

Language and Statistics II

Lecture 12: Probabilistic context-free grammars

Next Topic: Parsing

1. Motivations for parsing.
2. A really simple model for parsing: PCFG
 - PCFGs as a stochastic process
 - Relationship to other models we know
 - Naïve training of PCFGs

Parsing: Motivation

- Language modeling (Chelba & Jelinek, 1998) ... predict next word given left syntactic context (**syntax**) instead of previous two words (**trigram**):

My friend who eats cookies {love, loves} ...

Or, transformations on data:

- Machine translation (Alshawi, 1996; Wu, 1997; Chiang, 2005 ...)
- Information extraction (Hobbs, 1997; Viola & Narasimhan, 2005)
- Grammar checking (obviously!)
- NL interfaces to databases (Collins and Zettlemoyer, 2005)
- Lexical learning (Lin, 1997)

Why are there so many parsing papers?

- Parsing is hard! (cf. [tagging](#))
 - Low 90s right now, by most measures of accuracy
- So many theories ...
- Easy to evaluate given annotated data
 - Evaluation is uncontroversial & automatic (cf. [machine translation](#))
- Great problem for structured prediction
- Lots of room for magic: smoothing, search, model refinement
- History: parsing was **the** major problem in CL before the empirical paradigm shift - seen as crucial for “understanding”

Major Research Questions

- What's the right **representation**?
- What's the right **model**?
- How to learn to parse **empirically**?
- How to make parsers **efficient**?

First Answers

- What's the right **representation**? phrase structure
- What's the right **model**? PCFG
- How to learn to parse **empirically**? MLE/treebank
- How to make parsers **fast**? CKY/Earley's algorithm
- How to incorporate structure **downstream**? best parse

Context-Free Grammar

- Alphabet Σ
- Set of variables N
- Start symbol $S \in N$
- Rewrite rules: $X \rightarrow \alpha$, where $X \in N$ and $\alpha \in (N \cup \Sigma)^*$

CNF: Assume $\alpha \in N^2 \cup \Sigma$.

