Syntax natürlicher Sprachen

Vorlesung WS 2018/2019

Centrum für Informations- und Sprachverarbeitung

LMU München

Axel Wisiorek

0 Organisatorisches

0.1 Übersicht

Vorlesung: Di, 10-12 Uhr, Raum L155 (Axel Wisiorek)

Email: `wisiorek at cis.lmu.de`

Übung: Mi, 14-16 Uhr, Raum L155 (Martin Schmitt)

Email: `martin at cis.lmu.de`

Tutorium: Fr 17-18:30 s.t., Raum L155 (Swantje Kastrup)

Kursseite: `https://github.com/awisiorek/syntax-1819`

Programmsystem:

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

Materialien:

- **Vorlesungsfolien** und **Übungsblätter** (PDF) und auf Kursseite
- dort auch **interaktive Jupyter-Notebooks**

Anmeldung: LSF (ggf. Matrikelnummer mailen zum Nachmelden)

Klausur: Mi, 06.02.2019, 14-16 Uhr (Raum L155)

Prüfungsanmeldung: Januar 2019 im LSF; gemeinsame Modulprüfung Vorlesung + Übung

Informatik-Hauptfach: E-Mail und Matrikelnummer mailen (LSF-Anmeldung sowie Klausur-Anmeldung, Info folgt)

0.2 Literatur

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`

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:

Dür = Dürscheid, Christa (2010): *Syntax: Grundlagen Und Theorien*. (inkl. Glossar und Übungen; als Ebook verfügbar über OPAC UB)

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`

CuS = Carstensen, Kai-Uwe, Hrsg. (2010): *Computerlinguistik und Sprachtechnologie*.
Kapitel 2.2,2.3,3.4 und 3.5
als Ebook verfügbar über OPAC UB

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

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

Glossare:

- linguistisch:
  - Dürscheid:229ff.
  - http://www.mediensprache.net/de/basix/lexikon
  - http://www.glossary.sil.org
- computerlinguistisch:
  - http://www.nltk.org/book/term_index.html

Tagsets:

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

0.3 Inhalte und Lernziele

Inhalte Vorlesung:

- 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 Dependenzgrammatiken, Partielle Parsingmodelle)
- ebenso wie **syntaktische Annotationsstandards** (z.B. Penn Treebank, Universal Dependencies) vorgestellt und verwendet, um typische oder schwierige syntaktische Konstruktionen genau zu beschreiben.

Inhalte Übung:

- Die in der Vorlesung erklärten grammatikalischen Begriffsbildungen werden an Beispielen konkretisiert und vertieft.
- Ein **Programmsystem zu den in der Vorlesung verwendeten Formalismen wird auf Beispiele angewendet**.

Qualifikationsziele:

- Die Studierenden kennen **funktionale und strukturelle Begriffe der grammatischen Beschreibung** und können sie anwenden.
- Sie kennen einen **Grammatikformalismus** und können darin Analysen natürlichsprachlicher Sätze ausdrücken und Begriffe der Grammatik genau anwenden.
- Sie kennen ein **Programmsystem**, das einen Grammatik-Formalismus verwendet und können es bedienen.

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

0.4 Hinweise

- Morgen (Übung) Laptop mitbringen! (Mi, 17.10: 14 Uhr, L155) $\rightarrow$ **Installation NLTK, Jupyter etc.**
- bis dahin: Übungsblatt 1 herunterladen $\rightarrow$ **Installationsanleitungen**
- vorab: Software installieren (NLTK, Jupyter, Stanford-Parser), zumindest schon **herunterladen**
- NLTK und Jupyter lassen sich am einfachste mit `Anaconda` installieren (python-Distribution)

0.5 Themen

Themenübersicht

Themenübersicht

I Syntax - Linguistische und formale Grundlagen

II Parsing - CFG-Parsingalgorithmen und Unifikationsparsing

III Statistisches Parsing - PCFGs und Dependency Parsing

IV Partielles Parsing - Chunking und reguläre Grammatiken

Themenübersicht

Sitzungen

Sitzungen

Sitzungen

8 Komplexe Satzkonstruktionen und Wortstellung

II Parsing - CFG-Parsingalgorithmen und Unifikationsparsing

9 CFG-Parsing

10 Unifikation

III Statistisches Parsing - PCFGs und Dependency Parsing

11 Probabilistische kontextfreie Grammatiken

12 Statistische Syntaxmodelle

Sitzungen

IV Partielles Parsing - Chunking und reguläre Grammatiken

13 Partielles Parsing - Komplexität

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

- Parsing: automatische Syntaxanalyse
- 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**

