  - Accuracy (Genauigkeit)
  - Baseline (und evt. ein Upper bound)
  - plus weitere wichtige Kriterien (z.B. Domänenabhängigkeit, Effizienz)

## References I

- Tokenisierung: Abschnitt 4.2.2 (S. 124ff) aus Manning and Schütze (1999)
- Satzgrenzen: Abschnitt 4.2.4 (S. 134ff) aus Manning and Schütze (1999)

Kiss, T. and J. Strunk (2006).

Unsupervised multilingual sentence boundary detection.

*Computational Linguistics* 32(4), 485–525.

Kučera, H. and W. N. Francis (1967).

*Computational Analysis of Present-Day American English.*

Providence, RI: Brown University Press.

## References II

Manning, C. D. and H. Schütze (1999).  
*Foundations of Statistical Natural Language Processing.*  
Cambridge, MA: The MIT Press.

# Anhang

Einige spezielle Probleme

# Ambiguität der Interpunktionszeichen: Komma

Inwiefern ist z.B. das Komma ambig?

- Satzzeichen
  - *Ebensweit entfernt vom Banalen wie vom Exzentrischen, war sein Talent geschaffen, den Glauben des breiten Publikums und die bewundernde,fordernde Teilnahme der Wählerischen zugleich zu gewinnen.*
- Dezimalzeichen im Deutschen bzw. zur Gliederung großer Zahlen im Englischen
  - *3,14159*
  - *\$1,000,000*
- in Währungsangaben
  - *EUR 30,-*

# Ambiguität der Interpunktionszeichen: Doppelpunkt

- Satzzeichen
  - vor wörtlicher Rede
  - vor Aufzählungen
- Gliederungszeichen in Uhrzeiten, Fußballergebnissen
  - *2:15, 0:3*

# Ambiguität der Interpunktions: Bindestrich

- Silbentrennung am Zeilenende
  - *Thermodrucker arbeiten mit temperaturempfindlichem Spezialpapier.*
- Teil von Komposita
  - *AI-Studierende*
  - *Papier-und-Bleistift-Methode*
- Anzeigen von Ellipse bei Koordination
  - *An- und Verkauf*
  - *Bundestags- und -ratsabgeordnete*

# Ambiguität der Interpunktionszeichen: Bindestrich

- Gedankenstrich (oft als Halbgeviertstrich, d.h. länger ausgeführt: – oder —)
  - »*Sehen Sie, Aschenbach hat von jeher nur so gelebt*«–und der Sprecher schloß die Finger seiner Linken fest zur Faust–; »niemals so«–und er ließ die geöffnete Hand bequem von der Lehne des Sessels hängen.
- Aufzählungszeichen
  - – *Erstens*
  - – *Zweitens*
  - – *Drittens* ...
- Von-bis-Angaben bei Zahlen
  - *2–3 Stunden*

# Spezialprobleme bei der Satzgrenzenerkennung

- Dem Punkt als Satzgrenzenmarkierung folgt *in der Regel* ein Leerzeichen (oder das Zeilenende)
    - aber: weitere Satzzeichen können direkt nach einem Satzpunkt stehen!
    - *Er sagte, "Das war ein interessanter Bericht."*
  - Nach dem Satzpunkt wird *in der Regel* grossgeschrieben
    - aber: bestimmte Eigennamen werden immer kleingeschrieben
    - *Die vietnamesische Anwältin Bui Thi Kim Thanh wurde im Juli aus einem psychiatrischen Krankenhaus entlassen. ai hatte eine Eilaktion [...] (<http://www2.amnesty.de>)*
    - Aufzählungszeichen, math. Variablen usw. können auch am Satzanfang kleingeschrieben werden
      - *a. She saw a Woody Allen movie yesterday.*
      - *b. She saw a Woody Allen movie in Paris.*
- (Beispiel aus Manning and Schütze (1999, S. 103))

# Spezialprobleme bei der Satzgrenzenerkennung

- Orthographisch markierte Sätze sind ineinander verschachtelt (häufig bei Zitaten)
  - » *Man soll schweigen !* « *dachte Aschenbach erregt , indem er die Journale auf den Tisch zurückwarf.*
  - » *Es ist also kein Übel in Venedig ?* « *fragte Aschenbach sehr leise und zwischen den Zähnen.*

(Beispiele aus Thomas Manns Der Tod in Venedig)
- Nicht mit Satzzeichen markierte Satzgrenzen
  - z.B. Überschriften, Bildunterschriften usw.

# Spezialprobleme der Tokenisierung

Mehrere durch Whitespace getrennte Zeichenketten bilden intuitiv ein Wort/Token: **Multi-Word Expressions (MWE)**

- Mit Leerzeichen gegliederte Ziffernfolgen
  - *(0234) 32 25112*
- Komplexe Namen oder englische Komposita
  - *New York, Otto von Bismarck, noun phrase*
- Worttrennung am Zeilenende oder bei Ellipse
  - *Auf- und Untergang* (→ *Aufgang und Untergang?*)
- Eigentlich: solche Zeichenketten als 1 Token tokenisieren
- U.a.: Trennung am Zeilenende rückgängig machen
  - aber nicht immer trivial:  
*Gestern abend tra-  
fen wieder zwei BVB-  
Spieler.*

# Spezialprobleme der Tokenisierung

Orthographische Regeln sind sprachabhängig

- Dezimalkomma vs. Dezimalpunkt in DE vs. EN
  - *1.003,145* vs. *1,003.145*
- Markierung von Abkürzungen mit Doppelpunkt im Schwedischen
  - *c:a* (circa), *S:ta* (Sankta = Heilige)
- Unterschiedlicher Gebrauch von Anführungszeichen
  - “ ”, « » , » «
- In manchen Sprachen werden Wörter üblicherweise nicht durch Whitespace getrennt, z.B. im Chinesischen und im Japanischen
  - 晋朝，是中国历史上的朝代之一。  
Jin-Dynastie ist China Geschichte auf REL Dynastie REL eine.  
'Die Jin-Dynastie ist eine der Dynastien in der Geschichte Chinas.'

# Spezialprobleme der Tokenisierung

## Formatvielfalt — Beispiel: Telefonnummern

Phone number	Country	Phone number	Country
01713780647	UK	+45 43 48 60 60	Denmark
(44.171) 830 1007	UK	95-51-279648	Pakistan
+44 (0) 1225 753678	UK	+411/284 3797	Switzerland
01256 468551	UK	(94-1) 866854	Sri Lanka
(202) 522-2230	USA	+49 69 136-2 98 05	Germany
1-925-225-3000	USA	33 1 34 43 32 26	France
212. 995.5402	USA	++31-20-5200161	The Netherlands

Verschiedene Formate für Telefonnummern aus Anzeigen einer einzigen Ausgabe des *Economist* (nach Manning and Schütze (1999, S. 131))

# Spezialprobleme der Tokenisierung

- In einigen Sprachen gibt es sogenannte **Klitika** (Sg: *Klitikon*)
  - syntaktisch selbstständige Wörter
  - lehnen sich phonologisch an andere Wörter an
  - oft als "Kontraktionen" bezeichnet
  - Englisch: *They're my father's cousins.*
  - Französisch: *Montrez-le à l'agent!*
  - Deutsch: *Ich hab's ins Haus gebracht.*
- Sollen die Verbindungen von Basis und Klitikon als ein oder zwei Token betrachtet werden?

# Symbolische und Statistische Verfahren (CL2)

## 3: Wahrscheinlichkeitstheorie

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum



# Themenüberblick

- Ausblick: Sprachmodelle
  - was ist das?
  - wozu braucht man sie?
  - nächste Sitzung mehr dazu
- Wahrscheinlichkeitstheorie
- Die Formel von Bayes

# Wie geht es weiter?

- *grünen* \_\_
- *großen grünen* \_\_
- *den großen grünen* \_\_
- *schluckte den großen grünen* \_\_
- *Susanne schluckte den großen grünen* \_\_

Aufgaben eines klassischen **Sprachmodells**

- 1 Wahrscheinlichkeit einer Sequenz angeben
- 2 Vorhersagen des nächsten Wortes auf Basis der vorhergehenden

# Sprachmodelle

- Wozu braucht man ein Sprachmodell?
- Beispiel: Spracherkennung (Diktiersysteme, etc.)



(Synthesizer: AT&T Text-to-Speech; Demo: <http://www2.research.att.com/~ttsweb/tts/demo.php>)

- Zwei mögliche Hypothesen eines Spracherkenners:
  - 1 \*I wreck a nice beach.
  - 2 I recognize speech. ← wahrscheinlichere Hypothese
- Ein Sprachmodell bestimmt, welches die wahrscheinlichere Hypothese ist

# Sprachmodelle

- Einsatzgebiete von Sprachmodellen
  - Spracherkennung
    - Akustisches Modell + Sprachmodell
  - Handschrifterkennung
    - Schreibmodell + Sprachmodell
  - Rechtschreibkorrektur
    - Vertippermodell + Sprachmodell
  - Maschinelle Übersetzung (MT)
    - Übersetzungsmodell + Sprachmodell der Zielsprache
- Methoden auch auf andere Aufgaben übertragbar, z.B.
  - Word Sense Disambiguation (WSD)
  - Probabilistisches Parsing

# Gliederung

## 1 Wahrscheinlichkeitstheorie

- Bayes' Theorem

# Wahrscheinlichkeitstheorie

Wie wahrscheinlich ist es, dass ein bestimmtes Ereignis eintritt?

- z.B. 3x nacheinander eine 6 zu würfeln?
- z.B. nach *grünen* das Wort *Frosch* zu lesen/hören
- z.B. nach einem Adjektiv ein Nomen zu lesen/hören?

# Wichtige Konzepte/Terminologie

- **Zufallsexperiment** (“trial”): Vorgang mit beobachtbarem, zufälligen Ausgang
  - z.B. würfeln (welche Zahlen werden geworfen?)
  - z.B. einen Text lesen (welche Wörter kommen vor?)
- Ein mögliches **Ergebnis/Ausgang** (“basic **outcome**”, “sample point”)
  - z.B. ‘K(opf)’ beim Münzwurf (mit einer Münze)
  - z.B. ‘155’ bei drei unterscheidbaren Würfeln
    - Notation ‘155’: 1. Würfel = 1; 2.+3. Würfel = 5
- **Ergebnisraum  $\Omega$**  (“**sample space- dt. auch: Stichprobenraum, Ergebnismenge, ...
- z.B. {K, Z} bei einer Münze
- z.B. {111, 112, 113, ..., 666} bei drei Würfeln**

# Konzepte/Terminologie (Forts.)

- **Ereignis** oder **Event**: Teilmenge von  $\Omega$  (= Menge von einzelnen Outcomes)
  - Menge  $A$  von Einzelergebnissen
- Beispiele:
  - Kopf werfen (1 Münze)
    - $A = \{K\}$
  - Kopf oder Zahl werfen (1 Münze)
    - $A = \{K, Z\}$
  - eine ungerade Zahl würfeln (1 Würfel)
    - $A = \{1, 3, 5\}$
  - genau eine 2 würfeln (3 Würfel)
    - $A = \{211, 213, 214, \dots, 121, 123, \dots, 662\}$
  - mindestens eine 2 würfeln (3 Würfel)
    - $A = \{211, 212, 213, 214, \dots, 121, 122, \dots, 662\}$

# Konzepte/Terminologie (Forts.)

- Ereignisraum  $\mathcal{F}$  ("event space"): Potenzmenge von  $\Omega$ 
  - = Menge aller möglichen Ereignisse/Events
- Beispiel Münze:  $\mathcal{F} = \{\emptyset, \{K\}, \{Z\}, \{K, Z\}\}$ 
  - $\emptyset$ : das unmögliche Ereignis
  - $\{K, Z\} = \Omega$ : das sichere Ereignis

# Beispiel: 1 Münze

- Experiment (“trial”): Wurf mit einer Münze
- ein möglicher Outcome/Ausgang: z.B.  $Z$
- Sample space  $\Omega = \{K, Z\}$
- Event/Ereignis: Teilmenge von  $\Omega$ 
  - z.B. ‘Kopf werfen’:  $A = \{K\}$
- Ereignisraum  $\mathcal{F}$ : Potenzmenge von  $\Omega$ 
  - $\mathcal{F} = \{\emptyset, \{K\}, \{Z\}, \{K, Z\}\}$
- Aufgabe: Übertragen auf Wurf mit 3 (unterscheidbaren) Münzen
  - Notation: z.B. KKZ: Münze 1+2:  $K$ ; Münze 3:  $Z$

# Beispiel: 3 Münzen

- Experiment (“trial”): Wurf mit drei (geordneten) Münzen
- ein möglicher Outcome/Ausgang: z.B. ZKK
- Sample space  $\Omega$ : Menge der möglichen Outcomes:  
 $\Omega = \{KKK, KKZ, KZK, ZKK, KZZ, ZKZ, KZZ, ZZZ\}$
- Event/Ereignis: Teilmenge von  $\Omega$   
z.B. ‘1x Kopf werfen’:  $\{KZZ, ZKZ, ZZK\}$
- Ereignisraum  $\mathcal{F}$ : Potenzmenge von  $\Omega$   
 $\{\emptyset, \{KKK\}, \{KKZ\}, \dots, \{KKK, KKZ\}, \dots, \{KKK, KKZ, KZK, \dots\}, \dots\}$

# Wahrscheinlichkeitsfunktion $P$

- Ereignisse können unterschiedlich wahrscheinlich sein
- Z.B. bei 1 Münze:
  - $\{K\}$  ist unwahrscheinlicher als  $\{K, Z\}$
  - Frage: wie wahrscheinlich sind die beiden Ereignisse jeweils?

# Wahrscheinlichkeitsfunktion $P$

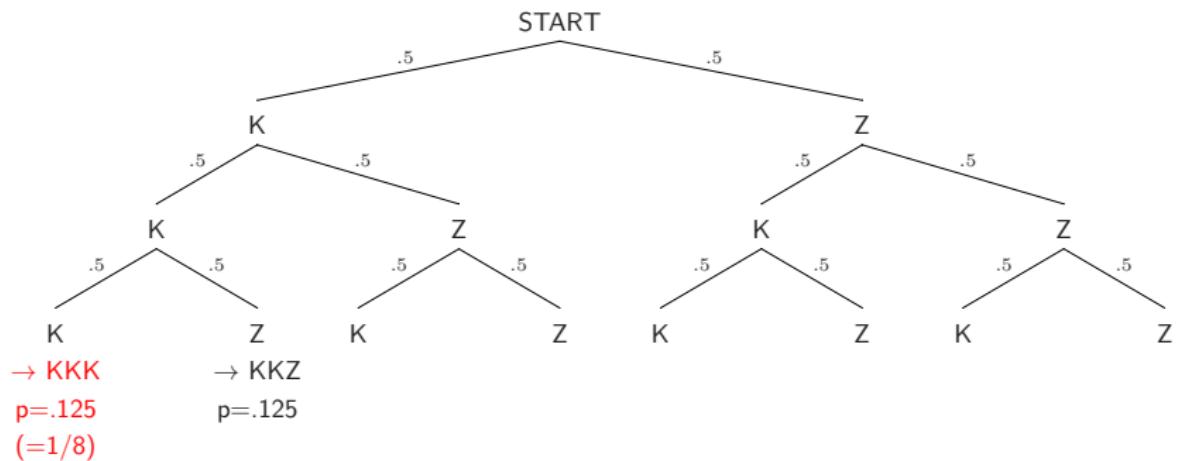
- Die Wahrscheinlichkeit eines Ereignisses ist eine Zahl zwischen 0 (“unmögliches Ereignis”) und 1 (“sicheres Ereignis”)
- Wahrscheinlichkeitsfunktion  $P$ :
  - bildet ein Ereignis auf seine Wahrscheinlichkeit ab
    - z.B.  $P(A) = 0.4$
  - verteilt eine Wahrscheinlichkeitsmasse von insgesamt 1 über die einzelnen Outcomes
    - d.h. die Summe der Einzelwahrscheinlichkeiten ist 1
- formal: Funktion  $P : \mathcal{F} \rightarrow [0, 1]$  mit
  - 1  $P(\Omega) = 1$  (“Wahrscheinlichkeitsfunktion/Verteilung”)
  - 2  $P(\emptyset) = 0$
  - 3 für disjunkte Mengen  $A, B \in \mathcal{F}$ :  $P(A \cup B) = P(A) + P(B)$  (“Additivität”)
  - Die Gesamtwahrscheinlichkeit  $P(A)$  des Ereignisses  $A$  berechnet sich als Summe der Wahrscheinlichkeiten der einzelnen Outcomes, aus denen sich das Ereignis zusammensetzt

# Zurück zum Münzen-Beispiel

Drei Münzen werden geworfen. Wie hoch ist die Wahrscheinlichkeit, dass 2x 'Zahl' kommt?

- Annahme: Münzen sind nicht getürkt, d.h. jedes Ergebnis (Outcome) ist gleich wahrscheinlich ("uniform distribution")
- Daher:  $P(KKK) = P(KKZ) = \dots = P(ZZZ) = \frac{1}{8}$ 
  - weil  $|\Omega| = 8$
- $P(A)?$ 
  - das Ereignis  $A$ , für das wir uns interessieren, umfasst 3 Outcomes:  $A = \{KZZ, ZKZ, ZZK\}$
  - daher:  
$$P(A) = P(ZZK) + P(ZKZ) + P(ZKZ) = \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8} = .375$$
  - oder einfacher:  $P(A) = \frac{|A|}{|\Omega|} = \frac{3}{8}$  (wegen uniformer Verteilung)

# Wahrscheinlichkeiten im Baumdiagramm



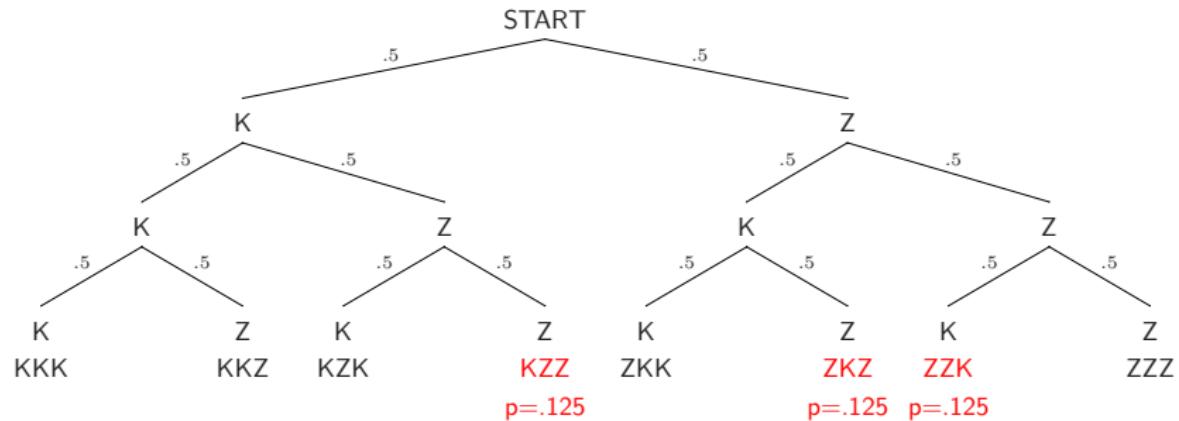
- $P(KKK) = \frac{1}{|\Omega|} = \frac{1}{8} = .125$

- Alternative Berechnung: Jedes der Teilereignisse  $K, Z$  (d.h. jeder Wurf einer einzelnen Münze) ist unabhängig voneinander:

- $P(K) = P(Z) = 0.5$

- $P(KKK) = P(K) * P(K) * P(K) = 0.5^3 = 0.125$

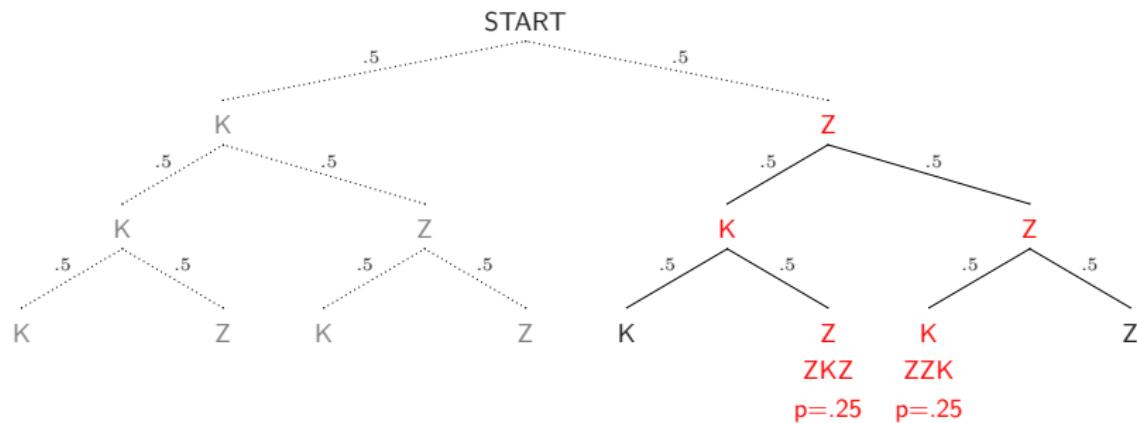
# Wahrscheinlichkeiten im Baumdiagramm: 2x Zahl



# Bedingte Wahrscheinlichkeiten

- Falls ein Teil-Ereignis bereits bekannt ist (z.B. 1 Wurf von 3), verändert sich typischerweise die Wahrscheinlichkeit des (Gesamt-)Ereignisses: **bedingte Wahrscheinlichkeit**
- Beispiel von oben (Wahrscheinlichkeit von 2x Zahl bei 3 Münzen): angenommen, die erste Münze ergab eine Zahl, wie hoch ist dann die Wahrscheinlichkeit unseres Ereignisses?
- Es bleiben noch 4 Outcomes übrig:  $ZKK$ ,  $ZKZ$ ,  $ZZK$ ,  $ZZZ$ , davon ergeben 2 das gesuchte Ereignis  $\rightarrow P(2xZ | Z) = .5$

# Bedingte Wahrscheinlichkeiten

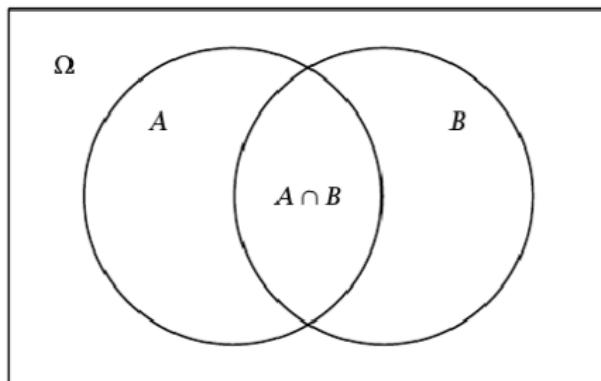


# Bedingte Wahrscheinlichkeiten

- Wahrscheinlichkeit des Events *ohne* die zusätzliche Information: **a-priori-Wahrscheinlichkeit** (prior probability) des Ereignisses
  - $P(2xZ) = 0.375$
- Wahrscheinlichkeit des Events *gegeben* die zusätzliche Information: **a-posteriori-Wahrscheinlichkeit** (posterior probability) des Ereignisses
  - $P(2xZ | Z) = 0.5$

# Bedingte Wahrscheinlichkeiten

- Sobald bekannt ist, dass das Ergebnis in B liegt, ist die Wahrscheinlichkeit von Ereignis A:  $\frac{P(A \cap B)}{P(B)}$ 
  - $A \cap B$ : A und B gelten gleichzeitig



- Bedingte Wahrscheinlichkeit** eines Events A, gegeben dass ein Event B eingetreten ist ( $P(B) > 0$ ):

$$P(A | B) := \frac{P(A \cap B)}{P(B)}$$

# Bedingte Wahrscheinlichkeiten: Münzwurf

- Bedingte Wahrscheinlichkeit:  $P(A | B) = \frac{P(A \cap B)}{P(B)}$
- Beispiel für den Einsatz dieser Formel: Angenommen, die erste Münze war eine Zahl, wie hoch ist dann die Wahrscheinlichkeit unseres Ereignisses '2x Zahl'?
  - Abkürzungen: '2x Zahl':  $2xZ$ , 1. Wurf = Zahl:  $1=Z$

$$P(2xZ | 1=Z)$$

$$= \frac{P(2xZ \cap 1=Z)}{P(1=Z)}$$

$$= \frac{P(\{ZZK, ZKZ\})}{P(\{ZKK, ZKZ, ZZK, ZZZ\})}$$

$$= \frac{2/8}{4/8} = .5$$

# Bedingte Wahrscheinlichkeiten umgestellt

- Bedingte Wahrscheinlichkeit:  $P(A | B) := \frac{P(A \cap B)}{P(B)}$
- Umstellung ergibt: **Multiplikationssatz** (Produktregel)
  - $P(A \cap B) = P(A | B) \cdot P(B) = P(B | A) \cdot P(A)$ 
    - das gilt sogar, wenn  $P(B) = 0$
    - Umdrehen möglich wegen Symmetrie des Schnitts
- Anwendungsbeispiel (mit nicht uniformer Verteilung)
  - Angenommen:
    - von den Kugeln in einer Urne sind 10% rot, 90% schwarz
    - 50% der roten Kugeln sind klebrig und 20% der schwarzen
  - Frage: mit welcher Wahrscheinlichkeit zieht man eine rote klebrige Kugel?
    - Frage:  $P(\text{klebrig} \cap \text{rot})?$
    - $P(k \cap r) = P(k | r) \cdot P(r) = 0.5 \cdot 0.1 = 0.05$

# Verallgemeinerter Multiplikationssatz

- Statt mit nur 2 Ereignissen beliebig viele:

$$P(A_1 \cap \dots \cap A_n)$$

$$= P(A_1) \cdot \frac{P(A_1 \cap A_2)}{P(A_1)} \cdot \frac{P(A_1 \cap A_2 \cap A_3)}{P(A_1 \cap A_2)} \dots \cdot \frac{P(A_1 \cap \dots \cap A_n)}{P(A_1 \cap \dots \cap A_{n-1})}$$

$$= P(A_1)P(A_2 | A_1)P(A_3 | A_1 \cap A_2) \dots P(A_n | \cap_{i=1}^{n-1} A_i)$$

- → **Chain rule**

- Sehr relevant für statistische NLP!

# Unabhängigkeit von Ereignissen

- 2 Ereignisse sind **unabhängig** voneinander, wenn  
 $P(A \cap B) = P(A) \cdot P(B)$
- Alternativ ausgedrückt (äquivalent):  $P(A) = P(A | B)$ 
  - (außer wenn  $P(B) = 0$ )
  - d.h.:  $B$  zu wissen beeinflusst nicht die Wahrscheinlichkeit von  $A$
  - Herleitung:

$$P(A | B) := \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$$

# Wahrscheinlichkeiten für “Oder-Ereignisse”

- Satz von der totalen Wahrscheinlichkeit: Setzt sich ein Ereignis  $A$  aus disjunkten Ereignissen  $B_1$  und  $\overline{B_1}$  zusammen, dann gilt:

$$\begin{aligned}P(A) &= P(A \cap B_1) + P(A \cap \overline{B_1}) \\&= P(A | B_1) \cdot P(B_1) + P(A | \overline{B_1}) \cdot P(\overline{B_1})\end{aligned}$$

- Anwendungsbeispiel

- Angenommen:

- von den Kugeln in einer Urne sind 10% rot, 90% schwarz
    - 50% der roten Kugeln sind klebrig und 20% der schwarzen

- Frage: mit welcher Wahrscheinlichkeit zieht man eine klebrige Kugel?

- Frage:  $P(\text{klebrig})$ ?

$$\begin{aligned}P(k) &= P(k | r) \cdot P(r) + P(k | s) \cdot P(s) \\&= 0.5 \cdot 0.1 + 0.2 \cdot 0.9 = 0.05 + 0.18 = 0.23\end{aligned}$$

# Gliederung

## 1 Wahrscheinlichkeitstheorie

- Bayes' Theorem

# Bayes' Formel/Theorem

- Mit der Bayes'schen Formel kann man Abhängigkeiten zwischen Ereignissen umdrehen
- D.h. wir können damit  $P(B | A)$  mit Hilfe von  $P(A | B)$  berechnen
- Die Formel ergibt sich direkt aus der Definition von bedingter Wahrscheinlichkeit und der Multiplikationsregel
- $$P(B | A) := \frac{P(B \cap A)}{P(A)}$$
  - nach Definition der bed. Wahrscheinlichkeit
- Zähler:  $P(B \cap A) = P(A | B) \cdot P(B)$ 
  - nach Multiplikationssatz
- **Bayes' Theorem:** 
$$P(B | A) = \frac{P(A | B) \cdot P(B)}{P(A)}$$

# Bayes' Formel/Theorem

- **Bayes' Theorem:**  $P(B | A) = \frac{P(A|B) \cdot P(B)}{P(A)}$
- Nenner:  $P(A)$ 
  - Normalisierungskonstante
  - garantiert, dass es eine Wahrscheinlichkeitsfunktion ist
  - oft aber in NLP: nur **maximales Argument**  $B$  interessant, das die Wahrscheinlichkeit  $P(B | A)$  maximiert
- $\operatorname{argmax}_B P(B | A)$ 
$$= \operatorname{argmax}_B \frac{P(A|B) \cdot P(B)}{P(A)}$$
$$= \operatorname{argmax}_B P(A | B) \cdot P(B)$$

# Bayes' Formel/Theorem

- **Bayes' Theorem:**  $P(B | A) = \frac{P(A|B) \cdot P(B)}{P(A)}$
- Der Nenner  $P(A)$  lässt sich aber prinzipiell auch berechnen
  - nach Satz der totalen Wahrscheinlichkeit
- Nenner:

$$\begin{aligned}P(A) &= P(A \cap B_1) + P(A \cap \overline{B_1}) \\&= P(A | B) \cdot P(B) + P(A | \overline{B}) \cdot P(\overline{B})\end{aligned}$$

# Wettervorhersage mit Bayes' Formel

Beispiel von <http://www.matheprisma.uni-wuppertal.de/>

- Im Ort "Schönwetter" regnet es im langjährigen Mittel an 3/4 aller Tage
- Die örtlichen Meteorologen sind recht erfolgreich bei der Vorhersage von schlechtem Wetter, weniger gut bei gutem Wetter
  - 90% aller Regentage werden korrekt vorhergesagt
  - nur 50% der (seltenen) Sonnentage werden korrekt vorhergesagt
- Wie sicher ist eine Vorhersage von Regen?
  - d.h.: Wie groß ist die Wahrscheinlichkeit dafür, dass auf die Vorhersage "Regen" auch wirklich Regen folgt?

# Wettervorhersage mit Bayes' Formel

- Erstmal sammeln: welche Informationen haben wir?
  - $R$ : es regnet am nächsten Tag  $\rightarrow P(R) = .75$
  - $\bar{R}$ : es regnet nicht  $\rightarrow P(\bar{R}) = .25$
  - $V$ : es ist Regen vorhergesagt  $\rightarrow P(V) = ?$
  - $\bar{V}$ : Sonne vorhergesagt  $\rightarrow P(\bar{V}) = ?$
- Und welche bedingten Wahrscheinlichkeiten?
  - Regen korrekt vorhergesagt: 90%  $\rightarrow P(V | R) = .90$
  - Sonne korrekt vorhergesagt: 50%  $\rightarrow P(\bar{V} | \bar{R}) = .50$
- Gesucht: Wie glaubhaft ist eine Vorhersage von Regen?  
 $\rightarrow P(R | V)$ 
  - gegeben aber stattdessen:  $P(V | R)!$
  - **Bayes' Formel/Theorem besagt, wie man  $P(R | V)$  aus  $P(V | R)$  berechnen kann**

# Wettervorhersage mit Bayes' Formel

$$P(R | V)$$

$$= \frac{P(R \cap V)}{P(V)}$$

per def.

$$= \frac{P(R \cap V)}{P(V \cap R) + P(V \cap \bar{R})}$$

totale Wahrsch. (Nenner)

$$= \frac{P(R \cap V)}{P(V|R) \cdot P(R) + P(V|\bar{R}) \cdot P(\bar{R})}$$

per def. (Nenner)

$$= \frac{P(R) \cdot P(V|R)}{P(V|R) \cdot P(R) + P(V|\bar{R}) \cdot P(\bar{R})}$$

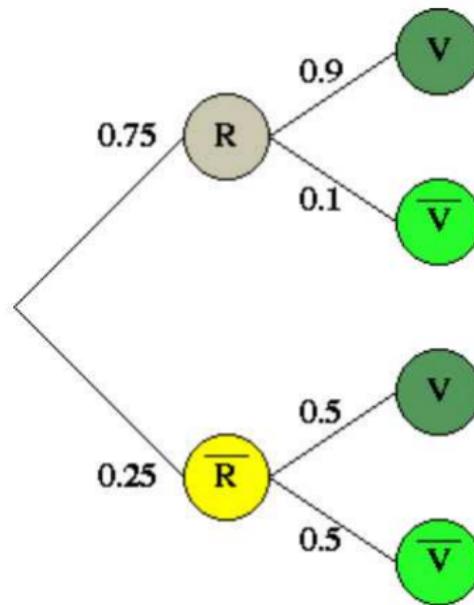
per def. (Zähler)

$$= \frac{.75 \cdot .9}{.9 \cdot .75 + .5 \cdot .25} = \frac{.675}{.675 + .125}$$

$$= .844$$

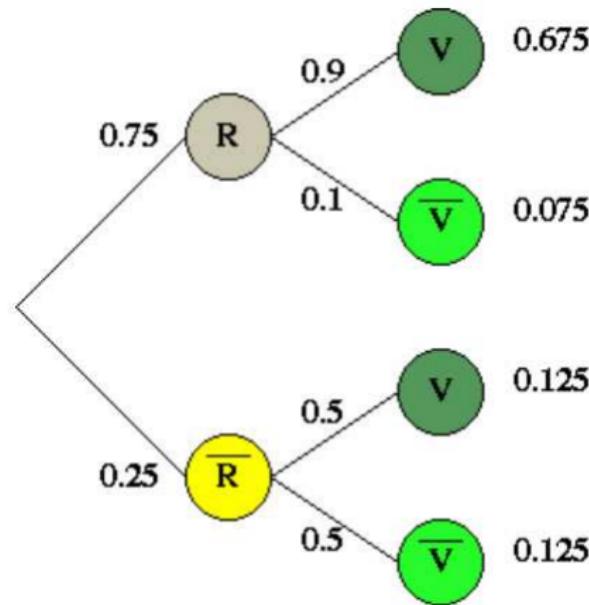
# Wettervorhersage mit Bayes' Formel

1.  $P(R) = 0.75$ ,  $P(V | R) = 0.9$ , zunächst gesucht:  $P(R \cap V)$



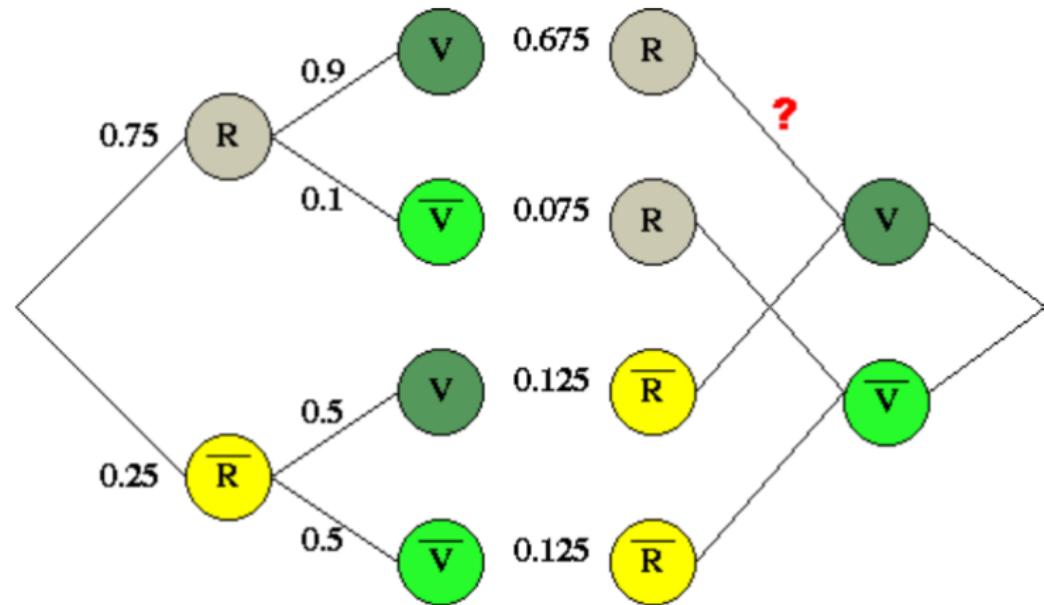
# Wettervorhersage mit Bayes' Formel

## 1. $P(R \cap V)$



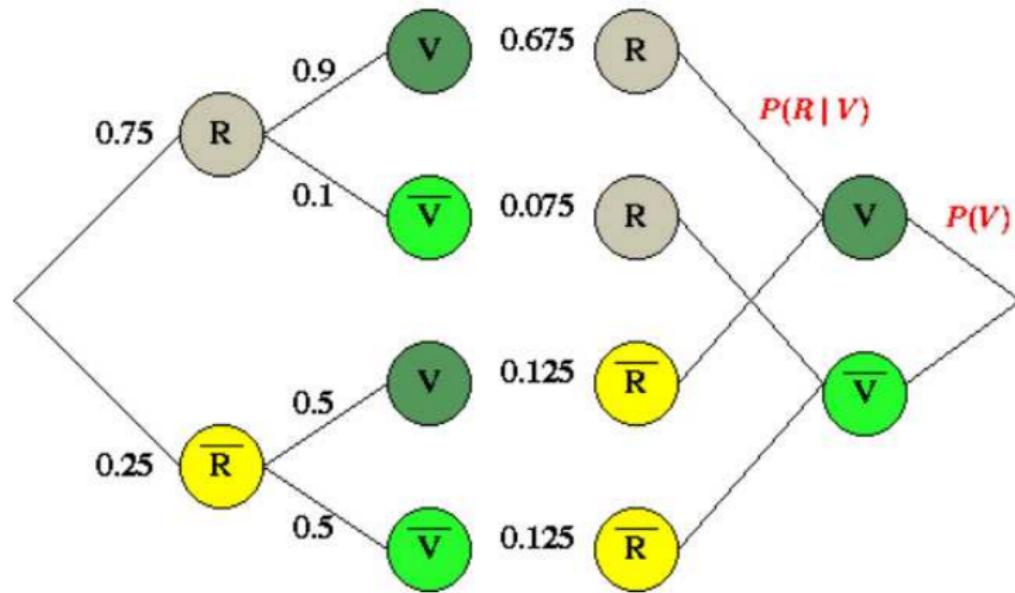
# Wettervorhersage mit Bayes' Formel

2. eigentlich gesucht:  $P(R | V)$



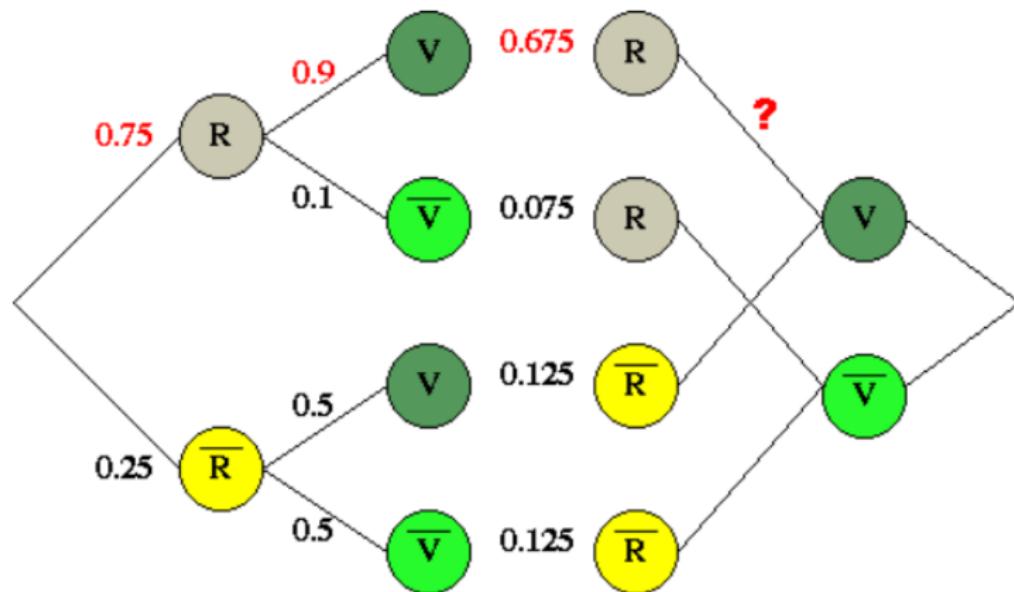
# Wettervorhersage mit Bayes' Formel

3. Gleichung für  $P(R | V) = \frac{P(R \cap V)}{P(V)}$



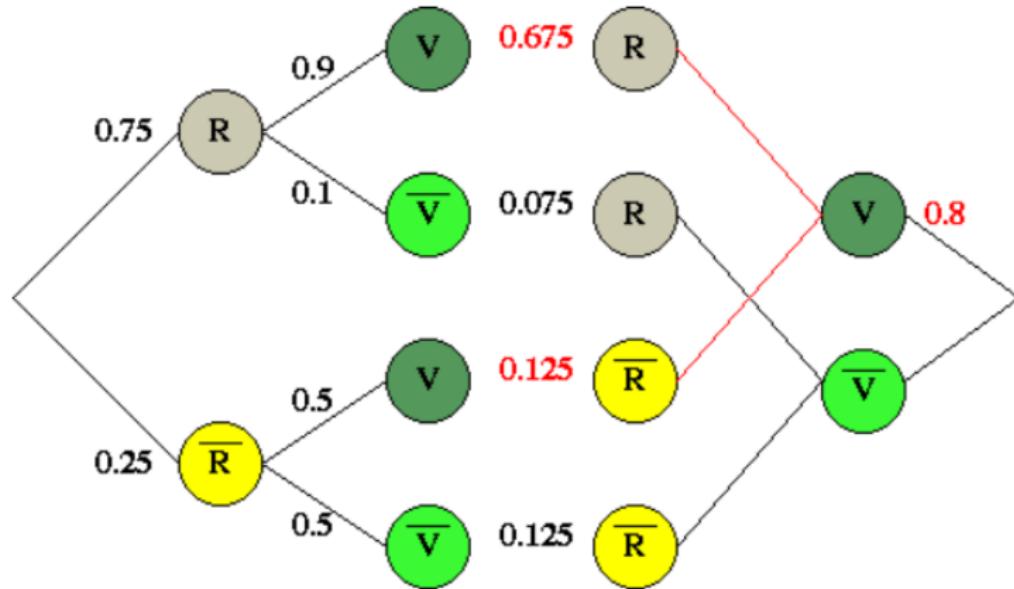
# Wettervorhersage mit Bayes' Formel

3a. Zähler ablesen:  $P(R \cap V) = 0.675$



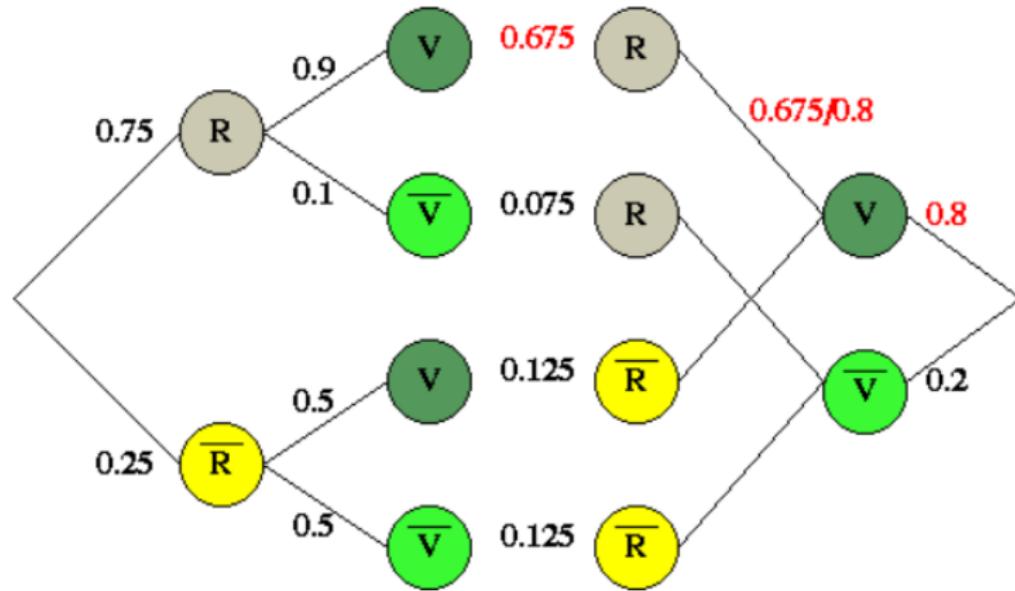
# Wettervorhersage mit Bayes' Formel

3b. Nenner ausrechnen:  $P(V) = P(V \cap R) + P(V \cap \bar{R}) = 0.675 + 0.125 = 0.8$



# Wettervorhersage mit Bayes' Formel

4. einsetzen:  $P(R | V) = \frac{P(R \cap V)}{P(V)} = \frac{0.675}{0.8} = 0.844$



# Ein linguistisches Beispiel

- Es gibt eine lange Forschungsgeschichte zu “parasitic gaps” (PG), einer seltenen syntaktischen Konstruktion
  - selten: im Durchschnitt 1x pro 100.000 Sätze
- Die niedrige Frequenz ist ein Problem für die Forschung: man findet zu wenig Beispiele, um genügend Material für die Forschung zu haben
- Aber eine Lösung ist in Sicht! Johanna Findevogel hat einen Pattern-Erkennner konstruiert, der ziemlich gut funktioniert, allerdings nicht perfekt:
  - falls ein Satz ein PG enthält, erkennt der Erkennner das in 95% der Fälle
  - falls ein Satz kein PG enthält, meldet der Erkennner fälschlicherweise in 0.5% der Fälle ein PG
- (Das hört sich sehr gut an, oder nicht?)
- Frage: Wie gut ist der Erkennner? Wieviel zahlen wir dafür?
  - konkret: angenommen, der Erkennner schlägt an. Wie hoch ist die Wahrscheinlichkeit, dass der Satz tatsächlich ein PG enthält?

# Ein linguistisches Beispiel (Forts.)

- Wir verwenden folgende Abkürzungen:
  - $G$ : Gap kommt vor;  $T$ : Erkenner schlägt an
- Bekannt:
  - $P(G) = 0.00001$  (1x pro 100.000 Sätze)
  - $P(T|G) = 0.95$  und  $P(T|\bar{G}) = 0.005$
- Gesucht:  $P(G|T)$  (Erkenner schlägt an und es ist ein Gap)
  - $P(G|T) = \frac{P(T|G) \cdot P(G)}{P(T)}$
  - zunächst Zähler:  

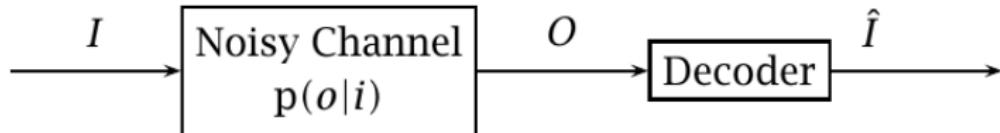
$$P(T|G) * P(G) = 0.95 \cdot 0.00001$$
  - Nenner:  

$$\begin{aligned} P(T) &= P(T|G)P(G) + P(T|\bar{G})P(\bar{G}) \\ &= 0.95 \cdot 0.00001 + 0.005 \cdot 0.99999 \\ &= 0.002 \end{aligned}$$
  - D.h. von 500 Sätzen, bei denen der Erkenner anschlägt, enthält nur 1 tatsächlich einen Gap!
    - Grund: die prior probability der Gaps ist extrem niedrig

# Noisy Channel-Modell

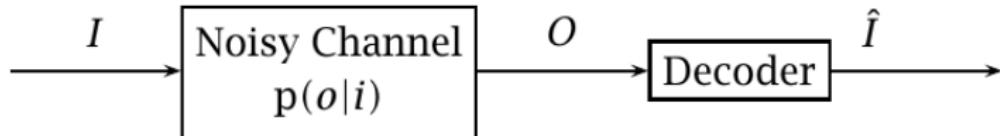
Bayes' Theorem spielt eine wichtige Rolle beim **Noisy Channel-Modell**

# Noisy Channel-Modell in der Sprachverarbeitung



- Idee des Noisy Channel-Modells:
  - wir beobachten das Datum  $O$ (utput) und gehen davon aus, dass das eigentliche Signal  $I$ (nput) war
  - $I$  wurde bei der Übertragung im **Noisy Channel** "gestört" und deformiert
  - wir müssen  $I$  aus  $O$  rekonstruieren = **dekodieren**
- Frage also: gegeben ein Output  $O$ , wie sieht der wahrscheinlichste Input  $I$  aus?

# Noisy Channel-Modell in der Sprachverarbeitung



- Viele NLP-Probleme können so aufgefasst werden: den wahrscheinlichsten Input aus dem gegebenen Output bestimmen
- Formel:  $\hat{I} = \operatorname{argmax}_i p(i | o) = \operatorname{argmax}_i \frac{p(i)*p(o|i)}{p(o)}$ 
  - nutzt Bayes' Formel!
- Da nach dem maximalen Argument gesucht wird, ist der absolute Wert irrelevant:

$$\dots \operatorname{argmax}_i \frac{p(i)*p(o|i)}{p(o)} = \operatorname{argmax}_i p(i) * p(o | i)$$

# Beispiel Spracherkennung

Das Problem der Spracherkennung:

$$\operatorname{argmax}_W P(W | A) = \frac{P(A | W) \cdot P(W)}{P(A)}$$

Wahrscheinlichkeit des akustischen Signals

- Ist eine Konstante (gleich für alle Kandidaten W)
- Kann außer Acht gelassen werden

# Beispiel Spracherkennung

Das Problem der Spracherkennung:

$$\operatorname{argmax}_W P(W | A) = \frac{P(A | W) \cdot P(W)}{P(A)}$$

## Akustisches Modell

- Modell von Wörtern und ihrer akustischen Realisierung
- oder: Schrift/Vertipper/...-Modell: Modell von Wörtern und ihrer handschriftlichen/getippten/... Realisierung

# Beispiel Spracherkennung

Das Problem der Spracherkennung:

$$\operatorname{argmax}_W P(W | A) = \frac{P(A | W) \cdot P(W)}{P(A)}$$

Sprachmodell; a priori-Wahrscheinlichkeit

- Wie wahrscheinlich sind bestimmte Wörter und Wortsequenzen generell?
  - unabhängig vom aktuellen akustischen Signal

# Was heute geschah

- Wahrscheinlichkeitstheorie
- Bayes' Regel
- Nächste Sitzung:
  - Sprachmodelle: Wahrscheinlichkeiten von Wortsequenzen
  - Markov-Modelle

# References I

- Abschnitt 2.1 aus Manning and Schütze (1999)

Manning, C. D. and H. Schütze (1999).  
*Foundations of Statistical Natural Language Processing.*  
Cambridge, MA: The MIT Press.

# Symbolische und Statistische Verfahren (CL2)

## 4: Sprachmodelle

Stefanie Dipper

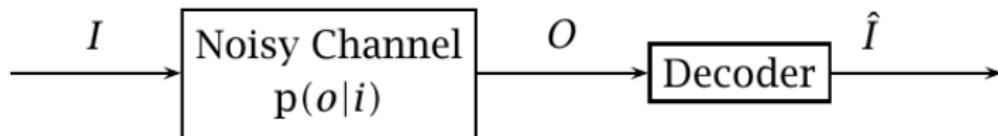
Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum



# Themenüberblick

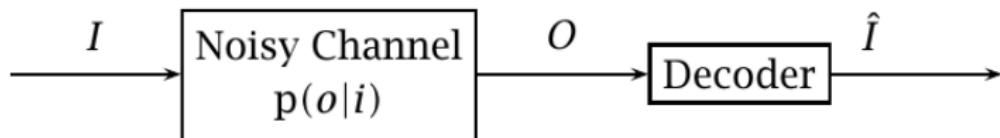
- Sprachmodelle in Form von Ngramm-Modellen
  - Markov-Annahme
  - Training und Anwendung
  - das Problem der knappen Daten ('sparse data')

# Vom letzten Mal: Noisy Channel-Modell in der Sprachverarbeitung



- Idee des Noisy Channel-Modells:
  - wir beobachten das Datum  $O$ (utput) und gehen davon aus, dass das eigentliche Signal  $I$ (nput) war
  - $I$  wurde bei der Übertragung im **Noisy Channel** "gestört" und deformiert
  - wir müssen  $I$  aus  $O$  rekonstruieren = **dekodieren**
- Frage also: gegeben ein Output  $O$ , wie sieht der wahrscheinlichste Input  $I$  aus?

# Noisy Channel-Modell in der Sprachverarbeitung



- Viele NLP-Probleme können so aufgefasst werden: den wahrscheinlichsten Input aus dem gegebenen Output bestimmen
- Formel:  $\hat{I} = \operatorname{argmax}_i p(i | o) = \operatorname{argmax}_i \frac{p(i)*p(o|i)}{p(o)}$ 
  - nutzt Bayes' Formel!
- Da nach dem maximalen Argument gesucht wird, ist der absolute Wert irrelevant:

$$\dots \operatorname{argmax}_i \frac{p(i)*p(o|i)}{p(o)} = \operatorname{argmax}_i p(i) * p(o | i)$$

# Beispiel Spracherkennung

Das Problem der Spracherkennung:

$$\operatorname{argmax}_W P(W | A) = \frac{P(A | W) \cdot P(W)}{P(A)}$$

Wahrscheinlichkeit des akustischen Signals

- Ist eine Konstante (gleich für alle Kandidaten W)
- Kann außer Acht gelassen werden

# Beispiel Spracherkennung

Das Problem der Spracherkennung:

$$\operatorname{argmax}_W P(W | A) = \frac{P(A | W) \cdot P(W)}{P(A)}$$

## Akustisches Modell

- Modell von Wörtern und ihrer akustischen Realisierung
- oder: Schrift/Vertipper/...-Modell: Modell von Wörtern und ihrer handschriftlichen/getippten/... Realisierung

# Beispiel Spracherkennung

Das Problem der Spracherkennung:

$$\operatorname{argmax}_W P(W | A) = \frac{P(A | W) \cdot P(W)}{P(A)}$$

Sprachmodell; a priori-Wahrscheinlichkeit

- Wie wahrscheinlich sind bestimmte Wörter und Wortsequenzen generell?
  - unabhängig vom aktuellen akustischen Signal
- Heute geht es nur um Sprachmodelle
  - ein Modell des Typs  $p(o | i)$  lernen wir später kennen

# Statistische Inferenz

Sprachmodellierung beruht auf **Statistischer Inferenz**

- Gegeben eine Menge von Daten, die gemäß einer unbekannten Distribution generiert wurde
  - z.B. ein Textkorpus
  - z.B. eine Baumbank (= ein Korpus mit syntaktischen Analysen)
- Ziel: die unbekannte Distribution (näherungsweise) erschließen/inferieren, so dass Vorhersagen gemacht werden können
  - z.B. Vorhersage des nächsten Worts
  - z.B. Vorhersagen zu PP-Attachment

# Statistische Inferenz

- 3 Problembereiche:
  - 1 Trainingsdaten in **Äquivalenzklassen** aufteilen
  - 2 geeignete statistische **Schätzer** für jede Klasse finden
  - 3 mehrere Schätzer geeignet kombinieren (nächste Sitzung)
- Wir nutzen das Beispiel der Sprachmodellierung, um Statistische Inferenz einzuführen

# Statistische Inferenz

- Grundidee von Inferenz: um ein Merkmal zu inferieren, suche andere Merkmale, die dieses Merkmal vorhersagen
- **Klassifikationsaufgabe**
  - Zielmerkmal ('target feature') auf Basis von ausgewählten Klassifikationsmerkmalen ('classificatory features') vorhersagen
  - z.B. das nächste Wort "klassifizieren" = auf Basis der vorhergehenden Wörter vorhersagen

# Statistische Inferenz

Annahmen:

- **Zeitliche Invarianz (Stationarität)**: die Wahrscheinlichkeiten im Modell ändern sich nicht mit der Zeit; d.h. vergangenes Verhalten ist ein guter Hinweis auf zukünftiges Verhalten
  - Verhalten in der Vergangenheit lässt Schlüsse auf die Zukunft zu
  - d.h. wenn zu früheren Zeitpunkten ein bestimmtes Wort oft auf bestimmte andere Wörter gefolgt ist, dann gilt das weitgehend auch heute so
  - ... und wir können das Modell also aus Trainingsdaten ableiten (s.u.)
- **Unabhängigkeit**: andere Merkmale als die verwendeten sind nicht bzw. wenig relevant für die Aufgabe

# Äquivalenzklassen

- Nicht sinnvoll, für sämtliche möglichen Kombinationen von Klassifikationsmerkmalen das Zielmerkmal zu lernen
- Stattdessen: Merkmale in Äquivalenzklassen gruppieren
- Klassen können unterschiedlich fein sein
  - je feinkörniger (= kleiner) die Klassen, desto mehr Merkmale → desto präziser die Vorhersagen
  - allerdings: je kleiner die Klassen, desto problematischer werden statistisch zuverlässige Schätzungen (zu wenig Trainingsdaten) → desto schwieriger zu erlernen
- Gesucht: Kompromiss zwischen beiden Anforderungen  
→ **Ngramm-Modelle**

# Gliederung

## 1 Sprachmodelle: Ngramm-Modelle

## 2 Sparse Data

# Sprachmodelle: Aufgabe

Nochmal ein Schritt zurück: Aufgabe des gesuchten Sprachmodells:

- die Wahrscheinlichkeit einer Sequenz von Wörtern zu berechnen
- den Suchraum für mögliche folgende Wörter einzuschränken
- in Kombination mit anderen Modellen (z.B. dem akustischen Modell) die Wahrscheinlichkeit der Hypothesen zu berechnen

# Modelltypen

- Sprachmodelle: Formale Modelle, die die Struktur der Sprache beschreiben
- Implementierungsmöglichkeiten:
  - Endliche Automaten
  - **Ngramm-Modelle (Markov-Modelle)**
  - Deterministische (kontextfreie) Grammatiken
  - Probabilistische kontextfreie Grammatiken
  - .....

