Training PCFGs

Computational Linguistics

Alexander Koller

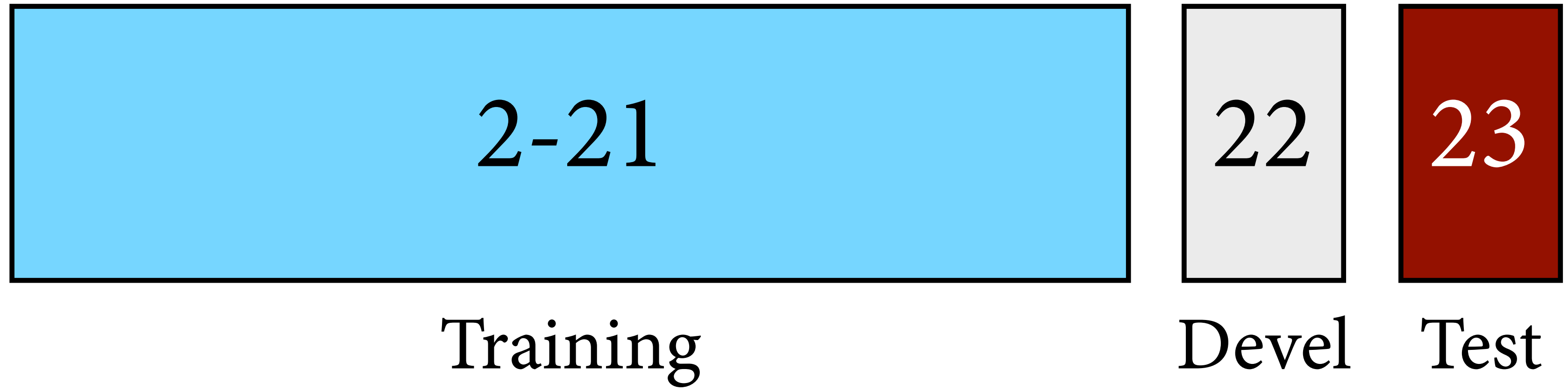
28 November 2023

Probabilistic CFGs

S → NP VP	[1.0]	VP → V NP	[0.5]
NP → Det N	[0.8]	VP → VP PP	[0.5]
NP → i	[0.2]	V → shot	[1.0]
N → N PP	[0.4]	PP → P NP	[1.0]
N → elephant	[0.3]	P → in	[1.0]
N → pyjamas	[0.3]	Det → an	[0.5]
		Det → my	[0.5]

Evaluation

- Step 1: Decide on training and test corpus.
  For WSJ corpus, there is a conventional split by sections:



Evaluation

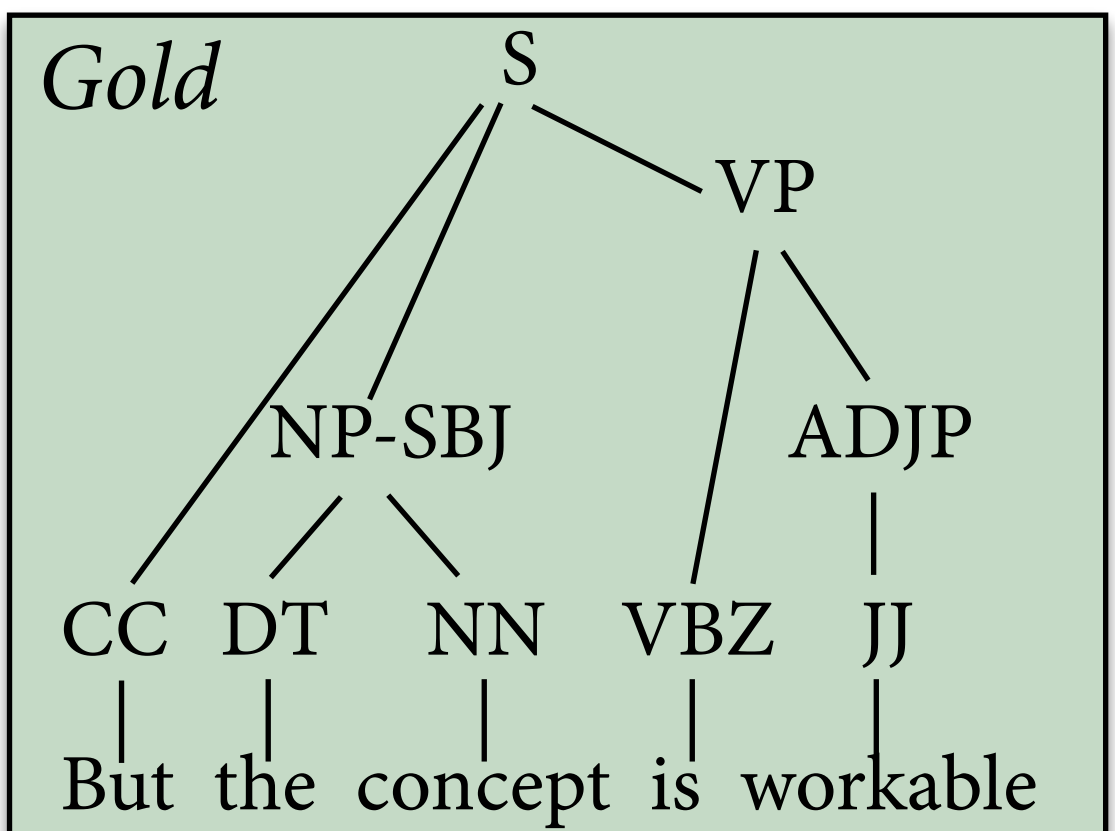
- Step 2: How should we measure the accuracy of the parser?

