Wiederholungsaufgaben 2: Themen 10-13
Lösung

1 Automatische Syntaxanalyse B

Grammatiktypen .....

- Nennen Sie die wichtigsten differenzierenden Merkmale der unten genannten Modellierungen der Syntax natürlicher Sprachen.
- Berücksichtigen Sie dabei auch folgende Kriterien:
  - Gegenstand der Strukturanalyse
  - Analysetiefe (hierarchische vs. flache Strukturanalyse)
  - formale Modellierung
  - Verarbeitung / verwendbare Parsingalgorithmen

1. **Merkmalsstrukturbasierte Grammatik (FCFG)**

Lösung:

- Modellierung von Phrasenstruktur und Morphosyntax
- hierarchische Konstituenten-Strukturanalyse
- Erweiterung atomarer CFG-Kategorien zu Merkmalstrukturen mit grammatischen Merkmalen
  - ermöglicht Bestimmung von morphosyntaktischen Constraints
  - verhindert Überproduktion
- Verarbeitung: Unifikation als zentraler Mechanismus

2. **Probabilistische kontextfreie Grammatik (PCFG)**

Lösung:

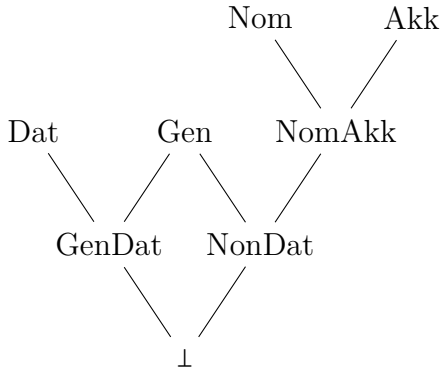
- statistische Modellierung von Phrasenstruktur (gewichtete CFG-Regeln)
- hierarchische Konstituenten-Strukturanalyse
- statistisches Modell (aus Korpus induzierte gewichtete CFG-Regeln)
  - Abschätzung der Regelwahrscheinlichkeit über MLE
  - verhindert Überproduktion (unwahrscheinlich = nicht-grammatisch)
  - ermöglicht strukturelle Disambiguierung (Auswahl wahrscheinlichster Ableitung)
- Verarbeitung u.a. mit Viterbi-Parser
- 2 Erweiterungen der Kategorien von PCFGS:
  - lexikalisierte PCFG: Erweiterung um Kopfannotation
  - *history-based* PCFG: Erweiterung um *Parent-Annotation*

3. **Partielles Parsing (Chunking)**

Lösung:

- Identifizierung der wichtigsten Syntaktischen Einheiten (NP, VP, PP)
- partielle, flache Konstituenten-Strukturanalyse
  - durch Hintereinanderschalten von Grammatiken: Erzeugung hierarchischer Strukturen
- formale Modellierung mit regulären Grammatiken oder statistischem Modell (gelernt aus IOB-Tag-Sequenzen)

10 Unifikationsparsing und getypte Merkmalstrukturen



Gegeben sei folgende Typhierarchie:

Typhierarchien .....

Unifizieren Sie die folgenden Paare von Typen. Typen, die nicht unifizieren, markieren Sie als *undefiniert*.

(a) Nom ⊔ Akk = _____

(b) GenDat ⊔ NonDat = _____

(c) GenDat ⊔ Gen = _____

(d) Nom ⊔ ⊥ = _____

(e) Akk ⊔ NomAkk = _____

Lösung:

(a) undefiniert

(b) Gen

(c) Gen

(d) Nom

(e) Akk

Subsumption .....

