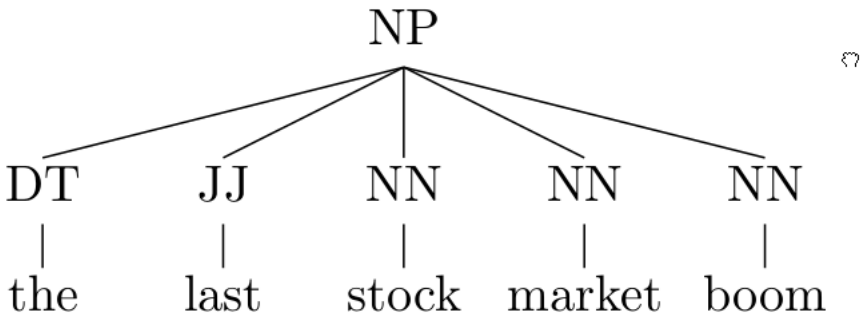
Precision

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 Sätzte sind schwieriger, richtig geparst zu werden als kurze Sätze, und sollen deswegen mehr Gewicht bekommen.



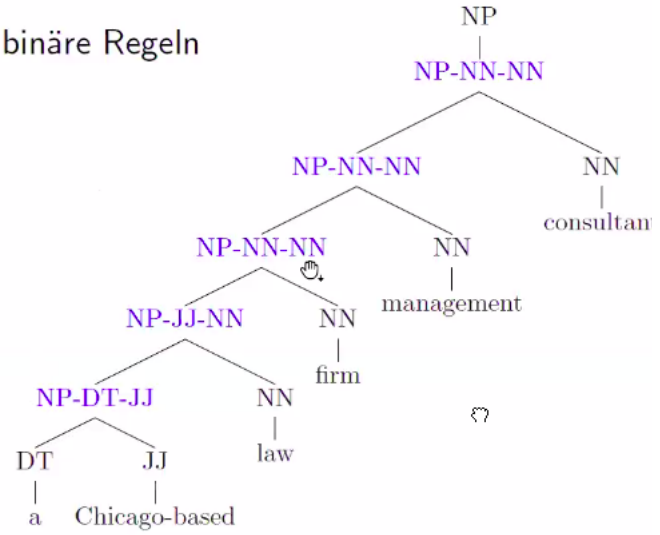
Führen Sie Markovisierung durch.

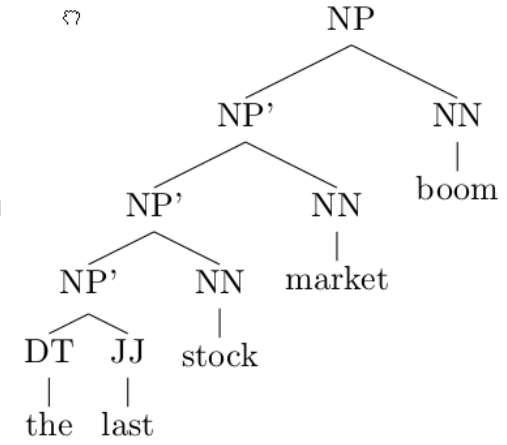
Markowisierung

- 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.

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

Precision = TP/(TP+FP) = 6/7
Recall = TP/(TP+FN) = 6/7
F-Score = 2 P R / (P+R) = 2*6/7*6/7/(6/7+6/7)

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

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

Anzahl aller Konstituenten, die in Goldstandard sind.

Anzahl aller Konstituenten, die der Parser ausgegeben hat.