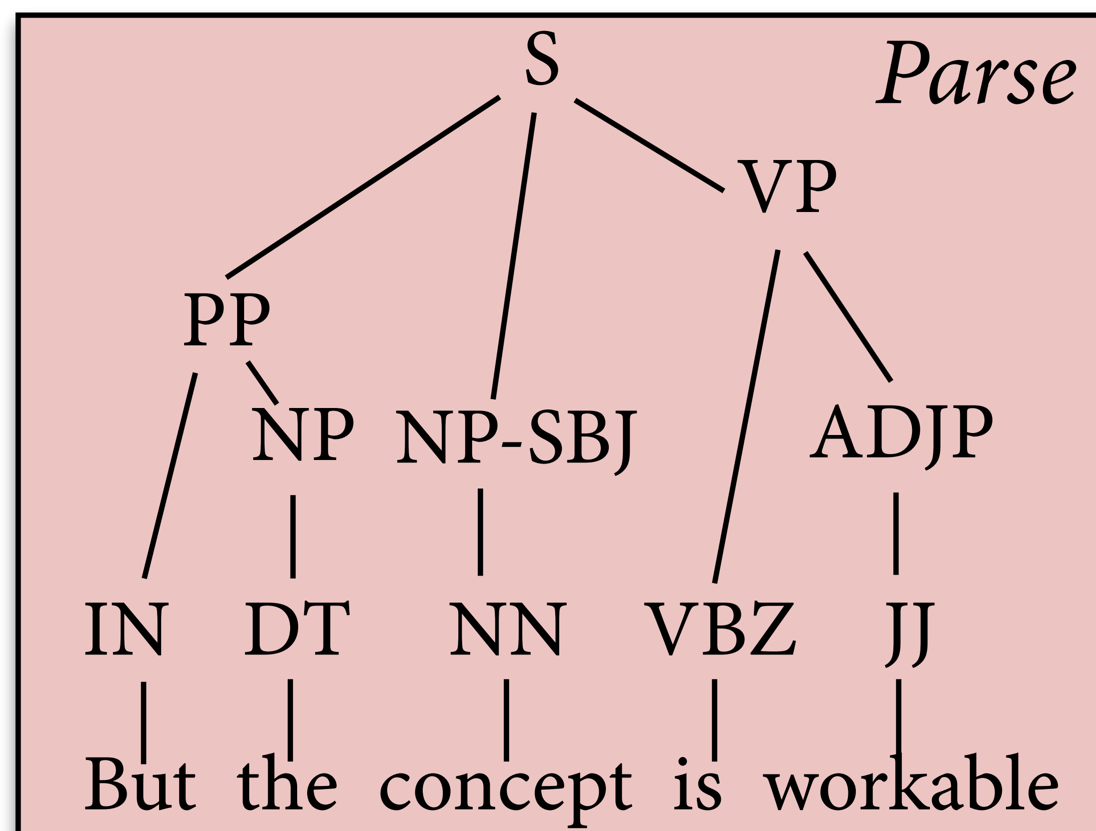
- Straightforward idea: Measure "exact match", i.e. proportion of gold standard trees that parser got right.

- This is too strict:
  - parser makes many decisions in parsing a sentence
  - a single incorrect parsing decision makes tree "wrong"
  - want more fine-grained measure

Comparing parse trees

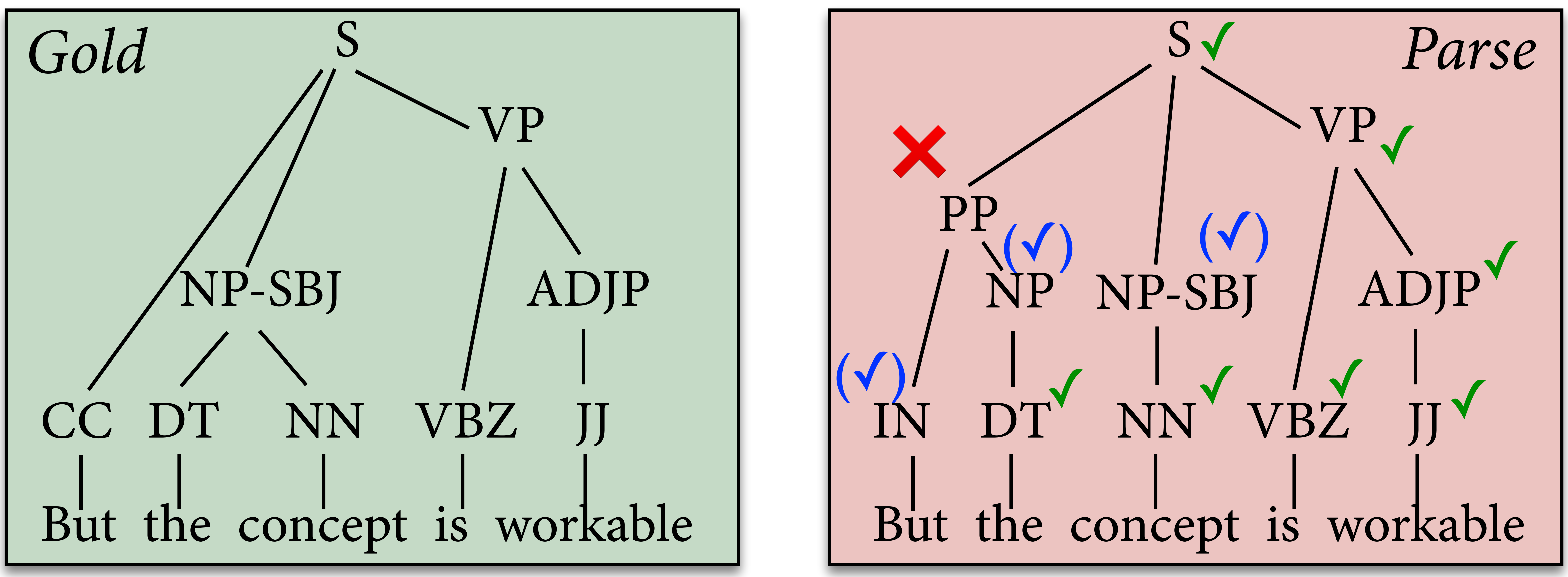
- Idea 2 (PARSEVAL): Compare *structure* of parse tree and gold standard tree.
  - Labeled: Which *constituents* (span + syntactic category) of one tree also occur in the other?
  - Unlabeled: How do the trees bracket the *substrings* of the sentence (ignoring syntactic categories)?