  $\rightarrow$ Parsing mit CFGs, Dependenzgrammatiken, PCFGs

- **NLTK-07**

  $\rightarrow$ partielles Parsing mit RegexpParser

- **NLTK-09**

  $\rightarrow$ 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

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

```
grammar = nltk.CFG.fromstring("""
    S → NP VP
    PP → P NP
    NP → Det N | Det N PP | 'I'
    VP → V NP | VP PP
    Det → 'an' | 'my'
    N → 'elephant' | 'pajamas'
    V → 'shot'
    P → 'in'
    """)

parser = nltk.ChartParser(grammar,trace=0)

for tree in parser.parse(sent):
    print(tree)
```

```
16
17 #(S
18 #   (NP I)
19 #   (VP
20 #     (VP (V shot) (NP (Det an) (N elephant)))
21 #     (PP (P in) (NP (Det my) (N pajamas)))))
22
23 #(S
24 #   (NP I)
25 #   (VP
26 #     (V shot)
27 #     (NP (Det an) (N elephant) (PP (P in) (NP
   (Det my) (N pajamas))))))
```

Auflistung 3: *Generierung Syntaxbaum Konstituentenanalyse*

```
1  from nltk.tree import Tree
2
3  tree1 = Tree.fromstring("""
4      (S
5      (NP I)
6      (VP
7      (VP (V shot) (NP (Det an) (N elephant)))
8      (PP (P in) (NP (Det my) (N pajamas)))))
9      """)
10
11 tree1.draw()
```

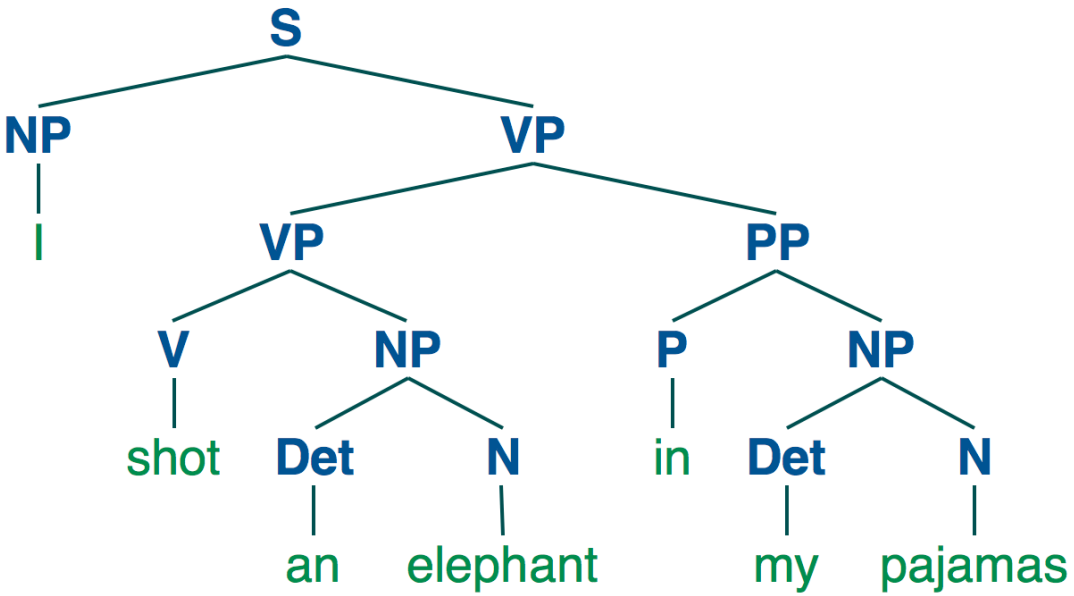

Abbildung 1: *Syntaxbaum Konstituentenanalyse*

Auflistung 4: *Generierung Syntaxbaum Konstituentenanalyse 2*

```
1 tree2 = Tree.fromstring("""
2     (S
3     (NP I)
4     (VP
5     (V shot)
6     (NP (Det an) (N elephant) (PP (P in) (NP
        (Det my) (N pajamas))))))
7     """)
8
9 tree2.draw()
```

