

# Syntax natürlicher Sprachen

## 12: Partielles Parsing

A. Wisiorek

Centrum für Informations- und Sprachverarbeitung,  
Ludwig-Maximilians-Universität München

21.01.2025

# Themen der heutigen Vorlesung

- 1 Partielles Parsing (Chunking)
- 2 Chunking mit regulärer Grammatik
  - RegExp-Chunk-Parser
  - Kaskadierende Chunker
  - Evaluation von Chunkern
- 3 Lernbasierte Chunking-Modelle (IOB-Tagger)

# 1. Partielles Parsing (Chunking)

## 1 Partielles Parsing (Chunking)

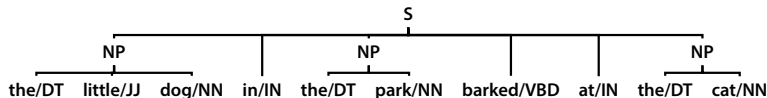
## 2 Chunking mit regulärer Grammatik

- RegExp-Chunk-Parser
- Kaskadierende Chunker
- Evaluation von Chunkern

## 3 Lernbasierte Chunking-Modelle (IOB-Tagger)

- für viele Anwendungen: **keine syntaktische Vollanalyse notwendig**
- **Partielles Parsing** als **unvollständige, flache Syntaxanalyse**
  - *nur die Konstituententypen mit **Inhaltswort als Kopf**:*  
*NP, VP, PP, AdjP*
  - *auch als **Chunking** bezeichnet (Abney 1991: 'parsing by chunks')*
- Partielle-Parsing-Methoden u.a. entwickelt für:
  - **Informationsextraktion**: Finden semantischer Einheiten
  - **Information Retrieval**: Phrasen als Indexterme

- **unvollständige Analyse:** nur die Wörter berücksichtigt, die **Element der relevanten syntaktischen Einheiten** sind
- **andere Wörter** werden **nicht syntaktisch annotiert**  
⇒ **Partielle syntaktische Analyse**
- z. B. nur NP-Chunks:



- flache, **nicht-hierarchische syntaktische Analyse**

- *Chunk-Bäume haben Tiefe 1*

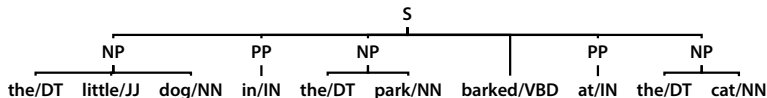
- **keine komplexen, verschachtelten Phrasen**

- **Chunk als base phrase:**

- *kleinere Einheiten als vollständige Phrasen*

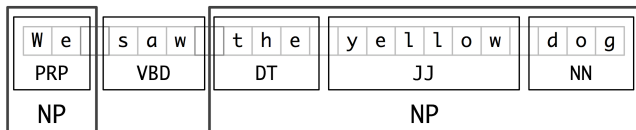
- *NP-Chunks ohne Rechtsattribute*

- *PP-Chunks bestehen nur aus Präposition:*



## 2 Aufgaben für Chunk-Parser

- 1 **Segmentierung** = **Grenzen** der Chunks **finden**  
→ Identifizierung von **Sequenzen von Tokens als Chunks**
- 2 **Labeling** der Chunks  
→ **Klassifikation der Einheiten = Tagging**



**Abbildung:** Segmentierung und Labeling auf Token- und Chunk-Ebene  
(<http://www.nltk.org/images/chunk-segmentation.png>)

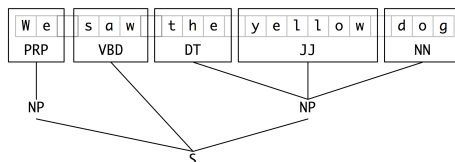
## 2 Methoden zur Erstellung von Chunk-Parsern

- ① **regelbasiert** mit regulärer Grammatik
- ② ***supervised machine learning***
  - zentrales Merkmal in beiden Ansätzen: **POS-Tags**



