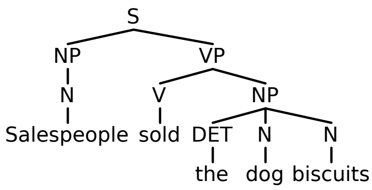
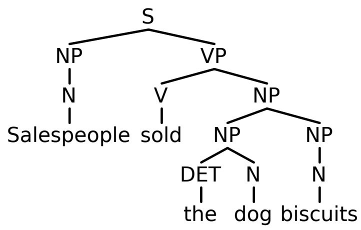
Programmierkurs Python II

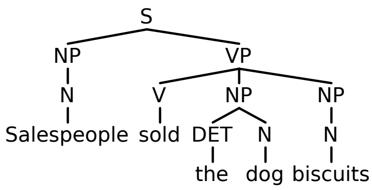
Michaela Regneri & Stefan Thater
FR 4.7 Allgemeine Linguistik (Computerlinguistik)
Universität des Saarlandes

Sommersemester 2010

Salespeople sold the dog biscuits







S → NP VP	NP → DET N N
VP → V NP	NP → N
VP → V NP NP	NP → NP NP
NP → DET N	

Ambiguität & Disambiguierung

- Probabilistische Disambiguierung
  wenn mehrere Ableitungsbäume für eine gegebene Eingabe möglich sind ⇒ wähle den wahrscheinlichsten
- Wir brauchen:
  - Probabilistisches Grammatikmodell
  - Methoden zur Ermittlung von Wahrscheinlichkeiten

(Weitere) Motivation

- Natürliche Sprache ist hochgradig ambig
  ⇒ Desambiguierung
- Grammatikentwicklung
  ⇒ Automatische Induktion von Grammatiken
- Effiziente Suche
  ⇒ wahrscheinlichste Hypothese(n) zuerst
- Robustheit

Probabilistische kontextfreie Grammatiken (PCFG)

- Probabilistische kontextfreie Grammatik (PCFG)
  - eine kontextfreie Grammatik $\langle V, \Sigma, R, S \rangle$
  - zusammen mit einer Funktion P, die jeder Regel einen Wert w $\in [0, 1]$ zuweist, so dass $\Sigma_{\beta \in V^*} P(A \to \beta) = 1$
- $P(A \to \beta)$ is die bedingte Wahrscheinlichkeit, dass in einer Ableitung das Symbol A zu β expandiert wird.
  - Alternative Notationen: $P(\beta \mid A)$, $P(A \to \beta \mid A)$, $A \to \beta$ [p]

Ableitungsbaum (Wdh.)

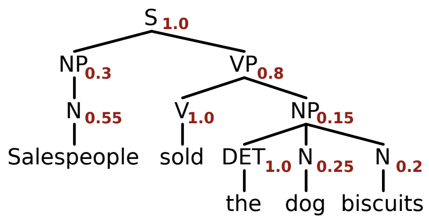
- Ableitungsbäume einer kontextfreien Grammatik G repräsentieren Ableitungen $S \Rightarrow^* w$:
  - Die Wurzel ist mit dem Startsymbol beschriftet
  - Blattknoten sind mit Terminalsymbolen oder ε beschriftet
  - Innere Knoten repräsentieren zusammen mit ihren Töchterknoten die bei der Ableitung verwendeten Regeln
- Parsing:
  Ermittlung eines oder mehrerer Ableitungsbäme
- Probabilistisches Parsing:
  Ermittlung des wahrscheinlichsten Ableitungsbaum

Probabilistische kontextfreie Grammatiken (PCFG)

- Eine PCFG G weist einem Satz w einen Ableitungsbaum T mit einer bestimmten Wahrscheinlichkeit zu.
- Die Wahrscheinlichkeit eines Ableitungsbaums T ist das Produkt der Wahrscheinlichkeit aller Regeln, die zum Aufbau des Ableitungsbaums T verwendet wurden:
  - $P(T, w) = P(T) = \prod_{n \in T} P(R(n))$
    - R(n) ist die Regel, mit der Knoten n expandiert wurde
  - Beachte: $P(T, w) = P(T)\, P(w \mid T) = P(T)$, da $P(w \mid T) = 1$
- Die Wahrscheinlichkeit eines Wortes (Satz) ist die Summe der Wahrscheinlichkeiten seiner Ableitungen:
  - Für $w \in L(G)$ ist $P(w) = \sum_T P(w, T)$

