

content

- Viterbi for PCFG
- Inside-Outside-Algorithmus
- Berkeley-Parser

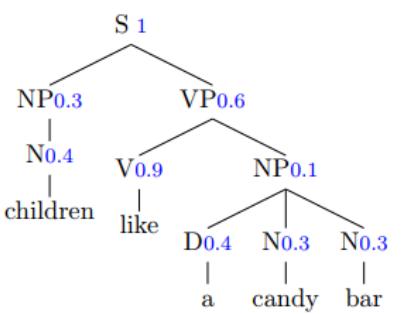
Viterbi for PCFG

What does the Viterbi algorithm compute and where and when do we use it?

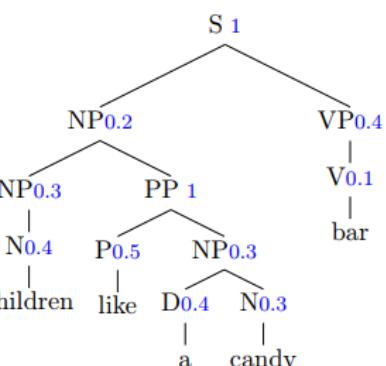
- we use it to find the best analysis (best parse tree) and the probability of it
- when the probability of each grammar rule is already given/computed.

Beispiele für Parsebaum-Wahrscheinlichkeiten

$S \rightarrow NP VP$	1
$NP \rightarrow NP PP$	0.2
$NP \rightarrow D N$	0.3
$NP \rightarrow N$	0.3
$NP \rightarrow D N N$	0.1
$NP \rightarrow N N$	0.1
$VP \rightarrow V NP$	0.6
$VP \rightarrow V$	0.4
$PP \rightarrow P NP$	1
$D \rightarrow \text{the}$	0.6
$D \rightarrow a$	0.4
$N \rightarrow \text{children}$	0.4
$N \rightarrow \text{candy}$	0.3
$N \rightarrow \text{bar}$	0.3
$V \rightarrow \text{bar}$	0.1
$V \rightarrow \text{like}$	0.9
$P \rightarrow \text{like}$	0.5
$P \rightarrow \text{for}$	0.5



$$p_1 = 0.0002333$$



$$p_2 = 0.0000173$$

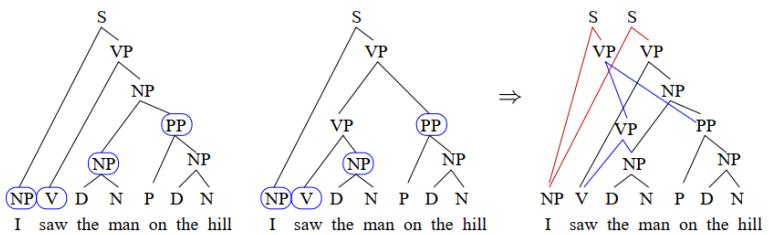
recap

- The initial problem that we want to solve is to find the best syntactic analysis of a sentence that can be parsed in more than one way by a symbolic syntactic parser (symbolische Parser)
- a symbolic parser (in this lecture) is a parser that uses some context-free grammar (without probability) to compute syntactic analyses of a sentence.
- In the example (left), we already have a PCFG (grammar rules with probabilities), so we can compute the probability of each tree for the sentence "children like a candy bar". The best tree is the one with the highest probability.
- we have learned how to estimate the prob for each rule (using Baumbank training or EM-training) and also how to compute the prob of a tree (by multiplying the prob of all nodes in the tree together)
- another way to determine the best tree is to use the **Viterbi algorithm**. To do that, we first have to merge both trees together to create a **Parsewald**.

Parsewald

Beispiel

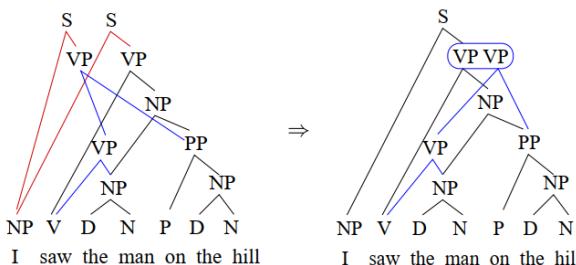
Zusammenfassen (maximaler) gemeinsamer Teilbäume



Die blauen Knoten der beiden Parsebäume werden jeweils zu einem Knoten zusammengefasst.

Beispiel

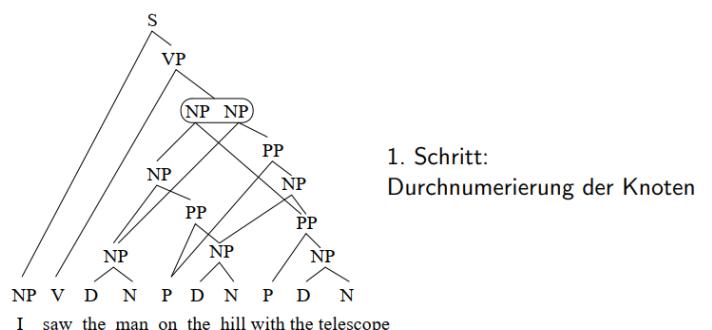
Zusammenfassen von Parsebäumen, die sich nur in einem Teilbaum unterscheiden



See the lecture video (Vorlesung 8) to see how to use Viterbi

Parsewälde als spezialisierte Grammatiken

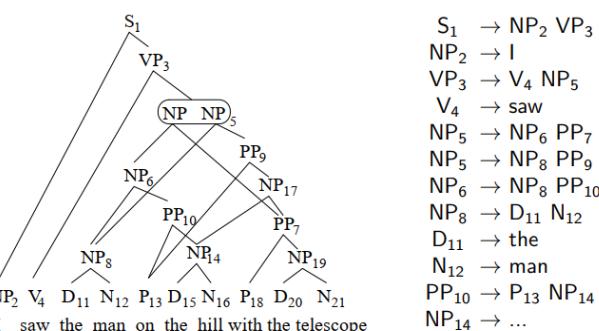
Wir definieren nun für jeden Parsewald eine Grammatik, die genau die darin enthaltenen Parsebäume generiert.



1. Schritt:
Durchnummerierung der Knoten

Parsewald als spezialisierte Grammatik

2. Schritt: Extraktion der Grammatikregeln



$S_1 \rightarrow NP_2 VP_3$
 $NP_2 \rightarrow I$
 $VP_3 \rightarrow V_4 NP_5$
 $V_4 \rightarrow \text{saw}$
 $NP_5 \rightarrow NP_6 PP_7$
 $NP_5 \rightarrow NP_8 PP_9$
 $NP_6 \rightarrow NP_8 PP_{10}$
 $NP_8 \rightarrow D_{11} N_{12}$
 $D_{11} \rightarrow \text{the}$
 $N_{12} \rightarrow \text{man}$
 $PP_{10} \rightarrow P_{13} NP_{14}$
 $NP_{14} \rightarrow \dots$

Diese Grammatik generiert genau die Parsebäume, die zu dem Parsewald zusammengefasst wurden (wenn man ignoriert, dass die Symbole noch Indizes tragen).

- Der **Viterbi**-Algorithmus berechnet die Wahrscheinlichkeit der **besten Analyse** jedes Parsewaldknotens

Berechnung der Viterbi-Wahrscheinlichkeiten

Viterbi-Algorithmus

$$\delta(a) = 1 \quad \text{für jedes Terminalsymbol } a$$

$$\delta(A \rightarrow X_1 \dots X_n) = p(A \rightarrow X_1 \dots X_n) \prod_{i=1}^n \delta(X_i) \quad \text{für Parsewald-Regeln}$$

$$\delta(A) = \max_{\alpha} \delta(A \rightarrow \alpha) \quad \text{für Nichtterminale A}$$

$$\psi(A) = \arg \max_{\alpha} \delta(A \rightarrow \alpha) \quad \text{beste Analyse von A}$$

$p(S_1 \rightarrow NP_2 VP_3)$ ist die Wahrscheinlichkeit $p(S \rightarrow NP VP)$ der entsprechenden PCFG-Regel.

Die ψ -Variable speichert die beste Parsewald-Regel für jedes Nichtterminal.

Viterbi-Beispiel

$$\delta(I) = 1$$

$$\delta(saw) = 1$$

...

$$\delta(NP_2) = \delta(NP_2 \rightarrow I)$$

$$= p(NP \rightarrow I) \delta(I)$$

$$\delta(V_4) = p(V \rightarrow saw) \delta(saw)$$

...

$$\delta(NP_8) = p(NP \rightarrow D N) \delta(D_{11}) \delta(N_{12})$$

...

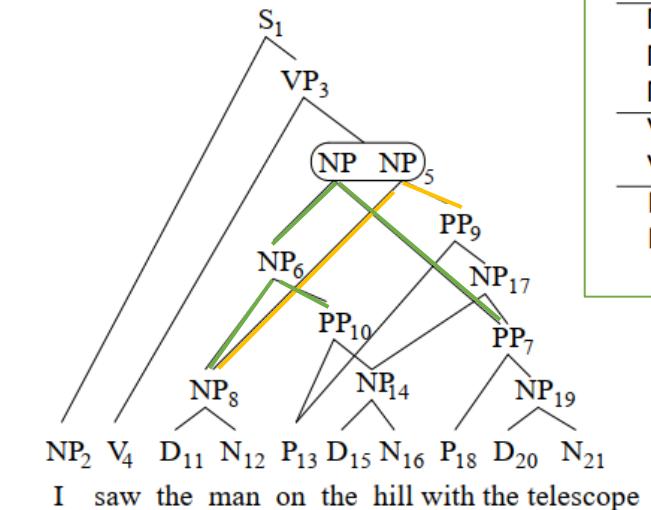
$$\delta(NP_6) = p(NP \rightarrow NP PP) \delta(NP_8) \delta(PP_{10})$$

...

$$\delta(NP_5) = \max(p(NP \rightarrow NP PP) \delta(NP_6) \delta(PP_7), p(NP \rightarrow NP PP) \delta(NP_8) \delta(PP_9))$$

...

$$\delta(S_1) = p(S \rightarrow NP VP) \delta(NP_2) \delta(VP_3)$$

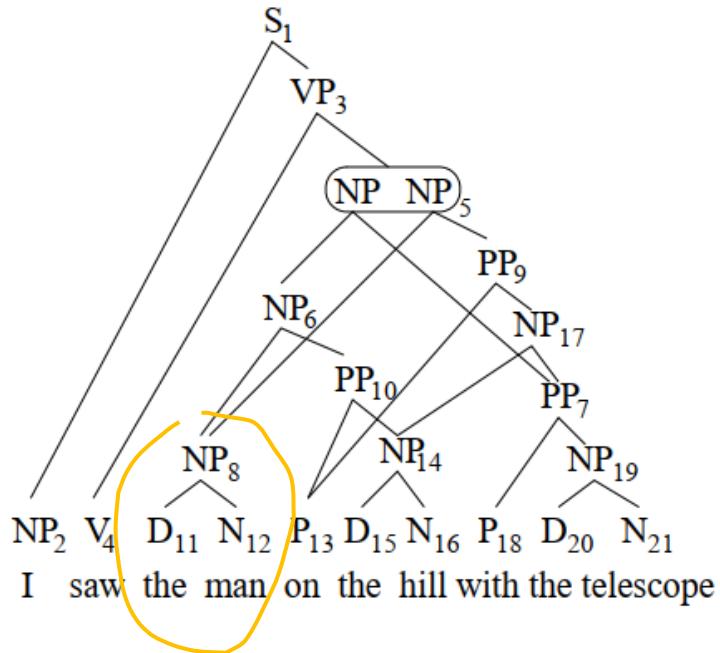


$S \rightarrow NP VP$	1
$NP \rightarrow NP PP$	0.2
$NP \rightarrow D N$	0.3
$NP \rightarrow N$	0.3
$NP \rightarrow D N N$	0.1
$NP \rightarrow N N$	0.1
$VP \rightarrow V NP$	0.6
$VP \rightarrow V$	0.4
$PP \rightarrow P NP$	1
$D \rightarrow \text{the}$	0.6
$D \rightarrow a$	0.4
$N \rightarrow \text{children}$	0.4
$N \rightarrow \text{candy}$	0.3
$N \rightarrow \text{bar}$	0.3
$V \rightarrow \text{bar}$	0.1
$V \rightarrow \text{like}$	0.9
$P \rightarrow \text{like}$	0.5
$P \rightarrow \text{for}$	0.5

Types of questions

- Berechnen Sie die Viterbi-Wk für Knoten 5 und 8
- Berechnen Sie die WK der besten Parse Baum (= compute vit prob for every node, and return vit(Start symbol))
- Explain how viterbi works
- For which purpose we do need viterbi?

