Probabilistic Context Free Grammars

Prof Dr. Matthew Crocker

Universität des Saarlandes

29. Juni 2015

Heute:

1. Warum probabilistisches Parsen?
2. Definition: Probabilistische Kontextfreie Grammatiken
3. Annahmen bei PCFGs
4. Wahrscheinlichkeiten von Sätzen und Bäumen in PCFGs
   - Inside-Algorithmus
   - Outside-Algorithmus
   - Viterbi-Algorithmus
   - Inside-Outside Algorithmus
5. Evaluation von Parsern

Rest des Semesters:

- Informationstheorie und CL Anwendungen
- Maschinelles Lernen und CL Anwendungen

Table of Contents

Parsing vs. n-gram Modelle oder POS-tagging

- Im vorigen Kursteil haben Sie n-gram Modelle (Markov Ketten) und Hidden Markov Modelle kennengelernt.
- Diese können als Sprachmodelle trainiert werden.
- Allerdings können sie die hierarchische Struktur von Sprache nicht ausdrücken.

Beispiel

The velocity of the seismic waves rises to...
HMM oder n-gram Model weist geringe Wahrscheinlichkeit zu, da singular Verb auf plural Nomen folgt.

- Um die hierarchische Struktur zu modellieren, können wir einem Satz Baumstrukturen zuweisen und deren Wahrscheinlichkeiten lernen.

Ambiguität beim Parsing

Dass man den Baumstrukturen Wahrscheinlichkeiten zuweisen kann ist wichtig:

- Wörter können verschiedene Bedeutungen haben und mehr als einer Wortkategorien angehören
- Verschiedene Wortkategorien führen zu verschiedenen Satzstrukturen
- Außerdem gibt es strukturelle Ambiguitäten, die nicht aus dem Lexikon stammen

Beim Probabilistischen Parsing wollen wir den besten Parsebaum t für den Satz bestimmen:

$$\underset{t}{\text{argmax}}\, P(t|Satz, Grammatik)$$

Table of Contents

Probabilistische kontextfreie Grammatiken (PCFGs)

- Eine PCFG ist eine kontextfreie Grammatik, in der jede Regel mit einer Wahrscheinlichkeit versehen ist
- Die Summe der Wahrscheinlichkeiten aller Regeln mit dem selben Symbol auf der *linken* Seite muss 1 betragen

Formelle Notation

Eine PCFG G besteht aus

$\{w^1, \cdots, w^V\}$	Terminal vocabulary
$\{N^1, \cdots, N^n\}$	Nonterminal vocabulary
N^1	Start symbol
$\{N^i \rightarrow \zeta^j\}$	Grammar rules, where ζ^j is a sequence of terminals and nonterminals
$\{P(N^i \rightarrow \zeta^j)\}$	Rule probabilities such that $\forall_i \sum_j P(N^i \rightarrow \zeta^j) = 1$

Probabilistische kontextfreie Grammatiken (PCFGs)

- Eine PCFG ist eine kontextfreie Grammatik, in der jede Regel mit einer Wahrscheinlichkeit versehen ist
- Die Summe der Wahrscheinlichkeiten aller Regeln mit dem selben Symbol auf der *linken* Seite muss 1 betragen

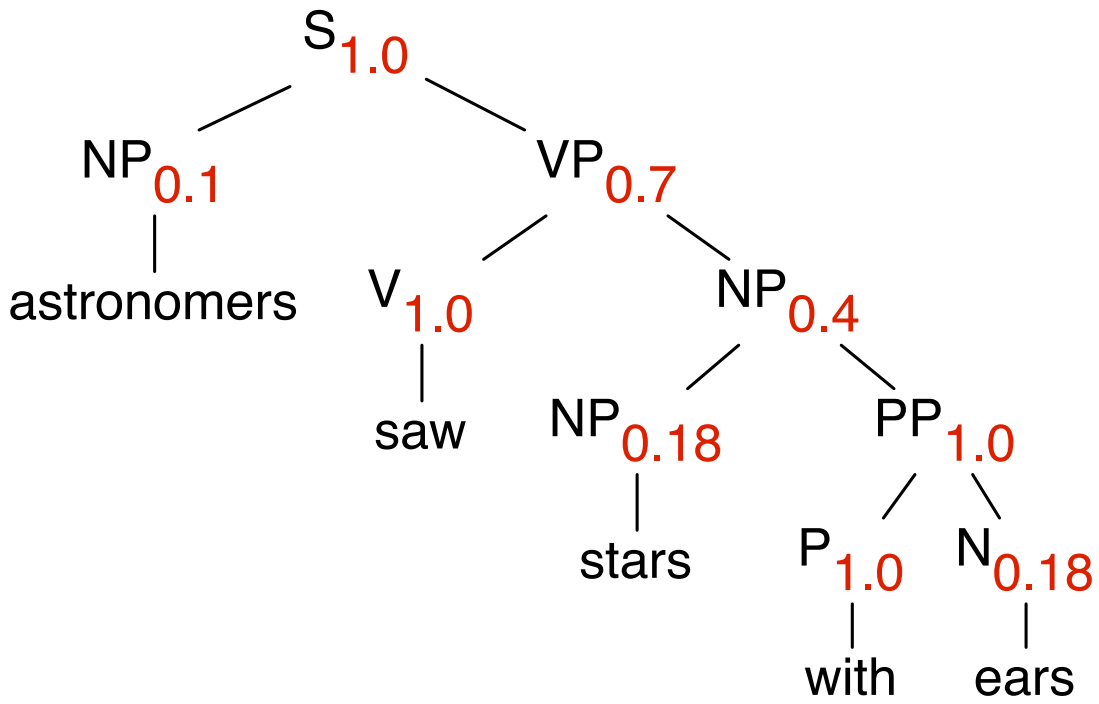
Beispiel

S	→	NP VP	1.0
VP	→	V NP	0.7
VP	→	VP PP	0.3
NP	→	NP PP	0.4
PP	→	P NP	1.0

NP	→	astronomers	0.1
NP	→	telescopes	0.1
NP	→	saw	0.04
NP	→	stars	0.18
NP	→	ears	0.18
P	→	with	1.0
V	→	saw	1.0