A CFG

• $S \rightarrow NP VP$

• $VP \rightarrow V NP$

• $VP \rightarrow V' PP$

• $V' \rightarrow V NP$

• $NP \rightarrow Det N'$

• $N' \rightarrow N PP$

• $PP \rightarrow P NP$

• $NP \rightarrow Det N$

• $NP \rightarrow I$

• $V \rightarrow \text{saw}$

• $V \rightarrow \text{hit}$

• $Det \rightarrow \text{the}$

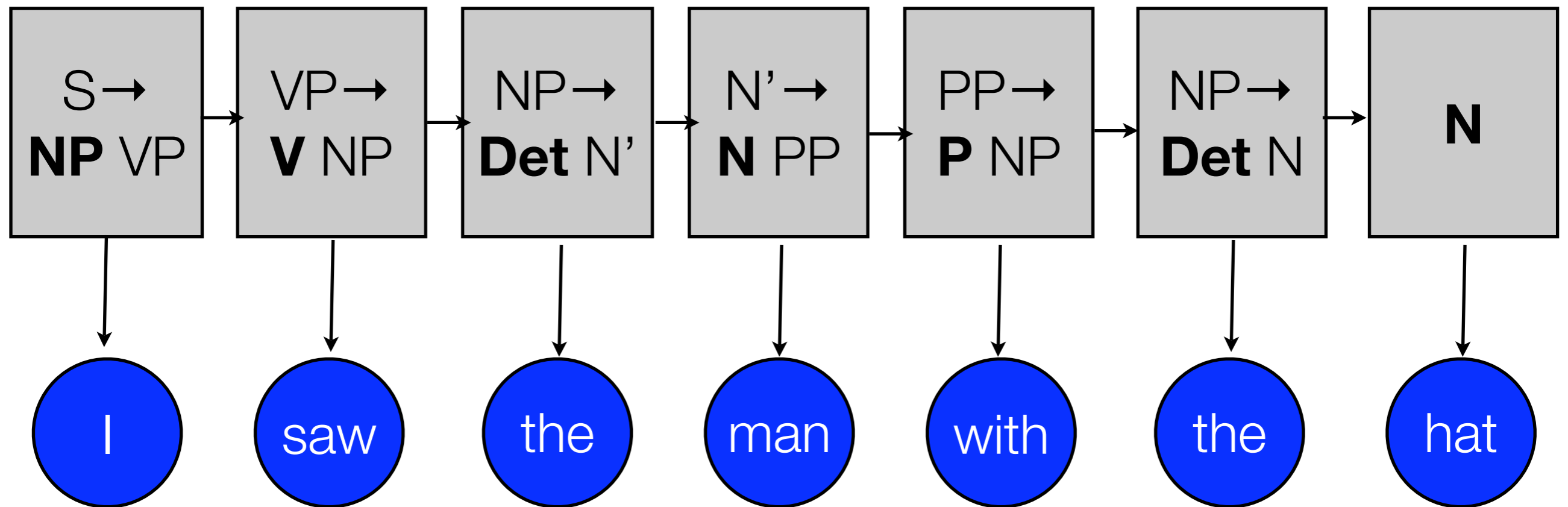
• $N \rightarrow \text{man}$

• $P \rightarrow \text{with}$

• $N \rightarrow \text{hat}$

• $N \rightarrow \text{bat}$

Right-branching tree as HMM derivation



Another Example

I hit the man with the bat

Disambiguation

S → NP VP

NP → I

V → hit

Det → the (2)

N → man

PP → P NP

P → with

NP → Det N

N → bat

VP → V NP

NP → Det N'

N' → N PP

VP → V' PP

NP → Det N

V' → V NP

Probabilistic CFG

- Alphabet Σ
- Set of variables N
- Start symbol $S \in N$
- Rewrite rules: $X \xrightarrow{p} \alpha$,
where $X \in N$, $\alpha \in (N \cup \Sigma)^*$, and $p \in \mathbb{R}_{\geq 0}$.
- All p s for a given X sum to one.

CNF: Assume $\alpha \in N^2 \cup \Sigma$.

PCFG as a Generative Process

- Instantiate the start symbol S as a single node and color it red.
- While there are any red symbols:
 - Choose a red node X and color it white.
 - Draw $\alpha = \langle \alpha_1, \alpha_2, \dots, \alpha_k \rangle$ according to the distribution defined by $X \xrightarrow{p} \bullet$.
 - Add $\langle \alpha_1, \alpha_2, \dots, \alpha_k \rangle$ to the tree as the sequence of children of the X you selected.
 - For any α_i that are nonterminals, color them red; color the terminals white.

Mathematical Properties

- The probability of generating a tree is simply a product of the **rule probabilities** for all rule tokens in the tree.
- Given a tree, it's $O(n)$ to compute the probability.
- The red-node-choosing policy doesn't matter as long as it's consistent. It doesn't affect the probabilities.
- Independence assumption?

PCFGs and HMMs

PCFG:

- Alphabet Σ
- Nonterminal set N
- Start nonterminal S
- Rules $X \xrightarrow{p} \alpha$

HMM (special case):

- Alphabet Σ
- State set N
- Start state S_0
- Rules
 - $X \xrightarrow{\eta(s | X)} s X'$
 - $X' \xrightarrow{\gamma(Y | X)} Y$
 - $X' \xrightarrow{\gamma(\text{stop} | X)} \varepsilon$

PCFGs and Log-Linear Models

Log-linear model:

- Set of inputs \mathcal{X}
- Set of outputs \mathcal{Y}
- Set of feature functions f_i :
 $(\mathcal{X}, \mathcal{Y}) \rightarrow \mathbb{R}_{\geq 0}$
- Set of weights θ_i
corresponding to f_i

PCFG:

- Σ^*
- Derivable productions
given the rules
- Counts of rules
- Logarithms of rule
probabilities

Major Research Questions

- ✓ What's the right **representation**?
- ✓ What's the right **model**?
(We've talked about one representation
and one model.)
- How to learn to parse **empirically**?
- How to make parsers **efficient**?

Learning from Data

1. Where do the **rules** come from?
2. Where do the rule **probabilities** come from?

First answer: Look at a huge collection of trees (a treebank).

$X \rightarrow \alpha$ is in the grammar iff it's in the treebank.

$p(\alpha \mid X)$ is proportional to the count of $X \rightarrow \alpha$.

Penn Treebank (Marcus et al., 1993)

- A million words (40K sentences) of *Wall Street Journal* text (late 1980s).
- Parsed by experts; consensus parse for each sentence was published.
- The structure is basically what you'd expect from a PCFG.
 - Tends to be “flat” where there's controversy.
 - Some “traces” for extraposed elements.

