# Probabilistic Parsing in Practice

*Lecture #15*

**Computational Linguistics**
**CMPSCI 591N, Spring 2006**

Andrew McCallum

(including slides from Michael Collins, Chris Manning, Jason Eisner, Mary Harper)

# Today’s Main Points

- Training data
- How to evaluate parsers
- Limitations of PCFGs, enhancements & alternatives
  - Lexicalized PCFGs
  - Structure sensitivity
  - Left-corner parsing
  - Faster parsing with beam search
  - Dependency parsers
- Current state of the art

# Treebanks

- Pure Grammar Induction Approaches tend not to produce the parse trees that people want

- Solution
  - Give a some example of parse trees that we want
  - Make a learning tool learn a grammar

- Treebank
  - A collection of such example parses
  - PennTreebank is most widely used

# Treebanks

- Penn Treebank
  - Trees are represented via bracketing
  - Fairly flat structures for Noun Phrases
    (NP Arizona real estate loans)
  - Tagged with grammatical and semantic functions
    (-SBJ , –LOC, …)
  - Use empty nodes(*) to indicate understood subjects and extraction gaps

```
(  ( S ( NP-SBJ  The move)
     ( VP  followed
          ( NP  ( NP a round )
               ( PP  of
                   (NP  ( NP similar increases )
                        ( PP by
                            ( NP other lenders ) )
                        ( PP against
                            ( NP Arizona real estate loans )))))
          ,
           ( S-ADV ( NP-SBJ * )
                  ( VP  reflecting
                        ( NP a continuing decline )
                        ( PP-LOC  in
                                  (NP  that market ))))))
  . )
```

# Treebanks

- Many people have argued that it is better to have linguists constructing treebanks than grammars

- Because it is easier
  - to work out the correct parse of sentences
- than
  - to try to determine what all possible manifestations of a certain rule or grammatical construct are

# Treebanking Issues

- Type of data
  - Task dependent (newspaper, journals, novels, technical manuals, dialogs, email)
- Size
  - The more the better!  (Resource-limited)
- Parse representation
  - Dependency vs Parse tree
  - Attributes.  What do encode? words, morphology, syntax, semantics...
  - Reference & bookkeeping: date time, who did what

# Organizational Issues

- Team
  - 1 Team leader; bookkeeping/hiring
  - 1 Guideline person
  - 1 Linguistic issues person
  - 3-5 Annotators
  - 1-2 Technical staff/programming
  - 2 Checking persons
- Double annotation if possible.

# Treebanking Plan

- The main points (after getting funding)
  - Planning
  - Basic guidelines development
  - Annotation & guidelines refinement
  - Consistency checking, guidelines finalization
  - Packaging and distribution

- Time needed
  - on the order of 2 years per 1 million words
  - only about 1/3 of the total effort is annotation

# Parser Evaluation

# Evaluation

Ultimate goal is to build system for IE, QA, MT

- People are rarely interested in syntactic analysis for its own sake
- Evaluate the system for evaluate the parser

For Simplicity and modularization, and Convenience

- Compare parses from a parser with the result of hand parsing of a sentence(gold standard)

What is objective criterion that we are trying to maximize?

# Evaluation

Tree Accuracy (Exact match)

- It is a very tough standard!!!
- But in many ways it is a sensible one to use

PARSEVAL Measures

- For some purposes, partially correct parses can be useful
- Originally for non-statistical parsers
- Evaluate the component pieces of a parse
- Measures : Precision, Recall, Crossing brackets

# Evaluation

## (Labeled) Precision

How many brackets in the parse match those in the correct tree (Gold standard)?

## (Labeled) Recall

How many of the brackets in the correct tree are in the parse?

## Crossing brackets

Average of how many constituents in one tree cross over constituent boundaries in the other tree

```
B1                      (                               )
B2              (                     )
B3      (                                                       )
B4                                              (               )
          w1      w2      w3      w4      w5      w6      w7      w8
```

# Problems with PARSEVAL

Even vanilla PCFG performs quite well

It measures success at the level of individual decisions

You must make many consecutive decisions correctly to be correct on the entire tree.

