

Syntax natürlicher Sprachen

Vorlesung WS 2020/2021

Centrum für Informations- und Sprachverarbeitung

LMU München

Axel Wisiolek, Martin Schmitt

0 Organisatorisches

Vorlesung: Di, 10-12 Uhr (Axel Wisioerek), per Zoom

Email: `wisioerek at cis.lmu.de`

Übung: Mi, 14-16 Uhr (Martin Schmitt), per Zoom

Email: `martin at cis.lmu.de`

Tutorium: *Termin folgt* (Swantje Kastrup), per Zoom

Zoom-Links in Moodle!

Kursseite: <https://github.com/awisioerek/syntax-2021>

Moodle: <https://moodle.lmu.de/course/view.php?id=11885>

Programmsystem:

- NLTK (python3)
- Jupyter-Notebooks
- spaCy / Stanford-Parser

Materialien (github-Kursseite):

- **Vorlesungsfolien**
- interaktive Übungsblätter (**Jupyter-Notebooks**)
- Übungen auch als **PDF-Version**

Anmeldung: Moodle

- *(Hauptfach-spezifischer Einschreibeschlüssel per LSF-Rundmail)*
- nicht-öffentliche Kurs-Informationen: **Zoom-Links** etc.
- **Kommunikationsplattform** (Foren, Nachrichten an Dozenten)
- Bereitstellung **PDF-Lösungsblätter**
- **Verwaltung Informatik-Studierende**

Prüfungsanmeldung:

- **CL-Hauptfach:** Januar 2021 im LSF
- **Informatik-Hauptfach:** Anmeldung über Moodle

Klausurtermin: *folgt*

Begleitende Literatur / Lehrbuch / Grundlage Übungen:

- **NLTK-Book** = Bird, Steven & Klein, Ewan & Loper, Edward (2009): *Natural Language Processing with Python*:
→ **HTML-Version:** <http://www.nltk.org/book>
- **Dürscheid, Christa (2010):** *Syntax: Grundlagen Und Theorien*.
 - inkl. Glossar und Übungen
 - als Ebook verfügbar über UB-E-Medien-Login: <https://login.emedien.ub.uni-muenchen.de/login?url=http://www.utb-studi-e-book.de/9783838533193>

NLTK-Book, Kapitel Syntaxanalyse:

- <http://www.nltk.org/book/ch08.html>
→ Analyzing Sentence Structure
- <http://www.nltk.org/book/ch09.html>
→ Feature Based Grammars, Statistical Parsing
- <http://www.nltk.org/book/ch07.html>
→ Shallow Parsing / Parsing as Tagging
- <http://www.nltk.org/book/ch08-extras.html>
→ Chunking vs Parsing, PCFG-Parsing

Weiterführende Literatur:

SLP2 = Jurafsky, Dan & Martin, James H. (2009): *Speech and Language Processing. 2. Ausgabe*: Kapitel 12-16, <http://www.cs.colorado.edu/~martin/slp2.html>

SLP3 = Jurafsky, Dan & Martin, James H. (2018): *Speech and Language Processing. 3. Ausgabe*: Kapitel 10-13

- online verfügbar: <https://web.stanford.edu/~jurafsky/slp3/>
<https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>

Carstensen, Kai-Uwe, Hrsg. (2010): *Computerlinguistik und Sprachtechnologie*. Kapitel 2.2,2.3,3.4 und 3.5

- als Ebook verfügbar über UB-E-Medien-Login: <https://login.emedien.ub.uni-muenchen.de/login/up?qurl=http://link.springer.com%2f10.1007%2f978-3-8274-2224-8>

Manning, Christopher D. & Schütze (1999): *Foundations of Statistical Natural Language Processing*.