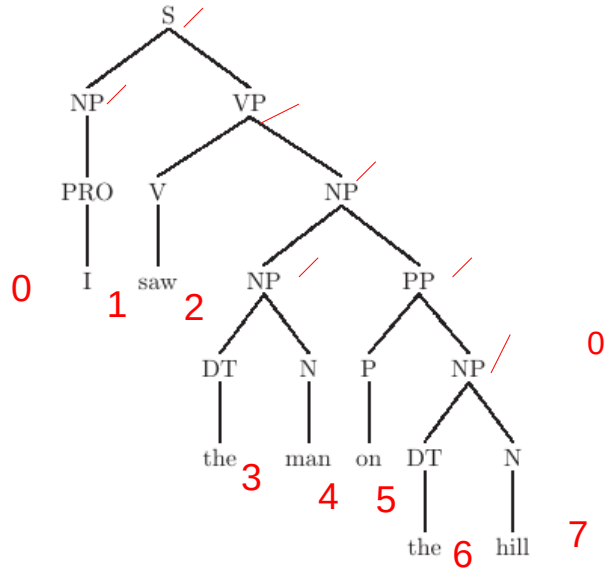
F-Score: harmonisches Mittel aus Precision und Recall

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

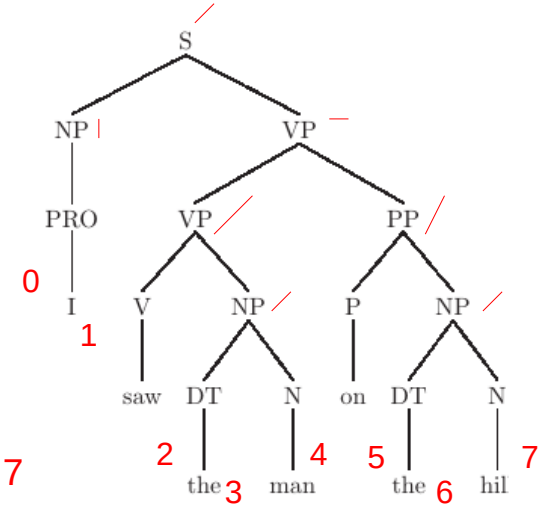
Der gewichtete F-Score gibt Precision β-mal mehr Gewicht als Recall:

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

Beispiel



Goldstandard

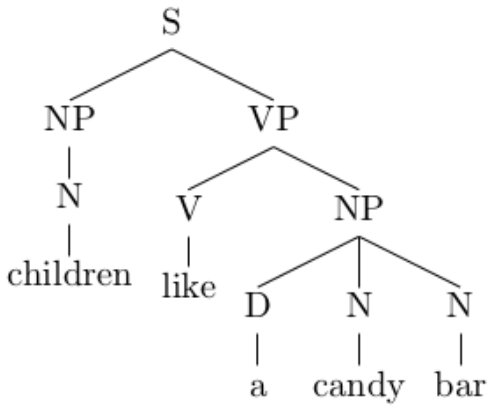


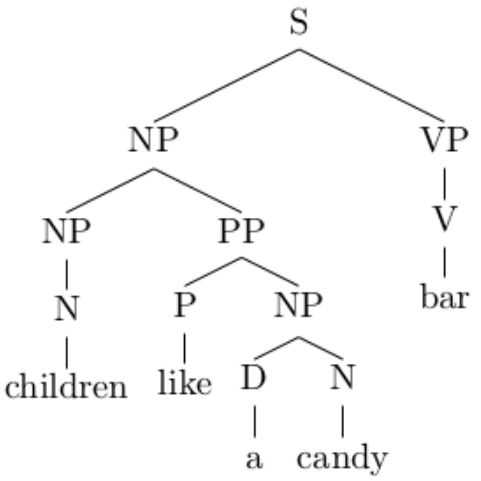
Parserausgabe

Goldstandard-Konstituenten:
(S,0,7) (NP,0,1) (VP,1,7) (NP,2,7) (NP,2,4) (PP,4,7) (NP,5,7)