Gegeben seien nun zusätzlich folgende Merkmalstrukturen mit $\theta(\text{FS1}) = \theta(\text{FS2}) = \theta(\text{FS3}) = \bot$.

$$\textsc{fs}1 = \begin{bmatrix} \textsc{cas} & \text{NomAkk} \\ \textsc{gen} & \text{mask} \end{bmatrix} \qquad \textsc{fs}2 = \begin{bmatrix} \textsc{cas} & \text{Nom} \\ \textsc{gen} & \text{mask} \\ \textsc{per} & 3 \end{bmatrix} \qquad \textsc{fs}3 = \begin{bmatrix} \textsc{cas} & \text{Akk} \\ \textsc{per} & 3 \end{bmatrix}$$

Entscheiden Sie jeweils mit *ja* oder *nein*:

(a) $\textsc{fs}2 \sqsubseteq \textsc{fs}3$? _____

(b) $\textsc{fs}1 \sqsubseteq \textsc{fs}2$? _____

(c) $\textsc{fs}2 \sqsubseteq \textsc{fs}1$? _____

(d) $\textsc{fs}3 \sqsubseteq \textsc{fs}2$? _____

(e) $\textsc{fs}3 \sqsubseteq \textsc{fs}3$? _____

Lösung:

(a) nein

(b) ja

(c) nein

(d) nein

(e) ja

Unifikation

Unifizieren FS2 und FS3? Falls ja, unifizieren Sie; falls nein, begründen Sie.

Lösung:

Nein, da FS2@CAS = Nom, FS3@CAS = Akk und Nom $\sqcup$ Akk undefiniert ist (bzw. weil Nom und Akk nicht unifizieren).

Bedingungen

Entscheiden Sie jeweils mit *ja* oder *nein*:

(a) FS2 $\models$ CAS : $\bot$? _____

(b) FS2 $\models$ GEN : neut ? _____

(c) FS3 $\models$ PER : 3 ? _____

(d) FS1 $\models$ NomAkk $\wedge$ mask ? _____

(e) FS2 $\models$ CAS : NomAkk ? _____

Lösung:

(a) ja

(b) nein

(c) ja

(d) nein

(e) ja

11 Statistisches Parsing

PCFG: Gewichte und Ableitungswahrscheinlichkeit

Betrachten Sie folgendes PCFG-Parsing (** = unkenntlich gemacht):

```
1  grammar = nltk.PCFG.fromstring("""
2      S    -> NP VP              [1.0]
3      VP   -> TV NP              [0.4]
4      VP   -> IV                 [**]
5      VP   -> DatV NP NP         [0.3]
6      TV   -> 'saw'              [1.0]
7      IV   -> 'ate'              [1.0]
8      DatV -> 'gave'             [1.0]
9      NP   -> 'telescopes'       [0.8]
10     NP   -> 'Jack'             [0.2]
11     """)
12 viterbi_parser = nltk.ViterbiParser(grammar)
13 for tree in viterbi_parser.parse(['Jack', 'saw', 'telescopes']):
14     print(tree)
15 (S (NP Jack) (VP (TV saw) (NP telescopes))) (p=0.064)
```

(a) Geben Sie die Berechnung für die Ableitungswahrscheinlichkeit in Zeile 15 an?

Lösung:
$1.0 * 0.2 * 0.4 * 1.0 * 0.8$ oder $0.2 * 0.4 * 0.8$ oder beliebige Permutationen

(b) Welchen Wert muss das Gewicht für die Regel **VP → IV** haben?

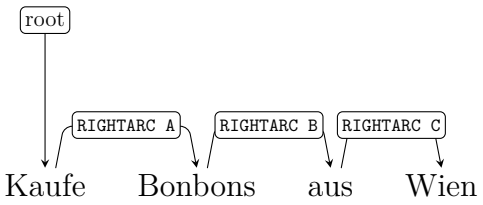
Lösung:
0.3

- Gewichte der beiden andere VP-Regeln:
  - **VP → TV NP**: 0.4
  - **VP → DatV NP NP**: 0.3
- Gesamtwahrscheinlichkeit für VP-Regeln muss 1 ergeben:
  1 - 0.4 - 0.3 = 0.3

Übergangsbasierter Shift-Reduce-Dependency-Parser .....