# Problems with PARSEVAL (2)

Behind story

The structure of Penn Treebank

Flat → Few brackets → Low Crossing brackets

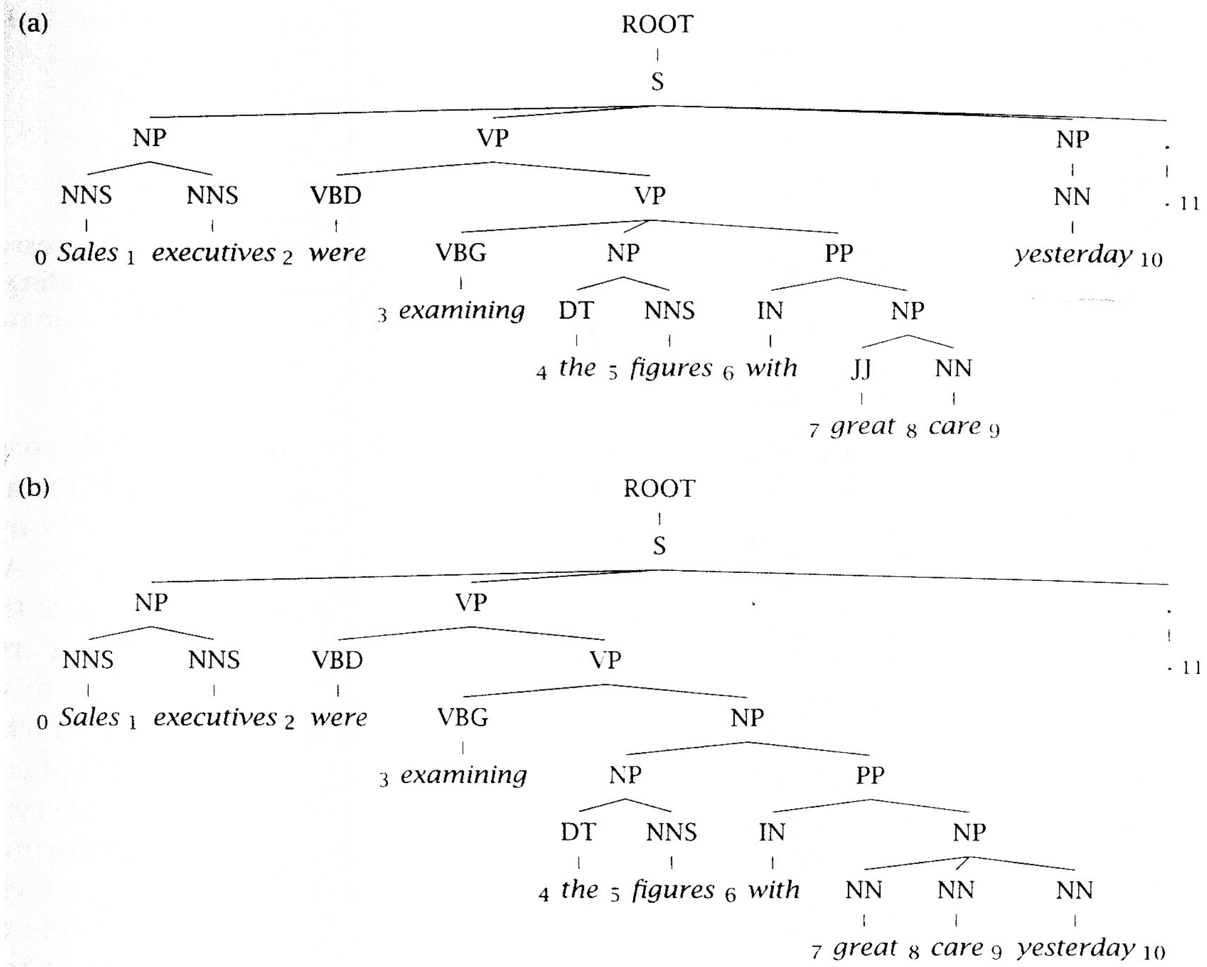
Troublesome brackets are avoided

→ High Precision/Recall

The errors in precision and recall are minimal

In some cases wrong PP attachment penalizes Precision, Recall and Crossing Bracket Accuracy minimally.

On the other hand, attaching low instead of high, then every node in the right-branching tree will be wrong: serious harm



(c) Brackets in gold standard tree (a.):
**S-(0:11), NP-(0:2),** VP-(2:9), VP-(3:9), **NP-(4:6),** PP-(6-9), NP-(7,9), *NP-(9:10)

(d) Brackets in candidate parse (b.):
**S-(0:11), NP-(0:2),** VP-(2:10), VP-(3:10), NP-(4:10), **NP-(4:6),** PP-(6-10), NP-(7,10)

| (e) | | | | |
|---|---|---|---|---|
| | Precision: | 3/8 = 37.5% | Crossing Brackets: | 3 |
| | Recall: | 3/8 = 37.5% | Crossing Accuracy: | 62% |
| | Labeled Precision: | 3/8 = 37.5% | Tagging Accuracy: | 10/11 = 90.9% |
| | Labeled Recall: | 3/8 = 37.5% | | |

# Evaluation

Do PARSEVAL measures succeed in real tasks?