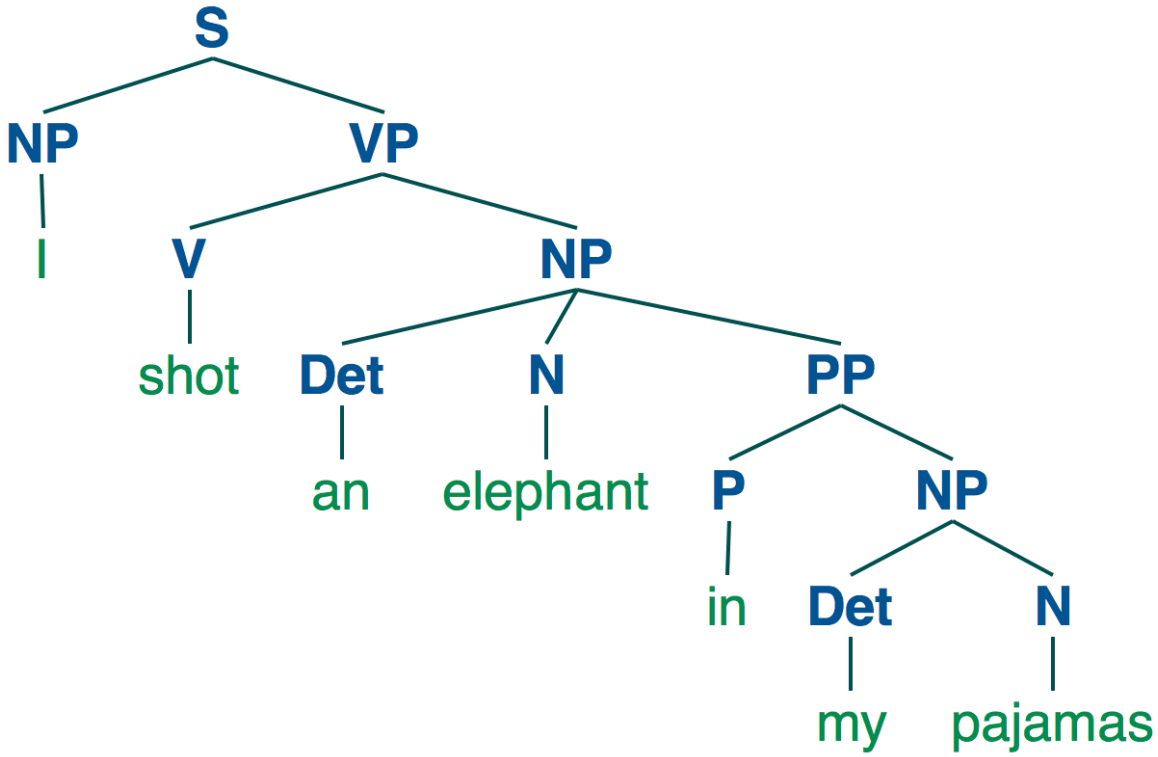

Abbildung 2: *Syntaxbaum Konstituentenanalyse 2*

1.2.2 Dependenz-Parsing

Auflistung 5: *Dependenzgrammatik*

```
grammar = nltk.DependencyGrammar.fromstring("""
    'shot' → 'I' | 'elephant' | 'in'
    'elephant' → 'an' | 'in'
    'in' → 'pajamas'
    'pajamas' → 'my'
    """)

parser =
  nltk.ProjectiveDependencyParser(grammar)

for tree in parser.parse(sent):
    print(tree)

#(shot I (elephant an (in (pajamas my))))
#(shot I (elephant an) (in (pajamas my)))
```

Auflistung 6: *Generierung Syntaxbaum Depdendenzanalyse*

```
1 from nltk.tree import Tree
2 from nltk.draw.tree import TreeView
3
4 #alternative Generierung mit Treeview:
5 DGTree1 = Tree.fromstring("(shot I (elephant
    an (in (pajamas my))))")
6 DGTree2 = Tree.fromstring("(shot I (elephant
    an) (in (pajamas my)))")
7
8 TreeView(DGTree1)
9 TreeView(DGTree2)
```