- In welcher Reihenfolge werden im Folgenden jeweils die angegebenen `REDUCE`-Übergange durchgeführt?
- Begründen Sie.
- Wie unterscheiden sich die beiden Syntaxbäume?

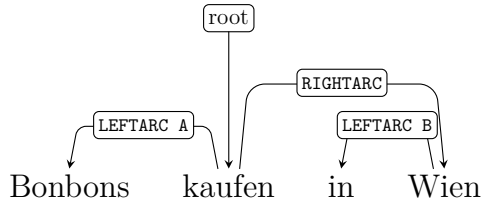
(a)



Lösung:

- `RIGHTARC C`, dann `RIGHTARC B`, dann `RIGHTARC A`
- – anderenfalls werden die Dependenten *Bonbons* (mit `RIGHTARC A`) bzw. *aus* (mit `RIGHTARC B`) zu früh vom Stack genommen
  – diese sind nämlich selbst wiederum Köpfe von Dependenten
  – würde man beispielsweise mit `RIGHTARC A` beginnen, könnte der `RIGHTARC B`-Übergang nicht mehr durchgeführt werden, da *Bonbons* als Dependent von *Kaufe* schon vom Stack gelöscht wäre.
- **`RICHTARC`-Regel:**
  – *`RICHTARC`-Übergang nur durchführen, wenn der Dependent der möglichen Relation nicht Kopf einer der Relationen aus der Menge offener Relationen ist*
  – *sonst: `SHIFT`-Übergang*

(b)



Lösung:

- `LEFTARC A`, dann `LEFTARC B`, dann `RIGHTARC`
- im Unterschied zu oben ist diese Reihenfolge (`RIGHTARC` zuletzt) nicht durch die `RICHTARC`-Regel (s.o.) bedingt, sondern kommt durch das sukzessive Hinzufügen der Wörter auf den Stack (`SHIFT`-Übergang):
  - zwischen *kaufen* und *in* gibt es hier keine Relation
  - entsprechend wird mit `SHIFT` *Wien* auf den Stack geschoben, der dann so aussieht: $[kaufen, in, Wien]$
  - nun wird mit `LEFTARC B` *in* als Dependent dieser Relation vom Stack entfernt; dieser sieht nun so aus: $[kaufen, Wien]$
  - jetzt kann mit `RICHTARC` *Wien* vom Stack gelöscht werden (da es keine offene Relation mehr gibt, in der *Wien* Kopf ist)
- Unterschied zu oben:
  - diese Dependenzanalyse folgt der *primacy of content words*-Maxime des UD-Schemas (Substantiv als Dependent statt Präposition)
  - Wortstellung (Imperativ vs Infinitiv)
  - Präpositionalphrase ist hier Adverbial, nicht Attribut

12 Datengestützte Syntaxanalyse

Datengestützte Methoden: Abschätzung Regelwahrscheinlichkeiten .....

(a) Für ein großes Korpus sei lediglich *part of speech* (POS) annotiert; Syntaxbäume stehen nicht zur Verfügung.
Welche Arten von Regeln einer kontextfreien Grammatik kann man mit diesen Daten automatisch generieren?

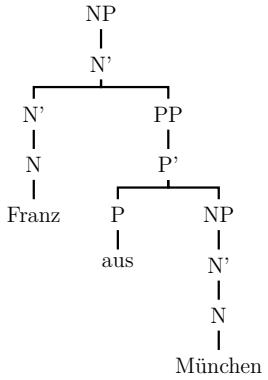
Lösung:
Lexikalische Regeln

(b) Folgende Häufigkeiten wurden aus einem Datensatz gezählt:
$count(VP \rightarrow V) = 200$, $count(VP \rightarrow V\ NP) = 100$, $count(VP \rightarrow \setminus *) = 300$.
Berechnen Sie $P(V\ NP \mid VP)$ mit der MLE-Methode.