Exercise1 : compute vit(NP8)



steps

1. compute vit(the), vit(man)
2. then vit(D11), vit(N12)
3. then vit(NP8)

grammar for this
Parsew ald

$S_1 \rightarrow NP_2 VP_3$
 $NP_2 \rightarrow I$
 $VP_3 \rightarrow V_4 NP_5$
 $V_4 \rightarrow \text{saw}$
 $NP_5 \rightarrow NP_6 PP_7$
 $NP_5 \rightarrow NP_8 PP_9$
 $NP_6 \rightarrow NP_8 PP_{10}$
 $NP_8 \rightarrow D_{11} N_{12}$
 $D_{11} \rightarrow \text{the}$
 $N_{12} \rightarrow \text{man}$
 $PP_{10} \rightarrow P_{13} NP_{14}$
 $NP_{14} \rightarrow \dots$

Viterbi-Algorithmus

- (1) $\delta(a) = 1$ für jedes Terminalsymbol a
- (3) $\delta(A \rightarrow X_1 \dots X_n) = p(A \rightarrow X_1 \dots X_n) \prod_{i=1}^n \delta(X_i)$ für Parsew ald-Regeln
- (2) $\delta(A) = \max_{\alpha} \delta(A \rightarrow \alpha)$ für Nichtterminale A
- $\psi(A) = \arg \max_{\alpha} \delta(A \rightarrow \alpha)$ beste Analyse von A

global grammar

$S \rightarrow NP VP$	1
$NP \rightarrow NP PP$	0.2
$NP \rightarrow D N$	0.3
$NP \rightarrow N$	0.3
$NP \rightarrow D N N$	0.1
$NP \rightarrow N N$	0.1
$VP \rightarrow V NP$	0.6
$VP \rightarrow V$	0.4
$PP \rightarrow P NP$	1
<u>$D \rightarrow \text{the}$</u>	0.6
<u>$D \rightarrow a$</u>	0.4
$N \rightarrow \text{children}$	0.4
$N \rightarrow \text{candv}$	0.3
$N \rightarrow \text{man}$	0.3
$V \rightarrow \text{bar}$	0.1
$V \rightarrow \text{like}$	0.9
$P \rightarrow \text{like}$	0.5
$P \rightarrow \text{for}$	0.5

explain

1. compute vit(the), vit(man) using (1)

$$\text{vit}(\text{the}) = 1$$

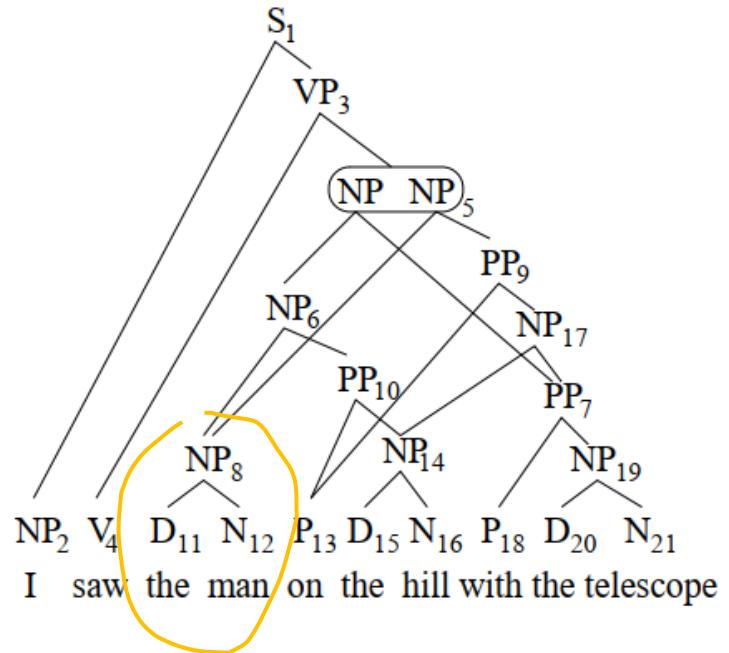
$$\text{vit}(\text{man}) = 1$$

2. then vit(D11), vit(N12) using (2)

for D11, look at the grammar for Parsew ald and find every rule that starts with D11, here we found only one ($D_{11} \rightarrow \text{the}$), so we have

$$\begin{aligned}
 \text{vit}(D_{11}) &= \max [\text{vit}(D_{11} \rightarrow \text{the})] \\
 &= \text{vit}(D_{11} \rightarrow \text{the}) \quad \# \text{ then use (3)} \\
 &= p(D \rightarrow \text{the}) \text{ vit}(\text{the}) \\
 &\quad \# \text{ look at the global grammar for } p(D \rightarrow \text{the}) \\
 &= 0.6 * 1 \\
 &= 0.6
 \end{aligned}$$

Exercise1 : compute vit(NP8)



steps

1. compute vit(the), vit(man)
2. then vit(D11), vit(N12)
3. then vit(NP8)

grammar for this
Parsewold

$$\begin{aligned}
 S_1 &\rightarrow NP_2 VP_3 \\
 NP_2 &\rightarrow I \\
 VP_3 &\rightarrow V_4 NP_5 \\
 V_4 &\rightarrow \text{saw} \\
 NP_5 &\rightarrow NP_6 PP_7 \\
 NP_5 &\rightarrow NP_8 PP_9 \\
 NP_6 &\rightarrow NP_8 PP_{10} \\
 NP_8 &\rightarrow D_{11} N_{12} \\
 D_{11} &\rightarrow \text{the} \\
 N_{12} &\rightarrow \text{man} \\
 PP_{10} &\rightarrow P_{13} NP_{14} \\
 NP_{14} &\rightarrow \dots
 \end{aligned}$$

explain (continue)

$$\begin{aligned}
 2. \text{ then } vit(D11), vit(N12) \text{ using (2)} \\
 vit(N12) &= \max(vit(N12 \rightarrow \text{man})) \\
 &= vit(N12 \rightarrow \text{man}) \\
 &= p(N \rightarrow \text{man}) vit(\text{man}) \\
 &= 0.3 * 1 \\
 &= 0.3
 \end{aligned}$$

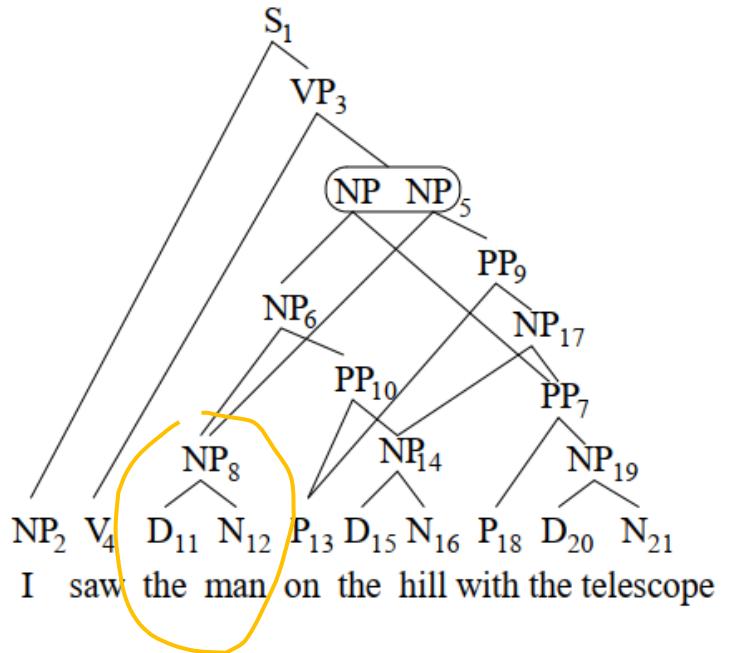
Viterbi-Algorithmus

$$\begin{aligned}
 (1) \quad \delta(a) &= 1 \quad \text{für jedes Terminalsymbol } a \\
 (3) \quad \delta(A \rightarrow X_1 \dots X_n) &= p(A \rightarrow X_1 \dots X_n) \prod_{i=1}^n \delta(X_i) \quad \text{für Parsewold-Regeln} \\
 (2) \quad \delta(A) &= \max_{\alpha} \delta(A \rightarrow \alpha) \quad \text{für Nichtterminale } A \\
 \psi(A) &= \arg \max_{\alpha} \delta(A \rightarrow \alpha) \quad \text{beste Analyse von } A
 \end{aligned}$$

global grammar

$S \rightarrow NP VP$	1
$NP \rightarrow NP PP$	0.2
$NP \rightarrow D N$	0.3
$NP \rightarrow N$	0.3
$NP \rightarrow D N N$	0.1
$NP \rightarrow N N$	0.1
$VP \rightarrow V NP$	0.6
$VP \rightarrow V$	0.4
$PP \rightarrow P NP$	1
$D \rightarrow \text{the}$	0.6
$D \rightarrow a$	0.4
$N \rightarrow \text{children}$	0.4
$N \rightarrow \text{candv}$	0.3
<u>$N \rightarrow \text{man}$</u>	<u>0.3</u>
$V \rightarrow \text{bar}$	0.1
$V \rightarrow \text{like}$	0.9
$P \rightarrow \text{like}$	0.5
$P \rightarrow \text{for}$	0.5

Exercise1 : compute vit(NP8)



steps

1. compute vit(the), vit(man)
2. then vit(D11), vit(N12)
3. then vit(NP8)

grammar for this
Parsewold

$S_1 \rightarrow NP_2 VP_3$
 $NP_2 \rightarrow I$
 $VP_3 \rightarrow V_4 NP_5$
 $V_4 \rightarrow \text{saw}$
 $NP_5 \rightarrow NP_6 PP_7$
 $NP_5 \rightarrow NP_8 PP_9$
 $NP_6 \rightarrow NP_8 PP_{10}$
 $NP_8 \rightarrow D_{11} N_{12}$
 $D_{11} \rightarrow \text{the}$
 $N_{12} \rightarrow \underline{\text{man}}$
 $PP_{10} \rightarrow P_{13} NP_{14}$
 $NP_{14} \rightarrow \dots$

explain (continue)