Salespeople sold the dog biscuits

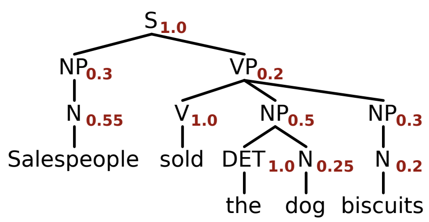
S → NP VP	[1.0]	DET → *the*	[1.0]
VP → V NP	[0.8]	N → *Salespeople*	[0.55]
VP → V NP NP	[0.2]	N → *dog*	[0.25]
NP → DET N	[0.5]	N → *biscuits*	[0.2]
NP → N	[0.3]	V → *sold*	[1.0]
NP → DET N N	[0.15]		
NP → NP NP	[0.05]		



$$P(t) = 1.0 \times 0.3 \times 0.55 \times 0.8 \times 1.0 \times 0.15 \times 1.0 \times 0.25 \times 0.2 = 9.9 \times 10^{-4}$$

Salespeople sold the dog biscuits

S → NP VP	[1.0]	DET → *the*	[1.0]
VP → V NP	[0.8]	N → *Salespeople*	[0.55]
VP → V NP NP	[0.2]	N → *dog*	[0.25]
NP → DET N	[0.5]	N → *biscuits*	[0.2]
NP → N	[0.3]	V → *sold*	[1.0]
NP → DET N N	[0.15]		
NP → NP NP	[0.05]		

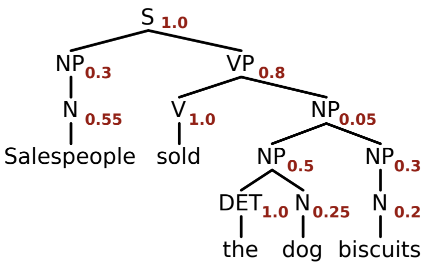


$$P(t) = 1.0 \times 0.3 \times 0.55 \times 0.2 \times 1.0 \times 0.5 \times 1.0 \times 0.25 \times 0.3 \times 0.2 = 2.475 \times 10^{-4}$$

Salespeople sold the dog biscuits

S → NP VP	[1.0]	DET → *the*	[1.0]
VP → V NP	[0.8]	N → *Salespeople*	[0.55]
VP → V NP NP	[0.2]	N → *dog*	[0.25]
NP → DET N	[0.5]	N → *biscuits*	[0.2]
NP → N	[0.3]	V → *sold*	[1.0]
NP → DET N N	[0.15]		
NP → NP NP	[0.05]		

$P(t) = 1.0 \times 0.3 \times 0.55 \times$
$0.8 \times 1.0 \times 0.05 \times$
$0.5 \times 1.0 \times 0.25 \times$
0.3×0.2
$= 4.95 \times 10^{-5}$



Probabilistische kontextfreie Grammatiken (PCFG)

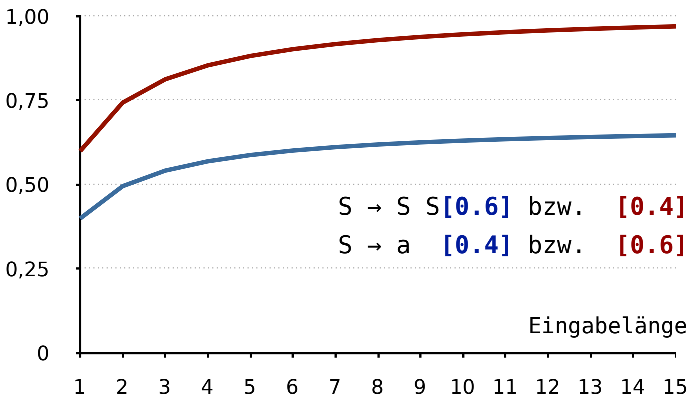
- Die Wahrscheinlichkeit eines Wortes (Satz) ist die Summe der Wahrscheinlichkeiten seiner Ableitungen:
  - Für $w \in L(G)$ ist $P(w) = \Sigma_T P(w, T)$
- Eine PCFG ist konsistent, wenn $\Sigma_{w \in L(G)} P(w) = 1$
- Rekursion kann zu inkonsistenten Grammatiken führen:
  - S → S S [0.6]
  - S → a [0.4]

Eine Inkonsistente PCFG

- S → S S [0.6] bzw. [0.4]
- S → a [0.4] bzw. [0.6]