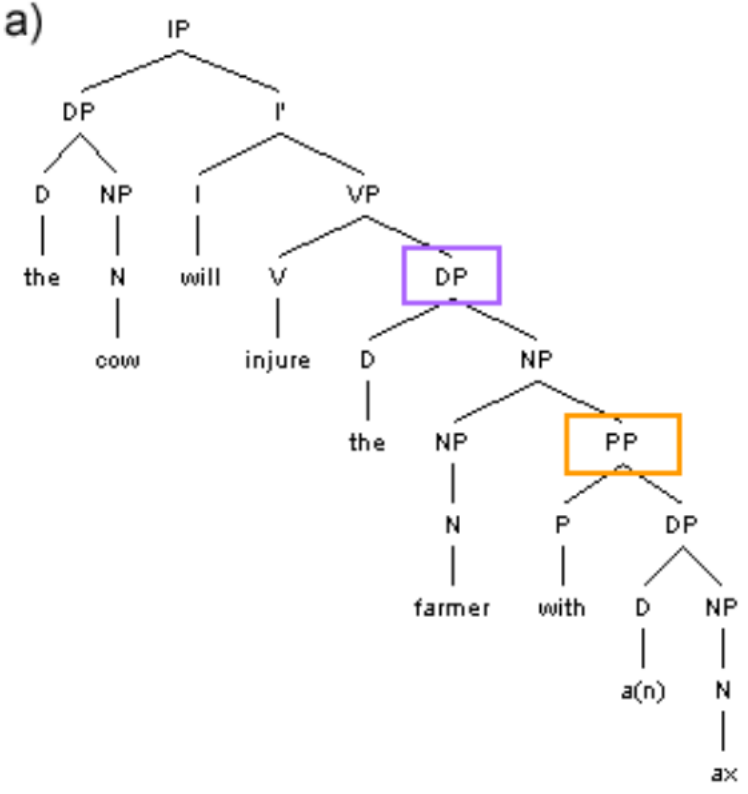
Konstituenten in Parserausgabe:
(S,0,7) (NP,0,1) (VP,1,7) (VP,1,4) (NP,2,4) (PP,4,7) (NP,5,7)

precision = richtig geparst (Grün)/ anzahl der Konstituenten, die der Parser ausgegeben hat
recall = richtig geparst (Grün)/ anzahl aller Goldstand-Konstituenten

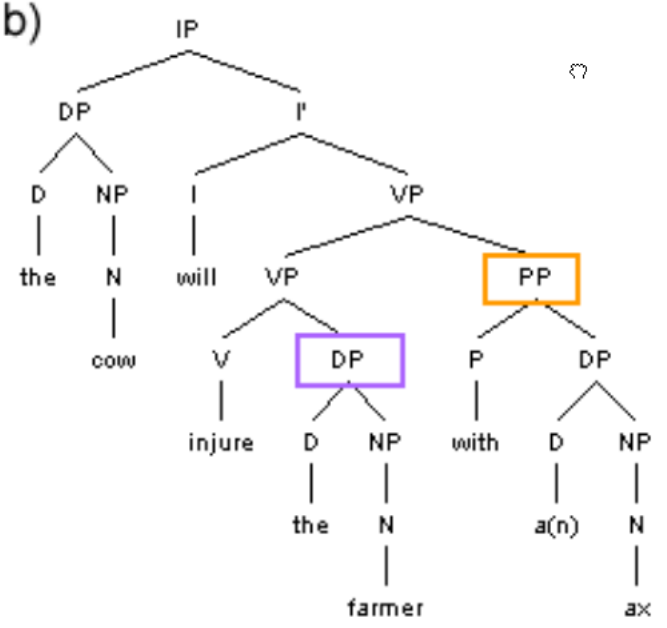

S
NP
VP
N
V
NP
children
like
D
N
N
a
candy
bar


S
NP
VP
NP
PP
V
N
P
NP
bar
children
like
D
N
a
candy

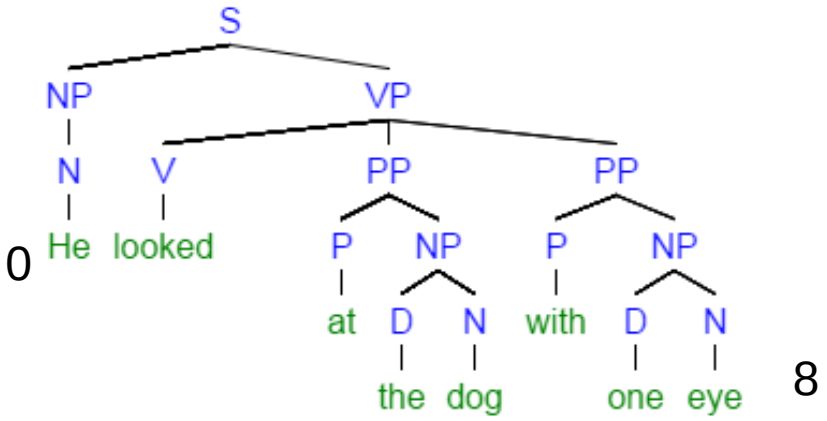
a)