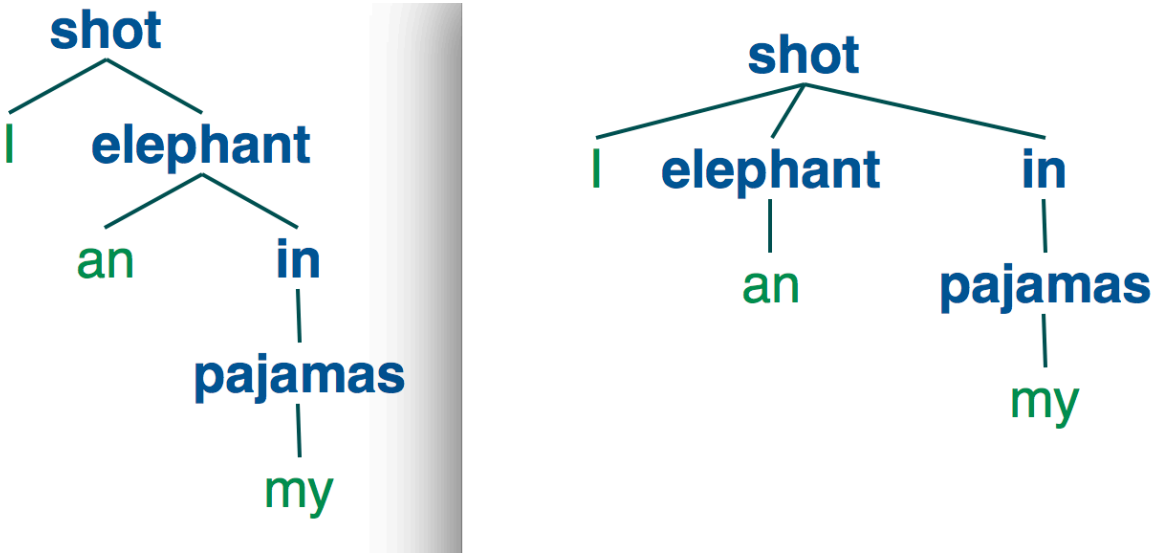

Abbildung 3: *Syntaxbäume Dependenzanalyse (Stemmas)*

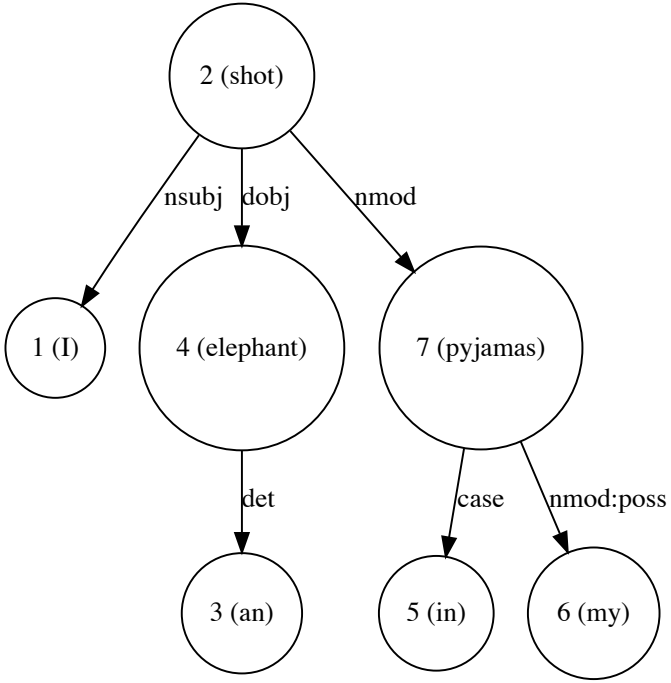

Abbildung 4: *Ausgabe Stanford Dependency Parser*

1.2.3 PCFG-Parsing

Auflistung 7: *Probabilistische kontextfreie Grammatik (PCFG)*

```
grammar1 = nltk.PCFG.fromstring("""
    S → NP VP [1.0]
    PP → P NP [1.0]
    NP → Det N [0.8] | Det N PP [0.1] | 'I' [0.1]
    VP → V NP [0.8] | VP PP [0.2]
    Det → 'an' [0.7] | 'my' [0.3]
    N → 'elephant' [0.5] | 'pajamas' [0.5]
    V → 'shot' [1.0]
    P → 'in' [1.0]
    """)
```

```
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

Auflistung 8: *feature-based grammar (Ausschnitt)*