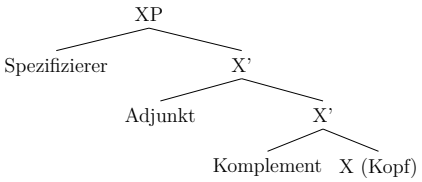
Lösung:
$\frac{100}{300} = \frac{1}{3} \approx 33.3\%$

Methoden für lexikalisierte und *history-based* PCFGs .....

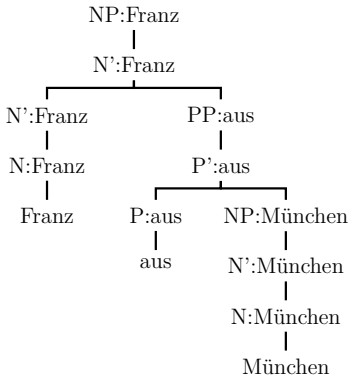
(a) Führen Sie im linken Syntaxbaum eine Kopfannotation durch; Geben Sie anschließend die lexikalisierte Regel für den Wurzelknoten an. Orientieren Sie bei der Kopfannotation an der Strukturposition des Kopfes im X-Bar-Schema (vgl. rechter Syntaxbaum).



Vgl.:

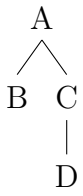


Lösung:



- **NP:Franz → N':Franz**

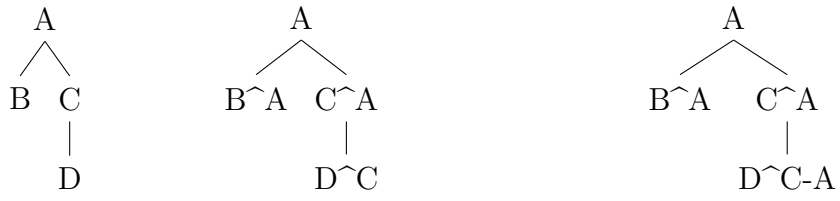
(b) Führen Sie für die CFG-Regeln **C → D** und **A → B** in unterem Syntaxbaum *parent-annotation* durch.



Lösung:

- **C^A → D^C**
- **A^? → B^A**

Original: **Parent Annotation:** **+Grandparent Annot.:**



13 Partielles Parsing

Chunking .....

Markieren Sie alle Nominalphrasen (NPs), indem Sie den folgenden deutschen Satz vollständig nach dem IOB-Tagging-Schema annotieren; verwenden Sie nur folgende Label: `B-NP, I-NP, O`.

Token	Der	junge	Mann	gab	ihr	das	Buch	.
Tag								

Lösung:

Token	Der	junge	Mann	gab	ihr	das	Buch	.
Tag	B-NP	I-NP	I-NP	O	B-NP	B-NP	I-NP	O

Kaskadierende Chunk-Parser .....

(a) Mit welcher Methode kann z.B. folgende hierarchische Struktur einer Präpositionalphrase mit flachen Chunk-Parsern erzeugt werden:

[PP auf/P [NP dem/DET Baum/N]]

Lösung:

- hintereinandergeschaltete flache Chunk-Parser
  (= **kaskadierender Chunk-Parser**)
- Output des einen als Input des folgenden Chunkers

Evaluationsmetriken .....

Berechnen Sie Accuracy, Precision und Recall für folgende korrekte Annotationen (`truth`) und folgende Hypothesen (`predict`). Geben Sie bitte jeweils Brüche an.

Sample	0	1	2	3	4	5	6	7	8	9
`truth`	PP	PP	PP	O	O	O	O	PP	O	PP
`predict`	PP	O	O	PP	O	PP	PP	O	PP	PP

Accuracy: _____

Precision (für die Klasse PP): _____

Recall (für die Klasse PP): _____

Lösung:

Accuracy: $\frac{3}{10} = 30\%$

Precision (für die Klasse PP): $\frac{2}{6} \approx 33.3\%$

Recall (für die Klasse PP): $\frac{2}{5} = 40\%$