- Many small parsing mistakes might not affect tasks of semantic interpretation
  - (Bonnema 1996,1997)
    - Tree Accuracy of the Parser : 62%
    - Correct Semantic Interpretations : 88%
  - (Hermajakob and Mooney 1997)
    - English to German translation

At the moment, people feel PARSEVAL measures are adequate for the comparing parsers

# Lexicalized Parsing

# Limitations of PCFGs

- PCFGs assume:
  - Place invariance
  - Context free: P(rule) independent of
    - words outside span
    - *also, words with overlapping derivation*
  - Ancestor free: P(rule) independent of
    - *Non-terminals above.*

- Lack of sensitivity to lexical information
- Lack of sensitivity to structural frequencies

# Lack of Lexical Dependency

Means that

P(VP → V NP NP)

is independent of the particular verb involved!

... but much more likely with ditransitive verbs (like ***gave***).

*He **gave** the boy a ball.*

*He **ran** to the store.*

# The Need for Lexical Dependency

Probabilities dependent on Lexical words

Problem 1 : Verb subcategorization

VP expansion is independent of the choice of verb

However …

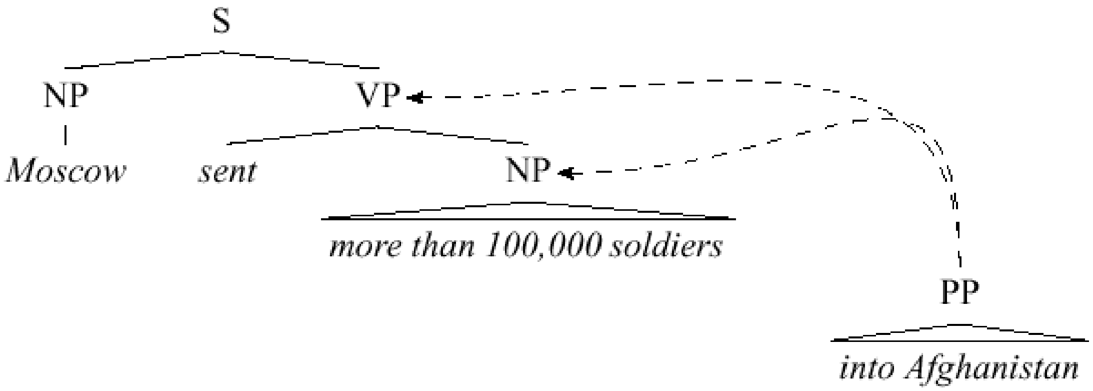
| | verb | | | |
|---|---|---|---|---|
| | come | take | think | want |
| VP -> V | 9.5% | 2.6% | 4.6% | 5.7% |
| VP -> V NP | 1.1% | 32.1% | 0.2% | 13.9% |
| VP -> V PP | 34.5% | 3.1% | 7.1% | 0.3% |
| VP -> V SBAR | 6.6% | 0.3% | 73.0% | 0.2% |
| VP -> V S | 2.2% | 1.3% | 4.8% | 70.8% |

Including actual words information when making decisions about tree structure is necessary

# Weakening the independence assumption of PCFG

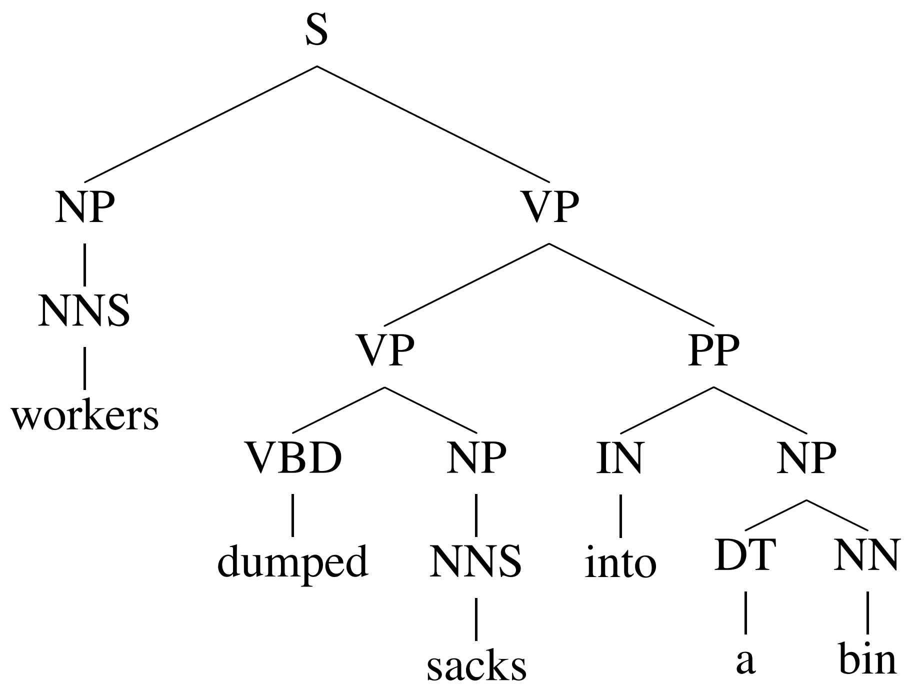
Probabilities dependent on Lexical words