3. compute vit(NP8)

$$\begin{aligned}
 \text{vit}(NP8) &= \max [\text{vit}(NP8 \rightarrow D_{11} N_{12})] \\
 &= \text{vit}(NP8 \rightarrow D_{11} N_{12}) \\
 &= p(NP \rightarrow D N) \text{ vit}(D_{11}) \text{ vit}(N_{12}) \\
 &= 0.3 * 0.6 * 0.3
 \end{aligned}$$

finished

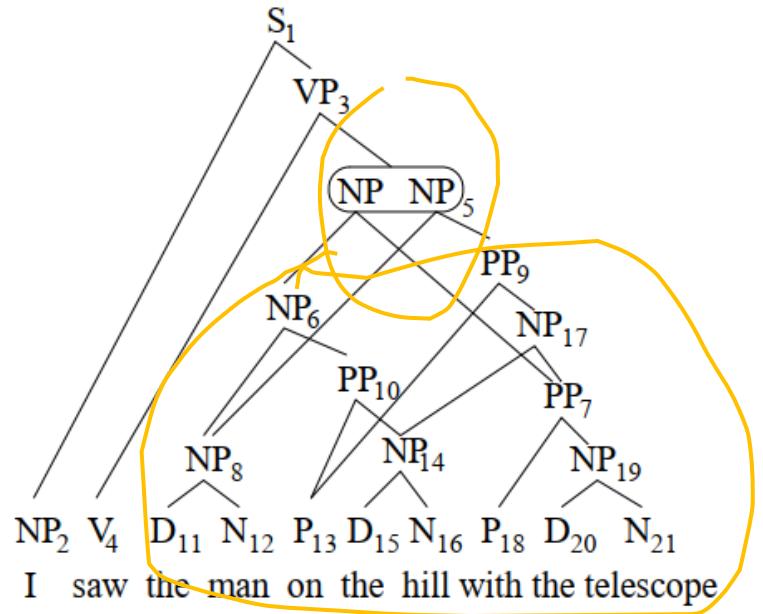
Viterbi-Algorithmus

- (1) $\delta(a) = 1$ für jedes Terminalsymbol a
- (3) $\delta(A \rightarrow X_1 \dots X_n) = p(A \rightarrow X_1 \dots X_n) \prod_{i=1}^n \delta(X_i)$ für Parsewold-Regeln
- (2) $\delta(A) = \max_{\alpha} \delta(A \rightarrow \alpha)$ für Nichtterminale A
- $\psi(A) = \arg \max_{\alpha} \delta(A \rightarrow \alpha)$ beste Analyse von A

global grammar

$S \rightarrow NP VP$	1
$NP \rightarrow NP PP$	0.2
$NP \rightarrow D N$	0.3
$NP \rightarrow N$	0.3
$NP \rightarrow D N N$	0.1
$NP \rightarrow N N$	0.1
$VP \rightarrow V NP$	0.6
$VP \rightarrow V$	0.4
$PP \rightarrow P NP$	1
$D \rightarrow \text{the}$	0.6
$D \rightarrow a$	0.4
$N \rightarrow \text{children}$	0.4
$N \rightarrow \text{candv}$	0.3
$N \rightarrow \text{man}$	0.3
$V \rightarrow \text{bar}$	0.1
$V \rightarrow \text{like}$	0.9
$P \rightarrow \text{like}$	0.5
$P \rightarrow \text{for}$	0.5

Exercise2 : compute vit(NP5)



steps (in the actual computation)

- compute vit prob of every node below NP5
- then compute vit(NP5)

In the exam, you may only have to write down the computation of vit(NP5) and do not need to actually compute it for other nodes below NP5 (see the answer example)

$S_1 \rightarrow NP_2 VP_3$
 $NP_2 \rightarrow I$
 $VP_3 \rightarrow V_4 NP_5$
 $V_4 \rightarrow \text{saw}$
 $NP_5 \rightarrow NP_6 PP_7$
 $NP_5 \rightarrow NP_8 PP_9$
 $NP_6 \rightarrow NP_8 PP_{10}$
 $NP_8 \rightarrow D_{11} N_{12}$
 $D_{11} \rightarrow \text{the}$
 $N_{12} \rightarrow \text{man}$
 $PP_{10} \rightarrow P_{13} NP_{14}$
 $NP_{14} \rightarrow \dots$

Viterbi-Algorithmus

- (1) $\delta(a) = 1$ für jedes Terminalsymbol a
- (3) $\delta(A \rightarrow X_1 \dots X_n) = p(A \rightarrow X_1 \dots X_n) \prod_{i=1}^n \delta(X_i)$ für Parsewold-Regeln
- (2) $\delta(A) = \max_{\alpha} \delta(A \rightarrow \alpha)$ für Nichtterminale A
- $\psi(A) = \arg \max_{\alpha} \delta(A \rightarrow \alpha)$ beste Analyse von A

$S \rightarrow NP VP$	1
$NP \rightarrow NP PP$	0.2
$NP \rightarrow D N$	0.3
$NP \rightarrow N$	0.3
$NP \rightarrow D N N$	0.1
$NP \rightarrow N N$	0.1
$VP \rightarrow V NP$	0.6
$VP \rightarrow V$	0.4
$PP \rightarrow P NP$	1
$D \rightarrow \text{the}$	0.6
$D \rightarrow a$	0.4
$N \rightarrow \text{children}$	0.4
$N \rightarrow \text{candv}$	0.3
$N \rightarrow \text{man}$	0.3
$V \rightarrow \text{bar}$	0.1
$V \rightarrow \text{like}$	0.9
$P \rightarrow \text{like}$	0.5
$P \rightarrow \text{for}$	0.5

Answer

$\text{vit}(NP5) = \max [\text{vit}(NP5 \rightarrow NP6 PP7), \text{vit}(NP5 \rightarrow NP8 PP9)]$ (2)
 where

$$\begin{aligned} \text{vit}(NP5 \rightarrow NP6 PP7) &= p(NP \rightarrow NP PP) \text{vit}(NP6) \text{vit}(PP7) \quad (3) \\ \text{vit}(NP5 \rightarrow NP8 PP9) &= p(NP \rightarrow NP PP) \text{vit}(NP8) \text{vit}(PP9) \end{aligned}$$

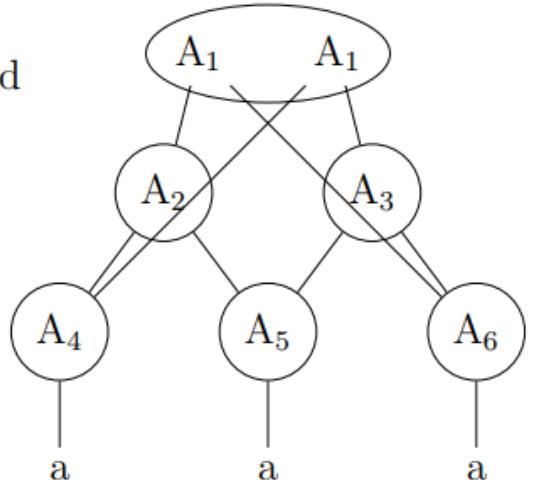
finished

Aufgabe 9) Wie berechnet der Viterbi-Algorithmus den besten Parse für einen gegebenen Parsewald? Wie werden die Viterbi-Wahrscheinlichkeiten berechnet (mit Formeln)?
(5 Punkte)

Inside Outside

Aufgabe 12) Gegeben sei eine PCFG mit zwei Regeln und den Wahrscheinlichkeiten $p(A \rightarrow a) = 1/4$ und $p(A \rightarrow A A) = 3/4$, wobei A das Startsymbol und a ein Terminalsymbol ist.

Betrachten Sie den Parsew ald



Berechnen Sie die Inside-Wahrscheinlichkeiten der Parsew ald-Nichtterminale A_1, \dots, A_6 .

Berechnen Sie dann ihre Outside-Wahrscheinlichkeiten.

Berechnen Sie zum Schluss die Aposteriori-Wahrscheinlichkeit (erwartete Häufigkeit) der Parsew aldregel $A_2 \rightarrow A_4 A_5$.

Tipp: Überlegen Sie, welche Knoten dieselben Inside-Wahrscheinlichkeiten besitzen.

Tipp 2: Berechnen Sie zur Kontrolle auch noch die Outside-Wahrscheinlichkeiten der Terminalknoten a . Diese müssen mit der Insidewahrscheinlichkeit von A_1 übereinstimmen.

Hilfsmittel: Liste der Zweierpotenzen:

1	2	3	4	5	6	7	8	9	10
2	4	8	16	32	64	128	256	512	1024

(6 Punkte)

Inside-Outside-Algorithmus

Der Inside-Outside-Algorithmus

Anwendung

- berechnet effizient die **erwarteten Regelhäufigkeiten** beim EM-Training
- und entspricht damit dem Forward-Backward-Algorithmus bei den HMMs.
- Er berechnet bottom-up **Inside**-Wahrscheinlichkeiten (ähnlich dem Viterbi-Algorithmus)
- und top-down **Outside**-Wahrscheinlichkeiten.
- Aus den Inside- und Outside-Wahrscheinlichkeiten berechnet er die **erwarteten Häufigkeiten**.

To use the Inside-Outside-Algorithm to compute "die erwartete Häufigkeit" $\gamma(A \rightarrow B)$, we first compute the Inside-WK for every node (1) and rule (2) of the given Parsewadl and grammar rule with prob. Then compute the Outside-WK. Then we can compute $\gamma(A \rightarrow B)$ for every rule $A \rightarrow B$ in the Parsewadl. Do it for all sentences in the training set, then from $\gamma(A \rightarrow B)$, we can compute "die erwartete Regelhäufigkeiten $f(A \rightarrow B)$ " that we can use in EM-Training.

Inside-Wahrscheinlichkeiten

- Der **Viterbi**-Algorithmus berechnet die Wahrscheinlichkeit der **besten Analyse** jedes Parsewadlknotens
- Der **Inside**-Algorithmus berechnet die Gesamtwahrscheinlichkeit **aller Analysen** für jeden Parsewadlknoten.

Inside-Algorithmus

$$(1) \alpha(a) = 1 \text{ für Terminalsymbol } a$$
$$(2) \alpha(A \rightarrow X_1 \dots X_n) = p(A \rightarrow X_1 \dots X_n) \prod_{i=1}^n \alpha(X_i) \text{ für Parsewadlregeln}$$
$$(1) \alpha(A) = \sum_{A \rightarrow \gamma} \alpha(A \rightarrow \gamma) \text{ für Nichtterminale A}$$

Der Unterschied zum Viterbi-Algorithmus ist die Summen-Operation statt der max-Operation.

Old method (inefficient)

Training auf unannotierten Daten

Ansatz 3: (Forts.)

EM-Training

- ① Initialisierung der Regelwahrscheinlichkeiten
 - ② für alle Sätze (E-Schritt)
 - ▶ Berechnung der Parsebäume für den Satz
 - ▶ Berechnung der Parsebaumgewichte $p(t|s)$
 - ▶ Extraktion der gewichteten Regelhäufigkeiten
 - ③ Neuschätzung der Regelwahrscheinlichkeiten (M-Schritt)
 - ④ Weiter mit Schritt 2 bis ein Endekriterium erfüllt ist
- compute f
- compute p

Regel	p_0	f_1	p_1	f_2	p_2	f_3	p_3
$S \rightarrow NP VP$	1.00	2.00	1.00	2.00	1.00	2.00	1.00
$NP \rightarrow D N$	0.25	0.50	0.11	0.10	0.02	0.00	0.00
$NP \rightarrow D N N$	0.25	0.50	0.11	0.90	0.22	1.00	0.25
$NP \rightarrow N$	0.25	3.00	0.67	3.00	0.73	3.00	0.75
$NP \rightarrow NP PP$	0.25	0.50	0.11	0.10	0.02	0.00	0.00
$VP \rightarrow V$	0.50	0.50	0.25	0.10	0.05	0.00	0.00
$VP \rightarrow V NP$	0.50	1.50	0.75	1.90	0.95	2.00	1.00
$PP \rightarrow P NP$	1.00	0.50	1.00	0.10	1.00	0.00	1.00
$D \rightarrow a$	0.50	1.00	1.00	1.00	1.00	1.00	1.00
$D \rightarrow the$	0.50	0.00	0.00	0.00	0.00	0.00	0.00
$N \rightarrow bar$	0.25	0.50	0.11	0.90	0.18	1.00	0.20
$N \rightarrow candy$	0.25	1.00	0.22	1.00	0.20	1.00	0.20
$N \rightarrow children$	0.25	2.00	0.44	2.00	0.41	2.00	0.40
$N \rightarrow chocolate$	0.25	1.00	0.22	1.00	0.20	1.00	0.20
$V \rightarrow bar$	0.50	0.50	0.25	0.10	0.05	0.00	0.00
$V \rightarrow like$	0.50	1.50	0.75	1.90	0.95	2.00	1.00
$P \rightarrow like$	1.00	0.50	1.00	0.10	1.00	0.00	1.00
-logprob		15.2		11.3		9.3	

Old method (inefficient)

Training auf unannotierten Daten

Ansatz 3: (Forts.)