Example Tree

```
( (S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken) )
    ( , , )
    (ADJP
      (NP (CD 61) (NNS years) )
      (JJ old) )
    ( , , ) )
  (VP (MD will)
    (VP (VB join)
      (NP (DT the) (NN board) )
      (PP-CLR (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director) ))
      (NP-TMP (NNP Nov.) (CD 29) )))
  ( . . ) ))
```

```

( (S
  (NP-SBJ-1
    (NP (NNP Rudolph) (NNP Agnew) )
    ( , , )
    (UCP
      (ADJP
        (NP (CD 55) (NNS years) )
        (JJ old) )
      (CC and)
      (NP
        (NP (JJ former) (NN chairman) )
        (PP (IN of)
          (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) )))
      ( , , ) )
    (VP (VBD was)
      (VP (VBN named)
        (S
          (NP-SBJ (-NONE- *-1) )
          (NP-PRD
            (NP (DT a) (JJ nonexecutive) (NN director) )
            (PP (IN of)
              (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate) ))
            )
          )
        )
      )
    )
  )
  ( . . ) )
)

```

Evaluating Parsers

- Take a sentence from the test set.
- Use your parser to propose a **hypothesis** parse.
- Treebank gives you the **correct** parse.
- How to compare?
 - {unlabeled, labeled} × {precision, recall}
 - crossing brackets statistics
 - evalb (<http://nlp.cs.nyu.edu/evalb>)
- Significance testing ...