Van Valin, Robert D. (2001): *An Introduction to Syntax*

- als Ebook über BSB: <https://opacplus.bsb-muenchen.de/title/BV043923920>

Glossare:

- linguistisch:
 - Dürscheid: 229 ff. (s. Moodle)
 - <http://www.mediensprache.net/de/basix/lexikon>
 - <http://www.glossary.sil.org>
- computerlinguistisch:
 - http://www.nltk.org/book/term_index.html

0.1 Inhalte und Lernziele

Inhalte Vorlesung (siehe Modulhandbuch):

- Die Vorlesung behandelt **Grundbegriffe der Grammatik** (wie Kongruenz, Rektion, Subkategorisierung und Valenz) und wesentliche syntaktische Konstruktionen des Deutschen im Hinblick auf eine **Verwendung in der maschinellen Sprachverarbeitung**
- Dazu werden die in neueren Grammatiktheorien verwendeten **Klassifizierungen von Phrasen**, deren **innere Struktur** sowie deren **relationale Abhängigkeiten** erklärt.

- Außerdem werden in der Computerlinguistik übliche **Grammatikformalismen** (wie Kontextfreie Grammatiken, Unifikationsgrammatiken, PCFGs, Datenbasierte Abhängenzgrammatiken, Partielle Parsingmodelle)
- ebenso wie **syntaktische Annotationsstandards** (z.B. Penn Treebank, Universal Dependencies) vorgestellt und verwendet, um typische oder schwierige syntaktische Konstruktionen genau zu beschreiben.

Tagsets:

- <http://universaldependencies.org/u/dep/all.html>
- <https://www.clips.uantwerpen.be/pages/mb-sp-tags>
- https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Übung:

- **Aufgaben:** (*Präsenz- und Hausaufgaben* im Notebook)
 - linguistische Aufgaben (interaktiv aufbereitet)
 - computerlinguistische Aufgaben im Programmsystem (NLTK)
- **Wiederholungsfragen:**
 - zu NLTK-Kapiteln (*Hausaufgaben*)
- Weitere Details morgen in der ersten Sitzung der Übung

Hinweis Übung/Tutorium:

- → **Installation NLTK, Jupyter etc.**
- bis dahin: Übungsblatt 1 herunterladen → **Installationsanleitungen**
- wenn möglich bereits vorab Software installieren (NLTK, Jupyter, spaCy)
- NLTK und Jupyter lassen sich am einfachsten mit **Anaconda** installieren (python-Distribution)
- *Tutorium erste Woche: nur Installation!*

0.2 Themen

Themenübersicht

- I Linguistische und formale Grundlagen**
- II Parsing - CFG-Parsingalgorithmen und Unifikationsparsing**
- III Statistisches Parsing - PCFGs und Dependency Parsing**
- IV Partielles Parsing**

Sitzungen

- 0 Organisatorisches
- 1 Syntaxanalyse mit NLTK

I Linguistische und formale Grundlagen

- 2 Einführung
- 3 Syntaktische Kategorien
- 4 Syntaktische Relationen: Konstituenz
- 5 Syntaktische Relationen: Dependenz
- 6 Morphologische Form syntaktischer Funktionen
- 7 Unifikationsgrammatiken
- 8 Komplexe Satzkonstruktionen und Wortstellung

II Parsing - CFG-Parsingalgorithmen und Unifikationsparsing

9 Parsing-Algorithmen

10 Getypte Merkmalstrukturen und Unifikation

III Statistisches Parsing - PCFGs und Dependency Parsing

11 Statistische Syntaxmodelle

12 Datengestützte Syntaxmodelle

IV Partielles Parsing

13 Partielles Parsing; Komplexität natürlicher Sprachen

1 Syntaxanalyse mit NLTK

1.1 Natural Language Toolkit

- Bündel von Python-Bibliotheken und Programmen für computerlinguistische Anwendungen
- quelloffen, für Lehre entwickelt
- Lehrbuch: <http://www.nltk.org/book>
- Dokumentation: <http://www.nltk.org/howto>
- Daten (Korpora, Grammatiken): <http://www.nltk.org/data.html>
- Interfaces, z.B. für *Stanford Parser*: <http://nlp.stanford.edu:8080/parser/>; <http://nlp.stanford.edu:8080/corenlp/process/>

1.2 Parsing mit NLTK

- Automatische Syntaxanalyse (Parsing):
 - Überprüfung der grammatischen Struktur einer Eingabe als **Suche einer Ableitung** aus den Regeln einer formalen Grammatik
 - Wiedergabe der grammatischen Struktur bei Wohlgeformtheit als **Ableitungsbaum** (auch: Parsebaum, Syntaxbaum)

Beispiele siehe:

- **NLTK-08**

→ Parsing mit CFGs, Dependenzgrammatiken, PCFGs

- **NLTK-07**

→ partielles Parsing mit RegexpParser

- **NLTK-09**

→ Parsing mit feature-based grammars

Eingabe:

- *One morning I shot an elephant in my pajamas.*

How he got into my pajamas I don't know.

