Syntax natürlicher Sprachen

Tutorium

Grammatikinduktion und Annotationen

Sarah Anna Uffelmann

19.01.2024

Termine Tutorium

26.01.	IOB-Parsing + Wiederholung
02.02.	Wiederholung
09.02	Wiederholung
(13.02	*Klausur)*

Grammatikinduktion

Statt eigene Grammatiken zu schreiben, können wir Grammatikregeln (Produktionsregeln) aus Korpora extrahieren. Die relativen Häufigkeiten der Regeln (für PCFGs) werden dabei berücksichtigt. Die Gewichtung der Regeln ist bei induzierten Grammatiken besonders wichtig, da wir (je nach Korpusgröße) sehr viele Regeln induzieren und daher ein hohes Level an Ambiguität erhalten.

Grammar Induction in NLTK mit der Methode induce_pcfg( ):

```
productions = []
S = nltk.Nonterminal('S')
for tree in nltk.corpus.treebank.parsed_sents('wsj_0001.mrg'):
    productions += tree.productions()
```

```
grammar = nltk.induce_pcfg(S, productions)
```

Berechnung von Regelwahrscheinlichkeiten

Regelwahrscheinlichkeiten werden aus Regelhäufigkeiten berechnet.

Beispiel: In einer Treebank haben wir folgende Regelhäufigkeiten gezählt:

Regel	Häufigkeit
NP -> Det N	200
NP -> Pron	175
NP -> NP PP	125

Wie berechnen wir die Regelwahrscheinlichkeit P für die Regel NP -> Det N?

Zu berechnen: bedingt Wahrscheinlichkeit P (NP -> Det N | NP)

Berechnung von Regelwahrscheinlichkeiten

Regelwahrscheinlichkeiten werden aus Regelhäufigkeiten berechnet.

Beispiel: In einer Treebank haben wir folgende Regelhäufigkeiten gezählt:

Regel	Häufigkeit
NP -> Det N	200
NP -> Pron	175
NP -> NP PP	125

Wie berechnen wir die Regelwahrscheinlichkeiten für die Regel NP -> Det N?

Formel: $$P(\alpha \rightarrow \beta|\alpha) = \frac{count(\alpha \rightarrow \beta)}{\sum_{\gamma} count(\alpha \rightarrow \gamma)} = \frac{count(\alpha \rightarrow \beta)}{count(\alpha)}$$

P (Det N | NP) = 200 / (200 + 175 + 125) = 200 / 500 = 0,4
P (Pron | NP) = 175 / 500 = 0,35
P (NP PP | NP) = 125 / 500 = 0,25

Berechnung von Regelwahrscheinlichkeiten

Regelwahrscheinlichkeiten werden aus Regelhäufigkeiten berechnet.

Beispiel: In einer Treebank haben wir folgende Regelhäufigkeiten gezählt:

Regel	Häufigkeit	Wahrscheinlichkeit
NP -> Det N	200	0,4
NP -> Pron	175	0,35
NP -> NP PP	125	0,25

Formel: $$P(\alpha \rightarrow \beta|\alpha) = \frac{count(\alpha \rightarrow \beta)}{\sum_{\gamma} count(\alpha \rightarrow \gamma)} = \frac{count(\alpha \rightarrow \beta)}{count(\alpha)}$$

Chomsky-Normalform

Jede kontextfreie Grammatik lässt sich in die Chomsky-Normalform umformen.

Bei einer CFG in Chomsky-Normalform haben wir nur binäre und unäre Verzweigungen. Alle Produktionsregeln haben eine der folgenden beiden Formen:

A -> B C
A -> a

A, B und C: Nicht-Terminale
a: Terminal

Außerdem ist die Regel S -> ε zulässig, wobei S das Startsymbol und ε das leere Wort ist. Ist diese Regel Teil der Grammatik, darf S jedoch nicht auf der rechten Seite der Produktionsregeln stehen.

Warum relevant? Einige Parsing-Algorithmen setzen CNF voraus, z.B. CYK

Chomsky-Normalform

Wie bringen wir diese Regel in Chomsky-Normalform?

A -> b C D e

Chomsky-Normalform

Wie bringen wir diese Regel in Chomsky-Normalform?

A -> b C D e

1. Regeln für die Terminale einführen

B -> b
E -> e
A -> B C D E

Chomsky-Normalform

Wie bringen wir diese Regel in Chomsky-Normalform?