```
1  ## Natural Language Toolkit: german.fcfg
2  ## Example of a feature-based grammar for
     German, illustrating
3  ## CASE and AGR features (PER, GND, NUM)
     working as a bundle.
4  ## Used in Feature-Based Grammars chapter.
5  % start S
6
7
8
9
10
11
12
13
```

```
14 ######################
15 # Grammar Productions
16 ######################
17 S → NP[CASE=nom, AGR=?a] VP[AGR=?a]
18
19 NP[CASE=?c, AGR=?a] → PRO[CASE=?c, AGR=?a]
20 NP[CASE=?c, AGR=?a] → Det[CASE=?c, AGR=?a]
     N[CASE=?c, AGR=?a]
21
22 VP[AGR=?a] → IV[AGR=?a]
23 VP[AGR=?a] → TV[OBJCASE=?c, AGR=?a]
     NP[CASE=?c]
24
25
26
27
```

```
28 ######################
29 # Lexical Productions
30 ######################
31 # Singular determiners
32
33 # masc
34 Det[CASE=nom, AGR=[GND=masc,PER=3,NUM=sg]] →
     'der'
35 Det[CASE=dat, AGR=[GND=masc,PER=3,NUM=sg]] →
     'dem'
36 Det[CASE=acc, AGR=[GND=masc,PER=3,NUM=sg]] →
     'den'
37
38 # fem
39 Det[CASE=nom, AGR=[GND=fem,PER=3,NUM=sg]] →
     'die'
```

```
40 Det[CASE=dat, AGR=[GND=fem,PER=3,NUM=sg]] →
     'der'
41 Det[CASE=acc, AGR=[GND=fem,PER=3,NUM=sg]] →
     'die'
42
43 # Plural determiners
44 Det[CASE=nom, AGR=[PER=3,NUM=pl]] → 'die'
45 Det[CASE=dat, AGR=[PER=3,NUM=pl]] → 'den'
46 Det[CASE=acc, AGR=[PER=3,NUM=pl]] → 'die'
47
48 # Nouns
49 N[AGR=[GND=masc,PER=3,NUM=sg]] → 'Hund'
50 N[CASE=nom, AGR=[GND=masc,PER=3,NUM=pl]] →
     'Hunde'
```

Auflistung 9: *feature-based parsing*

```
1  sent = 'der Hund sieht uns'.split()
2
3  from nltk import load_parser
4
5  parser =
     load_parser('./grammars/book_grammars/german.fcfg',
     trace=0)
6
7  for tree in parser.parse(sent):
8      print(tree)
9
10
11
12
13
```