# Sprachmodell

- Ziel: die Wahrscheinlichkeit einer Sequenz berechnen
- Gesucht also:  $P(A_1 \cap \dots \cap A_n)$

■ d.h. Wahrscheinlichkeit einer Sequenz von Wörtern  
 $A_1, \dots, A_n$

- Zur Erinnerung: Chain rule (Multiplikationssatz)

$$P(A_1 \cap \dots \cap A_n)$$

$$= P(A_1) \cdot \frac{P(A_1 \cap A_2)}{P(A_1)} \cdot \frac{P(A_1 \cap A_2 \cap A_3)}{P(A_1 \cap A_2)} \cdots \cdot \frac{P(A_1 \cap \dots \cap A_n)}{P(A_1 \cap \dots \cap A_{n-1})}$$

$$= P(A_1)P(A_2 | A_1)P(A_3 | A_1 \cap A_2) \dots P(A_n | \cap_{i=1}^{n-1} A_i)$$

- D.h. Wahrscheinlichkeit der Sequenz = Produkt der einzelnen bedingten Wahrscheinlichkeiten

# Ngramme-Modelle / Markov-Modelle

- Gesucht: Wahrscheinlichkeitsfunktion  $P(w_i|w_1, \dots, w_{i-1})$  für verschiedene  $i$  soll geschätzt werden
- Vereinfachung durch **Markov-Annahme**:  
$$P(w_i|w_1, \dots, w_{i-1}) \approx P(w_i|w_{i-1})$$
  - d.h. wir beschränken die **history** (den vorhergehenden Kontext) auf z.B. das vorhergende Wort
  - dann  $n = 2$  (Bigramme):  
$$P(w_i|w_1, \dots, w_{i-1}) \approx P(w_i|w_{i-1})$$
  - oder  $n = 3$  (Trigramme):  
$$P(w_i|w_1, \dots, w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$$
  - etc.
- Der betrachtete Kontext wird auf  $n - 1$  vorhergehende Token eingeschränkt!
  - d.h. wir vergrößern die Äquivalenzklassen
- Grund: sonst zuviele Parameter, um verlässliche Schätzer inferieren zu können (s.u.)

# Ngramme

- Eine Kombination von  $n$  aufeinander folgenden Token bezeichnet man als **Ngramm** (auch: N-Gramm, ‘ $n$ -gram’)

$n$	Ngramm-Typ	Beispiel
1	Unigramm	< <i>heute</i> >
2	Bigramm	< <i>heute, morgen</i> >
3	Trigramm	< <i>heute, morgen, um</i> >
4	Viergramm	< <i>heute, morgen, um, 10</i> >
5	Fünfgramm	< <i>heute, morgen, um, 10, Uhr</i> >
usw.		

- Traditionell hat man es bei computerlinguistischen Verfahren mit Uni-, Bi-, oder Trigramm-Modellen zu tun
  - Achtung: Es gibt auch Ngramme mit Buchstaben als Tokens!

# Markov-Annahme: beschränkte History

- Annahme: Die Wahrscheinlichkeit für ein Ereignis hängt nur vom direkt vorhergehenden Ereignis ab
  - beispielsweise nur vom vorhergehenden Token oder dem vorhergehenden Bigramm
- Beispiel mit unterschiedlich großen Kontexten:
  - *grünen* \_\_
  - *großen grünen* \_\_
  - *den großen grünen* \_\_
  - *schluckte den großen grünen* \_\_

# Markov-Modelle

- Alle Histories mit den gleichen  $n$  Token fallen in eine Äquivalenzklasse → Markov-Modell  $n$ -ter Ordnung
  - Bigramm-Modell: Modell erster Ordnung  
(das letzte Wort des Ngramms ist das vorherzusagende)
- Brauchen wir nicht mehr als Bi-/Trigramm-Modelle?
- Angenommen: Vokabular von 20,000 Wörtern:

Modell	Anzahl Parameter
Bigramm-Modell	$20,000 \times 19,999 = 4 \cdot 10^8$ (400 Mio)
Trigramm-Modell	$20,000^2 \times 19,999 = 8 \cdot 10^{12}$
Viergramm-Modell	$20,000^3 \times 19,999 = 1.6 \cdot 10^{17}$

- Daher (traditionell): Bi-/Trigramme und möglichst kleines Vokabular
- Googles Web-1T-5-gram-Daten enthalten Häufigkeiten für  $n = 1..5$ , die im Web gezählt worden sind  
(<https://catalog.ldc.upenn.edu/LDC2006T13>)

# Statistische Inferenz

## ■ 3 Problembereiche:

- 1** Trainingsdaten in **Äquivalenzklassen** aufteilen
  - DONE: Ngramm-Modelle
- 2** geeignete statistische **Schätzer** für jede Klasse finden
  - TODO: lernen, das target-Merkmal auf Basis der Äquivalenzklassen vorherzusagen
- 3** mehrere Schätzer geeignet kombinieren (nächste Sitzung)

# Training: Schätzer bestimmen

- Ziel: Trigramm-Modell
    - d.h. auf Basis zweier Vorgängerworte das nächste Wort vorhersagen
  - Angenommen, wir wollen das dritte Wort nach der Sequenz *comes across* vorherzusagen lernen
  - Angenommen, wir haben in einem Korpus beobachtet:
    - 10x die Folge *comes across*
    - 8x gefolgt von *as*, 1x von *more*, 1x von *a*
- Wir können beobachtete **relative Frequenzen** als Schätzer für Wahrscheinlichkeiten nutzen
- $P(as) = 0.8$
  - $P(more) = P(a) = 0.1$
  - $P(x) = 0.0$  für alle sonstigen Wörter  $x$

# Training: Maximum-Likelihood-Schätzung

- Ableitung eines Ngramm-Modells aus einem **Trainingskorpus**
- Zunächst zählen der absoluten Häufigkeiten aller Ngramm-Typen der relevanten Ordnungen (1..n) im Korpus
  - Wahrscheinlichkeit eines Ngramms ist gleich der relativen Häufigkeit im Trainingskorpus

$$P_{MLE}(w_1, \dots, w_n) = \frac{C(w_1, \dots, w_n)}{N_{\text{Ngramme}}}$$

$$P_{MLE}(w_n | w_1, \dots, w_{n-1}) = \frac{C(w_1, \dots, w_n)}{C(w_1, \dots, w_{n-1})}$$

- mit  $C(w_1, \dots, w_n)$ : Frequenz ('count') des Ngramms im Korpus
- und  $N_{\text{Ngramme}}$ : Anzahl aller Ngramme im Korpus

- **Maximum-Likelihood-Schätzung ('MLE')**

# Training: MLE

- Term “Maximum Likelihood Estimation”: Parameter sind so gewählt, dass sie den Beobachtungen im Korpus die höchste Wahrscheinlichkeit zuweisen
- Wir verwenden den Ausdruck  $P_{MLE}$  sowohl für “normale” wie auch bedingte Wahrscheinlichkeiten
  - welche Wahrscheinlichkeit jeweils gemeint ist, wird durch die Form des Arguments von  $P_{MLE}$  klar

## Beispiel: Wahrscheinlichkeit bestimmen (MLE)

- Nach MLE:

$$P_{MLE}(w_1, \dots, w_n) = \frac{C(w_1, \dots, w_n)}{N_{\text{Ngramme}}}$$

- Das Bigramm  $\langle \text{dem}, \text{Bewußtsein} \rangle$  kommt 6 mal im Freud-Korpus vor
- Das Korpus ist 18,498 Bigramme lang
- $P_{MLE}(\text{dem}, \text{Bewußtsein}) = 6/18498 \approx 0.000324359$

# Beispiel: bedingte Wahrscheinlichkeit bestimmen

- Nach MLE:

$$P_{MLE}(w_n | w_1, \dots w_{n-1}) = \frac{C(w_1, \dots w_n)}{C(w_1, \dots w_{n-1})}$$

- Das Bigramm  $\langle \text{dem}, \text{Bewußtsein} \rangle$  kommt 6 mal im Freud-Korpus vor
- Das Unigramm  $\langle \text{dem} \rangle$  kommt 96 mal im Freud-Korpus vor
- $P_{MLE}(\text{Bewußtsein}|\text{dem}) = 6/96 = 0.0625$

## Einschub: Beispiel Minikorpus

- Bsp: Minikorpus **a b c d**
- Frage: Welche Bi- und Trigramme enthält das Minikorpus?
  - Bigramme: **ab, bc, cd**
  - Trigramme: **abc, bcd**
- Problem? **b** und **c** kommen zu oft vor!
- Lösung: Padding

# Padding

- **Padding:** füge  $n - 1$  Dummy-Tokens am Anfang und Ende jedes Satzes an
  - bei Buchstaben-Ngrammen: jedes Wortes
- Sinn:
  - 1 um auch typische Satz/anfänge und -enden zu modellieren
  - 2 damit das erste Token eines Korpus "gleich viel" wie alle anderen Tokens zählt, d.h. auch in  $n$  Ngrammen vorkommt
- Dummy-Tokens
  - bei Buchstaben-Ngrammen häufig "#"
  - sonst auch Schlüsselwörter wie "<START>" und "<END>"
  - <START> und <END> sind komplementäre Tags, sollten beim Vokabular V daher wie 1 Tag gezählt werden!
- Teilt man ein Korpus in mehrere Teile, dann enden die Subkorpora möglicherweise mitten in einem Satz und die "Endtoken" werden dann nur einfach gezählt; bei großen Daten kann das vernachlässigt werden

## Beispiel Minikorpus mit Padding

- Minikorpus Bigramme: # a b c d #
  - #a, ab, bc, cd, d#
- Minikorpus Trigramme: # # a b c d # #
  - ##a, #ab, abc, bcd, cd#, d##

# Anwendung

- Das Ngramm-Modell mit den geschätzten Wahrscheinlichkeiten wird nun “angewendet”
- Z.B. gegeben eine Menge von Kandidatensätzen, bestimme den wahrscheinlichsten Satz
- Im folgenden: Berechnung der Wahrscheinlichkeit eines Kandidaten

# Anwendung

Berechnung der Wahrscheinlichkeit einer Sequenz  $w_1..w_i$

- Produkt der einzelnen bedingten Wahrscheinlichkeiten

$$P(w_1, \dots, w_i) = P(w_1|w_0..w_{i-n+1}) \cdot \dots \cdot P(w_i|w_{i-1}, \dots, w_{i-n+1})$$

- Bei einem Bigramm-Modell

$$P(w_1, \dots, w_i) = P(w_1|w_0) \cdot \dots \cdot P(w_i|w_{i-1})$$

- Beispiel:

$$P(\text{Vorher nur noch eine Bemerkung.})$$

$$\begin{aligned} &= P(\text{Vorher}|\text{<START>}) \cdot P(\text{nur}|\text{Vorher}) \cdot P(\text{noch}|\text{nur}) \\ &\quad \cdot P(\text{eine}|\text{noch}) \cdot P(\text{Bemerkung}|\text{eine}) \cdot P(\text{.}|\text{Bemerkung}) \end{aligned}$$

$$= 1/625 \cdot 1/1 \cdot 2/36 \cdot 1/39 \cdot 1/117 \cdot 1/1$$

$$\approx 1.948e - 08$$

- d.h. 625 Sätze im Korpus, davon fängt einer mit Vorher an

# Anwendung von Ngramm-Modellen

- Folgende Folien: Vergleich verschiedener Ngramm-Modelle mit  $n \in \{1, 2, 3, 4\}$
- Gegeben: entsprechende Ngramm-Frequenzen aus einem Korpus (Texte von Jane Austen)
- Aufgabe: Wahrscheinlichkeit von Satz (1) berechnen
  - dieser Satz kam so nicht im Korpus vor

## Example

1 *(In person) she was inferior to both sisters*

# Anwendung des Unigramm-Modells

1-gram	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	
1	the	0.034	the	0.034	the	0.034	the	0.034	the	0.034	
2	to	0.032	to	0.032	to	0.032	<b>to</b>	<b>0.032</b>	to	0.032	
3	and	0.030	and	0.030	and	0.030			and	0.030	
4	of	0.029	of	0.029	of	0.029			of	0.029	
...											
8	was	0.015	<b>was</b>	<b>0.015</b>	was	0.015			was	0.015	
...											
13	<b>she</b>	<b>0.011</b>			she	0.011			<b>she</b>	0.011	
...											
254					both	0.0005			<b>both</b>	<b>0.0005</b>	
...											
435					sisters	0.0003				<b>sisters</b>	<b>0.0003</b>
...											
1701					<b>inferior</b>	<b>0.00005</b>					

# Anwendung des Bigramm-Modells

2-gram		$P(\cdot   person)$		$P(\cdot   she)$		$P(\cdot   was)$		$P(\cdot   inferior)$		$P(\cdot   to)$		$P(\cdot   both)$	
1	and	0.099		had	0.141	not	0.065	<b>to</b>	<b>0.212</b>	be	0.111	of	0.066
2	who	0.099		<b>was</b>	<b>0.122</b>	a	0.052			the	0.057	to	0.041
3	to	0.076				the	0.033			her	0.048	in	0.038
4	in	0.045				to	0.031			have	0.027	and	0.025
...													
23	<b>she</b>	<b>0.009</b>								Mrs	0.006	she	0.009
...													
41										what	0.004	<b>sisters</b>	<b>0.006</b>
...													
293									<b>both</b>	<b>0.0004</b>			
...													
$\infty$						<b>inferior</b>	<b>0</b>						

# Anwendung des Trigramm-Modells

3-gram	$P(\cdot   In, person)$	$P(\cdot   person, she)$	$P(\cdot   she, was)$	$P(\cdot   was, inf.)$	$P(\cdot   inferior, to)$	$P(\cdot   to, both)$
1	UNSEEN	did 0.5	not 0.057	UNSEEN	the 0.286	to 0.222
2		was 0.5	very 0.038		Maria 0.143	Chapter 0.111
3			in 0.030		cherries 0.143	Hour 0.111
4			to 0.026		her 0.143	Twice 0.111
...						
$\infty$		inferior 0		both 0	sisters 0	

# Anwendung des Viergramm-Modells

4-gram	$P(\cdot   u, I, p)$	$P(\cdot   I, p, s)$	$P(\cdot   p, s, w)$	$P(\cdot   s, w, i)$	$P(\cdot   w, i, t)$	$P(\cdot   i, t, b)$
1	UNSEEN	UNSEEN	in	1.0	UNSEEN	UNSEEN
...						
$\infty$			inferior	0		

# Gliederung

## 1 Sprachmodelle: Ngramm-Modelle

## 2 Sparse Data

# 0-Wahrscheinlichkeiten

- Problem 1: unbekannte Wörter (**Unknowns, OOV**, “Out of Vocabulary”)
- Problem 2: prinzipiell mögliche, aber **zufällig im Trainingskorpus nicht vorkommende Ngramme**
- Das MLE-Modell weist ungesiehenen Ngrammen die **Wahrscheinlichkeit 0** zu
  - ... weil es den gesehenen Daten “maximale Wahrscheinlichkeit” (= ML) zuweist
- Wahrscheinlichkeit einer Sequenz errechnet sich durch Multiplikation
- Wenn eine Teilsequenz im Trainingskorpus nicht vorgekommen ist: Gesamtsequenz erhält dann die Wahrscheinlichkeit 0!

# Tatsächlich vorkommenden Typen

- Für Ngramme mit  $n > 1$  kommt jeweils nur ein Bruchteil der möglichen Ngramme vor
- Beispiel Freud-Korpus:
  - 1 Unigramme: 19,123 Token; 4,288 (verschiedene) Typen
    - davon: 2,716 Typen (63.34%) kommen nur 1x vor (**Hapax Legomena**)
  - 2 Bigramme: 18,498 Token; 13,088 Typen
    - davon: 11,286 Typen (86.23%) kommen nur 1x vor
    - insgesamt 18,386,944 mögliche Typen: nur 13,088 (0.07%) kommen vor
    - Frage: wie berechnet sich die Zahl möglicher Bigramm-Typen? (Anzahl Unigramme)<sup>2</sup>
  - 3 Trigramme: 17,873 Token und 16,545 Typen
    - 15,900 Typen (96.10%): kommen nur 1x vor
    - 78,843,215,872 mögliche Typen: nur 2,1e-7 kommen vor

## Einschub “e-Notation”

- Wissenschaftliche Schreibweise für sehr große und kleine Zahlen
- Vor dem Komma einstellig
- Bsp:  $1,6 \cdot 10^{11}$  bzw.  $1,6\text{e}11$  oder  $1,6\text{E}11$
- Exponent gibt an, um wieviel Stellen das Komma verschoben werden muss
  - positiv: nach rechts:  $2,1\text{e}3 = 2100$
  - negativ: nach links:  $2,1\text{e}-2 = 0,021$
  - $2,1\text{e}-7?$

# Das Problem der “Sparse Data”

- Der Anteil der tatsächlich auftretenden Ngramme ist schon bei Bigrammen gering und wird mit steigendem  $n$  immer geringer
- Gründe
  - Strukturiertheit natürlicher Sprachen
    - das Vorkommen eines Tokens ist immer abhängig vom vorhergehenden und nachfolgenden Kontext
  - Problem der Datenknappheit (engl. **data sparseness**)
    - viele der nicht-vorgekommenen Ngramme wären mögliche grammatische Wortfolgen, kommen aber zufällig nicht vor
    - dies ist ein großes Problem für alle statistischen Algorithmen zur Verarbeitung natürlicher Sprache

# Das Problem der “Sparse Data”

## Eigenschaften natürlicher Sprache

- Zipf'sches Gesetz
  - sehr wenige sehr, sehr häufige Worttypen
  - sehr viele sehr, sehr seltene Worttypen
- Außerdem: mangelnde Zeitinvarianz
  - d.h. fehlende Stationarität
- Auch immer größere Trainingskorpora helfen nicht gegen das Problem der knappen Daten!

# Zipf'sches Gesetz

- **Zipf'sches Gesetz:** Die absolute Häufigkeit eines Typs ist umgekehrt proportional zu seinem Rang in der Liste der nach Häufigkeit geordneten Typen

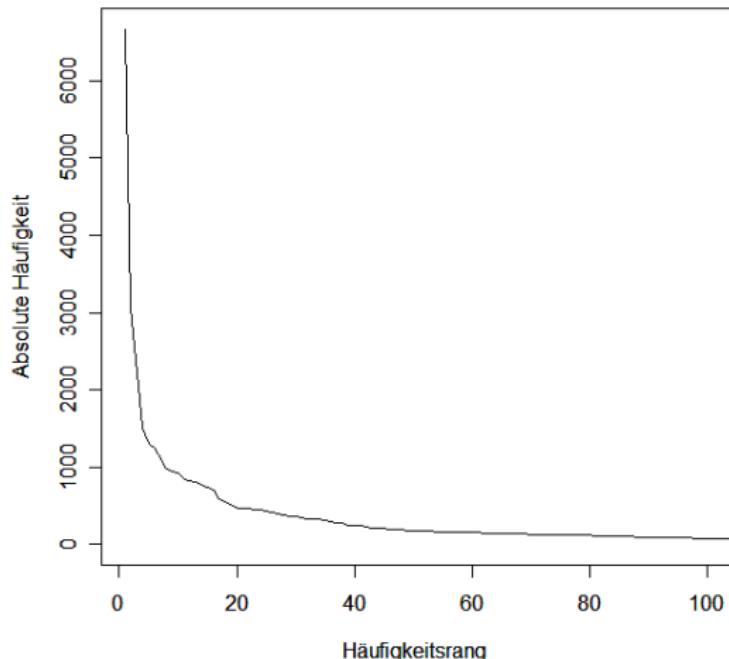
$$\text{Häufigkeit(Rang)} \sim 1/\text{Rang}$$

- Das zweithäufigste Typ ist halb so häufig wie der häufigste, der dritthäufigste Typ ist ein Drittel so häufig wie der häufigste Typ usw.
- Beispiel-Rangliste aus dem Rilke-Korpus

Typ	Rang	Anzahl	Typ	Rang	Anzahl
,	1	6666	es	6	1246
.	2	3055	sie	7	1108
und	3	2226	der	8	970
die	4	1513	war	9	945
ich	5	1301	...		

# Zipf'sches Gesetz

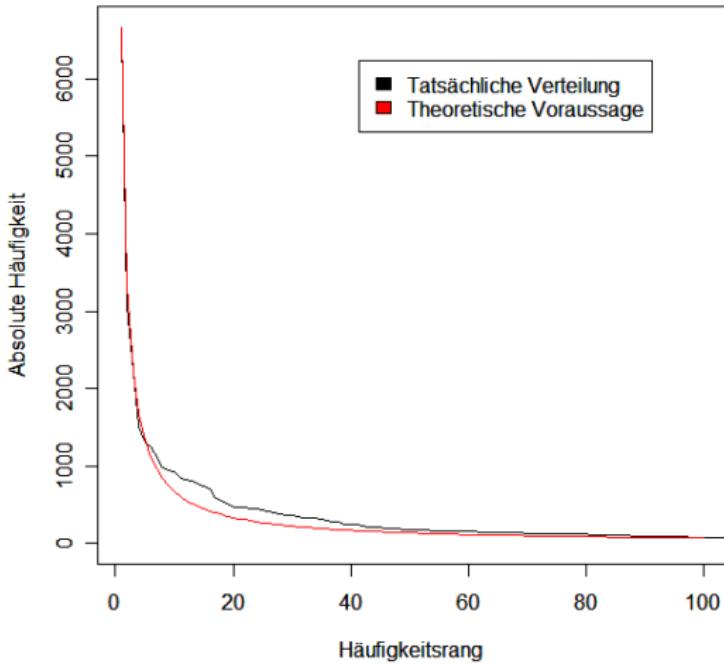
Worthäufigkeiten in einem deutschen Korpus



**LNRE-Verteilung**  
(“Large number of rare events”)

# Zipf'sches Gesetz

Worthäufigkeiten in einem deutschen Korpus



# Zusammenfassung

- Sprachmodelle: Wahrscheinlichkeiten von Wortsequenzen
- Markov-Modelle
- Ngramme-Modelle (traditionell Bi-/Trigramme)
- Schätzer: Maximum Likelihood Estimation
- Sparse data-Problem

# References I

- Ngramm-Modelle: Abschnitt 6.1–6.2.1 aus Manning and Schütze (1999)

Manning, C. D. and H. Schütze (1999).

*Foundations of Statistical Natural Language Processing.*  
Cambridge, MA: The MIT Press.

# Symbolische und Statistische Verfahren (CL2)

## 5: Smoothing

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum

RUB

# Themenüberblick

- Letztes Mal:
  - feinere Äquivalenzklassen = längere Ngramme erlauben zwar bessere Vorhersagen
  - aber sie sind schwieriger zu erlernen
  - Grund: data sparseness aufgrund der Struktur natürlicher Sprache (Zipf'sches Gesetz)
  - daher viele "0-Wahrscheinlichkeiten"
- Heute: wir "reparieren" dieses Problem
  - Smoothing: Umgang mit niedrigen Frequenzen

# Statistische Inferenz

## ■ 3 Problembereiche:

- 1** Trainingsdaten in **Äquivalenzklassen** aufteilen
  - DONE: Ngramm-Modelle
- 2** geeignete statistische **Schätzer** für jede Klasse finden
  - DONE: lernen, das target-Merkmal auf Basis der Äquivalenzklassen vorherzusagen
- 3** mehrere Schätzer geeignet kombinieren
  - TODO: heute

# Gliederung

## 1 Smoothing

Einschub: Trainings- und Testkorpora

# Glättung (Smoothing/Discounting)

- Zunächst Ableitung eines MLE-Modells aus dem Trainingskorpus
- Dann **Umverteilung von Wahrscheinlichkeitsmasse** von den vorgekommenen Ngrammen an die nicht vorgekommenen Ngramme: **Smoothing**
- Auch nicht im Trainingskorpus vorgekommene Ngramme erhalten eine Wahrscheinlichkeit  $> 0$

# Notation

Auf den weiteren Folien wird folgende Notation verwendet:

- $V$ : Anzahl Unigramm-Typen (“Vokabular”)
- $N$ : Anzahl Ngramm-Tokens (z.B. Bigramm-Tokens)
- $w_{1n}$ : Ngramm  $w_1 \dots w_n$  für  $n \in \mathbb{N}, n > 0$
- $C(w_1 \dots w_n) = C(w_{1n})$ : Frequenz des Ngramms  $w_1 \dots w_n$  (“count”)

Die Angaben zur Anzahl beziehen sich immer auf ein Korpus  
(meist: Trainingskorpus), falls nicht anders angegeben

# Notation: Beispiel

Minikorpus **a b b b b**

- **V**: Anzahl Unigramm-Typen
  - 2: a, b
- **N**: Anzahl Uni-/Bi-/Trigramm-Tokens
  - 5/4/3 (z.B. Trigramme: abb, bbb, bbb)
- **w<sub>13</sub>**: Ngramm  $w_1 \dots w_3$ 
  - a b b
- **C(w<sub>1</sub> … w<sub>3</sub>) = C(w<sub>13</sub>)**: Frequenz des Ngramms  $w_1 \dots w_3$ 
  - 1

# Notation: Beispiel

Minikorpus **a b b b b** – jetzt mit Padding!

- **V**: Anzahl Unigramm-Typen
  - 3: a, b, # (obwohl bei Unigrammen selbst nicht gepadded wird!)
- **N**: Anzahl Uni-/Bi-/Trigramm-Tokens
  - Zunächst: wie sehen die gepaddeten Versionen jeweils aus?
    - Uni: abbbb / Bi: #abbbb# / Tri: ##abbbb##
  - Dann zählen
    - 5 / 6 / 7
- **w<sub>13</sub>**: Ngramm  $w_1 \dots w_3$  (gepadded nach Bigrammen)
  - # a b
- **C(w<sub>1</sub> … w<sub>3</sub>) = C(w<sub>13</sub>)**: Frequenz des Ngramms  $w_1 \dots w_3$ 
  - 1

# Glättung/Smoothing

Vorgestellte Verfahren:

- 1a.** Add-One-Glättung (Laplace)
- 1b.** Lidstone/Jeffreys-Glättung (ELE)
- 2a.** Held-out
- 2b.** Deleted Estimation
- 2c.** Good-Turing-Glättung
- 3.** Back-off-Modelle

# Add-One-Glättung (Laplace)

- Man addiert zur absoluten Häufigkeit aller Ngramm-Typen den Wert 1 → keine Nullen mehr!
- Die Wahrscheinlichkeiten sind dann:

$$P_{Lap}(w_{1n}) = \frac{C(w_{1n}) + 1}{N + V^n}$$

$$P_{Lap}(w_n | w_1, \dots, w_{n-1}) = \frac{C(w_1, \dots, w_n) + 1}{C(w_1, \dots, w_{n-1}) + V}$$

- Zum neuen Nenner gleich mehr

# Add-One-Glättung (Laplace): Normierung

$$P_{Lap}(w_{1n}) = \frac{C(w_{1n})+1}{N+V^n}$$

- Warum neuer Nenner  $N + V^n$ ?
- Damit wieder eine Wahrscheinlichkeitsfunktion entsteht, d.h. mit einer Gesamtwahrscheinlichkeit von 100%
  - wie oft wird 1 im Zähler addiert?
  - für jeden Ngramm-Typ +1
    - für jeden *möglichen* Typ:  $C(w_{1n}) = 0$  ist möglich
  - $V^n$  = Anzahl der möglichen Ngramm-Typen der Länge  $n$ 
    - z.B. bei Trigrammen: #Unigramme<sup>3</sup>
  - $N + V^n$ : Anzahl vorkommende Ngramme + Anzahl mögliche Ngramm-Typen

# Add-One-Glättung (Laplace): Normierung

$$P_{Lap}(w_n | w_1, \dots, w_{n-1}) = \frac{C(w_1, \dots, w_n) + 1}{C(w_1, \dots, w_{n-1}) + V}$$

- Warum neuer Nenner  $C(w_1, \dots, w_{n-1}) + V$ ?
- $V$  = Anzahl der möglichen  $w_n$ 
  - $C(w_1, \dots, w_{n-1}) + V$ : Anzahl vorkommende Vorgänger + Anzahl mögliche Nachfolger
  - Nachfolger: bestehen aus 1 Token!

# Beispiel: Unigramme / Korpus: *abbbb*

- $P_{\text{Lap}}(w_{1n}) = \frac{C(w_{1n})+1}{N+V^n} \rightarrow \frac{C(w_1)+1}{N+V}$
- $P_{\text{Lap}}(w_n|w_1, \dots, w_{n-1})$ : im Unigramm-Modell irrelevant

$w_i$	$C(w_i)$	$P_{\text{MLE}}(w_i)$	$P_{\text{Lap}}(w_i)$
a	1	1/5	1+1 / 5+3
b	4	4/5	4+1 / 5+3
c	0	0/5	0+1 / 5+3
$\Sigma$	V=3	N=5	8/8 = 1

- $N = 5, V = 3$

# Beispiel: Bigramme / Korpus: aaaba

■  $P_{\text{Lap}}(w_{1n}) = \frac{C(w_{1n})+1}{N+V^n} \rightarrow \frac{C(w_1, w_2)+1}{N+V^2}$

■  $P_{\text{Lap}}(w_n|w_1, \dots, w_{n-1}) = \frac{C(w_1, \dots, w_n)+1}{C(w_1, \dots, w_{n-1})+V} \rightarrow \frac{C(w_{12})+1}{C(w_1)+V}$

$w_{ij}$	$C(w_{ij})$	$C(w_i)$	$P_{\text{MLE}}(w_{ij})$	$P_{\text{Lap}}(w_{ij})$	$P_{\text{MLE}}(w_j w_i)$	$P_{\text{Lap}}(w_j w_i)$
aa	2	4	2/6	2+1/6+9	2/4 = .5	2+1/4+3 = 3/7
ab	1	(a)	1/6	1+1/6+9	1/4 = .25	1+1/4+3 = 2/7
a#	1		1/6	1+1/6+9	1/4 = .25	1+1/4+3 = 2/7
ba	1	1	1/6	1+1/6+9	1/1 = 1	1+1/1+3 = .5
bb	0	(b)	0/6	0+1/6+9	0/1 = 0	0+1/1+3 = .25
b#	0		0/6	0+1/6+9	0/1 = 0	0+1/1+3 = .25
#a	1	1	1/6	1+1/6+9	1/1 = 1	1+1/1+3 = .5
#b	0	(#)	0/6	0+1/6+9	0/1 = 0	0+1/1+3 = .25
##	0		0/6	0+1/6+9	0/1 = 0	0+1/1+3 = .25
$\Sigma$	$V=3$	$N=6$	6	6/6	15/15	

■  $N = 6, V = 3, V^2 = 9$

# Beispiel: Trigramme / Korpus: *aaaaba*

- $P_{\text{Lap}}(w_{1n}) = \frac{C(w_{1n})+1}{N+V^n} \rightarrow \frac{C(w_{12})+1}{N+V^3}$

- $P_{\text{Lap}}(w_n|w_1, \dots, w_{n-1}) = \frac{C(w_1, \dots, w_n)+1}{C(w_1, \dots, w_{n-1})+V} \rightarrow \frac{C(w_{13})+1}{C(w_{12})+V}$

$w_{ik}$	$C(w_{ik})$	$C(w_i)$	$P_{\text{MLE}}(w_{ik})$	$P_{\text{Lap}}(w_{ik})$	$P_{\text{MLE}}(w_k w_{ij})$	$P_{\text{Lap}}(w_k w_{ij})$
aaa	2	3	2/8	2+1/8+27	2/3 = .67	2+1/3+3 = 3/6
aab	1	(aa)	1/8	1+1/8+27	1/3 = .33	1+1/3+3 = 2/6
aa#	0		0/8	1+1/8+27	0/3 = 0	0+1/3+3 = 1/6
aba	1	1	1/8	1+1/8+27	1/1 = 1	1+1/1+3 = 2/4
abb	0	(ab)	0/8	0+1/8+27	0/1 = 0	0+1/1+3 = 1/4
ab#	0		0/8	0+1/8+27	0/1 = 0	0+1/1+3 = 1/4
...						
$\Sigma$	V=3	N=8	8	8/8	35/35	

- $N = 6, V = 3, V^3 = 27$

# Add-One-Glättung (Laplace)

## Ein “echtes” Beispiel

- Nicht-vorgekommenes Bigramm < *heute, Morgen* >
  - $C(\text{heute}) = 6$
  - $C(\text{heute Morgen}) = 0$
  - $P_{MLE}(\text{Morgen}|\text{heute}) = 0$
  - $P_{\text{Lap}}(\text{Morgen}|\text{heute}) = \frac{0+1}{6+V} = \frac{1}{6+4288} \approx 0.00023$
- Vorgekommenes Bigramm < *dem, Bewußtsein* >
  - $C(\text{dem}) = 96$
  - $C(\text{dem Bewußtsein}) = 6$
  - $P_{MLE}(\text{Bewußtsein}| \text{dem}) = \frac{6}{96} \approx 0,0625$
  - $P_{\text{Lap}}(\text{Bewußtsein}| \text{dem}) = \frac{6+1}{96+V} = \frac{7}{96+4288} \approx 0,00160$

# Add-One-Glättung (Laplace)

Add-One-Glättung: sehr einfaches, aber auch ein sehr schlechtes Verfahren:

- Vergibt zu viel Wahrscheinlichkeitsmasse an ungewohnte Ereignisse
- Weist allen nicht aufgetretenen Ngrammen die gleiche Wahrscheinlichkeit zu
- Falls das Präfix nicht vorgekommen ist, ergibt sich sogar die Gleichverteilung:  $\frac{1}{V}$

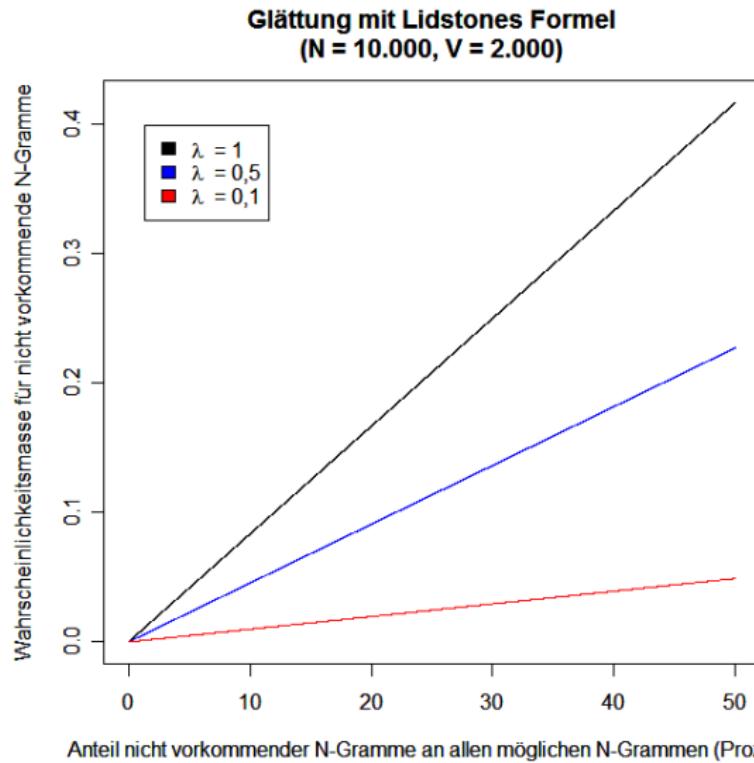
# Lidstone/Jeffreys-Perks-Glättung (ELE)

- Variante der Add-One-Glättung
- Statt 1 wird ein kleinerer Wert addiert
- Dadurch wird etwas weniger Wahrscheinlichkeitsmasse an ungewohnte Ereignisse vergeben
- Oft wird  $\lambda = 0.5$  verwendet (Jeffreys-Perks)
  - auch: Expected Likelihood Estimation (ELE)

$$P(w_{1n}) = \frac{C(w_{1n}) + \lambda}{N + \lambda \cdot V^n}$$

$$P(w_n | w_1, \dots, w_{n-1}) = \frac{C(w_1, \dots, w_n) + \lambda}{C(w_1, \dots, w_{n-1}) + \lambda \cdot V}$$

# Glättung mit Lidstones Formel

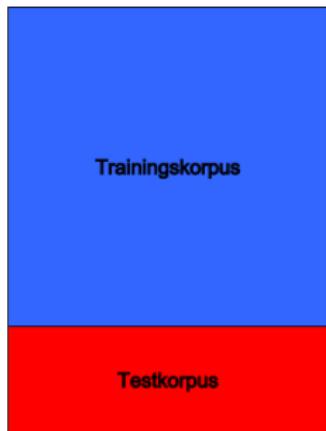


# Gliederung

## 1 Smoothing

- Einschub: Trainings- und Testkorpora

# Trainings- und Testkorpora



- Ein System darf nie auf dem Korpus evaluiert werden, auf dem es trainiert worden ist!
- Training des Systems auf dem **Trainingskorpus**
  - Schätzen der Parameter eines statistischen Modells aus den Daten im Trainingskorpus
  - Trainingskorpus dient als Referenz bei der Entwicklung eines nicht-statistischen Systems
- Test des trainierten Systems auf einem separaten **Testkorpus**

# Trainings- und Testkorpora

Komplexeres Szenario:



- Grundlegendes Training auf dem **Trainingskorpus** ('training data')
- Lernen zusätzlicher Parameter aus einem zusätzlichen **Validierungskorpus** ('held out data')
- Tests während der Entwicklung auf einem **Entwicklungstestkorpus** ('development test set')
- Evaluation für Veröffentlichung / Vergleich mit Konkurrenzsystemen auf dem eigentlichen **Testkorpus** ('test set')

# **k-fache Cross-Validierung**

- Aufteilen des Trainingskorpus in  $k$  gleich große Teile
- Training auf  $k - 1$  Teilen, Validieren auf 1 Teil
- Diese Prozedur wird  $k$  mal wiederholt, wobei jedes Mal ein anderer Teil als Validierungskorpus benutzt wird
- Schätzung der Parameter als Mittelwert aus den  $k$  Durchläufen
- Vorteile
  - effektivere Ausnutzung der Trainingsdaten
  - Schätzung der Varianz der Parameter möglich
- Auch bei der Evaluation möglich, z.B.:
  - Training auf 9/10 des gesamten Korpus
  - Testen auf 1/10 des gesamten Korpus
  - 10 Durchläufe

# 10-fache Cross-Validierung bei der Evaluation



# Beispiel für die Nutzung eines Validierungskorpus

- Schätzen des optimalen Parameterwerts  $\lambda$  für die Lidstone-Glättung aus einem zusätzlichen Korpus (= Validierungskorpus)
- Wie häufig kommt ein im Trainingskorpus nicht vorkommendes Ngramm durchschnittlich im Validierungskorpus vor?

# Beispiel $\lambda$ schätzen

Teilen des Freud-Korpus in zwei Hälften:  $K^1$  und  $K^2$

- $N_0$ : Anzahl der möglichen, aber nicht vorkommenden Bigramme in  $K^1$
- Einige dieser Bigramme kommen aber in  $K^2$  vor
- Wir nutzen deren Frequenzen in  $K^2$ , um die *erwartbare* Frequenz ungewöhnlicher Bigramme zu schätzen
- Schätzer: durchschnittliche Frequenz eines Bigramms in  $K^2$ , das nicht in  $K^1$  vorkam
  - 1 Anzahl  $N_0$  bestimmen: Differenz zwischen allen theoretisch möglichen und den tatsächlich vorkommenden Bigrammen
  - 2 Gesamtanzahl dieser Bigramme in  $K^2$  bestimmen:  $T_0^2$
  - 3 *erwartbare* durchschnittliche Frequenz:  
 $T_0^2 / N_0 = 3.042 / 5.662.979 \approx 0.000537$
- Wir könnten also beispielsweise  $\lambda = 0.000537$  setzen, also zu jedem Vorkommen nur 0.000537 statt 1 addieren

# Notation II

- $K^i$ : i-tes Teilkorpus
- $C^i(w_{1:n})$ : Frequenzen  $C(w_{1:n})$  in  $K^i$

## 1 Frequenzen bzgl. des Teilkorpus $K^1$ :

- $r$ : Frequenz eines Ngramms  $w_{1:n}$ , ermittelt durch  $C(w_{1:n})$
- $B_r$ : Menge ("Bin") mit Ngramm-Typen der gleichen Frequenz  $r$
- $N_r$ : Anzahl von Typen mit der gleichen Frequenz  $r$ 
  - **count-counts**, Frequenzen von Frequenzen
  - z.B. 30 Trigramm-Typen haben die Frequenz 7  $\rightarrow N_7 = 30$
  - $B_r$  hat  $N_r$  Elemente

## 2 Frequenzen bzgl. eines anderen Teilkorpus $K^i$ :

- $T_r^i$ : Gesamtanzahl ("Total") von Ngramm-Tokens mit  $K^1$ -Frequenz  $r$  im Teilkorpus  $K^i$ 
  - z.B. die 30 Trigramm-Typen kommen im zweiten Teilkorpus insgesamt 150x vor  $\rightarrow T_7^2 = 150$
- $S^i$ : Gesamtanzahl ("Sum") von Tokens in  $K^i$

# Held-out Estimation

- Zunächst das Trainingskorpus in 2 Teile teilen:  $K^1$  (“Trainingskorpus”) und  $K^2$  (“held-out data”)
- Dann Frequenzen für alle Ngramme in  $K^1$  berechnen
- ... und für alle vorkommenden Ngramm-Frequenzen  $r$  entsprechende Mengen  $B_r$  bilden
- Ziel: diese Frequenzen “korrigieren” anhand entsprechender Frequenzen in  $K^2$

$$P_{ho}(w_{1n}) = \frac{T_r^2}{N_r \cdot S^2} \text{ für } C^1(w_{1n}) = r$$

- 1 für jedes  $B_r$ : ermitte die summierte Frequenz  $T_r^2$  aller Ngramme aus  $B_r$  in  $K^2$ :  $T_r^2 = \sum_{\{w_{1n}: C^1(w_{1n})=r\}} C^2(w_{1n})$
- 2 ... und daraus die “erwartete” (“zukünftige”) durchschnittliche Frequenz der Ngramme berechnen:  $\frac{T_r^2}{N_r}$
- 3 daraus Wahrscheinlichkeit berechnen: durch  $S^2$  teilen

# Deleted Estimation (Jelinek and Mercer 1985)

- 2-fache Cross-Validierung mit Held-out Estimation
  - zuerst  $P_{ho}(w_{1n})$  für  $K^1$  mit Hilfe von  $K^2$  berechnen, dann umgekehrt
  - Durchschnitt aus beiden ergibt  $P_{del}(w_{1n})$
- $N_r^1, N_r^2$ : Anzahl der Typen mit Frequenz  $r$  in  $K^1$  bzw.  $K^2$
- $T_r^2$ : Summe der  $K^2$ -Vorkommen aller Typen mit Frequenz  $r$  in  $K^1$  ( $T_r^1$  analog)
- $S = 2 \cdot S^1 = 2 \cdot S^2$  (beachte:  $S^1 = S^2$ )
- Schätzung und Smoothing anhand beider Teilkorpora:

$$P_{del}(w_{1n}) = \frac{T_r^2 + T_r^1}{(N_r^1 + N_r^2) \cdot S}$$

# Vergleich verschiedener Glättungsverfahren

## Frequenz-Schätzungen

Tabelle aus Manning and Schütze (1999, S. 203)

$r = f_{MLE}$	$f_{\text{empirical}}$	$f_{\text{Lap}}$	$f_{\text{del}}$	$f_{\text{GT}}$	$N_r$	$T_r$
0	0.000027	0.000137	0.000037	0.000027	74 671 100 000	2 019 187
1	0.448	0.000274	0.396	0.446	2 018 046	903 206
2	1.25	0.000411	1.24	1.26	449 721	564 153
3	2.24	0.000548	2.23	2.24	188 933	424 015
4	3.23	0.000685	3.22	3.24	105 668	341 099
5	4.21	0.000822	4.22	4.22	68 379	287 776
6	5.23	0.000959	5.20	5.19	48 190	251 951
7	6.21	0.00109	6.21	6.21	35 709	221 693
8	7.21	0.00123	7.18	7.24	27 710	199 779
9	8.26	0.00137	8.18	8.25	22 280	183 971

Sehr ungenaue  
Schätzung

# Vergleich verschiedener Glättungsverfahren

## Frequenz-Schätzungen

Tabelle aus Manning and Schütze (1999, S. 203)

$r = f_{MLE}$	$f_{\text{empirical}}$	$f_{\text{Lap}}$	$f_{\text{del}}$	$f_{\text{GT}}$	$N_r$	$T_r$
0	0.000027	0.000137	0.000037	0.000027	74 671 100 000	2 019 187
1	0.448	0.000274	0.396	0.446	2 018 046	903 206
2	1.25	0.000411	1.24	1.26	449 721	564 153
3	2.24	0.000548	2.23	2.24	188 933	424 015
4	3.23	0.000685	3.22	3.24	105 668	341 099
5	4.21	0.000822	4.22	4.22	68 379	287 776
6	5.23	0.000959	5.20	5.19	48 190	251 951
7	6.21	0.00109	6.21	6.21	35 709	221 693
8	7.21	0.00123	7.18	7.24	27 710	199 779
9	8.26	0.00137	8.18	8.25	22 280	183 971

Relativ genaue  
Schätzung

# Deleted Estimation (Jelinek and Mercer 1985)

- Funktioniert gut!
- Überschätzt allerdings ungewohnte Bigramme
- Und unterschätzt Hapaxe

# Deleted Estimation: Beispiel (Bigramme)

$K^1:$  Freq  $r$  #Typen ( $N_r$ )

0	5,662,979
1	5,479
2	447
3	128
4	45
5	25
6	9
7	13
8	7
9	6
10	3
11	1
12	4
13	2
14	3
15	2
17	2
20	1
22	1
...	...

$K^2:$  Freq  $r$  #Typen ( $N_r$ )

0	8,046,406
1	6,914
2	563
3	168
4	57
5	34
6	30
7	10
8	8
9	10
10	8
11	4
12	3
13	3
14	4
15	1
16	3
17	1
18	3
...	...

# Deleted Estimation: Beispiel (Bigramme)

$K^1:$	$r$	$T^2$ (Sum $K_2$ )	$K^2:$	$r$	$T^1$ (Sum $K_1$ )
	0	7634		0	5695
	1	906		1	751
	2	377		2	269
	3	229		3	185
	4	129		4	64
	5	104		5	66
	6	62		6	96
	7	70		7	26
	8	87		8	28
	9	53		9	54
	10	29		10	39
	11	9		11	11
	12	57		12	16
	13	19		13	28
	14	57		14	33
	15	37		15	10
	...			...	