- Problem 2 : Phrasal Attachment
  - Lexical content of phrases provide information for decision
  - Syntactic category of the phrases provide very little information
  - Standard PCFG is worse than n-gram models

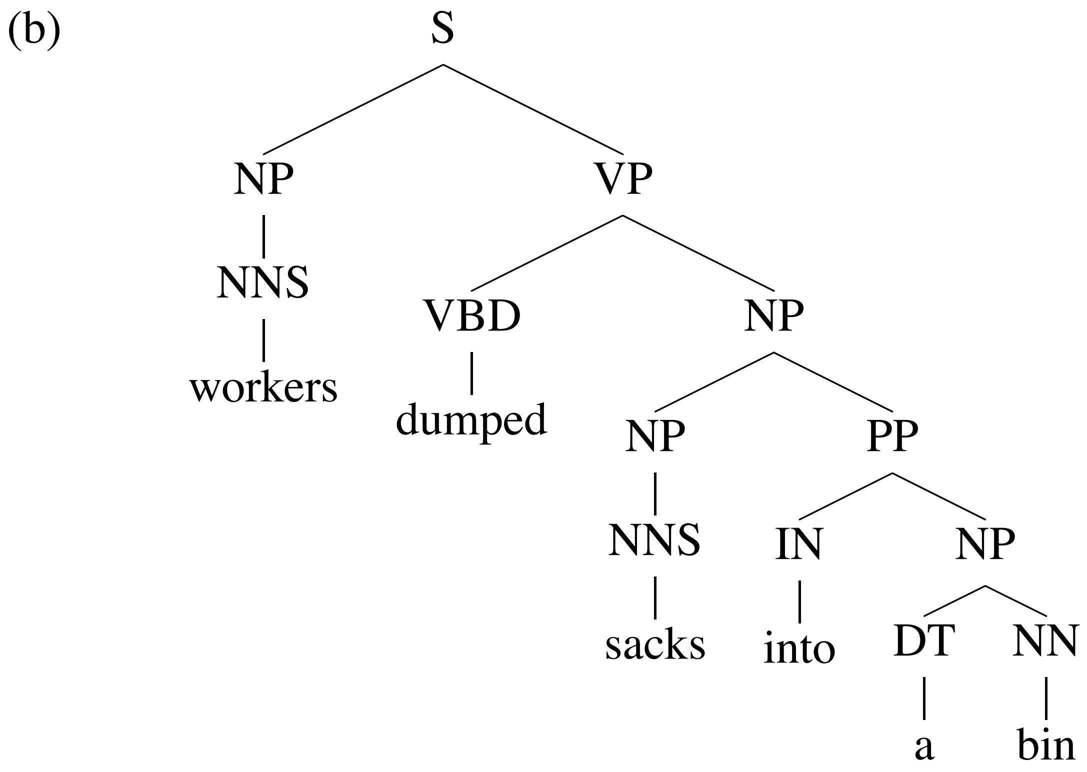


# Another case of PP attachment ambiguity

(a)



# Another case of PP attachment ambiguity



# Another case of PP attachment ambiguity

(a)

| Rules |
|---|
| S → NP VP |
| NP → NNS |
| **VP → VP PP** |
| VP → VBD NP |
| NP → NNS |
| PP → IN NP |
| NP → DT NN |
| NNS → workers |
| VBD → dumped |
| NNS → sacks |
| IN → into |
| DT → a |
| NN → bin |

(b)

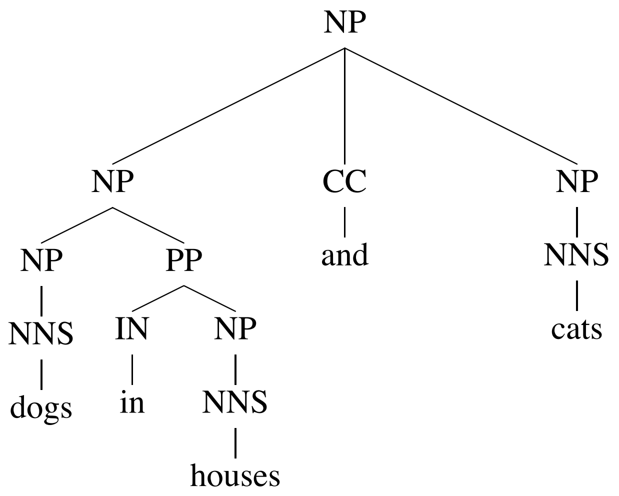
| Rules |
|---|
| S → NP VP |
| NP → NNS |
| **NP → NP PP** |
| VP → VBD NP |
| NP → NNS |
| PP → IN NP |
| NP → DT NN |
| NNS → workers |
| VBD → dumped |
| NNS → sacks |
| IN → into |
| DT → a |
| NN → bin |