```
14 #(S[]
15 #  (NP[AGR=[GND='masc', NUM='sg', PER=3], CASE='nom']
16 #    (Det[AGR=[GND='masc', NUM='sg', PER=3], CASE='nom'] der)
17 #    (N[AGR=[GND='masc', NUM='sg', PER=3]] Hund))
18 #  (VP[AGR=[NUM='sg', PER=3]]
19 #    (TV[AGR=[NUM='sg', PER=3], OBJCASE='acc'] sieht)
20 #    (NP[AGR=[NUM='pl', PER=1], CASE='acc']
21 #      (PRO[AGR=[NUM='pl', PER=1], CASE='acc'] uns))))
```

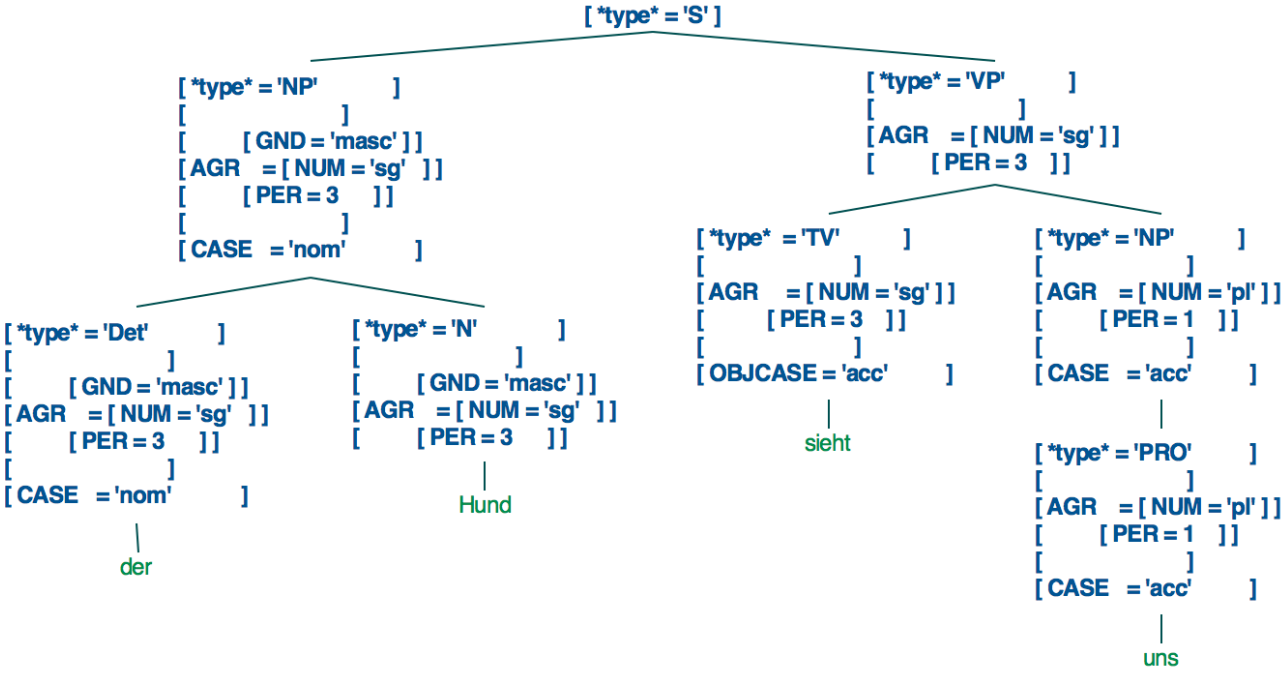

Abbildung 5: *Syntaxbaum feature-based*

1.2.5 Partielles Parsing mit RegexpParser

Auflistung 10: *Partielles Parsing (reguläre Grammatik)*

```
grammar = r"""
    NP: {<DT|PP\$>?<JJ>*<NN>}
# chunk determiner/possessive, adjectives and noun
    {<NNP>+}
# chunk sequences of proper nouns
    """
parser = nltk.RegexpParser(grammar)

sent = [("Rapunzel", "NNP"), ("let", "VBD"), ("down", "RP"), ("her", "PP$"), ("long", "JJ"), ("golden", "JJ"), ("hair", "NN")]


```

```
13 tree = parser.parse(sent)
14 print(tree)
15 #(S
16 # (NP Rapunzel/NNP)
17 #  let/VBD
18 #  down/RP
19 #  (NP her/PP$ long/JJ golden/JJ hair/NN))
20 tree.draw()
```

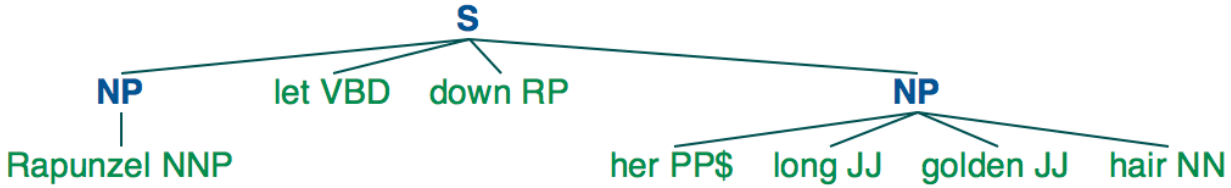

Abbildung 6: *Syntaxbaum NP-Chunking-Analyse*