# Vergleich von $P_{MLE}$ und $P_{dle}$

Folgende Folien: Vergleich der Wahrscheinlichkeiten für verschiedene frequente Bigramme

# Beispiel: Mittelfrequentes Bigramm: $P_{MLE}$

Maximum-Likelihood-Wahrscheinlichkeit für  $\langle als, ob \rangle$  im gesamten Freud-Korpus:

- $C(\text{als}, \text{ob}) = 4, N = 18498$
- $P_{MLE}(\text{als}, \text{ob}) = 0.00021624$
- $P_{MLE}(\text{ob} | \text{als}) = 4/103 \approx 0.038835$

# Beispiel: Mittelfrequentes Bigramm: $P_{dle}$

Deleted Estimation:

- Anzahl der Bigrammtypen mit Frequenz 4 in  $K^1$  bzw.  $K^2$ 
  - $N_4^1 = 45, \quad N_4^2 = 57$
- Summe der  $K^2/K^1$ -Vorkommen aller Typen mit Frequenz 4 in  $K^1/K^2$ 
  - $T_4^1 = 129, \quad T_4^2 = 64$
- Anzahl der Tokens pro Teil des Trainingskorpus:  $S = 9249$

$$P_{del}(\text{als, ob}) = \frac{129 + 64}{(45 + 57) * 9249} = 0,00020458$$

$$P_{del}(\text{ob} | \text{als}) = \frac{P_{del}(\text{als,ob})}{P_{MLE}(\text{als})} \approx \frac{0.00020458}{0.005386}$$

$$\approx 0.037984$$

# Beispiel: Bigramm mit Frequenz 0: $P_{MLE}$

Maximum-Likelihood-Wahrscheinlichkeit für  $\langle als, heute \rangle$  im gesamten Freud-Korpus:

- $C(\text{als}, \text{heute}) = 0, N = 18498$
- $P_{MLE}(\text{als}, \text{heute}) = 0$
- $P_{MLE}(\text{heute} | \text{als}) = 0/103 = 0$

# Beispiel: Bigramm mit Frequenz 0: $P_{del}$

- Anzahl der Bigrammtypen mit Frequenz 0 in  $K^1$  bzw.  $K^2$ 
  - $N_0^1 = 5,662,979, \quad N_0^2 = 8,046,406$
- Summe der  $K^2/K^1$ -Vorkommen aller Typen mit Frequenz 0 in  $K^1/K^2$ 
  - $T_0^2 = 7634, \quad T_0^1 = 5695$
- Anzahl der Tokens pro Teil des Trainingskorpus:  $N = 9249$

$$P_{del}(\text{als, heute}) = \frac{7634 + 5695}{(5662979 + 8046406) * 9249} \approx 0.000000105$$

$$P_{del}(\text{heute|als}) = \frac{P_{del}(\text{als,ob})}{P_{MLE}(\text{als})} \approx \frac{0,000000105}{0,005386}$$

$$\approx 0.000019495 > 0$$

# Good-Turing-Glättung $P_{GT}$

- Good (1953) beschreibt eine Methode von Turing, die es erlaubt, die Wahrscheinlichkeit für ungewohnte Ngramme sehr genau zu schätzen
- Die Auftretenshäufigkeit für Typen, die  $r$ -mal vorkommen ( $r > 0$ ), wird wie folgt angepasst:

$$r^* = (r + 1) \frac{E(N_{r+1})}{E(N_r)}$$

- $r^*$ : korrigiertes  $r$
- und:  $P_{GT}(w_{1n}) = \frac{r^*}{N}$
- Geglättete Auftretenshäufigkeit für ungewohnte Typen:

$$r_0^* = \frac{E(N_1)}{E(N_0)}$$

- und:  $P_{GT}(w_{1n}) = \frac{N_1}{N_0 N}$

# Good-Turing-Glättung

- Schätzung der Erwartungswerte  $E$  z.B. aus den tatsächlichen Vorkommen im Korpus
- Beispiel:  $N_8 = 1342$  und  $N_9 = 1106$

$$\begin{aligned}r_8^* &= (r + 1) \cdot \frac{E(N_{r+1})}{E(N_r)} \\&= (8 + 1) \cdot \frac{E(N_9)}{E(N_8)} \\&= 9 \cdot \frac{1106}{1342} \\&= 9 \cdot 0.82 = 7.42\end{aligned}$$

# Good-Turing-Glättung: ungewohnte Typen

- Anteil der Wahrscheinlichkeitsmasse, der für ungewohnte Ngramme reserviert wird, beträgt dann

$$\frac{N_1}{N}$$

- Bsp:
  - 138,741 Hapaxe ( $N_1$ ), 617,091 Tokens insgesamt ( $N$ )
  - reservierte Masse für ungewohnte Ngramme: 0.22  
(identisch mit Masse für Hapaxe)

# Good-Turing-Glättung: Beispiel

- Types = { a b c d e f g h i j k }
- Korpus = “a b c d d e e f f f”

$r$	$N_r$	$r^*$
0	5	$(0 + 1) \cdot \frac{3}{5} = 0.6$
1	3	$(1 + 1) \cdot \frac{2}{3} = 1.3$
2	2	$(2 + 1) \cdot \frac{1}{2} = 1.5$
3	1	$(3 + 1) \cdot \frac{0}{1} = 0.0$

- Test, ob Gesamtwahrscheinlichkeiten = 1:

$$\begin{aligned}
 & 5 \cdot \frac{0.6}{10} + 3 \cdot \frac{1.3}{10} + 2 \cdot \frac{1.5}{10} + 1 \cdot \frac{0}{13} \\
 &= \frac{3}{10} + \frac{4}{10} + \frac{3}{10} + \frac{0}{10} = 1
 \end{aligned}$$

# Count-counts aus dem gesamten Freud-Korpus

$r$	$N_r$	$r$	$N_r$
1	11286	20	3
2	1089	21	2
3	296	22	1
4	139	23	1
5	90	24	1
6	44	25	1
7	32	26	4
8	16	27	1
9	15	29	1
10	9	30	1
11	8	31	2
12	11	32	2
13	5	36	2
14	2	39	1
15	3	40	1
16	6	42	1
17	2	54	1
18	3	102	1
19	2	106	1
		127	1
		625	1

# Good-Turing-Glättung

- Problem:  $r_{54}^*$  lässt sich nicht gut schätzen, da  $N_{55} = 0$  (es gab keine Typen, die 55x vorgekommen sind)
- Insbesondere:  $r_{max}^* = 0$ , da  $N_{r_{max}+1} = 0$
- Schätzung insbesondere für hohe  $r$  nicht zuverlässig
  - Für hohe  $r$  muss daher entweder direkt der empirische Wert (MLE) benutzt werden, oder  $N_r$  muss mittels Interpolation berechnet werden
  - In jedem Fall muss renormalisiert werden, so dass die Summe aller Wahrscheinlichkeiten wieder 1 beträgt (siehe Jurafsky and Martin (2009, S. 216))
- GT: hängt nicht von der Anzahl prinzipiell möglicher Ngramme ab!
- Daher: keine “Bestrafung” für großes Vokabular

Genaue Beschreibung der Simple-Good-Turing-Methode in  
Gale and Geoffrey (1995) (<http://www.grsampson.net/AGtf1.html>)

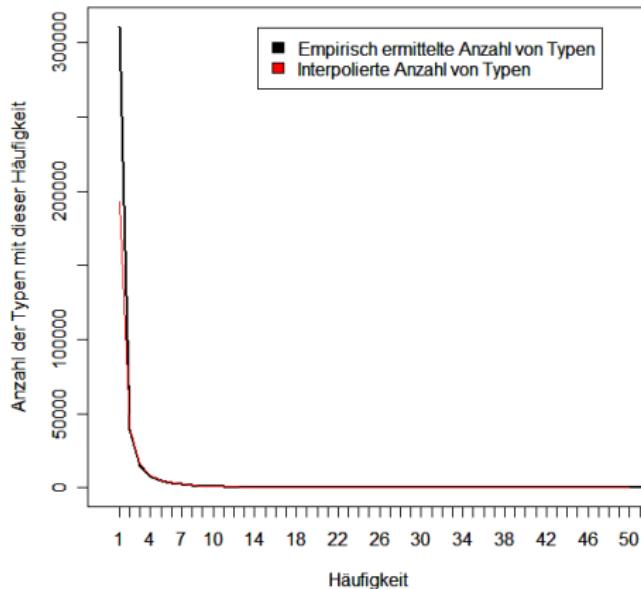
# Good-Turing-Glättung (Bigramme)

Tabelle 6.8 aus Manning and Schütze (1999, S. 203)

$r$	$r^*$	$P_{\text{GT}}(\cdot)$
0	0.0007	$1.058 \times 10^{-9}$
1	0.3663	$5.982 \times 10^{-7}$
2	1.228	$2.004 \times 10^{-6}$
3	2.122	$3.465 \times 10^{-6}$
4	3.058	$4.993 \times 10^{-6}$
5	4.015	$6.555 \times 10^{-6}$
6	4.984	$8.138 \times 10^{-6}$
7	5.96	$9.733 \times 10^{-6}$
8	6.942	$1.134 \times 10^{-5}$
9	7.928	$1.294 \times 10^{-5}$
10	8.916	$1.456 \times 10^{-5}$
...		
28	26.84	$4.383 \times 10^{-5}$
29	27.84	$4.546 \times 10^{-5}$
30	28.84	$4.709 \times 10^{-5}$
31	29.84	$4.872 \times 10^{-5}$
32	30.84	$5.035 \times 10^{-5}$
...		
1264	1263	0.002062
1366	1365	0.002228
1917	1916	0.003128
2233	2232	0.003644
2507	2506	0.004092

# Good-Turing-Glättung

Lineare Interpolation für Good-Turing Glättung  
(Häufigkeiten von Bigramm-Häufigkeiten)



# Vergleich verschiedener Glättungsverfahren

## Frequenz-Schätzungen

Tabelle aus Manning and Schütze (1999, S. 203)

$r = f_{MLE}$	$f_{\text{empirical}}$	$f_{\text{Lap}}$	$f_{\text{del}}$	$f_{\text{GT}}$	$N_r$	$T_r$
0	0.000027	0.000137	0.000037	0.000027	74 671 100 000	2 019 187
1	0.448	0.000274	0.396	0.446	2 018 046	903 206
2	1.25	0.000411	1.24	1.26	449 721	564 153
3	2.24	0.000548	2.23	2.24	188 933	424 015
4	3.23	0.000685	3.22	3.24	105 668	341 099
5	4.21	0.000822	4.22	4.22	68 379	287 776
6	5.23	0.000959	5.20	5.19	48 190	251 951
7	6.21	0.00109	6.21	6.21	35 709	221 693
8	7.21	0.00123	7.18	7.24	27 710	199 779
9	8.26	0.00137	8.18	8.25	22 280	183 971

Sehr ungenaue  
Schätzung

# Vergleich verschiedener Glättungsverfahren

## Frequenz-Schätzungen

Tabelle aus Manning and Schütze (1999, S. 203)

$r = f_{MLE}$	$f_{\text{empirical}}$	$f_{\text{Lap}}$	$f_{\text{del}}$	$f_{\text{GT}}$	$N_r$	$T_r$
0	0.000027	0.000137	0.000037	0.000027	74 671 100 000	2 019 187
1	0.448	0.000274	0.396	0.446	2 018 046	903 206
2	1.25	0.000411	1.24	1.26	449 721	564 153
3	2.24	0.000548	2.23	2.24	188 933	424 015
4	3.23	0.000685	3.22	3.24	105 668	341 099
5	4.21	0.000822	4.22	4.22	68 379	287 776
6	5.23	0.000959	5.20	5.19	48 190	251 951
7	6.21	0.00109	6.21	6.21	35 709	221 693
8	7.21	0.00123	7.18	7.24	27 710	199 779
9	8.26	0.00137	8.18	8.25	22 280	183 971

Relativ genaue  
Schätzung

# Ungesehene Ngramme

- Bisher gesehene Smoothing-Verfahren: weisen ungesehenen Ngrammen dieselbe Wahrscheinlichkeit zu
- Eigentlich: die Frequenz der Bestandteile eines Ngramms sollte eine Rolle spielen

# Back-Off-Modelle (Katz 1987)

- Intuition: Statt bei einem nicht vorkommenden Ngramm ganz auf Evidenz aus dem Kontext zu verzichten, weicht man auf das nächst niedrigere Ngramm-Modell aus
- Beispiel für Trigramme

$$P_{bo}(w_n | w_{n-2} w_{n-1}) =$$

$$\begin{cases} P_{discount}(w_n | w_{n-2} w_{n-1}) & \text{für } C(w_{n-2} w_{n-1} w_n) > 0 \\ \alpha_1 \cdot P(w_n | w_{n-1}) & \text{für } C(w_{n-2} w_{n-1} w_n) = 0 \\ & \text{und } C(w_{n-1} w_n) > 0 \\ \alpha_2 \cdot P(w_n) & \text{für } C(w_{n-1} w_n) = 0 \end{cases}$$

# Back-Off-Modelle (Katz 1987)

- Die höheren Ngramm-Modelle müssen jeweils etwas Wahrscheinlichkeitsmasse für die niedrigeren Back-Off-Modelle übriglassen
- Die Faktoren  $\alpha_1$  und  $\alpha_2$  stellen sicher, dass nur die übrig gelassene Wahrscheinlichkeitsmasse von den Back-Off-Modellen verbraucht wird
- Eine genauere Beschreibung findet sich in Jurafsky and Martin (2009, S. 139–141)

# Deleted Interpolation

- Ähnlich wie Back-Off-Modelle
- Aber: auch bei Trigrammen mit Frequenz  $> 0$  gehen die Frequenzen der enthaltenen Bi- und Unigramme ein

# Das Problem der unbekannten Wörter (OOV)

- Oft kommen im Testkorpus Wörter vor, die im Trainingskorpus nicht aufgetreten sind
- So lange man von einem festen und geschlossenen Vokabular  $V$  ausgeht, kann man den Raum der möglichen Ngramme berechnen
- Wenn man ein offenes Vokabular zulässt, ist das nicht mehr möglich
- Man kann aber schätzen, z.B. mit Hilfe der Good-Turing-Methode oder durch Training auf zusätzlichen Daten (Validierungskorpus), wie groß der Anteil aller ungesiehenen Ngramme an der Gesamtwahrscheinlichkeit ist

# Das Problem der unbekannten Wörter (OOV)

- Meist werden alle unbekannten Wörter zu einem abstrakten Typ wie z.B. <UNK> (für ‘unknown’) zusammengefasst (Manning and Schütze 1999, S. 199)
- Oft werden sogar auch Hapax Legomena auf das Token <UNK> abgebildet
- Variante: Unterscheidung von zwei Typen OOVs
  - ein Typ für alle Zahlen
  - einer für den Rest von OOV
- Nachteil: Der abstrakte Typ <UNK> ist wahrscheinlicher als viele Wörter, die tatsächlich aufgetreten sind

# Vergleich verschiedener Glättungsverfahren

Chen and Goodman (1998) vergleichen verschiedene Glättungsverfahren und erläutern noch einmal die Notwendigkeit für solche Verfahren: <http://www.ee.columbia.edu/~stanchen/papers/h015a-techreport.pdf>

# Zusammenfassung I

- Smoothing: Verfahren für niedrige/Null-Frequenzen
  - Umverteilung von Wahrscheinlichkeitsmasse
  - Add-One-Glättung
  - Lidstone/Jeffreys-Glättung (ELE)
  - Deleted Estimation
  - Good-Turing-Glättung
  - Back-off-Modelle

# References I

- Abschnitt 6 aus Manning and Schütze (1999)
- Perplexität: Abschnitt 4.4 aus Jurafsky and Martin (2009)

Chen, S. F. and J. T. Goodman (1998).

An empirical study of smoothing techniques for language modeling.

Technical Report TR-10-98, Computer Science Group, Harvard University.

Gale, W. A. and Geoffrey (1995).

Good–Turing frequency estimation without tears.

*Journal of Quantitative Linguistics*, 217–237.

Jelinek, F. and R. Mercer (1985).

Probability distribution estimation from sparse data.

*IBM Technical Disclosure Bulletin* 28, 2591–2594.

## References II

Jurafsky, D. and J. H. Martin (2009).

*Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics* (2nd ed.).

Upper Saddle River, NJ: Prentice-Hall.

Manning, C. D. and H. Schütze (1999).

*Foundations of Statistical Natural Language Processing*.

Cambridge, MA: The MIT Press.

# Symbolische und Statistische Verfahren (CL2)

## 6: Wortarten-Tagging: Grundlagen

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum

# Themenüberblick Tagging

- Heute: **Wortarten-Tagging**
- Nächste Sitzungen:
  - Hidden-Markov-Modell-Tagger
  - Brill-Tagger
  - Vergleich HMM-Tagger vs. Brill-Tagger
  - Evaluation

# Themenüberblick heute

## Wortarten-Tagging

- Nutzen
- STTS: ein Tagset für das Deutsche
- Probleme
  - unbekannte Wörter
  - Wortartenambiguität
- Stand der Forschung

# Gliederung

## 1 Tagging

- STTS: ein Tagset für das Deutsche
- Probleme
- Stand der Forschung

# Was bedeutet Tagging?

- Annotation von Token in einem Korpus mit zusätzlichen Informationen
- Meist Tagging von Wortarten (**Part of Speech, POS, PoS**)
  - ein Wort/Token im Korpus wird mit Informationen über seine syntaktische Kategorie (Wortart) annotiert
- aber z.B. auch: Tagging von Morphologie, Wortbedeutung,  
....
- Beispiel:  
*Trotz/APPR starker/ADJA Opposition/NN in/APPR  
seiner/PPOSAT eigenen/ADJA Partei/NN hatte/VAFIN  
sich/PRF Präsident/NN Clinton/NE in/APPR den/ART  
letzten/ADJA Wochen/NN mit/APPR grösster/ADJA  
Energie/NN für/APPR den/ART Vertrag/NN  
engagiert/VVPP ./\$.*

# Wozu Tagging?

Warum sind kategoriale Informationen (Wortarten) nützlich für die Sprachverarbeitung?

- Disambiguierung der Wortart hilft manchmal bei der **Disambiguierung der Wortbedeutung** (und für die Aussprache)

## Examples

- 1
  - a. Yeast is a **plant/NN**, according to the biologists
  - b. Let's **plant/VB** some trees!
- 2
  - a. Das ist sehr **modern/ADJD**
  - b. Sie **modern/VVFIN** vor sich hin

# Wozu Tagging?

Warum sind kategorische Informationen (Wortarten) nützlich für die Sprachverarbeitung?

- Disambiguierung der Wortart hilft manchmal bei der **Disambiguierung der Wortbedeutung** → auch wichtig bei der automatischen Übersetzung!

## Examples

- 1 Beispiel Google Translate (18.11.2019)
  - a. Ich lerne heute an der **Universität Wortartenanalyse**
  - b. I'm learning today at the **University of Speech Analysis**

# Wozu Tagging?

Warum sind kategorische Informationen (Wortarten) nützlich für die Sprachverarbeitung?

- **Information Retrieval:** Beschränkung bei Indizierung und Suche auf bestimmte Wortarten

## Examples

Holland ist ein Teil der Niederlande, der im Westen von der Nordsee und im Osten vom IJsselmeer begrenzt wird.

# Wozu Tagging?

Warum sind kategoriale Informationen (Wortarten) nützlich für die Sprachverarbeitung?

- **Informationsextraktion:** Interessante Ausdrücke wie z.B. komplexe Namen, Orts- oder Zeitangaben, Daten, Adressen usw. können als Abfolge bestimmter Tags definiert werden

## Examples

wo/PRELS es/PPER um/APPR das/ART Verhältnis/NN  
zwischen/APPR [den/ART Christdemokraten/NN] unter/APPR  
[Bundeskanzler/NN Kohl/NE] und/KON [den/ART Freien/ADJA  
Demokraten/NN] unter/APPR [Aussenminister/NN Kinkel/NE]  
ebenfalls/ADV nicht/PTKNEG allzu/PTKA rosig/ADV  
steht/VVFIN ./\$.

# Wozu Tagging?

Warum sind kategoriale Informationen (Wortarten) nützlich für die Sprachverarbeitung?

- Flache Satzverarbeitung: Suchen bestimmter sprachlicher Konstituenten, z.B. Nominalphrasen, komplexe Namen, Relativsätze, usw. mit Mustern für Tag-Abfolgen
  - schneller als vollständiges Parsing
  - reicht für viele Zwecke schon aus

# POS-Tags

- Tags sind eine Abstraktion über einzelne Worttypen und sind daher weniger selten
  - Verbesserung von Ngramm-Modellen durch Tag-Wahrscheinlichkeiten zur Milderung des Problems der Datenknappheit
  - Z.B. Modellierung des Kontextes durch Tags im SATZ-System zur Satzgrenzenerkennung (Palmer und Hearst 1998)

# POS-Tags

- Einschränkung des Suchraums beim Parsing
  - Schnellere tiefe Analyse von Sätzen durch Informationen über die syntaktischen Kategorien der einzelnen Token
- Interessant für Korpuslinguistik
  - Suche nach syntaktischen Konstruktionen möglich
    - statt Suche nach *unter Vorbehalt, unter dringendem Verdacht, unter Eid*
    - Suche nach einem Muster wie
      - APPRAJDADNN
      - Präposition Adjektiv(e) Nomen

# Gliederung

## 1 Tagging

- STTS: ein Tagset für das Deutsche
- Probleme
- Stand der Forschung

# Tagset für das Deutsche

- Das bekannteste Tagset für das Deutsche ist das **STTS** (Stuttgart-Tübingen-Tag-Set, Schiller et al. (1999))
  - [http://www.ims.uni-stuttgart.de/forschung/  
ressourcen/lexika/TagSets/stts-1999.pdf](http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-1999.pdf)

## ■ Überblickstabelle

- [https://www.ims.uni-stuttgart.de/forschung/  
ressourcen/lexika/germantagsets/#id-cfcfbf0a7-0](https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/germantagsets/#id-cfcfbf0a7-0)
- Entwicklung in den 1990ern in einer Kooperation der Universitäten Stuttgart und Tübingen
- Später Anpassungen (z.B. Universität Zürich) bzw. Erweiterungen davon (z.B. für Chat-Daten) (vgl. Zinsmeister, Heid, and Beck (Eds.) (2013))
- Genutzt sowohl in der Korpuslinguistik wie auch in der CL

## ■ Andere Tagsets für das Deutsche:

- [http://www.ims.uni-stuttgart.de/forschung/  
ressourcen/lexika/GermanTagsets.html](http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/GermanTagsets.html)

# STTS

- Unterscheidung von 11 Hauptwortarten

- |                         |                         |
|-------------------------|-------------------------|
| 1. N: Nomina            | 7. ADV: Adverbien       |
| 2. V: Verben            | 8. KO: Konjunktionen    |
| 3. ART: Artikel         | 9. AP: Adpositionen     |
| 4. ADJ: Adjektive       | 10. ITJ: Interjektionen |
| 5. P: Pronomina         | 11. PTK: Partikeln      |
| 6. CARD: Kardinalzahlen |                         |

- ... mit weiteren Unterteilungen → insgesamt 54 verschiedene Tags
- “Großes STTS”: enthält auch morphologische Tags

# STTS: Grundprinzipien

- Hierarchische Strukturierung
  - spiegelt sich in den Tagnamen wider
  - z.B. VAFIN = Verb + Auxiliar + Finit
  - kann genutzt werden für Unterspezifikation: V.\*, .\*FIN
- Disambiguierung: jede Wortform erhält genau 1 Tag
  - Tags auch für Satzzeichen, fremdsprachliches Material, Nichtwörter
  - Keine MWE (Multi-Word Expressions), d.h. Leerzeichen und Satzzeichen dienen als Tokengrenzen
- Abkürzungen: werden so getagged wie ihre entsprechende Langform
  - z. B.: z./APPRART B./NN
  - bei Zusammenschreibung Tag gemäß Funktion: z.B.: z.B./ADV
- Fehler: werden sinngemäß getagged (soweit möglich)
  - *Er hat im das gesagt.*: im/PPER

# STTS: Nomen

Tag	Beschreibung	Beispiel
NN	“normale” Nomen, inkl.: Maßangaben, Titel, Produkte, substantivierte Ausdrücke, De- terminativkomposita, Monats- namen, Sprachen, ...	<i>Haus</i> <i>Liter, Herr, Porsche, Reisen, Bachkantate, Juli, ...</i>
NE	Eigennamen, inkl.: Vor-, Familien-, Tier-, Firmen-, Orts-, Gewässernamen	<i>Hans</i> <i>Uli, Maier, Fifi, Mercedes, Stuttgart, Rhein</i>

# STTS: Adjektive

Tag	Beschreibung	Beispiel
ADJA	attributive Adjektive	[die] große [Stadt]
ADJD	prädikativ oder adverbial ge- brauchte Adjektive	[sie ist] groß, [er läuft] schneller

# STTS: Verben

Tag	Beschreibung	Beispiel
VV	“Vollverben”	<i>ankommen</i>
VA	Auxiliare	<i>haben, sein, werden</i>
VM	Modalverben	<i>können, müssen</i>

Sub-Tag	Beschreibung	Beispiel
...FIN	finit	<i>ankommt, ist, kann</i>
...INF	Infinitiv	<i>ankommen, sein, können</i>
...IZU	zu-Infinitiv	<i>anzukommen, –, –</i>
...IMP	Imperativ	<i>komm! sei! –</i>
...PP	Part.Perfekt	<i>gekommen, gewesen, gekonnt</i>

# STTS: Pronomen

Tag	Beschreibung	Beispiel
PPER	Personalpronomen	<i>ich, er</i>
PRF	Reflexivpronomen	<i>mich, sich</i>
PDS	subst. Dem.pron	<i>dieser [kommt]</i>
PDAT	attr. Dem.pron	<i>dieser [Mann kommt]</i>
PRELS	subst. Rel.pron	<i>[der Mann,] der [kam]</i>
PRELAT	attr. Rel.pron	<i>[der Mann,] dessen [Hund kam]</i>

Die S/AT-Unterscheidung (substituierend/attribuierend) betrifft:  
PD (demonstrativ), PI (indefinit), PPOS (possessiv), PREL  
(relativ), PW (interrogativ)

# STTS: Sonstiges (Auswahl)

Tag	Beschreibung	Beispiel
ART	Artikel	<i>der, ein</i>
APPR	Präposition	<i>wegen [der Leute]</i>
APPO	Postposition	<i>[der Leute] wegen</i>
APZR	Zirkumposition (rechts)	<i>[um der Leute] willen</i>
APPRART	Präposition+Artikel	<i>beim, zur</i>
ADV	Adverb	<i>gestern</i>
PADV	Pronominaladverb	<i>damit</i>
WADV	Interr.adverb	<i>wann</i>
PWAV	Interr.pron.adverb	<i>womit</i>
KON	koordinierende Konjunktion	<i>und, oder, denn</i>
KOUS	subordinierende Konjunktion	<i>als, seitdem, weil</i>
KOUI	subord. Konjunktion mit Infinitiv	<i>um (zu), ohne (zu)</i>
\$.	satzfinale Satzzeichen	<i>. ! ? : ;</i>
\$,	Komma	<i>,</i>
\$()	sonstige Satzzeichen; satzintern	<i>- [ ] ( )</i>

# STTS: Ein kleiner Probetext

*Das ist ein Übungstext zur Wortarten-Bestimmung. Der Inhalt des Textes ist recht einfach, den sollte jeder problemlos verstehen können. Eine Frage in der Linguistik ist, welche Wortarten es überhaupt gibt. Diese Frage, die seit Jahrhunderten viele umtreibt, sieht also wie folgt aus: Wie viele verschiedene Wortarten hat das Deutsche oder andere Sprachen dieser Welt? Vermutlich sind es unter 20 Wortarten.*

# STTS: Ein kleiner Probetext

Das/PDS ist/VAFIN ein/ART Übungstext/NN zur/APPRART Wortarten-Bestimmung/NN ./. Der/ART Inhalt/NN des/ART Textes/NN ist/VAFIN recht/ADV einfach/ADJD, ,/\$, den/PDS sollte/VMFIN jeder/PIS problemlos/ADJD verstehen/VVINF können/VMINF ./. Eine/ART Frage/NN in/APPR der/ART Linguistik/NN ist/VAFIN ,/\$, welche/PWAT Wortarten/NN es/PPER überhaupt/ADV gibt/VVFIN ./. Diese/PDAT Frage/NN ,/\$, die/PRELS seit/APPR Jahrhunderten/NN viele/PIS umtreibt/VVFIN ,/\$, sieht/VVFIN also/ADV wie/PWAV folgt/VVFIN aus/PTKVZ :\$. Wie/WAV viele/PIAT verschiedene/ADJA Wortarten/NN hat/VAFIN das/ART Deutsche/NN oder/KON andere/ADJA Sprachen/NN dieser/PDAT Welt/NN ?/\$. Vermutlich/ADV sind/VAFIN es/PPER unter/ADV 20/CARD Wortarten/NN ./\$.

# Vergleich: STTS, BNC und Penn (Auswahl)

## ■ STTS

- Nomina: NN oder NE
- Adjektive/Adverbien: ADJA oder ADJD / ADV
- Verben: VVFIN, VVINF, VVIMP, VVPP, ... (VA..., VM...)
- Pronomina: PPER, PPOSS, PRF, ...

## ■ BNC ("basic tagset", "C5")

- Nomina: NN0, NN1, NN2, NP0
- Adjektive/Adverbien: AJ0, AJC, AJS / AV0
- Verben: VBB, VBD, VBG, VBI, VBN, VBZ,  
... (VD/VH/VM/VV...)
- Pronomina: PNP, POS, PNX, ...

## ■ PENN

- Nomina: NN, NNS, NNP, NNPS
- Adjektive/Adverbien: JJ, JJR, JJS / RB, RBR, RBS
- Verben: VB, VBD, VBG, VBN, VBP, VBZ, MD
- Pronomina: PRP, PRP\$, ...

# Gliederung

## 1 Tagging

- STTS: ein Tagset für das Deutsche
- Probleme
- Stand der Forschung

# Herausforderungen beim Taggen

## Disambiguierung ambiger Wörter

- Viele Wortformen (insbesondere häufige Funktionswörter) sind ambig, d.h. sie können je nach Kontext zu verschiedenen Wortarten gehören

## Examples

- 1
  - a. auch/ADV wenn/KOUS der/ART Autor/NN sie/PPER nicht/PTKNEG so/ADV gemeint/VVPP hat/VAFIN
  - b. Ihre/PPOSAT Stärke/NN ,/, mit/APPR der/PRELS sie/PPER bisher/ADV politisch/ADV überlebt/VVPP haben/VAFIN
- 2
  - a. Das/PDS ist/VAFIN sehr/ADV modern/ADJD
  - b. Sie/PPER modern/VVFIN vor/APPR sich/PRF hin/PTKVZ

# Disambiguierung

- Woran sieht man, dass *der* in (1a) ein Artikel und in (1b) ein Relativpronomen ist?
- Am Kontext
  - In (1) geht *der* schon eine satzeinleitende Konjunktion (*wenn*) voraus (also kein Rel.pron) und es folgt ein Nomen (also eher Artikel)
  - In (2) folgt auf *der* ein Personalpronomen (*sie*) und davor steht ein Komma

# Unbekannte Wörter

- Der Tagger hat das korrekte Tag für unbekannte Wörter nicht im Lexikon gespeichert
  - Ich/PPER habe/VAFIN diese/PDAT Stoiberisierung/?? des/ART Unionswahlkampfes/?? obersatt/?? ./\$.
- Wie könnte man dieses Problem angehen?
  - Zuweisung des häufigsten Tags
  - Tag wird auf Grund des Kontextes bestimmt:
    - NN nach Demonstrativpronomen (PDAT) oder Artikel (ART)
  - Morphologische Analyse: *-ung* ist ein Nominalsuffix
  - Orientierung an Orthographie
    - Satzinterne Großschreibung → NN

# Wortartenambiguität

In einem kleinen Korpus aus Artikeln der Neuen Zürcher Zeitung hatten

- 7 Worttypen 3 verschiedene mögliche Tags
  - *als, das, dessen, statt, war, wie, zu*
  - Bsp. *zu*: PTKZU: 31; APPR: 10; PTKA: 2
- 46 Worttypen 2 verschiedene mögliche Tags
- Und die restlichen 1.810 Worttypen waren nicht ambig

# Wortartenambiguität

- Pro Token gibt es in diesem Korpus durchschnittlich 1.29 mögliche Tags
- Die Mehrzahl der Token ist nicht ambig
- **Baseline**
  - Wenn man jedem Token den Tag zuweist, mit dem es am häufigsten vorkommt, erhält man *bei bekanntem Vokabular* eine Fehlerrate von ca. 4%
  - Die Baseline-Fehlerrate steigt durch das Vorkommen unbekannter Wörter noch erheblich an

# Gliederung

## 1 Tagging

- STTS: ein Tagset für das Deutsche
- Probleme
- Stand der Forschung

# Universelles Tagset

- Vorschlag von Petrov, Das, and McDonald (2011): "Universelles Tagset" (auch: "Google tagset")
- 11 Tags: NOUN (nouns), VERB (verbs), ADJ (adjectives), ADV (adverbs), PRON (pronouns), DET (determiners and articles), ADP (prepositions and postpositions), NUM (numerals), CONJ (conjunctions), PRT (particles), ‘ ’ (punctuation marks), X (Rest, z.B. Abkürzungen, fremdsprachliche Ausdrücke)
- Mappings von existierenden Tagsets auf das Universelle Tagset
- Ziel: bessere Vergleichbarkeit von Ergebnissen verschiedener Systeme und Sprachen

# Universelles Tagset: Performanzvergleich

Daten	#Tags	O/O	U/U	O/U
Englisch (PennTreebank)	45	96.7	96.8	97.7
Deutsch (TIGER)	54	97.9	98.1	98.8
Deutsch (Negra)	54	96.9	97.9	98.6

- Tagger: TnT (HMM)
- #Tags: Originale Tagset-Größe
- Accuracies für Training/Evaluation auf O(riginalem) bzw.  
U(niversellem) Tagset

# Universelles Tagset: Performanzvergleich

Language	Source	# Tags	O/O	U/U	O/U
Arabic	PADT/CoNLL07 (Hajič et al., 2004)	21	96.1	96.9	97.0
Basque	Basque3LB/CoNLL07 (Aduriz et al., 2003)	64	89.3	93.7	93.7
Bulgarian	BTB/CoNLL06 (Simov et al., 2002)	54	95.7	97.5	97.8
Catalan	CESS-ECE/CoNLL07 (Martí et al., 2007)	54	98.5	98.2	98.8
Chinese	Penn ChineseTreebank 6.0 (Palmer et al., 2007)	34	91.7	93.4	94.1
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294	87.5	91.8	92.6
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	63	99.1	99.1	99.1
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25	96.2	96.4	96.9
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	12	93.0	95.0	95.0
English	PennTreebank (Marcus et al., 1993)	45	96.7	96.8	97.7
French	FrenchTreebank (Abeillé et al., 2003)	30	96.6	96.7	97.3
German	Tiger/CoNLL06 (Brants et al., 2002)	54	97.9	98.1	98.8
German	Negra (Skut et al., 1997)	54	96.9	97.9	98.6
Greek	GDT/CoNLL07 (Prokopidis et al., 2005)	38	97.2	97.5	97.8
Hungarian	Szeged/CoNLL07 (Cséndes et al., 2005)	43	94.5	95.6	95.8
Italian	ISST/CoNLL07 (Montemagni et al., 2003)	28	94.9	95.8	95.8
Japanese	Verbmobil/CoNLL06 (Kawata and Bartels, 2000)	80	98.3	98.0	99.1
Japanese	Kyoto4.0 (Kurohashi and Nagao, 1997)	42	97.4	98.7	99.3
Korean	Sejong ( <a href="http://www.sejong.or.kr">http://www.sejong.or.kr</a> )	187	96.5	97.5	98.4
Portuguese	Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)	22	96.9	96.8	97.4
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11	96.8	96.8	96.8
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	29	94.7	94.6	95.3
<small>Spanish</small>	<small>Álvarez-Gutiérrez et al. (2004)</small>	<small>47</small>	<small>96.3</small>	<small>96.3</small>	<small>96.0</small>

# Performanz auf anderen Daten

Giesbrecht and Evert (2009) vergleichen die Performanz verschiedener Tagger

- mit “Standard-Daten” (d.h. Zeitungstext)

	TreeTagger	Stanford	UIMA	TnT	SVM
total accuracy (%)	96.89±0.34	<b>97.63±0.24</b>	96.04±0.38	96.92±0.31	97.12±0.20
known words (%)	97.62±0.21	—	97.50±0.18	97.59±0.20	97.71±0.17
unknown words (%)	87.89±0.99	<b>91.66±0.83</b>	79.59±1.30	89.16±0.85	90.16±0.84
% of unknown words	7.44±0.78	7.52±0.46	8.10±0.71	7.85±0.88	7.82±0.82

- vs. Internet-Daten (Web-Korpus)

	TT-SPF <sup>a)</sup>	TT <sup>b)</sup>	Stanford	UIMA	TnT	SVM
total accuracy (%)	<b>93.71</b>	90.78	92.61	91.68	<b>92.69</b>	92.36
known words (%)	95.42	93.59	—	95.59	95.90	95.91
unknown words (%)	54.30	69.12	<b>75.35</b>	66.49	71.99	69.45
% of unknown words	4.15	11.48	13.00	13.43	13.44	13.43

<sup>a)</sup>TreeTagger with standard parameter file included in distribution

<sup>b)</sup>TreeTagger with parameter file trained on the TIGER treebank

# Performanz auf verschiedenen Genres (Giesbrecht and Evert 2009)

Genre	TT-SPF <sup>a)</sup>	TT <sup>b)</sup>	TnT	Stanford	SVM	UIMA
1. <i>TV episode guide</i>	<b>93.89</b>	90.87	92.79	92.83	92.78	89.91
2. news report (medicine)	<b>96.88</b>	97.12	95.92	96.16	95.68	94.26
3. political speech	<b>97.52</b>	96.56	96.42	96.15	93.81	95.61
4. job market news	<b>97.46</b>	93.65	96.19	96.95	95.18	95.44
5. story (Paul of Thebes)	<b>95.42</b>	94.84	95.08	95.37	95.08	93.87
6. exposition programme	<b>94.23</b>	92.13	92.83	92.66	93.01	90.75
7. <i>online forum</i>	<b>88.01</b>	79.97	85.56	84.47	84.51	84.47
8. report on infections	<b>98.25</b>	96.89	97.28	<b>98.25</b>	97.08	95.54
9. <i>conference information</i>	90.98	<b>89.18</b>	<b>92.01</b>	90.98	<b>93.30</b>	92.55
10. IT news (CeBIT)	93.69	92.73	92.93	94.07	94.07	<b>95.42</b>
11. info (support programme)	97.10	98.51	98.01	<b>99.50</b>	97.01	98.02
12. <i>news report (archbishop)</i>	91.97	<b>87.15</b>	91.97	91.97	<b>93.97</b>	90.80
13. synopsis of cold war	96.67	94.86	96.49	95.68	95.40	<b>97.30</b>
	94.77 ±3.04	92.65 ±5.04	94.11 ±3.31	94.23 ±3.85	93.91 ±3.15	93.38 ±3.67

# Englisch: State of the art

Stanford Tagger 2.0	Maximum entropy cyclic dependency network	Manning (2011)	Stanford Tagger <a href="#">♂</a>	Yes	97.32%	90.79%
LTAG-spinal	Bidirectional perceptron learning	Shen et al. (2007)	LTAG-spinal <a href="#">♂</a>	No	97.33%	Not available
SCCN	Semi-supervised condensed nearest neighbor	Søgaard (2011)	SCCN <a href="#">♂</a>	Yes	97.50%	Not available
structReg	CRFs with structure regularization	Sun(2014)	Not available	No	97.36%	Not available
BI-LSTM-CRF	Bidirectional LSTM-CRF Model	Huang et al. (2015)	Not available	No	97.55%	Not available

- Trainingsdaten: PennTreebank/WSJ
- Spalten: System; Modelltyp, Publikation, Software-Link, zusätzliche Trainingsdaten notwendig?; Genauigkeit auf allen Tokens/Unknowns
- Quelle: [http://aclweb.org/aclwiki/index.php?title=POS\\_Tagger:\\_%28State\\_of\\_the\\_art%29](http://aclweb.org/aclwiki/index.php?title=POS_Tagger:_%28State_of_the_art%29)

# Wie weiter? (Manning 2011)

- Aktuell: Genauigkeit pro Token: > 97%
- Problem aber: Genauigkeit pro Satz: 55-56%!
  - ebenso problematisch: andere Domäne, Epoche, ...
- Laut Doku von PennTreebank: Annotationsfehler > 3%
- Manuelle Tagging-Fehleranalyse (100 Instanzen):

Class	Frequency
1. Lexicon gap	4.5%
2. Unknown word	4.5%
3. Could plausibly get right	16.0%
4. Difficult linguistics	19.5%
5. Underspecified/unclear	12.0%
6. Inconsistent/no standard	28.0%
7. Gold standard wrong	15.5%

# Fehlertypen (Manning 2011)

- 1 Lexicon gap (4.5%): Wort kam nur mit anderen Tags in den Trainingsdaten vor
- 2 Unknown word (4.5%): Wort kam gar nicht in Trainingsdaten vor
- 3 Could plausibly get right (16%): vermeidbarer Fehler (Tagger sollte das eigentlich können)
- 4 Difficult linguistics (19.5%): größerer Kontext für Disambiguierung nötig
- 5 Underspecified/unclear (12.0%): linguistisch schwierige Unterscheidung
- 6 Gold standard inconsistent (28.0%): keine eindeutige Regel in Annotationsguidelines, daher inkonsistent annotiert in den Trainingsdaten
- 7 Gold standard wrong (15.5%): Annotationsfehler

→ Guidelines müssen verbessert werden!

# Evaluation (Manning 2011)

Vergleich Training/Evaluation auf originalem vs. korrigierten Daten

Model	Corrected	Corrected	#	Sent.	Token	Unk.
	Train	Test	Feats	Acc.	Acc.	Acc.
5WSHAPESDS	no	no	737,955	56.79%	97.28%	90.46%
	no	yes		57.95%	97.38%	90.60%
	yes	no	735,679	55.87%	97.21%	90.58%
	yes	yes		62.66%	97.75%	90.75%

# Zusammenfassung

- POS-Tags: kodieren nicht nur Wortart, sondern weitere Eigenschaften (Distribution, Morphologie)
- Deutsches Standardtagset STTS
- Warum ist es wichtig, das Tagset und die Definitionen der Tags genauer zu kennen?
  - weil es bei Taggingfehlern interessant und wichtig ist, sich die Art der Fehler anzuschauen: Liegt es am ungeeigneten Modell? Liegt es an ungeeigneten Tagset-Definitionen? Etc.
  - ... und entsprechend sollte man entweder am Modell oder am Tagset "schrauben"

# References I

Linguistische Grundlagen: Abschnitt 3.1 aus  
Manning and Schütze (1999)

Giesbrecht, E. and S. Evert (2009).

Is part-of-speech tagging a solved task? an evaluation of pos  
taggers for the German Web as Corpus.

In *Proceedings of the Fifth Web as Corpus Workshop (WAC5)*, pp.  
27–35.

Manning, C. D. (2011).

Part-of-speech tagging from 97% to 100%: Is it time for some  
linguistics?

In A. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text  
Processing, 12th International Conference, CICLing 2011,  
Proceedings, Part I*, Number 6608 in Lecture Notes in  
Computer Science, pp. 171–189.



## References II

- Manning, C. D. and H. Schütze (1999).  
*Foundations of Statistical Natural Language Processing*.  
Cambridge, MA: The MIT Press.
- Petrov, S., D. Das, and R. McDonald (2011).  
A universal part-of-speech tagset.  
In *Proceedings of LREC-2011*.
- Schiller, A., S. Teufel, C. Stöckert, and C. Thielen (1999).  
Guidelines für das Tagging deutscher Textcorpora mit STTS  
(Kleines und großes Tagset).  
Technical report, Universitäten Stuttgart und Tübingen, <http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-1999.pdf>.

## References III

Zinsmeister, H., U. Heid, and K. Beck (Eds.) (2013).  
Das Stuttgart-Tübingen Tagset – Stand und Perspektiven.

# Symbolische und Statistische Verfahren (CL2)

## 7: Hidden-Markov-Modelle

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum

# Themenüberblick Tagging

- Letzte Sitzung: Wortarten-Tagging
- Heute: **Hidden-Markov-Modell-Tagger**
- Nächste Sitzung
  - Brill-Tagger
  - Vergleich HMM-Tagger vs. Brill-Tagger
  - Evaluation

# Themenüberblick heute

- Generelle Idee eines Hidden-Markov-Modells (HMM)
- Auch: Verwendung zum Generieren und als Sprachmodell
- Eigentlich: Verwendung als Tagger
  - Training
    - Glättung/Smoothing
    - Behandlung unbekannter Wörter
  - Dekodierung = Tagging
    - Viterbi-Algorithmus

# Gliederung

## 1 Hidden Markov Models

## 2 Viterbi-Algorithmus

# (Hidden-)Markov-Modell-Tagging

- Ein verbreiteter probabilistischer Ansatz für das POS-Tagging basiert auf **Markov-Modellen** (Ngramm-Modellen)
- Evidenz aus dem Kontext wird durch **Übergangswahrscheinlichkeiten zwischen Tags** modelliert
  - Bsp: nach einem Komma ist ein Relativpronomen (PRELS) wahrscheinlicher als ein definiter Artikel (ARTDEF):  
 $P(PRELS|\$, ,) = 0.19 > P(ARTDEF|\$, ,) = 0.072$
- Wie bei den Sprachmodellen: auch Modelle höherer Ordnung (z.B. Trigramm-Modelle)
  - Bsp: ein finites Verb ist wahrscheinlicher als ein Infinitiv nach der Tag-Folge ARTDEF, NN (warum?)  
 $P(VVFIN|ARTDEF, NN) = 0.068 > P(VVINF|ARTDEF, NN) = 0.036$

Hinweis: Hier (und im Folgenden) kommen Tags wie ARTDEF vor, die nicht Teil des originalen STTS sind

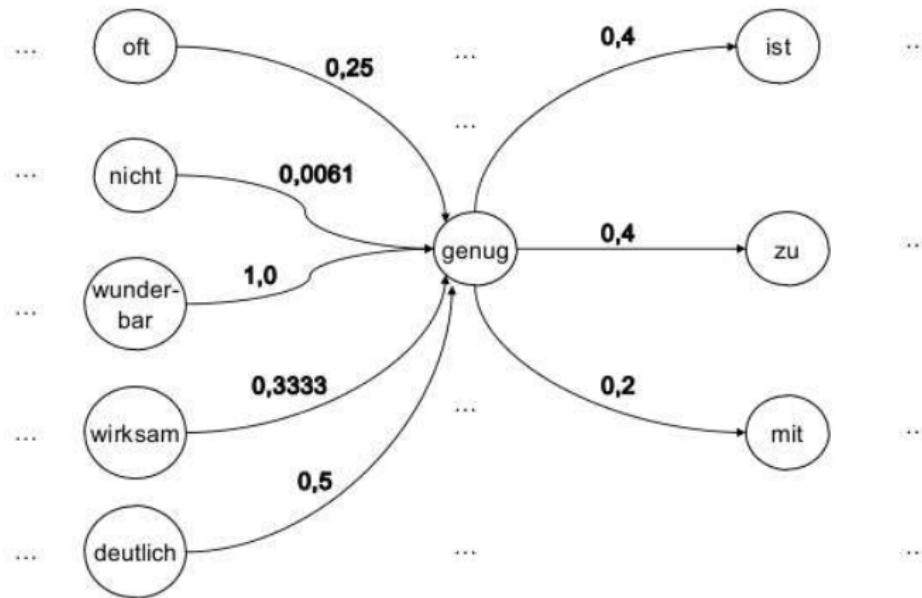
# Ngramm-Modell (einfaches Markov-Modell) als Sprachmodell

Zunächst nochmal zurück zum Sprachmodell in Form eines (einfachen, d.h. nicht “hidden”) Markov-Modells

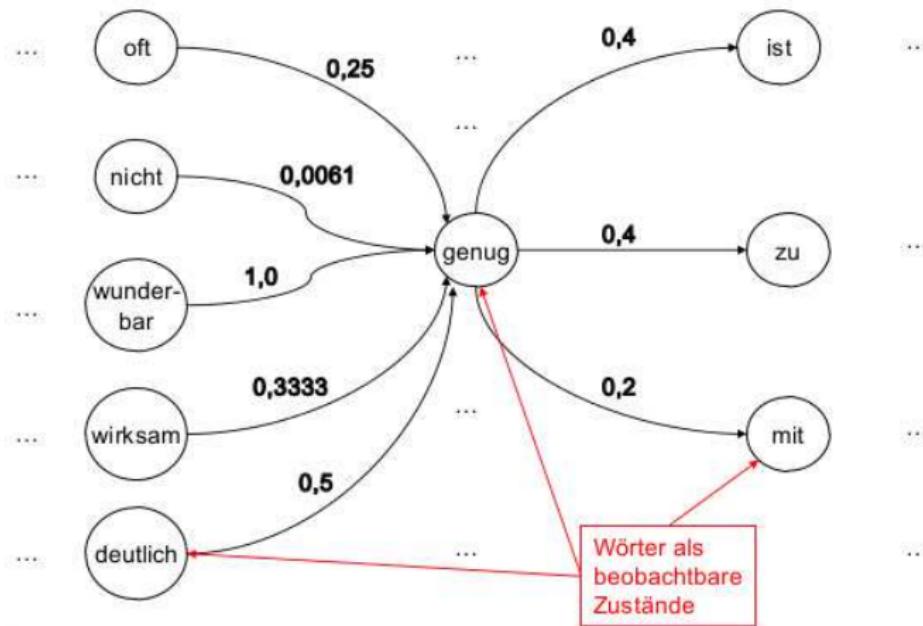
# (Hidden-)Markov-Modell-Tagging

- Notation
  - $X_i$ : Beobachtung an Position  $i$
  - $t^j$ :  $j$ -tes Tag aus dem Tagset
- Eigenschaften von Markov-Modellen (Wiederholung)
  - beschränkte History (**limited horizon**):  
 $P(X_{i+1} = t^j | X_1, \dots, X_i) = P(X_{i+1} = t^j | X_i)$
  - zeitliche Invarianz (**Stationarität**):  
 $P(X_{i+1} = t^j | X_i = t^i) = P(X_2 = t^j | X_1 = t^i)$

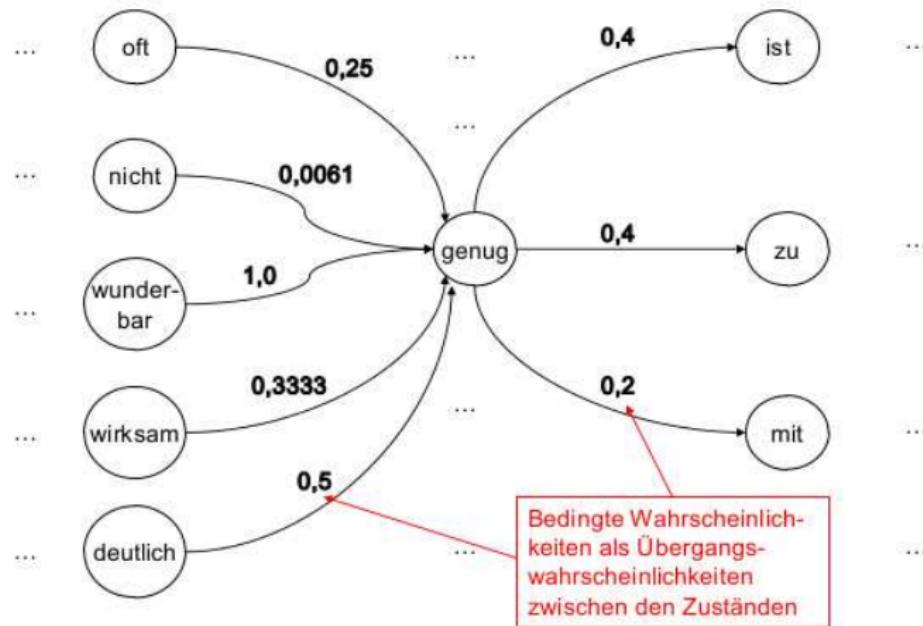
# Ngramm-Modell (einfaches Markov-Modell) als Sprachmodell



# Ngramm-Modell (einfaches Markov-Modell) als Sprachmodell



# Ngramm-Modell (einfaches Markov-Modell) als Sprachmodell



# Nicht-beobachtbare Kategorien

Annahme: Die tatsächlich beobachtete Abfolge von Wörtern wird durch eine zugrunde liegende Abfolge von nicht-beobachtbaren Kategorien (Tags) "generiert" (emittiert)

Die Ratifizierung im Senat

↑ ↑ ↑ ↑

ARTDEF NN APPRART NN

dürfte in den nächsten Tagen

↑ ↑ ↑ ↑ ↑

VMFIN APPR ARTDEF ADJA NN

problemlos über die Bühne gehen .

↑ ↑ ↑ ↑ ↑ ↑

ADJD APPR ARTDEF NN VVINF \$.

# Notationskonventionen (nach Charniak et al. 1993)

$w_i, t_i$	Wort / Tag an Position $i$ im Korpus (Tokens)
$w_{i,i+m}, t_{i,i+m}$	Wort-/Tagsequenz von Position $i$ bis $i+m$ im Korpus (Tokens)
$w^i, t^i$	das $i$ -te Wort im Lexikon / Tag im Tagset (Types)
$C(w^i), C(t^i)$	Anzahl der Vorkommen von $w^i, t^i$
$W, T$	Anzahl Wörter im Lexikon / Tags im Tagset ( $W$ entspricht $V$ , dem Vokabular)

# Nicht-beobachtbare Kategorien

- Die Kategorien (Tags) bilden die “versteckten” (= **hidden**) Zustände des Markovmodells
  - $t_1 \dots t_n$   
= Tag von  $w_1 \dots$  Tag von  $w_n$   
(= Tags des ersten bis letzten Wortes im Satz)
- Jeder Zustand (jede Kategorie) emittiert mit einer bestimmten Wahrscheinlichkeit ein spezifisches Wort
  - $P(w^1|t^1)$   
= Wahrscheinlichkeit des ersten Wortes im Lexikon,  
gegeben das erste Tag im Tagset
  - Beispiel:  $P(\text{aber}|\text{ADV}) = 0.0076$

# HMM für das Wortarten-Tagging

- Eigentlich unintuitiv: wir möchten ja die wahrscheinlichste Sequenz von Tags  $t_{1,n}$  für eine gegebene Folge  $w_{1,n}$  von Token (Wörtern) ermitteln:

$$t_{1,n}^* = \operatorname{argmax}_{t_{1,n}} P(t_{1,n} | w_{1,n})$$

- Dank Umstellung nach dem Satz von Bayes:

$$t_{1,n}^* = \operatorname{argmax}_{t_{1,n}} P(t_{1,n} | w_{1,n}) = \operatorname{argmax}_{t_{1,n}} \frac{P(w_{1,n} | t_{1,n}) \cdot P(t_{1,n})}{P(w_{1,n})}$$

- Vereinfachung:  $t_{1,n}^* = \operatorname{argmax}_{t_{1,n}} P(w_{1,n} | t_{1,n}) \cdot P(t_{1,n})$ 
  - Sequenz von Tokens  $w_{1,n}$  ist gegeben
  - ihre Wahrscheinlichkeit  $P(w_{1,n})$  bleibt für alle möglichen Tag-Sequenzen gleich und kann daher ignoriert werden

# HMM für das Wortarten-Tagging

- Endgültige Form

$$t_{1,n}^* = \operatorname{argmax}_{t_{1,n}} P(w_{1,n}|t_{1,n}) \cdot P(t_{1,n})$$

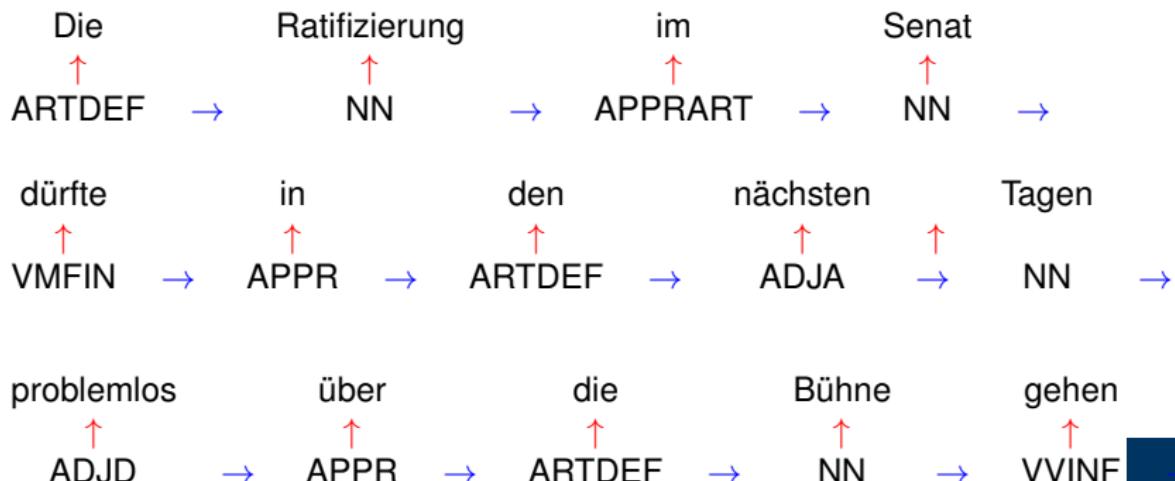
- Lexikalisches Modell
- Kontextmodell

- Vorteile

- Lexikalisches Modell und Kontextmodell können sauber getrennt werden
- Modellierung als Hidden-Markov-Modell möglich (ein Standard-Modell aus der Stochastik)

# Übergangswahrscheinlichkeiten

Annahme: Die syntagmatischen Beziehungen werden durch Übergangswahrscheinlichkeiten zwischen den zugrunde liegenden Zuständen, den Tags, modelliert

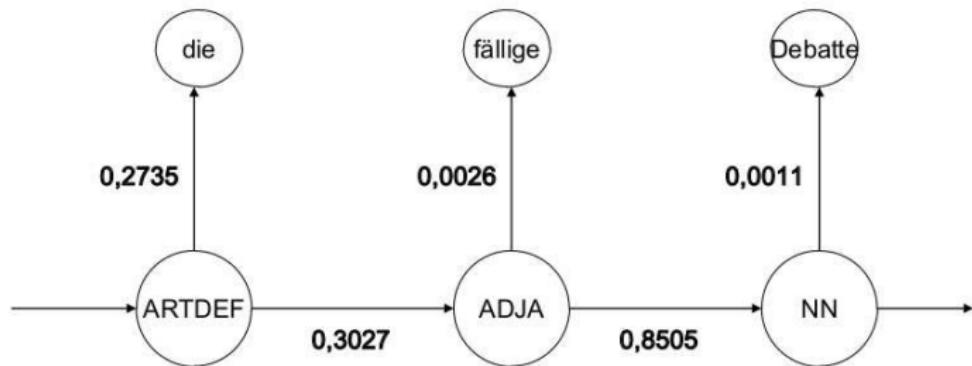


# Wahrscheinlichkeit einer Tag-Sequenz

- Wahrscheinlichkeit einer Sequenz von Tags
  - $P(t_{1,n})$
- ... berechnet als Produkt der bedingten Wahrscheinlichkeiten für die einzelnen Tags
  - $P(t_{1,n}) = P(t_1) \cdot P(t_2|t_1) \cdot P(t_3|t_1,2) \cdot \dots \cdot P(t_n|t_{1,n-1})$
- Markov-Annahme
  - der relevante Kontext kann auf  $n$  vorhergehende Tags eingeschränkt werden (Ngramm-Modell)
  - Beispiel Bigramm-Modell
$$P(t_{1,n}) = P(t_1|START) \cdot P(t_2|t_1) \cdot P(t_3|t_2) \cdot \dots \cdot P(t_n|t_{n-1})$$

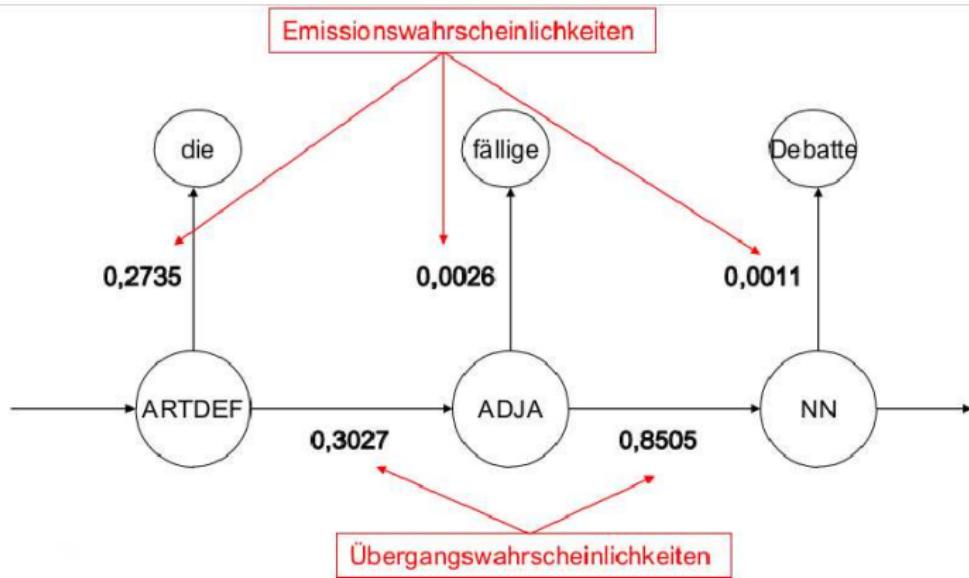
# Beispiel

## Bigramm-HMM (HMM 1. Ordnung)



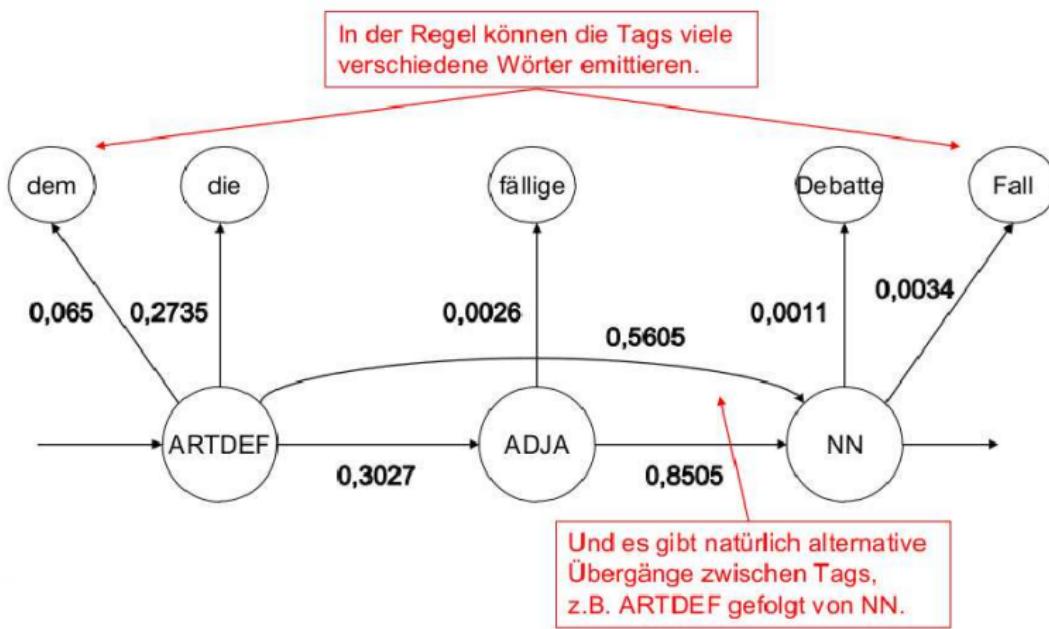
# Beispiel

## Bigramm-HMM (HMM 1. Ordnung)



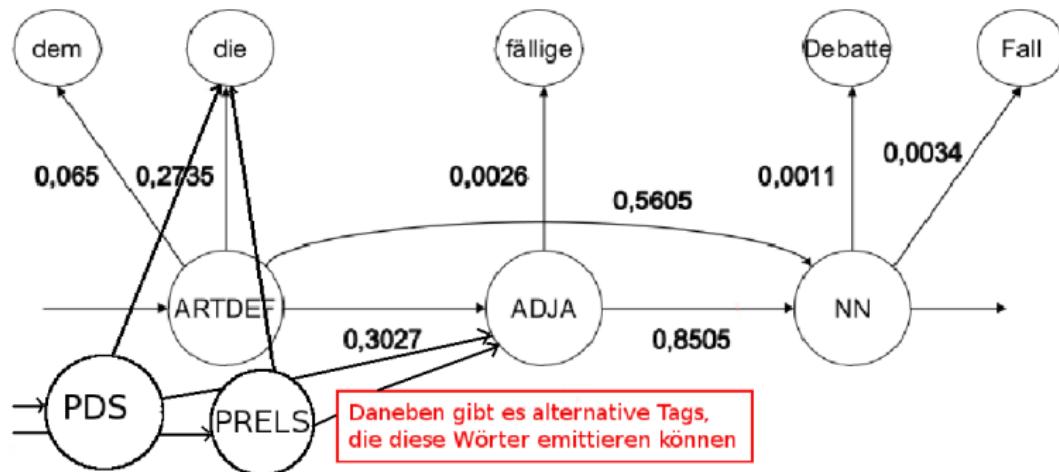
# Beispiel

## Bigramm-HMM (HMM 1. Ordnung)



# Beispiel

## Bigramm-HMM (HMM 1. Ordnung)



# Vereinfachende Unabhängigkeitsannahmen

- In der Regel werden beim HMM-Tagging folgende Unabhängigkeitsannahmen gemacht:
  - 1 **Tags:** der relevante Kontext ist auf die  $n$  vorhergehenden Tags beschränkt, z.B.:  $P(t_{i+1}|t_{1..i}) \approx P(t_{i+1}|t_i)$  (**Markov-Annahme**)
  - 2 **Tags:** die Kontextwahrscheinlichkeit eines Tags ist nur von den vorhergehenden *Tags* abhängig, nicht von den vorhergehenden *Tokens*:  $P(t_{i+1}|t_{1..i}, w_{1..i}) \approx P(t_{i+1}|t_{1..i})$
  - 3 **Tokens:** die Emissionswahrscheinlichkeit für ein bestimmtes Token hängt nur vom gegenwärtigen Zustand ab, in dem sich das Modell befindet (also vom aktuellen Tag), sie ist also kontextunabhängig:  
 $P(w_i|t_{1..i}, w_{1..w_{i-1}}) \approx P(w_i|t_i)$
- Diese Annahmen vereinfachen das Modell stark, sind aber natürlich nicht immer wohlbegründet . . .