(Groucho Marx, Animal Crackers, 1930)

Auflistung 1: Import NLTK, Einlesen Eingabe

```
1 import nltk
2
3 sent = 'I shot an elephant in my
    pajamas'.split()
4 print(sent)
5 #['I', 'shot', 'an', 'elephant', 'in', 'my',
    'pajamas']
```

1.2.1 CFG-Parsing

- **kontextfreie Grammatik**
- Analyse des **Aufbaus syntaktischer Einheiten**
- **Konstituentenstruktur**: Strukturinformationen in **Knoten** des Syntaxbaums
- *Problem 1: Auflösung struktureller Ambiguität*
- *Problem 2: Übergenerierung* (Ableitung ungrammatischer Sätze)

Auflistung 2: Konstituentengrammatik / Phrasenstrukturgrammatik / CFG (Kontextfreie Grammatik)

```
1 grammar = nltk.CFG.fromstring("""
2     S → NP VP
3     PP → P NP
4     NP → Det N | Det N PP | 'I'
5     VP → V NP | VP PP
6     Det → 'an' | 'my'
7     N → 'elephant' | 'pajamas'
8     V → 'shot'
9     P → 'in'
10    """)
11
12 parser = nltk.ChartParser(grammar, trace=0)
13
14 for tree in parser.parse(sent):
15     tree.draw()
```