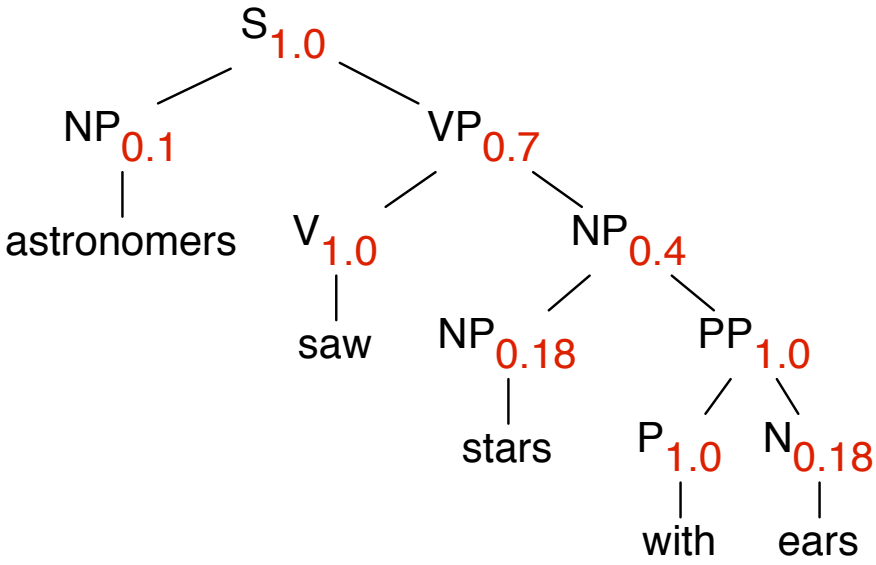
Wahrscheinlichkeit eines Parsebaums

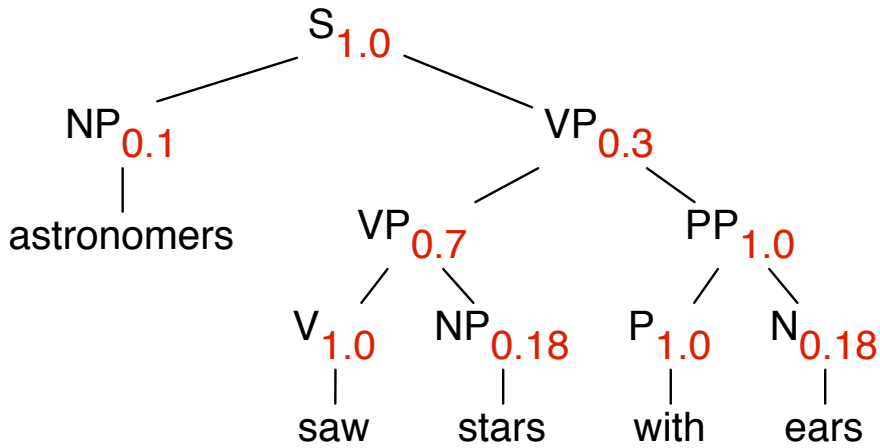
Die Wahrscheinlichkeit eines Parses ist das Produkt der Wahrscheinlichkeiten der Regeln, die während des Parsens angewandt werden



$$
\begin{aligned}
P(t1) &= 1.0 * 0.18 * 1.0 * 0.18 * 0.4 * 1.0 * 0.7 * 0.1 * 1.0 \\
&= 0.009072
\end{aligned}
$$

Wahrscheinlichkeit eines Satzes





$$P(t1) = 1.0 * 0.18 * 1.0 * 0.18 * 0.4 * 1.0 * 0.7 * 0.1 * 1.0 = 0.009072$$

$$P(t2) = 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 * 1.0 * 1.0 * 0.18 = 0.000680$$

Die Wahrscheinlichkeit eines Satzes ist die Summe der Wahrscheinlichkeiten aller Parsebäume:

$P(Satz) = P(t1) + P(t2) = 0.0015876$

Formell gesehen

Wahrscheinlichkeit eines Parsebaums:

$$P(t, Satz) = \prod_{n \in t} P(r(n))$$

Wahrscheinlichkeit eines Satzes:

$$P(Satz) = \sum_{t} P(t, Satz)$$

Notation

- t Parsebaum für den Satz
- n Knoten im Baum
- r Regel, die bei der Expansion eines Knotes angewandt wird

Table of Contents

Annahmen bei PCFGs

Wir treten nochmal einen Schritt zurück:

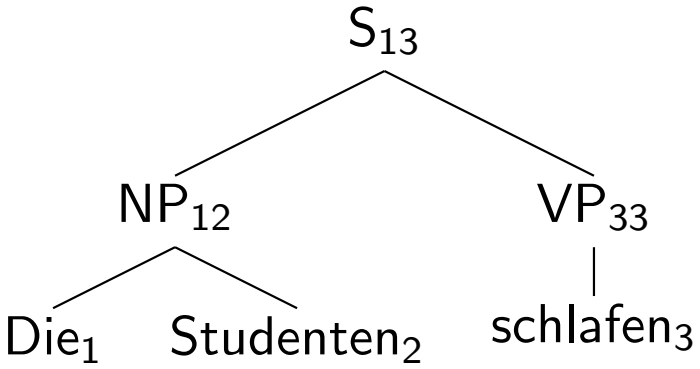
Wieso berechnen wir eigentlich die Wahrscheinlichkeit eines Parsebaums als das Produkt der Einzelwahrscheinlichkeiten der angewendeten Regeln?