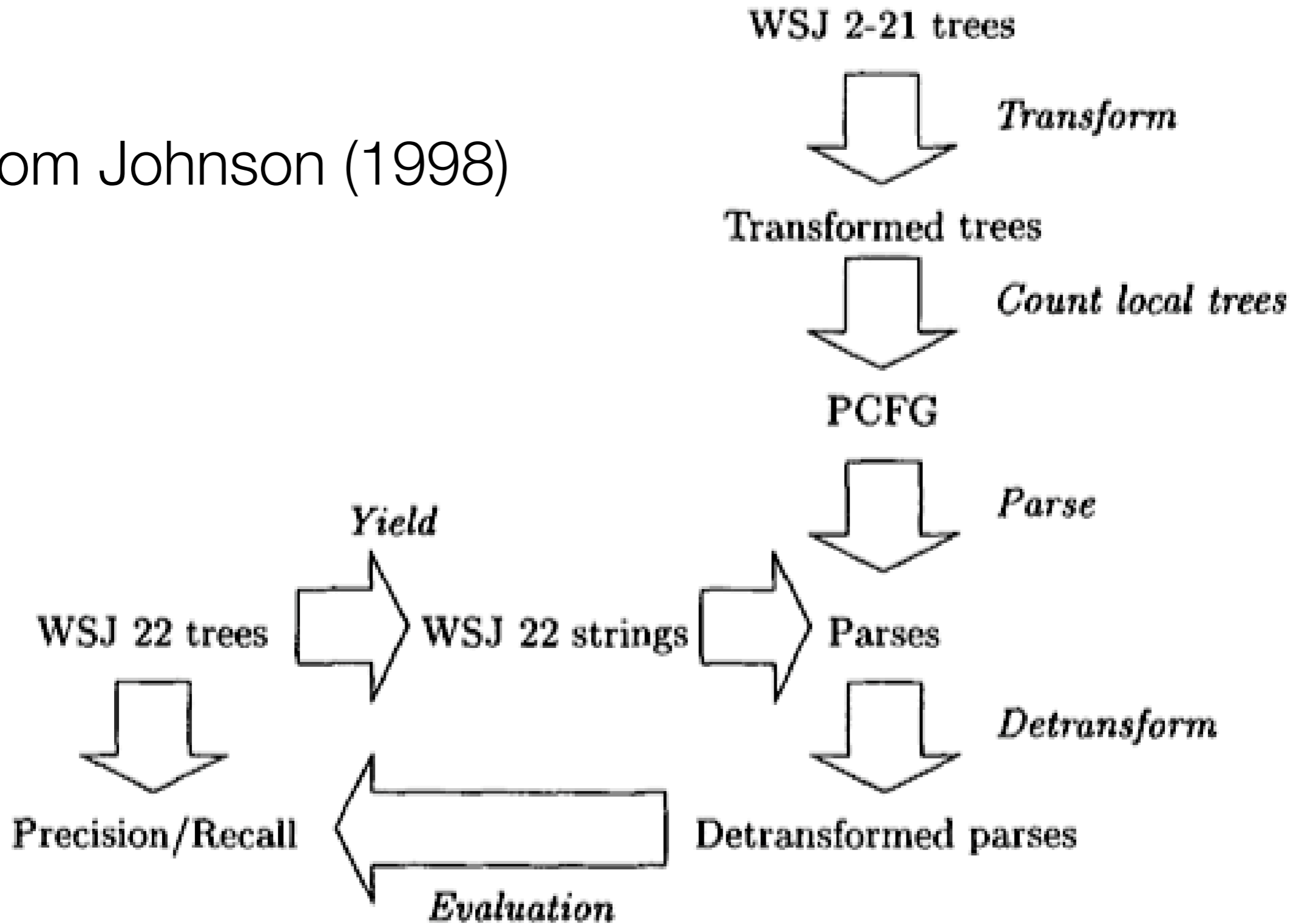
The Dark Side

- This is **the** way to train and test an English parser.
- There are some inconsistencies.
- Other treebank builders haven't always been as diligent; often tag labels, nonterminal labels, conventions are assumed to **port** to other languages.
- Better way of handling disagreement: publish different annotators' trees (not consensus)?

Training Parsers In Practice

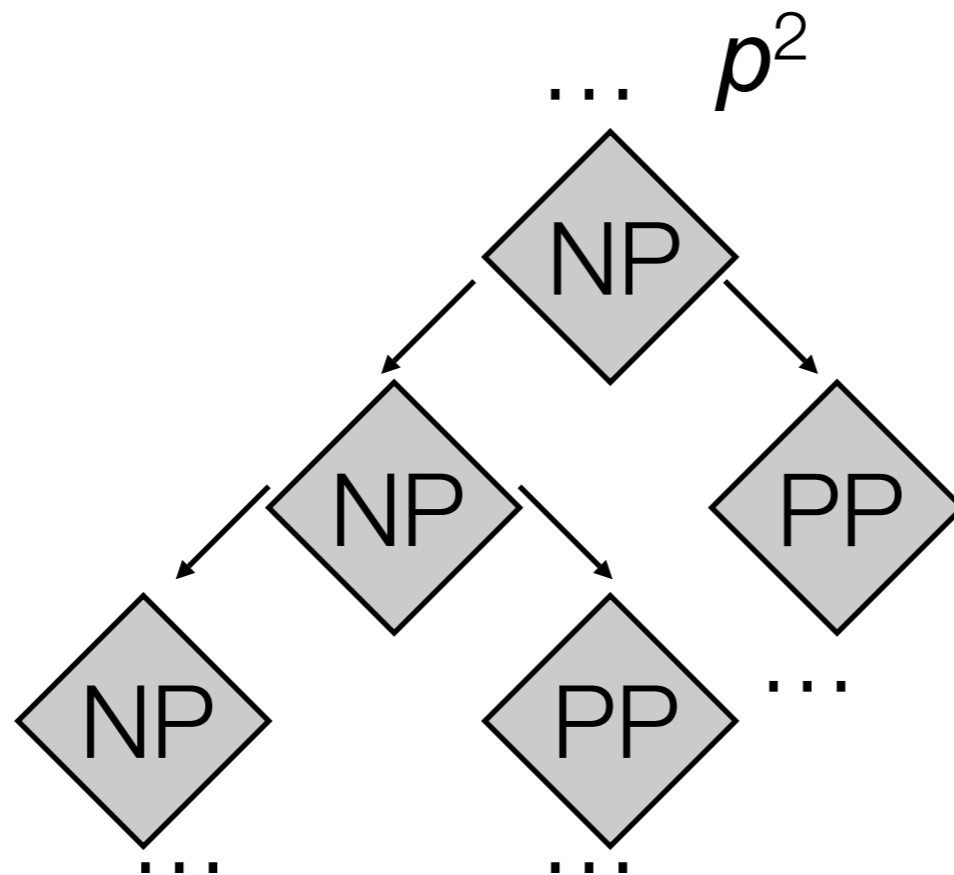
- Transformations on trees
 - Some of these are generally taken to be crucial
 - Some are widely debated
 - Lately, people have started **learning** these transformations
- Smoothing (crucial)
- We will come back to this as we explore some current state-of-the-art parsers.
 - Collins (1999; 2003)
 - Charniak (2000)
 - Klein and Manning (2003)
 - McDonald, Pereira, Ribarov, and Hajic (2005)

from Johnson (1998)