IP
DP
I'
D
NP
I
VP
the
N
will
V
DP
cow
injure
D
NP
the
NP
PP
N
P
DP
farmer
with
D
NP
a(n)
N
ax

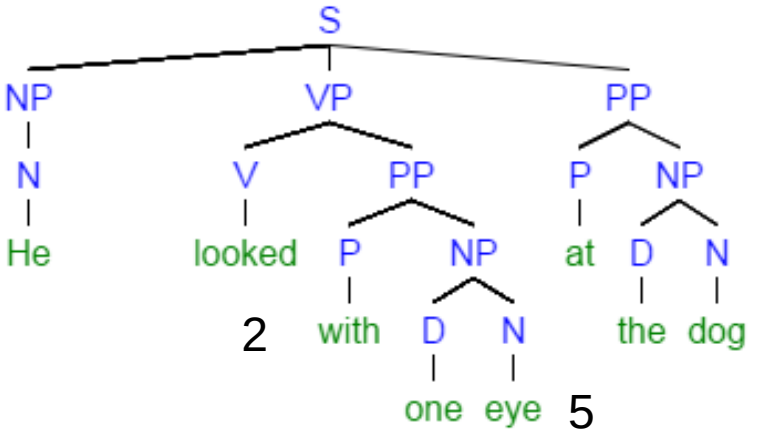
b)


IP
DP
I'
D
NP
I
VP
the
N
will
VP
PP
cow
V
DP
P
DP
injure
D
NP
with
D
NP
the
N
a(n)
N
farmer
ax