# HMM für das Wortarten-Tagging

Definition: Ein HMM für POS-Tagging ist ein Tupel  $\{T, W, \Pi, A, B\}$  mit

- Menge von **Zuständen**  $T = \{t^1, \dots, t^T\}$  (Tagset)
- Menge von **Ausgabesymbolen**  $W = \{w^1, \dots, w^W\}$  (Wörter)
- **Startwahrscheinlichkeiten**  $\Pi = \{\pi^i\}, i \in T$
- **Übergangswahrscheinlichkeiten**  $A = \{a_{ij}\}, i, j \in T$
- **Emissionswahrscheinlichkeiten**  $B = \{b_{jk}\}, j \in T, k \in W$

# HMM für das Wortarten-Tagging

**Startwahrscheinlichkeiten**  $\Pi = \{\pi^i\}, i \in T$

- Mit welchen Tags kann eine Sequenz mit welcher Wahrscheinlichkeit beginnen?
- $\pi^i$ : Wahrscheinlichkeit, dass das Markov-Modell mit Zustand  $i$  starten wird, d.h.  $p(t^i | START)$
- Falls  $\pi^j = 0$ :  $j$  ist kein möglicher Startzustand
- $\sum_{i=1}^n \pi^i = 1$
- Alternative: Spezieller Startzustand  $s$  mit der Startwahrscheinlichkeit 1

# HMM für das Wortarten-Tagging

**Übergangswahrscheinlichkeiten**  $A = \{a_{ij}\}, i, j \in T$

- $a_{ij}$ : Wahrscheinlichkeit, dass von Zustand  $t_i$  in den Zustand  $t_j$  übergegangen wird:  $p(t_j|t_i)$
- Für alle  $i$ :  $\sum_{j=1}^n a_{ij} = 1$  mit  $n$  = Anzahl der Tags

**Emissionswahrscheinlichkeiten**  $B = \{b_{jk}\}, j \in T, k \in W$

- Wahrscheinlichkeit, dass im Zustand  $t_j$  das Wort  $w_k$  emittiert (d.h. beobachtet) wird:  $p(w_k|t_j)$
- Für alle  $j$ :  $\sum_{k=1}^m b_{jk} = 1$

# Inwiefern sind die Zustände eines HMM-Taggers verborgen?

Time flies like an arrow .  
NN VBZ CS AT NN .

'Zeit verfliegt wie ein Pfeil.'

Time flies like bananas .  
NN NN VB NN .

'Zeitfliegen mögen Bananen.'

- Eine Wortsequenz kann durch mehrere zugrunde liegende Tagsequenzen erzeugt werden
- Der Tagger kann sich z.B. bei der Erzeugung von *like* im Zustand CS (Konjunktion) oder VB (Verb) befunden haben
  - Vorschau: Wahrscheinlichkeit einer Token-Sequenz als *Summe* aller möglichen Pfade durch das Modell (s.u.)

# Weitere Anwendungen eines HMM

Außer zum Taggen kann ein HMM auch eingesetzt werden als:

- 1** Sprachmodell
- 2** Generator

# 1. Sprachmodell

- **Wahrscheinlichkeit einer Sequenz:** Gegeben ein HMM  $\mu$  und eine Wortsequenz  $w_{1,n}$  von emittierten Tokens, bestimme  $P(w_{1,n}|\mu)$

$$P(w_{1,n}|\mu) = \sum_{t_{1,n}} \prod_{i=1}^n a_{ii+1} b_{iw_i}$$

- **Summe der Wahrscheinlichkeiten**, mit der alle möglichen Tag-Sequenzen  $t_{1,n}$  die Token-Sequenz  $w_{1,n}$  erzeugen

- $\sum_{t_{1,n}}$ : alle möglichen Tagsequenzen!

## 2. Zufälliges Generieren von Text mit einem HMM

- Starte im Startzustand (oder wähle zufällig einen Anfangszustand mit Hilfe von  $\Pi$ )
- Endlosschleife
  - wähle zufällig den nächsten Zustand  $t$ 
    - wobei die Wahrscheinlichkeit, mit der ein Tag  $t^j$  gewählt wird, die Übergangswahrscheinlichkeiten  $a_{ij}$  des HMMs berücksichtigen muss
  - emittiere zufällig ein Symbol  $w$ 
    - wobei die Wahrscheinlichkeit, mit der ein bestimmtes Symbol  $w^k$  gewählt wird, die Emissionswahrscheinlichkeiten  $b_{jk}$  für den gegenwärtigen Zustand berücksichtigen muss

# Trainieren und Anwenden eines HMM

- 1 Training:** Gegeben eine Wortsequenz  $w_{1,n}$  und eine Sequenz von Zuständen  $t_{1,n}$ , lerne die Übergangs- und Emissionswahrscheinlichkeiten  $A$  und  $B$
  - 2 Decoding:** Gegeben ein HMM  $\mu$  und eine Wortsequenz  $w_{1,n}$ , bestimme die wahrscheinlichste Sequenz verborgener Zustände  $\text{argmax}_{t_{1,n}} P(t_{1,n} | w_{1,n}, \mu)$
- Im Folgenden näher angeschaut

# 1. Training

## ■ Schätzung der Parameter durch überwachtes Training

- Lernen der Wahrscheinlichkeiten aus einem Trainingskorpus
- Übergangswahrscheinlichkeiten (MLE)

$$P(t^k | t^i) = \frac{C(t^i, t^k)}{C(t^i)}$$

- Emissionswahrscheinlichkeiten (MLE)

$$P(w^k | t^i) = \frac{C(w^k, t^i)}{C(t^i)}$$

- Smoothing

- vor allem die Emissionswahrscheinlichkeiten sollten geglättet werden (unbekannte Wörter)

## 2. Unüberwachtes Lernen

- Es gibt auch Methoden, um nur mit einem Lexikon, das alle möglichen Tags für die enthaltenen Wörter angibt, und einem unannotierten Trainingskorpus einen HMM-Tagger zu trainieren
- Forward-Backward-Algorithmus (Expectation Maximization / Baum-Welch)
  - Manning and Schütze (1999, S. 333ff)
  - Jurafsky and Martin (2009, S. 220-226)
  - Tutorial von Rabiner: <http://www.cs.ubc.ca/~murphyk/Software/HMM/rabiner.pdf>
- Intuition: viele Worttypen sind nicht ambig und schränken daher die möglichen Parameter des Modells ein

## 2. Decoding = Tagging

Die **wahrscheinlichste Sequenz von Tags** gegeben eine Folge von Token bestimmen

$$\begin{aligned} t_{1,n}^* &= \operatorname{argmax}_{t_{1,n}} P(t_{1,n} | w_{1,n}) \\ &= \operatorname{argmax}_{t_{1,n}} P(w_{1,n} | t_{1,n}) P(t_{1,n}) \\ &= \operatorname{argmax}_{t_{1,n}} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1}) \end{aligned}$$

# Gliederung

## 1 Hidden Markov Models

## 2 Viterbi-Algorithmus

# Tagging

- Ein naiver Ansatz zur Berechnung der wahrscheinlichsten Tag-Sequenz wäre sehr ineffizient
  - es gibt  $T$  verschiedene Tags
  - dann gibt es für einen Satz mit  $n$  Token  $T^n$  verschiedene Tag-Sequenzen
  - für jede dieser Sequenzen müssen jeweils Emissions- und Übergangswahrscheinlichkeiten multipliziert werden
- Alternative: **Viterbi-Algorithmus**
  - Berechnung mit Hilfe **dynamischer Programmierung**
  - innerhalb einer Tabelle  $(n + 1) \cdot T$
  - Funktion  $\delta$  berechnet für jeden möglichen Zustand  $t$  für jede Position  $i$  im Satz die maximal mögliche Wahrscheinlichkeit

# Viterbi-Algorithmus

- 1** Initialisierung der Tabelle
- 2** Rekursive Abarbeitung der Tabelle
- 3** Auslesen des wahrscheinlichsten Pfads

# Viterbi-Algorithmus

## 1. Initialisierung der Tabelle

$$\delta_0(\text{START}) = 1.0$$

$$\delta_0(t) = 0 \text{ für } t \neq \text{START}$$

- D.h. die Wahrscheinlichkeit für den Anfangszustand START vor  $w_1$  ist 1
- Und die Wahrscheinlichkeit für alle anderen Zustände  $t$  vor  $w_1$  ist 0

# Viterbi-Algorithmus

## 2. Rekursive Abarbeitung der Tabelle

Für alle Positionen  $i = 0$  bis  $n$

- Für alle möglichen Tags  $t^j$  an der aktuellen Position  $i + 1$ 
  - Für alle möglichen Vorgängerzustände  $t^k$  an der Position  $i$ :

$$\delta_{i+1}(t^j) := \max_{1 \leq k \leq T} [\delta_i(t^k) \times P(t^j | t^k) \times P(w_{i+1} | t^j)]$$

**Maximale Wahrscheinlichkeit** aller Pfade zur Position  $i + 1$  mit dem aktuellen Tag  $t^j$

$$\psi_{i+1}(t^j) := \operatorname{argmax}_{1 \leq k \leq T} [\delta_i(t^k) \times P(t^j | t^k) \times P(w_{i+1} | t^j)]$$

**Vorausgehendes Tag**  $t^k$ , das die höchste Wahrscheinlichkeit für das Tag  $t^j$  an der Position  $i + 1$  erbracht hat

# Viterbi-Algorithmus

## 3. Auslesen des wahrscheinlichsten Pfads

- Auslesen des wahrscheinlichsten Tags an der Position  $n$ , also am letzten Wort  $w_n$
- Für alle Positionen  $i = n$  bis 1:  $t_i = \psi_{i+1}(t_{i+1})$

# Viterbi-Algorithmus: Beispiel

Das  
europäische  
Parlament  
ist  
noch  
nicht  
so  
leicht  
zu  
verstehen  
<S>

- Erfundener Beispielsatz mit Wörtern, die in einem 424-Sätzen langen Trainingskorpus aus der Neuen Zürcher Zeitung vorgekommen sind
- D.h. in diesem Beispiel gibt es keine unbekannten Wörter
- Einige der Worttypen könnten mit mehreren verschiedenen Tags getaggt werden, d.h. sind kategorial ambig

# Viterbi-Algorithmus: Beispiel

Das	ARTDEF / PDS
europäische	ADJA
Parlament	NN
ist	VAFIN / VVFIN
noch	ADV / KON
nicht	PTKNEG
so	ADV / KON
leicht	(ADJD /) ADV
zu	APPR / PTKA
verstehen	PTKVZ / PTKZU
<S>	(VVFIN /) VVINF
	S

- Insgesamt gibt es 56 mögliche Tag-Sequenzen für diesen Satz
- Im Trainingskorpus nicht vorgekommene, aber prinzipiell mögliche Tag-Zuweisungen (z.B. prädikatives Adjektiv ADJD für *leicht*) sind dabei ausgenommen

# Beispiel: Übergangswahrscheinlichkeiten

$P(ARTDEF|START) = 0,19575472$   
 $P(PDS|START) = 0,01179245$   
 $P(ADJA|START) = 0,05424528$   
 $P(ADJA|ARTDEF) = 0,31788079$   
 $P(ADJA|PDS) = 0,0$   
 $P(NN|ADJA) = 0,82891247$   
 $P(VAFIN|NN) = 0,01951220$   
 $P(VVFIN|NN) = 0,10182927$   
 $P(ADV|VAFIN) = 0,16083916$   
 $P(ADV|VVFIN) = 0,09976247$   
 $P(KON|VAFIN) = 0,00699301$   
 $P(KON|VVFIN) = 0,03800475$   
 $P(PTKNEG|ADV) = 0,01635992$   
 $P(PTKNEG|KON) = 0,0$   
 $P(ADV|PTKNEG) = 0,32835821$   
 $P(KON|PTKNEG) = 0,0$   
 $P(ADV|ADV) = 0,15132924$   
 $P(ADV|KON) = 0,14159292$

$P(APPR|ADV) = 0,18200409$   
 $P(PTKA|ADV) = 0,00408998$   
 $P(PTKVZ|ADV) = 0,00817996$   
 $P(PTKZU|ADV) = 0,01431493$   
 $P(VVINF|APPR) = 0,0$   
 $P(VVINF|PTKA) = 0,0$   
 $P(VVINF|PTKVZ) = 0,02631579$   
 $P(VVINF|PTKZU) = 0,984375$   
 $P(S|VVINF) = 0,36734694$

Für dieses Beispiel können Sie annehmen, dass alle anderen Übergangswahrscheinlichkeiten gleich 0 sind

# Beispiel: Emissionswahrscheinlichkeiten

$$P(\text{Das}|\text{ARTDEF}) = 0,01721854$$

$$P(\text{Das}|\text{PDS}) = 0,04761882$$

$$P(\text{europ\"aische}|\text{ADJA}) = 0,00709220$$

$$P(\text{Parlament}|\text{NN}) = 0,00053763$$

$$P(\text{ist}|\text{VAFIN}) = 0,12587404$$

$$P(\text{ist}|\text{VVFIN}) = 0,09745763$$

$$P(\text{noch}|\text{ADV}) = 0,03106796$$

$$P(\text{noch}|\text{KON}) = 0,00873362$$

$$P(\text{nicht}|\text{PTKNEG}) = 0,99999851$$

$$P(\text{so}|\text{ADV}) = 0,01941748$$

$$P(\text{so}|\text{KON}) = 0,03056769$$

$$P(\text{leicht}|\text{ADV}) = 0,00970874$$

$$P(\text{zu}|\text{APPR}) = 0,04093567$$

$$P(\text{zu}|\text{PTKA}) = 0,66665556$$

$$P(\text{zu}|\text{PTKVZ}) = 0,05128205$$

$$P(\text{zu}|\text{PTKZU}) = 0,99999844$$

$$P(\text{verstehen}|\text{VVINF}) = 0,00591716$$

$$P(<\text{S}>|\text{S}) = 0,92705139$$

Für dieses Beispiel können Sie annehmen, dass alle anderen Emissionswahrscheinlichkeiten gleich 0 sind

	---	Das	europäische	Parlament	ist	
START	1,0					S
ADJA	0,0					A
ADV	0,0					A
APPR	0,0					A
ARTDEF	0,0					A
KON	0,0					K
NN	0,0					N
PDS	0,0					P
PTKA	0,0					P
PTKNEG	0,0					P
PTKVZ	0,0					P
PTKZU	0,0					P
S	0,0					S
VAFIN	0,0					V
VVFIN	0,0					V
VVINF	0,0					V

	---	Das	europäische	Parlament	ist	noch
START	1,0	0,0				
ADJA	0,0	0,0				
ADV	0,0	0,0				
APPR	0,0	0,0				
ARTDEF	0,0	0,00337061				
KON	0,0	0,0				
NN	0,0	0,0				
PDS	0,0					
PTKA	0,0	0,0				
PTKNEG	0,0	0,0				
PTKVZ	0,0	0,0				
PTKZU	0,0	0,0				
S	0,0	0,0				
VAFIN	0,0	0,0				
VVFIN	0,0	0,0				
VVINF	0,0	0,0				

$$\delta_0(\text{START}) = 1,0$$

$$P(\text{ARTDEF}|\text{START}) = 0,19575472$$

$$P(\text{Das}|\text{ARTDEF}) = 0,01721854$$

$$\begin{aligned}\delta_1(\text{ARTDEF}) &= 1,0 * 0,19575472 * 0,01721854 \\ &= 0,00337061\end{aligned}$$

	---	Das	europäische	Parlament	ist	
START	1,0	0,0				S
ADJA	0,0	0,0				A
ADV	0,0	0,0				A
APPR	0,0	0,0				A
ARTDEF	0,0	0,00337061				A
KON	0,0	0,0				K
NN	0,0	0,0				N
PDS	0,0	0,00056154				P
PTKA	0,0	0,0				P
PTKNEG	0,0	0,0				P
PTKVZ	0,0	0,0				P
PTKZU	0,0	0,0				P
S	0,0	0,0				S
VAFIN	0,0	0,0				V
VVFIN	0,0	0,0				V
VVINF	0,0	0,0				V

$$\delta_0(\text{START}) = 1,0$$

$$P(\text{PDS}|\text{START}) = 0,01179245$$

$$P(\text{Das}|\text{PDS}) = 0,04761882$$

$$\begin{aligned}\delta_1(\text{PDS}) &= 1,0 * 0,01179245 * 0,04761882 \\ &= 0,00056154\end{aligned}$$

	---	Das	europäische	Parlament	ist	S
START	1,0	0,0	0,0	0,0	0,0	A
ADJA	0,0	0,0	7,59895296e-06			A
ADV	0,0	0,0	0,0			A
APPR	0,0	0,0	0,0			A
ARTDEF	0,0	0,00337061	0,0			A
KON	0,0	0,0	0,0			K
NN	0,0	0,0	0,0			N
PDS	0,0	0,00056154	0,0			P
PTKA	0,0	0,0	0,0			P
PTKNEG	0,0	0,0	0,0			P
PTKVZ	0,0	0,0	0,0			P
PTKZU	0,0	0,0	0,0			P
S	0,0	0,0	0,0			S
VAFIN	0,0	0,0	0,0			V
VVFIN	0,0	0,0	0,0			V

$$\delta_1(\text{ARTDEF}) = 0,00337061$$

$$P(\text{ADJA}|\text{ARTDEF}) = 0,31788079$$

$$P(\text{europäische}|\text{ADJA}) = 0,0070922$$

$$\delta_2(\text{ADJA}) = 0,00337061 * 0,31788$$

$$0,00709220$$

$$= 7,59895296e-06$$

Der Pfad, der die höchste Wahrsch für  $\delta_2(\text{ADJA})$  liefert, verläuft durch  $\delta_1(\text{ARTDEF})$  und  $\delta_0(\text{START})$ ...

	---	Das	europäische	Parlament	ist	
START	1,0	0,0	0,0	0,0	0,0	S
ADJA	0,0	0,0	7,59895296e-06	0,0	0,0	A
ADV	0,0	0,0	0,0	0,0	0,0	A
APPR	0,0	0,0	0,0	0,0	0,0	A
ARTDEF	0,0	0,00337061	0,0	0,0	0,0	A
KON	0,0	0,0	0,0	0,0	0,0	K
NN	0,0	0,0	0,0	3,38648755e-09	0,0	N
PDS	0,0	0,00056154	0,0	0,0	0,0	P
PTKA	0,0	$\delta_4(VVFIN) = 3,36076341e-11$				P
PTKNEG	0,0	$P(ADV VVFIN) = 0,09976247$				P
PTKVZ	0,0	$P(\text{noch} ADV) = 0,03106796$				P
PTKZU	0,0	$\delta_5(ADV) = 1,04164057e-13$				P
S	0,0	Ist größer als der bisherige Wert für $\delta_5(ADV)$ : 4,15619897e-14				S
VAFIN	0,0	Daher wird nur der Pfad $VVFIN \rightarrow ADV$ gespeichert...				V
VVFIN	0,0					V
VVINF	0,0					V

	noch	nicht	so	leicht	zu	ver	
START	0,0	0,0	0,0	0,0	0,0		S
ADJA	0,0	0,0	0,0	0,0	0,0		A
ADV	1,04164057e-13	0,0	1,08652323e-17	1,59633727e-20	0,0		A
APPR	0,0	0,0	0,0	0,0	1,18934467e-22		A
ARTDEF	0,0	0,0	0,0	0,0	0,0		A
KON	1,11550197e-14	0,0	0,0	0,0	0,0		K
NN	0,0	0,0	0,0	0,0	0,0		N
PDS	0,0	0,0	0,0	0,0	0,0		P
PTKA	0,0	0,0	0,0	0,0	4,35258531e-23		P
PTKNEG	0,0	1,70411291e-15	0,0	0,0	0,0		P
PTKVZ	$\delta_{10}(\text{PTKZU}) = 1,33102749e-24$			0,0	6,69639670e-24		P
PTKZU	$P(S VVINF) = 0,36734694$			0,0	2,28514180e-22		P
S	$P(<S> S) = 0,92705139$			0,0	0,0		S
VAFIN	$\delta_{11}(S) = 1,33102749e-24 * 0,36734694 *$			0,0	0,0		V
VVFIN	$0,92705139$			0,0	0,0		V
VVINF	$= 4,53280734e-25$			0,0	0,0	1,331	V

	---	Das	europäische	Parlament	ist	noch
START	1,0	0,0	0,0	0,0	0,0	0,0
ADJA	0,0	0,0	7,59895296e-06	0,0	0,0	0,0
ADV	0,0	0,0	0,0	0,0	0,0	1,04164057e-13
APPR	0,0	0,0	0,0	0,0	0,0	0,0
ARTDEF	0,0	0,00337061	0,0	0,0	0,0	0,0
KON	0,0	0,0	0,0	0,0	0,0	1,11550197e-14
NN	0,0	0,0	0,0	3,38648755e-09	0,0	0,0
PDS	0,0	0,00056154	0,0	0,0	0,0	0,0
PTKA	0,0	0,0	0,0	0,0	0,0	0,0
PTKNEG	0,0	0,0	0,0	0,0	0,0	0,0
PTKVZ	0,0	0,0	0,0	0,0	0,0	0,0
PTKZU	0,0	0,0	0,0	0,0	0,0	0,0
S	0,0	0,0	0,0	0,0	0,0	0,0
VAFIN	0,0	0,0	0,0	0,0	8,31748024e-12	0,0
VVFIN	0,0	0,0	0,0	0,0	3,36076341e-11	0,0
VVINF	0,0	0,0	0,0	0,0	0,0	0,0

# Viterbi-Algorithmus: Beispiel

Das	ARTDEF
europäische	ADJA
Parlament	NN
ist	VVFIN
noch	ADV
nicht	PTKNEG
so	ADV
leicht	ADV
zu	PTKZU
verstehen	VVINF
<S>	S

- Endergebnis: s. links
- Diese Tag-Sequenz ist in diesem Fall auch korrekt
- Man könnte sich höchstens darüber streiten, ob ist hier ein Vollverb VVFIN oder ein Auxiliarverb VAFIN ist...

# Smoothing

- In einer realistischen Anwendung eines Hidden-Markov-Modells für das Tagging
  - Emissionswahrscheinlichkeiten  $> 0$  für unbekannte Wörter
  - Übergangswahrscheinlichkeiten  $> 0$  für unbekannte Übergänge
- Dann enthalten alle Zellen in der Viterbi-Tabelle in der Regel Wahrscheinlichkeiten  $> 0$
- Das Beispiel war etwas vereinfacht, um die Berechnung übersichtlicher zu machen

# Glättung der Übergangswahrscheinlichkeiten

- Die Übergangswahrscheinlichkeiten können z.B. mit Lidstone-Glättung geglättet werden

$$P(t_2|t_1) = \frac{C(t_1, t_2) + \lambda}{C(t_1) + \lambda \cdot V}$$

- $V$  ("Vokabular"): Anzahl der verschiedenen Tags, d.h. Größe des Tagsets
- Der Wert für  $\lambda$  kann dabei entweder willkürlich festgelegt werden oder aus einem Validierungskorpus (Held-Out-Korpus) geschätzt werden.

## Behandlung unbekannter Wörter

- Man kann allen Tags dieselbe Wahrscheinlichkeit zuweisen, dass sie ein unbekanntes Wort UNK emittieren
  - entweder willkürlich: Für alle Tags  $t$  ist  $P(UNK|t) = 0.1$
  - oder indem man in einem Validierungskorpus zählt, wie viele der dort vorkommende Token unbekannte Wörter sind

$$P(UNK|t) = \frac{C(UNK)}{N}$$

- Alternativ kann man für jedes Tag eine separate Wahrscheinlichkeit dafür lernen, dass es unbekannte Wörter emittiert
  - Als unbekannte Wörter sollten dabei nur Wörter behandelt werden, die wirklich vollkommen unbekannt sind und nicht solche, die bisher nur mit anderen Tags aufgetreten sind

# Behandlung unbekannter Wörter

Berechnung der Emissionswahrscheinlichkeiten mit Berücksichtigung unbekannter Wörter

- Emissionswahrscheinlichkeit für vollkommen unbekannte Wörter:  $P(w|t) = P(UNK|t)$
- Emissionswahrscheinlichkeiten für bekannte Wörter:

$$P(w|t) = (1 - P(UNK|t)) \cdot \frac{C(w, t)}{C(t)}$$

- Auch hier sollten als unbekannte Wörter nur Wörter behandelt werden, die wirklich vollkommen unbekannt sind, und nicht solche, die nur mit anderen Tags aufgetreten sind
- Für Wörter, die nur mit anderen Tags aufgetreten sind, gilt dann  $P(w|t) = 0$

# Zusammenfassung

- Ein HMM-Tagger kombiniert zwei Arten von Information
  - Wort+Tag (Emissionswahrscheinlichkeit)
    - Accuracy bei nur lexikalischer Information (majority vote): 90%
  - Tag+Tag (Übergangswahrscheinlichkeiten)
- Die Tags sind im HMM-Modell eine zu inferierende, "verborgene" ('hidden') Ebene
- Auch hier wieder: Smoothing

# References I

Abschnitt 10 aus Manning and Schütze (1999)

Jurafsky, D. and J. H. Martin (2009).

*Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics* (2nd ed.).

Upper Saddle River, NJ: Prentice-Hall.

Manning, C. D. and H. Schütze (1999).

*Foundations of Statistical Natural Language Processing.*  
Cambridge, MA: The MIT Press.

# Symbolische und Statistische Verfahren (CL2)

## 8: Brill-Tagger

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum

RUB

# Themenüberblick Tagging

- Letzte Sitzungen
  - Wortarten-Tagging
  - Hidden-Markov-Modell-Tagger
- Heute
  - **Brill-Tagger**
  - **Vergleich HMM-Tagger vs. Brill-Tagger**
  - **Evaluation**

# Themenüberblick heute

## Brill-Tagger

### 1 Training

- Lernen des Lexikons und Initiale Annotation
- Regel-Templates und Lernen der Transformationsregeln
- Finden der effektivsten Regeln (gierige Suche)

### 2 Tagging

## Evaluationsmaße

- Accuracy und Fehlerrate
- Baseline
- Coverage/Abdeckung
- Fehlertypen
- Precision, Recall, F-Measure
- Konfusionsmatrix

# Gliederung

## 1 Brill-Tagger

## 2 Evaluationsmaße

# Transformationsbasiertes Tagging (Brill 1995)

Regelbasiertes Verfahren: Regeln zur schrittweisen Korrektur (“Transformation”) einer fehlerhaften heuristischen Annotation

- Beispiel für eine Regel (im Regelformat von Brill (1992)):

NN VB PREV-TAG TO

- NN: Ursprungstag
- VB: korrigiertes Tag
- PREV-TAG TO: Bedingung

- “Ändere das Tag NN (Nomen) in VB (Verb), wenn das vorhergehende Tag TO ist” (TO: der englische Infinitivmarker *to*)

## Bsp: NN VB PREV-TAG TO

Diane	NP	Diane	NP
loved	VBD	loved	VBD
to	TO	to	TO
dance	NN	dance	VB
,	,	,	,
something	PN	something	PN
she	PPS	she	PPS
was	BEDZ	was	BEDZ
to	TO	to	TO
demonstrate	VB	demonstrate	VB
time	NN	time	NN
and	CC	and	CC
again	RB	again	RB
.	.	.	.

# Training

Regeln: mit einem einfachen Maschinenlernverfahren (transformation-based error-driven learning) aus einem annotierten Trainingskorpus gelernt

- Phase 1: Aufbau eines **Lexikons**
  - Lernen lexikalischer Regeln aus dem annotierten Teilkorpus  $K^1$
- Phase 2: Lernen von **Transformationsregeln**
  - Initial-State-Annotation des Teilkorpus  $K^2$
  - Lernen kontextabhängiger Regeln aus dem annotierten Teilkorpus  $K^2$
  - Final-State-Annotation

Hinweis: Auch hier wird wieder mit ARTDEF (und anderen Tags) gearbeitet, die nicht Teil des originalen STTS sind; Kopula-Verben werden als VV analysiert

## Phase 1: Lernen des Lexikons

Aufbau eines Lexikons aus einem umfangreichen getagten Trainingskorpus  $K^1$

- Für jeden Worttyp wird gezählt, wie häufig er mit welchem Tag aufgetreten ist

Worttyp	das	34	ARTDEF	24	PRELS	9	PDS	1
	↑	↑	↑	↑	↑	↑	↑	
	Gesamt- häufigkeit		Häufigkeit mit Tag ARTDEF (def. Artikel)		Häufigkeit mit Tag PRELS (Rel.pron.)		Häufigkeit mit Tag PDS (Dem.pron.)	

## Phase 2: Initiale Annotation von $K^2$

“Initiale Annotation” eines zweiten Trainingskorpus  $K^2$  (Held-Out-Data) nur mit Hilfe von lexikalischem Wissen und einigen einfachen Heuristiken:

- 1 Bei bekannten Worttypen: Zuweisung des häufigsten Tags für diesen Worttyp aus dem gelernten Lexikon
- 2 Bei unbekannten Worttypen
  - Zuweisung des häufigsten Tags überhaupt
  - evtl. zusätzlich: orthographische Heuristiken
    - Großschreibung → Nomen (NN)

## Phase 2: Initiale Annotation von $K^2$

Die	ARTDEF
neue	ADJA
Verfassung	NN
Russlands	NE
,	C
geplant	VVPP
und	KON
angestrebт	VVPP
von	APPR
Präsident	NN
Jelzin	NE
,	C
dem	ARTDEF
sie	PPER
künftig	ADV
weitgehende	ADJA
Vollmachten	NN
gewährt	VVFIN
,	C
ist	VVFIN
angenommen	VVPP
worden	VAPP
.	S



## Phase 2: Fehler in der initialen Annotation

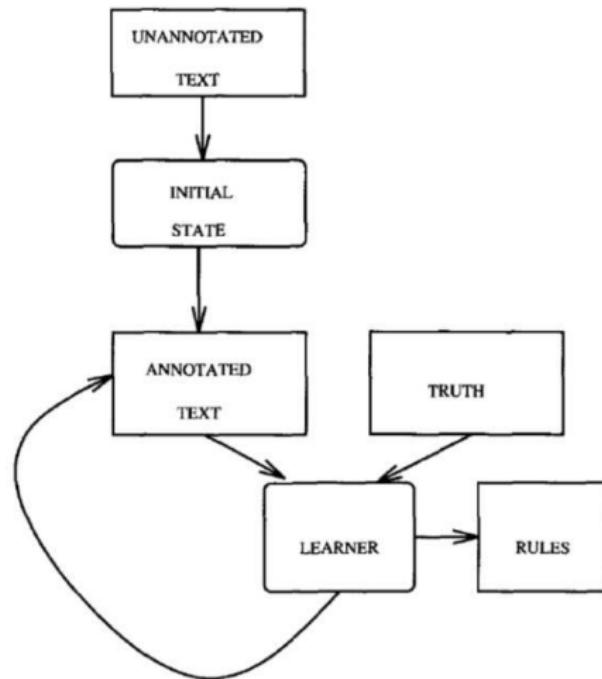
	Initiale Annot.	Korrekte Annot.
Die	ARTDEF	ARTDEF
neue	ADJA	ADJA
Verfassung	NN	NN
Russlands	NE	NE
,	C	C
geplant	VVPP	VVPP
und	KON	KON
angestrebtt	VVPP	VVPP
von	APPR	APPR
Präsident	NN	NN
Jelzin	NE	NE
,	C	C
dem	ARTDEF	PRELS
sie	PPER	PPER
künftig	ADV	ADV
weitgehende	ADJA	ADJA
Vollmachten	NN	NN
gewährt	VVFIN	VVFIN
,	C	C
ist	VVFIN	VAFIN
angenommen	VVPP	VVPP
worden	VAPP	VAPP
.	S	S



## Phase 2: Lernen der Regeln

- Lernen der kontextabhängigen Regeln durch Vergleich von:
  - Trainingskorpus  $K^2$ , das mit dem (Initial-State-)Tagger annotiert ist
  - der handannotierten “Gold”-Version von  $K^2$
- Daraus geordnete Folge von Transformationsregeln gelernt: überführen den Ist-Zustand (initiale Annotation) schrittweise in den Soll-Zustand (korrekte Annotation)

## Phase 2: Übersicht



## Schleife Ist → Soll

Solange bis keine Verbesserung mehr möglich ist (oder Verbesserung < Schwellenwert):

- Vergleiche Ist-Zustand des Trainingskorpus mit dem Soll-Zustand
  - Zähle vorkommende Fehler (Error Rate)
  - Konfusionsmatrix: < Tag<sub>richtig</sub>, Tag<sub>falsch</sub>, Anzahl >
- Generiere *alle möglichen* Regeln nach vorgegebenen Mustern (s.u.), um einen falschen Tag durch einen korrekten Tag zu ersetzen
- Wähle die Regel aus, die die höchste Fehlerreduktion auf ganz  $K^2$  bewirkt, und speichere diese Regel in der gelernten Regelabfolge
- Wende diese Regel auf den jetzigen Ist-Zustand an, um einen neuen Ist-Zustand zu erzeugen
- Beginne von vorne

## Initiale Annotation: Weiteres Beispiel

Das/ARTDEF Moskauer/ADJA Abkommen/NN über/APPR die/ARTDEF ukrainischen/ADJA strategischen/ADJA Atomwaffen/NN ist/VVFIN dagegen/PAV eine/ARTIND Leistung/NN ,/C die/ARTDEF → PRELS die/ARTDEF übrigen/ADJA eher/ADV zwiespältigen/ADJA Eindrücke/NN von/APPR Clintons/NE Reise/NN in/APPR die/ARTDEF richtige/ADJA Perspektive/NN rückt/VVFIN ./S Sollte/VMFIN die/ARTDEF Übereinkunft/NN ,/C die/ARTDEF → PRELS auf/APPR amerikanische/ADJA Vermittlung/NN zurückgeht/VVFIN ,/C diesmal/ADV verwirklicht/VVFIN → VVPP werden/VAINF ,/C könnte/VMFIN sie/PPER Präsident/NN Jelzin/NE den/ARTDEF Rücken/NN stärken/VVINF und/KON die/ARTDEF beträchtlichen/ADJA Reibungsflächen/NN verringern/VVINF ,/C die/ARTDEF → PRELS seit/APPR der/ARTDEF ukrainischen/ADJA Unabhängigkeit/NN das/ARTDEF Verhältnis/NN zwischen/APPR den/ARTDEF misstrauischen/ADJA Nachbarländern/NN prägen/VVFIN ./S

# Fehlerrate: Konfusionsmatrix

Falsch	Korrekt	Anzahl	Erklärung
VVFIN	VVPP	1	Part.Perf. statt finitem Verb
ARTDEF	PRELS	3	Rel.pron. statt def. Art.

## Regeltemplates (aus Brill (1992))

Change tag *a* to tag *b* when:

- 1 the preceding (following) word is tagged *z*.
- 2 the word two before (after) is tagged *z*.
- 3 one of the two preceding (following) words is tagged *z*.
- 4 one of the three preceding (following) words is tagged *z*.
- 5 the preceding word is tagged *z* and the following word is tagged *w*.
- 6 the preceding (following) word is tagged *z* and the word two before (after) is tagged *w*.
- 7 the current word is (is not) capitalized.
- 8 the previous word is (is not) capitalized.

# Instantiierung der Templates

- Erzeugung aller möglichen Transformationsregeln aus allen Regelmustern
- Beispiel für das Regelmuster *Change tag a to tag b when the preceding word is tagged z*
  - ARTDEF PRELS PREVTAG ADJA
  - ARTDEF PRELS PREVTAG ADV
  - ARTDEF PRELS PREVTAG APPR
  - ARTDEF PRELS PREVTAG ARTDEF
  - ARTDEF PRELS PREVTAG ARTINDEF
  - ARTDEF PRELS PREVTAG C
  - ARTDEF PRELS PREVTAG KON
  - ARTDEF PRELS PREVTAG NE
  - ARTDEF PRELS PREVTAG NN
  - ARTDEF PRELS PREVTAG PPER
  - ...

## Beispiel: welche Regel?

Das/ARTDEF Moskauer/ADJA Abkommen/NN über/APPR die/ARTDEF ukrainischen/ADJA strategischen/ADJA Atomwaffen/NN ist/VVFIN dagegen/PAV eine/ARTIND Leistung/NN ,/C die/ARTDEF → PRELS die/ARTDEF übrigen/ADJA eher/ADV zwiespältigen/ADJA Eindrücke/NN von/APPR Clintons/NE Reise/NN in/APPR die/ARTDEF richtige/ADJA Perspektive/NN rückt/VVFIN ./S Sollte/VMFIN die/ARTDEF Übereinkunft/NN ,/C die/ARTDEF → PRELS auf/APPR amerikanische/ADJA Vermittlung/NN zurückgeht/VVFIN ,/C diesmal/ADV verwirklicht/VVFIN → VVPP werden/VAINF ,/C könnte/VMFIN sie/PPER Präsident/NN Jelzin/NE den/ARTDEF Rücken/NN stärken/VVINF und/KON die/ARTDEF beträchtlichen/ADJA Reibungsflächen/NN verringern/VVINF ,/C die/ARTDEF → PRELS seit/APPR der/ARTDEF ukrainischen/ADJA Unabhängigkeit/NN das/ARTDEF Verhältnis/NN zwischen/APPR den/ARTDEF misstrauischen/ADJA Nachbarländern/NN prägen/VVFIN ./S

## Beispiel (Forts.)

- In dem Beispiel traten folgende Fehler auf:
  - < PRELS, ARTDEF, 3 >  
(Relativpronomen 3x falsch als Artikel getaggt)
  - < VVPP, VVFIN, 1 >  
(Partizip Perfekt 1x falsch als finites Vollverb getaggt)
- Auswahl der Regel, die insgesamt die meisten Fehler korrigiert
  - ARTDEF PRELS PREV-TAG C  
= wandle ARTDEF (Artikel) in PRELS (Relativpronomen) um, wenn ein Token mit dem Tag C (Komma) vorausgeht  
→ korrigiert 3 Fehler

## Beispiel: Auswahl der effektivsten Regel

Beispiel für das Regelmuster *Change tag a to tag b when the preceding word is tagged z*

- ARTDEF PRELS PREVTAG ADJA
- ARTDEF PRELS PREVTAG ADV
- ARTDEF PRELS PREVTAG APPR
- ARTDEF PRELS PREVTAG ARTDEF
- ARTDEF PRELS PREVTAG ARTINDEF
- ARTDEF PRELS PREVTAG C
- ARTDEF PRELS PREVTAG KON
- ARTDEF PRELS PREVTAG NE
- ARTDEF PRELS PREVTAG NN
- ARTDEF PRELS PREVTAG PPER
- ...

# Regelanwendung

Das/ARTDEF Moskauer/ADJA Abkommen/NN über/APPR die/ARTDEF ukrainischen/ADJA strategischen/ADJA Atomwaffen/NN ist/VVFIN dagegen/PAV eine/ARTIND Leistung/NN ,/C **die/PRELS** die/ARTDEF übrigen/ADJA eher/ADV zwiespältigen/ADJA Eindrücke/NN von/APPR Clintons/NE Reise/NN in/APPR die/ARTDEF richtige/ADJA Perspektive/NN rückt/VVFIN ./S Sollte/VMFIN die/ARTDEF Übereinkunft/NN ,/C **die/PRELS** auf/APPR amerikanische/ADJA Vermittlung/NN zurückgeht/VVFIN ,/C diesmal/ADV verwirklicht/VVFIN → VVPP werden/VAINF ,/C könnte/VMFIN sie/PPER Präsident/NN Jelzin/NE den/ARTDEF Rücken/NN stärken/VVINF und/KON die/ARTDEF beträchtlichen/ADJA Reibungsflächen/NN verringern/VVINF ,/C **die/PRELS** seit/APPR der/ARTDEF ukrainischen/ADJA Unabhängigkeit/NN das/ARTDEF Verhältnis/NN zwischen/APPR den/ARTDEF misstrauischen/ADJA Nachbarländern/NN prägen/VVFIN./S

# Regelanwendung

- In diesem Fall hat die Anwendung der Regel  
ARTDEF PRELS PREV-TAG C  
drei Fehler korrigiert, ohne neue Fehler zu erzeugen
- Übrig bleibt noch der folgende Fehler  
 $\langle \text{VVPP}, \text{VVFIN}, 1 \rangle$   
(Partizip Perfekt einmal falsch als finites Vollverb getaggt)
- Nächste Runde
  - Vergleich von Ist- und Soll-Zustand (Konfusionsmatrix)
  - Erzeugung aller möglichen Regeln
  - Auswahl der effektivsten Regel
- Nächste gelernte Regel: VVFIN VVPP NEXT-TAG VAINE

## Auswahl der effektivsten Regel

Wie wählt man in jedem Durchgang die effektivste Regel?

- BesteRegel = NULL
- Fehler = Anzahl falsche Tags im Ist-Zustand
- Für jede mögliche Regel R:
  - erzeuge eine Kopie des Korpus im Ist-Zustand
  - wende die Regel R auf diese Kopie an
  - vergleiche die veränderte Kopie mit dem Soll-Zustand und ermitte die Anzahl der Fehler: FehlerNeu
  - Wenn FehlerNeu < Fehler
    - BesteRegel = R
    - Fehler = FehlerNeu
- Am Ende enthält BesteRegel die effektivste Regel (oder es wurde keine einzige Regel gefunden, die zu einer Fehlerreduktion führt)

# Beispiel (aus Brill (1995))

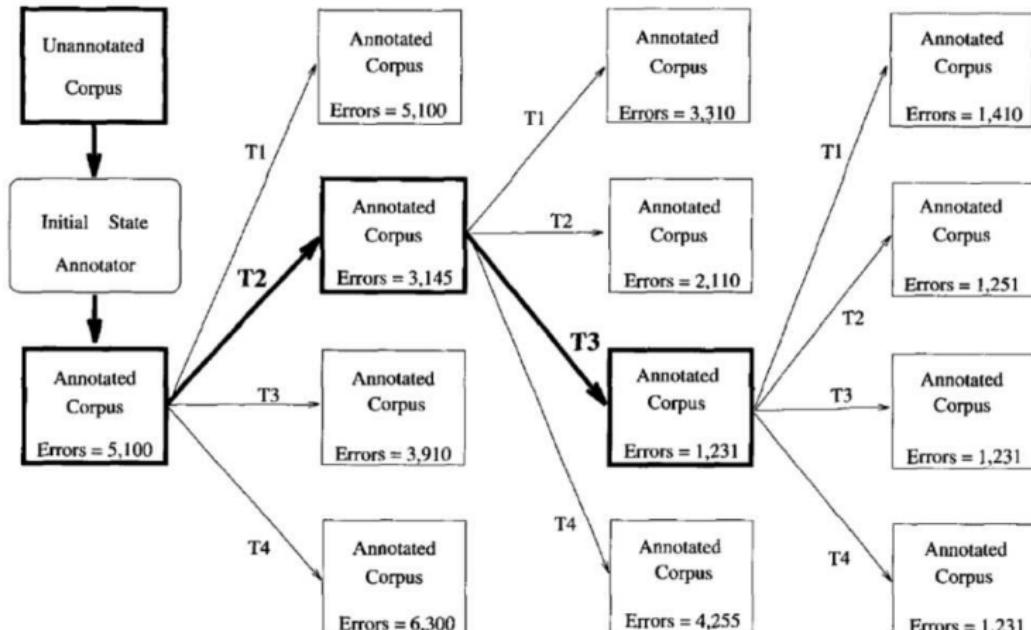


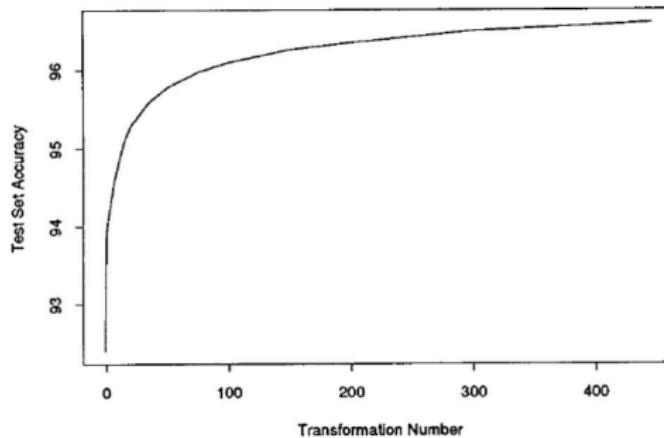
Figure 2  
An Example of Transformation-Based Error-Driven Learning.

## Pseudocode (aus Brill (1995))

```
1. apply initial-state annotator to corpus
2. while transformations can still be found do
3.   for from_tag = tag1 to tagn
4.     for to_tag = tag1 to tagn
5.       for corpus_position = 1 to corpus_size
6.         if (correct_tag(corpus_position) == to_tag)
7.           && current_tag(corpus_position) == from_tag)
8.             num_good_transformations(tag(corpus_position -1))++
9.         else if (correct_tag(corpus_position) == from_tag)
10.           && current_tag(corpus_position) == from_tag)
11.             num_bad_transformations(tag(corpus_position-1 ))++
12.         find maxT (num_good_transformations(T) - num_bad_transformations(T))
13.         if this is the best-scoring rule found yet then store as best rule:
14.           Change tag from from_tag to to_tag if previous tag is T
15.         apply best rule to training corpus
16.         append best rule to ordered list of transformations
```

Figure 3  
Pseudocode for learning transformations.

# Trainingsverlauf (aus Brill (1995))



- PennTreebank: 1,1 Mio Wörter
  - davon rund 30% für Phase 1 (Lexikon); 55% für Phase 2 (Transformationsregeln); 15% für Evaluation
- Regeln für Unknowns: 243; Transformationsregeln: 447
- Accuracy: 96.6%

# Top-10-Regeln für Englisch

aus Brill (1992)

- 1 TO IN NEXT-TAG AT  
(*to* als Präp. statt Infinitivmarker vor Artikel)
- 2 VBN VBD PREV-WORD-IS-CAP YES
- 3 VBD VBN PREV-1-OR-2-OR-3-TAG HVD
- 4 VB NN PREV-1-OR-2-TAG AT
- 5 NN VB PREV-TAG TO
- 6 TO IN NEXT-WORD-IS-CAP YES
- 7 NN VB PREV-TAG MD
- 8 PPS PPO NEXT-TAG .
- 9 VBN VBD PREV-TAG PPS
- 10 NP NN CURRENT-WORD-IS-CAP NO

# Top-10-Regeln für unbekannte Wörter

aus Brill (1995)

Erweiterte Version: kann Bezug auf Wortform nehmen:

From	To	Condition
NN	NNS	Has suffix <i>-s</i>
NN	CD	Has character <i>.</i>
NN	JJ	Has character <i>-</i>
NN	VBN	Has suffix <i>-ed</i>
NN	VBG	Has suffix <i>-ing</i>
??	RB	Has suffix <i>-ly</i>
??	JJ	Adding suffix <i>-ly</i> results in a word
NN	CD	The word <i>\$</i> can appear to the left
NN	JJ	Has suffix <i>-al</i>
NN	VB	The word <i>would</i> can appear to the left

# Taggen eines Korpus

Taggen eines neuen Korpus:

- Initiale Annotation mit demselben Initial-State-Tagger wie beim Training
  - z.B. häufigstes Tag für jedes bekannte Wort
  - Heuristiken für unbekannte Wörter
- Anwendung der gelernten Transformationsregeln in der gespeicherten Reihenfolge → korrigiert schrittweise Fehler in der initialen Annotation

Der gesamte Algorithmus ist ausführlich in Brill (1992), Brill (1995) und in Jurafsky and Martin (2009) beschrieben

# Vergleich HMM-Tagger vs. Brill-Tagger

## HMM-Tagger

Nur direkt angrenzender Kontext wird berücksichtigt

Genaue Wahrscheinlichkeiten werden ermittelt und auch beim Tagging genutzt

Die Auswirkungen der einzelnen Parameter nicht einfach zu durchschauen

Kurze Trainingszeit

Modell mit vielen Parametern

## Brill-Tagger

Beliebiger Kontext möglich

Quantitative Informationen werden nur beim Lernen genutzt und gehen danach verloren

Gelernte Regeln sind leicht verständlich

Training dauert relativ lange

Wenige Regeln

# Gliederung

## 1 Brill-Tagger

## 2 Evaluationsmaße

# Accuracy und Fehlerrate

Evaluationsmaße für Klassifikatoren: **Accuracy** (Genauigkeit) bzw. **Fehlerrate** der Klassifikation

- **Klassifikator**: System, das Beobachtungen klassifiziert  
= in Kategorien einteilt
- Bsp. Tagger, Wortbedeutungsdisambiguierer,  
Satzgrenzenerkenner

$$\text{Accuracy} = \frac{\text{richtig klassifiziert}}{\text{alle Klassifizierten}}$$

$$\text{Fehlerrate} = \frac{\text{falsch klassifiziert}}{\text{alle Klassifizierten}}$$

# Baseline

- Um abschätzen zu können, wie *schwer* eine Aufgabe ist, wird oft eine **Baseline** mit angegeben
  - Baseline: ein eher simpler Referenzalgorithmus, den das evaluierte System übertreffen sollte
  - Bsp. Tagging: jedes bekannte Wort bekommt sein häufigstes Tag: Accuracy/Baseline von 90% (Englisch)
- Oft: je mehr Forschung auf einem Gebiet gemacht wird, desto komplexere Baselines werden definiert

## Abdeckung/Coverage

Bei einem System, das nicht notwendigerweise jede Beobachtung klassifizieren muss (z.B. mangels Evidenz):  
**Coverage** (Abdeckung) der Testdaten relevant

$$\text{Coverage} = \frac{\text{klassifizierte Einheiten}}{\text{klassifizierbare Einheiten}}$$

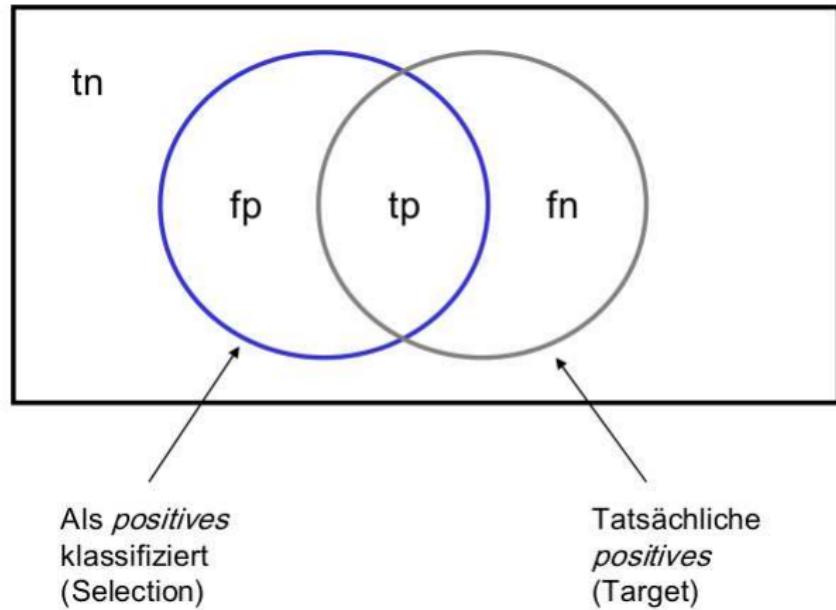
# Fehlertypen

Statt nur richtig/falsch: verschiedene Fehlertypen (meist bei Klassifikation in zwei mögliche Klassen benutzt)

- **True positives (TP):** korrekterweise klassifiziert
- **True negatives (TN):** korrekterweise nicht klassifiziert
- **False positives (FP):** fälschlicherweise klassifiziert  
(Typ II-Fehler)
- **False negatives (FN):** fälschlicherweise nicht klassifiziert  
(Typ I-Fehler)

		Goldstandard	
		A	$\neg A$
System	A	TP	FP
	$\neg A$	FN	TN

# Fehlertypen (nach Manning and Schütze (1999))



## Beispiel Satzgrenzenerkennung

Klassifikationsaufgabe: für jeden Punkt wird entschieden, ob es sich um eine Satzgrenze (positive) handelt oder nicht (negative)

- TP: Satzgrenzenpunkt wird vom System korrekt erkannt
  - *Und so weiter.<S> Aber das darf nicht dazu führen ...*
- TN: Punkt, der keine Satzgrenze markiert, wird vom System korrekt erkannt
  - *dem 15 Verbände bzw.<ABBR> 436 548 Männer und Frauen angeschlossen sind ...*
- FP: Punkt, der keine Satzgrenze markiert, wird als Satzgrenze klassifiziert
  - *Bereits am 13.<S> Februar will der Parteivorstand seinen Antrag ...*
- FN: Satzgrenzenpunkt wird nicht als Satzgrenze erkannt
  - *sondern auch den kritischen Journalisten im Inland usw.<ABBR> Und all das ...*

# Precision und Recall

- Bei Bool'scher Klassifikation (z.B. Satzgrenze ja/nein): als Evaluationsmaß meist **Precision** und **Recall**
- Auch Tagging kann so aufgefasst werden: für jede Wortart (z.B. Nomen) wird pro Wort geschaut: wurde es richtig als Nomen/Nicht-Nomen klassifiziert

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Correct system answers}}{\text{All system answers}}$$

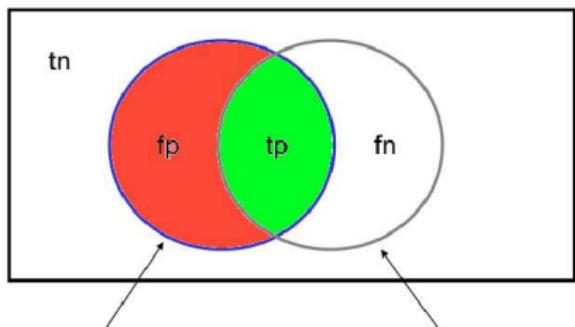
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Correct system answers}}{\text{All correct answers}}$$

- D.h. TN werden ignoriert
  - Grund: Maße kommen aus IR, dort TN irrelevant und oft unbekannt

# Precision und Recall

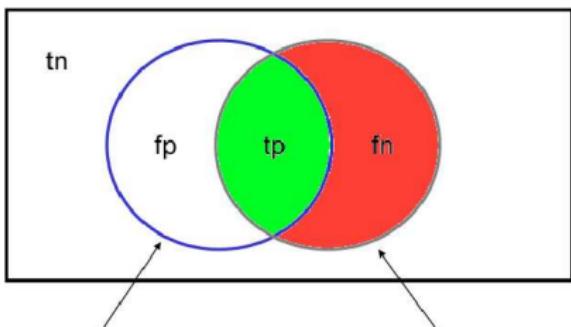
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Correct system answers}}{\text{All system answers}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Correct system answers}}{\text{All correct answers}}$$



Als positives  
klassifiziert  
(Selection)

Tatsächliche  
positives  
(Target)



Als positives  
klassifiziert  
(Selection)

Tatsächliche  
positives  
(Target)

# Precision-Recall-Trade-Off

Meist ergibt sich ein Trade-Off zwischen Precision und Recall

- Wenn man sehr vorsichtig klassifiziert, dann findet man nicht alle relevanten Beispiele, aber macht dabei auch wenig Fehler
  - hohe Precision, niedriger Recall
- Wenn man hingegen großzügig klassifiziert, findet man vielleicht alle relevanten Beispiele, aber auch viele irrelevante (FP)
  - hoher Recall, niedrige Precision

## F-Measure ( $F_1$ )

- Zum besseren Vergleich verschiedener Systeme:  
**F-Measure (F-Score)** als Zusammenfassung von Precision und Recall
- Harmonisches Mittel aus Precision und Recall

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Daneben sind andere Gewichtungen möglich (z.B.  $F_{0.5}$ ,  $F_2$ ), aber in der CL eher unüblich
  - gewichtet:  $F_\beta = (1 + \beta^2) \frac{P \cdot R}{(\beta^2 \cdot P) + R}$

# Vergleich Accuracy und Precision/Recall

aus Manning and Schütze (1999)

- Insgesamt 100,000 Datenpunkte
- Davon sind 150 Datenpunkte zu klassifizieren
- Im Folgenden beispielhafte Klassifizierungen:
  - a. hohe Präzision → hoher Recall
  - b. dasselbe bei gleichbleibender Accuracy

# Vergleich Accuracy und Precision/Recall

aus Manning and Schütze (1999)

	<i>tp</i>	<i>fp</i>	<i>fn</i>	<i>tn</i>	Prec	Rec	<i>F</i>	Acc
(a)	25	0	125	99,850	1.000	0.167	0.286	0.9988
	50	100	100	99,750	0.333	0.333	0.333	0.9980
	75	150	75	99,700	0.333	0.500	0.400	0.9978
	125	225	25	99,625	0.357	0.833	0.500	0.9975
	150	275	0	99,575	0.353	1.000	0.522	0.9973
(b)	50	0	100	99,850	1.000	0.333	0.500	0.9990
	75	25	75	99,825	0.750	0.500	0.600	0.9990
	100	50	50	99,800	0.667	0.667	0.667	0.9990
	150	100	0	99,750	0.600	1.000	0.750	0.9990

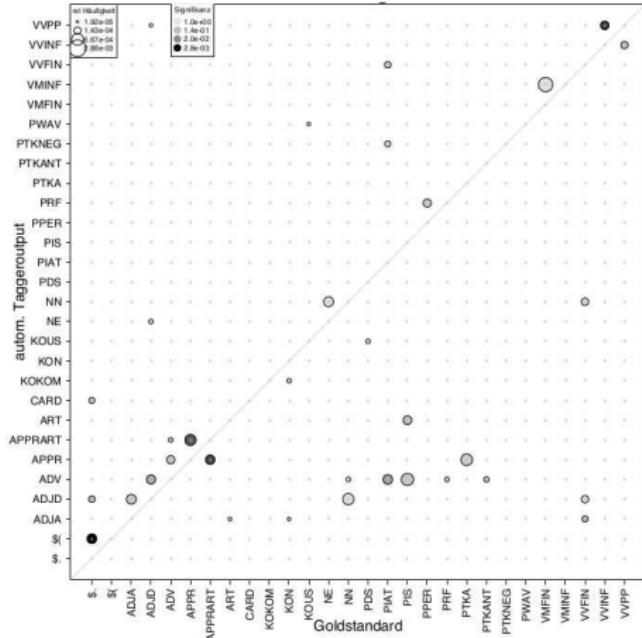
# Fehleranalyse per Konfusionsmatrix

## aus Giesbrecht (2008) (TIGER-Korpus)

	Adjective	Adverb	CARD	Noun	Preposition	Pronoun	Verb	\$	Conjunction	XY	Article	FM	Interjection	TRUNC	total
Adjective	70844	957	20	811	33	372	643		6	50	7	79		1	7 73830
Adverb	866	36050		38	120	1429	2		715	2		13		2	1 39238
CARD	6		15837	15		10									1 15869
Noun	1098	74	38	234069	42	58	82	3	13	544	8	931		5	29 236994
Preposition	44	541		129	87396	593	1		246	5		11			88966
Pronoun	218	498	3	67	291	68158	35		60	2	659	10			1 70002
Verb	880	27		181	3	68	106170		4	17		29		2	107381
\$					3	1		119602		3					119609
Conjunction		341		28	3229	1046			31703	9		4			36360
XY	1	4	3	66		4				233	1	48			360
Article	8	2	30	56		882					96622	2			97602
FM			8	694	4	1				146	1	136		1	991
Interjection		1		7						2		2		7	19
TRUNC	8			13											1325 1346
total	73973	38495	15939	236174	91121	72622	106933	119605	32747	1013	97298	1265		18	1364 888567

# Fehleranalyse per Konfusionsmatrix

## aus Reznicek and Zinsmeister (2013) (Lernerdaten)



## References I

- Brill-Tagger: Abschnitt 10.4 aus Manning and Schütze (1999) und Abschnitt 5.6 aus Jurafsky and Martin (2009)
- Evaluation: Abschnitt 8.1 aus Manning and Schütze (1999) und Abschnitt 5.7 aus Jurafsky and Martin (2009)

Brill, E. (1992).