Zugrundeliegend sind folgende Annahmen:

- **Positionsunabhängigkeit**: Die Wahrscheinlichkeit eines Teilbaums ist unabhängig davon, wo im Satz die entsprechende Wortfolge vorkommt (vgl. Zeitunabhängigkeit bei HMMs)
- **Kontextunabhängigkeit**: Die Wahrscheinlichkeit eines Teilbaums ist unabhängig von Wörtern, die er nicht dominiert
- **Vorfahrenunabhängigkeit**: Die Wahrscheinlichkeit eines Teilbaums ist unabhängig von Vorgängerknoten im Baum

Herleitung der Berechnung der Baumwahrscheinlichkeit

Wahrscheinlichkeit von Baumstrukur t für:



Notation

Subscripte $_{ab}$ an Knoten N bedeuten, dass N_{ab} Vorfahre der Worte mit Nummern a bis b ist.

$$
\begin{aligned}
&= P(S_{13} \rightarrow NP_{12}\,VP_{33}, NP_{12} \rightarrow Die_1\ Studenten_2, VP_{33} \rightarrow schlafen_3) \\
&= P(S_{13} \rightarrow NP_{12}\,VP_{33}) \times P(NP_{12} \rightarrow Die_1\ Studenten_2 | S_{13} \rightarrow NP_{12}\,VP_{33}) \\
&\quad \times P(VP_{33} \rightarrow schlafen_3 | S_{13} \rightarrow NP_{12}\,VP_{33},\ \ NP_{12} \rightarrow Die_1\ Studenten_2) \\
&= P(S_{13} \rightarrow NP_{12}\,VP_{33}) \times P(NP_{12} \rightarrow Die_1\ Studenten_2) \times P(VP_{33} \rightarrow schlafen_3) \\
&= P(S \rightarrow NP\ VP) \times P(NP \rightarrow Die\ Studenten) \times P(VP \rightarrow schlafen)
\end{aligned}
$$

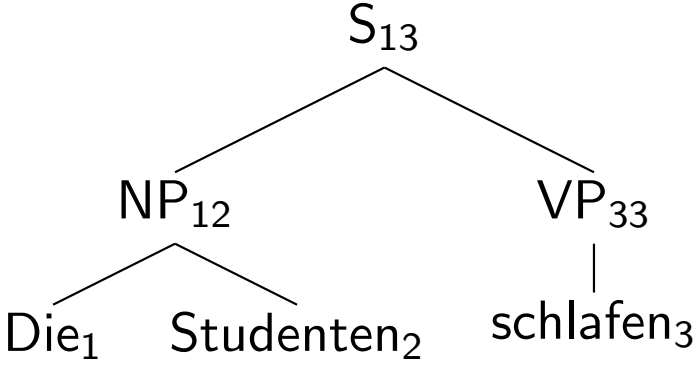
durch Anwendung von Kettenregel[1] und Unabhängigkeitsannahmen[2].

[1] Siehe Lieblang Skript Satz 2.1

[2] Siehe vorige Folie sowie Lieblang Skript Teil 3.4

Herleitung der Berechnung der Baumwahrscheinlichkeit

Wahrscheinlichkeit von Baumstrukur t für:



Notation

Subscripte $_{ab}$ an Knoten N bedeuten, dass N_{ab} Vorfahre der Worte mit Nummern a bis b ist.

Visuell weniger dichtes Konzept unserer Herleitung:

$$\begin{aligned} &= P(a,b,c) \\ &= P(a) * P(b|a) * P(c|a,b) \\ &= P(a) * P(b) * P(c) \end{aligned}$$

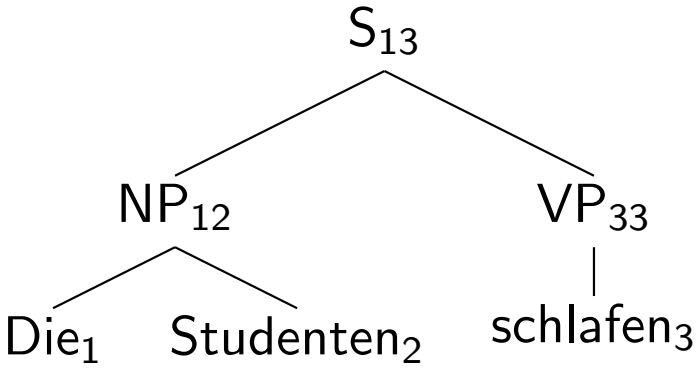
durch Anwendung von Kettenregel[1] und Unabhängigkeitsannahmen[2].

[1] Siehe Lieblang Skript Satz 2.1

[2] Siehe vorige Folie sowie Lieblang Skript Teil 3.4

Herleitung der Berechnung der Baumwahrscheinlichkeit

Wahrscheinlichkeit von Baumstrukur t für:



Notation

Subscripte $_{ab}$ an Knoten N bedeuten, dass N_{ab} Vorfahre der Worte mit Nummern a bis b ist.

$$
\begin{aligned}
&= P(S_{13} \rightarrow NP_{12}\,VP_{33}, NP_{12} \rightarrow Die_1\ Studenten_2, VP_{33} \rightarrow schlafen_3) \\
&= P(S_{13} \rightarrow NP_{12}\,VP_{33}) \times P(NP_{12} \rightarrow Die_1\ Studenten_2 | S_{13} \rightarrow NP_{12}\,VP_{33}) \\
&\quad \times P(VP_{33} \rightarrow schlafen_3 | S_{13} \rightarrow NP_{12}\,VP_{33},\ \ NP_{12} \rightarrow Die_1\ Studenten_2) \\
&= P(S_{13} \rightarrow NP_{12}\,VP_{33}) \times P(NP_{12} \rightarrow Die_1\ Studenten_2) \times P(VP_{33} \rightarrow schlafen_3) \\
&= P(S \rightarrow NP\ VP) \times P(NP \rightarrow Die\ Studenten) \times P(VP \rightarrow schlafen)
\end{aligned}
$$

durch Anwendung von Kettenregel[1] und Unabhängigkeitsannahmen[2].