A -> b C D e

1. Regeln für die Terminale einführen

B -> b
E -> e
A -> B C D E

2. Regeln verkürzen durch das Einführen von Zwischenebenen

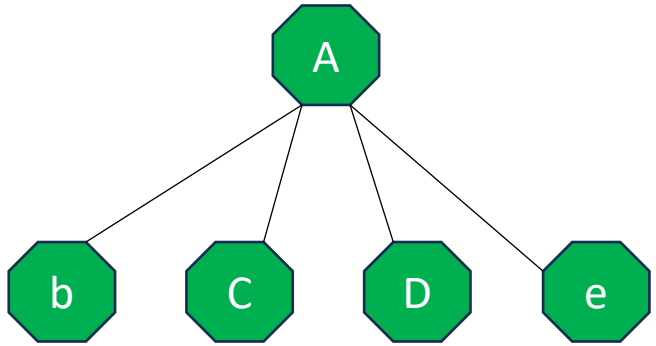
A –> B X1
X1 - > C X2
X2 -> D E

(Diese drei Regeln in Kombination leisten dasselbe wie die Regel A -> B C D E)

Chomsky-Normalform

Wie bringen wir diese Regel in Chomsky-Normalform?

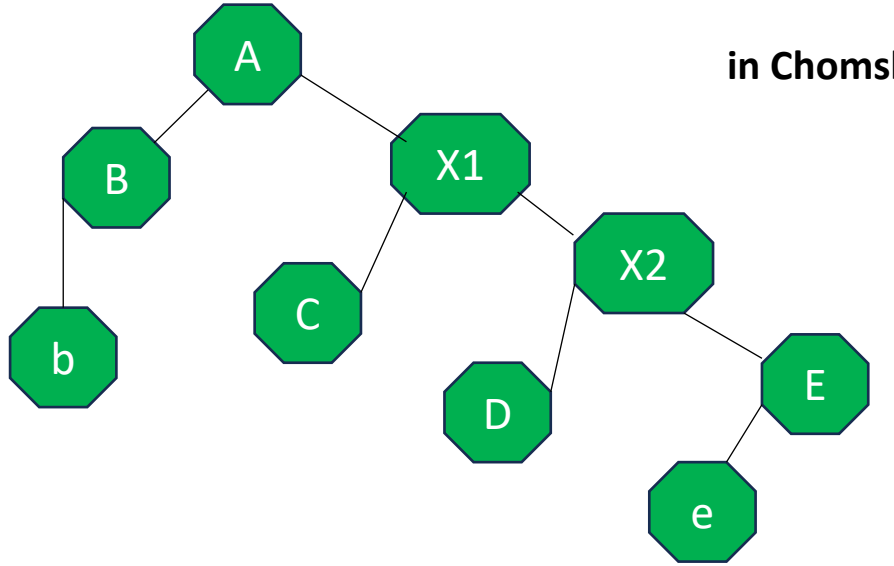
A -> b C D e



flache Struktur

Lösung:

B -> b
E -> e
A –> B X1
X1 - > C X2
X2 -> D E



in Chomsky-Normalform

Unabhängigkeitsannahmen bei PCFGs

(1) Die Wahrscheinlichkeit von Teilbäumen ist unabhängig von den Terminalen (Wörtern)

(2) Die Wahrscheinlichkeit von Teilbäumen ist unabhängig von den Elternknoten

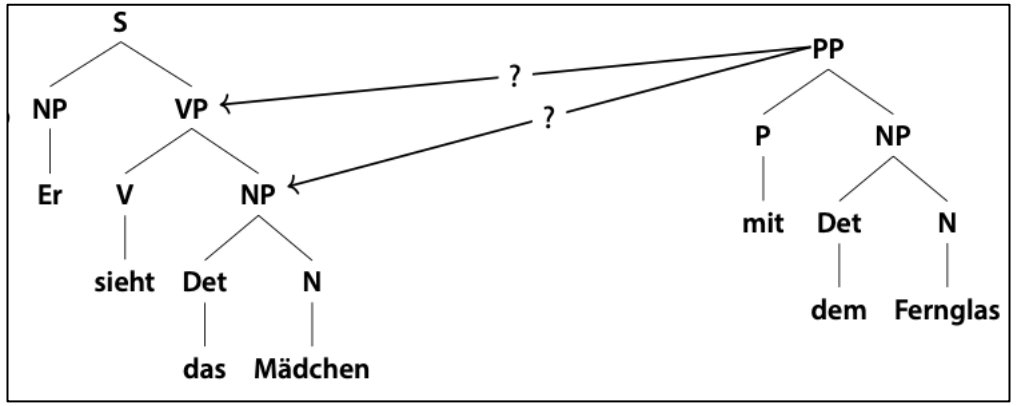
Um lexikalische und strukturelle Abhängigkeiten zu berücksichtigen und so beschreibungsadäquatere Syntaxmodelle zu erhalten, müssen wir diese Unabhängigkeitsannahmen zurücknehmen.