EM-Training

- ① Initialisierung der Regelwahrscheinlichkeiten
- ② für alle Sätze (E-Schritt)

- ▶ Berechnung der Parsebäume für den Satz
- ▶ Berechnung der Parsebaumgewichte $p(t|s)$
- ▶ Extraktion der gewichteten Regelhäufigkeiten

compute f

- ③ Neuschätzung der Regelwahrscheinlichkeiten (M-Schritt) compute p
- ④ Weiter mit Schritt 2 bis ein Endekriterium erfüllt ist

More efficient with Inside-outside-Algorithm

Neuschätzung der Regel-Wahrscheinlichkeiten

EM-Algorithmus: wiederholte Ausführung der beiden Schritte

① E-Schritt compute f

- ▶ Jeder Satz des Trainingskorpus wird geparsst und ein Parsewald erstellt.
- ▶ Die erwartete Häufigkeit $\gamma(A \rightarrow \delta)$ jeder Parsewaldregel $A \rightarrow \delta$ wird berechnet.
- ▶ Die erwarteten Häufigkeiten $\gamma(A \rightarrow \delta)$ werden für jede CFG-Regel über alle Trainingssätze zu $f(A' \rightarrow \delta')$ summiert, wobei sich A' und δ' aus A und δ durch Entfernen der Knotennummern ergeben.

② M-Schritt

Die Regel-Wahrscheinlichkeiten werden aus den erwarteten Häufigkeiten neu geschätzt:

$$\text{compute p} \quad p(A \rightarrow \delta) = \frac{f(A \rightarrow \delta)}{\sum_{\delta'} f(A \rightarrow \delta')}$$

Beispiel

Neuschätzung der Regel-Wahrscheinlichkeiten

EM-Algorithmus: wiederholte Ausführung der beiden Schritte

① E-Schritt compute f

+ initialize p uniformly (1)

- Jeder Satz des Trainingskorpus wird geparsst und ein Parsewälde erstellt.
- Die erwartete Häufigkeit $\gamma(A \rightarrow \delta)$ jeder Parsewälde-Regel $A \rightarrow \delta$ wird berechnet. (4)
- Die erwarteten Häufigkeiten $\gamma(A \rightarrow \delta)$ werden für jede CFG-Regel über alle Trainingssätze zu $f(A' \rightarrow \delta')$ summiert, wobei sich A' und δ' aus A und δ durch Entfernen der Knotennummern ergeben.

② M-Schritt

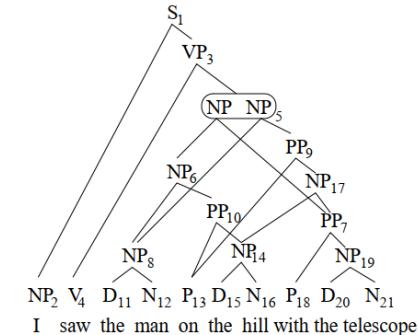
Die Regel-Wahrscheinlichkeiten werden aus den erwarteten Häufigkeiten neu geschätzt:

$$(3) \quad p(A \rightarrow \delta) = \frac{f(A \rightarrow \delta)}{\sum_{\delta'} f(A \rightarrow \delta')} \quad (2)$$

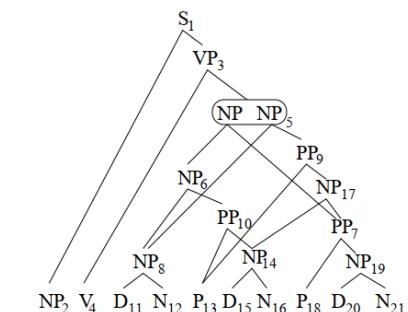
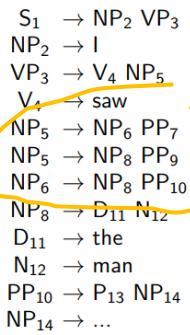
Suppose we have 2 sentences in our training set, want to compute $f(A \rightarrow \delta)$ (2) so that we can compute $p(A \rightarrow \delta)$ (3) and perform the EM-Training. Create a Parsewälde for every sentence to merge all possible analyses of the sentence together. Then compute the Inside and Outside WK for all Parsewälde. Using the Inside and Outside Wk, we can compute $\gamma(A \rightarrow \delta)$ (4) for every rule in each Parsewälde. Then we sum up $\gamma(A \rightarrow \delta)$ from all Parsewälde that has the same underlying rule together. The result is $f(A \rightarrow \delta)$ (2). For example, to compute $f(NP \rightarrow NP \ PP)$ we look in Parsewälde1 for all rules of this form, sum the gamma value of them. Then look at the Parsewälde2, do the same. Sum the results together.

$$f(NP \rightarrow NP \ PP) = \gamma(NP5 \rightarrow NP6 \ PP7) + \gamma(NP5 \rightarrow NP8 \ PP9) + \gamma(NP6 \rightarrow NP8 \ PP9) + \text{from Parsewälde1}$$

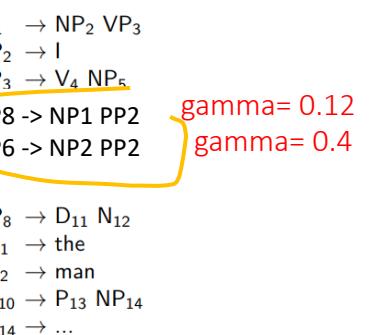
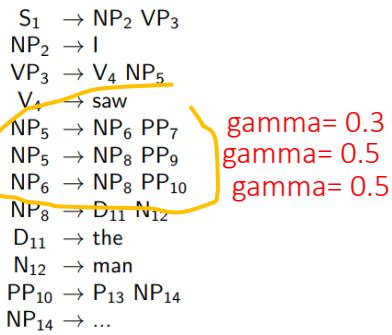
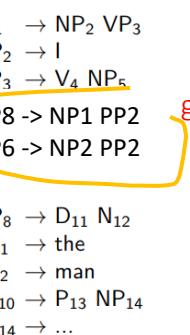
$$\gamma(NP8 \rightarrow NP1 \ PP2) + \gamma(NP6 \rightarrow NP2 \ PP2) \quad \text{from Parsewälde2}$$



sentence1



sentence2



(1) (2) (3)

Regel	p_0	f_1	p_1
$S \rightarrow NP \ VP$	1.00	2.00	1.00
$NP \rightarrow D \ N$	0.25	0.50	0.11
$NP \rightarrow D \ N \ N$	0.25	0.50	0.11
$NP \rightarrow N$	0.25	3.00	0.67
$NP \rightarrow NP \ PP$	0.25	0.50	0.11
$VP \rightarrow V$	0.50	0.50	0.25
$VP \rightarrow V \ NP$	0.50	1.50	0.75
$PP \rightarrow P \ NP$	1.00	0.50	1.00
$D \rightarrow a$	0.50	1.00	1.00
$D \rightarrow the$	0.50	0.00	0.00

Inside-Wahrscheinlichkeiten

- Der **Viterbi**-Algorithmus berechnet die Wahrscheinlichkeit der **besten Analyse** jedes Parsewaldknotens
- Der **Inside**-Algorithmus berechnet die Gesamtwahrscheinlichkeit **aller Analysen** für jeden Parsewaldknoten.

Inside-Algorithmus

$$\alpha(a) = 1 \quad \text{für Terminalsymbol } a$$

$$\alpha(A \rightarrow X_1 \dots X_n) = p(A \rightarrow X_1 \dots X_n) \prod_{i=1}^n \alpha(X_i) \quad \text{für Parsewaldregeln}$$

$$\alpha(A) = \sum_{A \rightarrow \gamma} \alpha(A \rightarrow \gamma) \quad \text{für Nichtterminale } A$$

Der Unterschied zum Viterbi-Algorithmus ist die Summen-Operation statt der max-Operation.

Berechnung der Outside-Wahrscheinlichkeiten

Outside-Algorithmus

$$\beta(S) = 1 \quad \text{für Startsymbol } S$$

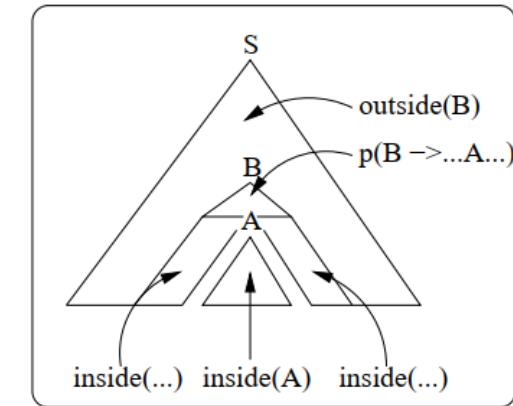
$$\beta(A) = \sum_{B \rightarrow \gamma A \delta} \beta(B \rightarrow \gamma A \delta)$$

$$\beta(B \rightarrow X_1 \dots X_m \underline{A} X_{m+1} \dots X_n) = \beta(B) p(B \rightarrow X_1 \dots X_m A X_{m+1} \dots X_n) \prod_{i=1}^n \alpha(X_i)$$

$$\beta(B \rightarrow \gamma \underline{A} \delta) = \beta(B) \frac{\alpha(B \rightarrow \gamma A \delta)}{\alpha(A)} \quad \text{mit } \gamma = X_1 \dots X_m \text{ und } \delta = X_{m+1}, \dots, X_n$$

$\beta(B \rightarrow \gamma \underline{A} \delta)$: Beitrag der Regel $B \rightarrow \gamma A \delta$ zur Outside-Wahrscheinlichkeit von A

Beachte, dass $p(B \rightarrow X_1 \dots X_m A X_{m+1} \dots X_n) \prod_{i=1}^n \alpha(X_i) = \alpha(B \rightarrow X_1 \dots X_m A X_{m+1} \dots X_n) / \alpha(A)$



Outside-Algorithmus

$$\beta(S) = 1 \quad \text{für Startsymbol } S$$

$$\beta(A) = \sum_{B \rightarrow \gamma A \delta} \beta(B \rightarrow \gamma \underline{A} \delta)$$

p

$S \rightarrow NP VP$	1
$NP \rightarrow NP PP$	0.2
$NP \rightarrow D N$	0.3
$NP \rightarrow N$	0.3
$NP \rightarrow D N N$	0.1
$NP \rightarrow N N$	0.1
$VP \rightarrow V NP$	0.6
$VP \rightarrow V$	0.4
$PP \rightarrow P NP$	1
$D \rightarrow \text{the}$	0.6
$D \rightarrow a$	0.4
$N \rightarrow \text{children}$	0.4
$N \rightarrow \text{candy}$	0.3
$N \rightarrow \text{bar}$	0.3
$V \rightarrow \text{bar}$	0.1
$V \rightarrow \text{like}$	0.9
$P \rightarrow \text{like}$	0.5
$P \rightarrow \text{for}$	0.5

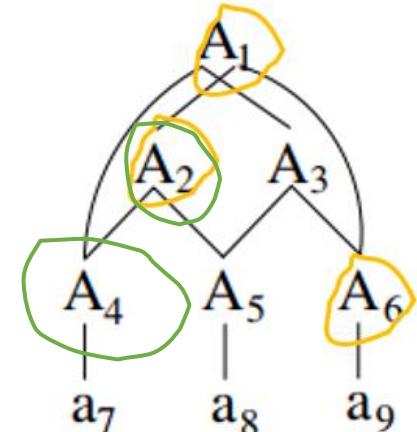
$$\begin{aligned} b(A_2) &= b(A_1 \rightarrow A_2 A_3) \\ &= b(A_1) p(A \rightarrow A A) a(A_3) \end{aligned}$$

$$b(A_4) = \text{Sum } (b(A_1 \rightarrow A_4 A_3), b(A_2 \rightarrow A_4 A_5))$$

where

$$b(A_1 \rightarrow A_4 A_3) = b(A_1) p(A \rightarrow A A) a(A_3) \# \text{ method1}$$

$$b(A_2 \rightarrow A_4 A_5) = b(A_2) p(A \rightarrow A A) a(A_5)$$



Methode 1)

$$\beta(B \rightarrow \gamma \underline{A} \delta) = \beta(B) \frac{\alpha(B \rightarrow \gamma A \delta)}{\alpha(A)}$$

mit $\gamma = X_1 \dots X_m$ und $\delta = X_{m+1}, \dots, X_n$

Methode 2)

$$\beta(B \rightarrow X_1 \dots X_m \underline{A} X_{m+1} \dots X_n) = \beta(B) p(B \rightarrow X_1 \dots X_m A X_{m+1} \dots X_n) \prod_{i=1}^n \alpha(X_i)$$

inside WK

Inside-Outside-Beispiel

PCFG:

$A \rightarrow A A$	0.6
$A \rightarrow a$	0.4

$$\alpha(a_7) = 1 = \alpha(a_8) = \alpha(a_9)$$

$$\alpha(A_4) = p(A \rightarrow a) \alpha(a_7) = 0.4 * 1 = 0.4 = \alpha(A_5) = \alpha(A_6)$$

$$\alpha(A_2) = p(A \rightarrow A A) \alpha(A_4) \alpha(A_5) = 0.6 * 0.4 * 0.4 = 0.096$$

$$\alpha(A_3) = \alpha(A_2)$$

$$\begin{aligned} \alpha(A_1) &= p(A \rightarrow A A) \alpha(A_4) \alpha(A_3) + p(A \rightarrow A A) \alpha(A_2) \alpha(A_6) \\ &= 0.6 * 0.4 * 0.096 * 2 = 0.04608 \end{aligned}$$

$$\beta(A_1) = 1$$

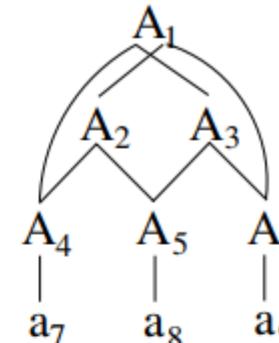
$$\beta(A_2) = \beta(A_1) p(A \rightarrow A A) \alpha(A_6) = 1 * 0.6 * 0.4 = 0.24 = \beta(A_3)$$

$$\begin{aligned} \beta(A_4) &= \beta(A_1) p(A \rightarrow A A) \alpha(A_3) + \beta(A_2) p(A \rightarrow A A) \alpha(A_5) \\ &= 1 * 0.6 * 0.096 + 0.24 * 0.6 * 0.4 = 0.1152 = \beta(A_6) \end{aligned}$$

$$\begin{aligned} \beta(A_5) &= \beta(A_2) p(A \rightarrow A A) \alpha(A_4) + \beta(A_3) p(A \rightarrow A A) \alpha(A_6) \\ &= 0.24 * 0.6 * 0.4 * 2 = 0.1152 \end{aligned}$$

$$\beta(a_7) = \beta(A_4) p(A \rightarrow a) = 0.1152 * 0.4 = 0.04608$$

$$\begin{aligned} \gamma(A_2 \rightarrow A_4 A_5) &= \beta(A_2) p(A \rightarrow A A) \alpha(A_4) \alpha(A_5) / \alpha(A_1) \\ &= 0.24 * 0.6 * 0.4 * 0.4 / 0.04608 = 0.5 \end{aligned}$$

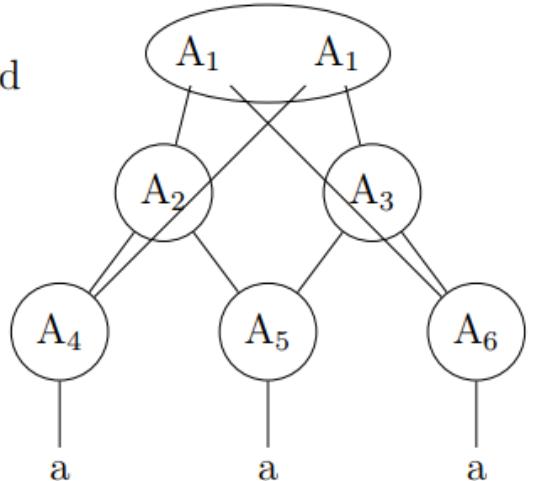


- erwartete Regelhäufigkeit

$$\gamma(A \rightarrow \delta) = \alpha(A \rightarrow \delta) \beta(A) / \alpha(S)$$

Aufgabe 12) Gegeben sei eine PCFG mit zwei Regeln und den Wahrscheinlichkeiten $p(A \rightarrow a) = 1/4$ und $p(A \rightarrow A A) = 3/4$, wobei A das Startsymbol und a ein Terminalsymbol ist.

Betrachten Sie den Parsew ald



Berechnen Sie die Inside-Wahrscheinlichkeiten der Parsew ald-Nichtterminale A_1, \dots, A_6 .

Berechnen Sie dann ihre Outside-Wahrscheinlichkeiten.

Berechnen Sie zum Schluss die Aposteriori-Wahrscheinlichkeit (erwartete Häufigkeit) der Parsew aldregel $A_2 \rightarrow A_4 A_5$.

Tipp: Überlegen Sie, welche Knoten dieselben Inside-Wahrscheinlichkeiten besitzen.

Tipp 2: Berechnen Sie zur Kontrolle auch noch die Outside-Wahrscheinlichkeiten der Terminalknoten a . Diese müssen mit der Insidewahrscheinlichkeit von A_1 übereinstimmen.

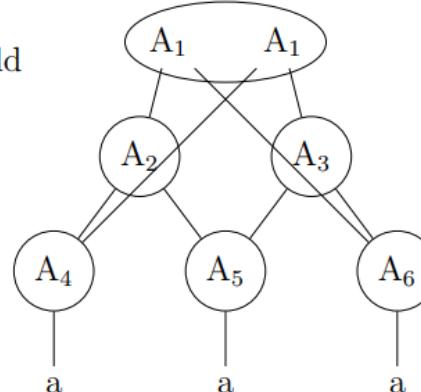
Hilfsmittel: Liste der Zweierpotenzen:

1	2	3	4	5	6	7	8	9	10
2	4	8	16	32	64	128	256	512	1024

(6 Punkte)

Aufgabe 12) Gegeben sei eine PCFG mit zwei Regeln und den Wahrscheinlichkeiten $p(A \rightarrow a) = 1/4$ und $p(A \rightarrow A A) = 3/4$, wobei A das Startsymbol und a ein Terminalsymbol ist.

Betrachten Sie den Parsewald



Berechnen Sie die Inside-Wahrscheinlichkeiten der Parsewald-Nichtterminale A_1, \dots, A_6 .

$$\begin{aligned}
 \text{in}(a) &= 1 \\
 \text{in}(A4) &= \text{in}(A4 \rightarrow a) \\
 &= p(A \rightarrow a) \text{ in}(a) \\
 &= (1/4) * 1 \\
 \text{in}(A5) &= \text{same as in}(A4) \\
 \text{in}(A6) &= \text{same as in}(A4) \\
 \text{in}(A2) &= \text{in}(A2 \rightarrow A4, A5) \\
 &= p(A \rightarrow A, A) \text{ in}(A4) \text{ in}(A5) \\
 &= (3/4) * (1/4) * (1/4) \\
 \text{in}(A3) &= \text{in}(A3 \rightarrow A5, A6) \\
 &= p(A \rightarrow A, A) \text{ in}(A5) \text{ in}(A6) \\
 &= (3/4) * (1/4) * (1/4) \\
 \text{in}(A1) &= \text{in}(A1 \rightarrow A2, A6) + \text{in}(A2 \rightarrow A4, A3) \\
 &= p(A \rightarrow A, A) \text{ in}(A2) \text{ in}(A6) + p(A \rightarrow A, A) \text{ in}(A4) \text{ in}(A3) \\
 &= (3/4) * [(3/4) * (1/4) * (1/4)] * (1/4) + (3/4) * (1/4) * [(3/4) * (1/4) * (1/4)]
 \end{aligned}$$

Inside-Algorithmus

$$(1) \quad \alpha(a) = 1 \quad \text{für Terminalsymbol } a$$

$$(3) \quad \alpha(A \rightarrow X_1 \dots X_n) = p(A \rightarrow X_1 \dots X_n) \prod_{i=1}^n \alpha(X_i) \quad \text{für Parsewaldregeln}$$

$$(2) \quad \alpha(A) = \sum_{A \rightarrow \gamma} \alpha(A \rightarrow \gamma) \quad \text{für Nichtterminale } A$$

Hilfsmittel: Liste der Zweierpotenzen:

1	2	3	4	5	6	7	8	9	10
2	4	8	16	32	64	128	256	512	1024

$$\text{in}(A1) = \text{in}(A1 \rightarrow A2, A6) + \text{in}(A2 \rightarrow A4, A3)$$

$$= p(A \rightarrow A, A) \text{ in}(A2) \text{ in}(A6)$$

$$= (3/4) * [(3/4) * (1/4) * (1/4)] * (1/4) + p(A \rightarrow A, A) \text{ in}(A4) \text{ in}(A3)$$

$$+ (3/4) * (1/4) * [(3/4) * (1/4) * (1/4)]$$

$$\frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{4}$$

$$\left(\frac{3}{4}\right)^2 \cdot \left(\frac{1}{4}\right)^3 + \left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^3$$

$$\frac{3^2}{4^2} \cdot \frac{1^3}{4^3} + \text{same}$$

$$\frac{3^2}{(2^2)^2} \cdot \frac{1^3}{(2^2)^3} + \text{same}$$

$$\frac{3^2}{2^2 \cdot 2} \cdot \frac{1^3}{2^{2 \cdot 3}} + \text{same}$$

$$\frac{3^2}{2^4} \cdot \frac{1}{2^6} + \text{same}$$

$$\frac{3^2}{2^4 \cdot 2^6} + \text{same}$$

$$\frac{3^2}{2^{4+6}} + \text{same}$$

$$= \frac{9}{2^{10}} + \frac{9}{2^{10}}$$

$$= \frac{\cancel{2} \cdot 9}{\cancel{2}^{10} \cancel{9}}$$

$$= \frac{9}{2^9}$$

$$= \frac{9}{512}$$

Hilfsmittel: Liste der Zweierpotenzen:

1	2	3	4	5	6	7	8	9	10
2	4	8	16	32	64	128	256	512	1024

Power of Power Multiplication

Let's consider $(2^2)^3 \times (2^4)^5$

We apply the Power Rule to both items:

$$(2^2)^3 \times (2^4)^5 = 2^{2 \times 3} \times 2^{4 \times 5}$$

We now finish our task by using the ADD RULE for Multiplying Power terms which have the exact same Base.

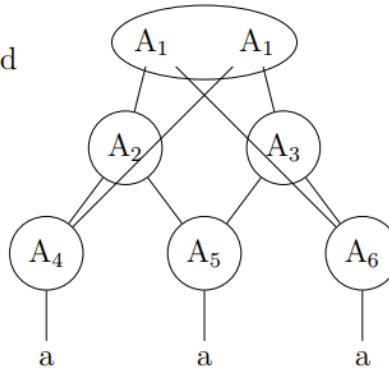
$$= 2^6 \times 2^{20}$$

$$= 2^{20+6} = 2^{26} \checkmark$$

outside

Aufgabe 12) Gegeben sei eine PCFG mit zwei Regeln und den Wahrscheinlichkeiten $p(A \rightarrow a) = 1/4$ und $p(A \rightarrow A A) = 3/4$, wobei A das Startsymbol und a ein Terminalsymbol ist.

Betrachten Sie den Parsewald



Berechnen Sie die Inside-Wahrscheinlichkeiten der Parsewald-Nichtterminale A_1, \dots, A_6 .