[1] Siehe Lieblang Skript Satz 2.1

[2] Siehe vorige Folie sowie Lieblang Skript Teil 3.4

Implikationen der Unabhängigkeitsannahmen

- Vorteil: weniger Datenspärlichkeit durch Unabhängigkeitssannahmen.
- Nachteil: Wahrscheinlichkeiten rein strukturell, daher manchmal unintuitiv – Probleme mit PCFGs:
  - Kontext spielt keine Rolle, aber wir wissen, dass Personalpronomen in Subjektposition häufiger als in Objektposition sind (*"NP → Pronoun"* müsste zwei unterschiedliche, kontextabhängige Wahrscheinlichkeiten haben)
  - einfache PCFGs modellieren keine Subkategorisierung oder Selektionseinschränkungen
  - Globale strukturelle Präferenzen haben keine Auswirkung (z.B. in Bezug auf die Anbindung von PPs, Relativsätze, Adverbien, usw.)

Chomsky Normalform

Die Algorithmen gelten im Prinzip nur für Grammatiken in Chomsky-Normalform, d. h. Grammatiken, in denen

- alle Grammatikregeln binär sind,
  $N^i \rightarrow N^j\ N^k$
- und nur lexikalische Regeln unär
  $N^i \rightarrow w^n$

  (N: Nicht-Terminale, w: Wort)

Alle sonstige Grammatiken können aber in eine Grammatik in Chomsky-Normalform umgeformt werden.

Beispiel

$$A \rightarrow B\ C\ D \quad \Longrightarrow \quad \begin{array}{l} A \rightarrow B\ E \\ E \rightarrow C\ D \end{array}$$

Table of Contents

Zu beantwortenden Fragen

1. Was ist die Wahrscheinlichkeit eines Satzes gegeben eine Grammatik?
   → Inside- (und Outside-) Algorithmen
2. Was ist der wahrscheinlichste Parsebaum für einen Satz?
   → Viterbi-Algorithmus
3. Parameterschätzung: Wie ordnen wir den Regeln Wahrscheinlichkeiten zu?
   → Inside-Outside Algorithmus

Parallel mit HMMs

- Diese Algorithmen sind ähnlich den Algorithmen, die beim PoS-Tagging eingesetzt werden
- Inside- und Outside-Wahrscheinlichkeiten entsprechen Forward- und Backward-Wahrscheinlichkeiten

Algorithmen

Wie bei HMMs:

- Der naive Algorithmus, um die Wahrscheinlichkeit eines Satzes zu berechnen, ist exponential.
- Lösung:
  Teilergebnisse (Wahrscheinlichkeit einzelner Wortketten) speichern anstatt sie immer wieder neu zu berechnen!

Speichervariablen

$\beta_{N^i}(p, q)$: Wahrscheinlichkeit, dass das Nicht-Terminal N^i eine Wortkette mit Anfangspunkt p und Endpunkt q überspannt

Algorithmen

Wie bei HMMs:

- Der naive Algorithmus, um die Wahrscheinlichkeit eines Satzes zu berechnen, ist exponential.
- Lösung:
  Teilergebnisse (Wahrscheinlichkeit einzelner Wortketten) speichern anstatt sie immer wieder neu zu berechnen!

Speichervariablen

$\beta_{N^i}(p, q)$: Wahrscheinlichkeit, dass das Nicht-Terminal N^i eine Wortkette mit Anfangspunkt p und Endpunkt q überspannt

Der Inside-Algorithmus

Induktionsanfang

Die Wahrscheinlichkeit des Vor-Terminal-Baums mit Mutter N und Tochter p ist die Wahrscheinlichkeit der entsprechenden Regel:

$$\beta_N(p, p) = P(N \rightarrow p)$$

Induktionsschritt (Bottom-Up)

Die Wahrscheinlichkeit eines Subbaums ist die Summe über alle möglichen Regelanwendungen des Produktes von Regelwahrscheinlichkeit und Wahrscheinlichkeiten der jeweiligen Unterteile

$$\beta_N(p, r) = \sum_{Y,Z} \sum_{q} P(N \rightarrow Y\ Z)\ \beta_Y(p, q)\ \beta_Z(q{+}1, r)$$

Der Inside-Algorithmus

Induktionsanfang

Die Wahrscheinlichkeit des Vor-Terminal-Baums mit Mutter N und Tochter p ist die Wahrscheinlichkeit der entsprechenden Regel:

$$\beta_N(p, p) = P(N \rightarrow p)$$

Induktionsschritt (Bottom-Up)

Die Wahrscheinlichkeit eines Subbaums ist die Summe über alle möglichen Regelanwendungen des Produktes von Regelwahrscheinlichkeit und Wahrscheinlichkeiten der jeweiligen Unterteile

$$\beta_N(p, r) = \sum_{Y,Z} \sum_{q} P(N \rightarrow Y\ Z)\ \beta_Y(p, q)\ \beta_Z(q{+}1, r)$$

Inside-Algorithmus – Beispiel

Eingabesatz: Astronomers_1 saw_2 stars_3 with_4 ears_5

Induktionsanfang: $\beta_N(p,p) = P(N \rightarrow p)$

$$\begin{aligned}
\beta_{NP}(1,1) &= P(NP \rightarrow \text{astronomers}) = 0.1 \\
\beta_{V}\ (2,2) &= P(V \rightarrow \text{saw}) = 1.0 \\
\beta_{NP}(2,2) &= P(NP \rightarrow \text{saw}) = 0.04 \\
\beta_{NP}(3,3) &= P(NP \rightarrow \text{stars}) = 0.18 \\
\beta_{P}\ (4,4) &= P(P \rightarrow \text{with}) = 1.0 \\
\beta_{NP}(5,5) &= P(NP \rightarrow \text{ears}) = 0.18
\end{aligned}$$

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astronomers	0.1
NP	→ telescopes	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Inside-Algorithmus – Beispiel, Teil 2

Astronomers$_1$ saw$_2$ stars$_3$ with$_4$ ears$_5$

Induktionsschritt:

$\beta_N(p,r) = \sum_{Y,Z} \sum_q P(N \rightarrow Y\ Z)\beta_Y(p,q)\beta_Z(q{+}1,r)$

$$
\begin{aligned}
\beta_{VP}(2,3) &= P(VP \rightarrow V\ NP)\ \ \beta_V(2,2)\ \ \beta_{NP}(3,3) \\
&= 0.126 \\
\beta_{PP}(4,5) &= P(PP \rightarrow P\ NP)\ \ \beta_P(4,4)\ \ \beta_{NP}(5,5) \\
&= 0.18 \\
\beta_{NP}(3,5) &= P(NP \rightarrow NP\ PP)\ \beta_{NP}(3,3)\ \beta_{PP}(4,5) \\
&= 0.01296 \\
\beta_{VP}(2,5) &= P(VP \rightarrow V\ NP)\ \ \beta_V(2,2)\ \ \beta_{NP}(3,5) \\
&+ P(VP \rightarrow VP\ PP)\ \beta_{VP}(2,3)\ \beta_{PP}(4,5) \\
&= 0.009072 + 0.006804 = 0.015876 \\
\beta_S\ (1,5) &= P(S \rightarrow NP\ VP)\ \ \beta_{NP}(1,1)\ \beta_{VP}(2,5) \\
&= 0.0015876
\end{aligned}
$$

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astronomers	0.1
NP	→ telescopes	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Inside-Algorithmus – Beispiel

Zur besseren Übersichtlichkeit kann man das Ganze als Tabelle darstellen:

Induktionsanfang: $\beta_N(p,p) = P(N \rightarrow p)$

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_V = 1.0$ $\beta_{NP} = 0.04$			
3			$\beta_{NP} = 0.18$		
4				$\beta_P = 1.0$	
5					$\beta_{NP} = 0.18$
	astronomers	saw	stars	with	ears

Table: Tabellenfeld (p,q) zeigt die Inside Wahrscheinlichkeiten von $\beta_i(p,q)$.

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astronomers	0.1
NP	→ telescopes	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Inside-Algorithmus – Beispiel

Zur besseren Übersichtlichkeit kann man das Ganze als Tabelle darstellen:

Induktionsanfang: $\beta_N(p,p) = P(N \rightarrow p)$
Induktionsschritt:
$\beta_N(p,r) = \sum_{Y,Z} \sum_q P(N \rightarrow Y\ Z)\ \beta_Y(p,q)\ \beta_Z(q{+}1,r)$

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_V = 1.0$ $\beta_{NP} = 0.04$	$\beta_{VP} = 0.126$		
3			$\beta_{NP} = 0.18$		
4				$\beta_P = 1.0$	
5					$\beta_{NP} = 0.18$
	astronomers	saw	stars	with	ears

Table: Tabellenfeld (p,q) zeigt die Inside Wahrscheinlichkeiten von $\beta_i(p,q)$.

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astronomers	0.1
NP	→ telescopes	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Inside-Algorithmus – Beispiel

Zur besseren Übersichtlichkeit kann man das Ganze als Tabelle darstellen:

Induktionsanfang: $\beta_N(p,p) = P(N \rightarrow p)$
Induktionsschritt:
$\beta_N(p,r) = \sum_{Y,Z} \sum_q P(N \rightarrow Y\ Z)\ \beta_Y(p,q)\ \beta_Z(q{+}1,r)$

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_V = 1.0$ $\beta_{NP} = 0.04$	$\beta_{VP} = 0.126$		
3			$\beta_{NP} = 0.18$		
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	saw	stars	with	ears

Table: Tabellenfeld (p,q) zeigt die Inside Wahrscheinlichkeiten von $\beta_i(p,q)$.

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astronomers	0.1
NP	→ telescopes	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Inside-Algorithmus – Beispiel

Zur besseren Übersichtlichkeit kann man das Ganze als Tabelle darstellen:

Induktionsanfang: $\beta_N(p,p) = P(N \rightarrow p)$
Induktionsschritt:
$\beta_N(p,r) = \sum_{Y,Z} \sum_q P(N \rightarrow Y\ Z)\ \beta_Y(p,q)\ \beta_Z(q{+}1,r)$

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_V = 1.0$ $\beta_{NP} = 0.04$	$\beta_{VP} = 0.126$		
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	saw	stars	with	ears

Table: Tabellenfeld (p,q) zeigt die Inside Wahrscheinlichkeiten von $\beta_i(p,q)$.

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astron.	0.1
NP	→ telesc.	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Inside-Algorithmus – Beispiel

Zur besseren Übersichtlichkeit kann man das Ganze als Tabelle darstellen:

Induktionsanfang: $\beta_N(p,p) = P(N \rightarrow p)$
Induktionsschritt:
$\beta_N(p,r) = \sum_{Y,Z} \sum_q P(N \rightarrow Y\ Z)\ \beta_Y(p,q)\ \beta_Z(q{+}1,r)$

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_V = 1.0$ $\beta_{NP} = 0.04$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.015876$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	saw	stars	with	ears

Table: Tabellenfeld (p,q) zeigt die Inside Wahrscheinlichkeiten von $\beta_i(p,q)$.

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astron.	0.1
NP	→ telesc.	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Inside-Algorithmus – Beispiel

Zur besseren Übersichtlichkeit kann man das Ganze als Tabelle darstellen:

Induktionsanfang: $\beta_N(p,p) = P(N \rightarrow p)$
Induktionsschritt:
$\beta_N(p,r) = \sum_{Y,Z} \sum_q P(N \rightarrow Y\ Z)\ \beta_Y(p,q)\ \beta_Z(q{+}1,r)$

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				$\beta_S = 0.0015876$
2		$\beta_V = 1.0$ $\beta_{NP} = 0.04$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.015876$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	saw	stars	with	ears

Table: Tabellenfeld (p,q) zeigt die Inside Wahrscheinlichkeiten von $\beta_i(p,q)$.

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astron.	0.1
NP	→ telesc.	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Outside-Algorithmus

Inside-Algorithmus is bottom-up, Outside-Algorithmus ist top-down.