Precision

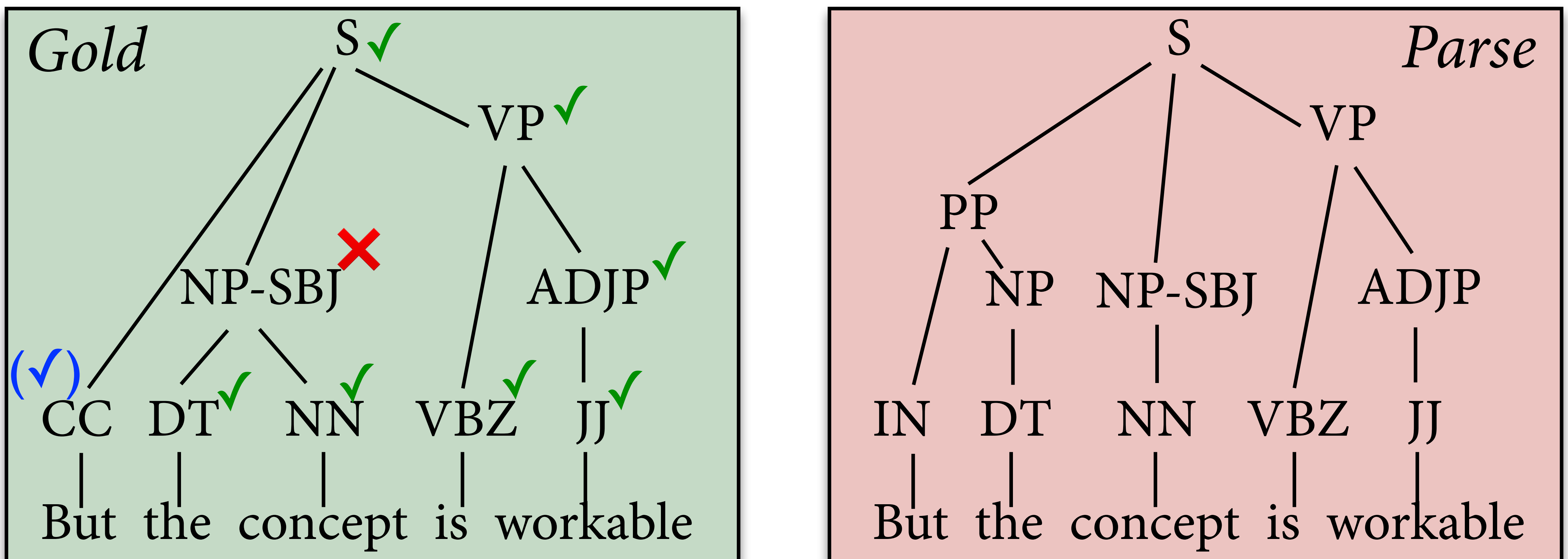
What proportion of constituents in *parse tree* is also present in *gold tree?*



Labeled Precision = 7 / 11 = 63.6%
Unlabeled Precision = 10 / 11 = 90.9%

Recall

What proportion of constituents in *gold tree* is also present in *parse tree?*



Labeled Recall = 7 / 9 = 77.8%
Unlabeled Recall = 8 / 9 = 88.9%

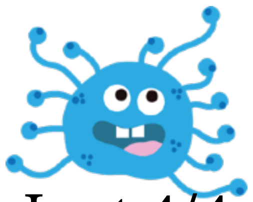
F-Score

- Precision and recall measure opposing qualities of a parser ("soundness" and "completeness")

- Summarize both together in the *f-score:*

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

- In the example, we have labeled f-score 70.0 and unlabeled f-score 89.9.



Last 4/4

Outline

1. Maximum likelihood estimation for PCFGs.
2. Unsupervised training: Hard and soft EM.
3. More accurate models for PCFG parsing.

ML Estimation

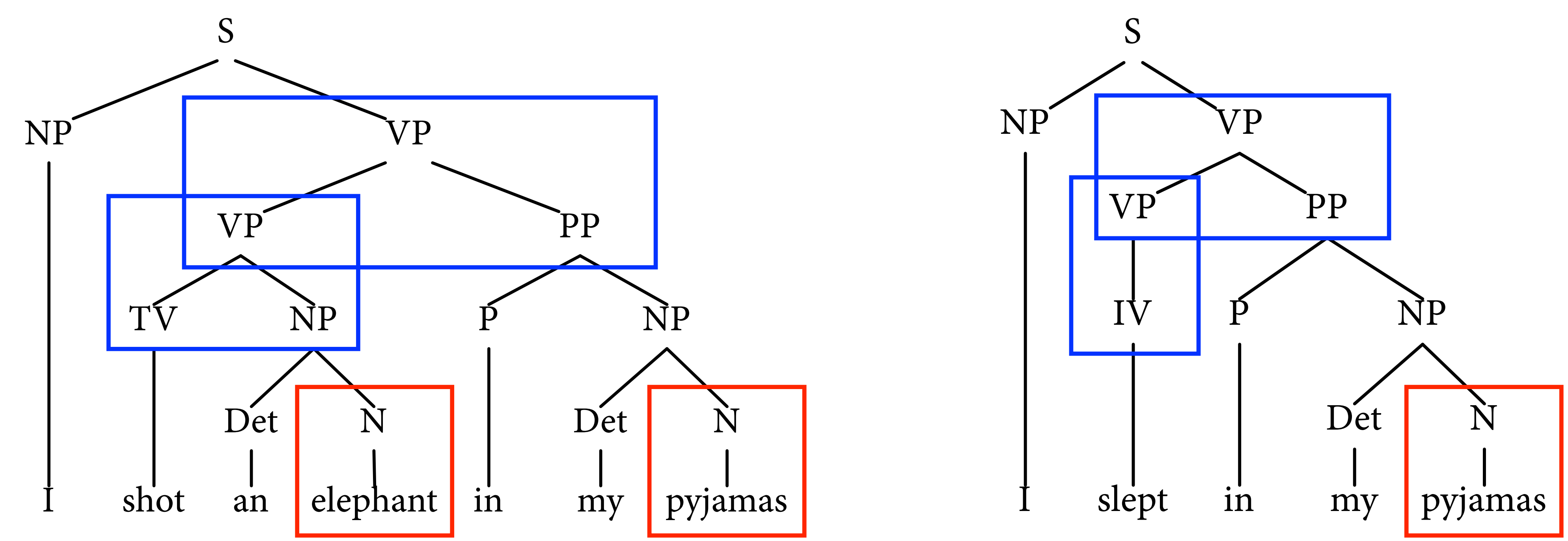
- Assume we have a treebank.
  - that is, every sentence annotated by hand with its "correct" parse tree

- Then we can use MLE to obtain rule probabilities:

$$P(A \to w) = \frac{C(A \to w)}{C(A \to \bullet)} = \frac{C(A \to w)}{\sum_{w'} C(A \to w')}$$

- Standard way of parameter estimation in practice. Works well, smoothing only needed for unknown words (or replace by POS tags).