A simple rule-based part of speech tagger.

In *Proceedings of the third Conference on Applied Natural Language Processing ANLP'92*, pp. 152–155.

Brill, E. (1995).

Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging.

*Computational Linguistics* 21(4), 543–565.

## References II

- Giesbrecht, E. (2008).  
Evaluation of POS tagging for web as corpus.  
Master's thesis, University of Osnabrück.
- Jurafsky, D. and J. H. Martin (2009).  
*Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics* (2nd ed.).  
Upper Saddle River, NJ: Prentice-Hall.
- Manning, C. D. and H. Schütze (1999).  
*Foundations of Statistical Natural Language Processing*.  
Cambridge, MA: The MIT Press.

## References III

Reznicek, M. and H. Zinsmeister (2013).  
STTS-Konfusionsklassen beim Tagging von  
Fremdsprachlernertexten.  
*JLCL* 28(1), 63–83.



# Symbolische und Statistische Verfahren (CL2)

## 9: Word Sense Disambiguation

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum

The logo of Ruhr-Universität Bochum (RUB) is located in the bottom right corner. It consists of the letters "RUB" in a white, bold, sans-serif font, all contained within a solid dark blue square.

# Themenüberblick

## Wortbedeutungsdisambiguierung

- Lexikalische Semantik
- Lexikalische Ambiguität in der Sprachverarbeitung
- Wortbedeutungsdisambiguierung mit Naive Bayes
- Evaluation und manuelle Annotation: IAA

# Gliederung

## 1 Word Sense Disambiguation (WSD)

- Lexikalische Ambiguität in der Sprachverarbeitung
- Wortbedeutungsdisambiguierung mit Naive Bayes
- Evaluation und manuelle Annotation

# Ambige Lexeme

- Viele Wörter (Lexeme) haben mehrere mögliche Bedeutungen
- Beispiel *Bank* im Deutschen Wörterbuch (Paul 2002)
  - 1. a. "Sitzgelegenheit für mehrere Personen"  
b. "Unterlage für Handwerkerarbeit" (z.B. Drehbank, Hobelbank, ...)  
c. "übertr. von bankähnlichen Naturgebilden" (z.B. Sandbank)
  - 2. a. "Wechselbank", öffentliche Kasse  
b. "Gebäude, in welchem sich eine solche [...] befindet"  
c. "auf Glücksspiele bezogen 'das Geld desjenigen, gegen welchen die übrigen Spieler spielen'"  
d. "Sammelstelle" (z.B. Blut-, Daten-, Samenbank)

# Homophonie, Homographie, Homonymie

## Homophonie ("gleiche Lautung")

- 1 *phonologische* Wortform – mehrere unterschiedliche Bedeutungen
- Beispiele?
  - *Leute* vs. *(ich) läute*
  - *Lehre* vs. *Leere*
  - *Fähre* vs. *faire*
  - *Biss* vs. *bis*
  - *Feld* vs. *fällt*
  - *Fliege* vs. *(ich) fliege*
  - *Haut* vs. *(er) haut*

# Homophonie, Homographie, Homonymie

## Homographie ("gleiche Schreibung")

- 1 orthographische Wortform – mehrere unterschiedliche Bedeutungen
- Beispiele?
  - (zu) *rasten* vs. (*sie*) *rasten* (Prät.)
  - *über'setzen* vs. *'übersetzen*
  - *um'fahren* vs. *'umfahren*
  - *Hero'in* vs. *He'roin*
  - *mo'dern* (Adj.) vs. (*sie*) *'modern* (Verb)

# Homophonie, Homographie, Homonymie

## Homonymie ("gleicher Name")

- 1 *phonologische und orthographische* Wortform – mehrere unterschiedliche Bedeutungen
- Keine plausible Beziehung zwischen den Bedeutungen
- d.h. zwei Lexeme haben *rein zufällig* dieselbe Form
- Beispiele?
  - (Definitionen aus Paul 2002)
  - *Tau*
    - 1 "Niederschlag der feuchten Nachtluft"
    - 2 "starkes Schiffsseil"
  - *Mark*
    - 1 "weiches Gewebe"
    - 2 "Grenze" (veraltet)
    - 3 "Geldstück"

# Polysemie

## Polysemie (“viele Bedeutungen”)

- 1 *Lexem* (nicht: Wortform!) – mehrere *miteinander verwandte* Bedeutungen
- Beispiele
  - *Lehre*: Lehrtätigkeit, Lehrmeinung usw.
  - *Schule*: Institution, Gebäude usw.
  - *Stimme*: beim Reden, bei einer Wahl usw.
  - *Spitze*: eines Pfeils, einer Demonstration, einer Rangliste
  - *arm*: elend, mittellos
  - *fliegen*: als Passagier, als Pilot, als Vogel

# Lexikalische Semantik

Beispiel “Bank” im Deutschen Wörterbuch (Paul 2002)

- 1**
  - a. “Sitzgelegenheit für mehrere Personen”
  - b. “Unterlage für Handwerkerarbeit” (z.B. Drehbank, Hobelbank, . . . )
  - c. “übertr. von bankähnlichen Naturgebilden” (z.B. Sandbank)
- 2**
  - a. “Wechselbank”, öffentliche Kasse
  - b. “Gebäude, in welchem sich eine solche [...] befindet”
  - c. “auf Glücksspiele bezogen ‘das Geld desjenigen, gegen welchen die übrigen Spieler spielen’”
  - d. “Sammelstelle” (z.B. Blut-, Daten-, Samenbank)

→ Lesarten 1. vs. 2.: Homonymie

→ Lesarten a. vs. b. etc.: Polysemie

# Gliederung

## 1 Word Sense Disambiguation (WSD)

- Lexikalische Ambiguität in der Sprachverarbeitung
- Wortbedeutungsdisambiguierung mit Naive Bayes
- Evaluation und manuelle Annotation

# Lexikalische Ambiguität in der Sprachverarbeitung

Welche Probleme ergeben sich durch lexikalische Ambiguität für die Verarbeitung natürlicher Sprache?

- Maschinelle Übersetzung
  - Homonyme haben oft unterschiedliche Übersetzungen
  - Deutsch *Bank* = Englisch *bank* oder *bench*
  - Englisch *bank* = Deutsch *Bank* oder *Ufer*
- Text-to-Speech-Systeme
  - Bei homographen aber nicht homophonen Wörtern muss die korrekte Aussprache gewählt werden
  - *Das ist sehr mo'dern* vs. *Sie 'modern in ihren Gräbern*
- Spracherkennung
  - Homophone Wörter müssen auseinander gehalten werden
  - *Der Zustand der Leere/Lehre ist besorgniserregend*

# Lexikalische Ambiguität in der Sprachverarbeitung

Welche Probleme ergeben sich durch lexikalische Ambiguität für die Verarbeitung natürlicher Sprache?

- Information Retrieval
  - wenn man im Internet nach *Kiefer* sucht, meint man entweder nur den Baum oder nur den Knochen oder nur den Künstler Anselm Kiefer
- Immer wenn man für statistische Verfahren Wortformen zählt, gibt es Probleme mit Homographen
  - N-Gramm-Modelle
  - Tagging
  - Abkürzungserkennung
  - ...

# Wortbedeutungsdisambiguierung

## Engl. Word sense disambiguation (WSD)

- Disambiguierung einer lexikalisch ambigen Wortform durch Zuweisung der korrekten Bedeutung im jeweiligen Kontext
- Korrekte Bedeutung: z.B. gemäß einem Standardlexikon oder Wortnetz
- Beispiel: Bank\_1 = Kreditinstitut vs. Bank\_2 = Sitzmöbel
  - *die Million jederzeit auf ein Sperrkonto bei einer Bank\_1 einzuzahlen*
  - *Sie erhielten denn auch meist einige Stellen oder eine besondere Bank\_2 im Rat*

# Wortbedeutungsdisambiguierung

Disambiguierung = eine Art Tagging-Problem: weise jedem Wort eine Bedeutung aus dem Lexikon zu (“Sense tagging”)

- Unterschied zu POS-Tagging: POS-Tagging nutzt v.a. lokale Information (z.B. Tag des Vorgängerworts), während WSD v.a. einen größeren Kontext benötigt
- Aber: im Gegensatz zu POS keine klar definierten “Tags”!
- Beispiel *title*:
  - 1 Name/heading of a book, statute, work of art or music, etc.
  - 2 Material at the start of a film
  - 3 The right of legal ownership (of land)
  - 4 The document that is evidence of this right
  - 5 An appellation of respect attached to a person's name
  - 6 A written work [by synecdoche, i.e., putting a part for the whole]

# Wortbedeutungsdisambiguierung

- Grundsätzlich: zwei verschiedene Ansätze
  - 1 **Supervised**: mit gelabelten Trainingsdaten
    - Klassifikationsaufgabe
  - 2 **Unsupervised**: mit ungelabelten Trainingsdaten
    - Clustering-Aufgabe
- Praktisch: oft Mischung aus beidem bzw. Nutzung zusätzlicher Ressourcen
  - z.B. unsupervised, aber mit bilingualem Lexikon

# Gliederung

## 1 Word Sense Disambiguation (WSD)

- Lexikalische Ambiguität in der Sprachverarbeitung
- Wortbedeutungsdisambiguierung mit Naive Bayes
- Evaluation und manuelle Annotation

# Naiver Bayes-Ansatz (Gale, Church, and Yarowsky 1992)

- Sehr einfaches statistisches Klassifikationsverfahren
  - **Bayes'sche Entscheidungsregel**
  - Annahme von statistischer Unabhängigkeit zwischen den verwendeten Klassifikationsmerkmalen (= "naiv")
- Überwachtes Lernen
  - Lernen anhand annotierter Beispiele des ambigen Worts (Trainingskorpus)
- Mögliche Merkmale
  - Wörter im Kontext und deren relative Position
  - **Bag of Words** ("bow", Wörter im Kontext ohne Position)
  - POS-Tags im Kontext
  - ...

# Bayes'scher Klassifikator

- Weise dem zu klassifizierenden Beispiel die Bedeutung  $s'$  aus der Menge aller möglichen Bedeutungen  $S$  zu, wenn  $s'$  gegeben den Kontext  $c$  am wahrscheinlichsten ist

$$s' = \operatorname{argmax}_{s \in S} P(s|c)$$

- Anwendung der Bayes'schen Regel:  $P(s|c) = \frac{P(c|s) \cdot P(s)}{P(c)}$ 
  - A-posteriori-, Merkmals- und A-priori-Wahrscheinlichkeit
- Wahrscheinlichste Bedeutung

$$s' = \operatorname{argmax}_{s \in S} \frac{P(c|s) \cdot P(s)}{P(c)} = \operatorname{argmax}_{s \in S} P(c|s) \cdot P(s)$$

- Wahrscheinlichkeit des Kontextes ist für alle möglichen Klassen gleich → wird weggelassen

# Naive Bayes

## (Naive) Unabhängigkeitsannahme

- Alle einzelnen Merkmale  $c_j$  des Kontextes  $c$ , die bei der Klassifikation benutzt werden, sind voneinander unabhängig

$$P(c|s) = \prod_{c_j \in c} P(c_j|s)$$

### Vorteile

- weniger Probleme mit Datenknappheit: einzelne Merkmale sind viel häufiger als einzelne Kombinationen aus vielen Merkmalen
- beliebige Merkmale können sehr einfach kombiniert werden

### Nachteile

- die Merkmale im Kontext sind natürlich nicht wirklich voneinander unabhängig
- genauere Schätzung der Wahrscheinlichkeiten wäre möglich

## Kontext als “Moving Window”

Welche Bedeutung des Lexems “Bank” liegt in folgendem Beispiel vor?

Dieses sieht vor, dass jede Bank eine eigene Bad Bank gründet und dafür Garantien vom Sonderfonds Finanzmarktstabilisierung (SoFFin) bekommt.

→ Die am besten geeignete Größe des Kontextfensters muss empirisch ermittelt werden

# Mögliche Kontextmerkmale

- 1 Häufigkeit des Worttyps  $w$  in Kontexten, in denen die Bedeutung  $s$  korrekt ist: **Bag-of-Words-Modell**
  - *die Million jederzeit auf ein Sperrkonto bei einer Bank\_1 einzuzahlen*
  - Vektor: [*bei* 1, *einer* 1, *einzuzahlen* 1, *Sperrkonto* 1]
- 2 Häufigkeit des Worttyps  $w$  an der relativen Position  $p$  in Kontexten, in denen die Bedeutung  $s$  korrekt ist:  
Kollokationsmodell
  - *die Million jederzeit auf ein Sperrkonto bei einer Bank\_1 einzuzahlen*
  - Vektor: [<-3: *Sperrkonto* 1>, <-2: *bei* 1>, <-1: *einer* 1>, <+1: *einzuzahlen* 1>]
- 3 Weitere Merkmale der Wörter im Kontext: z.B. Wortart, Lemma, Stamm etc.

# Training: Maximum-Likelihood-Estimation (MLE)

- A-Priori-Wahrscheinlichkeit  $P(s) = \frac{C(s,w)}{C(w)}$ 
  - bestimme die Anteile (= Wahrscheinlichkeiten) der verschiedenen möglichen Bedeutungen  $s$  eines Wortes  $w$
- Indiv. Merkmalswahrscheinlichkeiten  $P(c_j|s) = \frac{C(c_j,s)}{C(s)}$ 
  - zähle, wie häufig jedes Klassifikationsmerkmal mit den verschiedenen möglichen Bedeutungen auftritt
- Beispiel: Bag-of-Words-Modell
  - Auswahl eines Kontextfensters, z.B.  $-10..+10$  Wörter um das zu disambiguierende Wort
  - gegeben eine Wortform  $w$ : ermittle für alle ihre möglichen Bedeutungen  $s$ , wie oft welche Worttypen innerhalb des Kontextfensters um  $w$  herum im Trainingskorpus vorkommen
    - die Frequenzen der Kontextwörter aus verschiedenen Vorkommen werden addiert

## Beispiel: engl. *bar*

- Beispiel mit *bar*
- Zwei Bedeutungen:
  - 1 “Snack-Bar”
  - 2 “Stange”

# Beispiel: engl. **bar**: Trainingskorpus

## 1. Bedeutung “Snack-Bar”

- Kontexte im Trainingskorpus ( $\pm 4$  Wörter)

- [*shop , a snack bar and a petrol station*]
  - [*, and a snack bar and T . V*]
  - [*tail at the snack bar ; it's actually not*]
  - [*pong table , snack bar or VCR , but*]

- Bag-of-Words

- als Vektor (ohne Satzzeichen und Padding-Dummys)

[*a* 3, *shop* 1, *snack* 4, *and* 3, *petrol* 1, *station* 1, *T* 1, *V* 1,  
*tail* 1, *at* 1, *the* 1, *it's* 1, *actually* 1, *not* 1, *pong* 1, *table* 1, *or*  
1, *VCR* 1, *but* 1 ]

## Beispiel: engl. **bar**: Trainingskorpus

### 2. Bedeutung “Stange”

- Kontexte im Trainingskorpus ( $\pm 4$  Wörter)
  - [*burn through the metal bars . DUMMY DUMMY DUMMY*]
  - [*a companion with metal bars as they left an*]
- Bag-of-Words (Vektor; ohne Satzzeichen und Dummys)  
[*burn 1, through 1, the 1, metal 2, a 1, companion 1, with 1, as 1, they 1, left 1, an 1*]

# Beispiel: engl. **bar**: Training

## 1. Bedeutung “Snack-Bar”

- A priori:  $P(\text{“Snack-Bar”}) = \frac{C(\text{“Snack-Bar”})}{C(\text{bar})} = \frac{4}{6} = \frac{2}{3}$
- Merkmals-Wahrscheinlichkeiten  
 $[a\ 3, \text{shop}\ 1, \text{snack}\ 4, \text{and}\ 3, \text{petrol}\ 1, \text{station}\ 1, T\ 1, V\ 1, \text{tail}\ 1, \text{at}\ 1, \text{the}\ 1, \text{it's}\ 1, \text{actually}\ 1, \text{not}\ 1, \text{pong}\ 1, \text{table}\ 1, \text{or}\ 1, \text{VCR}\ 1, \text{but}\ 1]$ 
  - $P(a|\text{“Snack-Bar”}) = \frac{C(a, \text{“Snack-Bar”})}{C(\text{“Snack-Bar”})} = \frac{3}{4}$
  - $P(\text{shop}|\text{“Snack-Bar”}) = \frac{1}{4}$
  - $P(\text{snack}|\text{“Snack-Bar”}) = 1.0$
  - $P(\text{and}|\text{“Snack-Bar”}) = \frac{3}{4}$
  - $P(\text{petrol}|\text{“Snack-Bar”}) = P(\text{station}|\text{“Snack-Bar”}) = P(T|\text{“Snack-Bar”}) = P(V|\text{“Snack-Bar”}) = P(\text{tail}|\text{“Snack-Bar”}) = P(\text{at}|\text{“Snack-Bar”}) = P(\text{the}|\text{“Snack-Bar”}) = P(\text{it's}|\text{“Snack-Bar”}) = P(\text{actually}|\text{“Snack-Bar”}) = P(\text{not}|\text{“Snack-Bar”}) = P(\text{table}|\text{“Snack-Bar”}) = P(\text{or}|\text{“Snack-Bar”}) = P(\text{VCR}|\text{“Snack-Bar”}) = P(\text{but}|\text{“Snack-Bar”}) = \frac{1}{4}$

## Beispiel: engl. **bar**: Training

### 2. Bedeutung “Stange”

- A priori:  $P(\text{“Stange”}) = \frac{C(\text{“Stange”})}{C(\text{bar})} = \frac{2}{6} = \frac{1}{3}$
- Merkmals-Wahrscheinlichkeiten
  - [*burn* 1, *through* 1, *the* 1, *metal* 2, *a* 1, *companion* 1, *with* 1, *as* 1, *they* 1, *left* 1, *an* 1]
  - $P(\text{metal}|\text{“Stange”}) = \frac{C(\text{metal}, \text{“Stange”})}{C(\text{“Stange”})} = \frac{2}{2} = 1.0$
  - $P(\text{burn}|\text{“Stange”}) = P(\text{the}|\text{“Stange”}) =$   
 $P(\text{through}|\text{“Stange”}) = P(\text{a}|\text{“Stange”}) =$   
 $P(\text{companion}|\text{“Stange”}) = P(\text{with}|\text{“Stange”}) =$   
 $P(\text{as}|\text{“Stange”}) = P(\text{they}|\text{“Stange”}) = P(\text{left}|\text{“Stange”}) =$   
 $P(\text{an}|\text{“Stange”}) = 0.5$

# Klassifikation (“Semantisches Tagging”)

- Ermittle die Merkmale  $c_{1..n}$  im Kontext  $c$  des zu disambiguierenden Worttokens
- Und berechne die wahrscheinlichste Bedeutung für das Worttoken
  - für jede mögliche Bedeutung  $s$ : berechne die Wahrscheinlichkeit, dass  $s$  mit den Merkmalen  $c_{1..n}$  auftritt:  
 $P(c|s) = \prod_{c_j \in c} P(c_j|s)$
  - wähle die Bedeutung  $s'$ , für die  $P(c|s) \cdot P(s)$  maximal wird
- Geeignetes Smoothing durchführen, sonst ergibt  $\prod_{c_j \in c} P(c_j|s) = 0$  bei unbekannten Wörtern innerhalb des Kontexts  $c$ !

# Smoothing

- Für unbekannte Wörter: z.B. sehr einfache Glättungsmethode von Ng (1997)
- $P(c_j|s_n) = \frac{P(s_n)}{N} = \frac{C(s_n)}{N^2}$  für  $C(c_j, s_n) = 0$   
mit N = Anzahl Traingsdaten
- Im Beispiel:
  - $P(c_j|\text{"Snack-Bar"}) = \frac{4}{6^2} = \frac{1}{9}$  für  $C(c_j, \text{"Snack-Bar"}) = 0$
  - $P(c_j|\text{"Stange"}) = \frac{2}{6^2} = \frac{1}{18}$  für  $C(c_j, \text{"Stange"}) = 0$

# Klassifikation einer Instanz von *bar*

- C = [*a seat at the bar which serves up surprisingly*]
- Wahrscheinlichkeit für Bedeutung s = “Snack-Bar”
  - Produkt der Merkmals-Wahrscheinlichkeiten:  
 $P(c|s) = \prod_{c_j \in c} P(c_j|s)$ 
    - $\prod_{c_j \in c} P(c_j|“Snack-Bar”) =$   
 $P(\text{a}|“Snack-Bar”) \cdot P(\text{seat}|“Snack-Bar”) \cdot P(\text{at}|“Snack-Bar”) \cdot$   
 $P(\text{the}|“Snack-Bar”) \cdot P(\text{which}|“Snack-Bar”) \cdot$   
 $P(\text{serves}|“Snack-Bar”) \cdot P(\text{up}|“Snack-Bar”) \cdot$   
 $P(\text{surprisingly}|“Snack-Bar”)$
  - ... und mit A-Priori-Wahrscheinlichkeit multiplizieren:
    - ...  $P(“Snack-Bar”)$
  - damit:  $P(“Snack-Bar”|C)$   
 $= \frac{3}{4} \cdot \frac{1}{9} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{9} \cdot \frac{1}{9} \cdot \frac{1}{9} \cdot \frac{1}{9} \cdot \frac{2}{3}$   
 $= \frac{6}{11337408} \approx 5.29e-07$

## Klassifikation einer Instanz von *bar*

$C = [\text{a seat at the bar which serves up surprisingly}]$

1. Wahrscheinlichkeit für Bedeutung  $s = \text{"Stange"}$

■  $P(\text{"Stange"})|C) =$

$$\begin{aligned} & P(\text{a}|\text{"Stange"}) \cdot P(\text{seat}|\text{"Stange"}) \cdot P(\text{at}|\text{"Stange"}) \cdot \\ & P(\text{the}|\text{"Stange"}) \cdot P(\text{which}|\text{"Stange"}) \cdot P(\text{serves}|\text{"Stange"}) \cdot \\ & P(\text{up}|\text{"Stange"}) \cdot P(\text{surprisingly}|\text{"Stange"}) \cdot P(\text{"Stange"}) \end{aligned}$$

$$= \frac{1}{2} \cdot \frac{1}{18} \cdot \frac{1}{18} \cdot \frac{1}{2} \cdot \frac{1}{18} \cdot \frac{1}{18} \cdot \frac{1}{18} \cdot \frac{1}{18} \cdot \frac{1}{3}$$

$$= \frac{1}{408,146,688} \approx 2.45\text{e-}09$$

# Klassifikation eines Beispiels aus dem Internet

C = [a seat at the bar which serves up surprisingly]

2.  $P(\text{"Snack-Bar"}|C) \approx 5.29e-07 > P(\text{"Stange"}|C) \approx 2.45e-09$
- Hier also:  $P(\text{"Snack-Bar"}|C) > P(\text{"Stange"}|C)$   
→ dem neuen Beispiel wird die Bedeutung "Snack-Bar" zugewiesen  
→ korrekte Entscheidung

# Gliederung

## 1 Word Sense Disambiguation (WSD)

- Lexikalische Ambiguität in der Sprachverarbeitung
- Wortbedeutungsdisambiguierung mit Naive Bayes
- Evaluation und manuelle Annotation

# Evaluation

- Systeme zur Wortbedeutungsdisambiguierung: Evaluation meist anhand der Accuracy

$$\text{Accuracy} = \frac{\text{korrekt klassifiziert}}{\text{falsch klassifiziert} + \text{richtig klassifiziert}}$$

- Zusätzlich: Upper und Lower Bounds angeben, um die Schwierigkeit der Aufgabe einzuschätzen
  - **Upper bound**: üblicherweise Performanz menschlicher Annotatoren (z.B. gegeben ein Kontextfenster von  $\pm n$  Wörtern)
  - **Lower bound** = Baseline: A-Priori-Wahrscheinlichkeit, d.h. jedem zu disambiguierenden Wort wird die wahrscheinlichste Bedeutung (unabhängig vom Kontext), d.h. die insgesamt häufigste Bedeutung, zugeordnet

# Manuelle Annotationen

- Oft wird als upper bound die Performanz menschlicher Annotatoren genommen
- Frage: woher weiß man, wie gut die Annotatoren sind?
  - Lösung: man vergleicht ihre Ergebnisse mit einem **Goldstandard** = einer korrekten Referenzannotation
- Frage: woher kommt der Goldstandard?
  - Lösung: mehrere Annotatoren annotieren unabhängig voneinander, vergleichen anschließend ihre Ergebnisse und kommen gemeinsam zu einem Ergebnis

# Abweichungen der Annotatoren

- Beim Erstellen eines Goldstandards spielt eine Rolle, wie sehr sich die Annotatoren voneinander unterscheiden haben
- Gründe für große Abweichungen?
  - schlechte Annotatoren
  - schwierige Aufgabe (subjektive Aufgabe)
  - schlechte Anweisungen (Guidelines)

# Inter-annotator agreement

- Daher: üblicherweise wird zu einem Goldstandard bzw. einem annotierten Korpus auch das **Inter-annotator agreement** (IAA) mit angegeben
  - andere Namen: inter-coder agreement, inter-rater reliability (IRR)
- Verschiedene Maße zum Messen der Übereinstimmung: Scott's  $\pi$ , Cohen's  $\kappa$  u.a.
- Erkenntnis: einiges Agreement ist per Zufall zu erwarten
  - z.B. 50%, wenn 2 Annotatoren 2 Werte rein zufällig annotieren
- "Sinnvolles" Agreement: oberhalb des Zufalls

# Erwartete Übereinstimmung

- **Observed agreement** ( $A_o$ ): Anteil des tatsächlichen Agreements
- **Expected agreement** ( $A_e$ ): erwarteter Wert von  $A_o$
- Agreement oberhalb Zufall:  $A_o - A_e$ 
  - Maximum mögliches Agreement oberhalb Zufall:  $1 - A_e$
- Anteil des “sinnvollen” Agreements:

$$\frac{A_o - A_e}{1 - A_e}$$

- Frage: wie berechnet man das Zufallsagreement? ( $A_e$ )
  - die verschiedenen IAA-Maße unterscheiden sich darin, wie  $A_e$  berechnet wird
  - im folgenden: beispielhaft Cohen's  $\kappa$

## $\kappa$ (Cohen's kappa)

- Annahme: unterschiedliche Wahrscheinlichkeiten für Annotatoren und Kategorien (Cohen 1960; Carletta 1996)
- Gesamtanzahl an Markables:  $i$
- Wahrscheinlichkeit, dass ein Annotator  $c_x$  eine bestimmte Kategorie  $q_a$  auswählt:  $\frac{n_{c_x q_a}}{i}$
- Wahrscheinlichkeit, dass beide Annotatoren eine bestimmte Kategorie  $q_a$  auswählen:  $\frac{n_{c_1 q_a}}{i} * \frac{n_{c_2 q_a}}{i}$
- Wahrscheinlichkeit, dass zwei Annotatoren dieselben Kategorien auswählen:

$$A_e^\kappa = \sum_q \frac{n_{c_1 q}}{i} * \frac{n_{c_2 q}}{i} = \frac{1}{i^2} \sum_q n_{c_1 q} n_{c_2 q}$$

# Beispiele

	A	B	Summe
A	41	3	44
B	9	47	56
Summe	50	50	100

$$A_o: \frac{41+47}{100} = .88$$

$$\kappa = \frac{.88 - .5}{1 - .5} = .76$$

	A	B	Summe
A	20	10	30
B	60	10	70
Summe	80	20	100

$$A_o: \frac{20+10}{100} = .3$$

$$\kappa = \frac{.3 - .38}{1 - .38} = -.129$$

# Beurteilung von Agreement-Scores

- $\kappa = 0$  bedeutet: keine höhere Übereinstimmung als Zufall  
 $\kappa = 1$  bedeutet: vollkommene Übereinstimmung
- $\kappa < .7$  oft als schlecht beurteilt, d.h. Goldstandard dann unbrauchbar (aber umstritten)
- Landis and Koch (1977): agreement

$\kappa < 0$	“poor”
$0 < \kappa < .2$	“slight”
$.21 < \kappa < .4$	“fair”
$.41 < \kappa < .6$	“moderate”
$.61 < \kappa < .8$	“substantial”
$\kappa > .81$	“near perfect”

## References I

- Abschnitte 7 aus Manning and Schütze (1999)
- Abschnitt 17 aus Jurafsky and Martin (2009)

Carletta, J. (1996).

Assessing agreement on classification tasks: the kappa statistic.  
*Computational Linguistics* 22(2), 249–254.

Cohen, J. (1960).

A coefficient of agreement for nominal scales.

*Educational and Psychological Measurement* 20(1), 37–46.

Gale, W. A., K. W. Church, and D. Yarowsky (1992).

A method for disambiguating word senses in a large corpus.

*Computers and the Humanities* (26), 415–439.

## References II

Jurafsky, D. and J. H. Martin (2009).

*Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics* (2nd ed.).

Upper Saddle River, NJ: Prentice-Hall.

Landis, J. R. and G. G. Koch (1977).

The measurement of observer agreement for categorical data.  
*Biometrics* 33(1).

Manning, C. D. and H. Schütze (1999).

*Foundations of Statistical Natural Language Processing*.  
Cambridge, MA: The MIT Press.

## References III

Ng, H. T. (1997).

Exemplar-based word sense disambiguation: Some recent improvements.

In *Proceedings of EMNLP-97*, pp. 208–213.

Paul, H. (1897/2002).

*Deutsches Wörterbuch. Bedeutungsgeschichte und Aufbau unseres Wortschatzes* (10th ed.).

Tübingen: Niemeyer.

Revised and extended by Helmut Henne, Heidrun Kämper und Georg Objartel.

# Symbolische und Statistische Verfahren (CL2)

## 10: CYK- und Earley-Algorithmus

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum

# Themenüberblick Parsing

Heute:

- Parsing: Strategien, Backtracking
- Parsing-Algorithmen
  - **Cocke-Younger-Kasami-Algorithmus**
  - Appendix: einfacher Shift-Reduce-Parser
  - Appendix: Earley-Algorithmus
- Nächste Sitzung
  - Probabilistisches Parsing
  - Baumbanken
  - Evaluation

# Gliederung

## 1 Parsing

- Backtracking

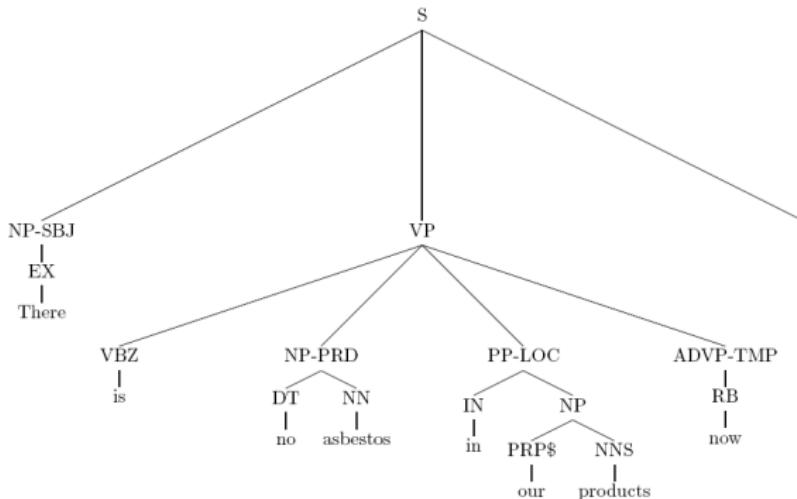
## 2 Cocke-Younger-Kasami-Parser

## 3 Appendix

- Shift-Reduce-Parser
- Earley-Parser

# Was bedeutet Parsing?

- “Parsing” = in Teile zerlegen (lat. pars = Teil)
- Das Ziel beim Parsing ist es, die **syntaktische Struktur** eines Satzes automatisch zu analysieren



# Wozu Parsing?

Wofür wird die syntaktische Struktur benötigt?

- Semantische Analyse

Eine vollständige Bedeutungsanalyse setzt eine Analyse der syntaktischen Struktur voraus

- Textgenerierung

Die Erzeugung natürlichsprachlicher Sätze z.B. aus den Ergebnissen einer Datenbankabfrage

- Sprachmodell

Eine Grammatik kann auch als alternatives Sprachmodell dienen, das im Gegensatz zu einem N-Gramm-Modell nicht-lokale Abhängigkeiten modellieren kann

# Kontextfreie Grammatiken

- Die meisten einfachen Parser verwenden kontextfreie Grammatiken
- Definition: Eine **kontextfreie Grammatik** besteht aus
  - einer Menge von **nicht-terminalen Symbolen**  $N$
  - einer Menge von **terminalen Symbolen**  $\Sigma$
  - einer Menge von **Regeln** der Form  $A \rightarrow \alpha$ , wobei  $A$  ein nicht-terminales Symbol ist:  $A \in N$  und  $\alpha$  eine Folge von terminalen und/oder nicht-terminalen Symbolen:  $\alpha = (\Sigma \cup N)^*$
  - einem speziellen **Startsymbol**  $S$

# Kontextfreie Regeln

- Die Grammatikregeln werden dabei als sog. **Rewrite-Rules** (Ersetzungsregeln) interpretiert
  - Die Regel  $S \rightarrow NP\ VP$  besagt z.B., dass das Symbol  $S$  durch die beiden Symbole  $NP$   $VP$  ersetzt werden darf
  - Kontextfreie Grammatiken heißen deshalb kontextfrei, weil z.B. das Symbol  $S$  unabhängig von dem Kontext, in dem es steht, immer durch  $NP$  und  $VP$  ersetzt werden darf

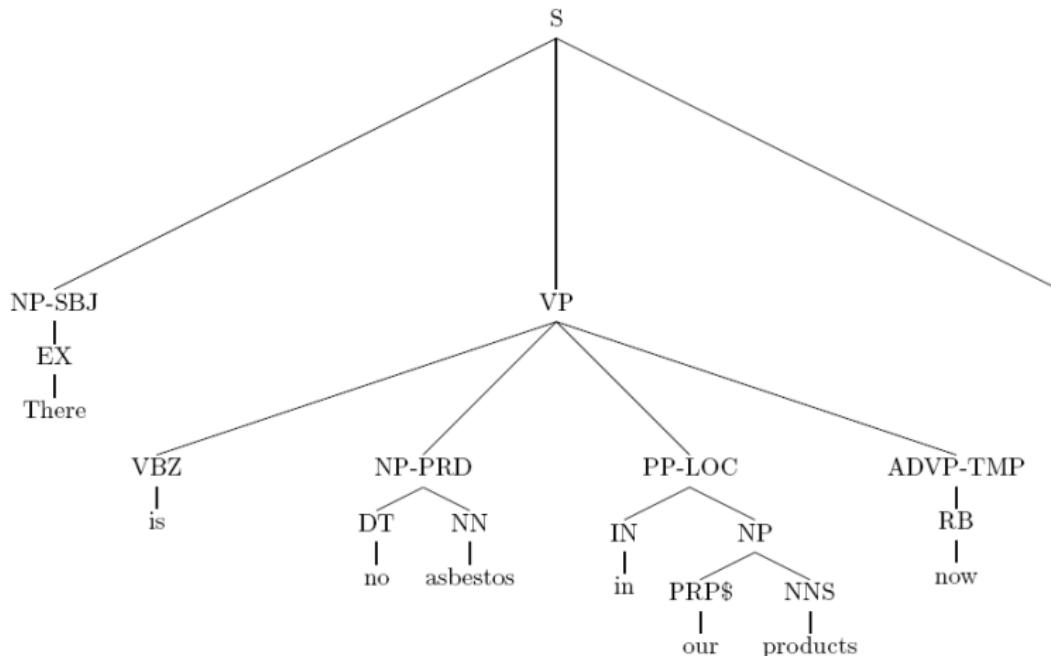
# Kontextfreie Sprachen

- Definition: Eine **kontextfreie Sprache**  $L_G$  ist die (potenziell unendliche) Menge von Ketten terminaler Symbole der Grammatik, die mit Hilfe der Regeln aus dem Startsymbol  $S$  abgeleitet werden können

$$L_G = \{ w \mid w \in \Sigma^* \text{ und } S \rightarrow^* w \}$$

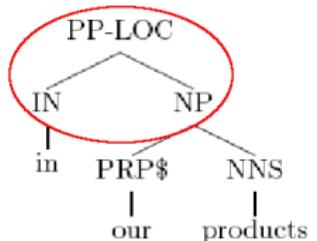
- D.h.: es gibt für jeden Satz der Sprache  $L_G$  auch mindestens eine Ableitung nach der Grammatik  $G$
- Solche Ableitungen (**Derivationen**) werden meist als Baumdiagramme dargestellt

# Beispielbaum aus der Penn-Treebank



# Kontextfreie Grammatiken

- Jede Regel der Grammatik beschreibt einen möglichen Teilbaum
- Schema: PP-LOC → IN NP  
Ein Knoten PP-LOC kann in zwei Unterkonstituenten der Kategorien IN und NP zerfallen



Beschriebener Teilbaum

- Klammerstruktur: [PP-LOC [IN in] [NP ...]]

# Terminologie

Jede Regel beschreibt die Beziehung zwischen einem **Mutterknoten** und einer oder mehreren **Töchtern**

- NP-SUBJ → EX  
eine Mutter – eine Tochter (**unäre Regel**)
- NP-PRD → DT NN  
eine Mutter – zwei Töchter (**binäre Regel**)
- Ternäre Regeln: eine Mutter – drei Töchter usw.

# Terminologie

PP-LOC → IN NP

- “PP-LOC kann expandiert werden zu IN und NP”
- “PP-LOC geht über in/geht nach IN und NP”

Auch:

- “IN und NP können eine Konstituente PP-LOC bilden”
- “PP-LOC dominiert IN und NP”
- “IN und NP sind Schwesterknoten”

# CL-Beispielgrammatik für das Deutsche

$S \rightarrow NP\ VP$

$VP \rightarrow V$

$VP \rightarrow V\ NP$

$VP \rightarrow V\ NP\ NP$

$VP \rightarrow VP\ PP$

$NP \rightarrow D\ N$

$NP \rightarrow D\ AP\ N$

$AP \rightarrow A$

$PP \rightarrow P\ NP$

$XP \rightarrow XP$  und  $XP$

$V \rightarrow$  gibt, geben, schläft,  
schlafen, sieht, sehen

$P \rightarrow$  auf, hinter, in, vor

$D \rightarrow$  das, der, den, die, diese,  
dem, ein, eine, einen

$N \rightarrow$  Junge, Jungen, Hund,  
Kinder, Keks, Baum, Mädchen, Blume

$A \rightarrow$  kleines, kleine, kleinen,  
kleiner

Hinweis: die Regel  $S \rightarrow NP\ VP$  sieht man oft in CL-Grammatiken, sie ist aber linguistisch eigentlich nicht sinnvoll (im Deutschen)

# Grammatiken in der CL

- Wörter werden als terminale Symbole modelliert (und meist aus der eigentlichen Grammatik in ein **Lexikon** ausgelagert)
- Wörter werden mit Hilfe unärer Regeln verschiedenen lexikalischen Kategorien (Wortarten) — den **präterminalen Symbolen** — zugeordnet
  - V → schläft
- Größere Konstituenten (syntaktische Phrasen) werden durch **nicht-terminale Symbole** bezeichnet
  - S: wird in der Linguistik meist als “Satz” interpretiert
  - NP: Nominalphrase etc.

# Übung

Können die folgenden Sätze mit der gegebenen Grammatik analysiert werden?

- *der kleine Junge schläft auf dem Baum*
- *das Mädchen und der Junge geben dem Hund einen Keks*
- *das Mädchen sieht Kinder*

$S \rightarrow NP\ VP$

$VP \rightarrow V$

$VP \rightarrow V\ NP$

$VP \rightarrow V\ NP\ NP$

$VP \rightarrow VP\ PP$

$NP \rightarrow D\ N$

$NP \rightarrow D\ AP\ N$

$AP \rightarrow A$

$PP \rightarrow P\ NP$

$XP \rightarrow XP \text{ und } XP$

$V \rightarrow \text{gibt, geben, schläft, schlafen,}$   
 $\quad \quad \quad \text{sieht, sehen}$

$P \rightarrow \text{auf, hinter, in, vor}$

$D \rightarrow \text{das, der, den, die, diese, dem,}$   
 $\quad \quad \quad \text{ein, eine, einen}$

$N \rightarrow \text{Junge, Jungen, Hund, Kinder,}$   
 $\quad \quad \quad \text{Keks, Baum, Mädchen, Blume}$

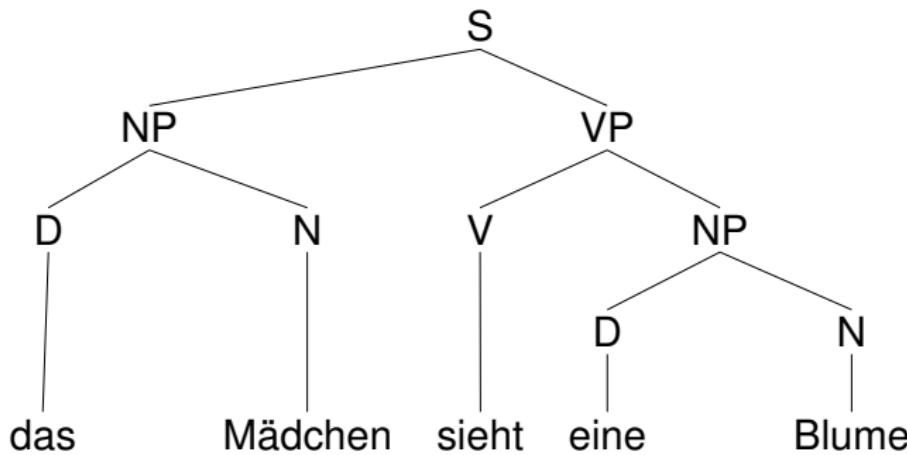
$A \rightarrow \text{kleines, kleine, kleinen, kleiner}$

# Parsing: Aufgabe

- Gegeben eine Grammatik und ein Satz: finde eine (oder mehrere) Ableitungen für den Satz gemäß der Grammatik
- D.h. finde einen (oder mehrere) Bäume mit dem Wurzelknoten S, die alle Wörter des Satzes überspannen

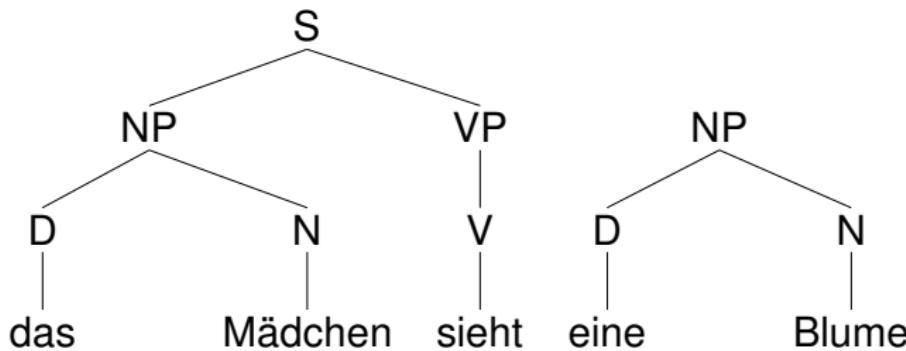
# Strategien beim Parsing: Top-Down

**Top-Down:** Suche nach einer Ableitung, die zu dem vorliegenden Satz führt, ausgehend vom Startsymbol S



## Strategien beim Parsing: Bottom-Up

**Bottom-Up:** Datengetriebene Suche nach einer Ableitung ausgehend von den Wörtern (terminalen Symbolen)

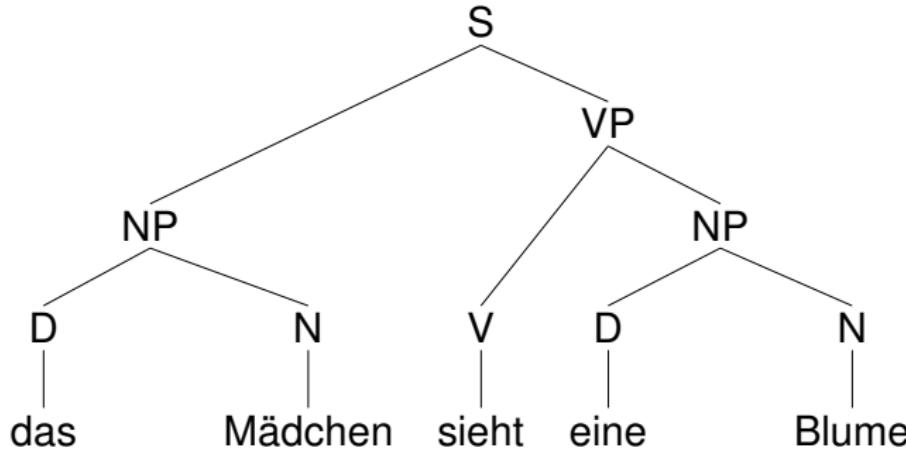


Gescheiterte Derivation: Satz nicht vollständig geparsst!

→ **Backtracking:** zum letzten Teilbaum zurückgehen, zu dem es eine alternative anwendbare Regel gibt (hier: VP-Regel)

# Backtracking

**Bottom-Up:** Datengetriebene Suche nach einer Ableitung  
ausgehend von den Wörtern (terminalen Symbolen)



# Strategien beim Parsing

## 1. Analyserichtung (vertikal)

- von der Wurzel S zum Eingabestring (Top-down)
- oder vom Eingabestring zur Wurzel (Bottom-up)

## 2. Verarbeitungsrichtung (horizontal)

- von links nach rechts, rechts nach links oder bi-direktional
- z.B. links nach rechts: zuerst D expandieren, dann N
- (für welche Sprachen könnte rechts-nach-links besser funktionieren?)

## Strategien beim Parsing (Forts.)

3. Alternativen: Bei mehreren passenden Regeln: welche Regel wird zuerst angewandt?
  - z.B. nach Reihenfolge der Regeln in der Grammatik
  - z.B. NP soll expandiert werden, benutze als erstes die erste NP-Regel; falls die scheitert, probiere die nächste NP-Regel (Backtracking)
  - oder: alle Regeln parallel anwenden (Parallel Processing)

### 4. Suchstrategie

- Suche zunächst in die Tiefe: **depth-first search**  
Der aktuelle Knoten wird zunächst bis zu seinen terminalen Knoten expandiert, bevor der nächste Schwesterknoten betrachtet wird (analog für bottom-up: aktueller Knoten möglichst weit auf dominierende Knoten zurückgeführt)

- Suche zunächst in die Breite: **breadth-first search**  
Bearbeite alle Symbole "der Reihe nach", wie sie entstehen



# Strategien beim Parsing (Forts.)

5. Exhaustivität: Abbruch nach dem Finden der ersten möglichen Derivation oder Finden aller möglichen Derivationen

# Übung: Top-down-Parsing

Suchen Sie nach einer Analyse für die folgenden Sätze:

- *kleiner Hund schläft*
- *das Mädchen und der Junge geben dem Hund einen Keks*

Nutzen Sie dabei die Strategien: top-down, rechts-links, erste Regel, depth-first, nicht-exhaustiv

$S \rightarrow NP VP$	$V \rightarrow$ gibt, geben, schläft, schlafen,
$VP \rightarrow V$	sieht, sehen
$VP \rightarrow V NP$	$P \rightarrow$ auf, hinter, in, vor
$VP \rightarrow V NP NP$	$D \rightarrow$ das, der, den, die, diese, dem,
$VP \rightarrow VP PP$	ein, eine, einen
$NP \rightarrow D N$	$N \rightarrow$ Junge, Jungen, Hund, Kinder,
$NP \rightarrow D AP N$	Keks, Baum, Mädchen, Blume
$AP \rightarrow A$	$A \rightarrow$ kleines, kleine, kleinen, kleiner
$PP \rightarrow P NP$	
$XP \rightarrow XP \text{ und } XP$	

# Übung: Bottom-Up-Parsing

Suchen Sie nach einer Analyse für den folgenden Satz:

■ *die kleinen Kinder sehen einen Hund auf dem Baum*

Nutzen Sie dabei die Strategien: bottom-up, rechts-links, erste Regel, breadth-first, nicht-exhaustiv

$S \rightarrow NP VP$	$V \rightarrow$ gibt, geben, schläft, schlafen,
$VP \rightarrow V$	sieht, sehen
$VP \rightarrow V NP$	$P \rightarrow$ auf, hinter, in, vor
$VP \rightarrow V NP NP$	$D \rightarrow$ das, der, den, die, diese, dem,
$VP \rightarrow VP PP$	ein, eine, einen
$NP \rightarrow D N$	$N \rightarrow$ Junge, Jungen, Hund, Kinder,
$NP \rightarrow D AP N$	Keks, Baum, Mädchen, Blume
$AP \rightarrow A$	$A \rightarrow$ kleines, kleine, kleinen, kleiner
$PP \rightarrow P NP$	
$XP \rightarrow XP \text{ und } XP$	

# Herausforderungen beim Parsing

- Falsche Wahl → Backtracking
- Ambiguitäten

# Gründe für Backtracking

Wenn der Parser zu einem Zeitpunkt der Analyse mehrere mögliche nächste Operationen hat, spricht man von einem Auswahlpunkt (**choice point**)

- Beispiel Top-Down-Parsing:  $VP$  könnte expandiert werden zu
  - V
  - V NP
  - ...
- Beispiel Bottom-Up-Parsing: Die Sequenz NP VP PP könnte ersetzt werden durch
  - S PP
  - NP VP
  - ...