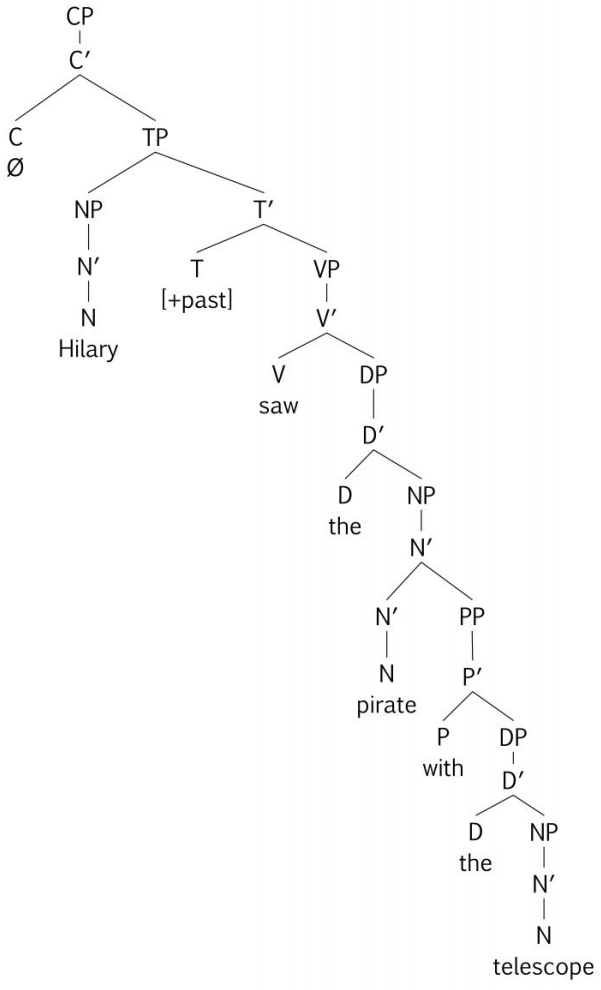
Gold

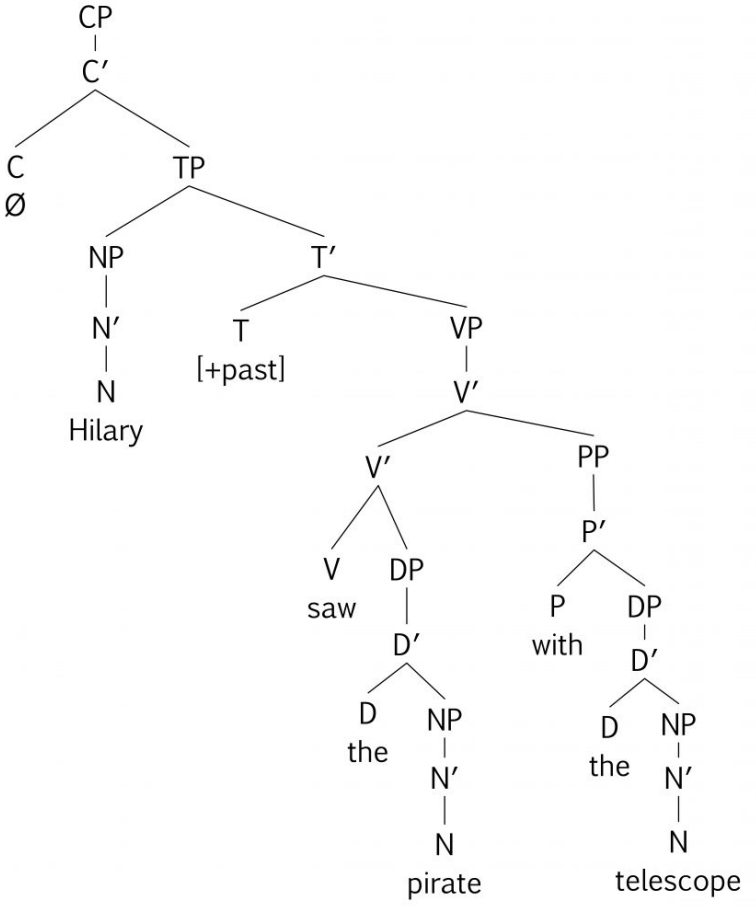


Parser

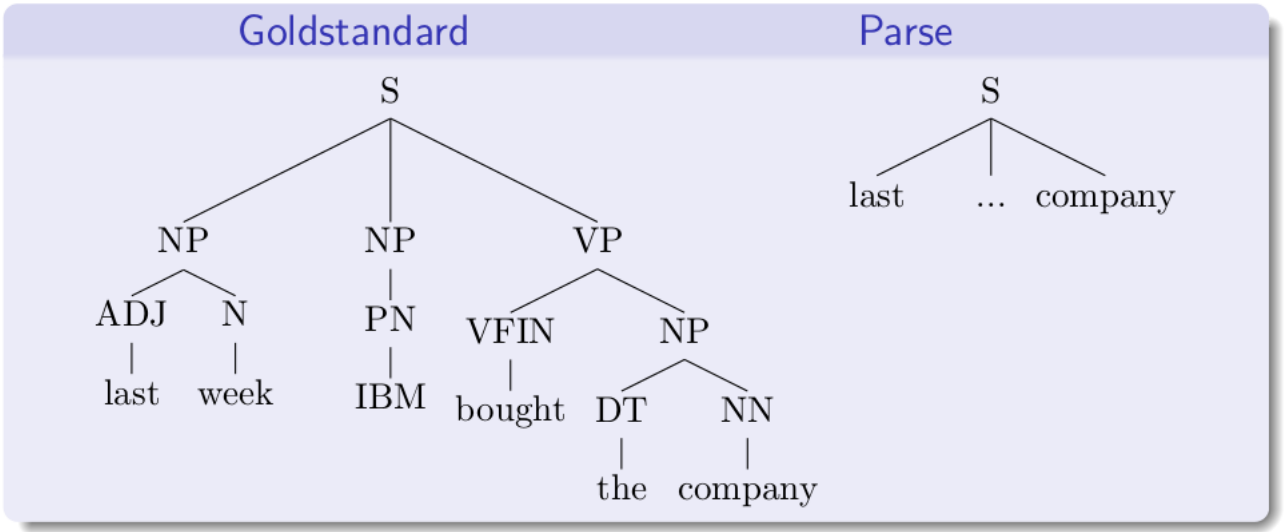


Gold Konstiuenten:
(S,0,8)(NP,0,1)(VP1,8)(PP,2,5) (NP,3,5) (PP,5,8) (NP,6,8)
Parser Konstituenten:
(S,0,8) (NP,0,1) (vp,1,5) PP (2,5) NP(3,5) PP(5,8) NP(6,8)
Precision: 6/7
Recall:6/ 7
F1:


CP
C′
C
Ø
TP
NP
N′
N
Hilary
T′
T
[+past]
VP
V′
V
saw
DP
D′
D
the
NP
N′
N′
