Syntax natürlicher Sprachen

Tutorium 10:

Probabilistic Context Free Grammar

Shuyan Liu

10.01.2025

Einige Beispiele kommt aus der Vorlesungsfolien, Aufgaben sowie Übungen.

Die Hauptteile der Slides dieser Woche stammen von Sarah Anna Uffelmann

aus Wintersemester 2023/24 und wurden bearbeitet. Verwendung mit Dank.

Was ist Probabilistic Context Free Grammar?

- Context-Free-Grammar + Probability
- Ziel:
  - korrekte, plausible oder wahrscheinlichere Struktur-Interpretationen vorzuschlagen.
  - Syntax-Disambiguierung
    - **hohe Abdeckung** (viele Regeln, großes Lexikon mit ambigen Einträgen) **und Input langer (komplexer) Sätze** führen zu:
      → ***hoher Aufwand beim Parsing***
      → ***große Anzahl an Ableitungen/Analysen (Ambiguität)***
    - z. B. durch Ambiguität im Lexikon:

      [NP Time] [V flies] like an arrow.

      [V Time] [NP flies] like an arrow.

      [NP Time flies] [V like] an arrow.

Was ist Probabilistic Context Free Grammar?

- Context-Free-Grammar + Probability
- Ziel:
  - korrekte, plausible oder wahrscheinlichere Struktur-Interpretationen vorzuschlagen.
  - Syntax-Disambiguierung
- Liefert die möglichste Parsebäume (basiert auf Korpusdaten) zurück
- Regeln:
  - Die Wahrscheinlichkeiten aller Regeln mit derselben linken Seite summieren zu 1 (d.h. sie ergeben eine Wahrscheinlichkeitsverteilung)
  - Die Wahrscheinlichkeit eines Parsebaumes ist das Produkt seiner Regelwahrscheinlichkeiten

Wahrscheinlichkeit-Regel 1

1. S → NP VP
2. NP → DET N
3. VP → V NP
4. VP → V
5. DET → "der" | "die" | "das" | "ein" | "eine"
6. N → "Hund" | "Katze" | "Maus" | "Apfel"
7. V → "frisst" | "schläft" | "läuft" | "spielt"

WK: 70% für VP -> V NP

Wahrscheinlichkeit für VP -> V?

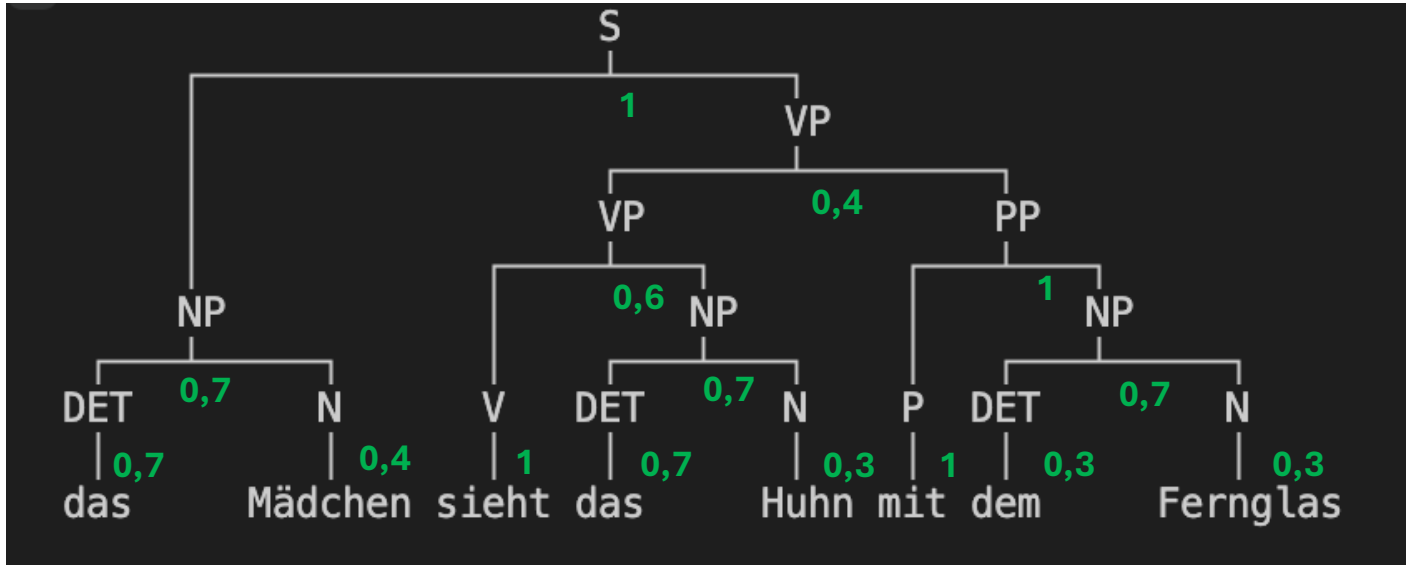
Antwort: 30%
denn Wahrscheinlichkeiten aller Regeln für die Expansion eines bestimmten Nonterminals addieren sich zu 1

Wahrscheinlichkeit-Regeln 2

„das Mädchen sieht das Huhn mit dem Fernglas.“

S	-> NP VP	(1)
NP	-> Det N	(0,7)
NP	-> NP PP	(0,3)
VP	-> V NP	(0,6)
VP	-> VP PP	(0,4)
PP	-> P NP	(1)
Det	-> das	(0,7)
Det	-> dem	(0,3)
N	-> Mädchen	(0,4)
N	-> Huhn	(0,3)
N	-> Fernglas	(0,3)
V	-> sieht	(1)
P	-> mit	(1)

Lesart 1:



Die Wahrscheinlichkeit eines Parsebaumes ist das Produkt seiner Regelwahrscheinlichkeiten:

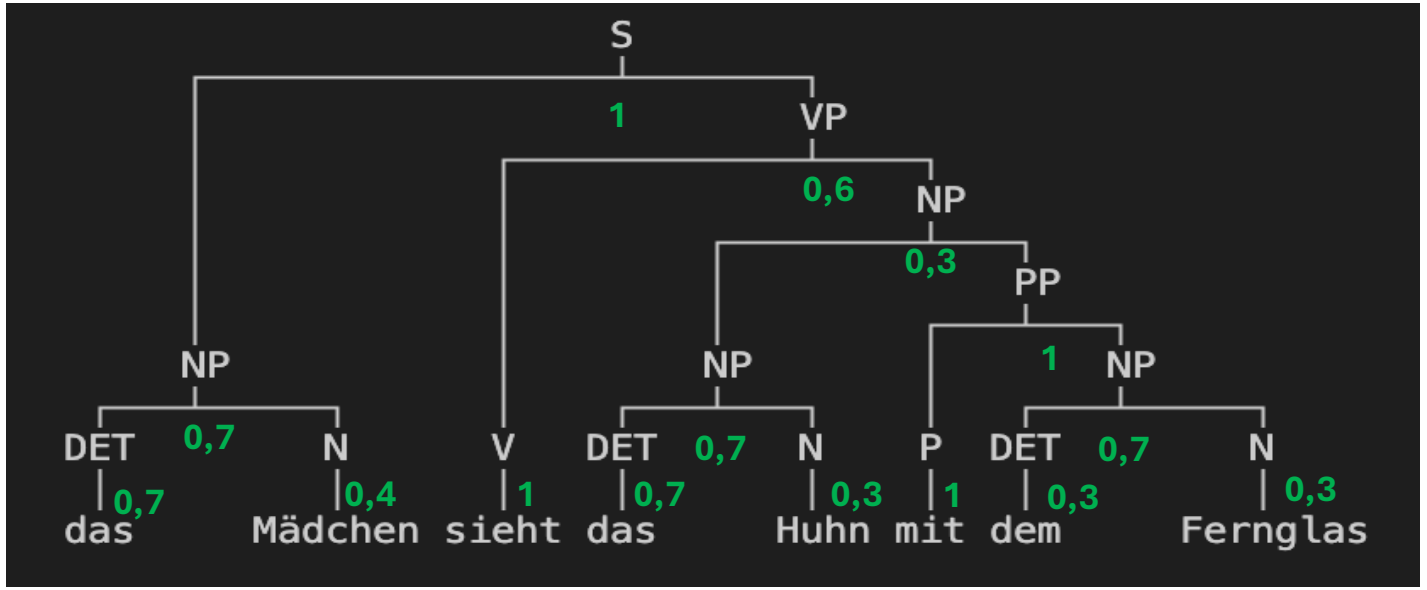
P (T1) = 1 * 0,7 * 0,7 * 0,4 * 0,4 * 0,6 * 1 * 0,7 * 0,7 * 0,3 * 1 * 1 * 0,7 * 0,3 * 0,3
= 0,0004356374

Wahrscheinlichkeit-Regeln 2

„das Mädchen sieht das Huhn mit dem Fernglas.“

S	-> NP VP	(1)
NP	-> Det N	(0,7)
NP	-> NP PP	(0,3)
VP	-> V NP	(0,6)
VP	-> VP PP	(0,4)
PP	-> P NP	(1)
Det	-> das	(0,7)
Det	-> dem	(0,3)
N	-> Mädchen	(0,4)
N	-> Huhn	(0,3)
N	-> Fernglas	(0,3)
V	-> sieht	(1)
P	-> mit	(1)

Lesart 2:



Die Wahrscheinlichkeit eines Parsebaumes ist das Produkt seiner Regelwahrscheinlichkeiten:

P (T2) = 1 * 0,7 * 0,7 * 0,4 * 0,6 * 1 * 0,3 * 0,7 * 0,7 * 0,3 * 1 * 1 * 0.7 * 0,3 * 0,3
= 0,0003267281

PCFG-Regeln

P (T1) = 0,0004356374
P (T2) = 0,0003267281

-> **T1 ist der wahrscheinlichere Parsebaum**

Die **Wahrscheinlichkeit des Satzes** ist die Summe der Wahrscheinlichkeiten aller seiner möglichen Ableitungen (Parsebäume).

In unserem Beispiel: P (S) = P (T1) + P (T2) = 0,0007623655

Woher erhalten wir die Regelwahrscheinlichkeiten?
-> Sie werden aus Regelhäufigkeiten geschätzt.

Regelhäufigkeiten erhalten wir
(1) durch Zählen der Regeln in einer Treebank
(2) aus einem automatisch geparsten und disambiguierten Korpus

PCFG Parsing

Viterbi Parser

-> Gibt den **wahrscheinlichsten** Parsebaum (und **nur** diesen!) zurück

Probalistische Chart-Parser

- probabilistische Varianten von Chart-Parsing-Algorithmen (Earley, CYK)
- NLTK: InsideChartParser, LongestChartParser
- haben Zustandsmengen (**edge queue**), die nach unterschiedlichen Kriterien sortiert werden
- können:
- **lowest cost first** (bei InsideChartParser)
  Sortierung nach Wahrscheinlichkeit, findet also immer die wahrscheinlichste Ableitung
- **beam search** (bei Inside Chart Parser beam_size definieren)
  wie lowest cost first, aber es werden nur die n (beam size) wahrscheinlichsten Ableitungen behalten
- **best first search** (bei LongestChartParser)
  Sortierung nach Länge der Ableitung (hat zur Folge, dass evtl. nicht die wahrscheinlichste Ableitung an erster Stelle steht)