If $P(\text{NP} \rightarrow \text{NP PP} \mid \text{NP}) > P(\text{VP} \rightarrow \text{VP PP} \mid \text{VP})$ then (b) is more probable, else (a) is more probable.

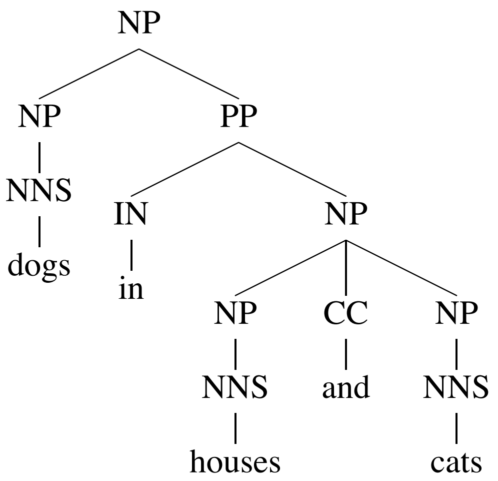
**Attachment decision is completely independent of the words**

# A case of coordination ambiguity

(a)



(b)



(a)

| Rules |
|---|
| NP → NP CC NP |
| NP → NP PP |
| NP → NNS |
| PP → IN NP |
| NP → NNS |
| NP → NNS |
| NNS → dogs |
| IN → in |
| NNS → houses |
| CC → and |
| NNS → cats |

(b)

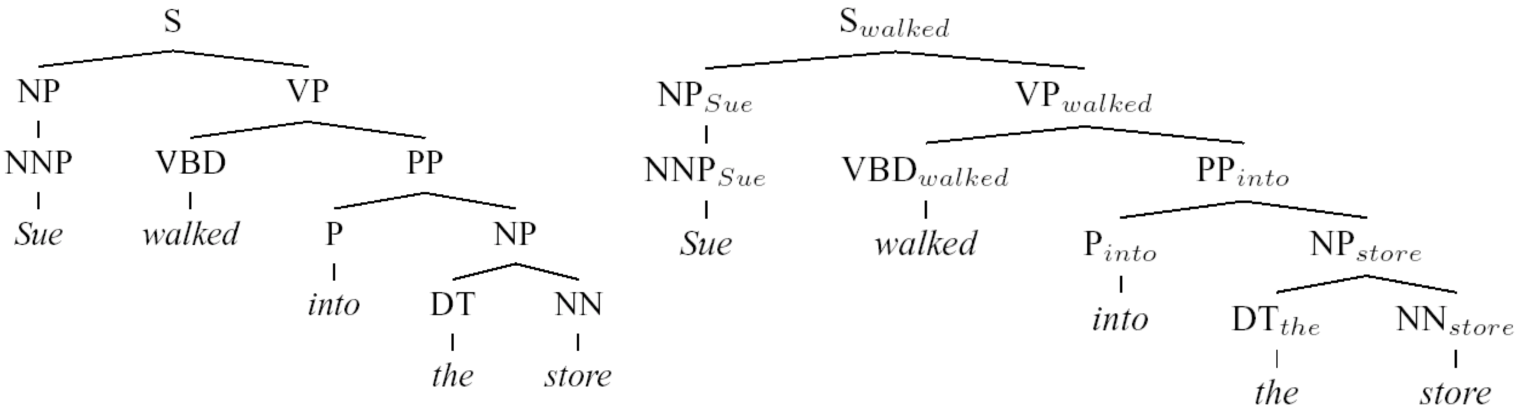
| Rules |
|---|
| NP → NP CC NP |
| NP → NP PP |
| NP → NNS |
| PP → IN NP |
| NP → NNS |
| NP → NNS |
| NNS → dogs |
| IN → in |
| NNS → houses |
| CC → and |
| NNS → cats |

**Here the two parses have identical rules, and therefore have identical probability under any assignment of PCFG rule probabilities**

# Weakening the independence assumption of PCFG

Probabilities dependent on Lexical words

- Solution
  - Lexicalize CFG : Each phrasal node with its head word



- Background idea
  - Strong lexical dependencies between heads and their dependents