Example



$N \rightarrow N\ PP$ [0]

$N \rightarrow$ elephant [1/3]

$N \rightarrow$ pyjamas [2/3]

$VP \rightarrow TV\ NP$ [1/4]

$VP \rightarrow IV$ [1/4]

$VP \rightarrow VP\ PP$ [1/2]

1/3

“Hard” aka Viterbi EM

- In the absence of syntactic annotations, learner must invent its own parse trees.

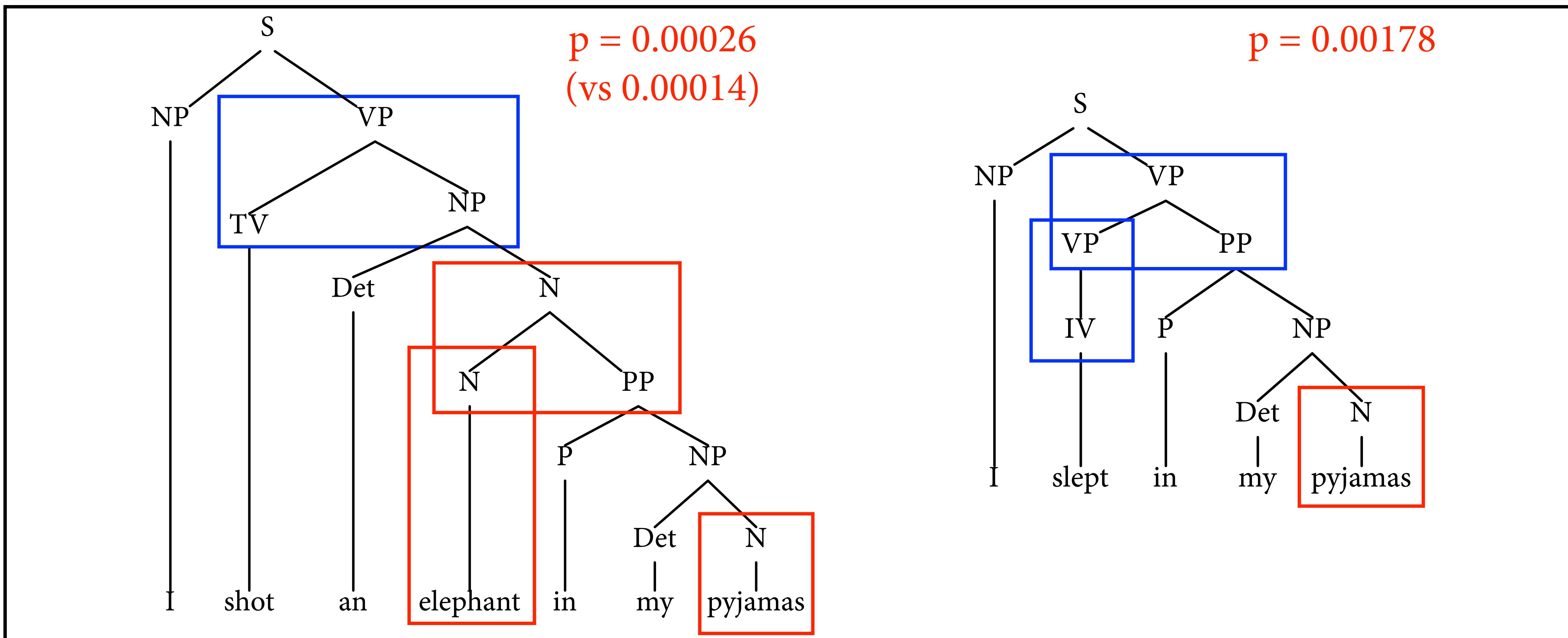
- Viterbi EM:
  - start with some parameter estimate
  - produce “syntactic annotations” by computing best tree for each sentence using Viterbi
  - apply MLE to re-estimate parameters
  - repeat as long as needed

- This is *not* real EM!

Example

1

N → N PP	[0.6]	VP → TV NP	[1/3]
N → elephant	[0.2]	VP → IV	[1/3]
N → pyjamas	[0.2]	VP → VP PP	[1/3]