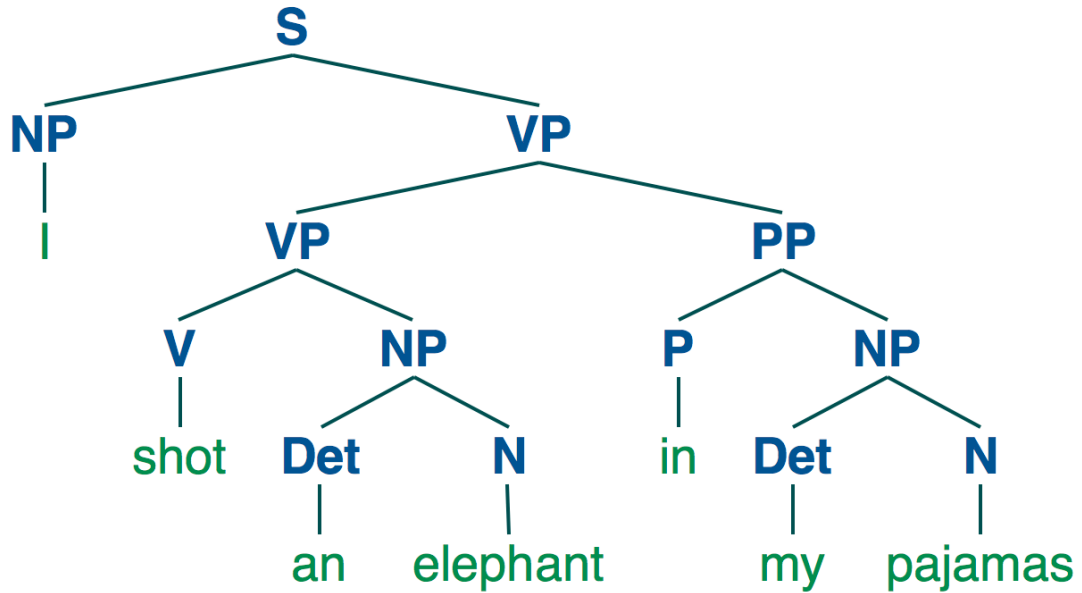


Abbildung 1: Syntaxbaum Konstituentenanalyse

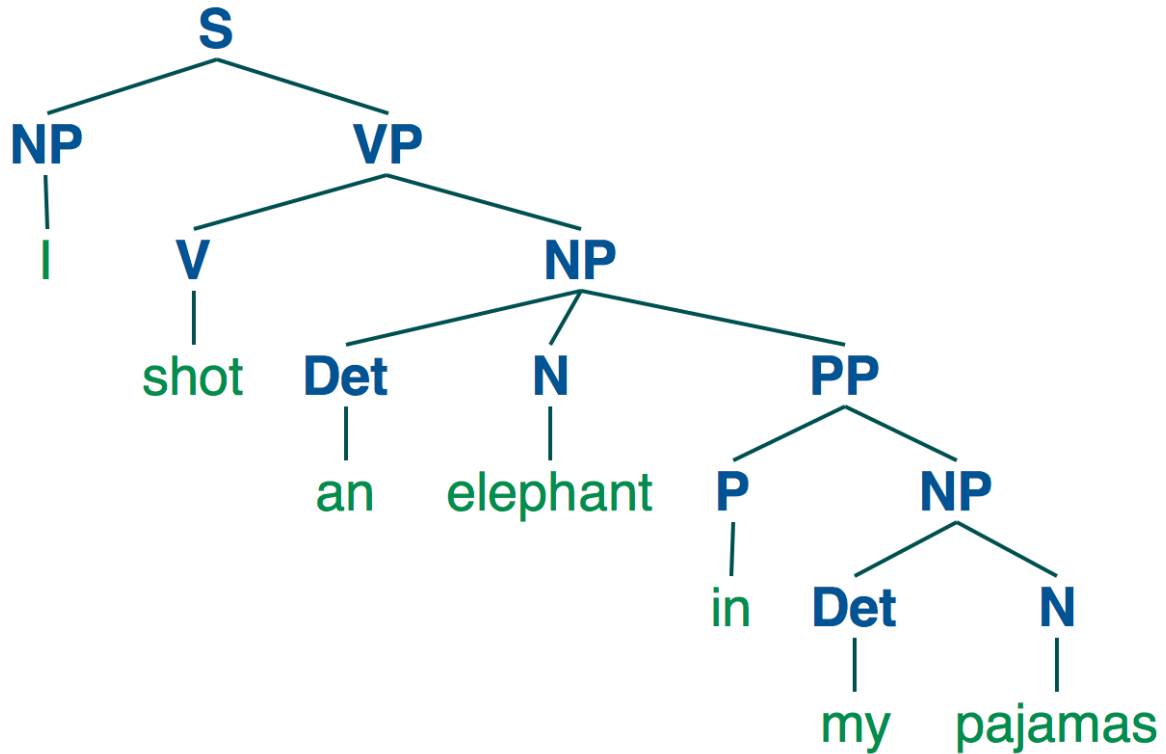


Abbildung 2: Syntaxbaum Konstituentenanalyse 2

1.2.2 Dependenz-Parsing

- **Dependenzgrammatik**
- Analyse der **Abhängigkeitsrelationen** zwischen **Wörtern**
- **Dependenzstruktur**: Strukturinformationen in **Kanten** des Syntaxbaums

Auflistung 3: *Dependenzgrammatik*

```
1 grammar = nltk.DependencyGrammar.fromstring("""
2     'shot' → 'I' | 'elephant' | 'in'
3     'elephant' → 'an' | 'in'
4     'in' → 'pajamas'
5     'pajamas' → 'my'
6     """)
7
8 parser =
9     nltk.ProjectiveDependencyParser(grammar)
10
11 for tree in parser.parse(sent):
12     print(tree)
13     tree.draw()
14
15 #(shot I (elephant an (in (pajamas my))))
16 #(shot I (elephant an) (in (pajamas my)))
```