# Heads in Context-Free Rules

**Add annotations specifying the “head” of each rule:**

| | | | |
|---|---|---|---|
| S | ⇒ | NP | **VP** |
| VP | ⇒ | **Vi** | |
| VP | ⇒ | **Vt** | NP |
| VP | ⇒ | **VP** | PP |
| NP | ⇒ | DT | **NN** |
| NP | ⇒ | **NP** | PP |
| PP | ⇒ | **IN** | NP |

| | | |
|---|---|---|
| Vi | ⇒ | sleeps |
| Vt | ⇒ | saw |
| NN | ⇒ | man |
| NN | ⇒ | woman |
| NN | ⇒ | telescope |
| DT | ⇒ | the |
| IN | ⇒ | with |
| IN | ⇒ | in |

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

# More about heads

- Each context-free rule has one "special" child that is the head of the rule. e.g.,

| | | | | | |
|---|---|---|---|---|---|
| S | ⇒ | NP | VP | | (VP is the head) |
| VP | ⇒ | Vt | NP | | (Vt is the head) |
| NP | ⇒ | DT | NN | NN | (NN is the head) |

- A core idea in linguistics
  (X-bar Theory, Head-Driven Phrase Structure Grammar)

- Some intuitions:
  - The central sub-constituent of each rule.
  - The semantic predicate in each rule.

# Rules which recover heads: Example rules for NPs

**If** the rule contains NN, NNS, or NNP:
Choose the rightmost NN, NNS, or NNP

**Else If** the rule contains an NP: Choose the leftmost NP

**Else If** the rule contains a JJ: Choose the rightmost JJ

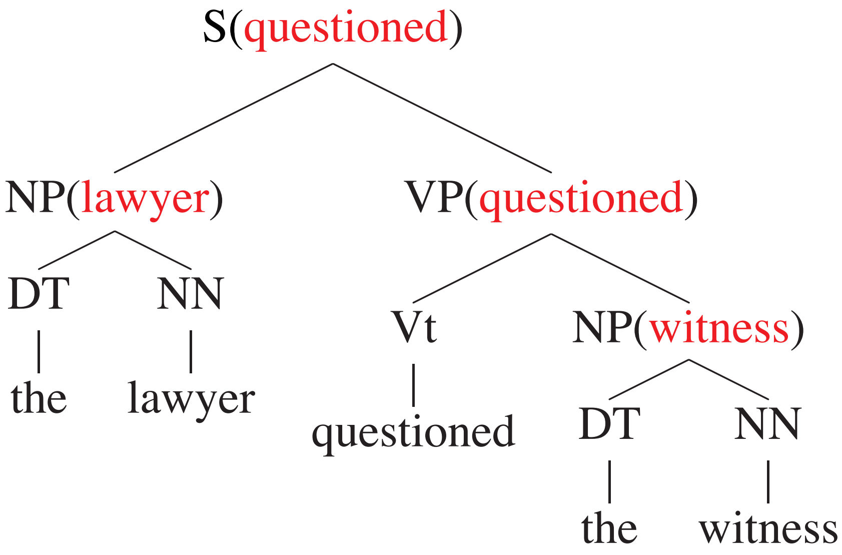
**Else If** the rule contains a CD: Choose the rightmost CD