Berechnen Sie dann ihre Outside-Wahrscheinlichkeiten.

$$\begin{aligned} \text{out}(A1) &= 1 \\ \text{out}(A2) &= \text{out}(A1 \rightarrow A2, A6) \\ \text{out}(A3) &= \text{out}(A1 \rightarrow A3, A3) \\ \text{out}(A4) &= \text{out}(A2 \rightarrow A4, A5) + \text{out}(A1 \rightarrow A4, A3) \\ \text{out}(A5) &= \text{out}(A3 \rightarrow A5, A6) + \text{out}(A2 \rightarrow A4, A5) \\ \text{out}(A6) &= \text{out}(A1 \rightarrow A2, A6) + \text{out}(A3 \rightarrow A5, A6) \\ \text{out}(a) &= \text{out}(A4 \rightarrow a) + \text{out}(A5 \rightarrow a) + \text{out}(A6 \rightarrow a) \end{aligned}$$

Outside-Algorithmus

$$\beta(S) = 1 \quad \text{für Startsymbol } S$$

$$\beta(A) = \sum_{B \rightarrow \gamma A \delta} \beta(B \rightarrow \gamma A \delta)$$

$$\beta(B \rightarrow X_1 \dots X_m \underline{A} X_{m+1} \dots X_n) = \beta(B) p(B \rightarrow X_1 \dots X_m \underline{A} X_{m+1} \dots X_n) \prod_{i=1}^n \alpha(X_i)$$

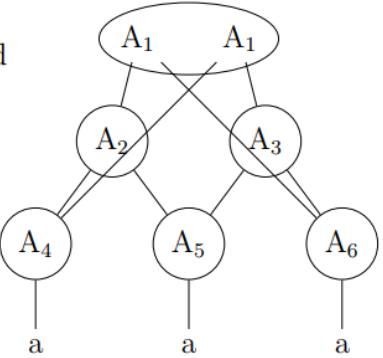
Beispiel

$$\begin{aligned} \text{out}(A2) &= \text{out}(A1 \rightarrow A2, A6) \\ &= \text{out}(A1) p(A \rightarrow A, A) \text{ in}(A6) \\ &= 1 * (3/4) * (1/4) \end{aligned}$$

Aposteriori (gamma)

Aufgabe 12) Gegeben sei eine PCFG mit zwei Regeln und den Wahrscheinlichkeiten $p(A \rightarrow a) = 1/4$ und $p(A \rightarrow A_1 A_2) = 3/4$, wobei A das Startsymbol und a ein Terminalsymbol ist.

Betrachten Sie den Parsewald



Berechnen Sie die Inside-Wahrscheinlichkeiten der Parsewald-Nichtterminale A_1, \dots, A_6 .

Berechnen Sie dann ihre Outside-Wahrscheinlichkeiten.

Berechnen Sie zum Schluss die Aposteriori-Wahrscheinlichkeit (erwartete Häufigkeit) der Parsewaldregel $A_2 \rightarrow A_4 A_5$.

- erwartete Regelhäufigkeit

$$\gamma(A \rightarrow \delta) = \alpha(A \rightarrow \delta) \beta(A)/\alpha(S)$$

$$\text{gamma}(A_2 \rightarrow A_4, A_5) = \frac{\text{in}(A_2 \rightarrow A_4, A_5) \text{out}(A_2)}{\text{in}(A_1)}$$

$$\begin{aligned}\text{in}(A_2) &= \text{in}(A_2 \rightarrow A_4, A_5) \\ &= p(A \rightarrow A_1 A_2) \text{in}(A_1) \text{in}(A_2) \\ &= (3/4) * (1/4) * (1/4)\end{aligned}$$

Aufgabe 11) Wofür wird der Inside-Outside-Algorithmus benutzt?  (1 Punkt)

Der **Inside-Outside-Algorithmus**

- berechnet effizient die **erwarteten Regelhäufigkeiten** beim EM-Training

Aufgabe 10) Wie arbeitet der Inside-Outside-Algorithmus und wie trainiert man damit eine PCFG?
□ (4 Punkte)

- We can use it to compute "die erwartete HF" in the E-Step of the EM-Training.
- Inside-Outside-Algo computes the inside and outside probability for each node in a given Parsewadl and a set of grammar rules with probabilities. From the inside and outside probabilities, we can compute the "erwartete Häufigkeit" (gamma) of each rule.

Sum gamma(rule) over all training samples to get $f(\text{rule}) \rightarrow$ E-step
Then use $f(\text{rule})$ to compute $p(\text{rule}) \rightarrow$ M step

Neuschätzung der Regel-Wahrscheinlichkeiten

EM-Algorithmus: wiederholte Ausführung der beiden Schritte

① E-Schritt compute f

- ▶ Jeder Satz des Trainingskorpus wird geparst und ein Parsewadl erstellt.
- ▶ Die erwartete Häufigkeit $\gamma(A \rightarrow \delta)$ jeder Parsewadlregel $A \rightarrow \delta$ wird berechnet.
- ▶ Die erwarteten Häufigkeiten $\gamma(A \rightarrow \delta)$ werden für jede CFG-Regel über alle Trainingssätze zu $f(A' \rightarrow \delta')$ summiert, wobei sich A' und δ' aus A und δ durch Entfernen der Knotennummern ergeben.

② M-Schritt

Die Regel-Wahrscheinlichkeiten werden aus den erwarteten Häufigkeiten neu geschätzt:

$$p(A \rightarrow \delta) = \frac{f(A \rightarrow \delta)}{\sum_{\delta'} f(A \rightarrow \delta')}$$

Aufgabe 4) Der Inside-Outside-Algorithmus kann zum Training von probabilistischen kontextfreien Grammatiken (PCFG) auf unannotierten Daten benutzt werden.

Was benötigen Sie außer dem Trainingskorpus sonst noch für das Training? (1 Punkt)

Wie lauten die Formeln zur Berechnung der Inside- und Outside-Wahrscheinlichkeiten?
(3 Punkte)

Wie berechnen Sie die erwartete Häufigkeit der Parsewaldregel $A \rightarrow B_1 \dots B_m$, wenn die Inside- und Outside-Wahrscheinlichkeiten bereits berechnet wurden?

(3 Punkte)

Was wird im M-Schritt des EM-Algorithmus getan?
(1 Punkt)

If the training corpus means a corpus of texts, then you might answer it requires a "Symbolischer Parser" to generate parse trees from a sentence.

Aufgabe 7) Erläutern Sie den EM-Algorithmus am Beispiel des unüberwachten Trainings von PCFGs (also Training auf Rohtexten). Welche Daten benötigen Sie? Welche Berechnungsschritte führt der EM-Algorithmus aus?

(4 Punkte)

- ▶ Training auf automatisch geparsten Texten (von einem Parser erzeugte Parsewälder)

EM-Algorithmus: wiederholte Ausführung der beiden Schritte

① E-Schritt

- ▶ Jeder Satz des Trainingskorpus wird geparsst und ein Parsewald ausgegeben.
- ▶ Die erwartete Häufigkeit jeder Parsewaldregel wird berechnet.
- ▶ Die erwarteten Häufigkeiten werden für jede CFG-Regel über alle Trainingssätze summiert.

② M-Schritt

Die Regel-Wahrscheinlichkeiten werden aus den erwarteten Häufigkeiten neu geschätzt:

$$p(A \rightarrow \delta) = \frac{\gamma(A \rightarrow \delta)}{\sum_{\delta'} \gamma(A \rightarrow \delta')}$$

Parser-Evaluierung

Precision

Es werden korrekte Konstituenten statt korrekter Parsebäume gezählt

Eine Konstituente ist korrekt, wenn der Goldstandard-Parse eine Konstituente mit derselben Start- und Endposition und derselben Kategorie enthält.

$$\text{Precision} = \frac{\text{Anzahl korrekte Konstituenten}}{\text{Anzahl ausgegebene Konstituenten}}$$

F-Score

F-Score: harmonisches Mittel aus Precision und Recall

$$F_1 = \frac{1}{\frac{1}{P} + \frac{1}{R}} = \frac{2}{\frac{R}{PR} + \frac{P}{PR}} = \frac{2PR}{P+R}$$

Der gewichtete F-Score gibt Precision β -mal mehr Gewicht als Recall:

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2P+R}$$

Berechnung von Precision und Recall

TP (True Positives): Zahl der ausgegebenen Konstituenten, die korrekt sind

FP (False Positives): Zahl der ausgegebenen Konstituenten, die falsch sind

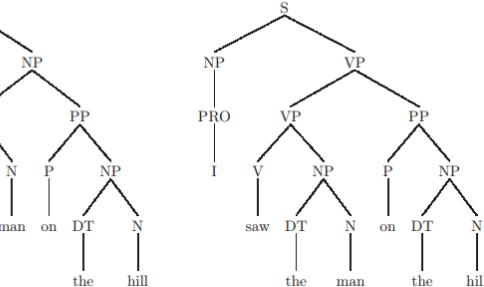
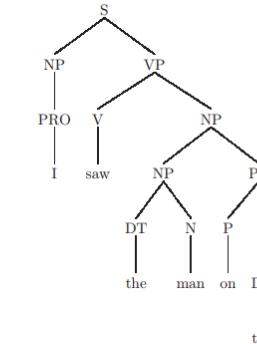
FN (False Negatives): Zahl der Goldstandard-Konstituenten, die fehlten

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

Mit dem **Recall** messen wir zusätzlich, wieviel Prozent der Goldstandard-Konstituenten im Parse enthalten waren.

Zusammen ergeben Precision und Recall ein genaues Bild der Parsing-Genauigkeit.

Beispiel



Goldstandard-Konstituenten:

(S,0,7) (NP,0,1) (VP,1,7) (NP,2,7) (NP,2,4) (PP,4,7) (NP,5,7)

Konstituenten in Parserausgabe:

(S,0,7) (NP,0,1) (VP,1,7) (VP,1,4) (NP,2,4) (PP,4,7) (NP,5,7)

True Positives = 6

False Positives = 1

False Negatives = 1

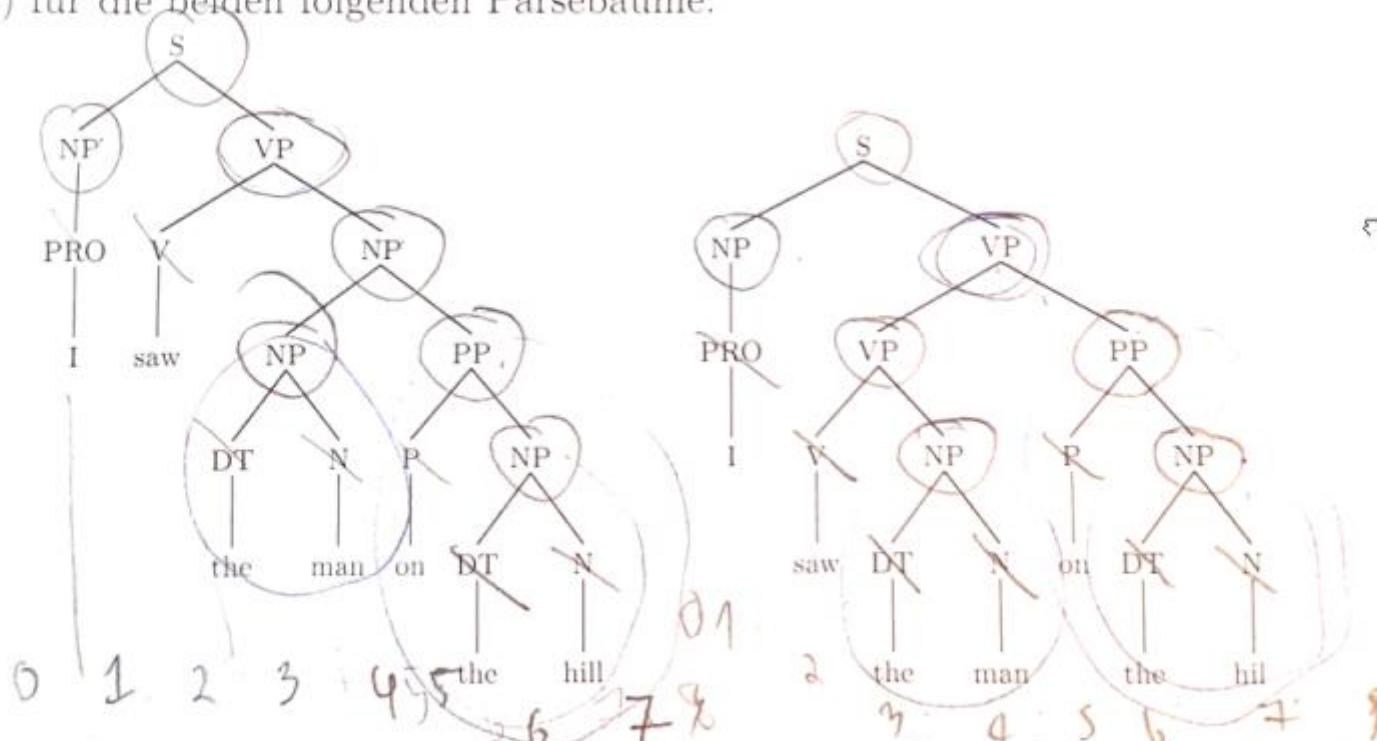
$$\text{Precision} = \text{TP}/(\text{TP}+\text{FP}) = 6/7$$