# Wann ist Backtracking notwendig?

## ■ Top-Down-Parsing

- Die aktuelle Expansion einer Regel führt zu einem terminalen Symbol, das nicht an der aktuellen Position in der Kette vorkommt
- Der aktuelle Baum sagt mehr terminale Knoten vorher als in der Kette tatsächlich vorkommen
- Der Baum kann nicht weiter expandiert werden, aber die Kette der terminalen Symbole ist noch nicht vollständig abgedeckt

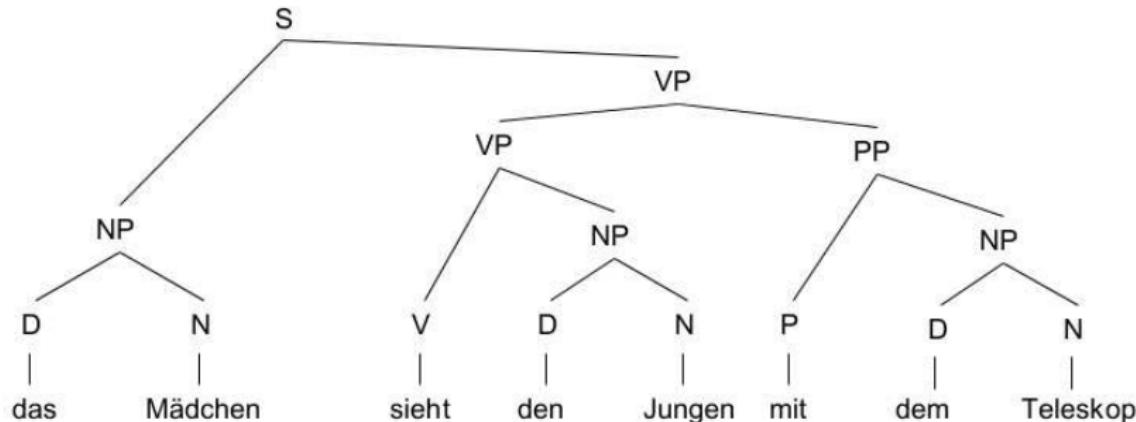
## ■ Bottom-Up-Parsing

- Das Startsymbol als Wurzelknoten des Baumes ist erreicht, aber es sind nicht alle terminalen Symbole der Kette in den Baum integriert worden
- Das Startsymbol ist noch nicht erreicht und es gibt keine Möglichkeit, mehrere Teilbäume durch eine Regel zu einem neuen Knoten zusammenzufassen

# Backtracking und strukturelle Ambiguitäten

Backtracking ist auch bei strukturell ambigen Sätzen notwendig, um alle möglichen Derivationen zu finden

- *das Mädchen sieht den Jungen mit dem Teleskop*



# Vergleich Top-down vs. Bottom-up Parser

## ■ Top-down:

- “Theoriegetriebenes Parsing”
- Vorteil: kreiert nur Ableitungen mit Wurzelknoten S
- Nachteil: kreiert auch Ableitungen, die inkonsistent mit dem Input sind
- Problem: mit depth-first, left-to-right: Algorithmus terminiert nicht bei Linksrekursion  
(Bsp: NP → NP PP)

## ■ Bottom-up:

- “Datengetriebenes Parsing”
- Vorteil: kreiert nur Ableitungen, die konsistent mit dem Input sind
- Nachteil: kreiert Ableitungen, die nicht zum Wurzelknoten S kommen

# Gliederung

## 1 Parsing

- Backtracking

## 2 Cocke-Younger-Kasami-Parser

## 3 Appendix

- Shift-Reduce-Parser
- Earley-Parser

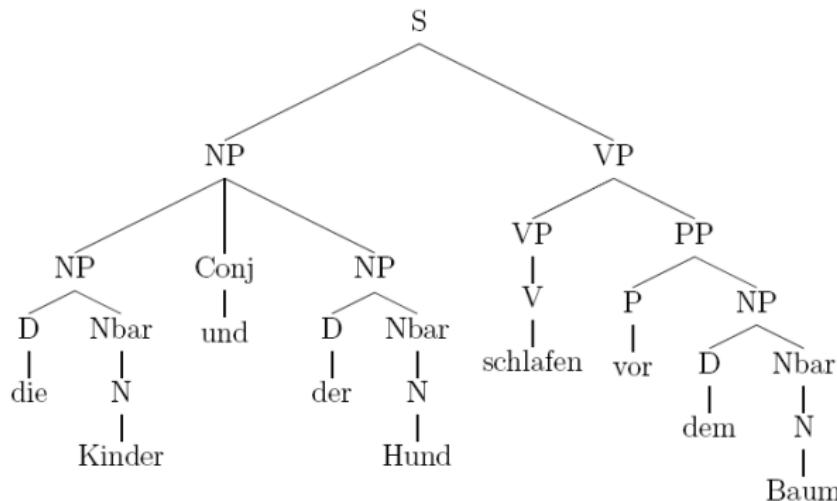
# Backtracking

Welches Problem haben einfache Parser mit Backtracking?

- Bei einem einfachen Parser mit Backtracking wird ein und dieselbe Kette von Wörtern immer wieder neu verarbeitet, wenn nach einer falschen Entscheidung Backtracking nötig wird
- Dabei kommt es auch vor, dass mehrfach die gleiche Analyse für Teilketten angenommen wird

# Backtracking: Beispiel

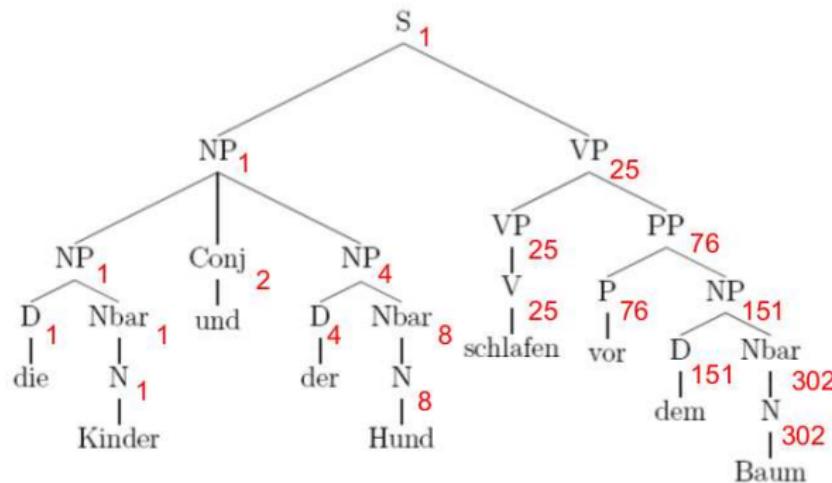
*Die Kinder und der Hund schlafen vor dem Baum*



■ Korrekte Analyse

# Backtracking: Beispiel

*Die Kinder und der Hund schlafen vor dem Baum*



- Die Zahlen geben an, wie oft eine bestimmte Teilkette des Satzes während des Parsings zu der jeweiligen Kategorie reduziert worden ist

# Backtracking: Beispiel

- Das heißt, dass z.B. die Wortfolge *dem Baum* 151x wieder von Neuem als NP analysiert worden ist
- Und dass die VP *schlafen vor dem Baum* insgesamt 25mal abgeleitet worden ist
- Statt sich also einmal zu merken, dass *dem Baum* als NP analysiert werden kann, macht der Parser die gleiche Arbeit immer wieder von Neuem
- Die Verarbeitungszeit beim Parsing mit einem naiven Shift-Reduce-Parser (s. Appendix) mit Backtracking wächst **exponentiell** mit der Länge der Eingabekette

# Effizientes Backtracking

Wie könnte man dieses Problem vermeiden?

- Der Parser sollte sich merken, welche Teilketten er schon analysiert hat und zu welchen Kategorien diese reduziert werden können
- Im Folgenden werden wir einen Parsing-Algorithmus betrachten, die Zwischenergebnisse für Teilketten in einer Tabelle (**Chart**) speichern und so Mehrfachableitungen vermeiden (**dynamische Programmierung**)

# Parsing-Algorithmen

- Ein naiver (ineffizienter) Algorithmus: Shift-Reduce-Parser (s. Appendix)
- Ein effizienter Algorithmus:  
Cocke-Younger-Kasami-Algorithmus (CYK)
- Ein weiterer effizienter Algorithmus: Earley-Algorithmus (s. Appendix)

# Gliederung

## 1 Parsing

- Backtracking

## 2 Cocke-Younger-Kasami-Parser

## 3 Appendix

- Shift-Reduce-Parser
- Earley-Parser

# Cocke-Younger-Kasami-Parser

- Beim CYK-Algorithmus werden die möglichen Kategorien von Teilketten in einer zweidimensionalen Tabelle (Chart) gespeichert
- Wenn die Eingabekette  $n$  Wörter enthält, hat die Tabelle die Größe  $n \times n$ .
- Zelle  $[x, y]$ : steht für die Teilkette von Wort  $x$  bis Wort  $y$

<i>Die</i>	<i>Kinder</i>	<i>und</i>	<i>der</i>	<i>Hund</i>	<i>schlafen</i>	<i>vor</i>	<i>dem</i>	<i>Baum</i>
1	2	3	4	5	6	7	8	9

- Beispiele:
  - $[1,1]$  steht für die Kette von Wort 1 bis Wort 1, also für *Die*
  - $[1,5]$  steht für die Kette von Wort 1 bis Wort 5, also für *Die Kinder und der Hund*
  - $[6,9]$  steht für die Kette von Wort 6 bis Wort 9, also für *schlafen vor dem Baum*

# Cocke-Younger-Kasami-Parser

- Die Teilkette [3,5] entspricht der von [5,3] (*und der Hund*)  
→ nur eine Hälfte der Tabelle ist auszufüllen
- Der Parser füllt die Tabelle bottom-up aus
- ... und zwar so, dass sukzessive immer längere Teilketten betrachtet werden:
  - erst alle Teilketten der Länge 1
  - dann alle Teilketten der Länge 2
  - ...
  - dann die komplette Eingabekette der Länge  $n$
- Initialisiert wird die Tabelle mit den möglichen Kategorien der einzelnen Wörter in den Zellen  $[1, 1], [2, 2], [3, 3], \dots, [n, n]$  (d.h. die Diagonale wird ausgefüllt)
- Lexikalische Ambiguität: Falls ein Wort mehrere mögliche Kategorien hat, werden diese alle in die zu dem Wort gehörenden Zelle eingetragen

# Initialisierung

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D								
2 Kinder		N							
3 und			Conj						
4 der				D					
5 Hund					N				
6 schlafen						V			
7 vor							P		
8 dem								D	
9 Baum									N

## CYK-Algorithmus

Der CYK-Algorithmus sucht jeweils für die Kette  $[x, y]$  nach allen möglichen Arten, wie sich diese aus zwei Teilen  $[x, m]$  und  $[m + 1, y]$  zusammensetzen kann

- Z.B. kann man *die Kinder und der Hund* [1,5] auf folgende Arten aufteilen:
  - die [1,1] + Kinder und der Hund [2,5]
  - die Kinder [1,2] + und der Hund [3,5]
  - die Kinder und [1,3] + der Hund [4,5]
  - die Kinder und der [1,4] + Hund [5,5]
- Das sind auch alle möglichen Konstituenten, aus denen die gesamte Phrase *die Kinder und der Hund* bestehen kann
- Der CYK-Algorithmus kann daher nur mit binären Regeln umgehen (in **Chomsky-Normalform**)!

# Einschub: Grammatik in Chomsky-Normalform (CNF)

- Grammatik in CNF enthält ausschließlich Regeln der Form
  - $A \rightarrow BC$  (2 Nichtterminale auf der rechten Seite)
  - $A \rightarrow w$  (1 Terminal)
- d.h. Eliminierung von Epsilon-Regeln, ternären Regeln und Kettenregeln (= Regeln der Form  $A \rightarrow B$ )

# Beispiel Regel-Umwandlung: ternäre Regel

- Originale Regel:  
 $S \rightarrow S \text{ Conj } S$
- Füge neue “Zwischenkategorie” ein: “Conj+S”
- Neue Regeln in CNF:  
 $S \rightarrow S \text{ Conj+S}$   
 $\text{Conj+S} \rightarrow \text{Conj } S$

# Bsp-Grammatik in Chomsky-Normalform

$S \rightarrow NP\ VP$

$S \rightarrow N\ VP$

$S \rightarrow N\ V$

$S \rightarrow NE\ VP$

$S \rightarrow NE\ V$

$S \rightarrow PRON\ VP$

$S \rightarrow PRON\ V$

$S \rightarrow S\ Conj+S$

$Conj+S \rightarrow Conj\ S$

$VP \rightarrow V\ NP$

$VP \rightarrow V\ PP$

$VP \rightarrow V\ NP+NP$

$VP \rightarrow VP\ PP$

$VP \rightarrow VP\ Conj+VP$

$VP \rightarrow V\ Conj+VP$

$Conj+VP \rightarrow Conj\ VP$

$Conj+VP \rightarrow Conj\ V$

$NP+NP \rightarrow NP\ NP$

$NP \rightarrow D\ N$

$N \rightarrow AP\ N$

$N \rightarrow A\ N$

$N \rightarrow N\ PP$

$NP \rightarrow NP\ Conj+NP$

$NP \rightarrow N\ Conj+NP$

$NP \rightarrow NE\ Conj+NP$

$NP \rightarrow PRON\ Conj+NP$

$Conj+NP \rightarrow Conj\ NP$

$Conj+NP \rightarrow Conj\ N$

$Conj+NP \rightarrow Conj\ NE$

$Conj+NP \rightarrow Conj\ PRON$

$PP \rightarrow P\ NP$

$PP \rightarrow P\ N$

$PP \rightarrow P\ NE$

$PP \rightarrow P\ PRON$

$PP \rightarrow PP\ Conj+PP$

$Conj+PP \rightarrow Conj\ PP$

$AP \rightarrow Adv\ AP$

$AP \rightarrow Adv\ A$

$AP \rightarrow AP\ Conj+AP$

$AP \rightarrow A\ Conj+AP$

$Conj+AP \rightarrow Conj\ AP$

$Conj+AP \rightarrow Conj\ A$

$Conj \rightarrow und$

$D \rightarrow der$

$D \rightarrow die$

$D \rightarrow dem$

$N \rightarrow Hund$

$N \rightarrow Kinder$

$N \rightarrow Baum$

$P \rightarrow vor$

$V \rightarrow schlafen$

...

# CYK-Algorithmus: 1. Initialisierung

Für  $l = 1 \dots n$

Für jede Regel  $A \rightarrow w \in G$

Falls  $w = w_l$ , setze  $[l, l] := [l, l] \cup \{A\}$

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D								
2 Kinder		N							
3 und			Conj						
4 der				D					
5 Hund					N				
6 schlafen						V			
7 vor							P		
8 dem								D	
9 Baum									N

Beispiel:  $D \in [1, 1]$ , da:  $D \rightarrow \text{die}$ ,  $w_1 = \text{die}$

# CYK-Algorithmus: 2. Induktion

Für  $l = 2 \dots n$

(für jeden Teilstring der Länge  $l$ )

Für  $i = 1 \dots (n - l + 1)$

(für alle entsprechenden Zellen  $[i,j]$ )

Setze  $j := i + l - 1$

Für  $k = i \dots i + l - 2$

(für alle "Töchterzellen"  $[i,k], [k+1,j]$ )

Setze  $[i,j] := [i,j] \cup \{A \mid A \rightarrow BC \in G,$

$B \in [i,k], C \in [k+1,j]\}$

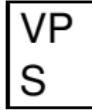
	1 die	2 Kinder	3 und	4 c
1 die	D [1,1]	NP [1,2]	[1,3]	
2 Kinder		N [2,2]	[2,3]	
3 und			Conj [3,3]	
A				r

- Beispiel:  $NP \in [1, 2]$ , da:  $NP \rightarrow D \ N$  und  $D \in [1, 1]$ ,  $N \in [2, 2]$
- Die auszufüllende Zelle ist immer der Schnittpunkt zwischen den beiden zu kombinierenden Zellen

## CYK-Algorithmus: 3. Terminierung

Wenn am Ende der Iteration gilt:  $S \in [1, n]$ , dann wird der Satz von der Grammatik erkannt bzw. erzeugt

# Zelleninhalt

- In jede Zelle der Chart können eine oder mehrere Kategorien eingetragen werden.
- Beispiel: Der Eintrag  in Zelle [1,6] bedeutet, dass die Wörter 1-6 zusammen entweder eine VP oder ein S bilden können

Auf den folgenden Folien wird ein Anwendungsbeispiel gezeigt

# CYK: Teilstrings der Länge 1 (Initialisierung)

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D								
2 Kinder		N							
3 und			Conj						
4 der				D					
5 Hund					N				
6 schlafen						V			
7 vor							P		
8 dem								D	
9 Baum									N

# CYK: Teilstrings der Länge 2

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D								
2 Kinder		N							
3 und			Conj						
4 der				D					
5 Hund					N				
6 schlafen						V			
7 vor							P		
8 dem								D	
9 Baum									N

# CYK: Teilstrings der Länge 3

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D	NP							
2 Kinder		N							
3 und			Conj						
4 der				D	NP				
5 Hund					N	S			
6 schlafen						V			
7 vor							P		
8 dem								D	NP
9 Baum									N

# CYK: Teilstrings der Länge 4

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D	NP							
2 Kinder		N							
3 und			Conj		Conj+NP				
4 der				D	NP	S			
5 Hund					N	S			
6 schlafen						V			
7 vor							P		PP
8 dem								D	NP
9 Baum									N

# CYK: Teilstrings der Länge 5

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D	NP							
2 Kinder		N			NP				
3 und			Conj		Conj+NP	Conj+S			
4 der				D	NP	S			
5 Hund					N	S			
6 schlafen						V			VP
7 vor							P		PP
8 dem								D	NP
9 Baum									N

# CYK: Teilstrings der Länge 6

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D	NP			NP				
2 Kinder		N			NP	S			
3 und			Conj		Conj+NP	Conj+S			
4 der				D	NP	S			
5 Hund					N	S			S
6 schlafen						V			VP
7 vor							P		PP
8 dem								D	NP
9 Baum									N

# CYK: Teilstrings der Länge 7

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D	NP			NP	S			
2 Kinder		N			NP	S			
3 und			Conj		Conj+NP	Conj+S			
4 der				D	NP	S			S
5 Hund					N	S			S
6 schlafen						V			VP
7 vor							P		PP
8 dem								D	NP
9 Baum									N

# CYK: Teilstrings der Länge 8

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D	NP			NP	S			
2 Kinder		N			NP	S			S
3 und			Conj		Conj+NP	Conj+S			Conj+S
4 der				D	NP	S			S
5 Hund					N	S			S
6 schlafen						V			VP
7 vor							P		PP
8 dem								D	NP
9 Baum									N

# CYK: Strings der Länge 9 (= kompletter Satz)

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D	NP			NP	S			
2 Kinder		N			NP	S			S
3 und			Conj		Conj+NP	Conj+S			Conj+S
4 der				D	NP	S			S
5 Hund					N	S			S
6 schlafen						V			VP
7 vor							P		PP
8 dem								D	NP
9 Baum									N

# CYK-Parser: Recognizing vs. Parsing

## ■ Recognizer

- Ein Satz wird dann von einem CYK-Parser als grammatisch erkannt, wenn beim Parsing (mindestens einmal) das Startsymbol in der rechten oberen Ecke  $[1, n]$  eingetragen wird, d.h. die gesamte Kette der Länge  $n$  auf das Startsymbol reduziert werden kann

## ■ Parser

- Alle Derivationen, die zum Startsymbol in  $[1, n]$  geführt haben, müssen aus der Chart extrahiert werden
- Die Information über die Derivation ist aber erstmal nicht in der Chart enthalten!

# CYK-Parser: Referenzen

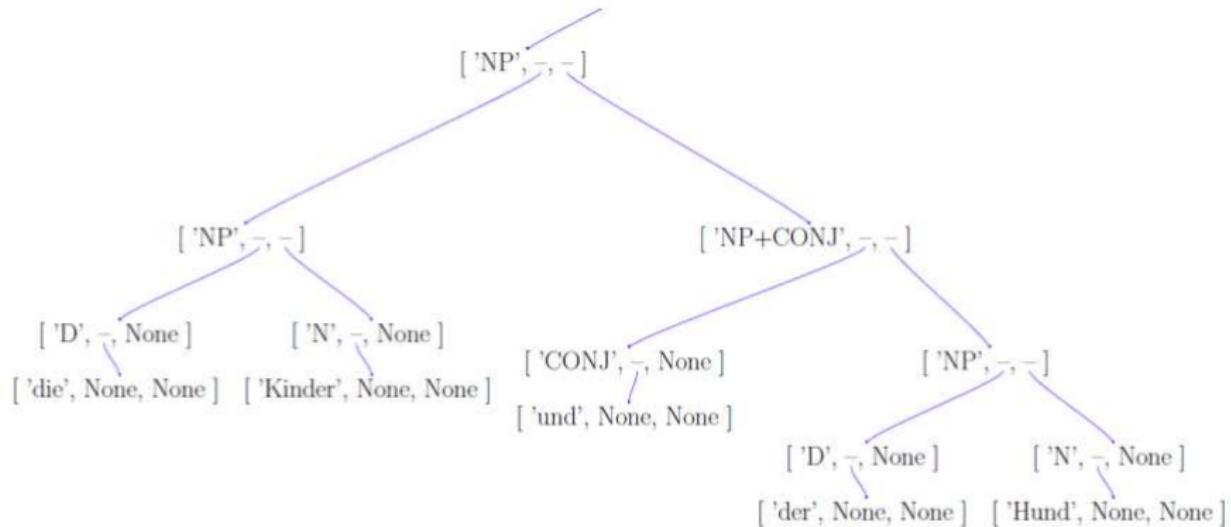
- Wenn am Ende alle möglichen Baumstrukturen aus der Chart ausgelesen werden sollen (**Parsing**), muss für jede eingetragene Kategorie zusätzlich gespeichert werden, aus welchen Subkonstituenten sie zusammengebaut worden ist
- Eine Möglichkeit besteht darin, statt einer einzelnen Kategorie eine Liste mit der Kategorie und Zeigern (Referenzen) auf ihre linke und rechte Subkonstituente einzutragen

## CYK-Parser: Referenzen

- Beispiel: Wenn aus der Kombination einer NP in Zelle [1,3] und einer VP in Zelle [4,8] ein S in Zelle [1,8] eingefügt worden ist, könnte der Eintrag schematisch wie folgt aussehen (der Pfeil steht für eine Referenz):  
(‘S’, → NP[1,3], → VP[4,8])
- Wichtig: ein und dieselbe Kategorie in ein und derselben Zelle kann sich auf mehrere Arten zusammensetzen; daher kann auch eine Zelle mehrere Einträge mit der gleichen Kategorie enthalten, z.B.

(‘S’, → NP[1,3], → VP[4,8])  
(‘S’, → NP[1,5], → VP[6,8])

# CYK-Parser: Parse mit Referenzen



# CYK-Parser: Auslesen der Parses

- Zum Auslesen kann ein rekursiver Algorithmus verwendet werden, der ausgehend von allen Einträgen der Kategorie des Startsymbols in der Zelle  $[1, n]$  deren Unterkonstituenten ermittelt, danach deren Bestandteile usw.
- Wenn es mehr als eine Derivation gibt, die zu einem Startsymbol in  $[1, n]$  führt, dann ist der geparsete Satz strukturell ambig

# Anwendungsbeispiel mit Referenzen

	1 die	2 Kinder	3 und	4 der	5 Hund	6 schlafen	7 vor	8 dem	9 Baum
1 die	D	NP		NP	S			S	
2 Kinder		N		NP	S			S	
3 und			Conj	Conj+NP	Conj+S			Conj+S	
4 der				D	NP	S		S	
5 Hund				N	S			S	
6 schlafen					V			VP	
7 vor						P	PP		
8 dem							D	NP	
9 Baum							N		

In der Derivation benutzte Zellen:  
 Strukturbaum rekonstruierbar...

# Zusammenfassung: Cocke-Younger-Kasami-Algorithmus

## Vorteile:

- speichert Zwischenergebnisse in einer Chart (Tabelle)
- einfach zu implementieren
- sehr effizient:
  - Zeitkomplexität:  $O(n^3)$
  - Speicherbedarf:  $O(n^2)$
  - Achtung: Auslesen der Parses kostet zusätzlich!  
(dagegen: naiver Parser: exponentieller Aufwand)

## Nachteile

- Grammatik muss in Chomsky-Normalform vorliegen
  - jede kontextfreie Grammatik kann automatisch in die Chomsky-Normalform überführt werden
  - aber: die resultierenden Strukturen sind dann oft linguistisch nicht mehr besonders plausibel

# Gliederung

## 1 Parsing

- Backtracking

## 2 Cocke-Younger-Kasami-Parser

## 3 Appendix

- Shift-Reduce-Parser
- Earley-Parser

# Gliederung

## 1 Parsing

- Backtracking

## 2 Cocke-Younger-Kasami-Parser

## 3 Appendix

- Shift-Reduce-Parser
- Earley-Parser

# Shift-Reduce-Parser

- Ein **Shift-Reduce-Parser** ist ein einfaches Bottom-Up-Parsing-Verfahren
- Dieses Verfahren benötigt nur zwei Operationen (Shift, Reduce) und einige wenige Datenstrukturen
- Datenstrukturen
  - Liste mit den Eingabetoken [sieht, den, Baum]
  - Stapel (Stack) mit den bereits hergeleiteten Kategorien [D, Hund] ← obere Seite

# Elementare Operationen: Shift

Nimm das nächste Token von der Eingabeliste und lege es oben auf den Stapel

- *Eingabeliste*: [sieht, den, Baum] > [den, Baum]  
*Stapel*: [D, Hund] > [D, Hund, sieht]

## Elementare Operationen: Reduce

Falls die letzten  $n$  Symbole auf dem Stapel zu der rechten Seite einer Regel der Grammatik passen, nimm sie vom Stapel und lege dafür das Symbol der linken Regelseite auf den Stapel

- *Regel:*  $N \rightarrow \text{Hund}$   
*Stapel:* [D, Hund]  $>$  [D, N]
- *Regel:*  $\text{NP} \rightarrow \text{D N}$   
*Stapel:* [D, N]  $>$  [NP]

# Terminierung

Es gibt eine mögliche Struktur für den Satz, wenn:

- alle Symbole auf der Eingabeliste aufgebraucht sind, d.h.  
die Eingabeliste leer ist:  
*Eingabeliste*: [ ]
- und als einziges Symbol das Startsymbol auf dem Stapel  
steht:  
*Stapel*: [S]

# Recognizer vs. Parser

- Ein **Recognizer** testet lediglich, ob ein Satz von einer Grammatik erzeugt werden kann oder nicht
- Ein **Parser** gibt auch alle möglichen Ableitungen des Satzes nach der Grammatik zurück
  - die einzelnen Ableitungen müssen als weitere Datenstruktur (z.B. als rekursiver Baum) bei der Analyse aufgebaut werden

## Beispiel: *das Mädchen sieht den Jungen*

- *Eingabesymbole:* [das, Mädchen, sieht, den, Jungen]
- *Stapel:* [ ]
- *Baum:*

## Beispiel: *das Mädchen sieht den Jungen*

*Operation:* Shift

- *Eingabesymbole:* [Mädchen, sieht, den, Jungen]
- *Stack:* [das]
- *Baum:* das

das

## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $D \rightarrow \text{das}$

- Eingabesymbole: [Mädchen, sieht, den, Jungen]
- Stapel: [D]
- Baum: [.D das ]



## Beispiel: *das Mädchen sieht den Jungen*

*Operation:* Shift

- *Eingabesymbole:* [sieht, den, Jungen]
- *Stapel:* [D, Mädchen]
- *Baum:* [.D das ] Mädchen

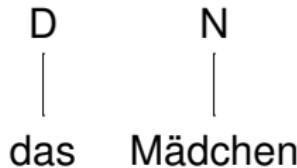


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $N \rightarrow \text{Mädchen}$

- Eingabesymbole: [sieht, den, Jungen]
- Stapel: [D, N]
- Baum: [.D das ] [.N Mädchen ]

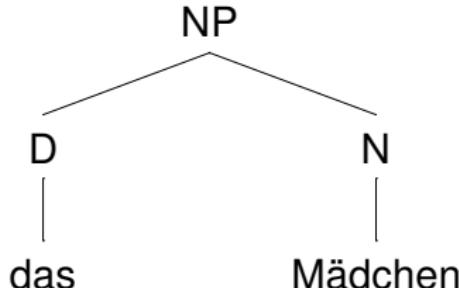


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $NP \rightarrow D\ N$

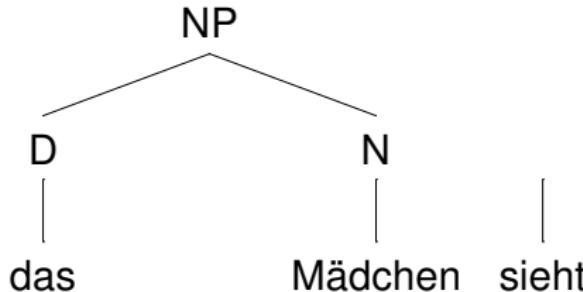
- Eingabesymbole: [sieht, den, Jungen]
- Stapel: [NP]
- Baum: [.NP [.D das ] [.N Mädchen ]]



## Beispiel: *das Mädchen sieht den Jungen*

Operation: Shift

- Eingabesymbole: [den, Jungen]
- Stapel: [NP, sieht]
- Baum: [.NP [.D das ] [.N Mädchen ]] sieht

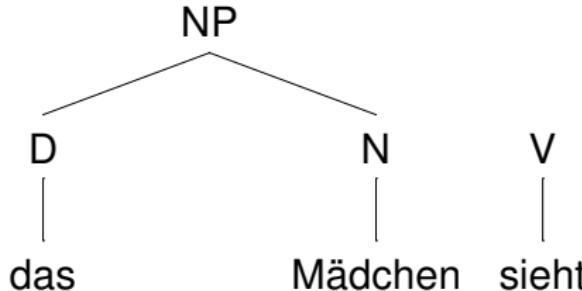


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $V \rightarrow \text{sieht}$

- Eingabesymbole: [den, Jungen]
- Stapel: [NP, V]
- Baum: [.NP [.D das ] [.N Mädchen ]] [.V sieht ]

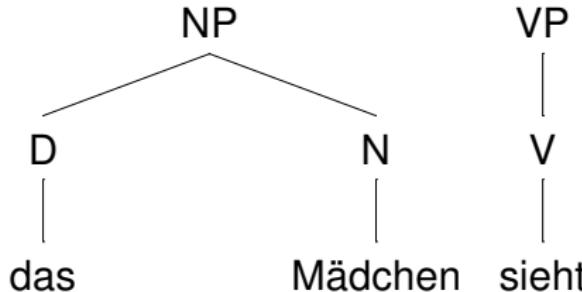


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $VP \rightarrow V$

- Eingabesymbole: [den, Jungen]
- Stapel: [NP, VP]
- Baum: [.NP [.D das ] [.N Mädchen ]] [.VP [.V sieht ]]

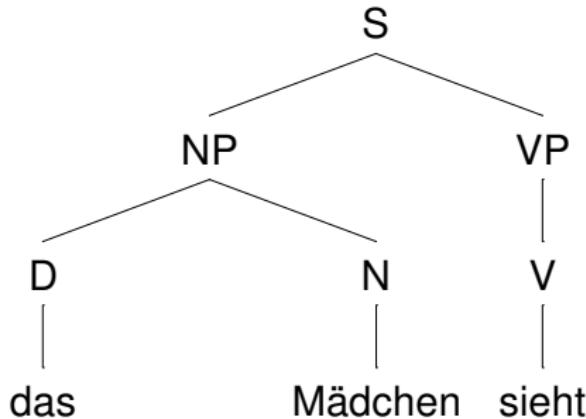


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $S \rightarrow NP\ VP$

- Eingabesymbole: [den, Jungen]
- Stapel: [S]
- Baum: [.S [.NP [.D das ] [.N Mädchen ]] [.VP [.V sieht ]]]



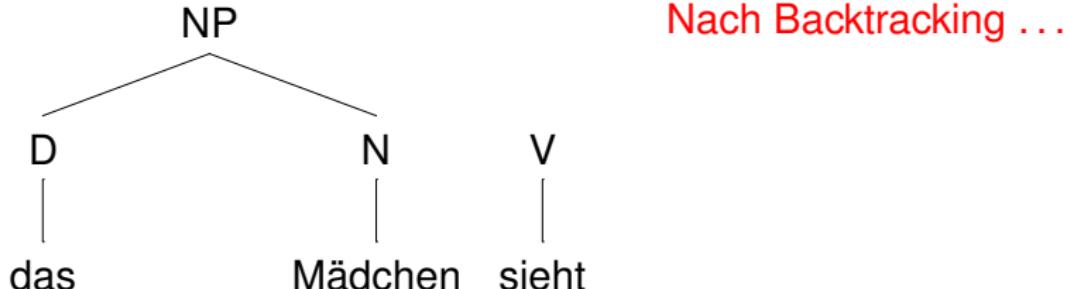
Sackgasse! Backtracking ist notwendig! Auf dem Stapel steht zwar [S], aber die Liste der Eingabesymbole ist nicht leer!

## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $V \rightarrow \text{sieht}$

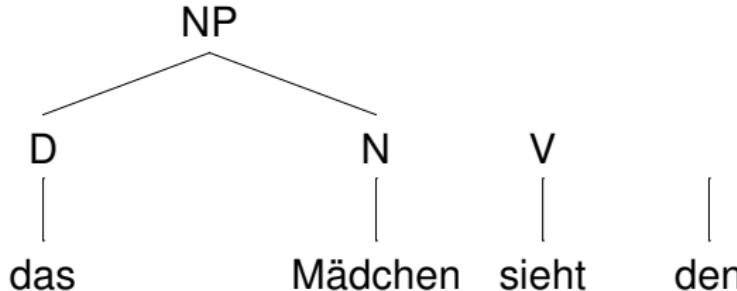
- Eingabesymbole: [den, Jungen]
- Stapel: [NP, V]
- Baum: [.NP [.D das ] [.N Mädchen ]] [.V sieht ]



## Beispiel: *das Mädchen sieht den Jungen*

Operation: Shift

- Eingabesymbole: [Jungen]
- Stapel: [NP, V, den]
- Baum: [.NP [.D das] [.N Mädchen]] [.V sieht] den

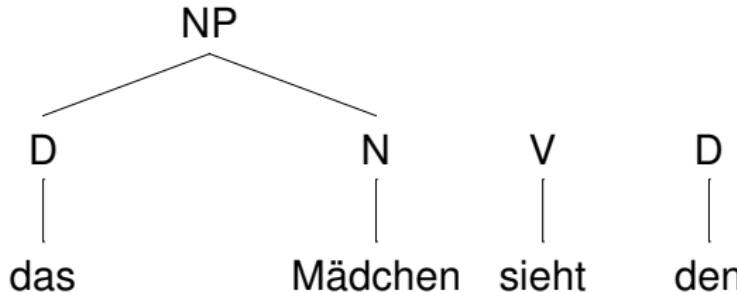


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $D \rightarrow \text{den}$

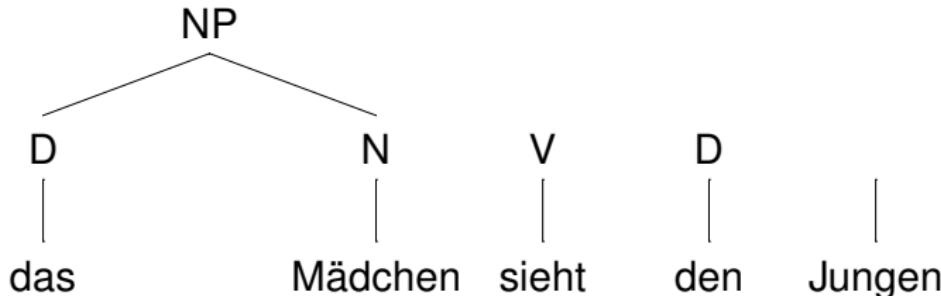
- Eingabesymbole: [Jungen]
- Stapel: [NP, V, D]
- Baum: [.NP [.D das] [.N Mädchen]] [.V sieht] [.D den]



## Beispiel: *das Mädchen sieht den Jungen*

Operation: Shift

- Eingabesymbole: [ ]
- Stapel: [NP, V, D, Jungen]
- Baum: [.NP [.D das ] [.N Mädchen ]] [.V sieht ] [.D den ]  
Jungen

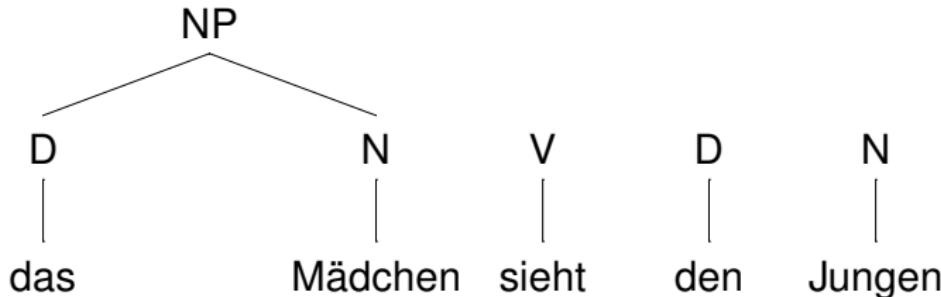


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $N \rightarrow \text{Jungen}$

- Eingabesymbole: [ ]
- Stapel: [NP, V, D, N]
- Baum: [.NP [.D das] [.N Mädchen]] [.V sieht] [.D den]  
[.N Jungen]

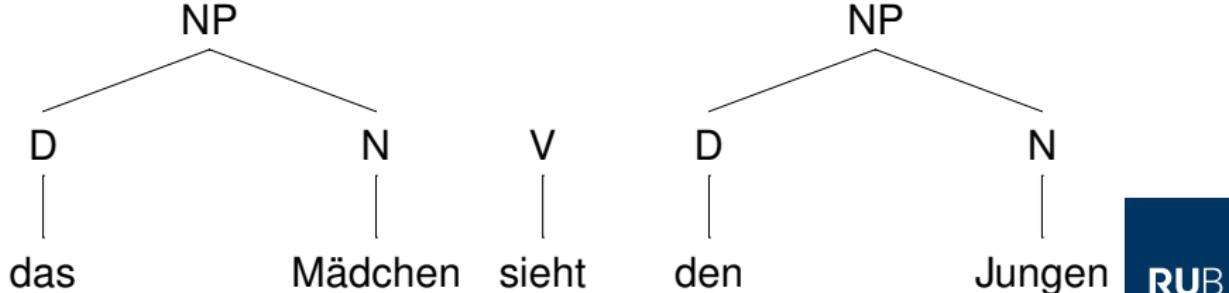


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel: NP → D N

- Eingabesymbole: [ ]
- Stapel: [NP, V, NP]
- Baum: [.NP [.D das ] [.N Mädchen ]] [.V sieht ] [.NP [.D den ] [.N Jungen ]]

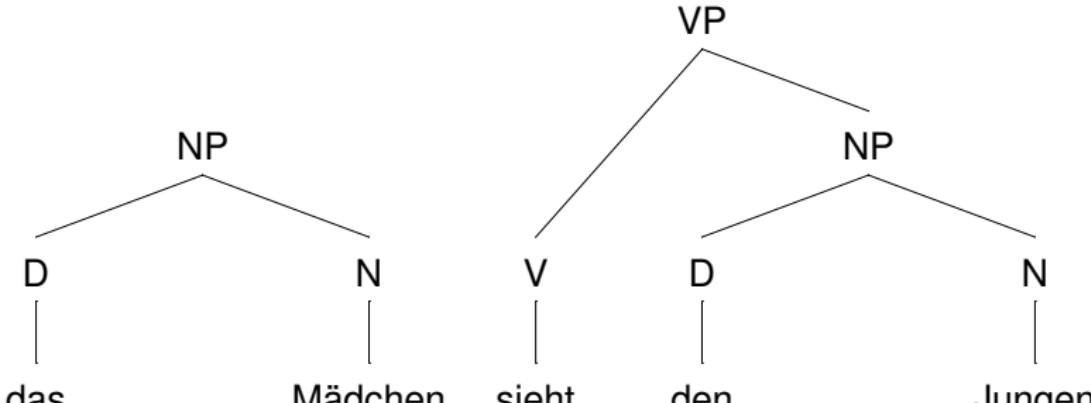


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $VP \rightarrow V\ NP$

- Eingabesymbole: [ ]
- Stapel: [NP, VP]
- Baum: [.NP [.D das] [.N Mädchen]] [.VP [.V sieht] [.NP [.D den] [.N Jungen]]]

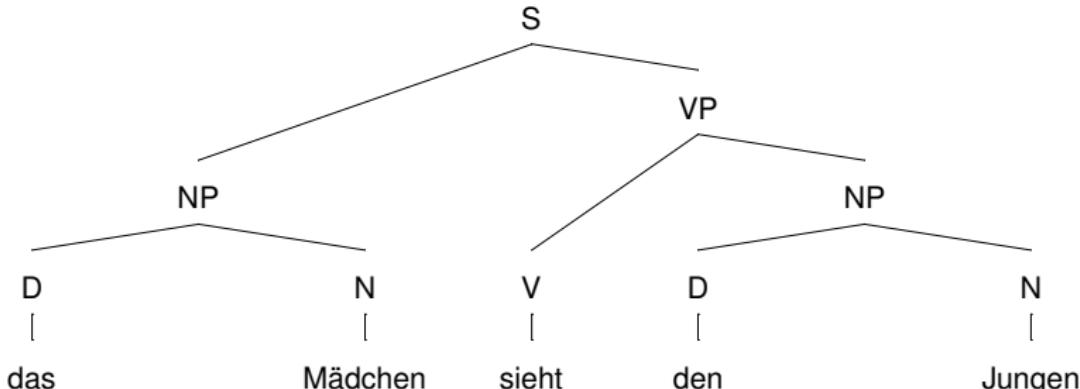


## Beispiel: *das Mädchen sieht den Jungen*

Operation: Reduce

Regel:  $S \rightarrow NP\ VP$

- Eingabesymbole: [ ]
- Stapel: [S]
- Baum: [.S [.NP [.D das] [.N Mädchen]] [.VP [.V sieht] [.NP [.D den] [.N Jungen]]]]]



# Im Beispiel: welche Strategien?

1. Analyserichtung (vertikal): bottom-up
2. Verarbeitungsrichtung (horizontal): links-rechts
3. Alternativen: Bei mehreren passenden Regeln: welche Regel wird zuerst angewandt?  
n.a. (keine relevanten Alternativen)
4. Suchstrategie: depth-first (Reduce bevorzugt)
5. Exhaustivität: nein

Anmerkung: Stapel nur oben (= von rechts) zugänglich, daher Bevorzugung von Reduce und damit depth-first-Strategie

# Backtracking

- Sowohl ein Shift-Reduce-Recognizer als auch ein Shift-Reduce-Parser benötigen Backtracking
- Wie kann man Backtracking praktisch implementieren?
  - zweiten Stapel mit Entscheidungspunkten anlegen:  
*Entscheidungspunkte:* [ ] ← obere Seite
  - immer wenn man mehrere Auswahlmöglichkeiten hat:
    - entweder Shift oder Reduce
    - oder: verschiedene Möglichkeiten, Reduce anzuwenden
  - ... dann wählt man eine Möglichkeit aus und merkt sich die Alternativen als Entscheidungspunkt

# Backtracking: Implementation

- *Eingabesymbole*: [Jungen]  
*Stapel*: [NP, V, den]  
*Baum*: [.NP [.D das] [.N Mädchen]] [.V sieht] den
- *Mögliche Operationen*:
  - Shift oder Reduce: D → den
  - wähle z.B. Shift
- Speichere den Entscheidungspunkt und den Zustand des Parsers zu diesem Punkt

```
Entscheidungspunkt1 =  
[ Eingabesymbole: [Jungen],  
  Stapel: [NP, V, den],  
  Baum: [.NP [.D das][.N Mädchen]][.V sieht] den,  
  Alternativen: [Reduce D → den]  
 ]
```

# Backtracking: Implementation

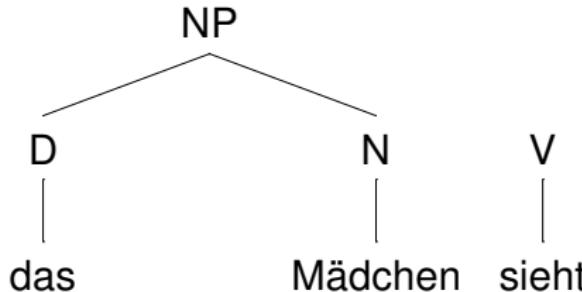
- Lege den Entscheidungspunkt auf den Stapel aller bisherigen Entscheidungspunkte
  - Entscheidungspunkte = [ . . . , Entscheidungspunkt1]       $\leftarrow$  obere Seite
- Backtracking: falls der Parser in eine Sackgasse gerät
  - nimm den letzten Entscheidungspunkt vom Stapel der Entscheidungspunkte
  - ersetze die gegenwärtigen Datenstrukturen durch die gespeicherten:
    - Eingabesymbole
    - Stapel
    - Baum
  - wähle eine der alternativen Operationen aus und setze das Parsing fort

# Implementation: Beispiel

*Operation:* Reduce / Shift

*Regel:* VP → V

- *Eingabesymbole:* [den, Jungen]
- *Stapel:* [NP, V]
- *Baum:* [.NP [.D das ] [.N Mädchen ]] [.V sieht ]

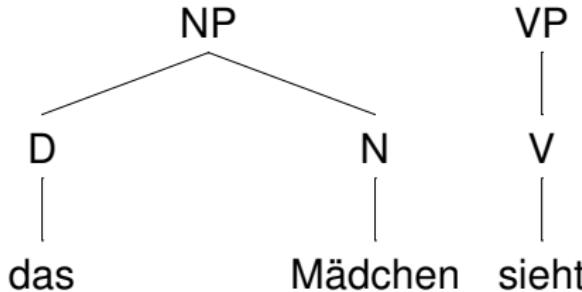


# Implementation: Beispiel

*Operation:* wähle Reduce und speichere Shift

*Regel:* VP → V

- *Entscheidungspunkt:*  
[[den, Jungen],[NP, V], [ Baum (s.o.) ], [Shift]]
- *Eingabesymbole:* [den, Jungen]
- *Stack:* [NP, VP]
- *Baum:* [.NP [.D das ] [.N Mädchen ]] [.VP [.V sieht ]]

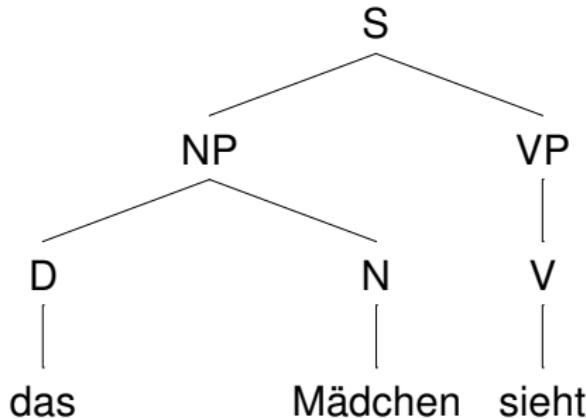


# Implementation: Beispiel

*Operation:* Reduce (Shift-Alternativen hier ignoriert)

*Regel:*  $S \rightarrow NP\ VP$

- *Eingabesymbole:* [den, Jungen]
- *Stapel:* [S]
- *Baum:* [.S [.NP [.D das ] [.N Mädchen ]] [.VP [.V sieht ]]]



Sackgasse! Backtracking ist notwendig! Auf dem Stapel steht zwar [S], aber die Liste der Eingabesymbole ist nicht leer

# Implementation: Beispiel

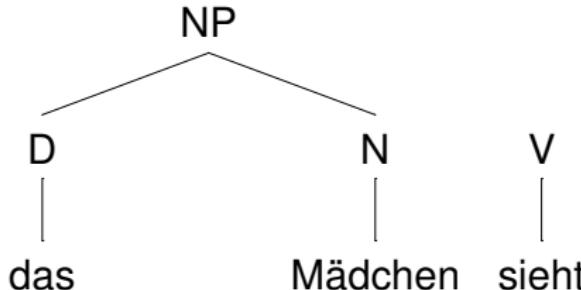
*Operation:* Backtracking: hole letzten Entscheidungspunkt vom Stapel und stelle vorherigen Zustand wieder her

- *Entscheidungspunkt:*

[[den, Jungen],[NP, V], [ Baum ], [**Shift**]]

**Shift: einzige alternative Operation an diesem Punkt**

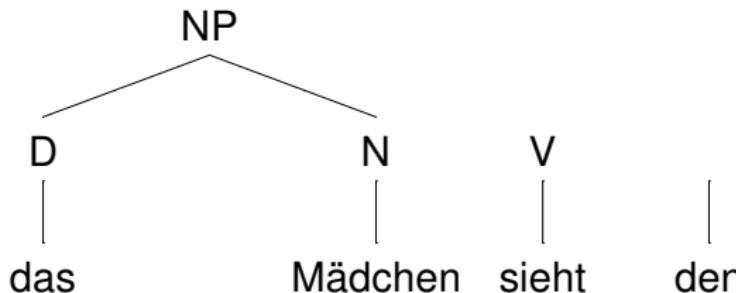
- *Eingabesymbole:* [den, Jungen]
- *Stapel:* [NP, V]
- *Baum:* [.NP [.D das ] [.N Mädchen ]] [.V sieht ]



# Implementation: Beispiel

*Operation:* Shift

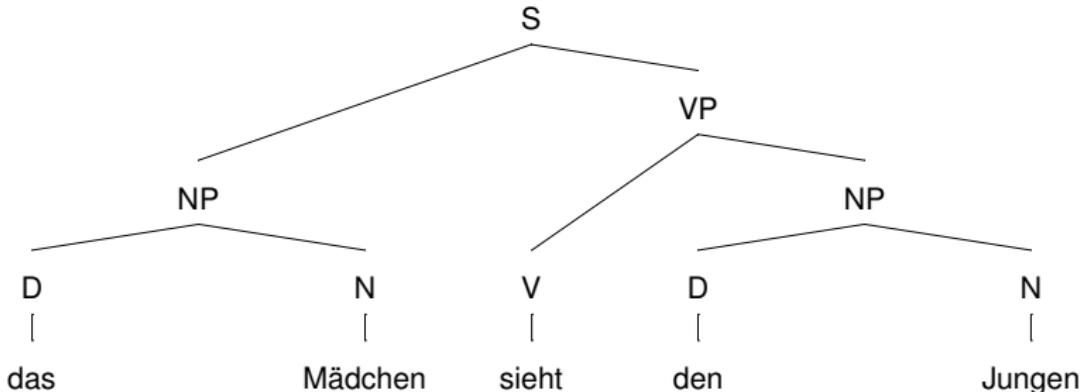
- *Eingabesymbole:* [Jungen]
- *Stapel:* [NP, V, den]
- *Baum:* [.NP [.D das ] [.N Mädchen ]] [.V sieht ] den



# Implementation: Beispiel

Setze das Parsing bis zum Ende fort:

- *Eingabesymbole*: [ ]
- *Stapel*: [S]
- *Baum*: [.S [.NP [.D das] [.N Mädchen]] [.VP [.V sieht] [.NP [.D den] [.N Jungen]]]]



# Zusammenfassung I

- Parsing: syntaktische Strukturen bestimmen
  - Kontextfreie Grammatiken
  - Parsing-Strategien
    - Analyserichtung
    - Verarbeitungsrichtung
    - Alternativen
    - Suchstrategie
    - Exhaustivitt
  - Backtracking
  - Strukturelle Ambiguitt
- Shift-Reduce-Parser (bottom-up, depth-first, left-to-right)

# Gliederung

## 1 Parsing

- Backtracking

## 2 Cocke-Younger-Kasami-Parser

## 3 Appendix

- Shift-Reduce-Parser
- Earley-Parser

# Earley-Algorithmus

- Der Earley-Algorithmus (Earley 1970) ist eine flexiblere Alternative zum CYK-Parser
  - kann beliebige kontextfreie Grammatiken verarbeiten
    - dagegen: beim CYK-Parser muss die Grammatik zunächst in Chomsky-Normalform umgewandelt werden, da der Algorithmus nur mit binären Regeln funktioniert
  - ist trotzdem sehr effizient
- Beim Earley-Parser werden die schon analysierten Teilketten ebenfalls in einer Tabelle (Chart) gespeichert
- Die Tabelle ist jedoch eindimensional und besteht aus  $n + 1$  (komplexen) Zellen:
  - Zelle [0] beschreibt den Zustand des Parsers vor dem ersten Wort
  - Zelle [5] beschreibt den Zustand des Parsers nach dem fünften Wort
    - die • Kinder • und • der • Hund • schlafen • vor • dem • Baum •

0    1    2    3    4    5    6    7    8    9

# Struktur der Tabelle

- Im Gegensatz zum CYK-Parser speichert der Earley-Parser die Positionen und Längen der gefundenen Teilbäume nicht mittels der Struktur der Tabelle, sondern in den einzelnen Einträgen in den Zellen
- Ein Zelleneintrag besteht aus
  - 1 einer sog. "dotted rule"
  - 2 Angaben zur Position der Konstituente
  - 3 einem Zeiger auf den vorherigen Zustand (zum späteren Auslesen der geparsten Struktur)
- Der Earley-Parser verwendet hauptsächlich Top-Down-Voraussagen beim Parsing

## Dotted Rules

Ein Punkt zeigt an, welcher Teil einer Regel schon gefunden und verarbeitet worden ist und welcher Teil noch auftreten muss, um die Kategorie auf der linken Seite komplett zu machen:

■  $S \rightarrow \bullet NP VP$

Es wird top-down ein S vorausgesagt; von diesem ist noch keine Konstituente gefunden worden

■  $VP \rightarrow V \bullet NP$

Es wurde eine VP vorausgesagt, von der auch schon das Verb gefunden worden ist

■  $NP \rightarrow Det A N \bullet$

Es ist eine NP erwartet und auch schon komplett gefunden worden

# Position der Konstituente

Zusätzlich wird der von der Regel abgedeckte Teil der Eingabekette angegeben:

- $S \rightarrow \bullet NP VP [0,0]$

Kategorie S beginnt am Anfang der Kette, deckt aber noch keine Wörter ab

- $VP \rightarrow V \bullet NP [5,6]$

Der schon gefundene Teil der VP, nämlich V, erstreckt sich von Position 5 bis Position 6 (*schlafen*)

- $S \rightarrow NP VP \bullet [0,9]$

S ist komplett gefunden worden und umfasst den ganzen Satz

• *die* • *Kinder* • *und* • *der* • *Hund* • *schlafen* • *vor* • *dem* • *Baum* •  
0      1      2      3      4      5      6      7      8      9

# Operationen

Der Earley-Parser verfügt über drei mögliche Operationen:

- 1 Predict:** Der Predictor macht Top-Down-Voraussagen darüber, in welche Konstituenten eine Kategorie expandieren kann
- 2 Scan:** Der Scanner liest ein Wort aus der Eingabekette ein, falls es identisch ist mit einem vom Predictor vorhergesagten Wort
- 3 Complete:** Der Completer verändert den Zustand aller Regeln, die eine gerade vervollständigte Konstituente vorhergesagt haben

# Beispielgrammatik

Eingabekette:

• die • Kinder • und • der • Hund • schlafen • vor • dem • Baum •  
0      1      2      3      4      5      6      7      8      9

## Regeln

$S \rightarrow NP VP$   
 $S \rightarrow S Conj S$   
 $VP \rightarrow V$   
 $VP \rightarrow V NP$   
 $VP \rightarrow V NP NP$   
 $VP \rightarrow VP PP$   
 $VP \rightarrow VP Conj VP$   
 $NP \rightarrow D Nbar$   
 $NP \rightarrow Nbar$   
 $NP \rightarrow NE$   
 $NP \rightarrow PRON$   
 $NP \rightarrow NP Conj NP$

$Nbar \rightarrow AP Nbar$   
 $Nbar \rightarrow Nbar PP$   
 $Nbar \rightarrow N$   
 $PP \rightarrow P NP$   
 $PP \rightarrow PP Conj PP$   
 $AP \rightarrow AdvP AP$   
 $AP \rightarrow A$   
 $AP \rightarrow AP Conj AP$   
 $AdvP \rightarrow Adv$

## Lexikon

$Conj \rightarrow und$   
 $D \rightarrow dem, der, die$   
 $N \rightarrow Baum, Kinder, Hund$   
 $P \rightarrow vor$   
 $V \rightarrow schlafen$   
... →

# Initialisierung [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Die Chart wird an Position 0 mit einer Dummy-Regel initialisiert, die das eigentliche Startsymbol vorhersagt

Position 0	Position 1
$S_1: \gamma \rightarrow \bullet S [0,0] []$	

# Predict

- Die Operation **Predict** schaut nach, ob an der gegenwärtigen Position in der Tabelle Regeln stehen, in denen **der Punkt vor einem Symbol** steht
  - z.B.:  $S \rightarrow \bullet NP VP [0,0]$
- Falls es solche Regeln gibt, fügt Predict alle in der Grammatik vorhandenen Regeln, die dieses Symbol expandieren, zur gegenwärtigen Position hinzu
  - es werden nur Regeln hinzugefügt, die nicht schon in der Zelle vorkommen (wobei Regeln nur dann identisch sind, wenn auch der Punkt und die Indizes übereinstimmen)
  - der Punkt steht dabei vor dem ersten Symbol auf der rechten Seite der Regel
  - als Start- und Endposition wird die gegenwärtige Position eingetragen
  - z.B. wird hinzugefügt:  
 $NP \rightarrow \bullet D N [0,0]$ ,  $NP \rightarrow \bullet PRON [0,0]$ ,  $D \rightarrow \bullet die [0,0]$  etc.