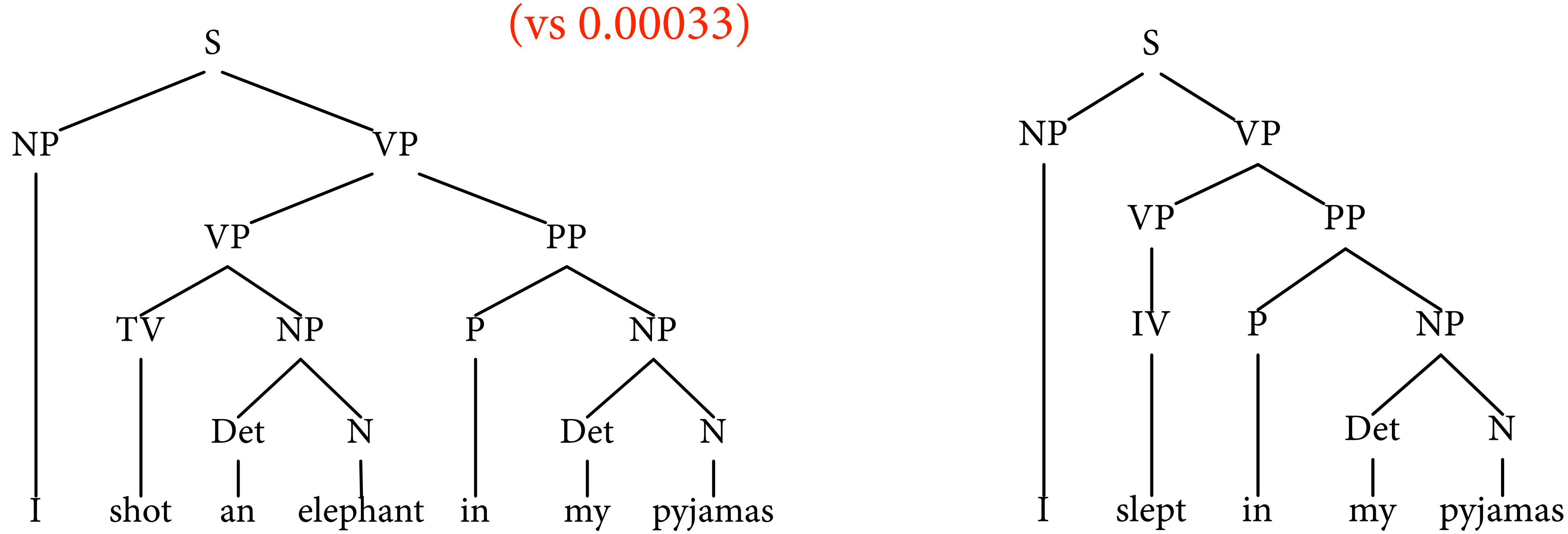
MLE on Viterbi parses

2

N → N PP	[1/4]	VP → TV NP	[1/3]
N → elephant	[1/4]	VP → IV	[1/3]
N → pyjamas	[1/2]	VP → VP PP	[1/3]

p = 0.00044
(vs 0.00033)

p = 0.00889



Some things to note

- In this example, the likelihood increased.
  - this need not always be the case for Viterbi EM
- Viterbi EM commits to a single parse tree per sentence. This has advantages and disadvantages:
  - parse tree easy to compute, and can simply apply MLE
  - ignores all uncertainty we had about correct parse (winning parse tree takes all)

"Real" (aka "soft") EM

idea: weighted counting of rules in all parse trees

Viterbi-EM

$$1 \cdot C_{t1}(r) \quad \Big| \quad + \ 0 \cdot C_{t2}(r) \quad + \ 0 \cdot C_{t3}(r) \quad + \ 0 \cdot C_{t4}(r)$$

EM

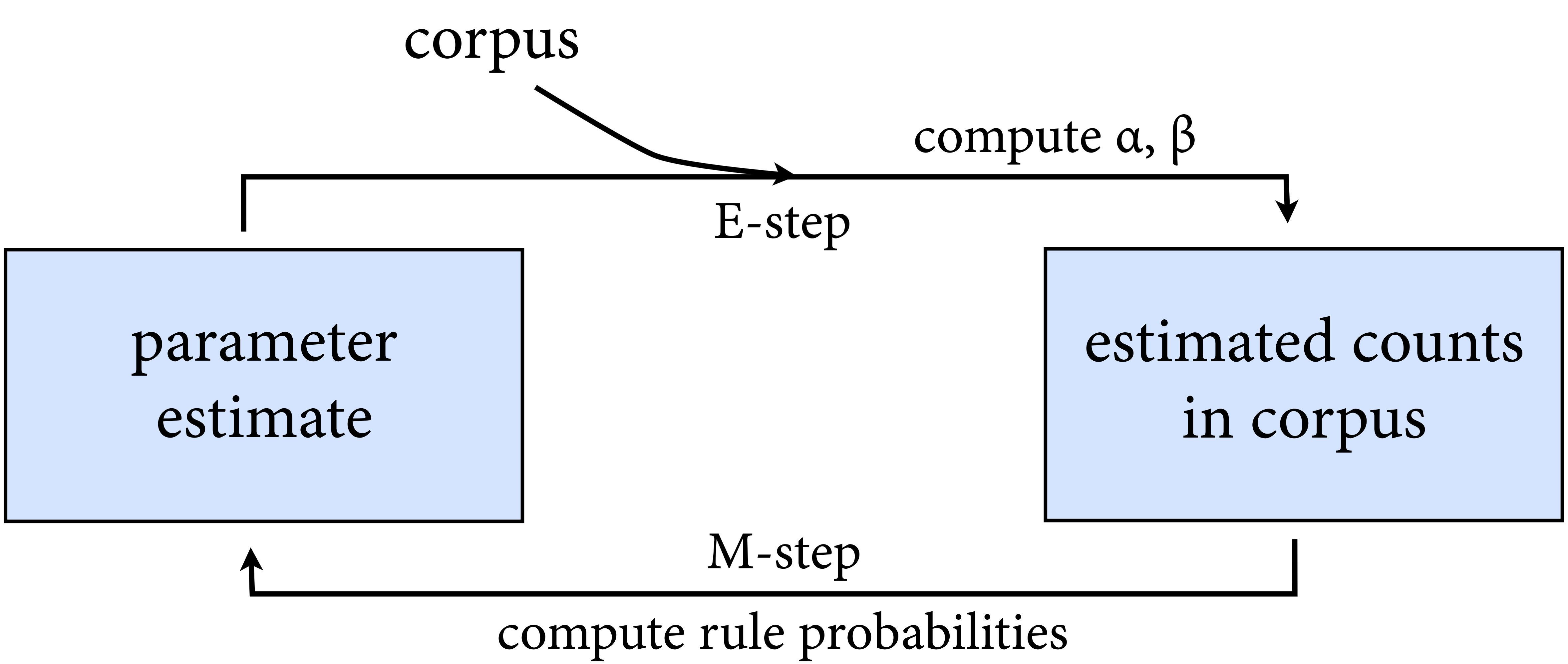
$$P(t_1 \mid w) \cdot C_{t1}(r) + P(t_2 \mid w) \cdot C_{t2}(r) + P(t_3 \mid w) \cdot C_{t3}(r) + P(t_4 \mid w) \cdot C_{t4}(r)$$

Inside-Outside Algorithm

- EM needs to sum over exponentially many parse trees → naive algorithm is infeasible.
- Similar rebracketing trick as in forward-backward permits doing EM loop in cubic time per sentence:
  - inside-outside algorithm
  - inside algorithm = Viterbi-CKY with sum instead of max

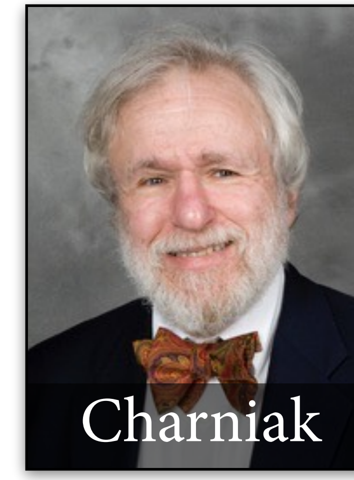
Inside-Outside Algorithm

Initialization: start with some estimate of parameters.