- $P(a^i) = \#\text{Bäume}(a^i) \times 0.6^{i-1} \times 0.4^i = 0.4$
  - P(a) = 0.4, P(aa) = 0.096, P(aaa) = 0.0461, …
- $P(a^i) = \#\text{Bäume}(a^i) \times 0.4^{i-1} \times 0.6^i = 0.4$
  - P(a) = 0.6, P(aa) = 0.144, P(aaa) = 0.06912, …
- Anzahl Ableitungsbäme für a^{i+1} = *i*-te Catalanzahl

Eine Inkonsistente PCFG

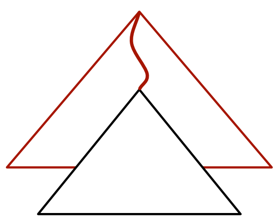


Probabilistisches Parsing

- Sprachmodelle („inside probabilities“)
  ermittle zu einer gegebenen Eingabekette w ∈ L(G) die Wahrscheinlichkeit, dass S ⇒* w
  - $P(w) = \sum_T P(w, T)$
- Probabilistisches Parsing („viterbi scores“)
  ermittle zu einer gegebenen Eingabekette w ∈ L(G) den wahrscheinlichsten Ableitungsbaums T(w).
  - $$\begin{aligned} T(w) &= \arg\max_T P(T \mid w) \\ &= \arg\max_T \frac{P(T, w)}{P(w)} \\ &= \arg\max_T P(T) \end{aligned}$$

Eigenschaften von PCFGen

- Die Wahrscheinlichkeit eines Teilbaums ist unabhängig vom Kontext in dem er vorkommt
- Die Wahrscheinlichkeit eines Teilbaums ist unabhängig von den Knoten, die ihn dominieren



Probabilistisches CYK-Parsing

- Bringe die Grammatik in Chomsky-Normalform
- Erweitere den CYK-Algorithmus:
  - Speichere zusätzlich Wahrscheinlichkeiten $T[i, j, A] \in [0, 1]$
- Wahrscheinlichkeite einer Kette
  - $T[i, j, A]$ = Summe der Wahrscheinlichkeiten aller Ableitungen $A \Rightarrow^* w_{i+1} \dots w_j$
- Wahrscheinlichkeit eines Ableitungsbaums
  - $T[i, j, A]$ = die Wahrscheinlichkeit der wahrscheinlichsten Ableitung $A \Rightarrow^* w_{i+1} \dots w_j$
  - $B[i, j, A]$ repräsentiert den Ableitungsbaum („Backpointer“)

CYK (ohne Wahrscheinlichkeiten)

```
CYK(w = w₁ ... wₙ):
    initialisiere T[i, j] = ∅ für alle 0 ≤ i < j ≤ n
    for i in 1 ... n:
        T[i-1, i] = { A | A → wᵢ }
    for j in 2 ... k:
        for i in j - 2 ... 0:
            for k in i + 1 ... j - 1:
                for all A → B C:
                    if B ∈ T[i, k] und C ∈ T[k, j] then
                        T[i, j] = T[i, j] ∪ {A}
    return S ∈ T[0, n]
```


CYK mit Wahrscheinlichkeiten (Erkenner)

```
PCYK(w = w₁ ... wₙ):
    initialisiere π[i, j, A] = 0 für alle 0 ≤ i < j ≤ n, A ∈ N
    for i in 1 ... n:
        π[i-1, i, A] = P(A → wᵢ)
    for j in 2 ... k:
        for i in j - 2 ... 0:
            for k in i + 1 ... j - 1:
                for all A → B C:
                    if π[i, k, B] > 0 und π[k, j, C] > 0 then
                        π[i, j, A] = π[i, j, A] +
                            P(A → B C) * π[i, k, B] * π[k, j, C]
    return π[0, n, S]
```

Beachte: Wahrscheinlichkeit eines Wortes = Summe der Wahrscheinlichkeiten seiner Ableitungen

CYK mit Wahrscheinlichkeiten (Parser)

```
PCYK(w = w1 ... wn):
    ...
    for j in 2 ... k:
        for i in j - 2 ... 0:
            for k in i + 1 ... j - 1:
                for all A → B C:
                    pr = π[i, k, B] * π[k, j, C] * P(A → B C)
                    if pr > π[i, j, A] then
                        π[i, j, A] = pr
                        B[i, j, A] = ⟨Backpointer aktualisieren⟩
    return ⟨besten Baum + Wahrscheinlickeit π[0, n, S]⟩
```


Regelwahrscheinlichkeiten

- Möglichkeiten zur Ermittlung („Abschätzung“) der Regelwahrscheinlichkeiten:
  - über relative Häufigkeiten aus mit syntaktisch annotierten Korpora („Baumbanken“)
  - ohne annotierte Daten durch Anwendung des Inside-Outside-Algorithmus

Regelwahrscheinlichkeiten

- Ausgangspunkt:
  syntaktisch annotiertes Korpus = Menge von Bäumen
- Extraktion einer Grammatik:
  Regeln = Menge der „Teilbäume“ der Tiefe 1
- Abschätzung der Regelahrscheinlichkeiten:
  - $P(A \rightarrow \alpha) = \dfrac{\mathtt{count}(A \rightarrow \alpha)}{\sum_\beta \mathtt{count}(A \rightarrow \beta)}$
  - count(A → α) gibt an, wie Häufig eine Regel in allen Bäumen verwendet wurde.