# Repräsentationsformate für Chunk-Analysen

- **Darstellung über flache Bäume:**



**Abbildung:** Baum-Repräsentation von Chunks  
(<http://www.nltk.org/images/chunk-treerep.png>)

- **Darstellung über Tags (IOB-Format):**

W	e	s	a	w	t	h	e	y	e	l	l	o	w	d	o	g
PRP		VBD			DT		JJ						NN			
B-NP		O			B-NP		I-NP						I-NP			

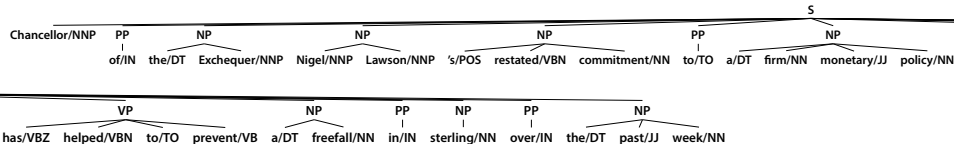
**Abbildung:** Tag-Repräsentation von Chunks im IOB-Format  
(<http://www.nltk.org/images/chunk-tagrep.png>)

## Vorteile

- **einfaches Modell**, in vielen Anwendungsfällen ausreichend
- **weniger Disambiguierungsprobleme** als bei PSG-Grammatiken  
→ z. B. *Vermeidung PP-Attachment-Ambiguität durch nicht-hierarchische Analyse*
- **bessere Performance** gegenüber CFGs oder Unifikationsgrammatiken, z.B. **Chunker als reguläre Grammatik**
- **hohe Accuracy** aufgrund flacher, unvollständiger Strukturanalyse

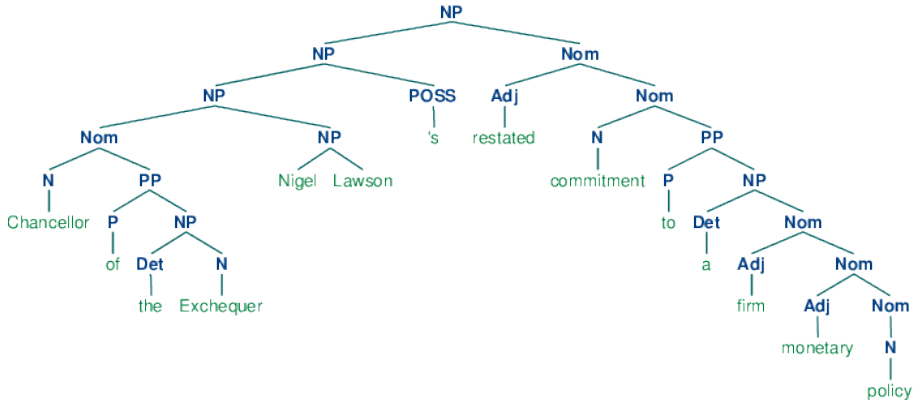
## Nachteile

- nur **unvollständige Beschreibung** syntaktischer Struktur
- **grammatische Beziehungen** in der syntaktischen Struktur **durch die flache Analyse** nicht direkt modelliert
  - *nur heuristisch erfassbar, z. B.:*
  - *NP vor Verb ist Subjekt oder erste NP ist Subjekt*
  - *Heuristik kann bei komplexen NPs fehlschlagen:*



## Listing 1: NLTK: Beispiel flach analysierter komplexer NP (aus con112000-Korpus)

```
1 #(S
2 #  Chancellor/NNP
3 #  (PP of/IN)
4 #  (NP the/DT Exchequer/NNP)
5 #  (NP Nigel/NNP Lawson/NNP)
6 #  (NP 's/POS restated/VBN commitment/NN)
7 #  (PP to/TO)
8 #  (NP a/DT firm/NN monetary/JJ policy/NN)
9 #  (VP has/VBZ helped/VBN to/TO prevent/VB)
10 #  (NP a/DT freefall/NN)
11 #  (PP in/IN)
12 #  (NP sterling/NN)
13 #  (PP over/IN)
14 #  (NP the/DT past/JJ week/NN)
15 #  ./.)
```