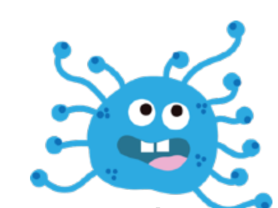


Continue computation until parameters don't change much.

Some remarks



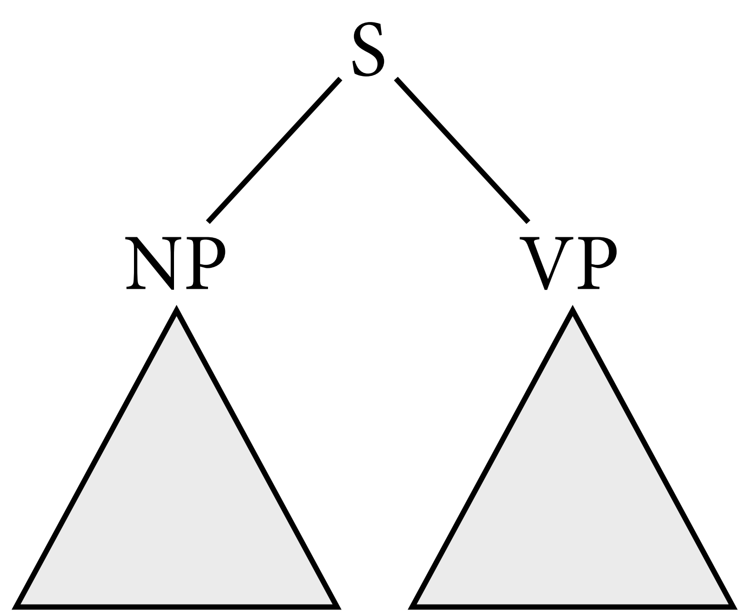
- Inside-outside increases likelihood in each step.
- But huge problems with local maxima.
  - Carroll & Charniak 92 find 300 different local maxima for 300 different initial parameter estimates.
- Therefore, EM doesn't really work for totally unsupervised PCFG training.



2/3

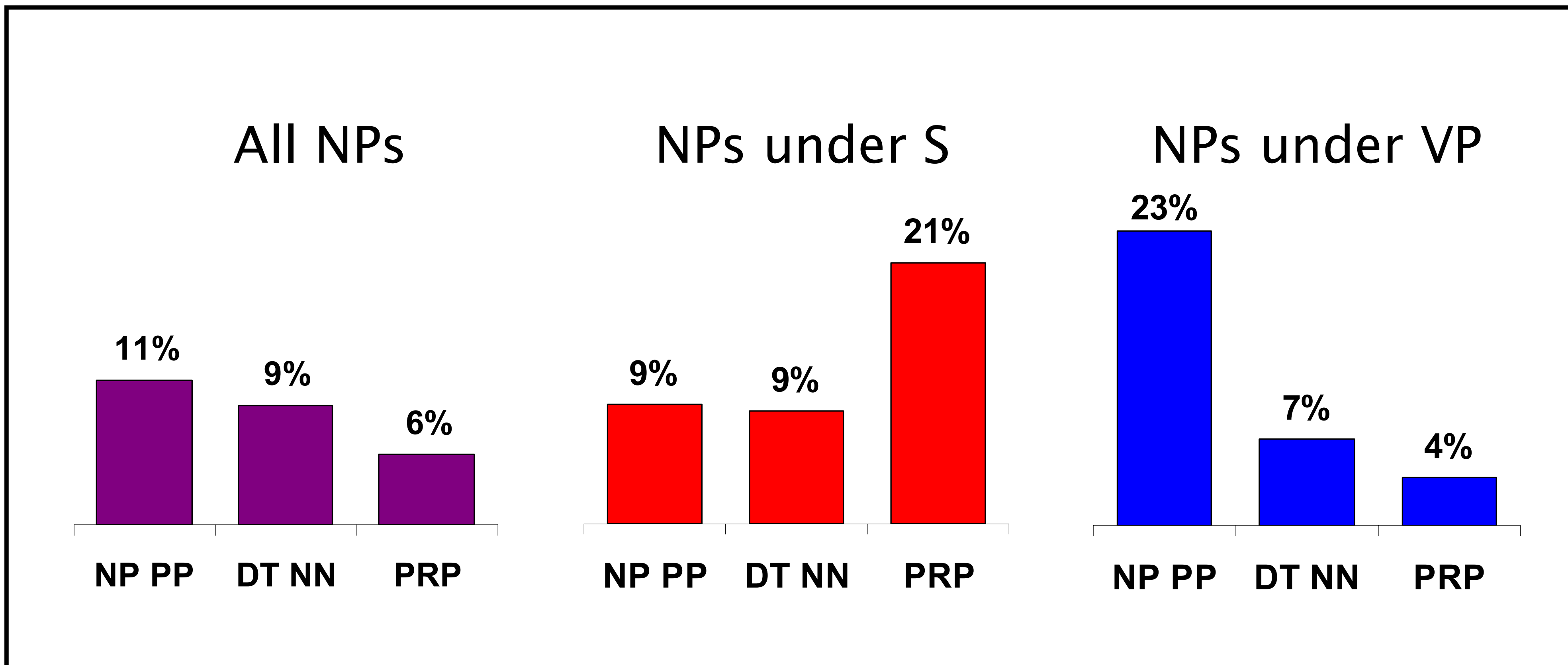
Fundamental problem of PCFGs

- Context-free grammar: One rule can only “see” parent and its children, not anything above or below.
- PCFG: Assumes statistical independence of all rewrite events.



NP → PRP?
NP → Det N?