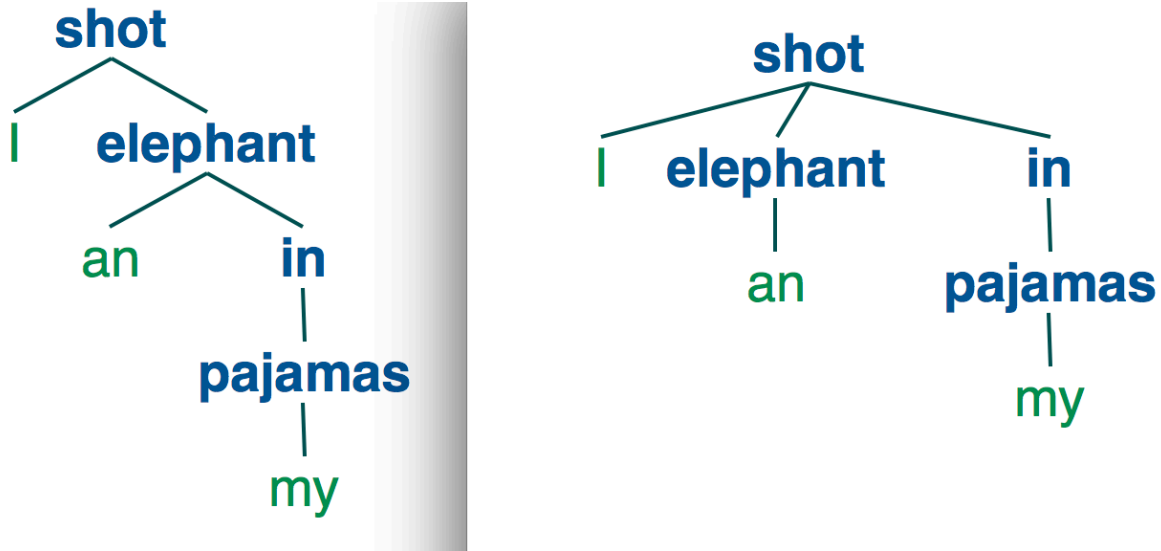


Abbildung 3: Syntaxbäume *Dependenzanalyse (Stemmas)*

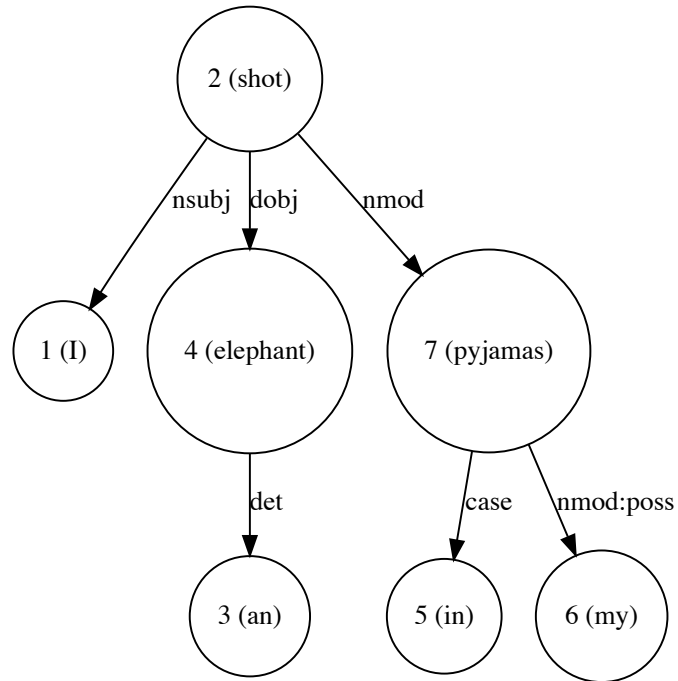


Abbildung 4: Ausgabe Stanford Dependency Parser

1.2.3 PCFG-Parsing

- kontextfreie Grammatik mit **gewichteten Regeln**
- Wahrscheinlichkeiten werden **aus syntaktisch annotiertem Korpus gelernt**
- löst *strukturelle Ambiguität* auf durch Berechnung der **wahrscheinlichsten Ableitung**
- löst Problem der *Übergenerierung* durch Ausschluss ungrammatische Sätze als **unwahrscheinlich**