Induktionsanfang

Die Wahrscheinlichkeit des Wurzelsymbols N^1 mit Töchtern Y, Z ist 1 (falls es ein festgelegtes Startsymbol gibt):

$$\alpha_1(1, m) = 1; \qquad \alpha_j(1, m) = 0 \text{ für } j \neq 1$$

Induktionsschritt (Top-Down)

Die Outside-Wahrscheinlichkeit eines Knotens N im Baum ist die Summe über alle möglichen Regelanwendungen des Produktes von Outside-Wahrscheinlichkeit des Elternknotens, Regelwahrscheinlichkeit und Inside-Wahrscheinlichkeit des Geschwisterknotens.

Outside-Algorithmus

Inside-Algorithmus is bottom-up, Outside-Algorithmus ist top-down.

Induktionsanfang

Die Wahrscheinlichkeit des Wurzelsymbols N^1 mit Töchtern Y, Z ist 1 (falls es ein festgelegtes Startsymbol gibt):

$$\alpha_1(1, m) = 1; \qquad \alpha_j(1, m) = 0 \text{ für } j \neq 1$$

Induktionsschritt (Top-Down)

Die Outside-Wahrscheinlichkeit eines Knotens N im Baum ist die Summe über alle möglichen Regelanwendungen des Produktes von Outside-Wahrscheinlichkeit des Elternknotens, Regelwahrscheinlichkeit und Inside-Wahrscheinlichkeit des Geschwisterknotens.

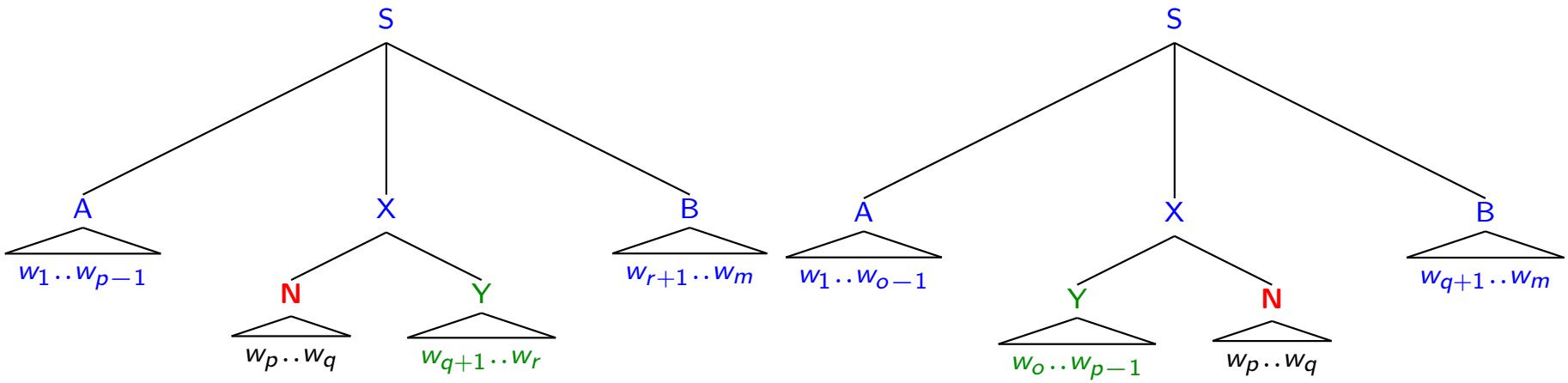
Outside-Algorithmus

Induktionsschritt (Top-Down)

Die Outside-Wahrscheinlichkeit eines Knotens N im Baum ist die Summe über alle möglichen Regelanwendungen des Produktes von Outside-Wahrscheinlichkeit des Elternknotens, Regelwahrscheinlichkeit und Inside-Wahrscheinlichkeit des Geschwisterknotens.

$$\alpha_N(p,q) = \sum_{X,Y} \sum_{r=q+1}^{m} \alpha_X(p,r) \;\; P(X \rightarrow N\ Y) \;\; \beta_Y(q+1,r)$$

$$+ \sum_{X,Y} \sum_{o=1}^{p-1} \alpha_X(o,q) \;\; P(X \rightarrow Y\ N) \;\; \beta_Y(o,p-1)$$



Algorithmen

Der Viterbi-Algorithmus

- Wie bei HMMs gibt es einen Inside-ähnlichen Algorithmus, der den wahrscheilichsten Parsebaum eines Satzes berechnet: **Viterbi**
- Es werden Speichervariablen eingeführt, die die Wahrscheinlichkeit des **besten** Unterbaums speichern: $\delta_j(p, q)$
- Außerdem werden Rückverfolgungsvariablen ψ eingesetzt, um sich zu merken, welche Regel zum besten Teilbaum führte:

  $\psi_N(p, q) = (Y, Z, r)$ speichert die Anwendung von $N \rightarrow Y\ Z$, wobei die Unterbäume nach r getrennt werden

Der Viterbi-Algorithmus

Induktionsanfang

Wie bei Inside: $\delta_N(p, p) = P(N \rightarrow p)$

Induktionsschritt

$$\delta_N(p, r) = \max_{Y,Z,q} P(N \rightarrow Y\ Z)\ \delta_Y(p, q)\ \delta_Z(q{+}1, r)$$

Falls es kein eindeutiges Maximum gibt, findet eine zufällige Auswahl statt.

Verfolgungsvariable speichern:

$$\psi_N(p, r) = \operatorname*{argmax}_{Y,Z,q} P(N \rightarrow Y\ Z)\ \delta_Y(p, q)\ \delta_Y(q{+}1, r)$$

Der Viterbi-Algorithmus

Die Wahrscheinlichkeit des wahrscheinlichsten Baumes ist somit

$$\max_{1 \leq i \leq n} \delta_i(1, n)$$

Zurückverfolgung des Pfades

Der wahrscheinlichste Baum kann wie folgt gefunden werden:

1. Die Wurzel des wahrscheinlichsten Baums ist $N^i_{1,n}$ (n: Länge der Wortkette)
2. Sei $\psi_N(p, r) = (Y, Z, q)$ die Verfolgungsvariable von N
3. Die linke und rechte Tochter von N sind:

$$\begin{aligned} \text{left}(N) &= Y_{p,q} \\ \text{right}(N) &= Z_{q+1,r} \end{aligned}$$