Independence assumptions

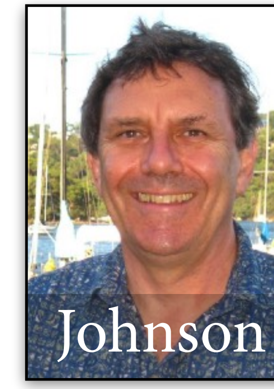


PTB statistics, from slides by Dan Klein

Independence assumptions

- Accurate disambiguation of PP attachment requires lexical information.
  - I shot the elephant with a long trunk.
  - I shot the elephant with a long rifle.
- PP attachment influenced by choice of P.
  - Collins note: “workers dumped sacks *into* a bin”
  - *into*-PPs in PTB 9x more likely to attach to VP than to N
- PCFGs rely on nonterminals alone,
  cannot “see” lexical information.

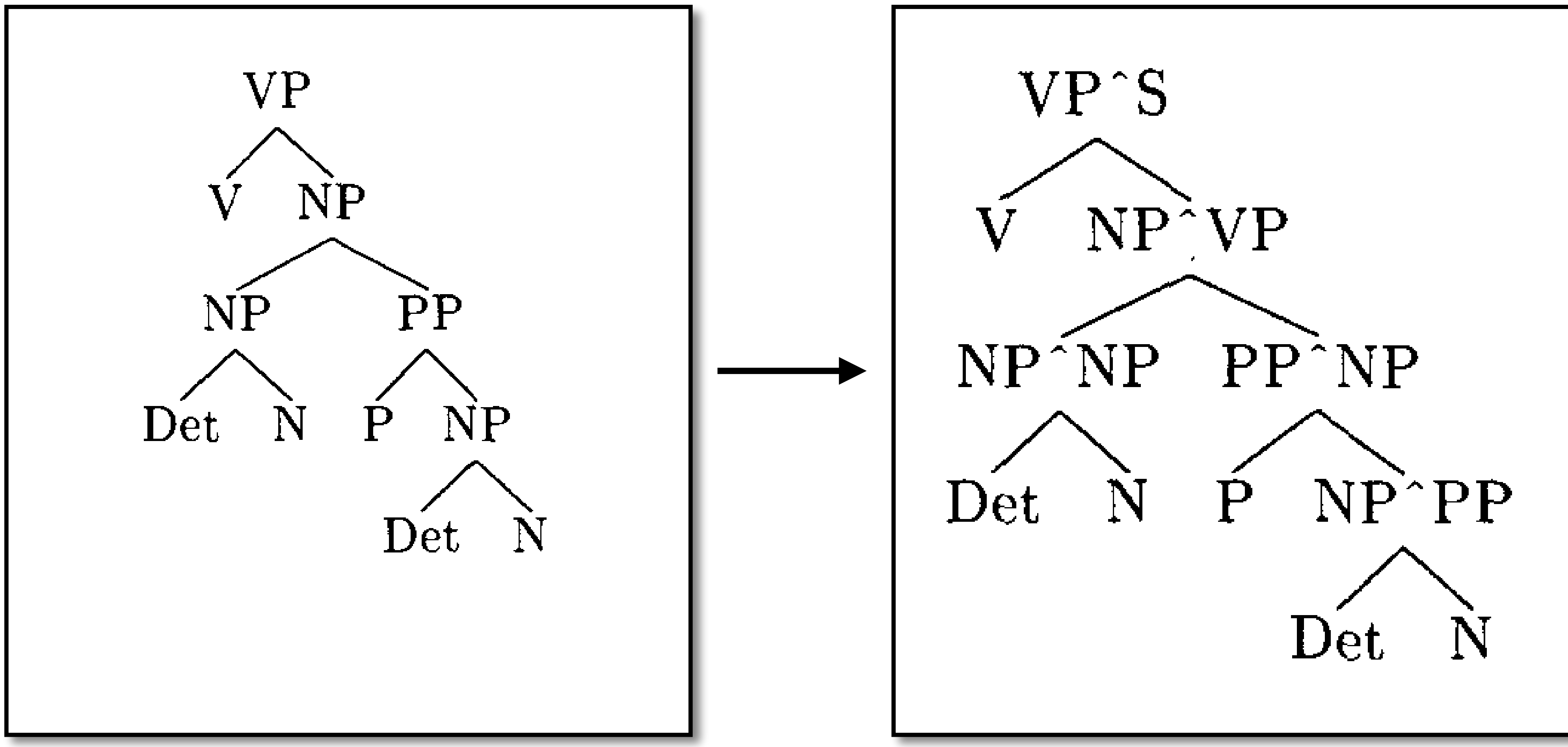
Parent annotations



- Discusses PTB preprocessing and impact of PTB representation changes.

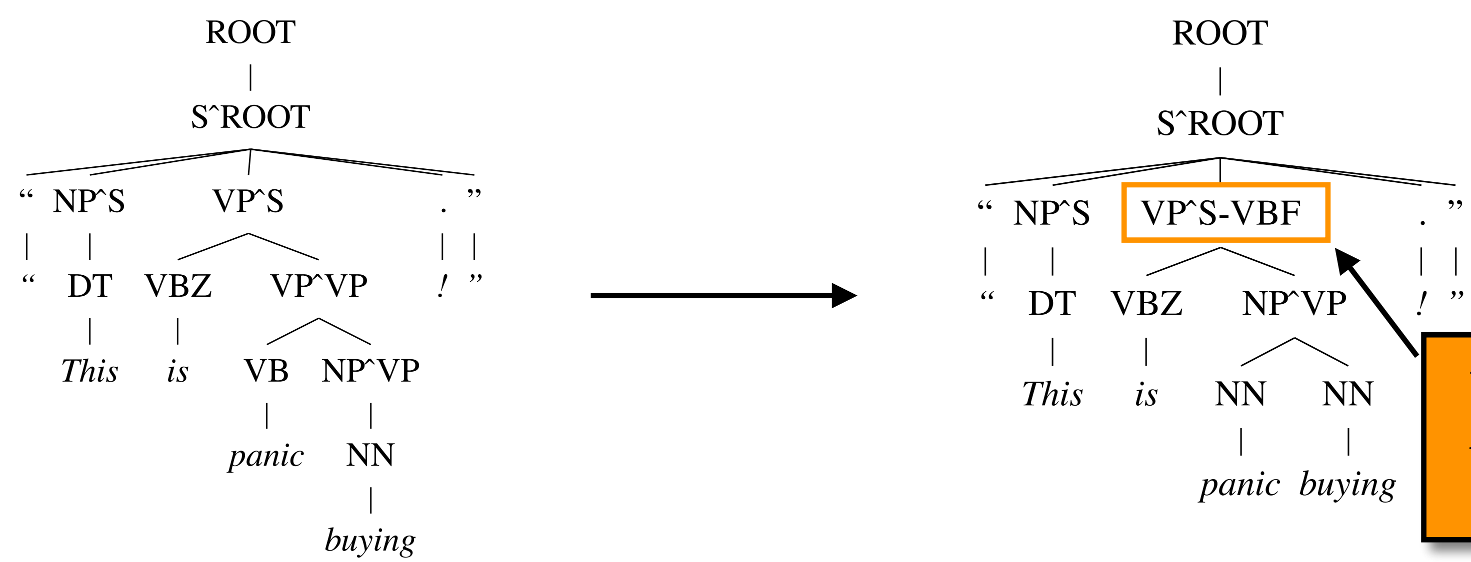
- One key idea: *parent annotations* (Johnson 1998).
    - If parent of NP makes such a difference in how it should be expanded, why don't we encode the parent of the NP?
    - Replace nonterminal NP by NP^S (NP as child of S), NP^VP (NP as child of VP), and so on in PTB trees.
    - Train grammar on modified treebank.
      After parsing, remove annotations and compare to gold standard tree.