PP
N
pirate
P′
P
with
DP
D′
D
the
NP
N′
N
telescope


CP
C′
C
Ø
TP
NP
N′
N
Hilary
T′
T
[+past]
VP
V′
V′
PP
V
saw
DP
P′
D′
P
with
DP
D
the
NP
D′
N′
D
the
NP
N
pirate
N′
N
telescope

Problem Der Precision-Wert sagt nicht, wieviele der Goldstandard-Konstituenten der Parser ausgegeben hat.



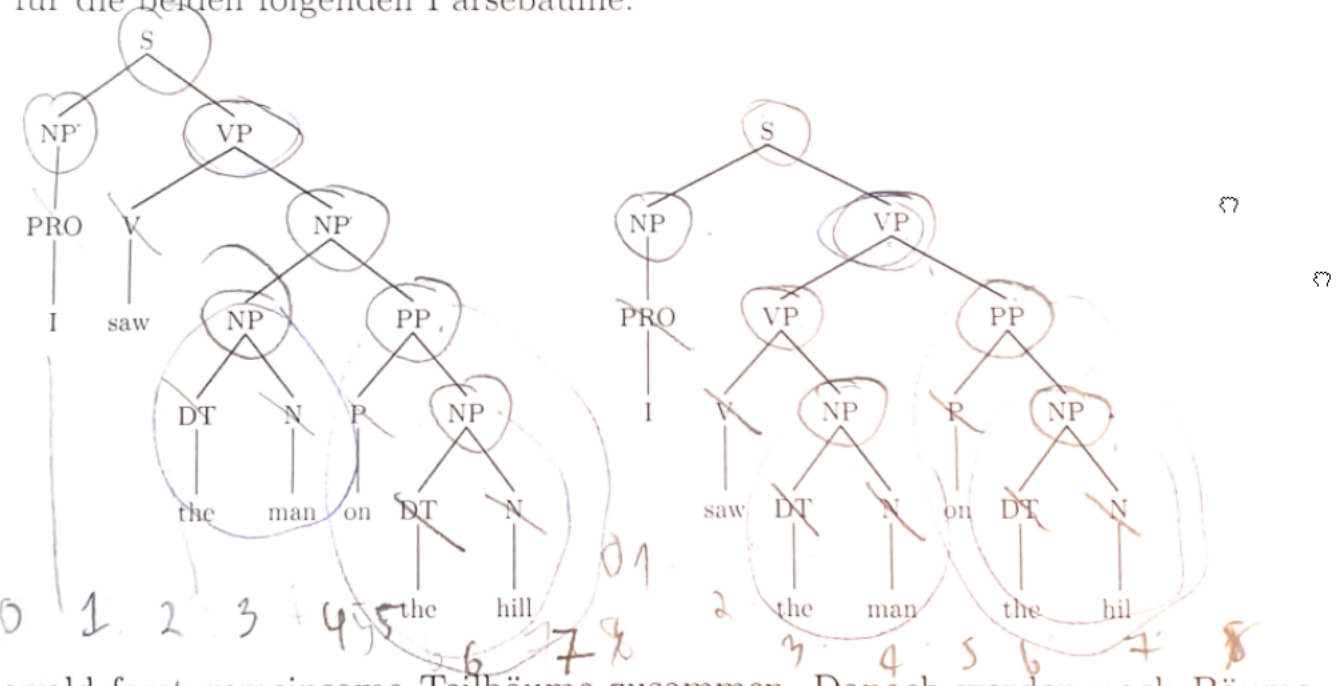
Der obige Parse mit nur einer Konstituente erzielt eine Precision von 100%!