Viterbi-Algorithmus – Beispiel

Induktionsanfang

$$\begin{aligned}
\delta_{NP}(1,1) &= P(NP \rightarrow \text{astronomers}) = 0.1 \\
\delta_{V}\ (2,2) &= P(V \rightarrow \text{saw}) = 1.0 \\
\delta_{NP}(2,2) &= P(NP \rightarrow \text{saw}) = 0.04 \\
\delta_{NP}(3,3) &= P(NP \rightarrow \text{stars}) = 0.18 \\
\delta_{P}\ (4,4) &= P(P \rightarrow \text{with}) = 1.0 \\
\delta_{NP}(5,5) &= P(NP \rightarrow \text{ears}) = 0.18
\end{aligned}$$

Induktionsschritt

$$\begin{aligned}
\delta_{VP}\,(2,3) &= P(VP \rightarrow V\ NP)\ \delta_{V}(2,2)\ \delta_{NP}(3,3) \\
&= 0.126 \\
\psi_{VP}(2,3) &= (V, NP, 2)
\end{aligned}$$

Viterbi-Algorithmus – Beispiel, Teil 2

Induktionsschritt:

$$
\begin{aligned}
\delta_{VP}(2,3) &= P(VP \rightarrow V\ NP)\ \ \delta_V(2,2)\ \ \delta_{NP}(3,3) \\
&= 0.126 \\
\psi_{VP}(2,3) &= (V, NP, 2) \\
\delta_{PP}(4,5) &= P(PP \rightarrow P\ NP)\ \ \delta_P(4,4)\ \ \delta_{NP}(5,5) \\
&= 0.18 \\
\psi_{PP}(4,5) &= (P, NP, 4) \\
\delta_{NP}(3,5) &= P(NP \rightarrow NP\ PP)\ \delta_{NP}(3,3)\ \delta_{PP}(4,5) \\
&= 0.01296 \\
\psi_{NP}(3,5) &= (NP, PP, 3)
\end{aligned}
$$

Grammatik

S	→ NP VP	1.0
VP	→ V NP	0.7
VP	→ VP PP	0.3
NP	→ NP PP	0.4
PP	→ P NP	1.0
NP	→ astronomers	0.1
NP	→ telescopes	0.1
NP	→ saw	0.04
NP	→ stars	0.18
NP	→ ears	0.18
P	→ with	1.0
V	→ saw	1.0

Vergleich Tabellen Inside und Viterbi Algorithmus

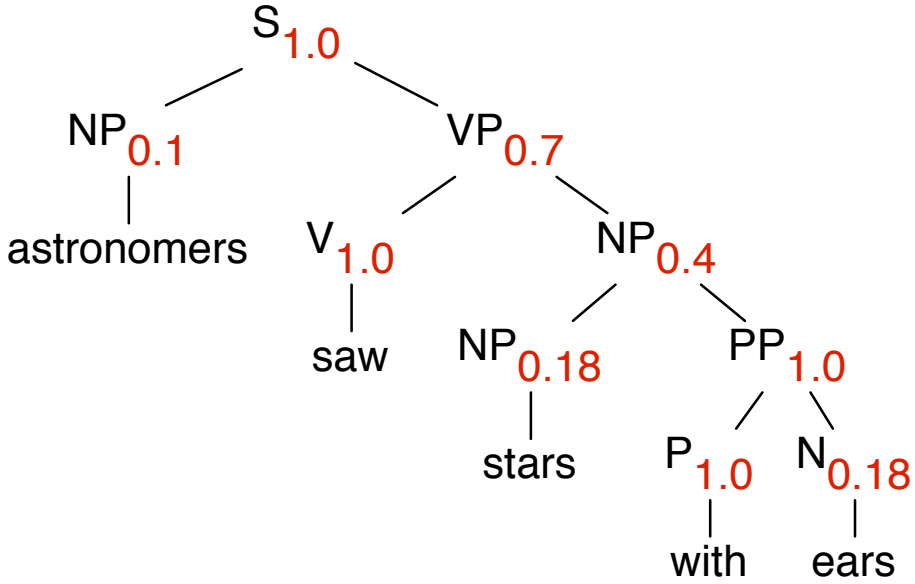
Inside	1	2	3	4	5
1	$\beta_{NP} = 0.1$				$\beta_S = 0.0015876$
2		$\beta_V = 1.0$ $\beta_{NP} = 0.04$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.015876$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	saw	stars	with	ears

Viterbi	1	2	3	4	5
1	$\delta_{NP} = 0.1$				$\delta_S = 0.0009072$
2		$\delta_V = 1.0$ $\delta_{NP} = 0.04$	$\delta_{VP} = 0.126$		$\delta_{VP} = 0.009072$
3			$\delta_{NP} = 0.18$		$\delta_{NP} = 0.01296$
4				$\delta_P = 1.0$	$\delta_{PP} = 0.18$
5					$\delta_{NP} = 0.18$
	astronomers	saw	stars	with	ears

Viterbi Beispiel

Zurückverfolgung des Pfades

$$
\begin{aligned}
\psi_S(1,5) &= (NP, VP, 1) \\
\text{left}(S) &= NP_{1,1} \\
\text{right}(S) &= VP_{(1+1),5} = VP_{2,5} \\
\psi_{VP}(2,5) &= (V, NP, 2) \\
\text{left}(VP) &= V_{2,2} \\
\text{right}(VP) &= NP_{3,5} \\
\psi_{NP}(3,5) &= (NP, PP, 3) \\
\text{left}(NP) &= NP_{3,3} \\
\text{right}(NP) &= PP_{4,5} \\
\psi_{PP}(4,5) &= (P, NP, 4) \\
\text{left}(PP) &= P_{4,4} \\
\text{right}(PP) &= NP_{5,5}
\end{aligned}
$$



Erwerb der Regelwahrscheinlichkeiten

Wie kommen wir and die Wahrscheinlichkeiten dran, die wir in unserer PCFG brauchen?Zwei Möglichkeiten:

1. Automatischer Erwerb:
   Parameter finden, bei denen ein gewisses Training-Korpus am wahrscheinlichsten ist
   - Erfordert gute Grammatik, um Anzahl der Kombinationen einzuschränken
   - Viele lokale Maxima $\rightarrow$ funktioniert selten gut
2. Externe Wahrscheinlichkeiten verwenden
   z. B. Wahrscheinlichkeiten aus einer Baumbank lernen (mit MLE: zählen, wie oft jede Regel vorkommt).
   - Schränkt Sprachen und Korpora ein, die verwendet werden können
   - Sparse-Data Probleme

Training PCFGs

- Mit dem Inside-Outside Algorithmus kann man die Wahrscheinlichkeit der Trainingsdaten auf effiziente Weise maximieren.
  1. Beginne mit zufälligen Wahrscheinlichkeiten. Berechne $P(w_{1m}|G)$.
  2. Finde heraus welche Regeln am häufigsten verwendet wurden.
  3. Erhöhe die Wahrscheinlichkeit dieser Regeln.
  4. Wiederhole das Ganze bis ein (lokales) Maximum erreicht wird.
- Der Inside-Outside Algorithmus ist eine Version des Expectation Maximization (EM) Algorithmus.

Training von PCFGs

PCFG Training ist in begrenztem Sinne Grammatikinduktion:

- Die Struktur der Grammatik ist schon vorhanden (d.h. eine Menge von Terminal- und Nichtterminalsymbolen).
- Wir können auch davon ausgehen, dass die Regeln gegeben sind, wir lernen nur die Wahrscheinlichkeiten.
- Alternativ können wir auch alle möglichen Regeln aus den Terminal- und Nichtterminalsymbolen generieren (z.B. alle binären Regeln).
- Während des Trainings lernen wir dann, dass einige Regeln nie auftauchen, so lernen wir welche Grammatikregeln möglich sind.

Zusammenfassung

PCFGs sind ein guter Weg, um statistische Information zu benutzen um mehrdeutige Sätze zu ranken.

- Einfache Erweiterung von normalen CFGs.
- Parameterschätzung durch Inside-Outside Algorithmus oder von Baumbank.
- Kann entweder als Sprachmodel (Wahrscheinlichkeit eines Satzes) oder als Parsingmodel (Wahrscheinlichkeit eines Baumes) benutzt werden.
- Effiziente Algorithmen zur Berechnung dieser Wahrscheinlichkeiten (Inside Algorithmus, Viterbi).
- In "naiver" Form nicht so erfolgreich, wegen Unabhängigkeitsannahmen. Verbesserung durch: Lexikalisierung und Kontextabhängigkeit von Regeln

Eine Grammatik lexikalisieren

Lexikalisierung der ersten VP-Regel, VP → V NP (0.7)

VP_{saw}	→	$V_{saw}\ NP_{astronomers}$	0.1
VP_{saw}	→	$V_{saw}\ NP_{ears}$	0.15
VP_{saw}	→	$V_{saw}\ NP_{saw}$	0.05
VP_{saw}	→	$V_{saw}\ NP_{stars}$	0.3
VP_{saw}	→	$V_{saw}\ NP_{telescopes}$	0.1

- Jetzt können wir abschätzen, ob "Sterne" oder "Sägen" bessere Objekte von "sehen" sind.
- Aber: aus einer Regel werden fünf! Gilt für alle Regeln mit Nichtterminalen ($S_{saw} \rightarrow NP_{astronomers}\ VP_{saw}$).
- Extrem viele Parameter müssen geschätzt werden, besonders bei realistischeren Lexika und Regelmengen (Sparse-Data)
- Typischerweise wird nur den lexikalischen Kopf als Bedingung genommen: $VP_{saw} \rightarrow V_{saw}$ NP 0.7

Struktureller Kontext

Strukturelle Unabhängigkeitsannahme

PCFGs nehmen an, dass eine Kategorie mit gleichen Wahrscheinlichkeiten zu anderen Kategorien expandiert, egal, wo in der Satzstruktur sie sich befindet (z.B. hat die Regel NP→NP PP nur eine Wahrscheinlichkeit).

Echte Daten

Regel	als Subj	als Obj
NP → PRP	13.7%	2.1%
NP → DT NN	5.6%	4.6%
NP → NP PP	5.6%	14.1%

- Um diese Verteilung zu erfassen, müssen wir Kontexte finden, auf denen wir konditionieren können, um entsprechende separate Wahrscheinlichkeiten zu testen.

Table of Contents

Evaluation von Parsern

Standard: PARSEVAL
Wir wollen die Ausgabe eines Parsers mit einem "Gold Standard" (z.B. annotierte Baumbank) vergleichen.

- Eine Konstituente ist **korrekt**, wenn es in der Baumbank für den Satz eine Konstituente mit gleichem Startpunkt, Endpunkt und Nichtterminalsymbol gibt.
- **labeled recall**:

$$\frac{\text{Anzahl korrekter Konstituenten im Parse}}{\text{Anzahl korrekter Konstituenten in der Baumbank}}$$

- **labeled precision**:

$$\frac{\text{Anzahl korrekter Konstituenten im Parse}}{\text{Gesamtanzahl aller Konstituenten im Parse}}$$

- **crossed brackets**: Wie viele Konstituenten haben die überkreuzt? z.B. ((A B) C) anstelle von (A (B C))

Vergleich einfache PCFG vs. PCFG mit mehr Kontext

Vergleich:

Parser	Labeled Recall	Labeled Precision
Standard PCFG	71.7	75.8
Lexicalized PCFG	83.4	84.1
Charniak (2000)	91.1	90.1

PCFGs

- Unabhängigkeitsannahmen werden der Wirklichkeit nicht gerecht
- Um PCFGs zu besseren Modellen zu machen kann man die Unabhängigkeitsannahmen aufweichen
  - Regeln lexikalisieren
  - strukturelle Information hinzufügen durch Konditionierung auf andere Knoten im Baum
- Evaluation von Parsern: Parseval, Labelled Precision, Labelled Recall