## Predict (formal)

Wenn die Chart bereits eine Kante der Form

$$A \rightarrow \alpha \bullet B \beta [i, j] \text{ (mit } i \leq j\text{)}$$

enthält, dann wird für jede Grammatikregel der Form

$$B \rightarrow \gamma \in R$$

ein neues Chart-item der Form

$$B \rightarrow \bullet \gamma [j, j]$$

angelegt.

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 0		Position 1
S <sub>1</sub> :	$\gamma \rightarrow \bullet S$	[0,0] []
S <sub>2</sub> :	$S \rightarrow \bullet NP VP$	[0,0] []
S <sub>3</sub> :	$S \rightarrow \bullet S Conj S$	[0,0] []
S <sub>4</sub> :	$NP \rightarrow \bullet D Nbar$	[0,0] []
S <sub>5</sub> :	$NP \rightarrow \bullet Nbar$	[0,0] []
S <sub>6</sub> :	$NP \rightarrow \bullet NE$	[0,0] []
S <sub>7</sub> :	$NP \rightarrow \bullet PRON$	[0,0] []
S <sub>8</sub> :	$NP \rightarrow \bullet NP Conj NP$	[0,0] []
S <sub>9</sub> :	$D \rightarrow \bullet dem$	[0,0] []
S <sub>10</sub> :	$D \rightarrow \bullet der$	[0,0] []
S <sub>11</sub> :	$D \rightarrow \bullet die$	[0,0] []
S <sub>12</sub> :	$Nbar \rightarrow \bullet AP Nbar$	[0,0] []
S <sub>13</sub> :	$Nbar \rightarrow \bullet Nbar PP$	[0,0] []
S <sub>14</sub> :	$Nbar \rightarrow \bullet N$	[0,0] []
S <sub>15</sub> :	$AP \rightarrow \bullet AdvP AP$	[0,0] []
S <sub>16</sub> :	$AP \rightarrow \bullet A$	[0,0] []
S <sub>17</sub> :	$AP \rightarrow \bullet AP Conj AP$	[0,0] []
S <sub>18</sub> :	$N \rightarrow \bullet Baum$	[0,0] []
S <sub>19</sub> :	$N \rightarrow \bullet Kinder$	[0,0] []
S <sub>20</sub> :	$N \rightarrow \bullet Hund$	[0,0] []

# Predict

Hinweis:

- Die vorgegebene Grammatik enthält nur wenige Terminalregeln (und am Ende: ... )
- Eigentlich würde Predict noch weitere Vorhersagen machen, z.B.:
  - NE → • ...
  - PRON → • ...
  - A → • ...
  - Adv → • ...
- Da es dazu keine explizit genannten Regeln gibt, wurden diese Einträge in der Tabelle ganz weggelassen

# Scan

- Die Operation **Scan** schaut für jede Regel an der gegenwärtigen Position, in der **der Punkt vor einem terminalen Symbol** steht, ob das Wort *nach der gegenwärtigen Position* mit diesem terminalen Symbol übereinstimmt
- Falls das nächste Wort mit dem vorhergesagten übereinstimmt, wird in der nächsten Zelle der Tabelle die entsprechende fertige präterminale Regel mit dem Punkt am Ende der rechten Seite eingetragen
- Beispiel:
  - gegeben:  $D \rightarrow \bullet \text{ die } [0,0]$
  - das nächste Wort ist **die**
  - füge die Regel  $D \rightarrow \text{die } \bullet [0,1]$  zur *nächsten* Zelle hinzu

## Scan (formal)

Wenn die Chart ein Item der Form

$$A \rightarrow \alpha \bullet w_j \beta [i, j - 1]$$

enthält, dann wird ein neues Item der Form

$$A \rightarrow \alpha w_j \bullet \beta [i, j]$$

angelegt.

# Scan [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 0

$S_1:$	$\gamma \rightarrow \bullet S$	[0,0] []
$S_2:$	$S \rightarrow \bullet NP VP$	[0,0] []
$S_3:$	$S \rightarrow \bullet S Conj S$	[0,0] []
$S_4:$	$NP \rightarrow \bullet D Nbar$	[0,0] []
$S_5:$	$NP \rightarrow \bullet Nbar$	[0,0] []
$S_6:$	$NP \rightarrow \bullet NE$	[0,0] []
$S_7:$	$NP \rightarrow \bullet PRON$	[0,0] []
$S_8:$	$NP \rightarrow \bullet NP Conj NP$	[0,0] []
$S_9:$	$D \rightarrow \bullet dem$	[0,0] []
$S_{10}:$	$D \rightarrow \bullet der$	[0,0] []
$S_{11}:$	$D \rightarrow \bullet die$	[0,0] []
$S_{12}:$	$Nbar \rightarrow \bullet AP Nbar$	[0,0] []
$S_{13}:$	$Nbar \rightarrow \bullet Nbar PP$	[0,0] []
$S_{14}:$	$Nbar \rightarrow \bullet N$	[0,0] []
$S_{15}:$	$AP \rightarrow \bullet AdvP AP$	[0,0] []
$S_{16}:$	$AP \rightarrow \bullet A$	[0,0] []
$S_{17}:$	$AP \rightarrow \bullet AP Conj AP$	[0,0] []
$S_{18}:$	$N \rightarrow \bullet Baum$	[0,0] []
$S_{19}:$	$N \rightarrow \bullet Kinder$	[0,0] []
$S_{20}:$	$N \rightarrow \bullet Hund$	[0,0] []

Position 1

$S_{22}: D \rightarrow die \bullet [0,1] []$   
 $\bullet die \bullet Kinder \bullet und \bullet der \bullet Hund \bullet \dots$   
 0 1 2 3 4 5

# Complete

- Die Operation **Complete** wird angewendet, sobald eine Regel an der gegenwärtigen Position der Tabelle komplett gefunden worden ist, d.h. der Punkt darin am Ende steht
- Dann wird in der Zelle, die dem linken Rand der gerade fertiggestellten Konstituente entspricht, nach nicht-kompletten Regeln gesucht, in denen der Punkt vor der gerade komplettierten Kategorie steht
- Wenn eine oder mehrere solcher Regeln gefunden worden sind, werden diese in die aktuelle Zelle kopiert, der Punkt wird über die fertiggestellte Kategorie bewegt und der rechte Rand wird auf den Index der aktuellen Zelle gesetzt
- Außerdem wird ein Zeiger auf die abgeschlossene Regel hinzugefügt, die die Complete-Operation ausgelöst hat (für das spätere Auslesen der Derivation)

# Complete

Beispiel:

- Gegeben:  $D \rightarrow \text{die} \bullet [0,1]$
- sowie die Regel  $NP \rightarrow \bullet D N [0,0]$ ,
- dann füge die Regel  $NP \rightarrow D \bullet N [0,1]$  hinzu

## Complete (formal)

Wenn die Chart bereits eine Kante der Form

$$A \rightarrow \alpha \bullet B \beta [i, j]$$

und eine weitere Kante der Form

$$B \rightarrow \gamma \bullet [j, k]$$

enthält, dann wird eine neues Item

$$A \rightarrow \alpha B \bullet \beta [i, k]$$

in die Chart eingetragen.

# Earley-Algorithmus

- 1 Initialisierung: Erzeuge einen Eintrag für das Startsymbol S der Form:

$$\gamma \bullet \rightarrow S [0, 0]$$

- 2 Erzeugung weiterer Chart-Kanten: Für alle Positionen  $i = 0, \dots, j$  und  $j = 0, \dots, n$ :
  - (a) wende PREDICT und COMPLETE auf alle Items mit der Startposition  $i$  und der Endposition  $j$  solange an, bis diese beiden Prozeduren keine neuen Kanten mehr erzeugen
  - (b) wende die Prozedur SCAN auf alle Items mit der Start- und Endposition  $i$  an
- 3 Auswertung: Wenn die Chart ein Item der Form

$$\gamma \rightarrow S \bullet [0, n]$$

enthält, dann akzeptiert die Grammatik den Satz,  
andernfalls nicht

# Complete [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 0

S <sub>1</sub> :	$\gamma \rightarrow \bullet S$	[0,0] []
S <sub>2</sub> :	$S \rightarrow \bullet NP VP$	[0,0] []
S <sub>3</sub> :	$S \rightarrow \bullet S Conj S$	[0,0] []
S <sub>4</sub> :	$NP \rightarrow \bullet D Nbar$	[0,0] []
S <sub>5</sub> :	$NP \rightarrow \bullet Nbar$	[0,0] []
S <sub>6</sub> :	$NP \rightarrow \bullet NE$	[0,0] []
S <sub>7</sub> :	$NP \rightarrow \bullet PRON$	[0,0] []
S <sub>8</sub> :	$NP \rightarrow \bullet NP Conj NP$	[0,0] []
S <sub>9</sub> :	$D \rightarrow \bullet dem$	[0,0] []
S <sub>10</sub> :	$D \rightarrow \bullet der$	[0,0] []
S <sub>11</sub> :	$D \rightarrow \bullet die$	[0,0] []
S <sub>12</sub> :	$Nbar \rightarrow \bullet AP Nbar$	[0,0] []
S <sub>13</sub> :	$Nbar \rightarrow \bullet Nbar PP$	[0,0] []
S <sub>14</sub> :	$Nbar \rightarrow \bullet N$	[0,0] []
S <sub>15</sub> :	$AP \rightarrow \bullet AdvP AP$	[0,0] []
S <sub>16</sub> :	$AP \rightarrow \bullet A$	[0,0] []
S <sub>17</sub> :	$AP \rightarrow \bullet AP Conj AP$	[0,0] []
S <sub>18</sub> :	$N \rightarrow \bullet Baum$	[0,0] []
S <sub>19</sub> :	$N \rightarrow \bullet Kinder$	[0,0] []
S <sub>20</sub> :	$N \rightarrow \bullet Hund$	[0,0] []

## Position 1

S <sub>22</sub> :	$D \rightarrow die \bullet$	[0,1] []
S <sub>23</sub> :	$NP \rightarrow D \bullet Nbar$	[0,1] [S <sub>22</sub> ]

[S<sub>22</sub>]: Zeiger auf die abgeschlossene Regel, um die geparsste Struktur später auslesen zu können

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 0

S <sub>1</sub> :	$\gamma \rightarrow \bullet S$	[0,0] []
S <sub>2</sub> :	$S \rightarrow \bullet NP VP$	[0,0] []
S <sub>3</sub> :	$S \rightarrow \bullet S Conj S$	[0,0] []
S <sub>4</sub> :	$NP \rightarrow \bullet D Nbar$	[0,0] []
S <sub>5</sub> :	$NP \rightarrow \bullet Nbar$	[0,0] []
S <sub>6</sub> :	$NP \rightarrow \bullet NE$	[0,0] []
S <sub>7</sub> :	$NP \rightarrow \bullet PRON$	[0,0] []
S <sub>8</sub> :	$NP \rightarrow \bullet NP Conj NP$	[0,0] []
S <sub>9</sub> :	$D \rightarrow \bullet dem$	[0,0] []
S <sub>10</sub> :	$D \rightarrow \bullet der$	[0,0] []
S <sub>11</sub> :	$D \rightarrow \bullet die$	[0,0] []
S <sub>12</sub> :	$Nbar \rightarrow \bullet AP Nbar$	[0,0] []
S <sub>13</sub> :	$Nbar \rightarrow \bullet Nbar PP$	[0,0] []
S <sub>14</sub> :	$Nbar \rightarrow \bullet N$	[0,0] []
S <sub>15</sub> :	$AP \rightarrow \bullet AdvP AP$	[0,0] []
S <sub>16</sub> :	$AP \rightarrow \bullet A$	[0,0] []
S <sub>17</sub> :	$AP \rightarrow \bullet AP Conj AP$	[0,0] []
S <sub>18</sub> :	$N \rightarrow \bullet Baum$	[0,0] []
S <sub>19</sub> :	$N \rightarrow \bullet Kinder$	[0,0] []
S <sub>20</sub> :	$N \rightarrow \bullet Hund$	[0,0] []

## Position 1

S <sub>22</sub> :	$D \rightarrow die \bullet$	[0,1] []
S <sub>23</sub> :	$NP \rightarrow D \bullet Nbar$	[0,1] [S <sub>22</sub> ]
S <sub>24</sub> :	$Nbar \rightarrow \bullet N$	[1,1] []
S <sub>25</sub> :	$Nbar \rightarrow \bullet AP Nbar$	[1,1] []
S <sub>26</sub> :	$Nbar \rightarrow \bullet Nbar PP$	[1,1] []
S <sub>27</sub> :	$AP \rightarrow \bullet AdvP AP$	[1,1] []
S <sub>28</sub> :	$AP \rightarrow \bullet A$	[1,1] []
S <sub>29</sub> :	$AP \rightarrow \bullet AP Conj AP$	[1,1] []
S <sub>30</sub> :	$N \rightarrow \bullet Baum$	[1,1] []
S <sub>31</sub> :	$N \rightarrow \bullet Kinder$	[1,1] []
S <sub>32</sub> :	$N \rightarrow \bullet Hund$	[1,1] []
S <sub>33</sub> :	$AdvP \rightarrow \bullet Adv$	[1,1] []

# Scan [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 1

$S_{22}: D \rightarrow \text{die} \bullet$	[0,1] [ ]
$S_{23}: NP \rightarrow D \bullet Nbar$	[0,1] [S <sub>22</sub> ]
$S_{24}: Nbar \rightarrow \bullet N$	[1,1] [ ]
$S_{25}: Nbar \rightarrow \bullet AP Nbar$	[1,1] [ ]
$S_{26}: Nbar \rightarrow \bullet Nbar PP$	[1,1] [ ]
$S_{27}: AP \rightarrow \bullet AdvP AP$	[1,1] [ ]
$S_{28}: AP \rightarrow \bullet A$	[1,1] [ ]
$S_{29}: AP \rightarrow \bullet AP Conj AP$	[1,1] [ ]
$S_{30}: N \rightarrow \bullet Baum$	[1,1] [ ]
$S_{31}: N \rightarrow \bullet Kinder$	[1,1] [ ]
$S_{32}: N \rightarrow \bullet Hund$	[1,1] [ ]
$S_{33}: AdvP \rightarrow \bullet Adv$	[1,1] [ ]

## Position 2

$S_{34}: N \rightarrow \text{Kinder} \bullet$  [1,2] [ ]  
 • die • Kinder • und • der • Hund • ...  
 0      1      2      3      4      5

# Complete [0 die 1 Kinder 2 und 3 der 4 Hund 5 schlafen 6 vor 7 dem 8 Baum 9]

Position 1

$S_{22}: D \rightarrow \text{die} \bullet$	[0,1] [ ]
$S_{23}: NP \rightarrow D \bullet Nbar$	[0,1] [S <sub>22</sub> ]
$S_{24}: Nbar \rightarrow \bullet N$	[1,1] [ ]
$S_{25}: Nbar \rightarrow \bullet AP Nbar$	[1,1] [ ]
$S_{26}: Nbar \rightarrow \bullet Nbar PP$	[1,1] [ ]
$S_{27}: AP \rightarrow \bullet AdvP AP$	[1,1] [ ]
$S_{28}: AP \rightarrow \bullet A$	[1,1] [ ]
$S_{29}: AP \rightarrow \bullet AP Conj AP$	[1,1] [ ]
$S_{30}: N \rightarrow \bullet Baum$	[1,1] [ ]
$S_{31}: N \rightarrow \bullet Kinder$	[1,1] [ ]
$S_{32}: N \rightarrow \bullet Hund$	[1,1] [ ]
$S_{33}: AdvP \rightarrow \bullet Adv$	[1,1] [ ]

Position 2

$S_{34}: N \rightarrow \text{Kinder} \bullet$	[1,2] [ ]
$S_{35}: Nbar \rightarrow N \bullet$	[1,2] [S <sub>34</sub> ]

# Complete [0 die 1 Kinder 2 und 3 der 4 Hund 5 schlafen 6 vor 7 dem 8 Baum 9]

## Position 1

$S_{22}: D \rightarrow \text{die} \bullet$	[0,1] [ ]
$S_{23}: NP \rightarrow D \bullet Nbar$	[0,1] [S <sub>22</sub> ]
$S_{24}: Nbar \rightarrow \bullet N$	[1,1] [ ]
$S_{25}: Nbar \rightarrow \bullet AP Nbar$	[1,1] [ ]
$S_{26}: Nbar \rightarrow \bullet Nbar PP$	[1,1] [ ]
$S_{27}: AP \rightarrow \bullet AdvP AP$	[1,1] [ ]
$S_{28}: AP \rightarrow \bullet A$	[1,1] [ ]
$S_{29}: AP \rightarrow \bullet AP Conj AP$	[1,1] [ ]
$S_{30}: N \rightarrow \bullet Baum$	[1,1] [ ]
$S_{31}: N \rightarrow \bullet Kinder$	[1,1] [ ]
$S_{32}: N \rightarrow \bullet Hund$	[1,1] [ ]
$S_{33}: AdvP \rightarrow \bullet Adv$	[1,1] [ ]

## Position 2

$S_{34}: N \rightarrow Kinder \bullet$	[1,2] [ ]
$S_{35}: Nbar \rightarrow N \bullet$	[1,2] [S <sub>34</sub> ]
$S_{36}: NP \rightarrow D Nbar \bullet$	[0,2] [S <sub>22</sub> , S <sub>35</sub> ]
$S_{37}: Nbar \rightarrow Nbar \bullet PP$	[1,2] [S <sub>35</sub> ]

# Complete [0 die 1 Kinder 2 und 3 der 4 Hund 5 schlafen 6 vor 7 dem 8 Baum 9]

Position 0	Position 2
<p>...</p> <p><math>S_2: S \rightarrow \bullet NP VP</math> [0,0] []</p> <p>...</p> <p><math>S_8: NP \rightarrow \bullet NP Conj NP</math> [0,0] []</p> <p>...</p>	<p><math>S_{34}: N \rightarrow Kinder \bullet</math> [1,2] []</p> <p><math>S_{35}: Nbar \rightarrow N \bullet</math> [1,2] [S<sub>34</sub>]</p> <p><math>S_{36}: NP \rightarrow D Nbar \bullet</math> [0,2] [S<sub>22</sub>, S<sub>35</sub>]</p> <p><math>S_{37}: Nbar \rightarrow Nbar \bullet PP</math> [1,2] [S<sub>35</sub>]</p> <p><math>S_{38}: S \rightarrow NP \bullet VP</math> [0,2] [S<sub>36</sub>]</p> <p><math>S_{39}: NP \rightarrow NP \bullet Conj NP</math> [0,2] [S<sub>36</sub>]</p>

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 1		Position 2	
S <sub>22</sub> : D → die •	[0,1] []	S <sub>34</sub> : N → Kinder •	[1,2] []
S <sub>23</sub> : NP → D • Nbar	[0,1] [S <sub>22</sub> ]	S <sub>35</sub> : Nbar → N •	[1,2] [S <sub>34</sub> ]
S <sub>24</sub> : Nbar → • N	[1,1] []	S <sub>36</sub> : NP → D Nbar •	[0,2] [S <sub>22</sub> , S <sub>35</sub> ]
S <sub>25</sub> : Nbar → • AP Nbar	[1,1] []	S <sub>37</sub> : Nbar → Nbar • PP	[1,2] [S <sub>35</sub> ]
S <sub>26</sub> : Nbar → • Nbar PP	[1,1] []	S <sub>38</sub> : S → NP • VP	[0,2] [S <sub>36</sub> ]
S <sub>27</sub> : AP → • AdvP AP	[1,1] []	S <sub>39</sub> : NP → NP • Conj NP	[0,2] [S <sub>36</sub> ]
S <sub>28</sub> : AP → • A	[1,1] []	S <sub>40</sub> : PP → • P NP	[2,2] []
S <sub>29</sub> : AP → • AP Conj AP	[1,1] []	S <sub>41</sub> : PP → • PP Conj PP	[2,2] []
S <sub>30</sub> : N → • Baum	[1,1] []	S <sub>42</sub> : VP → • V	[2,2] []
S <sub>31</sub> : N → • Kinder	[1,1] []	S <sub>43</sub> : VP → • V NP	[2,2] []
S <sub>32</sub> : N → • Hund	[1,1] []	S <sub>44</sub> : VP → • V NP NP	[2,2] []
S <sub>33</sub> : AdvP → • Adv	[1,1] []	S <sub>45</sub> : VP → • VP PP	[2,2] []
		S <sub>46</sub> : VP → • VP Conj VP	[2,2] []
		S <sub>47</sub> : Conj → • und	[2,2] []
		S <sub>48</sub> : P → • vor	[2,2] []
		S <sub>49</sub> : V → • schlafen	[2,2] []

# Scan [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 2

S <sub>34</sub> : N → Kinder •	[1,2] []
S <sub>35</sub> : Nbar → N •	[1,2] [S <sub>34</sub> ]
S <sub>36</sub> : NP → D Nbar •	[0,2] [S <sub>22</sub> , S <sub>35</sub> ]
S <sub>37</sub> : Nbar → Nbar • PP	[1,2] [S <sub>35</sub> ]
S <sub>38</sub> : S → NP • VP	[0,2] [S <sub>36</sub> ]
S <sub>39</sub> : NP → NP • Conj NP	[0,2] [S <sub>36</sub> ]
S <sub>40</sub> : PP → • P NP	[2,2] []
S <sub>41</sub> : PP → • PP Conj PP	[2,2] []
S <sub>42</sub> : VP → • V	[2,2] []
S <sub>43</sub> : VP → • V NP	[2,2] []
S <sub>44</sub> : VP → • V NP NP	[2,2] []
S <sub>45</sub> : VP → • VP PP	[2,2] []
S <sub>46</sub> : VP → • VP Conj VP	[2,2] []
S <sub>47</sub> : Conj → • und	[2,2] []
S <sub>48</sub> : P → • vor	[2,2] []
S <sub>49</sub> : V → • schlafen	[2,2] []

## Position 3

**S<sub>50</sub>: Conj → und • [2,3] []**

**• die • Kinder • und • der • Hund • ...**  
 0      1      2      3      4      5

# Complete [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 2		Position 3	
S <sub>34</sub> :	N → Kinder •	[1,2]	[ ]
S <sub>35</sub> :	Nbar → N •	[1,2]	[S <sub>34</sub> ]
S <sub>36</sub> :	NP → D Nbar •	[0,2]	[S <sub>22</sub> , S <sub>35</sub> ]
S <sub>37</sub> :	Nbar → Nbar • PP	[1,2]	[S <sub>35</sub> ]
S <sub>38</sub> :	S → NP • VP	[0,2]	[S <sub>36</sub> ]
S <sub>39</sub> :	NP → NP • Conj NP	[0,2]	[S <sub>36</sub> ]
S <sub>40</sub> :	PP → • P NP	[2,2]	[ ]
S <sub>41</sub> :	PP → • PP Conj PP	[2,2]	[ ]
S <sub>42</sub> :	VP → • V	[2,2]	[ ]
S <sub>43</sub> :	VP → • V NP	[2,2]	[ ]
S <sub>44</sub> :	VP → • V NP NP	[2,2]	[ ]
S <sub>45</sub> :	VP → • VP PP	[2,2]	[ ]
S <sub>46</sub> :	VP → • VP Conj VP	[2,2]	[ ]
S <sub>47</sub> :	Conj → • und	[2,2]	[ ]
S <sub>48</sub> :	P → • vor	[2,2]	[ ]
S <sub>49</sub> :	V → • schlafen	[2,2]	[ ]

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 2

S <sub>34</sub> : N → Kinder •	[1,2] []
S <sub>35</sub> : Nbar → N •	[1,2] [S <sub>34</sub> ]
S <sub>36</sub> : NP → D Nbar •	[0,2] [S <sub>22</sub> , S <sub>35</sub> ]
S <sub>37</sub> : Nbar → Nbar • PP	[1,2] [S <sub>35</sub> ]
S <sub>38</sub> : S → NP • VP	[0,2] [S <sub>36</sub> ]
S <sub>39</sub> : NP → NP • Conj NP	[0,2] [S <sub>36</sub> ]
S <sub>40</sub> : PP → • P NP	[2,2] []
S <sub>41</sub> : PP → • PP Conj PP	[2,2] []
S <sub>42</sub> : VP → • V	[2,2] []
S <sub>43</sub> : VP → • V NP	[2,2] []
S <sub>44</sub> : VP → • V NP NP	[2,2] []
S <sub>45</sub> : VP → • VP PP	[2,2] []
S <sub>46</sub> : VP → • VP Conj VP	[2,2] []
S <sub>47</sub> : Conj → • und	[2,2] []
S <sub>48</sub> : P → • vor	[2,2] []
S <sub>49</sub> : V → • schlafen	[2,2] []

## Position 3

S <sub>50</sub> : Conj → und •	[2,3] []
S <sub>51</sub> : NP → NP Conj • NP	[0,3] [S <sub>36</sub> , S <sub>50</sub> ]
S <sub>52</sub> : NP → • D Nbar	[3,3] []
S <sub>53</sub> : NP → • Nbar	[3,3] []
S <sub>54</sub> : NP → • NE	[3,3] []
S <sub>55</sub> : NP → • PRON	[3,3] []
S <sub>56</sub> : NP → • NP Conj NP	[3,3] []
S <sub>57</sub> : D → • dem	[3,3] []
S <sub>58</sub> : D → • der	[3,3] []
S <sub>59</sub> : D → • die	[3,3] []
S <sub>60</sub> : Nbar → • AP Nbar	[3,3] []
S <sub>61</sub> : Nbar → • Nbar PP	[3,3] []
S <sub>62</sub> : Nbar → • N	[3,3] []
S <sub>63</sub> : AP → • AdvP AP	[3,3] []
S <sub>64</sub> : AP → • A	[3,3] []
S <sub>65</sub> : AP → • AP Conj AP	[3,3] []
S <sub>66</sub> : N → • Baum	[3,3] []
S <sub>67</sub> : N → • Kinder	[3,3] []
S <sub>68</sub> : N → • Hund	[3,3] []
S <sub>69</sub> : AdvP → • Adv	[3,3] []

# Scan [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 3

S <sub>50</sub> : Conj → und •	[2,3] []
S <sub>51</sub> : NP → NP Conj • NP	[0,3] [S <sub>36</sub> , S <sub>50</sub> ] []
S <sub>52</sub> : NP → • D Nbar	[3,3] []
S <sub>53</sub> : NP → • Nbar	[3,3] []
S <sub>54</sub> : NP → • NE	[3,3] []
S <sub>55</sub> : NP → • PRON	[3,3] []
S <sub>56</sub> : NP → • NP Conj NP	[3,3] []
S <sub>57</sub> : D → • dem	[3,3] []
S <sub>58</sub> : D → • der	[3,3] []
S <sub>59</sub> : D → • die	[3,3] []
S <sub>60</sub> : Nbar → • AP Nbar	[3,3] []
S <sub>61</sub> : Nbar → • Nbar PP	[3,3] []
S <sub>62</sub> : Nbar → • N	[3,3] []
S <sub>63</sub> : AP → • AdvP AP	[3,3] []
S <sub>64</sub> : AP → • A	[3,3] []
S <sub>65</sub> : AP → • AP Conj AP	[3,3] []
S <sub>66</sub> : N → • Baum	[3,3] []
S <sub>67</sub> : N → • Kinder	[3,3] []
S <sub>68</sub> : N → • Hund	[3,3] []
S <sub>69</sub> : AdvP → • Adv	[3,3] []

Position 4

**S<sub>70</sub>: D → der • [3,4] []**  
**• die • Kinder • und • der • Hund • ...**  
 0      1      2      3      4      5

# Complete [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 3

S <sub>50</sub> : Conj → und •	[2,3] []
S <sub>51</sub> : NP → NP Conj • NP	[0,3] [S <sub>36</sub> , S <sub>50</sub> ]
S <sub>52</sub> : NP → • D Nbar	[3,3] []
S <sub>53</sub> : NP → • Nbar	[3,3] []
S <sub>54</sub> : NP → • NE	[3,3] []
S <sub>55</sub> : NP → • PRON	[3,3] []
S <sub>56</sub> : NP → • NP Conj NP	[3,3] []
S <sub>57</sub> : D → • dem	[3,3] []
S <sub>58</sub> : D → • der	[3,3] []
S <sub>59</sub> : D → • die	[3,3] []
S <sub>60</sub> : Nbar → • AP Nbar	[3,3] []
S <sub>61</sub> : Nbar → • Nbar PP	[3,3] []
S <sub>62</sub> : Nbar → • N	[3,3] []
S <sub>63</sub> : AP → • AdvP AP	[3,3] []
S <sub>64</sub> : AP → • A	[3,3] []
S <sub>65</sub> : AP → • AP Conj AP	[3,3] []
S <sub>66</sub> : N → • Baum	[3,3] []
S <sub>67</sub> : N → • Kinder	[3,3] []
S <sub>68</sub> : N → • Hund	[3,3] []
S <sub>69</sub> : AdvP → • Adv	[3,3] []

## Position 4

S <sub>70</sub> : D → der	[3,4] []
S <sub>71</sub> : NP → D • Nbar	[3,4] [S <sub>70</sub> ]

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 3

S <sub>50</sub> : Conj → und •	[2,3] []
S <sub>51</sub> : NP → NP Conj • NP	[0,3] [S <sub>36</sub> , S <sub>50</sub> ]
S <sub>52</sub> : NP → • D Nbar	[3,3] []
S <sub>53</sub> : NP → • Nbar	[3,3] []
S <sub>54</sub> : NP → • NE	[3,3] []
S <sub>55</sub> : NP → • PRON	[3,3] []
S <sub>56</sub> : NP → • NP Conj NP	[3,3] []
S <sub>57</sub> : D → • dem	[3,3] []
S <sub>58</sub> : D → • der	[3,3] []
S <sub>59</sub> : D → • die	[3,3] []
S <sub>60</sub> : Nbar → • AP Nbar	[3,3] []
S <sub>61</sub> : Nbar → • Nbar PP	[3,3] []
S <sub>62</sub> : Nbar → • N	[3,3] []
S <sub>63</sub> : AP → • AdvP AP	[3,3] []
S <sub>64</sub> : AP → • A	[3,3] []
S <sub>65</sub> : AP → • AP Conj AP	[3,3] []
S <sub>66</sub> : N → • Baum	[3,3] []
S <sub>67</sub> : N → • Kinder	[3,3] []
S <sub>68</sub> : N → • Hund	[3,3] []
S <sub>69</sub> : AdvP → • Adv	[3,3] []

## Position 4

S <sub>70</sub> : D → der	[3,4] []
S <sub>71</sub> : NP → D • Nbar	[3,4] [S <sub>70</sub> ]
S <sub>72</sub> : Nbar → • AP Nbar	[4,4] []
S <sub>73</sub> : Nbar → • Nbar PP	[4,4] []
S <sub>74</sub> : Nbar → • N	[4,4] []
S <sub>75</sub> : AP → • AdvP AP	[4,4] []
S <sub>76</sub> : AP → • A	[4,4] []
S <sub>77</sub> : AP → • AP Conj AP	[4,4] []
S <sub>78</sub> : N → • Baum	[4,4] []
S <sub>79</sub> : N → • Kinder	[4,4] []
S <sub>80</sub> : N → • Hund	[4,4] []
S <sub>81</sub> : AdvP → • Adv	[4,4] []

# Scan [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 4

$S_{70}: D \rightarrow \text{der}$	[3,4] [ ]
$S_{71}: NP \rightarrow D \bullet Nbar$	[3,4] [S <sub>70</sub> ] [ ]
$S_{72}: Nbar \rightarrow \bullet AP Nbar$	[4,4] [ ]
$S_{73}: Nbar \rightarrow \bullet Nbar PP$	[4,4] [ ]
$S_{74}: Nbar \rightarrow \bullet N$	[4,4] [ ]
$S_{75}: AP \rightarrow \bullet AdvP AP$	[4,4] [ ]
$S_{76}: AP \rightarrow \bullet A$	[4,4] [ ]
$S_{77}: AP \rightarrow \bullet AP Conj AP$	[4,4] [ ]
$S_{78}: N \rightarrow \bullet Baum$	[4,4] [ ]
$S_{79}: N \rightarrow \bullet Kinder$	[4,4] [ ]
$S_{80}: N \rightarrow \bullet Hund$	[4,4] [ ]
$S_{81}: AdvP \rightarrow \bullet Adv$	[4,4] [ ]

Position 5

$S_{82}: N \rightarrow \text{Hund} \bullet$	[4,5] [ ]
---	-----------

# Complete [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 4

$S_{70}: D \rightarrow \text{der}$	[3,4] [ ]
$S_{71}: NP \rightarrow D \bullet Nbar$	[3,4] [S <sub>70</sub> ]
$S_{72}: Nbar \rightarrow \bullet AP Nbar$	[4,4] [ ]
$S_{73}: Nbar \rightarrow \bullet Nbar PP$	[4,4] [ ]
$S_{74}: Nbar \rightarrow \bullet N$	[4,4] [ ]
$S_{75}: AP \rightarrow \bullet AdvP AP$	[4,4] [ ]
$S_{76}: AP \rightarrow \bullet A$	[4,4] [ ]
$S_{77}: AP \rightarrow \bullet AP Conj AP$	[4,4] [ ]
$S_{78}: N \rightarrow \bullet Baum$	[4,4] [ ]
$S_{79}: N \rightarrow \bullet Kinder$	[4,4] [ ]
$S_{80}: N \rightarrow \bullet Hund$	[4,4] [ ]
$S_{81}: AdvP \rightarrow \bullet Adv$	[4,4] [ ]

## Position 5

$S_{82}: N \rightarrow \text{Hund} \bullet$	[4,5] [ ]
$S_{83}: Nbar \rightarrow N \bullet$	[4,5] [S <sub>82</sub> ]
$S_{84}: NP \rightarrow D Nbar \bullet$	[3,5] [S <sub>70</sub> , S <sub>83</sub> ]
$S_{85}: Nbar \rightarrow Nbar \bullet PP$	[4,5] [S <sub>83</sub> ]
$S_{86}: NP \rightarrow NP Conj NP \bullet$	[0,5] [S <sub>36</sub> , S <sub>50</sub> , S <sub>88</sub> ]
$S_{87}: S \rightarrow NP \bullet VP$	[0,5] [S <sub>86</sub> ]
$S_{88}: NP \rightarrow NP \bullet Conj NP$	[3,5] [S <sub>84</sub> ]

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 4		Position 5	
S <sub>82</sub> : N → Hund •	[4,5] []	S <sub>89</sub> : PP → • P NP	[5,5] []
S <sub>83</sub> : Nbar → N •	[4,5] [S <sub>82</sub> ]	S <sub>90</sub> : PP → • PP Conj PP	[5,5] []
S <sub>84</sub> : NP → D Nbar •	[3,5] [S <sub>70</sub> , S <sub>83</sub> ]	S <sub>91</sub> : VP → • V	[5,5] []
S <sub>85</sub> : Nbar → Nbar • PP	[4,5] [S <sub>83</sub> ]	S <sub>92</sub> : VP → • V NP	[5,5] []
S <sub>86</sub> : NP → NP Conj NP •	[0,5] [S <sub>36</sub> , S <sub>50</sub> , S <sub>84</sub> ]	S <sub>93</sub> : VP → • V NP NP	[5,5] []
S <sub>87</sub> : S → NP • VP	[0,5] [S <sub>86</sub> ]	S <sub>94</sub> : VP → • VP PP	[5,5] []
S <sub>88</sub> : NP → NP • Conj NP	[3,5] [S <sub>84</sub> ]	S <sub>95</sub> : VP → • VP Conj VP	[5,5] []
		S <sub>96</sub> : Conj → • und	[5,5] []
		S <sub>97</sub> : P → • vor	[5,5] []
		S <sub>98</sub> : V → • schlafen	[5,5] []

# Scan [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 5		Position 6	
S <sub>89</sub> :	PP → • P NP	[5,5]	[ ]
S <sub>90</sub> :	PP → • PP Conj PP	[5,5]	[ ]
S <sub>91</sub> :	VP → • V	[5,5]	[ ]
S <sub>92</sub> :	VP → • V NP	[5,5]	[ ]
S <sub>93</sub> :	VP → • V NP NP	[5,5]	[ ]
S <sub>94</sub> :	VP → • VP PP	[5,5]	[ ]
S <sub>95</sub> :	VP → • VP Conj VP	[5,5]	[ ]
S <sub>96</sub> :	Conj → • und	[5,5]	[ ]
S <sub>97</sub> :	P → • vor	[5,5]	[ ]
S <sub>98</sub> :	V → • schlafen	[5,5]	[ ]

# Complete [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 5

$S_{89}: PP \rightarrow \bullet P NP$	[5,5] []
$S_{90}: PP \rightarrow \bullet PP Conj PP$	[5,5] []
$S_{91}: VP \rightarrow \bullet V$	[5,5] []
$S_{92}: VP \rightarrow \bullet V NP$	[5,5] []
$S_{93}: VP \rightarrow \bullet V NP NP$	[5,5] []
$S_{94}: VP \rightarrow \bullet VP PP$	[5,5] []
$S_{95}: VP \rightarrow \bullet VP Conj VP$	[5,5] []
$S_{96}: Conj \rightarrow \bullet und$	[5,5] []
$S_{97}: P \rightarrow \bullet vor$	[5,5] []
$S_{98}: V \rightarrow \bullet schlafen$	[5,5] []

## Position 6

$S_{99}: V \rightarrow schlafen \bullet$	[5,6] []
$S_{100}: VP \rightarrow V \bullet$	[5,6] [S <sub>99</sub> ]
$S_{101}: VP \rightarrow V \bullet NP$	[5,6] [S <sub>99</sub> ]
$S_{102}: VP \rightarrow V \bullet NP NP$	[5,6] [S <sub>99</sub> ]
$S_{103}: VP \rightarrow VP \bullet PP$	[5,6] [S <sub>100</sub> ]
$S_{104}: VP \rightarrow VP \bullet Conj VP$	[5,6] [S <sub>100</sub> ]
$S_{105}: S \rightarrow NP VP \bullet$	[0,6] [S <sub>86</sub> , S <sub>100</sub> ]

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 5

$S_{89}: PP \rightarrow \bullet P NP$	[5,5] []
$S_{90}: PP \rightarrow \bullet PP Conj PP$	[5,5] []
$S_{91}: VP \rightarrow \bullet V$	[5,5] []
$S_{92}: VP \rightarrow \bullet V NP$	[5,5] []
$S_{93}: VP \rightarrow \bullet V NP NP$	[5,5] []
$S_{94}: VP \rightarrow \bullet VP PP$	[5,5] []
$S_{95}: VP \rightarrow \bullet VP Conj VP$	[5,5] []
$S_{96}: Conj \rightarrow \bullet und$	[5,5] []
$S_{97}: P \rightarrow \bullet vor$	[5,5] []
$S_{98}: V \rightarrow \bullet schlafen$	[5,5] []

## Position 6

$S_{99}: V \rightarrow schlafen \bullet$	[5,6] []
$S_{100}: VP \rightarrow V \bullet$	[5,6] [S <sub>99</sub> ]
$S_{101}: VP \rightarrow V \bullet NP$	[5,6] [S <sub>99</sub> ]
$S_{102}: VP \rightarrow V \bullet NP NP$	[5,6] [S <sub>99</sub> ]
$S_{103}: VP \rightarrow VP \bullet PP$	[5,6] [S <sub>100</sub> ]
$S_{104}: VP \rightarrow VP \bullet Conj VP$	[5,6] [S <sub>100</sub> ]
$S_{105}: S \rightarrow NP VP \bullet$	[0,6] [S <sub>86</sub> , S <sub>100</sub> ]
<b>Ab hier nur noch ausgewählte Regeln!</b>	
$S_{106}: PP \rightarrow \bullet P NP$	[6,6] []
$S_{107}: P \rightarrow \bullet vor$	[6,6] []

# Scan+Compl. [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 6

$S_{99}: V \rightarrow \text{schlafen} \bullet$	[5,6] []
$S_{100}: VP \rightarrow V \bullet$	[5,6] [ $S_{99}$ ]
$S_{101}: VP \rightarrow V \bullet NP$	[5,6] [ $S_{99}$ ]
$S_{102}: VP \rightarrow V \bullet NP NP$	[5,6] [ $S_{99}$ ]
$S_{103}: VP \rightarrow VP \bullet PP$	[5,6] [ $S_{100}$ ]
$S_{104}: VP \rightarrow VP \bullet Conj VP$	[5,6] [ $S_{100}$ ]
$S_{105}: S \rightarrow NP VP \bullet$	[0,6] [ $S_{86}, S_{100}$ ]
$S_{106}: PP \rightarrow \bullet P NP$	[6,6] []
$S_{107}: P \rightarrow \bullet vor$	[6,6] []

## Position 7

$S_{108}: P \rightarrow vor \bullet$	[6,7] []
$S_{109}: PP \rightarrow P \bullet NP$	[6,7] [ $S_{108}$ ]

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

## Position 6

$S_{99}: V \rightarrow \text{schlafen} \bullet$	[5,6] []
$S_{100}: VP \rightarrow V \bullet$	[5,6] [ $S_{99}$ ]
$S_{101}: VP \rightarrow V \bullet NP$	[5,6] [ $S_{99}$ ]
$S_{102}: VP \rightarrow V \bullet NP NP$	[5,6] [ $S_{99}$ ]
$S_{103}: VP \rightarrow VP \bullet PP$	[5,6] [ $S_{100}$ ]
$S_{104}: VP \rightarrow VP \bullet Conj VP$	[5,6] [ $S_{100}$ ]
$S_{105}: S \rightarrow NP VP \bullet$	[0,6] [ $S_{86}, S_{100}$ ]
$S_{106}: PP \rightarrow \bullet P NP$	[6,6] []
$S_{107}: P \rightarrow \bullet vor$	[6,6] []

## Position 7

$S_{108}: P \rightarrow vor \bullet$	[6,7] []
$S_{109}: PP \rightarrow P \bullet NP$	[6,7] [ $S_{108}$ ]
$S_{110}: NP \rightarrow \bullet D Nbar$	[7,7] []
$S_{111}: D \rightarrow \bullet dem$	[7,7] []

# Scan+Compl.

[<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 7

S<sub>108</sub>: P → vor • [6,7] []  
S<sub>109</sub>: PP → P • NP [6,7] [S<sub>108</sub>]  
S<sub>110</sub>: NP → • D Nbar [7,7] []  
S<sub>111</sub>: D → • dem [7,7] []

Position 8

S<sub>112</sub>: D → dem • [7,8] []  
S<sub>113</sub>: NP → D • Nbar [7,8] [S<sub>112</sub>]

# Predict [<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 7

$S_{108}: P \rightarrow \text{vor } \bullet$	[6,7] []
$S_{109}: PP \rightarrow P \bullet NP$	[6,7] [ $S_{108}$ ]
$S_{110}: NP \rightarrow \bullet D Nbar$	[7,7] []
$S_{111}: D \rightarrow \bullet dem$	[7,7] []

Position 8

$S_{112}: D \rightarrow dem \bullet$	[7,8] []
$S_{113}: NP \rightarrow D \bullet Nbar$	[7,8] [ $S_{112}$ ]
$S_{114}: Nbar \rightarrow \bullet N$	[8,8] []
$S_{115}: N \rightarrow \bullet Baum$	[8,8] []

# Scan+Compl.

[<sub>0</sub> die <sub>1</sub> Kinder <sub>2</sub> und <sub>3</sub> der <sub>4</sub> Hund <sub>5</sub> schlafen <sub>6</sub> vor <sub>7</sub> dem <sub>8</sub> Baum <sub>9</sub>]

Position 8

$S_{112}:$	$D \rightarrow \text{dem} \bullet$	[7,8] []
$S_{113}:$	$NP \rightarrow D \bullet Nbar$	[7,8] [S <sub>112</sub> ]
$S_{114}:$	$Nbar \rightarrow \bullet N$	[8,8] []
$S_{115}:$	$N \rightarrow \bullet Baum$	[8,8] []

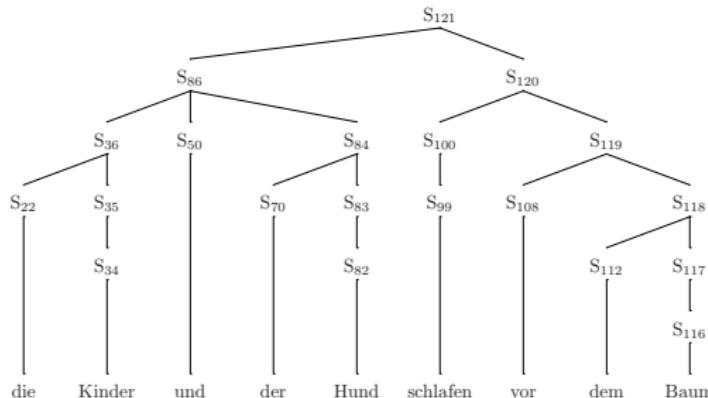
Position 9

$S_{116}:$	$N \rightarrow \text{Baum} \bullet$	[8,9] []
$S_{117}:$	$Nbar \rightarrow N \bullet$	[8,9] [S <sub>116</sub> ]
$S_{118}:$	$NP \rightarrow D Nbar \bullet$	[7,9] [S <sub>112</sub> , S <sub>117</sub> ]
$S_{119}:$	$PP \rightarrow P NP \bullet$	[6,9] [S <sub>108</sub> , S <sub>118</sub> ]
$S_{120}:$	$VP \rightarrow VP PP \bullet$	[5,9] [S <sub>99</sub> , S <sub>119</sub> ]
$S_{121}:$	$S \rightarrow NP VP \bullet$	[0,9] [S <sub>86</sub> , S <sub>120</sub> ]
$S_{121}:$	$S \rightarrow NP VP \bullet$	[0,9] [S <sub>86</sub> , S <sub>120</sub> ]
$S_{122}:$	$\gamma \rightarrow S \bullet$	[0,9] [S <sub>120</sub> ]

vollständiger Parse!

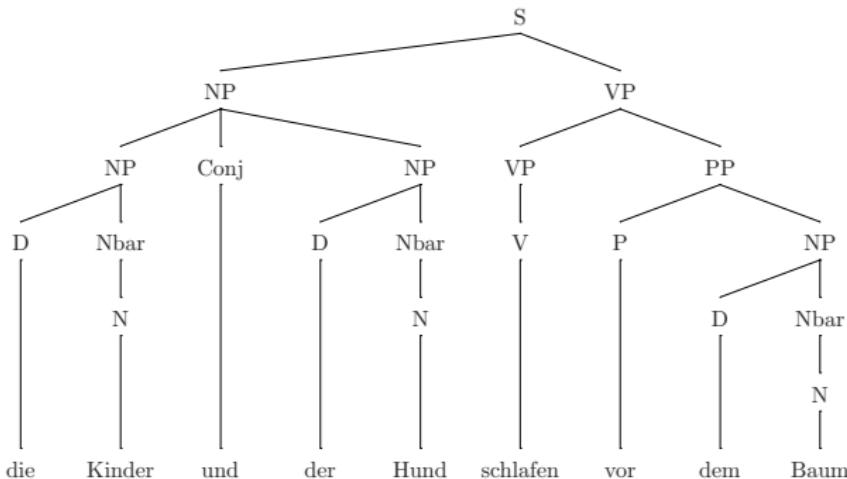
# Vollständiger Parse

- Ein Satz wird dann von der Grammatik lizenziert, wenn in der letzten Zelle der Tabelle eine fertiggestellte Regel mit dem Dummy-Startsymbol  $\gamma$  auf der linken Seite steht, die alle Wörter des Satzes umfasst  $[0, n]$
- Mit Hilfe der Zeiger kann jetzt die ermittelte Struktur des Satzes ausgelesen werden



# Vollständiger Parse

Da die Einträge in der Chart auch die angewandten Regeln enthalten, kann die geparsste Struktur inklusive der Kategorien aus der Tabelle rekonstruiert werden



# Vollständiger Parse

- Falls es mehrere Regeln der Form  $\gamma \rightarrow \dots [0, n]$  [Zeiger] in der letzten Tabellenzelle gibt, ist der Satz ambig, d.h. es gibt mehrere mögliche Analysen
- Dazu müssen mehrere Regeln der gleichen Form (nur mit unterschiedlichen Zeigern) in einer Zelle erlaubt sein
- Dann können alle Analysen nacheinander aus der Chart ausgelesen werden

# Zusammenfassung: Earley-Parser

## ■ Vorteile

- Der Earley-Parser kann mit allen kontextfreien Grammatiken umgehen (keine Umwandlung in Chomsky-Normalform nötig)
- Er besitzt ebenfalls eine Zeitkomplexität von  $O(n^3)$
- Keine Probleme mit Linksrekursion (im Gegensatz zu einfachen Top-Down-Parsern)

## ■ Nachteile

- Etwas komplizierter zu implementieren als der CYK-Parser
- Wie bei allen Top-Down-Parsern viele überflüssige Top-Down-Voraussagen

## References I

- Linguistische Grundlagen: Abschnitt 3.2 aus Manning and Schütze (1999)
- Zum Parsing: Abschnitte 10.1–10.3 aus Jurafsky and Martin (2009)
- Chomsky-Normalform: Abschnitt 7.1.5 aus Hopcroft, Motwani, and Ullman (2001)
- CYK-Algorithmus: Abschnitt 7.4.4 aus Hopcroft, Motwani, and Ullman (2001)
- Earley-Algorithmus: Abschnitt 10.4 aus Jurafsky and Martin (2009)

## References II

- Hopcroft, J. E., R. Motwani, and J. D. Ullman (2001).  
*Introduction to Automata Theory, Languages, and Computation*  
(2nd ed.).  
Addison-Wesley.
- Jurafsky, D. and J. H. Martin (2009).  
*Speech and Language Processing: An Introduction to Natural  
Language Processing, Speech Recognition, and  
Computational Linguistics* (2nd ed.).  
Upper Saddle River, NJ: Prentice-Hall.
- Manning, C. D. and H. Schütze (1999).  
*Foundations of Statistical Natural Language Processing*.  
Cambridge, MA: The MIT Press.