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN}) = 6/7$$

$$\text{F-Score} = 2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall}) = 2 \cdot 6/7 \cdot 6/7 / (6/7 + 6/7)$$

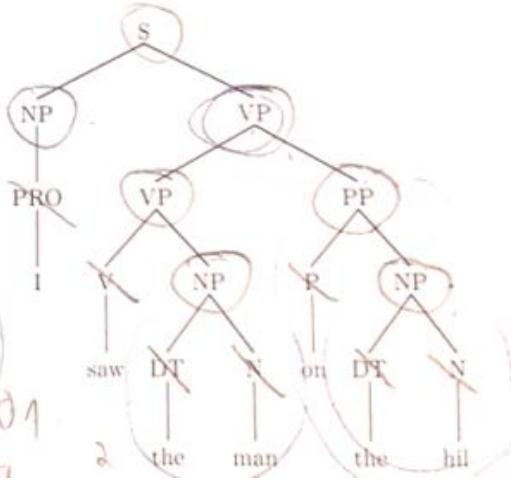
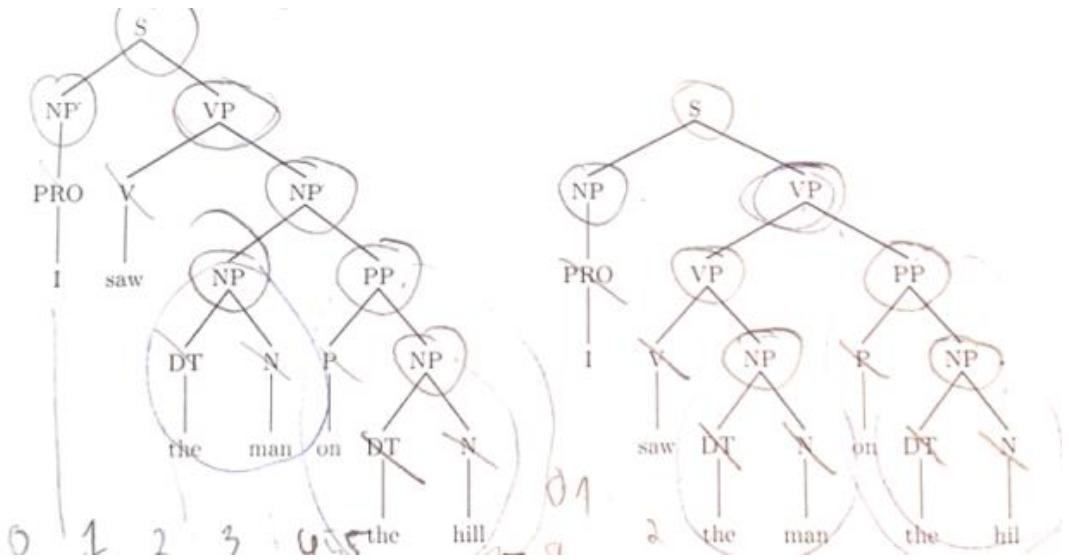
Navigation icons

Aufgabe 12) Zeichnen Sie den **Parsewald** (=kompakte Repräsentation mehrerer Parsebäume) für die beiden folgenden Parsebäume:



(Der Parsewald fasst gemeinsame Teilbäume zusammen. Danach werden noch Bäume zusammengefasst, die bis auf einen einzigsten Teilbaum identisch sind.) (2 Punkte)

note: ignore the scribble



Aufgabe 13) Welche Werte für Precision, Recall und F-Score bekommen Sie für die beiden Parsebäumen oben, wenn Sie die Wortart-Tags (PRO, DT, N, V, P) bei der Bewertung ignorieren? Schreiben Sie auch die Zwischenschritte bei der Berechnung hin.

o

(3 Punkte)

Aufgabe 10) Wie wird üblicherweise die Genauigkeit von (Konstituenten-)Parseern gemessen? Erklären Sie die Methode und wie das Maß berechnet wird. (3 Punkte)

Precision

Es werden korrekte Konstituenten statt korrekter Parsebäume gezählt

Eine Konstituente ist korrekt, wenn der Goldstandard-Parse eine Konstituente mit derselben Start- und Endposition und derselben Kategorie enthält.

$$\text{Precision} = \frac{\text{Anzahl korrekte Konstituenten}}{\text{Anzahl ausgegebene Konstituenten}}$$

Precision

?

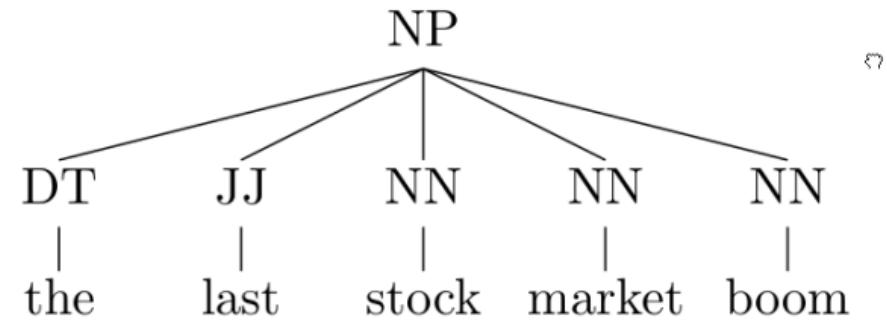
Es werden korrekte Konstituenten statt korrekter Parsebäume gezählt

Eine Konstituente ist korrekt, wenn der Goldstandard-Parse eine Konstituente mit derselben Start- und Endposition und derselben Kategorie enthält.

$$\text{Precision} = \frac{\text{Anzahl korrekte Konstituenten}}{\text{Anzahl ausgegebene Konstituenten}}$$

Warum benutzen wir Konstituenten Statt den ganzen Baum für die Evaluierung?

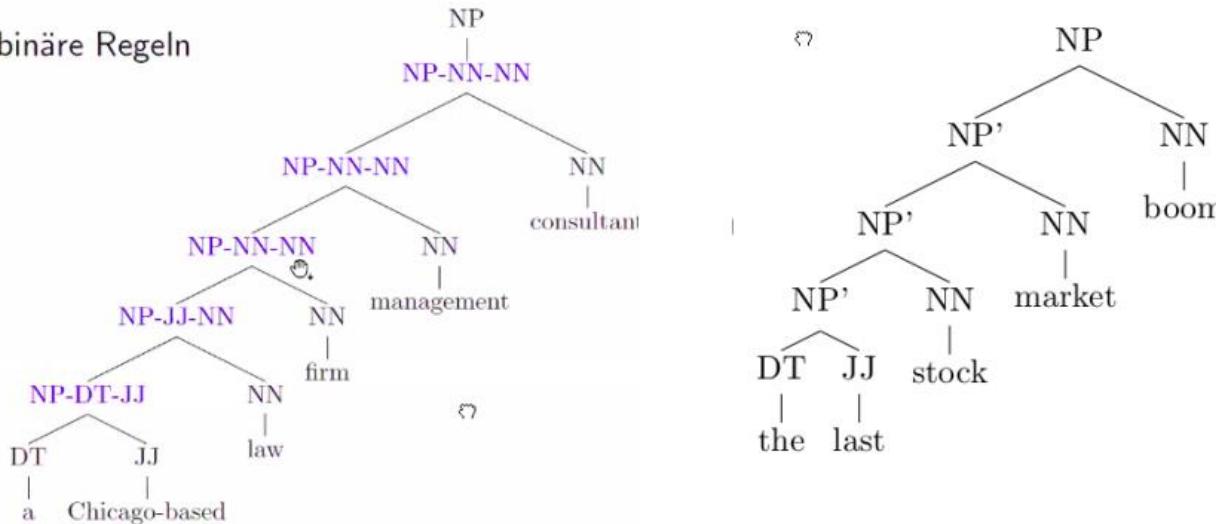
- Wenn wir den Baum zur Evaluierung benutzen (Wenn ein Baum ist richtig geparst, wird das als ein True Positive gezählt), haben kurze und lange Sätze dasselbe Gewicht.
- Lange Sätze sind schwieriger, richtig geparst zu werden als kurze Sätze, und sollen deswegen mehr Gewicht bekommen.



Führen Sie Markovisierung
durch.

Markowisierung

- Aufspaltung von langen Regeln in binäre Regeln
- Neue Hilfskategorien mit z.B.
 - ▶ der Elternkategorie und
 - ▶ den Kategorien der beiden letzten Tochterknoten
- Das Entfernen der Hilfsknoten liefert wieder den Originalparse
 - ⇒ Reduktion der Grammatikgröße
 - ⇒ Verbesserung ihrer Abdeckung



Die gezeigte Markowisierung ist äquivalent dazu, dass wir die rechten Seiten (und Wahrscheinlichkeiten) von bspw. NP-Regeln mit einem Markow-Modell 2. Ordnung erzeugen.

Aufgabe 11) Wie wird eine Grammatik **markowisiert**. Was ist der Vorteil der Markowisierung? (2 Punkte)

Markowisierung

- Aufspaltung von langen Regeln in binäre Regeln

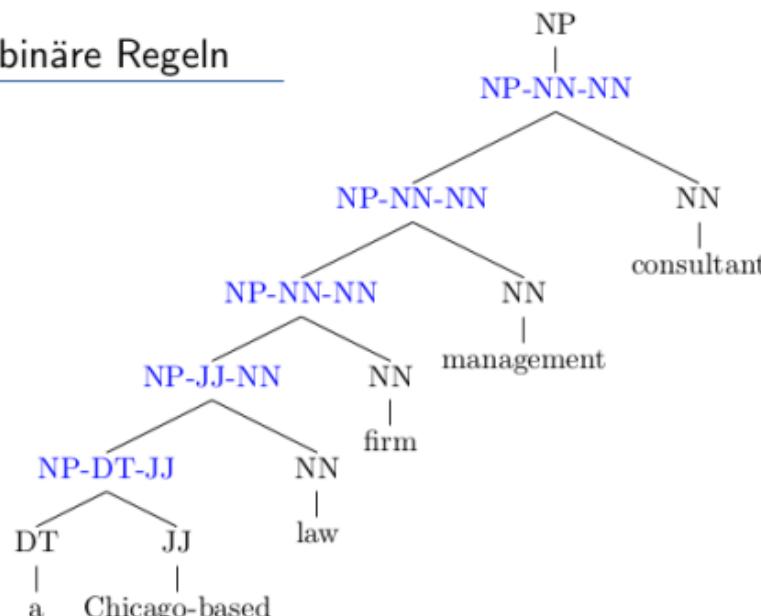
- Neue Hilfskategorien mit z.B.

- ▶ der Elternkategorie und
- ▶ den Kategorien der beiden letzten Tochterknoten

- Das Entfernen der Hilfsknoten liefert wieder den Originalparse

⇒ Reduktion der Grammatikgröße

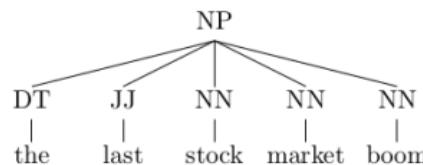
⇒ Verbesserung ihrer Abdeckung



Die gezeigte Markowisierung ist äquivalent dazu, dass wir die rechten Seiten (und Wahrscheinlichkeiten) von bspw. NP-Regeln mit einem Markow-Modell 2. Ordnung erzeugen.