Auflistung 4: *Probabilistische kontextfreie Grammatik (PCFG)*

```
1 grammar1 = nltk.PCFG.fromstring("""
2     S → NP VP [1.0]
3     PP → P NP [1.0]
4     NP → Det N [0.8] | Det N PP [0.1] | 'I'
        [0.1]
5     VP → V NP [0.8] | VP PP [0.2]
6     Det → 'an' [0.7] | 'my' [0.3]
7     N → 'elephant' [0.5] | 'pajamas' [0.5]
8     V → 'shot' [1.0]
9     P → 'in' [1.0]
10    """)
11
12
13
14
```

```
15 parser = nltk.ViterbiParser(grammar1)
16
17 for tree in parser.parse(sent):
18     print(tree)
19
20 #(S
21 #  (NP I)
22 #  (VP
23 #    (VP (V shot) (NP (Det an) (N elephant)))
24 #    (PP (P in) (NP (Det my) (N pajamas))))))
    (p=0.0005376)
25
26
27
28
29
```

```
30 grammar2 = nltk.PCFG.fromstring("""
31     S → NP VP [1.0]
32     PP → P NP [1.0]
33     NP → Det N [0.7] | Det N PP [0.2] | 'I'
        [0.1]
34     VP → V NP [0.8] | VP PP [0.2]
35     Det → 'an' [0.7] | 'my' [0.3]
36     N → 'elephant' [0.5] | 'pajamas' [0.5]
37     V → 'shot' [1.0]
38     P → 'in' [1.0]
39     """)
40
41
42
43
44
```

```
45 parser = nltk.ViterbiParser(grammar2)
46 for tree in parser.parse(sent):
47     print(tree)
48 #(S
49 #  (NP I)
50 #  (VP
51 #    (V shot)
52 #    (NP
53 #      (Det an)
54 #      (N elephant)
55 #      (PP (P in) (NP (Det my) (N
    pajamas)))))) (p=0.000588)
```


1.2.4 feature-based-Parsing

- Berücksichtigung von **morphologischen *Constraints***:
 - Kasus
 - Kongruenz/Agreement
 - Subkategorisierung (Anzahl und Art von Argumenten)
- verhindert *Übergenerierung* durch linguistisch adäquate Modellierung

Auflistung 5: *feature-based grammar (Ausschnitt)*

```
1  ## Natural Language Toolkit: german.fcfg
2  % start S
3
4  #####
5  # Grammar Productions
6  #####
7  S → NP[CASE=nom, AGR=?a] VP[AGR=?a]
8
9  NP[CASE=?c, AGR=?a] → PRO[CASE=?c, AGR=?a]
10 NP[CASE=?c, AGR=?a] → Det[CASE=?c, AGR=?a]
    N[CASE=?c, AGR=?a]
11
12 VP[AGR=?a] → IV[AGR=?a]
13 VP[AGR=?a] → TV[OBJCASE=?c, AGR=?a]
    NP[CASE=?c]
```

```
14 #####
15 # Lexical Productions
16 #####
17 # Singular determiners
18 # masc
19 Det [CASE=nom , AGR=[GND=masc , PER=3 , NUM=sg]] →
    'der '
20 Det [CASE=dat , AGR=[GND=masc , PER=3 , NUM=sg]] →
    'dem '
21 Det [CASE=acc , AGR=[GND=masc , PER=3 , NUM=sg]] →
    'den '
22 # fem
23 Det [CASE=nom , AGR=[GND=fem , PER=3 , NUM=sg]] →
    'die '
24
25 ##usw...
```

Auflistung 6: feature-based parsing

```
1 sent = 'der Hund sieht uns'.split()
2
3 grammar =
    nltk.grammar.FeatureGrammar.fromstring(gramstring)
4 parser = nltk.parse.FeatureChartParser(grammar)
5
6 for tree in parser.parse(sent):
7     tree.draw()
```