# Symbolische und Statistische Verfahren (CL2)

## 11: Probabilistisches Parsing

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum

RUB

# Themenüberblick Parsing

- Parsing
  - **Probabilistisches Parsing**
  - Baumbanken
  - Evaluation

# Themenüberblick heute

- Probabilistisches Parsing
  - Probabilistische kontextfreie Grammatiken (PCFG)
  - Parsing mit PCFGs
  - Mögliche Erweiterungen von PCFGs
- Baumbanken
  - Penn Treebank, TIGER, TüBa-DZ
- Evaluation
  - Parseval, Leaf-Ancestor, Dependenz-basiert

# Gliederung

## 1 Probabilistische Grammatiken

- Parsing mit PCFGs
- Mögliche Erweiterungen von PCFGs

# Probabilistische Grammatiken

- Es gibt verschiedene Arten probabilistischer Grammatiken
- Wir werden uns hauptsächlich mit einfachen probabilistischen kontextfreien Grammatiken (PCFGs) beschäftigen
  - einige Erweiterungen am Ende der Sitzung

# Anwendungen probabilistischer Grammatiken

- 1 Probabilistische Disambiguierung: Auswahl der wahrscheinlichsten syntaktischen Struktur eines Satzes
- 2 Verwendung als Sprachmodell
- 3 → **Probabilistisches Parsing**

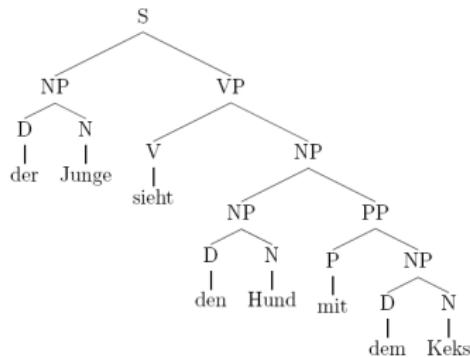
# Probabilistische Grammatiken zur Disambiguierung

Wahl der Baumstruktur  $T^*$  für eine Kette  $S$  von Wörtern  $w_1..w_n$  mit der maximalen Wahrscheinlichkeit, gegeben eine probabilistische Grammatik  $G_{prob}$

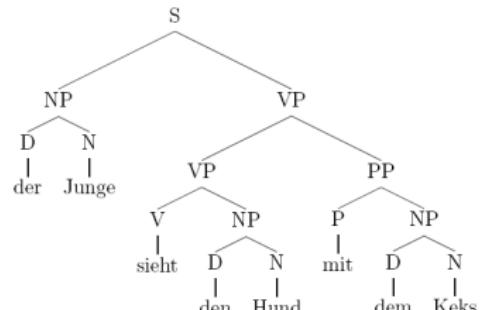
$$\begin{aligned} T^* &= \operatorname{argmax}_{T \in \tau_{Gprob}(S)} P_{Gprob}(T|S) \\ &= \operatorname{argmax}_{T \in \tau_{Gprob}(S)} P_{Gprob}(T) \end{aligned}$$

wobei  $\tau_{Gprob}(S)$  die Menge aller möglichen Baumstrukturen für  $S = w_1..w_n$  nach  $G_{prob}$  ist

# Disambiguierung: Welche ist die wahrscheinlichste Analyse?



PP modifiziert NP (*den Hund*)



PP modifiziert VP (*sieht den Hund*)

# Probabilistische Grammatiken als Sprachmodell

- Wahrscheinlichkeit für eine Kette von Wörtern  $S = w_1 \dots w_n$  gegeben eine Grammatik  $G_{prob}$

$$P_{G_{prob}}(S) = \sum_{T \in \tau_{G_{prob}}(S)} P(T, S)$$

Summe der Wahrscheinlichkeiten aller möglichen Baumstrukturen  $\tau$  der Kette  $S = w_1 \dots w_n$  nach  $G_{prob}$

- Eine mögliche Anwendung ist die Spracherkennung; dafür muss eine inkrementelle Berechnung für Präfixe der Gesamtkette möglich sein (vgl. z.B. Stolcke 1995 für den Earley-Algorithmus)

# Probabilistische kontextfreie Grammatiken

Eine **probabilistische kontextfreie Grammatik** (PCFG) ist ein 5-Tupel  $\langle N, \Sigma, P, S, D \rangle$  mit

- $N$ : eine Menge von **nicht-terminalen Symbolen**
- $\Sigma$ : eine Menge von **terminalen Symbolen**
- $P$ : eine Menge von **Regeln/Produktionen** der Form  
 $A \rightarrow \alpha$ , wobei A ein nicht-terminales Symbol ist:  $A \in N$   
und  $\alpha$  eine Folge von terminalen und/oder nicht-terminalen  
Symbolen:  $\alpha = (\Sigma \cup N)^*$
- $S$ : das **Startsymbol** ( $S \in N$ )
- $D$ : eine Funktion, die jeder Regel in der Grammatik eine  
**Wahrscheinlichkeit**  $P(A \rightarrow \alpha | A)$  zuweist, wobei gilt, dass  
die Summe der Wahrscheinlichkeiten aller Regeln mit  
derselben linken Seite A gleich 1 sein muss

# (Un-)Abhängigkeitsannahmen in Standard-PCFGs

Eine einfache probabilistische kontextfreie Grammatik macht folgende (vereinfachende) Annahmen:

- Die Wahrscheinlichkeit, dass ein nichtterminales Symbol  $N$  in die Symbole  $\alpha_1 \dots \alpha_n$  expandiert, hängt nur von der Kategorie von  $N$  ab:  $P(N \rightarrow \alpha_1 \dots \alpha_n | N)$
- D.h. die Wahrscheinlichkeit einer Expansion ist:
  - ... unabhängig von der Position im gesamten Baum:  
**Positionsinvarianz**

$$\forall k : P(N_{k(k+c)} \rightarrow \zeta) \text{ ist gleich}$$

( $N_{pq}$ : das Nichtterminal  $N$  überspannt die Positionen  $p$  bis  $q$  im Satz)

- ... unabhängig vom umgebenden Kontext, der nicht von  $N$  dominiert wird: **Kontextfreiheit**

$$P(N_{kl} \rightarrow \zeta | \text{etwas außerhalb von } k \dots l) = P(N_{kl} \rightarrow \zeta)$$

# (Un-)Abhängigkeitsannahmen in Standard-PCFGs (Forts.)

- D.h. die Wahrscheinlichkeit einer Expansion ist:

- ... unabhängig von Knoten oberhalb von  $N$ :  
**Vorfahrenfreiheit**

$$P(N_{kl} \rightarrow \zeta \mid \text{ein Vorfahrknoten von } N) = P(N_{kl} \rightarrow \zeta)$$

- ... unabhängig von dem durch  $N$  dominierten lexikalischen Material ( $N$  kein Präterminal): **nicht lexikalisiert**

$$P(N_{kl} \rightarrow \zeta \mid \text{String von k...l}) = P(N_{kl} \rightarrow \zeta)$$

# Probabilistische kontextfreie Grammatik (in CNF)

## Regeln

S	$\rightarrow$ NP VP	[.7]
S	$\rightarrow$ NP V	[.2]
S	$\rightarrow$ S Conj+S	[.1]
Conj+S	$\rightarrow$ Conj S	[1]
VP	$\rightarrow$ V NP	[.3]
VP	$\rightarrow$ V NP+NP	[.2]
VP	$\rightarrow$ VP PP	[.1]
VP	$\rightarrow$ V PP	[.1]
VP	$\rightarrow$ VP Conj+VP	[.15]
VP	$\rightarrow$ V Conj+VP	[.15]
Conj+VP	$\rightarrow$ Conj V	[.3]
Conj+VP	$\rightarrow$ Conj VP	[.7]
NP	$\rightarrow$ D N	[.6]
NP	$\rightarrow$ D AP+N	[.1]
NP	$\rightarrow$ NP Conj+NP	[.1]
NP	$\rightarrow$ NP PP	[.2]
NP+NP	$\rightarrow$ NP NP	[1]
Conj+NP	$\rightarrow$ Conj NP	[1]

PP	$\rightarrow$ P NP	[.8]
PP	$\rightarrow$ PP Conj+PP	[.2]
Conj+PP	$\rightarrow$ Conj PP	[1]
AP+N	$\rightarrow$ A N	[.8]
AP+N	$\rightarrow$ AP N	[.2]
AP	$\rightarrow$ AP Conj+AP	[1]
Conj+AP	$\rightarrow$ Conj AP	[.4]
Conj+AP	$\rightarrow$ Conj A	[.6]

## Lexikon

A	$\rightarrow$ kleine [.3], kleinen [.2], kleiner [.3], kleines [.2]
Conj	$\rightarrow$ und [1]
D	$\rightarrow$ das [.1], dem [.1], den [.1], der [.2], die [.2], ein [.1], eine [.1], einen [.1]
N	$\rightarrow$ Baum [.2], Blume [.1], Hund [.1], Junge [.2], Jungen [.05], Keks [.1], Kinder [.05], Mädchen [.2]
P	$\rightarrow$ hinter [.1], in [.5], mit [.2], vor [.2]
V	$\rightarrow$ geben [.2], gibt [.1], schläft [.2], schlafen [.3], sehen [.1], sieht [.1]

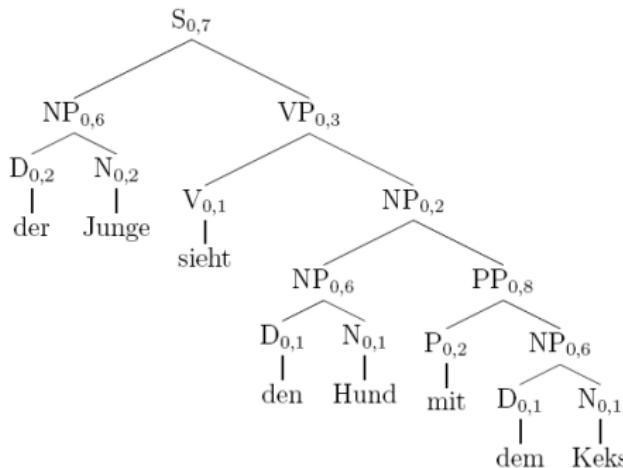
# Wahrscheinlichkeit einer Baumstruktur

Die Wahrscheinlichkeit einer möglichen Struktur eines Satzes  $S$  ist das Produkt der Wahrscheinlichkeiten aller Regeln, die in dieser Baumstruktur vorkommen

$$P_{pcfg}(T, S) = \prod_{n \in T} p(r(n))$$

- $n$ : nicht-terminale Knoten aus dem Baum  $T$
- $p(r(n))$ : Wahrscheinlichkeit der Expansion des Knotens  $n$  nach der Grammatik  $G_{pcfg}$

# Wahrscheinlichkeit einer Baumstruktur: Beispiel NP-PP

**Regeln**

S → NP VP [.7]

VP → V NP [.3]

VP → VP PP [.1]

NP → D N [.6]

NP → NP PP [.2]

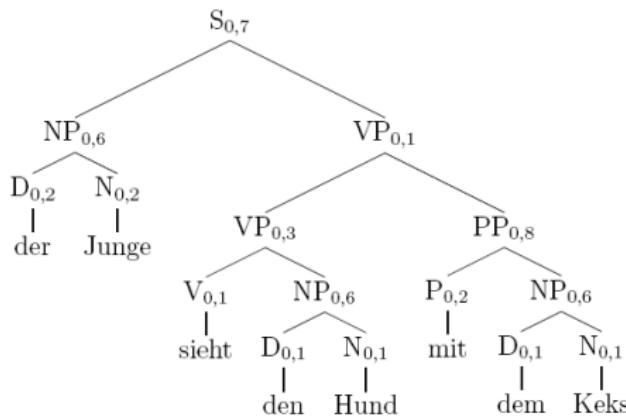
**Lexikon**

D → ..., der [.2], ...

...

$$\begin{aligned}
 P_{pcfg}(T_{NP-PP}) &= .7 \cdot .6 \cdot .2 \cdot .2 \cdot .3 \cdot .1 \cdot .2 \cdot .6 \cdot .1 \cdot .1 \cdot .8 \cdot .2 \cdot .6 \cdot .1 \cdot .1 \\
 &= 5.80608e-10
 \end{aligned}$$

# Wahrscheinlichkeit einer Baumstruktur: Beispiel VP-PP

**Regeln** $S \rightarrow NP\ VP$ 

[.7]

 $\dots$  $VP \rightarrow V\ NP$ 

[.3]

 $VP \rightarrow VP\ PP$ 

[.1]

 $\dots$  $NP \rightarrow D\ N$ 

[.6]

 $NP \rightarrow NP\ PP$ 

[.2]

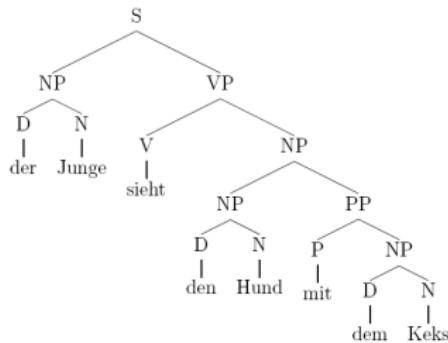
**Lexikon** $D \rightarrow \dots, der$ 

[.2], ...

 $\dots$ 

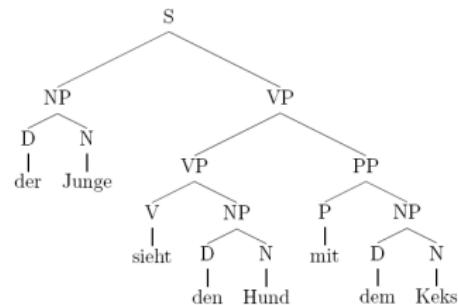
$$\begin{aligned}
 P_{pcfg}(T_{VP-PP}) &= .7 \cdot .6 \cdot .2 \cdot .2 \cdot .1 \cdot .3 \cdot .1 \cdot .6 \cdot .1 \cdot .1 \cdot .8 \cdot .2 \cdot .6 \cdot .1 \cdot .1 \\
 &= 2.90304e-10
 \end{aligned}$$

# Disambiguierung: Welche ist die wahrscheinlichste Analyse?



PP modifiziert NP (*den Hund*)  
 $P_{pcfg}(T_{NP-PP}) = 5.80608e-10$

→ nach der Grammatik  
**wahrscheinlicher!**



PP modifiziert VP (*sieht den Hund*)  
 $P_{pcfg}(T_{VP-PP}) = 2.90304e-10$

# Wahrscheinlichkeit eines Satzes (PCFG als Sprachmodell)

Die Wahrscheinlichkeit eines Satzes  $S$  ist die Summe der Wahrscheinlichkeiten aller möglichen Baumstrukturen nach der Grammatik

$$\begin{aligned} & P_{pcfg}(\text{"der Junge sieht den Hund mit dem Keks"}) \\ &= P_{pcfg}(T_{NP-PP}) + P_{pcfg}(T_{VP-PP}) \\ &= 5.80608e-10 + 2.90304e-10 \\ &= 8.70912e-09 \end{aligned}$$

# Training von probabilistischen Grammatiken

- Achtung: Die Wahrscheinlichkeiten der Beispielgrammatik oben waren willkürlich zugewiesen!
- Woher kommen die Wahrscheinlichkeiten, d.h. die Parameter, einer probabilistischen Grammatik?
  - 1. Möglichkeit: Training auf einem annotierten Korpus, einer sogenannten **Baumbank**
  - 2. Möglichkeit: Iteratives Training auf nicht annotierten Daten mit Hilfe des Inside-Outside-Algorithmus (Baker 1979; vgl. Manning and Schütze (1999))

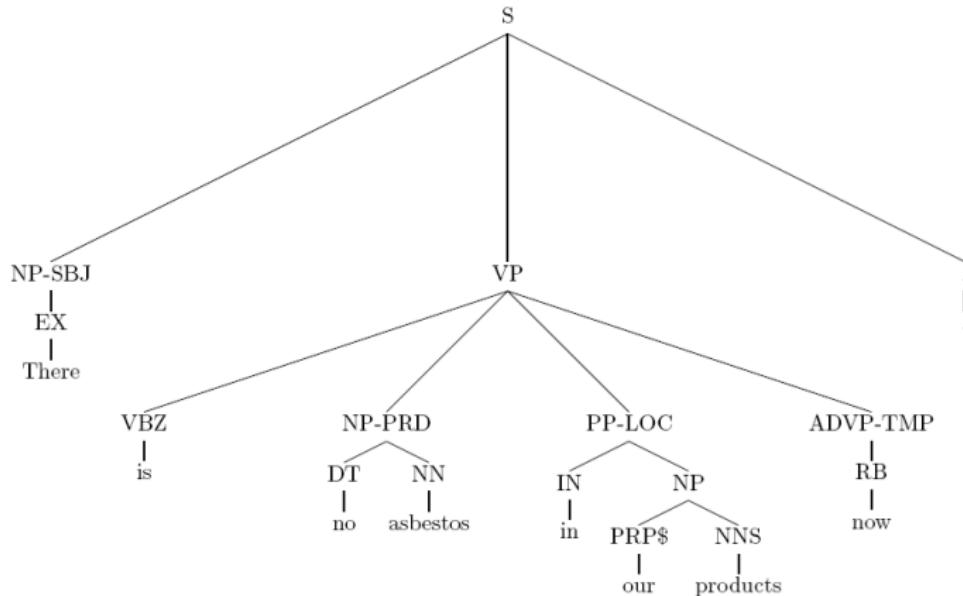
# Baumbanken: Penn Treebank

## Penn-Treebank-Format

- Stellt Teilbäume (Konstituenten) durch Klammerung dar
  - direkt nach öffnender Klammer: Mutterknoten
  - Rest bis zur schließenden Klammer: Subkonstituenten (Töchter) dieses Knotens
  - vgl. <http://www.cis.upenn.edu/~treebank/>

```
( S
( NP-SBJ ( EX There ) )
( VP ( VBZ is )
  ( NP-PRD ( DT no ) ( NN asbestos ) )
  ( PP-LOC ( IN in )
    ( NP ( PRP$ our ) ( NNS products ) ) )
( ADVP-TMP ( RB now ) )
( . . . ) )
```

# Baumbanken: Penn-Treebank



Mehr zu Baumbanken später

# Training von probabilistischen Grammatiken

- Zählen der Häufigkeit verschiedener Teilbäume in der Baumbank
- = zählen, wie häufig bestimmte Grammatikregeln angewandt worden sind
- Bedingte Wahrscheinlichkeit für eine Regel  $A \rightarrow \alpha$ : Häufigkeit dieses Teilbaums geteilt durch die Häufigkeit aller Teilbäume mit demselben Mutterknoten  $A$

$$P(A \rightarrow \alpha | A) = \frac{C(A \rightarrow \alpha)}{C(A)}$$

# Training von probabilistischen Grammatiken

## ■ Fiktives Beispiel

- NNP kommt 600 mal mit den Töchtern DT NN vor
- und 400 mal nur mit der Tochter NN
- insgesamt gibt es also 1000 Teilbäume mit dem Mutterknoten NNP

## ■ Dann haben die beiden kontextfreien Regeln die folgenden Wahrscheinlichkeiten:

- $P(NNP \rightarrow DT\ NN|NNP) = 600/1000 = 0.6$
- $P(NNP \rightarrow NN|NNP) = 400/1000 = 0.4$

# Gliederung

## 1 Probabilistische Grammatiken

- Parsing mit PCFGs
- Mögliche Erweiterungen von PCFGs

# Parsing mit PCFGs

## ■ Naiver Ansatz

- Parsing der Eingabekette zunächst ohne Berechnung von Wahrscheinlichkeiten (Chart-Parsing)
- Sammeln aller möglichen Analysen nach der Grammatik
- Berechnen der Wahrscheinlichkeit für jede Analyse
- evtl. Auswahl der wahrscheinlichsten Struktur

## ■ Probleme

- sehr ineffiziente Berechnung der Wahrscheinlichkeiten
- bei realistischen Sätzen und einer realistischen Grammatik kann es Tausende von möglichen Analysen geben ("Ambiguitätsexplosion")
- Chart-Parsing ist effizient, nachträgliche Berechnung der Wahrscheinlichkeiten nicht

# Parsing mit PCFGs

- Effizienterer Ansatz: Integration der Berechnung der Wahrscheinlichkeiten in einen Chart-Parsing-Algorithmus
- Vorteile
  - gleichzeitiges Parsing und Berechnung der Wahrscheinlichkeiten
  - die Wahrscheinlichkeit für einen Teilbaum wird nur einmal berechnet und dann zur Wiederverwendung gespeichert (dynamische Programmierung)
  - die wahrscheinlichste Struktur kann nach erfolgreichem Parsing direkt aus der Chart ausgelesen werden
  - Einschränkung des Suchraums möglich (z.B. "Beam Search": nicht alle Zwischenresultate werden gespeichert)

# Probabilistischer Cocke-Younger-Kasami-Parser

- Zunächst ist eine Umwandlung der PCFG in die Chomsky-Normalform nötig
  - dabei Wahrscheinlichkeiten korrekt behandeln!
- Die einzelnen Einträge in den Zellen der Chart enthalten nun auch die berechnete Wahrscheinlichkeit und haben folgendes Format:
  - (*Kategorie, Verweis\_auf\_linke\_Tochter, Verweis\_auf\_rechte\_Tochter, Wahrscheinlichkeit*)
  - Beispiel: ('S', → NP\_1[1, 2], → VP\_1[3, 8], 5.80608e-10)
- In der unten folgenden Beispielchart sind Verweise auf die Töchter durch Pfeile dargestellt

# Probabilistische kontextfreie Grammatik (in CNF)

## Regeln

S	$\rightarrow$ NP VP	[.7]
S	$\rightarrow$ NP V	[.2]
S	$\rightarrow$ S Conj+S	[.1]
Conj+S	$\rightarrow$ Conj S	[1]
VP	$\rightarrow$ V NP	[.3]
VP	$\rightarrow$ V NP+NP	[.2]
VP	$\rightarrow$ VP PP	[.1]
VP	$\rightarrow$ V PP	[.1]
VP	$\rightarrow$ VP Conj+VP	[.15]
VP	$\rightarrow$ V Conj+VP	[.15]
Conj+VP	$\rightarrow$ Conj V	[.3]
Conj+VP	$\rightarrow$ Conj VP	[.7]
NP	$\rightarrow$ D N	[.6]
NP	$\rightarrow$ D AP+N	[.1]
NP	$\rightarrow$ NP Conj+NP	[.1]
NP	$\rightarrow$ NP PP	[.2]
NP+NP	$\rightarrow$ NP NP	[1]
Conj+NP	$\rightarrow$ Conj NP	[1]

PP	$\rightarrow$ P NP	[.8]
PP	$\rightarrow$ PP Conj+PP	[.2]
Conj+PP	$\rightarrow$ Conj PP	[1]
AP+N	$\rightarrow$ A N	[.8]
AP+N	$\rightarrow$ AP N	[.2]
AP	$\rightarrow$ AP Conj+AP	[1]
Conj+AP	$\rightarrow$ Conj AP	[.4]
Conj+AP	$\rightarrow$ Conj A	[.6]

## Lexikon

A	$\rightarrow$ kleine [.3], kleinen [.2], kleiner [.3], kleines [.2]
Conj	$\rightarrow$ und [1]
D	$\rightarrow$ das [.1], dem [.1], den [.1], der [.2], die [.2], ein [.1], eine [.1], einen [.1]
N	$\rightarrow$ Baum [.2], Blume [.1], Hund [.1], Junge [.2], Jungen [.05], Keks [.1], Kinder [.05], Mädchen [.2]
P	$\rightarrow$ hinter [.1], in [.5], mit [.2], vor [.2]
V	$\rightarrow$ geben [.2], gibt [.1], schläft [.2], schlafen [.3], sehen [.1], sieht [.1]

# Probabilistischer CYK-Algorithmus: Initialisierung

Initialisierung der Diagonale der Chart mit allen möglichen Wortarten, die das jeweilige Wort haben kann, und deren Wahrscheinlichkeiten aus der PCFG:

Für  $I = 1 \dots n$

Für jede Regel  $A \rightarrow w$   $[P_{Regel}] \in G$

Falls  $w = w_I$ , setze  $[I, I] := [I, I] \cup \{A [P_{Regel}]\}$

$D \rightarrow \text{der}$	[.2]
$N \rightarrow \text{Junge}$	[.2]
$V \rightarrow \text{sieht}$	[.1]
$D \rightarrow \text{den}$	[.1]
$N \rightarrow \text{Hund}$	[.1]
$P \rightarrow \text{mit}$	[.2]
$D \rightarrow \text{dem}$	[.1]
$N \rightarrow \text{Keks}$	[.1]

(hier: keine lexikalischen Ambiguitäten)

# Probabilistischer CYK-Algorithmus: Initialisierung

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]							
2 Junge		N [0,2]						
3 sieht			V [0,1]					
4 den				D [0,1]				
5 Hund					N [0,1]			
6 mit						P [0,2]		
7 dem							D [0,1]	
8 Keks								N [0,1]

# Probabilistischer CYK-Algorithmus: Induktion

Für  $l = 2 \dots n$

für jede Kette der Länge  $l$

Für  $i = 1 \dots n - l + 1$

für alle entsprechenden Zellen  $[i,j]$

Setze  $j := i + l - 1$

Für  $k = i \dots i + l - 2$

für alle Teilstrings  $[i,k], [k+1,j]$

Setze  $[i,j] := [i,j] \cup \{A [P_A] | A \rightarrow BC [P_{Regel}] \in G,$   
 $B [P_B] \in [i,k], C [P_C] \in [k+1,j]$   
 $\text{mit } P_A := P_B \times P_C \times P_{Regel}\}$

# Probabilistischer CYK-Algorithmus

- Dieser Algorithmus: ermittelt die Wahrscheinlichkeiten aller einzelnen Lösungen
- Alternativ: für jede Kategorie maximal *einen* Eintrag in ein und derselben Zelle zulassen, wobei man nur den Eintrag mit der höchsten Wahrscheinlichkeit behält
- Bei einem erfolgreichen Parse: am Ende die wahrscheinlichste Analyse direkt aus der Zelle  $[1, n]$  ablesen
- Wenn man die Wahrscheinlichkeit der Kette berechnen möchte (Sprachmodell): die Wahrscheinlichkeiten mehrerer Einträge derselben Kategorie in einer Zelle aufzaddieren

# Probabilistischer CYK-Parser: Initialisierung

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]							
2 Junge		N [0,2]						
3 sieht			V [0,1]					
4 den				D [0,1]				
5 Hund					N [0,1]			
6 mit						P [0,2]		
7 dem							D [0,1]	
8 Keks								N [0,1]

# Prob. CYK: Teilstrings der Länge 2

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]	NP[0,024]						
2 Junge		N [0,2]						
3 sieht			V [0,1]					
4 den				D [0,1]				
5 Hund					N [0,1]			
6 mit		NP [0,2*0,2*0,6 = 0,024]				P [0,2]		
7 dem							D [0,1]	
8 Keks								N [0,1]

# Prob. CYK: Teilstrings der Länge 3

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]	NP[0,024]	S [0,00048]					
2 Junge		N [0,2]						
3 sieht			V [0,1]					
4 den				D [0,1]	NP[0,006]			
5 Hund					N [0,1]			
6 mit	S [0,024*0,1*0,2 = 0,00048]				P [0,2]			
7 dem						D [0,1]	NP[0,006]	
8 Keks							N [0,1]	

# Prob. CYK: Teilstrings der Länge 4

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]	NP[0,024]	S [0,00048]					
2 Junge		N [0,2]						
3 sieht			V [0,1]		VP [0,00018]			
4 den				D [0,1]	NP[0,006]			
5 Hund					N [0,1]			
6 mit						P [0,2]	PP [0,00096]	
7 dem							D [0,1]	NP[0,006]
8 Keks								N [0,1]

# Prob. CYK: Teilstrings der Länge 5

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]	NP[0,024] [0,00048]	S		S [3e-06]			
2 Junge		N [0,2]						
3 sieht			V [0,1]		VP [0,00018]			
4 den				D [0,1]	NP[0,006]			
5 Hund					N [0,1]			
6 mit			S [0,024*0,00018*0,7 = 3,024e-06]		P [0,2]	PP [0,00096]		
7 dem						D [0,1]	NP[0,006]	
8 Keks							N [0,1]	

# Prob. CYK: Teilstrings der Länge 6

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]	NP [0,024]	S [0,00048]		S [3e-06]			
2 Junge		N [0,2]						
3 sieht			V [0,1]		VP [0,00018]			VP [3,5e-8]
4 den				D [0,1]	NP [0,006]			NP [1,15e-06]
5 Hund					N [0,1]			
6 mit						P [0,2]		PP [0,00096]
7 dem							D [0,1]	NP [0,006]
8 Keks								N [0,1]

VP [0,1 \* 1,15e-06 \* 0,3 = 3,456e-08]

# Prob. CYK: Teilstrings der Länge 7

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]	NP[0,024]	S [0,00048]		S [3e-06]			
2 Junge		N [0,2]						
3 sieht			V [0,1]		VP [0,00018]			VP[3,5e-8] VP[1,7e-8]
4 den				D [0,1]	NP[0,006]			NP [1,15e-06]
5 Hund					N [0,1]			
6 mit						P [0,2]		PP [0,00096]
7 dem							D [0,1]	NP[0,006]
8 Keks								N [0,1]

# Prob. CYK: Teilstrings der Länge 8

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]	NP[0,024] [0,00048]	S [0,00048]		S [3e-06]			S[5,8e-10]
2 Junge		N [0,2]						
3 sieht			V [0,1]		VP [0,00018]			VP[3,5e-8] VP[1,7e-8]
4 den				D [0,1]	NP[0,006]			NP [1,15e-06]
5 Hund					N [0,1]			
6 mit						P [0,2]		PP [0,00096]
7 dem							D [0,1]	NP[0,006] N [0,1]
8 Keks								

S [0,024\*3,456e-08\*0,7 = 5,80608e-10]

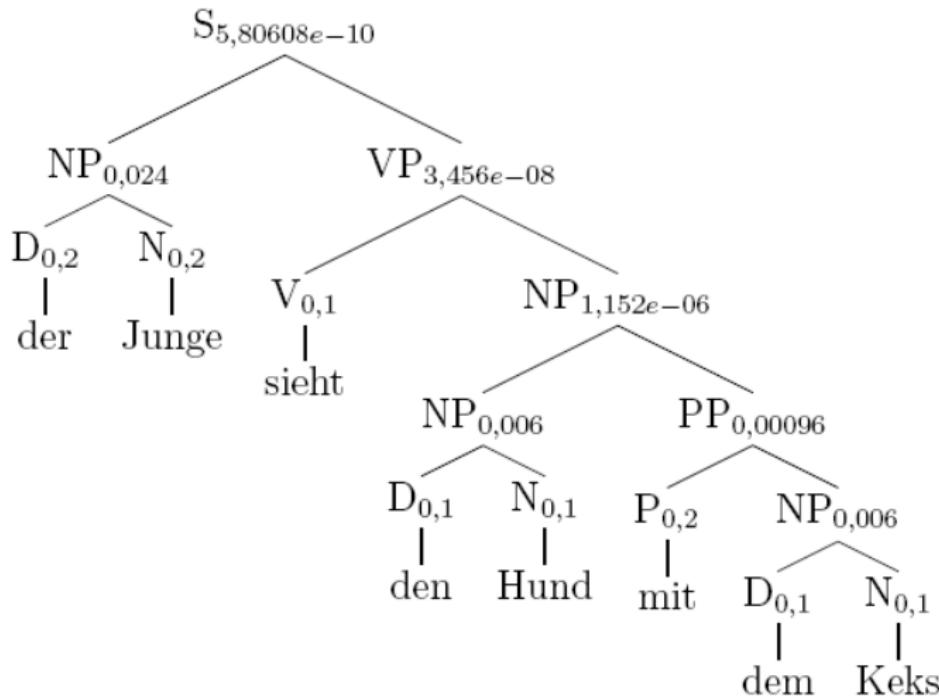
# Prob. CYK: Wahrscheinlichster Baum

	1 der	2 Junge	3 sieht	4 den	5 Hund	6 mit	7 dem	8 Keks
1 der	D [0,2]	NP[0,024]	S [0,00048]		S [3e-06]			S[5,8e-10] S[2,9e-10]
2 Junge		N [0,2]						
3 sieht			V [0,1]		VP [0,00018]			VP[3,5e-8] VP[1,7e-8]
4 den				D [0,1]	NP[0,006]			NP [1,15e-06]
5 Hund					N [0,1]			
6 mit						P [0,2]		PP [0,00096]
7 dem							D [0,1]	NP[0,006]
8 Keks								N [0,1]

# Disambiguierung: Wahrscheinlichster Baum

- Suche in der rechten oberen Zelle  $[1, n]$  nach dem Eintrag des Startsymbols mit der größten Wahrscheinlichkeit  
 $\rightarrow S[5.80608e-10]$
- Verfolge rekursiv die Verweise von diesem Eintrag, um den wahrscheinlichsten Baum zu rekonstruieren

# Disambiguierung: Wahrscheinlichster Baum



# Gliederung

## 1 Probabilistische Grammatiken

- Parsing mit PCFGs
- Mögliche Erweiterungen von PCFGs

# Unabhängigkeitsannahmen in Standard-PCFGs

- **Positionsinvarianz, Kontextfreiheit, Vorfahrenfreiheit:**  
d.h. die Wahrscheinlichkeit einer Expansion hängt nicht ab von der Position im gesamten Baum, vom umgebenden Kontext oder von Knoten oberhalb von  $N$
- Das ist klar falsch!
- Z.B. Subjekt-NPs sind viel häufiger pronominal als Objekt-NPs

	Pronomen	Volle NP
Subjekt	91%	9%
Objekt	34%	66%

(Daten aus Switchboard-Korpus, Francis et al. 1999, zitiert nach Jurafsky and Martin (2009))

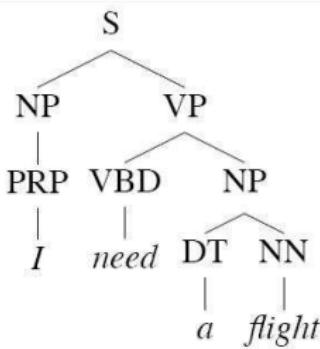
# Mögliche Lösungen

- Unterscheidung mehrerer Kategorien, z.B: NP-SBJ vs. NP-OBJ (sog. funktionale Labels)
- Berücksichtigung der “Großmutter”, d.h. Aufgeben der “Vorfahrenfreiheit”:

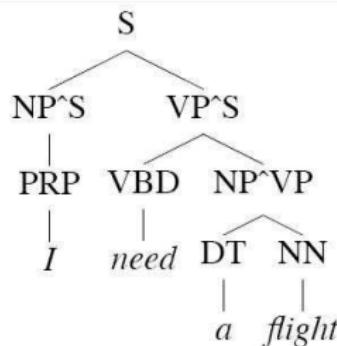
NP[unter S]	→ PRON	[.8]
NP[unter S]	→ D N	[.2]
NP[unter VP]	→ PRON	[.3]
NP[unter VP]	→ D N	[.7]

# Berücksichtigung der “Großmutter”

a)



b)



Jurafsky and Martin (2009)

# Unabhängigkeitsannahmen in Standard-PCFGs

- **Nicht lexikalisiert:** Die Wahrscheinlichkeit einer Expansion hängt nicht ab von dem durch N dominierten lexikalischen Material
- Auch falsch!
- Beispiel
  - Hans sieht den Hund mit der Leine
  - Hans isst das Steak mit der Gabel
- Die Wahrscheinlichkeit, dass die PP das Verb oder das Nomen modifiziert, hängt u.a. von dem spezifischen Verb und Nomen ab!

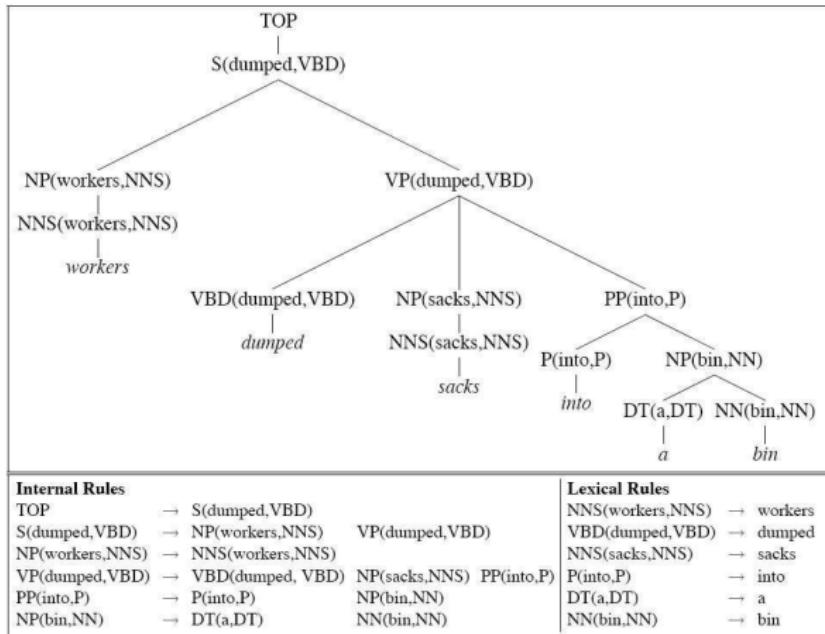
# Mögliche Lösung

Lexikalisierung der Grammatik:

- Der lexikalische Kopf jeder Phrase wird als weiterer Parameter für die bedingten Regelwahrscheinlichkeiten hinzugefügt
- Die Wahrscheinlichkeit für eine Expansion eines nichtterminalen Knotens  $n$  hängt dann sowohl von der Kategorie von  $n$  als auch dem lexikalischen Kopf von  $n$ ,  $h(n)$ , ab:

$$p(r(n)|n, h(n))$$

# Kopf-lexikalisierte PCFG



aus Jurafsky and Martin (2009)

# Probabilistische Grammatiken: Zusammenfassung

- ... helfen dabei, mit der exponentiellen Explosion von strukturellen Ambiguitäten umzugehen
- ... können als Sprachmodell benutzt werden, das der Komplexität menschlicher Sprache prinzipiell besser entspricht ist als Markov-Modelle
- Verschiedene Varianten probabilistischer kontextfreier Grammatiken
  - u.a.: Grammatiken mit globalem Wahrscheinlichkeitsmodell, z.B. auf der Basis von sog. Maximum-Entropy-Modellen
- Auch: probabilistisch gesteuerte Parser, die nur eine einzige Struktur aufbauen
  - die jeweils nächste Parsing-Operation wird mit Hilfe eines Wahrscheinlichkeitsmodells ausgewählt

# Zusammenfassung

- Probabilistische kontextfreie Grammatiken
  - Algorithmus: erweiterter CYK
  - Training auf Baumbank
- Mögliche Erweiterungen probabilistischer kontextfreier Grammatiken
  - z.B. Lexikalisierung

## References I

Abschnitte 11 und 12 aus Manning and Schütze (1999)

Jurafsky, D. and J. H. Martin (2009).

*Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics* (2nd ed.).  
Upper Saddle River, NJ: Prentice-Hall.

Manning, C. D. and H. Schütze (1999).

*Foundations of Statistical Natural Language Processing.*  
Cambridge, MA: The MIT Press.

# Symbolische und Statistische Verfahren (CL2)

## 12: Baumbanken und Evaluation

Stefanie Dipper

Sprachwissenschaftliches Institut  
Ruhr-Universität Bochum

RUB

# Themenüberblick Parsing

- Parsing
  - Probabilistisches Parsing
  - **Baumbanken**
  - **Evaluation**
- CL in Deutschland und der Welt

# Themenüberblick heute

- Probabilistisches Parsing
  - Probabilistische kontextfreie Grammatiken (PCFG)
  - Parsing mit PCFGs
  - Mögliche Erweiterungen von PCFGs
- Baumbanken
  - Penn Treebank, TIGER, TüBa-DZ
- Evaluation
  - Parseval, Leaf-Ancestor, Dependenz-basiert

# Gliederung

## 1 Baumbanken

## 2 Evaluation

# Trainingsdaten

- Trainingsdaten für eine PCFG: **Baumbank (Treebank)**
  - Korpus (“Datenbank”) mit Syntaxbäumen
- Diverse Baumbanken verfügbar
  - z.B. Liste unter <https://en.wikipedia.org/wiki/Treebank>
- Aber: viele unterschiedliche Formate,  
Annotationskonventionen, ...
- Daher: oft Präprozessierungsschritte notwendig
  - damit die Weiterverarbeitung klappt (Formatkonversion)
  - damit die Ergebnisse vergleichbar sind (Interpretation)

# Baumbanken: Penn Treebank

- Die “Mutter” aller Baumbanken
- Englische Baumbank, University of Pennsylvania
- Texte: Wall Street Journal, Brown Corpus, Switchboard Corpus (Telefon-Gespräche)
  - 4.5 Mio Tokens POS-tagged, 2/3 davon auch syntaktisch annotiert
  - automatisch vorgeparst, manuell nachkorrigiert
- Annotation in mehreren Phasen (1989-2000)
  - sukzessive Erweiterung des Annotationsschemas

# Besonderes Merkmal: leere Elemente

- Für logische Subjekte von Infinitiven und Imperativen
- Für “Spuren” (traces) von wh-Bewegung
- Beispiele
  - \* *Put it on the table*
  - *What is Tim eating* \*T\*?

```
(S (NP *)  
  (VP Put  
    (NP it)  
    (PP on the table)))
```

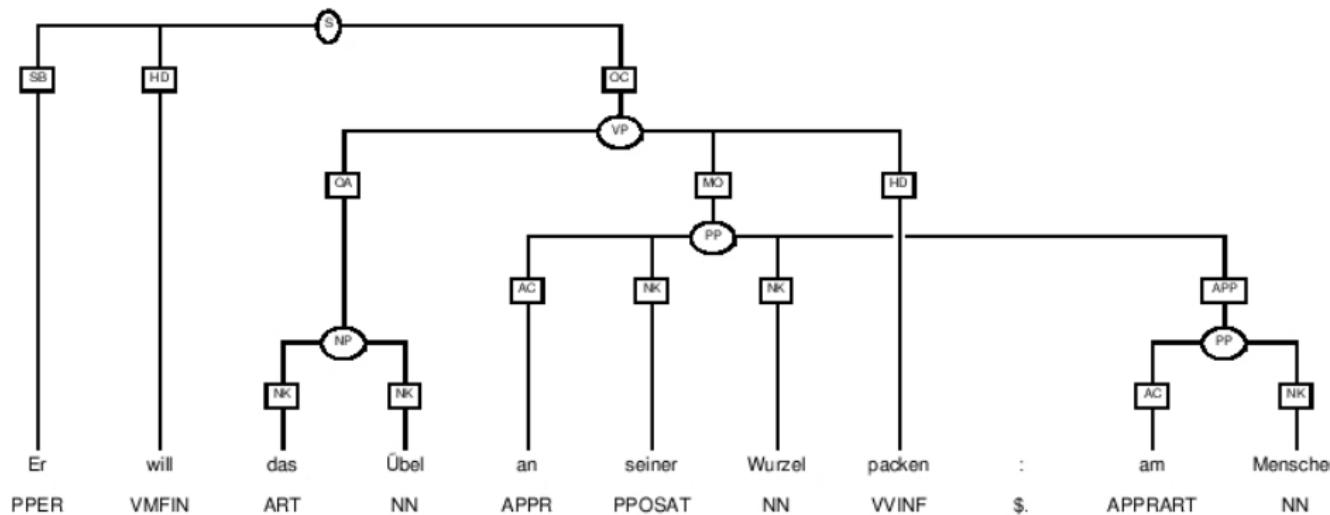
```
(SBARQ (WHNP-1 What)  
  (SQ is  
    (NP-SBJ Tim)  
    (VP eating  
      (NP *T*-1)))  
  ?)
```

- Problem: unbekannte (neue) zu parsende Sätze enthalten aber keine leeren Elemente! → Konversion

# Deutsche Baumbanken: TIGER

- Annotationsprojekt in Saarbrücken/Stuttgart/Potsdam, 1999–2004
  - ca. 50.000 Sätze/1 Mio Tokens, Frankfurter Rundschau
  - semi-automatisch annotiert: POS (STTS), Morph, Lemma, Syntaxbäume
- Hybrides Annotationsschema
  - Konstituentenstruktur kombiniert mit diskontinuierlichen Konstituenten
  - flache Analyse: “redundante” Knoten ausgelassen

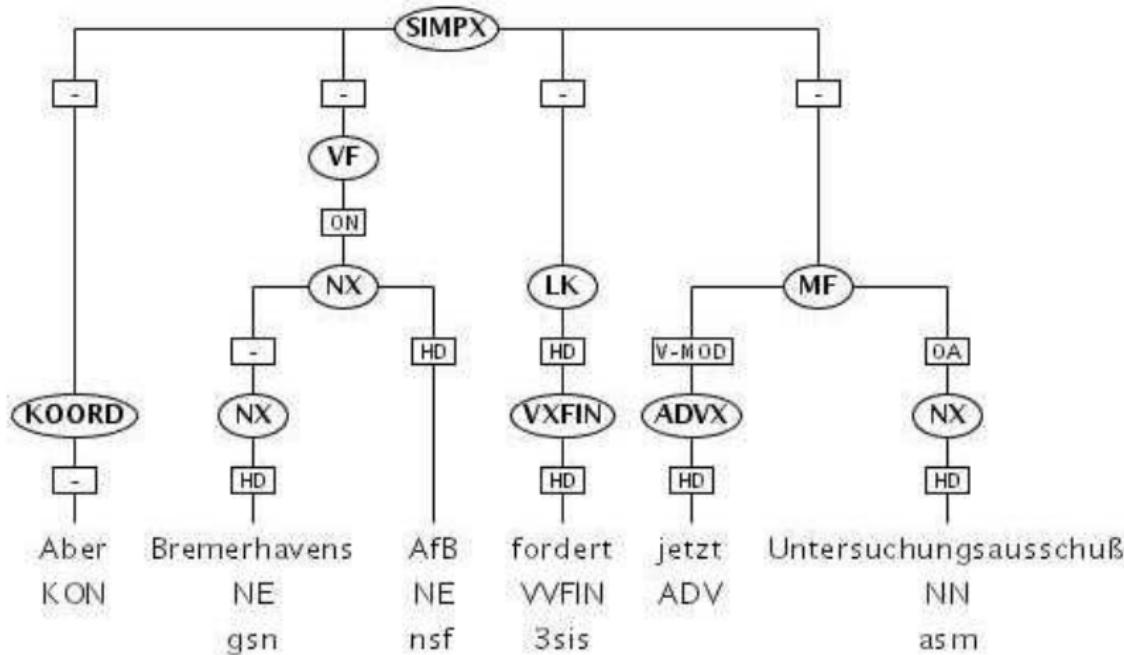
# Deutsche Baumbanken: TIGER



# TüBa-DZ

- “Tübinger Baumbank des Deutschen/Zeitungskorpus”, Universität Tübingen
  - knapp 100.000 Sätze/1.8 Mio Tokens (Release 10.0), taz
  - POS (STTS), Morph, Syntaxbäume
  - Named Entities, Koreferenz, Diskurskonnektoren
- Annotationsschema
  - komplexe Labels für lange Abhängigkeiten (V-MOD, OA-MOD)
  - **topologische Felder** (Vorfeld, Mittelfeld, . . . )

# TüBa-DZ



# Gliederung

## 1 Baumbanken

## 2 Evaluation

# Evaluation

Wie evaluiert man Parser?

- Die Maße TP, FP, F-Score etc. sind für die Evaluation von Klassifikatoren definiert und werden tokenbasiert berechnet
- Parsing: gelabelte Bäume, d.h. komplexe Strukturen
- Wie soll hier Accuracy berechnet werden?
- Zudem: oft (kleinere) Unterschiede in den Analysen, je nach Grammatiktheorie
- Wie sollen die Ergebnisse von Parsern, die auf verschiedenen Baumbanken trainiert wurden, verglichen werden?

# Evaluation: verschiedene Methoden

Baumvergleich: einfache Accuracy/Fehlerrate

- D.h. vom Parser erzeugter Baum muss 100% mit Gold-Standard-Baum übereinstimmen
- Vorteil: einfach zu berechnen
- Nachteil: es muss alles korrekt sein, teilweise korrekte Strukturen bekommen keine Punkte

# Evaluation: verschiedene Methoden

Liste mit grammatischen und ungrammatischen Sätzen (evtl. mit Angaben zur Zahl der verschiedenen Analysen/Lesarten), darauf Accuracy berechnen

- D.h. kein direkter Vergleich von Gold-Standard-Strukturen mit den erzeugten Strukturen
- Vorteil: es wird von Eigenheiten der Grammatik abstrahiert
- Nachteil: falsche Analysen können als korrekt zählen, da nur die Zahl der Analysen, nicht die Analyse selbst berücksichtigt wird

# Evaluation: verschiedene Methoden

Extraktion relevanter grammatischer Funktionen

- D.h. nur ausgewählte Merkmale (z.B. welche Konstituente ist Subjekt) werden evaluiert
- Vorteil: Vergleich von Parsern möglich, die mit verschiedenen Grammatiken arbeiten
- Nachteil: aufwändig

# Parseval

Vergleich von Bäumen mit dem Maß **Parseval**  
(Black et al. 1991)

- “Labelled Parseval”: berücksichtigt auch die Labels (Kategorien)
- “Unlabelled Parseval”: berücksichtigt nur die Spans
- Eine System-Konstituente zählt als korrekt, wenn sie dieselben Wörter wie eine Gold-Konstituente überspannt
- POS-Tags nicht mit evaluiert (Trennung Tagging – Parsing)

Hinweis: das originale Parseval-Maß involviert außerdem verschiedene Baum-“Normalisierungen”

- z.B. Löschen von Interpunktions, leeren Kategorien, Auxiliaries, unären Kanten, ...

# Parseval

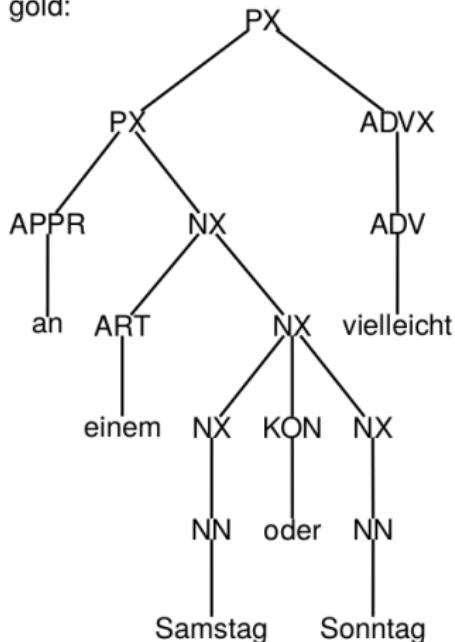
Parseval: Maße für Precision, Recall, Anzahl kreuzender Kanten

- Precision =  $\frac{C(\text{correct constituents})}{C(\text{constituents in parser output})}$
- Recall =  $\frac{C(\text{correct constituents})}{C(\text{constituents in gold standard})}$
- (Relative) Cross Brackets =  $\frac{C(\text{system const. that cross gold const.})}{C(\text{system constituents})}$

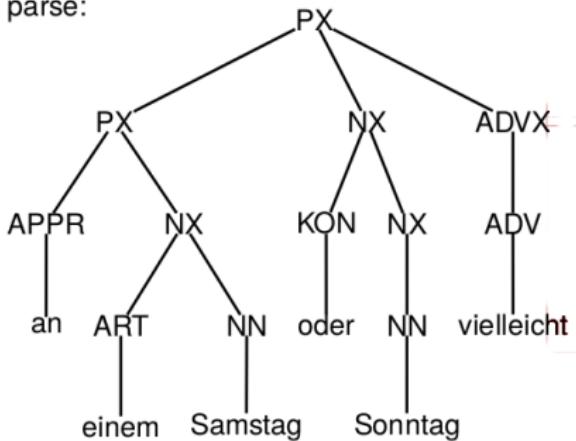
- Problem: bevorzugt tiefe Bäume mit mehr Klammern
- dann “zählt” eine falsche Klammer nicht viel

# Parseval: Beispiel

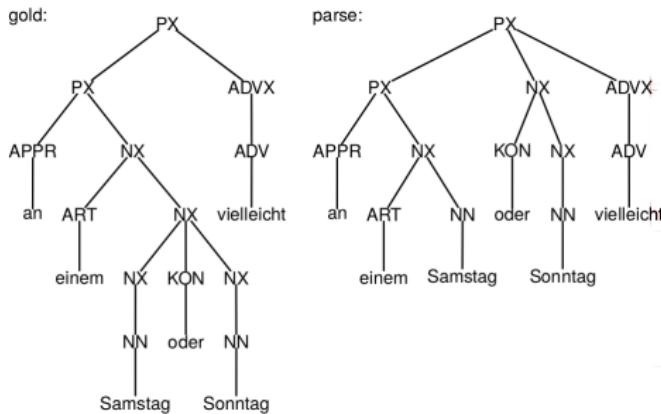
gold:



parse:



# Parseval: Beispiel



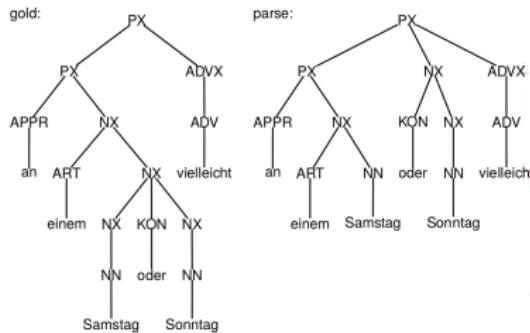
- Korrekte Konstituenten? → zunächst Spans der Konstituenten bestimmen

GOLD	Span	SYSTEM	Span
PX	1–6	PX	1–6
PX	1–5	PX	1–3
NX	2–5	NX	2–3
NX	3–5	NX	4–5
NX	3	NX	5
NX	5	ADGX	6
ADGX	6		
SUMME	7	SUMME	6

# Parseval: Beispiel

- Korrekte Konstituenten: 3
- Precision =  $\frac{C(\text{correct constituents})}{C(\text{constituents in parser output})} = \frac{3}{6} = 0.5$
- Recall =  $\frac{C(\text{correct constituents})}{C(\text{constituents in gold standard})} = \frac{3}{7} = 0.43$
- F-Score =  $\frac{2*P*R}{P+R} = 0.46$

# Parseval: Beispiel



- Kreuzende Kanten? → Span-Overlaps bestimmen

GOLD	Span	SYSTEM	Span
PX	1–6	PX	1–6
PX	1–5	PX	1–3 (vs. 3–5 (und 2–5))
NX	2–5	NX	2–3 (vs. 3–5)
NX	3–5	NX	4–5 (kompatibel, da Substring)
NX	3		
NX	5	NX	5
AD VX	6	AD VX	6
SUMME	7	SUMME	6

# Parseval: Beispiel

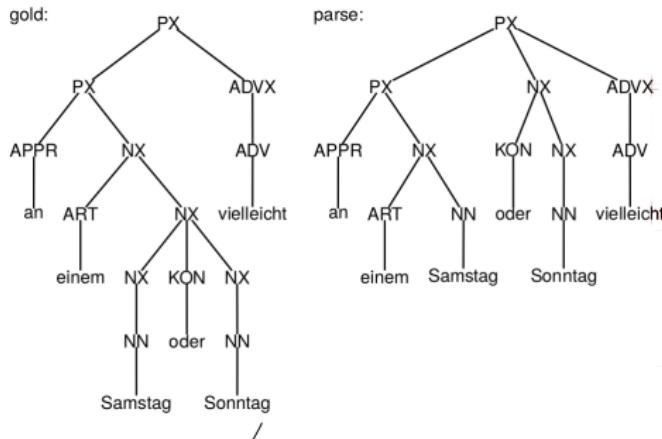
- Kreuzende Klammern: 2
- (Relative) Cross Brackets
  - =  $\frac{C(\text{system const. that cross gold const.})}{C(\text{system const})}$
  - =  $\frac{2}{6} = 0.33$

# Leaf-Ancestor metric

Vergleich von Bäumen mit **Leaf-Ancestor metric** (LA)  
(Sampson and Babarczy 2003)

- Betrachte Pfad ('lineage') von Nonterminalen zwischen einem Terminalknoten und dem Wurzelknoten
    - Klammern '[' bzw. ']' zeigen den höchsten Knoten an, an dessen linker bzw. rechter Kante das Terminal liegt (s.u.)
    - '[' wird vor und ']' hinter diesem höchsten Knoten eingefügt
  - Berechne Levenshtein-Distanz (Edit Distance) zwischen den System-Pfaden und den Gold-Pfaden
    - Löschen/einfügen: 1
    - Ersetzen (= löschen + einfügen): 2
    - Ausnahme: "verwandte Symbole" ersetzen: 0.5
  - Pro Wort:  $\text{Simil} = 1 - \frac{\text{Kosten}}{\text{Gesamtanzahl Knoten}}$
- Problem: bevorzugt ebenfalls tiefe Bäume
- bei langen Pfaden "zählt" ein falscher Pfadeintrag nicht so viel

# Leaf-Ancestor metric: Beispiel



Terminal	Gold	: System	Kosten	#Knoten	Simil.
<i>an</i>	PX [ PX	: PX [ PX	0	6	1
<i>einem</i>	[ NX PX PX	: [ NX PX PX	0	8	1
<i>Samstag</i>	NX [ NX NX PX PX	: NX PX ] PX	4	10	.60
<i>oder</i>	NX NX PX PX	: [ NX PX	3	7	.57
<i>Sonntag</i>	NX NX <u>NX PX</u> ] PX	: NX NX ] PX	2	10	.80
<i>vielleicht</i>	[ ADVX PX ]	: [ ADVX PX ]	0	8	1
<b>DURCHSCHNITT</b>					<b>.83</b>

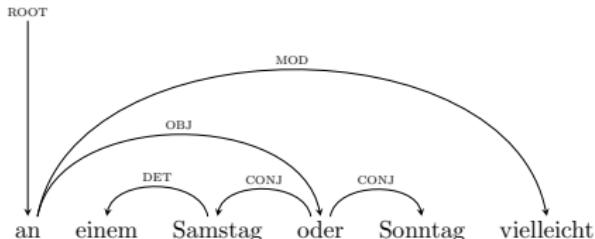
# Dependenz-basierte Evaluation

Vergleich von Bäumen auf Basis von **Dependenz-Relationen**

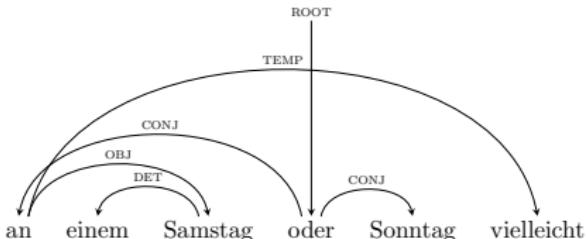
- Extrahiere Triples <Wort, Kopf, Relation> aus der Baumbank
- Maße:
  - 1 **LAS**: labelled attachment score: Anzahl Wörter mit dem korrekten Head und Dependenz-Label
  - 2 **UAS**: unlabelled: Anzahl Wörter mit dem korrekten Head
  - 3 LabAcc: Anzahl Wörter mit dem korrekten Dependenz-Label

# Dependenz-basierte Evaluation

Gold



System



- (Dependenzrepräsentation erstellen und) Triples extrahieren
- Dependenzen (hier): erfunden ...

Gold	System
<an,--,ROOT>	<an,oder,CONJ>
<einem, Samstag,DET>	<einem, Samstag,DET>
<Samstag,oder,CONJ>	<Samstag,an,OBJ>
<oder,an,OBJ>	<oder,--,ROOT>
<Sonntag,oder,CONJ>	<Sonntag,oder,CONJ>
<vielleicht,an,MOD>	<vielleicht,an,TEMP>

→ LAS: korrekte Triples (ink. Labels):  $\frac{2}{6} = .33$

→ UAS: korrekte Triples (ohne Labels):  $\frac{3}{6} = .50$

# CL in Deutschland und der Welt

Zum Schluss: CL in Deutschland und der Welt: einige Hinweise

# Wichtige Organisationen in der CL

- Association for Computational Linguistics (ACL)  
<http://www.aclweb.org>
- European Chapter of the ACL (EACL)  
<http://www.eacl.org>
- Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft (DGfS)  
[http://www.linguistics.ruhr-uni-bochum.de/  
dgfs-cl](http://www.linguistics.ruhr-uni-bochum.de/dgfs-cl)
- Gesellschaft für Sprachtechnologie und Computerlinguistik  
<http://www.gscl.org/>

# Konferenzen und Zeitschriften

- ACL: "Annual Meetings of the Association for Computational Linguistics", jährliche internationale Konferenz, seit 1963
- COLING: "International Conference on Computational Linguistics", zweijährliche internationale Konferenz, seit 1965
- Zeitschrift: Computational Linguistics  
<http://www.mitpressjournals.org/loi/coli>
- ACL-Anthologie — Sammlung von CL-Artikeln (Zeitschrift CL und Konferenz/Workshop-Proceedings)  
<http://www.aclweb.org/anthology/>

# Zusammenfassung

- Baumbanken: verschiedene Formate, Theorien, ...
- Evaluation
  - Parseval (labelled, unlabelled)
  - Leaf-Ancestor metric
  - Dependenz-basiert (LAS, UAS)

## References I

Abschnitte 11 und 12 aus Manning and Schütze (1999)

Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. L. Klavans, M. Liberman, M. P. Marcus, S. Roukos, B. Santorini, and T. Strzałkowski (1991).

A procedure for quantitatively comparing the syntactic coverage of english grammars.

In *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 306–311.

Manning, C. D. and H. Schütze (1999).

*Foundations of Statistical Natural Language Processing.*  
Cambridge, MA: The MIT Press.

## References II

Sampson, G. and A. Babarczy (2003).  
A test of the Leaf-Ancestor Metric for parse accuracy.  
*Natural Language Engineering* (9), 365–380.