Example

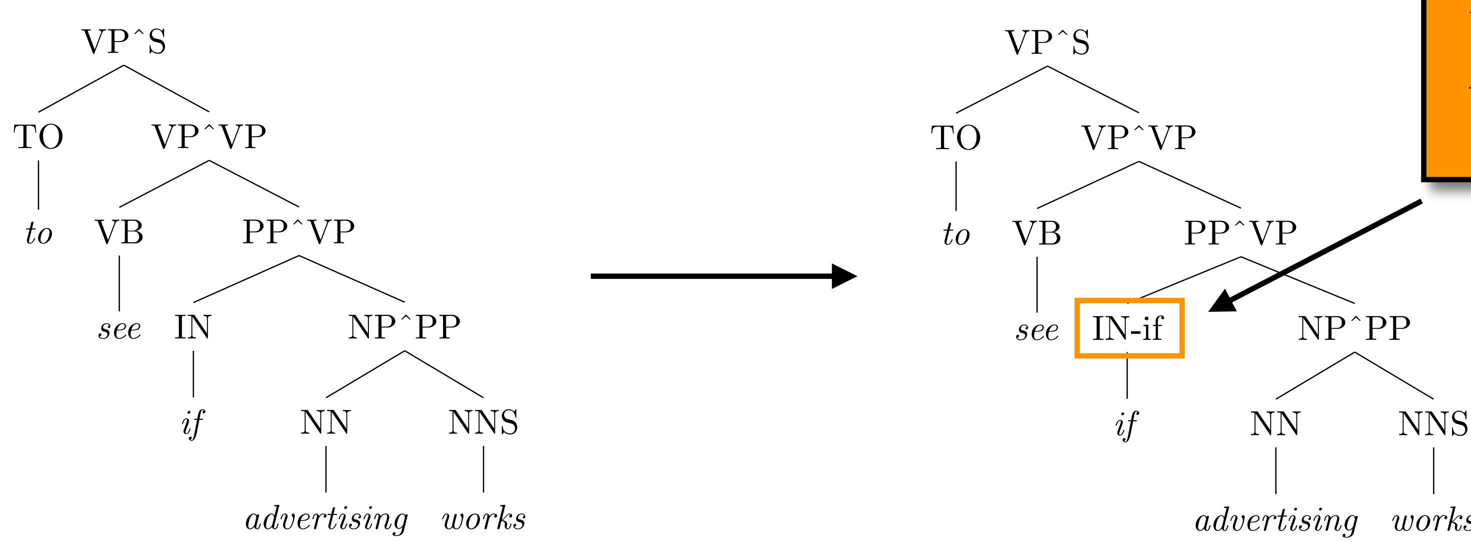


Result: Labeled f-score on Section 22 jumps from 71.5 to 79.6.
Number of production rules grows from 15,000 to 22,000.

Rule-based state splitting



Not just an VP^S,
but one whose head is a finite verb.



Not just a preposition,
but one that is like “if”.

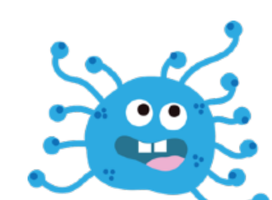
Results

Baseline grammar:
- smart parent annotations
- smart binarizations

Compares favorably to *lexicalized* parsers, in which nonterminals are labeled with head words (Collins, f-score ~89).

Accuracy can be improved further by automatically splitting nonterminals, using EM (Petrov & Klein).

	Cumulative			Indiv.
Annotation	Size	F_1	Δ F_1	Δ F_1
Baseline ($v \leq 2, h \leq 2$)	7619	77.77	–	–
UNARY-INTERNAL	8065	78.32	0.55	0.55
UNARY-DT	8066	78.48	0.71	0.17
UNARY-RB	8069	78.86	1.09	0.43
TAG-PA	8520	80.62	2.85	2.52
SPLIT-IN	8541	81.19	3.42	2.12
SPLIT-AUX	9034	81.66	3.89	0.57
SPLIT-CC	9190	81.69	3.92	0.12
SPLIT-%	9255	81.81	4.04	0.15
TMP-NP	9594	82.25	4.48	1.07
GAPPED-S	9741	82.28	4.51	0.17
POSS-NP	9820	83.06	5.29	0.28
SPLIT-VP	10499	85.72	7.95	1.36
BASE-NP	11660	86.04	8.27	0.73
DOMINATES-V	14097	86.91	9.14	1.42
RIGHT-REC-NP	$\boxed{15276}$	$\boxed{87.04}$	9.27	1.94



Summary

- PCFGs that we read off of treebank suffer from overly strong independence assumptions.
- Improve parser accuracy by encoding context in nonterminal vocabulary.
  - parent annotations
  - lexicalization
  - rule-based and automatically computed state splitting
- PCFG-based Berkeley parser: f-score ~90
  Neural Berkeley parser: f-score ~96.
  - The future may be neurosymbolic.