**Abbildung:** Vollständige Phrasenstrukturanalyse der komplexen Subjekt-NP  
 ([http://www.nltk.org/book/tree\\_images/ch08-extras-tree-1.png](http://www.nltk.org/book/tree_images/ch08-extras-tree-1.png))

- in **hierarchischer Phrasenstrukturanalyse** wird die **komplexe NP unter einem NP-Knoten** zusammengefasst (statt in viele NP-Chunks aufgesplittet)
  - *komplexe NPs werden in ihrem hierarchischem Aufbau analysiert und der Kopf der Phrase ist in einem X-Bar-Schema über Strukturposition eindeutig identifizierbar: NP, NOM+, N*
- **Subjekt- und Objekt-Funktion** sind über **Position im Strukturbaum** repräsentiert
  - *Subjekt-NP unmittelbar dominiert von S*
  - *Objekt-NP unmittelbar dominiert von VP*

## 2. Chunking mit regulärer Grammatik

1 Partielles Parsing (Chunking)

2 Chunking mit regulärer Grammatik

- RegExp-Chunk-Parser
- Kaskadierende Chunker
- Evaluation von Chunkern

3 Lernbasierte Chunking-Modelle (IOB-Tagger)

## 2.1. RegExp-Chunk-Parser

- 1 Partielles Parsing (Chunking)
- 2 Chunking mit regulärer Grammatik
  - **RegExp-Chunk-Parser**
  - Kaskadierende Chunker
  - Evaluation von Chunkern
- 3 Lernbasierte Chunking-Modelle (IOB-Tagger)



- **Beschreibung von POS-Folgen** durch **reguläre Ausdrücke** (Tag-Pattern)
- **Tag-Muster** = Regeln einer **Chunk-Grammatik**
  - *NLTK-eigene Syntax: POS-Tags in eckigen Klammern:*
  - z. B. *Chunk als POS-Folge Artikel - Adjektivattribute (optional) - Nomen:*  
<DT><JJ>\*<NN>
- Chunking als **sukzessive Anwendung der Regeln** einer Chunk-Grammatik auf die Sätze eines POS-getaggten Textes
- **FSTs** = *finite state transducers* ermöglichen **Segmentierung und Labeling** (Output durch *transducer*): NP: <DT><JJ>\*<NN>

- **bereits gefundene Chunk-Elemente:** in folgenden Regeln **nicht mehr verfügbar**
  - *nur nicht-überlappende Chunks* werden gefunden
  - **Reihenfolge der Regeln** in Grammatik ist also **wichtig**
  - z.B. wird bei invertierter Regel-Abfolge die *POSS-ADJ-NN-Folge* nicht gefunden, wenn *NN* schon als NP erkannt
- **Tracing** hilfreich beim **Entwickeln von Chunk-Grammatiken**
  - *Beschreibung der Regeln durch Kommentare*

- **Chunking-Regel:** Definition Chunk als Muster einer POS-Folge:  
→ *NP*: {<NNP>+}
- **Chinking-Regel:** Definition eines Musters einer POS-Folge, die **nicht in Chunk enthalten** ist:  
→ }<VBD/IN>+{  
[ the/DT little/JJ yellow/JJ dog/NN ] barked/VBD at/IN [ the/DT cat/NN ]  
→ *Anwendung auf bereits gefundene Chunks*
- **Möglichkeiten beim Chinking:**
  - wenn Chink-Muster in **Mitte** von Chunk  
⇒ Chunk wird entsprechend **gesplittet**
  - wenn Chink-Muster **kompletter Chunk**  
⇒ Chunk wird **entfernt**
  - wenn Chink-Muster am **Anfang oder Ende** von Chunk  
⇒ Chunk wird entsprechend **beschnitten**

- **Split-Regeln**

- *Anwendung auf gefundene Chunks*

- z. B. *Splitten* von gefundener NP am Determinierer:

- $\langle . * \rangle \{ \langle DT \rangle$

## 2.2. Kaskadierende Chunker

- 1 Partielles Parsing (Chunking)
- 2 Chunking mit regulärer Grammatik
  - RegExp-Chunk-Parser
  - **Kaskadierende Chunker**
  - Evaluation von Chunkern
- 3 Lernbasierte Chunking-Modelle (IOB-Tagger)

- durch **Kombination** von (flachen) **Chunk-Parsern** können **hierarchische, tiefere Strukturen** erzeugt werden
  - ***Output eines Chunkers dient als Input des Folgenden***
  - ***Regeln können Chunk-Tags enthalten:***
    - ⇒ *hierarchische Strukturen, z. B. PP: {<IN><NP>}*)
  - ***Approximation Output eines PSG-Parsers***
- **Loopen eines Chunk-Parsers** über von diesem Parser erkannte Muster:
  - ***findet Chunks, die in einem ersten Durchlauf nicht erkannt wurden***

# Beispiel: hintereinandergeschaltete Chunk-Parser für komplexe PPs

- 1 Suche **Adjektivattribute (AdjP)**: <JJ>+
- 2 Suche **NPs**: <DT><ADJP><NN>
- 3 Suche **PPs**: <P><NP>

Satz	on/IN	the/DT		black/JJ		mat/NN
Chunker 1	on/IN	the/DT	[ADJP	black/JJ	]	mat/NN
Chunker 2	on/IN	[NP the/DT	[ADJP	black/JJ	]	mat/NN ]
Chunker 3	[PP on/IN	[NP the/DT	[ADJP	black/JJ	]	mat/NN ] ]

**Abbildung:** Beispiel kaskadierender Chunk-Parser zur Analyse einer komplexen PP

- Beispiel: in Objektsatz eingebetteter Objektsatz:
  - *John thinks Mary saw the cat sit on the mat*
- Regeln müssen zwei Mal angewendet werden, um die zweifache Clause-Einbettung zu erkennen



## 2.3. Evaluation von Chunkern

- 1 Partielles Parsing (Chunking)
- 2 Chunking mit regulärer Grammatik
  - RegExp-Chunk-Parser
  - Kaskadierende Chunker
  - **Evaluation von Chunkern**
- 3 Lernbasierte Chunking-Modelle (IOB-Tagger)

- **con112000-Korpus** im NLTK (270.000 Tokens)
  - *POS- und Chunk-getaggetes Korpus mit NPs, VPs, und PPs*
  - *Aufteilung in Test- und Trainingsmenge*
- Auswahl **Chunks bestimmter Typen** mit `chunk_types`-Argument
- Das Korpus kann zum **Evaluieren von Chunk-Parsern** verwendet werden
- **Vergleich Chunker-Output** (Hypothese) mit Test-Menge als *gold-standard-Referenz-Korpus* (annotiert von Experten)

- **korrekter Chunk = korrekte Spanne und korrektes Label**
- **Recall:**  $R = \frac{(\text{Anzahl von korrekten Chunks in Chunker-Output})}{(\text{Anzahl aller Chunks in Referenz-Korpus})}$   
→ Anteil der Chunks des Referenz-Korpus, die vom Chunker korrekt identifiziert wurden
- **Precision:**  $P = \frac{(\text{Anzahl von korrekten Chunks in Chunker-Output})}{(\text{Anzahl aller Chunks in Chunker-Output})}$   
→ Anteil der vom Chunker identifizierten Chunks, die korrekt sind
- **F-score** =  $\frac{(\beta^2 + 1)PR}{\beta^2 P + R}$   
→ Kombination von Precision und Recall in einem Maß  
→  $\beta$ : Parameter zur Gewichtung von  $P$  und  $R$ , gleichgewichtet:

$$F_1 = \frac{2PR}{P + R} \quad (\text{harmonisches Mittel von } P \text{ und } R)$$

### 3. Lernbasierte Chunking-Modelle (IOB-Tagger)

- 1 Partielles Parsing (Chunking)
- 2 Chunking mit regulärer Grammatik
  - RegExp-Chunk-Parser
  - Kaskadierende Chunker
  - Evaluation von Chunkern
- 3 Lernbasierte Chunking-Modelle (IOB-Tagger)