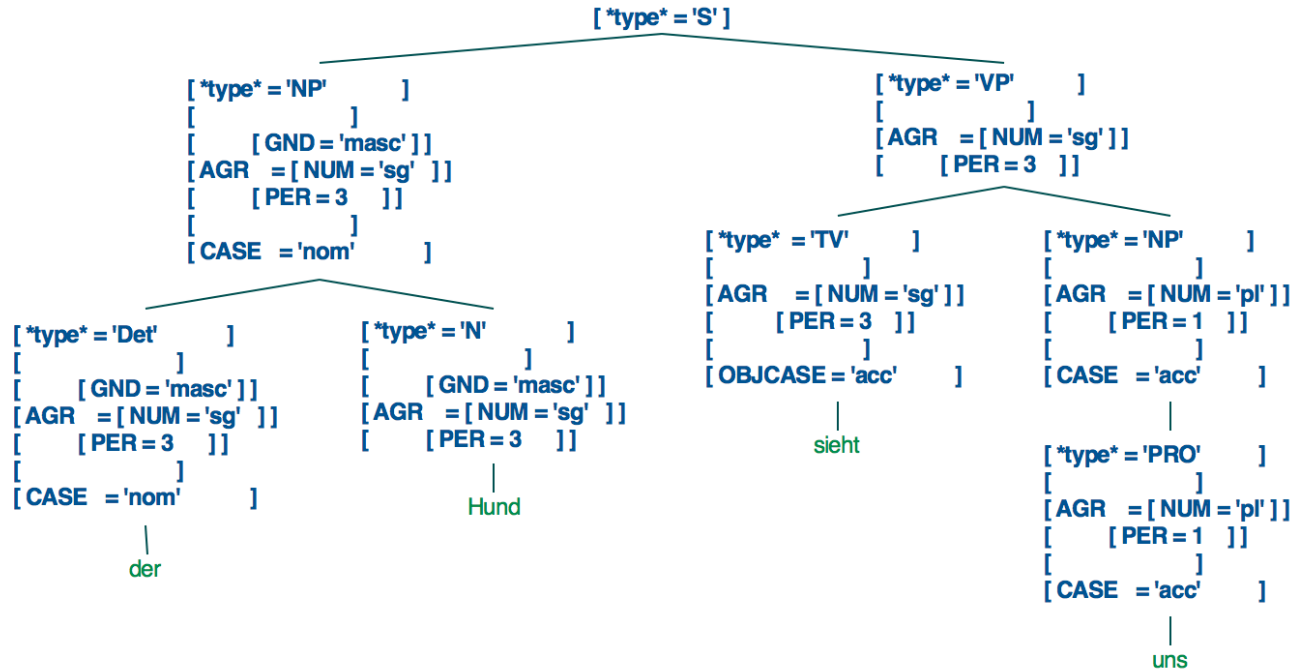


Abbildung 5: Syntaxbaum feature-based

1.2.5 Partielles Parsing mit RegexpParser

- **flache**, nicht-hierarchische Analyse
- **partielle** Analyse: nur wichtigste Konstituenten
- für Anwendungen wie Informationsextraktion oder *information retrieval*:
 - **keine syntaktische Vollanalyse notwendig**
 - einfaches Syntaxmodell ausreichend (reguläre Grammatik)

Auflistung 7: *Partielles Parsing (reguläre Grammatik)*

```
1 grammar = r"""
2     NP: {<DT|PP\$>?<JJ>*<NN>}
3 # chunk determiner/possessive, adjectives and
4     noun
5     {<NNP>+}
6 # chunk sequences of proper nouns
7     """
8 parser = nltk.RegexpParser(grammar)
9 sent = [("Rapunzel", "NNP"), ("let", "VBD"),
10        ("down", "RP"), ("her", "PP$"), ("long",
11        "JJ"), ("golden", "JJ"), ("hair", "NN")]
12 tree = parser.parse(sent)
13 tree.draw()
```

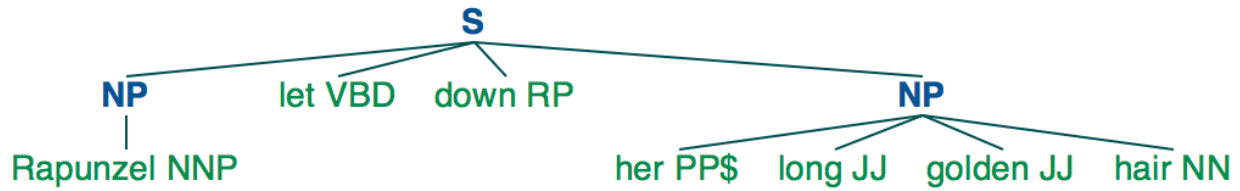


Abbildung 6: Syntaxbaum NP-Chunking-Analyse