Dazu können wir bestimmte Annotationen verwenden:

- Lexikalisierte PCFGs – Kopfannotation
- history-based PCFGs – Parent Annotation

Lexikalische PCFGs

Eine nicht-lexikalisierte PCFG gibt bei ambigen Sätzen immer dieselbe (die wahrscheinlichere) Struktur zurück, unabhängig von den verwendeten Wörtern.



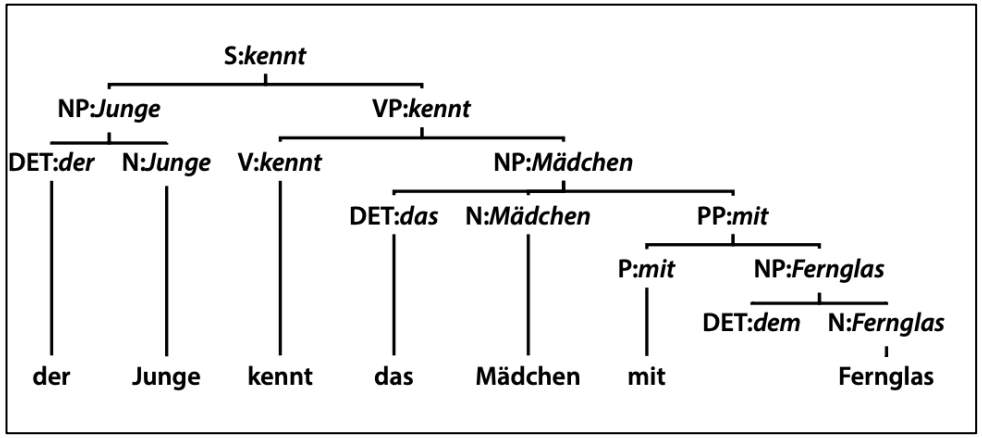
Bsp. PP-Attachment Ambiguität

Welche Struktur jedoch tatsächlich wahrscheinlichere (bzw. sinnvollere) ist, hängt vom Vokabular ab:

Er sieht das Mädchen mit dem Fernglas.	- Beide Lesarten sinnvoll
Er sieht das Huhn mit dem Fernglas.	- VP-Attachment bevorzugt
Er kennt das Mädchen mit dem Fernglas.	- NP-Attachment bevorzugt

Lexikalische PCFGs

Lösung: Kopfannotation



Phrasenköpfe werden hochgereicht (Kopf-Perkolation) und jedes Nicht-Terminal wird mit dem Phrasenkopf annotiert.

Einige Probleme:

- Regelvervielfachung: statt der allgemeinen Regel VP –> V NP haben wir jetzt die Regeln:
  VP(sieht) –> V(sieht) NP(Mädchen)
  VP(kennt) -> V(kennt) NP(Mädchen). usw. für das gesamte Vokabular
- umfangreiche Trainigsdaten notwendig
- Probleme bei ungesehenen Wörtern (sparse-data problem)