Aufgabe 5) Warum wird beim PCFG-Parsen die Grammatik oft markowisiert? Welches Problem soll dadurch gelöst werden? (3 Punkte)

- Viele Baumbanken verwenden flache Strukturen.



- Die extrahierten Grammatiken enthalten viele Regeln mit langen rechten Seiten, die nur einmal auftauchen.

$NP \rightarrow DT\ JJ\ NN\ NN\ NN$

- Andere ähnliche Regeln fehlen, werden aber für das Parsen mancher Sätze benötigt.

$NP \rightarrow DT\ JJ\ NN\ NN\ NN\ NN$

⇒ Reduktion der Grammatikgröße

⇒ Verbesserung ihrer Abdeckung

Aufgabe 5) Geben Sie an, wie bei einer probabilistischen kontextfreien Grammatik folgende Werte definiert sind:

□

- die Wahrscheinlichkeit eines Parsebaumes
- die Wahrscheinlichkeit eines Satzes (= Menge von Parsebäumen)
- die Wahrscheinlichkeit eines Korpus (= Folge von Sätzen) (2 Punkte)

Korpuswahrscheinlichkeit $p(C) = \prod_{s \in C} p(s)$

Satzwahrscheinlichkeit $p(s) = \sum_{t \in T(s)} p(t)$

Parsewahrscheinlichkeit $p(t) = p(r_1, \dots, r_n) = \prod_{i=1}^n p(r_i)$

$T(s)$ sind die Analysen des Satzes s .

r_1, \dots, r_n sind die Regeln der Linkstableitung von t .

Berkeley-Parser

Aufgabe 7) Erklären Sie ausführlich die Grundidee des Berkeley-Parsers von Petrov und Klein und wie er trainiert wird (ohne Formeln). (4 Punkte)

□

Wir wollen die Parse-Kategorien(die nicht-terminalen Symbole wie S, NP, PRO,...) in SubKategorien verfeinern.

Die Methode von Petrov und Klein soll die Subkategorisierung automatisch lernt.

Jede Kategorie wird in zwei neue Kategorien aufgespaltet. Z.B. NP wird NP/0 und NP/1.

Damit bekommen wir auch neue Regel z.B. NP/0 → PRO/0 oder NP/0 → PRO/1 und viele Parsebäume aus der neuen Regeln. Die neuen Parsebäume sollen mit EM-Training trainiert werden (EM schätzt die WK der neuen Regeln).

Nach dem Training: Wenn wir die Regeln nach der Kategorie auf der linken Seite gruppieren und die Regeln nach der Regel-WK sortieren, sehen wir die gelernte Subkategorisierung.

Beispiel: wahrscheinlichste Expansionen der DT-(Unter-)Kategorien

DT the (0.50) a (0.24) The (0.08) ...

DT/0 that (0.15) this (0.14) some (0.11) ...

DT/1 the (0.54) a (0.25) The (0.09) ...

DT/0 → that (0.15)

DT/0 → this (0.14)

DT/0 → some (0.11)

DT/1 → the (0.54)

DT/1 → a (0.25)

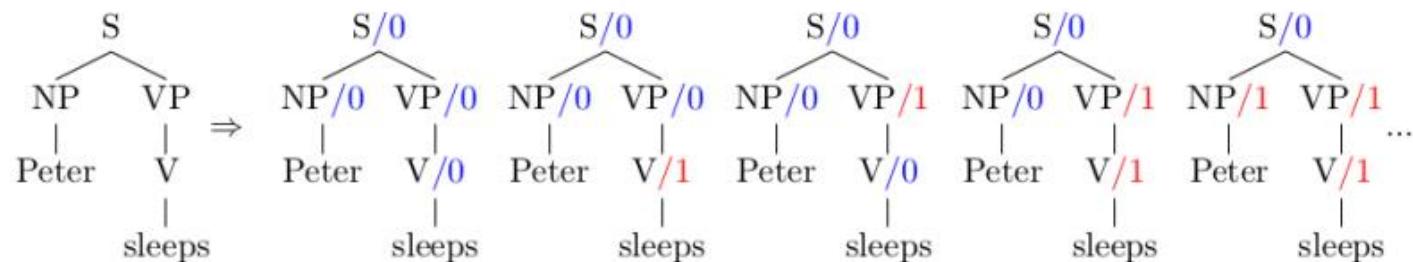
DT/1 → The (0.09)

Der originale Parsebaum wird dann mit den neuen Subkategorien annotiert. Dadurch dass die Kategorien spezifischer werden, wird das Performanz der Parser auch erhöht.

Synthetische Merkmale

Grundidee (von Petrov/Klein)

- Alle Kategorien werden durch ein synthetisches Merkmal mit den Werten 0 bzw. 1 aufgespalten.
- Jeder Parse der Baumbank kann von der neuen Grammatik auf viele unterschiedliche Arten generiert werden.
- Durch EM-Training wird die neue Grammatik an die Baumbank angepasst.



Die modifizierte Grammatik liefert für den alten Parsebaum $2^4 = 16$ neue Parsebäume.

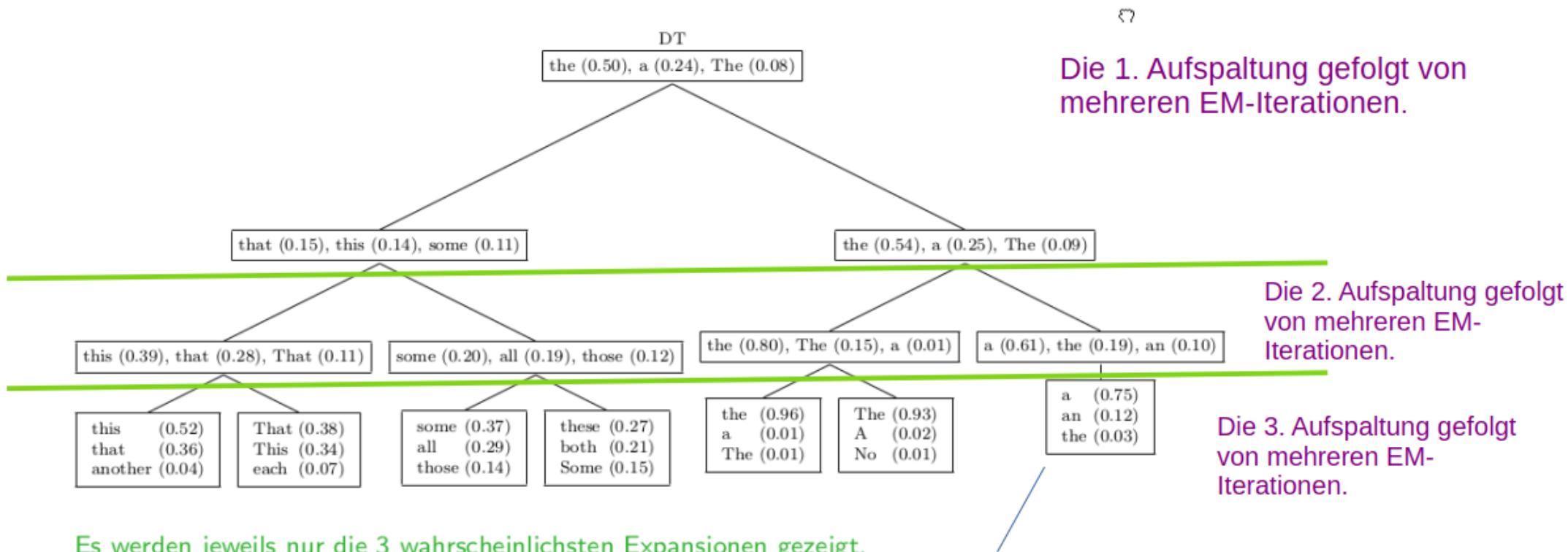
NP'/0 → NP'/0 NN/0	0.24
NP'/0 → NP'/0 NN/1	0.26
NP'/0 → NP'/1 NN/0	0.25

...

Die Subkategorisierungen können noch weiter verfeinert werden.

Rekursive Verfeinerung

Rekursives Aufspalten der Kategorien gefolgt von mehreren EM-Iterationen verfeinert die Kategorien immer mehr.



Beispiel für andere Regeln : hier sehen wir, nach mehrmaligen Aufspaltungen und EM-Trainings wurde ADVP in 9 Subkategorien unterteilt.

ADVP			
ADVP-0	RB-13 NP-2	RB-13 PP-3	IN-15 NP-2
ADVP-1	NP-3 RB-10	NP-3 RBR-2	NP-3 IN-14
ADVP-2	IN-5 JJS-1	RB-8 RB-6	RB-6 RBR-1
ADVP-3	RBR-0	RB-12 PP-0	RP-0
ADVP-4	RB-3 RB-6	ADVP-2 SBAR-8	ADVP-2 PP-5
ADVP-5	RB-5	NP-3 RB-10	RB-0
ADVP-6	RB-4	RB-0	RB-3 RB-6
ADVP-7	RB-7	IN-5 JJS-1	RB-6
ADVP-8	RB-0	RBS-0	RBR-1 IN-14
ADVP-9	RB-1	IN-15	RBR-0

Original paper: Learning Accurate, Compact, and Interpretable Tree Annotation
<https://dl.acm.org/doi/pdf/10.3115/1220175.1220230>

Vereinigung

- Ab einem bestimmten Punkt ist eine weitere Aufspaltung der Kategorien nicht mehr vorteilhaft, weil sie zu [Sparse-Data-Problemen](#) führt.

Die Penn-Treebank verwendet bspw. eine eigene Kategorie „,” für Kommas, deren Aufspaltung nichts nützt.

- ⇒ Unterkategorien, die sich zu ähnlich sind, werden daher nach dem EM-Training wieder [zusammengefasst](#).

Strategie:

- Nach jedem EM-Training und vor dem Aufspalten der Kategorien, werden [50%](#) der vorherigen Aufspaltungen rückgängig gemacht.
- Für jede Aufspaltung wird berechnet, wie stark sich die [Wahrscheinlichkeit der Trainingsdaten](#) verringert, wenn die Aufspaltung rückgängig gemacht wird. ([Details im Originalartikel](#))
- Die Aufspaltungen mit der [kleinsten Differenz](#) werden rückgängig gemacht.

Experimentelle Ergebnisse

- “Berkeley”-Parser von Slav Petrov und Dan Klein
- Der Parser liefert für viele verschiedene Sprachen und Baumbanken gute Ergebnisse
- Deutsch (Precision=80.1%, Recall=80.1%)
- Englisch (Precision=90.2%, Recall=89.9%)
- Chinesisch (Precision=84.8%, Recall=81.9%)

Aber: Neuere Parser auf Basis neuronaler Netze sind noch besser.

Kurzzusammenfassung: Berkeley-Parser

- ① gegeben: eine **Baumbank**
- ② **Binarisierung** der Parsebäume durch Einfügen von Hilfsknoten
- ③ Extraktion einer Grammatik mit Regelhäufigkeiten
- ④ für mehrere Iterationen
 - ① Aufspaltung jeder Kategorie X in zwei neue Kategorien X1 und X2
 - ② annähernd uniforme Verteilung der Häufigkeit einer Regel auf die neuen Regeln
 - ③ für mehrere Iterationen
 - ① Schätzung der Regelwahrscheinlichkeiten aus den Regelhäufigkeiten
 - ② Für jeden Parsebaum der Baumbank wird der Parsewald mit Analysen entsprechend der verfeinerten Grammatik berechnet.
 - ③ Mit dem Inside-Outside-Algorithmus werden erwartete Häufigkeiten für die Parsewaldregeln berechnet.
 - ④ Die erwarteten Regelhäufigkeiten werden über alle Sätze summiert.
 - ④ Die schlechtere Hälfte der Aufspaltungen wird rückgängig gemacht.