Regelwahrscheinlichkeiten

- Eine einfache Baumbank:
  - S1: [$_S$ [$_{NP}$ grass] [$_{VP}$ grows]]
  - S2: [$_S$ [$_{NP}$ grass] [$_{VP}$ grows] [$_{AP}$ fast]]
  - S3: [$_S$ [$_{NP}$ grass] [$_{VP}$ grows] [$_{AP}$ slowly]]
  - S4: [$_S$ [$_{NP}$ bananas] [$_{VP}$ grow]]
- Regeln & Regelwahrscheinlichkeiten:
  - S → NP VP 2/4
  - S → NP VP AP 2/4
  - NP → grass 3/4
  - ...
  -

Regelwahrscheinlichkeiten

Nr.	Regel	P(A → α)
r_1	S → NP VP	2/4
r_2	S → NP VP AP	2/4
r_3	NP → grass	3/4
r_4	NP → bananas	1/4
r_5	VP → grows	3/4
r_6	VP → grow	1/4
r_7	AP → fast	1/2
r_8	AP → slowly	1/2

Regelwahrscheinlichkeiten

- Wahrscheinlichkeiten der Beispielsätze:
  - $P(S1) = P(r_1) \cdot P(r_3) \cdot P(r_5) = 2/4 \cdot 3/4 \cdot 3/4 = 0.28125$
  - $P(S2) = P(r_2) \cdot P(r_3) \cdot P(r_5) \cdot P(r_7) = 0.140625$
  - $P(S3) = P(r_2) \cdot P(r_3) \cdot P(r_5) \cdot P(r_7) = 0.140625$
  - $P(S4) = P(r_1) \cdot P(r_4) \cdot P(r_6) = 0.03125$

Evaluation

- Coverage: Wieviele Sätze aus dem Testkorpus werden von der Grammatik als wohlgeformt erkannt?
  - Annahme: die Sätze des Korpus sind wohlgeformt
- Akkuratheit: Wieviele der laut Grammatik wahrscheinlichsten Bäume sind laut Testkorpus korrekt?
  - wird als „relative Korrektheit“ bzgl. Kategorie, Anfangs- und Endposition aller Konstituenten gemessen
  - Labelled precision: Anzahl korrekter Konstituenten im Ableitungsbaum relativ zu allen Konstituenten des Baums
  - Labelled recall: Anzahl korrekter Konstituenten im Ableitungsbaum relativ zu Referenzkorpus

Evaluation

- Labelled Precision:

$$\frac{\textit{\# korrekte Konstituenten im Ableitungsbaum für Eingabe}}{\textit{\# Konstituenten im Ableitungsbaum für Eingabe}}$$

- Labelled Recall:

$$\frac{\textit{\# korrekte Konstituenten im Ableitungsbaum für Eingabe}}{\textit{\# Konstituenten im Referenzkorpus für EIngabe}}$$

- Korrekt: die Konstituente hat die richtige Kategorie und überspannt den richtigen Teil der Eingabe.
  - Satzzeichen etc. werden üblicherweise ignoriert

Chomsky-Normalform

- Wenn bereits eine Grammatik vorliegt:
  - ersetze Regel $A \rightarrow A_1\ A_2\ A_3\ \ldots\ A_k$ [p] durch
  - $\langle A_1,\ldots,A_{k-1}\rangle \rightarrow A_1\ \ldots\ A_{k-1}$ [1.0]
  - $A \rightarrow \langle A_1,\ldots,A_{k-1}\rangle\ A_k$ [p]
- Wenn die Grammatik aus einer Baumbank extrahiert wird, kann man alternativ zuerst die Bäume der Baumbank binarisieren.

Literatur

- Jurafsky & Martin (2009) Speech and Language Processing Kapitel 14.
- Manning & Schütze (1999). Foundations of Statistical Natural Language Processing. Kapitel 11 & 12.
- Eugene Charniak (1993). Statistical Language Learning. Kapitel 5.