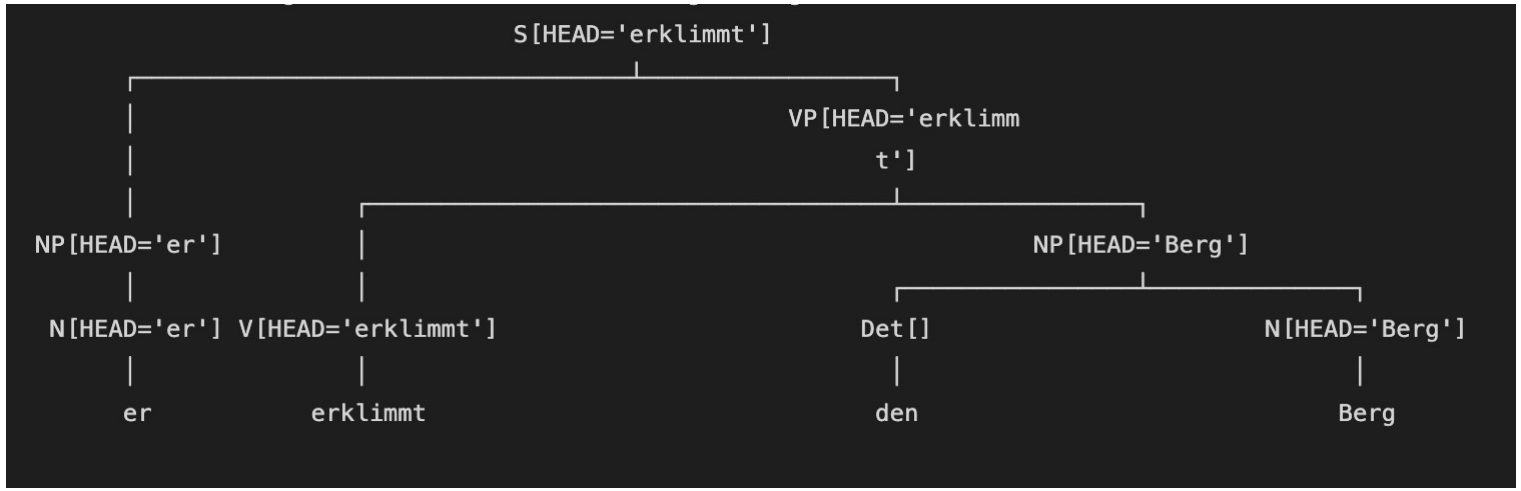
Lexikalische PCFGs

Beispiel einer Kopfannotation mit Hilfe eines HEAD-Features in einer FCFG

```
sentence = "er erklimmt den Berg"

gramstring = r"""
% start S
    S[HEAD=?v]    -> NP[] VP[HEAD=?v]
    VP[HEAD=?v]   -> V[HEAD=?v] NP[]
    NP[HEAD=?n]   -> Det[] N[HEAD=?n]
    NP[HEAD=?n]   -> N[HEAD=?n]

    Det[] -> "den"
    N[HEAD="er"]    -> "er"
    N[HEAD="Berg"]    -> "Berg"
    V[HEAD="erklimmt"]    -> "erklimmt"
"""
```



History-based PCFGs

Eine herkömmliche PCFG berücksichtigt nicht die Position einer Regelanwendung im Parsebaum, d.h. der Kontext wird in die Berechnung einer Regelwahrscheinlichkeit nicht mit einbezogen.
Oft sind die Regelwahrscheinlichkeiten jedoch abhängig von den zuvor angewandten Regeln.

Bsp.:
Die Wahrscheinlichkeit der Regel NP -> Pron
ist höher bei Subjekt-NPs (wenn die zuvor angewandte Regel S –> NP VP war)
als bei Objekt-NPs (wenn die zuvor angewandte Regel VP -> V NP war).

History-based PCFGs

erwünschte Regelgewichtung Subjekt (S-dominiert):

NP → PRON **0.91**
NP → DET N 0.09

erwünschte Regelgewichtung Objekt (VP-dominiert):

NP → PRON 0.34
NP → DET N **0.66**

normale PCFG (keine Differenzierung, Daten aus Korpus):

NP → PRON **0.25**
NP → DET N **0.28**

Lösung: Splitting NP-Kategoriensymbol (*parent annotation*):

NP^S → PRON 0.91
NP^S → DET N 0.09
NP^VP → PRON 0.34
NP^VP → DET N 0.66

Lösung: Parent Annotation

- nicht-terminale Knoten werden mit der Kategorie des Elternknotens (= history) annotiert
- als Trennzeichen verwenden wir das Zeichen ^ (z.B. NP^S)
- Nicht-Terminale werden dadurch in mehrere Kategorien aufgespalten

Probleme:

- Regelvervielfachung
- Probleme bei unbekannter Vorgängerkategorie

History-based PCFGs

Beispiel einer Parent Annotation

```
sentence = "er erklimmt den Berg"

grammar = nltk.CFG.fromstring("""
    S    -> NP^S VP^S
    VP^S  -> V^VP NP^VP
    NP^VP  -> Det^NP N^NP
    NP^S  -> N^NP

    Det^NP -> "den"
    N^NP   -> "er"
    N^NP   -> "Berg"
    V^VP   -> "erklimmt"
```