1. Warum Recall alleine ist kein gutes Maß?
Geben Sie ein Beispiel, wo Recall 100% ist, obwohl der Parser viele Fehler macht.

- Recall sagt nicht, wie viele False-Positive(rot) der Parser macht.
- Wenn der Parser alle Gold-Konstituenten korrekt ausgibt, dann hat er 100% Recall, obwohl er auch viele unkorrekte Konstututen(FP) ausgibt.

Gold: (S,0,7) (NP,0,1) (VP,1,7) (NP,2,7)
Parser: (S,0,7) (NP,0,1) (VP,1,7) (NP,2,7) (NP,2,4) (PP,4,7) (NP,5,7) (VP,1,2) (N,0,1)

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 einzigen Teilbaum identisch sind.) (2 Punkte)

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 neue Parsebäume soll mit EM-Training trainiert werden (EM schätzt die WK der neuen Regeln).
Nach dem Training: Wenn wir die Regeln nach dem Kategorie auf der linken Seite grouppieren und die Regeln nach der Regel-WK sortieren, sehen wir die gelernte Subkagerierung.

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)

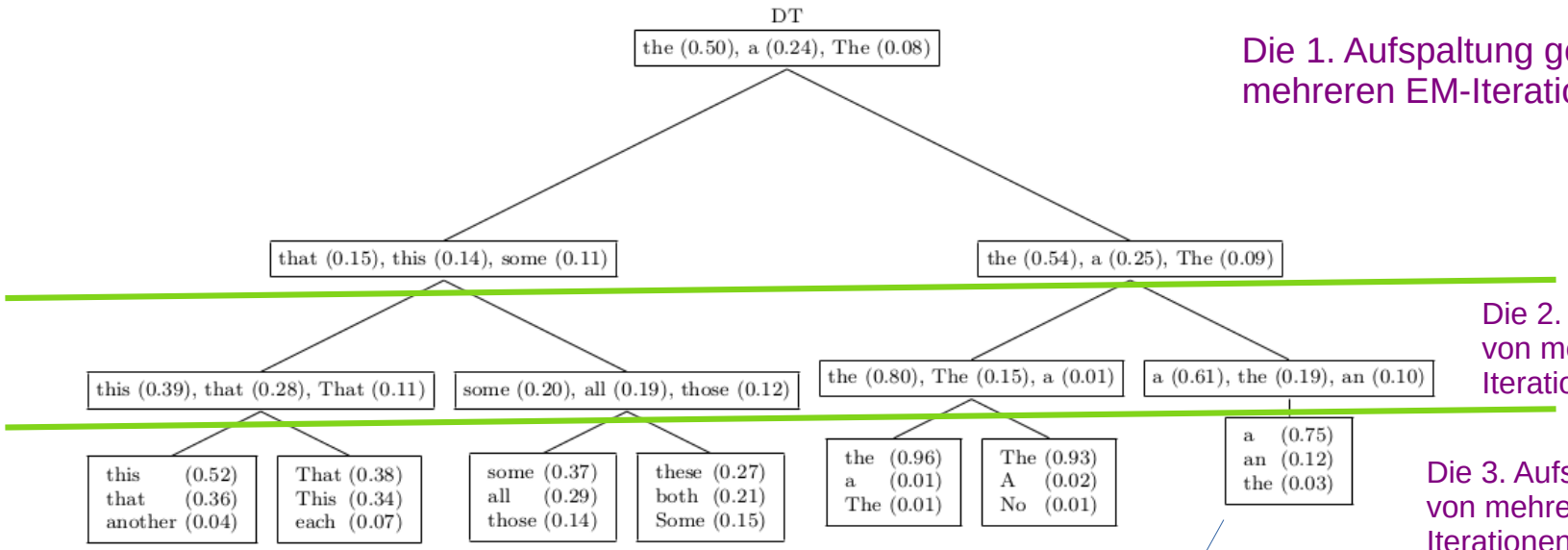
DT/1 → the (0.54)
DT/1 → a (0.25)
DT/1 → The (0.09)

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

Die Subkategorisieren können noch weiter verfeinert werden.