- **Training eines Klassifikators, der die Wörter auf Chunk-Klassen abbildet**
- Vorgehen **analog zu POS-Tagging** mit *supervised machine-learning*
  - **POS-Tagging**: Abbildung *Token auf POS-Tag*
  - **Chunking**: Abbildung *Token-POS-Tupel auf IOB-Tag (IOB-Tagging)*
  - = **Sequenzklassifikation** ('parsing as tagging')
- für **Training** ist ein **Chunk-getaggtes Korpus** im IOB-Format notwendig, z. B. das **con112000-Korpus** im NLTK

W	e	s	a	w	t	h	e	y	e	l	l	o	w	d	o	g
PRP		VBD			DT		JJ				NN					
B-NP		O			B-NP		I-NP				I-NP					

Abbildung: IOB-Tags als Chunk-Tags  
(<http://www.nltk.org/images/chunk-tagrep.png>)

- **IOB = 'Inside-Outside-Beginning'**
- IOB-Tag repräsentiert gleichzeitig **Segmentierung+Label**
- **wortweise** Auszeichnung von **verbundenen Sequenzen**
- **Ende eines Chunks implizit kodiert:**  
→ *Übergang von I oder B zu B (neuer Chunk) oder O (Outside)*

# Chunking als *supervised*-Sequenzklassifikation

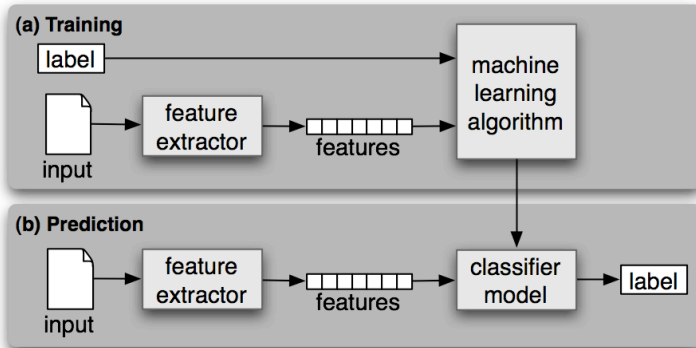
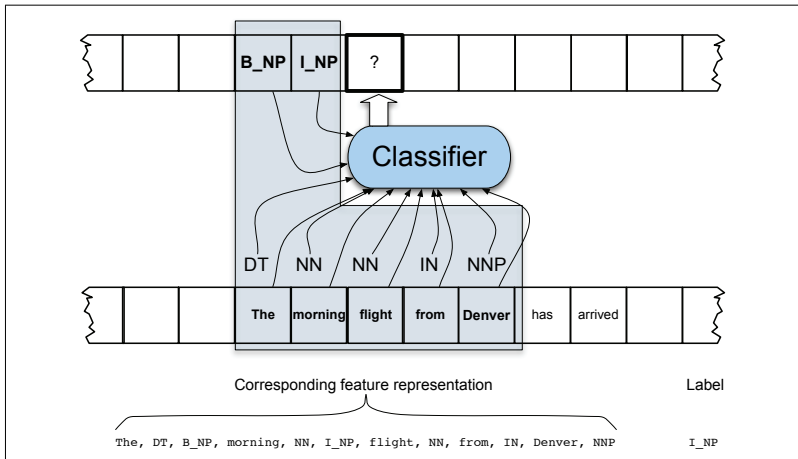


Abbildung: Schema *supervised*-Klassifikation  
(<http://www.nltk.org/images/supervised-classification.png>)



**Abbildung:** Chunking als Sequenzklassifikation (momentaner Zustand: Tagging von *flight*); Input wird durch eine aus Kontextfenster extrahierte Feature-Menge repräsentiert (Abbildung nach: Speech and Language Processing. Daniel Jurafsky & James H. Martin. Draft of August 28, 2017, Figure 12.8, p. 12, <https://web.stanford.edu/~jurafsky/slp3/12.pdf>)