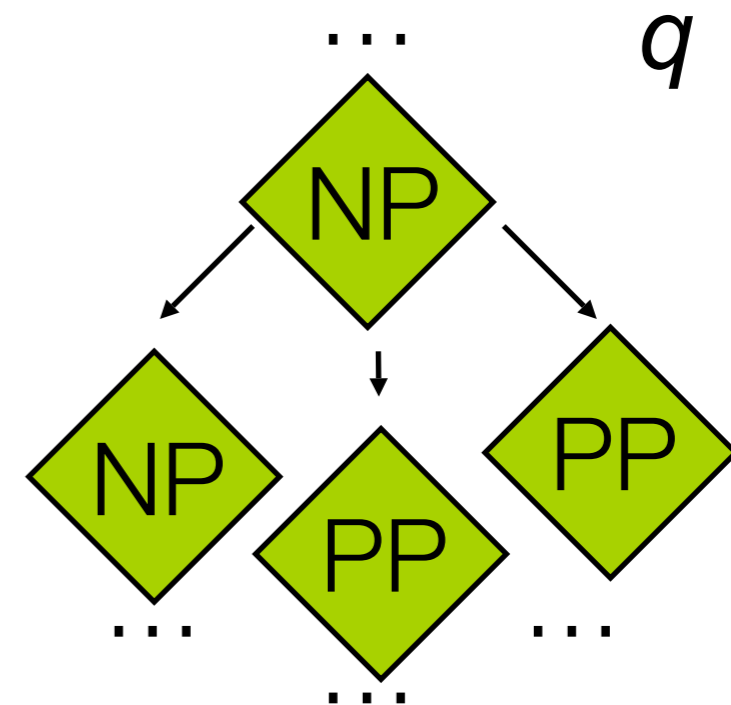


Parent Annotation

NP \rightarrow^p NP PP



NP \rightarrow^q NP PP PP

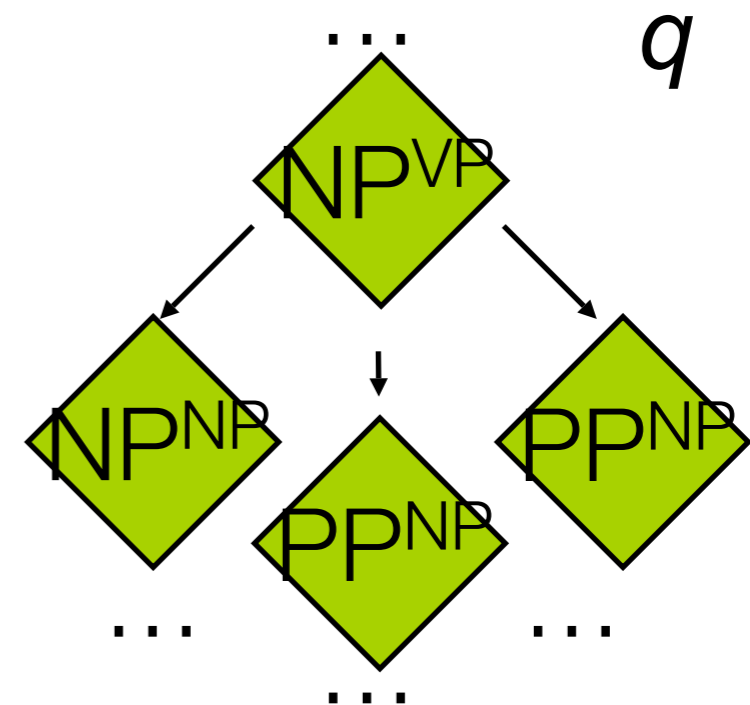
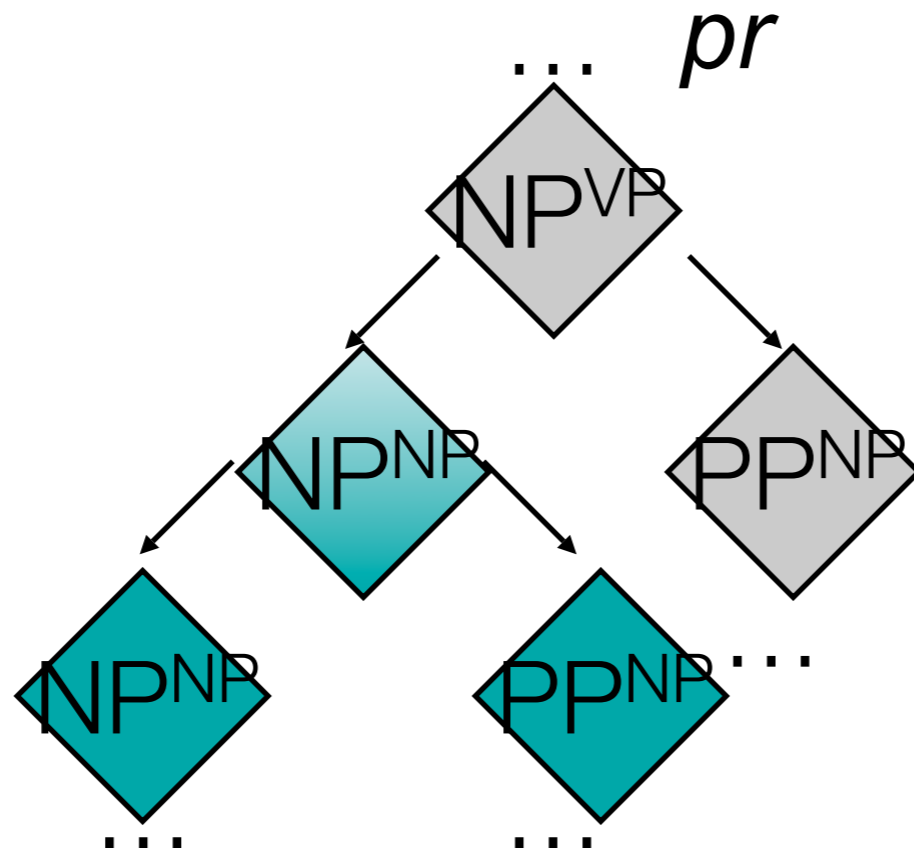


Parent Annotation

$NPVP \xrightarrow{p} NPNP \ PPNP$

$NPNP \xrightarrow{r} NPNP \ PPNP$

$NPVP \xrightarrow{q} NPNP \ PPNP \ PPNP$



Parent Annotation

- Another way to think about it ...

Before:
$$p(\text{tree}) = \prod_{n \in \text{nodes}(\text{tree})} \rho(\text{childsequence}(n) \mid n)$$

Now:
$$p(\text{tree}) = \prod_{n \in \text{nodes}(\text{tree})} \rho(\text{childsequence}(n) \mid n, \text{parent}(n))$$

- This could conceivably **help** performance (weaker independence assumptions)
- This could conceivably **hurt** performance (data sparseness)

Parent Annotation

- From Johnson (1998):

PCFG from WSJ Treebank: 14,962 rules

- Of those, 1,327 would **always** be subsumed!

After parent annotation: 22,773 rules

- Only 965 would always be subsumed!

Recall 69.7% → 79.2%; precision 73.5% → 80.0%

- Trick: check for subsumed rules, remove them from the grammar → faster parsing.

Head Annotation

“I love all my children, but one of them is **special.**”

S → NP VP

VP → VBD NP

NP → DT NNS PP

Heads not in the Treebank.

Usually people use **deterministic head rules** (Magerman, 1995).

Lexicalization

- Every nonterminal node is annotated with a word from its yield; such that

$$\text{lex}(n) = \text{lex}(\text{head}(n))$$

Lexicalization

- Every nonterminal node is annotated with a word from its yield; such that

$$\text{lex}(n) = \text{lex}(\text{head}(n))$$

- What might this allow?
- What might we worry about?

Currently, this is controversial (we'll see why)!

Dependencies

- Take away the nonlexical parts.
- Merge redundant nodes with the same head.

Crucial Point

- By “decorating” the treebank, we have been carrying additional information around the trees.
- The **hope** is to improve the ability of a PCFG to predict syntactic structure correctly.
- The **worry** is that our grammar will get really big and the probabilities too hard to estimate.
 - Also, speed. More rules → bigger grammar → slower parsing.