**Else** Choose the rightmost child

e.g.,

NP ⇒ DT NNP NN
NP ⇒ DT NN NNP
NP ⇒ NP PP
NP ⇒ DT JJ
NP ⇒ DT

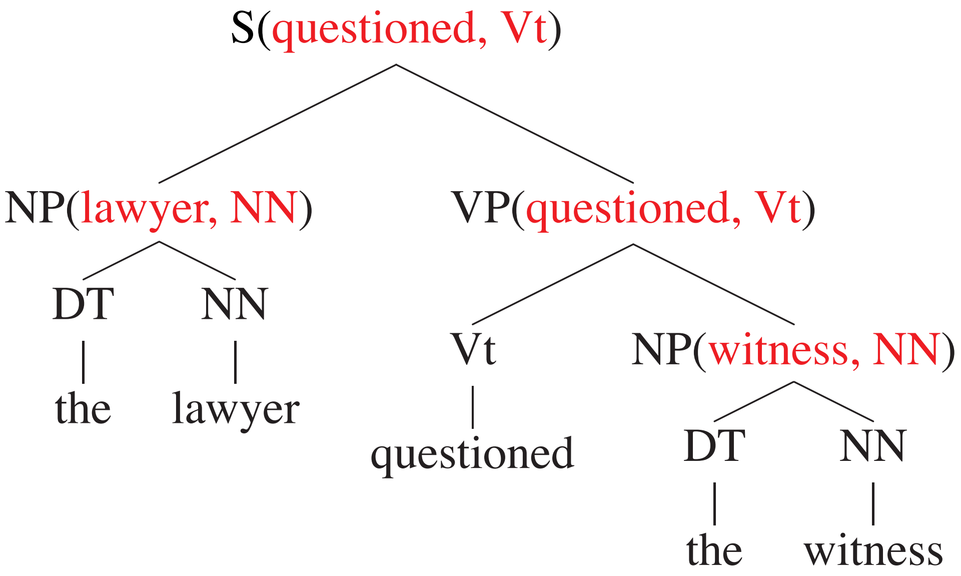
# Adding Headwords to Trees



- A constituent receives its headword from its **head child**.

| | | | | | |
|---|---|---|---|---|---|
| S | ⇒ | NP | VP | | (S receives headword from VP) |
| VP | ⇒ | Vt | NP | | (VP receives headword from Vt) |
| NP | ⇒ | DT | | NN | (NP receives headword from NN) |

# Adding Headtags to Trees



- Also propogate **part-of-speech tags** up the trees

# Explosion of number of rules

New rules might look like:

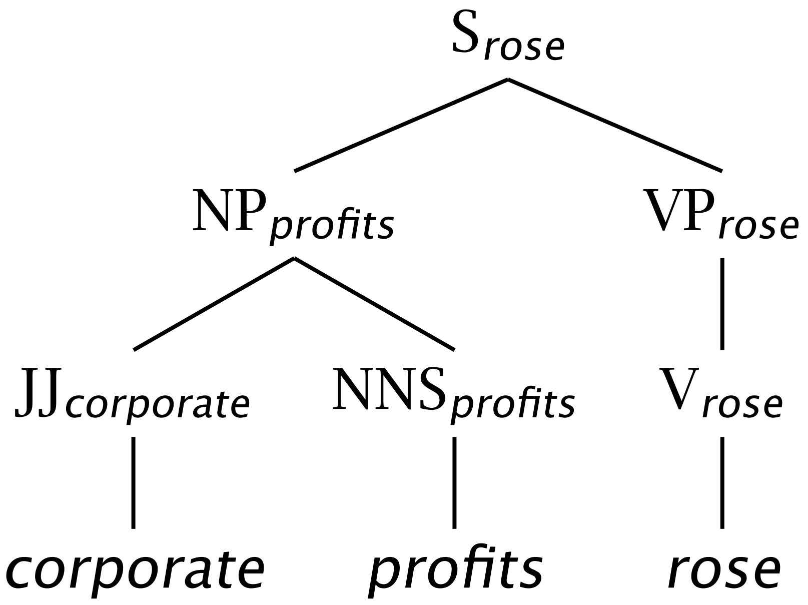
VP[gave] → V[gave] NP[man] NP[book]

But this would be a massive explosion in number of rules (and parameters)

# Lexicalized Parsing, with smoothing

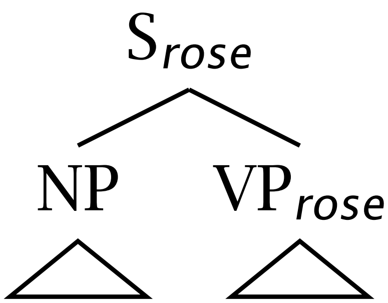
# Lexicalized parsing [Charniak 1997]

- A very simple, conservative model of lexicalized PCFG
- Probabilistic conditioning is “top-down” (but actual computation is bottom-up)



# [Charniak 1997]

## Generate head, then head constituent & rule

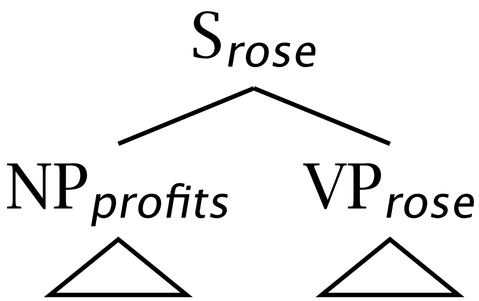


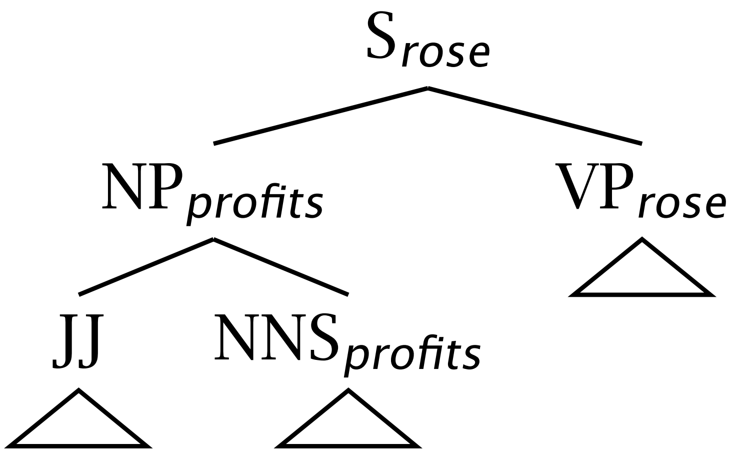
$h = \mathit{profits};\ c = \mathrm{NP}$