Rekursive Verfeinerung

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



Die 1. Aufspaltung gefolgt von mehreren EM-Iterationen.

Die 2. Aufspaltung gefolgt von mehreren EM-Iterationen.

Die 3. Aufspaltung gefolgt von mehreren EM-Iterationen.

Es werden jeweils nur die 3 wahrscheinlichsten Expansionen gezeigt.

Wenn die Aufspaltung nicht mehr vorteilhaft ist, werden die Subkatogorien wieder zusammengefasst

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

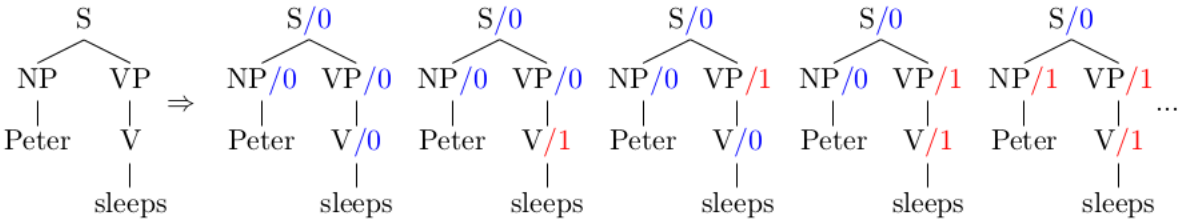
ADVP			
ADVP-0	RB-13 NP-2	RB-13 PP-3	IN-15 NP-2
ADVP-1	NP-3 RB-10	NP-3 RBR-2	NP-3 IN-14
ADVP-2	IN-5 JJS-1	RB-8 RB-6	RB-6 RBR-1
ADVP-3	RBR-0	RB-12 PP-0	RP-0
ADVP-4	RB-3 RB-6	ADVP-2 SBAR-8	ADVP-2 PP-5
ADVP-5	RB-5	NP-3 RB-10	RB-0
ADVP-6	RB-4	RB-0	RB-3 RB-6
ADVP-7	RB-7	IN-5 JJS-1	RB-6
ADVP-8	RB-0	RBS-0	RBR-1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

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.



...

NP'/0 → NP'/0 NN/0 0.24

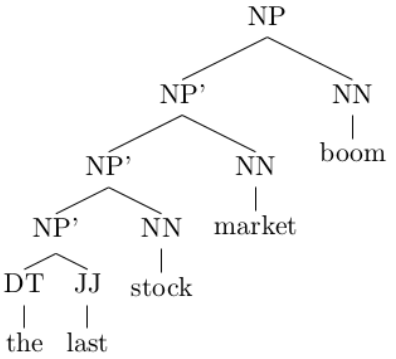
NP'/0 → NP'/0 NN/1 0.26

NP'/0 → NP'/1 NN/0 0.25

...

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

Vorverarbeitung



1. Binarisierung der Parsebäume
2. Extraktion einer Grammatik mit Häufigkeiten

   NP → NP' NN 1
   NP' → NP' NN 2
   NP' → DT JJ 1

3. Aufspaltung jeder Kategorie in 2 neue Kategorien
   Uniforme Verteilung der Häufigkeiten über die neuen Regeln
   (mit kleinen Abweichungen, um die Symmetrie zu brechen.
   Sonst bleibt das EM-Training im Anfangszustand stecken.)

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

EM-Training

1. Aus den Regelhäufigkeiten werden Maximum-Likelihood-Schätzungen berechnet.
$$p(A \to \alpha) = \frac{f(A \to \alpha)}{\sum_{A \to \beta} f(A \to \beta)}$$
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.
5. Weiter mit 1)

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.