$ph = \mathit{rose};\ pc = \mathrm{S}$

$P(h|ph,c,pc)$

$P(r|h,c,pc)$





h=head word, c=head consituent

ph=parent head word, parent head constituent

# Smoothing in [Charniak 1997]

$$
\begin{aligned}
\hat{P}(h|ph,c,pc) &= \lambda_1(e)P_{\mathsf{MLE}}(h|ph,c,pc) \\
&\quad +\lambda_2(e)P_{\mathsf{MLE}}(h|C(ph),c,pc) \\
&\quad +\lambda_3(e)P_{\mathsf{MLE}}(h|c,pc) + \lambda_4(e)P_{\mathsf{MLE}}(h|c)
\end{aligned}
$$

- $\lambda_i(e)$ is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- $C(ph)$ is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction

# [Charniak 1997] smoothing example

| | $P(\text{prft}\mid\text{rose},\text{NP},\text{S})$ | $P(\text{corp}\mid\text{prft},\text{JJ},\text{NP})$ |
|---|---|---|
| $P(h\mid ph,c,pc)$ | 0 | 0.245 |
| $P(h\mid C(ph),c,pc)$ | 0.00352 | 0.0150 |
| $P(h\mid c,pc)$ | 0.000627 | 0.00533 |
| $P(h\mid c)$ | 0.000557 | 0.00418 |

- Allows utilization of rich highly conditioned estimates, but smoothes when sufficient data is unavailable
- One can’t just use MLEs: one commonly sees previously unseen events, which would have probability 0.

# [Charniak 1997]
## Rule probability with similar smoothing

$$\begin{aligned}
P(r|h,hc,pc) = {} & \lambda_1(e)P(r|h,hc,pc) \\
& \lambda_2(e)P(r|h,hc) \\
& \lambda_3(e)P(r|\mathrm{C}(h),hc) \\
& \lambda_4(e)P(r|hc,pc) \\
& \lambda_5(e)P(r|hc)
\end{aligned}$$

# Sparseness and the Penn Treebank

- The Penn Treebank – 1 million words of parsed English *WSJ* – has been a key resource (because of the widespread reliance on supervised learning)
- But 1 million words is like nothing:
  - □ 965,000 constituents, but only 66 WHADJP, of which only 6 aren't *how much* or *how many*, but there is an infinite space of these (*how clever/original/incompetent* (*at risk assessment and evaluation*))
- Most of the probabilities that you would like to compute, you can't compute

# Sparseness and the Penn Treebank

- Most intelligent processing depends on bilexical statistics: likelihoods of relationships between pairs of words.
- Extremely sparse, even on topics central to the *WSJ*:

| | |
|---|---|
| stocks plummeted | 2 occurrences |
| stocks stabilized | 1 occurrence |
| stocks skyrocketed | 0 occurrences |
| #stocks discussed | 0 occurrences |

- So far there has been very modest success augmenting the Penn Treebank with extra unannotated materials or using semantic classes or clusters (cf. Charniak 1997, Charniak 2000) – as soon as there are more than tiny amounts of annotated training data.

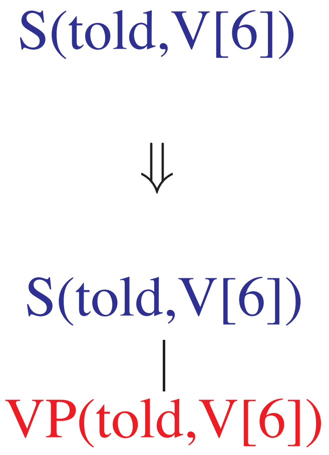
# Lexicalized, Markov out from head

# Collins 1997: Markov model out from head

- Charniak (1997) expands each phrase structure tree in a single step.
- This is good for capturing dependencies between child nodes
- But it is bad because of data sparseness
- A pure dependency, one child at a time, model is worse
- But one can do better by in between models, such as generating the children as a Markov process on both sides of the head (Collins 1997; Charniak 2000)

# Modeling Rule Productions as Markov Processes

- Step 1: generate category of head child



S(told,V[6])

⇓

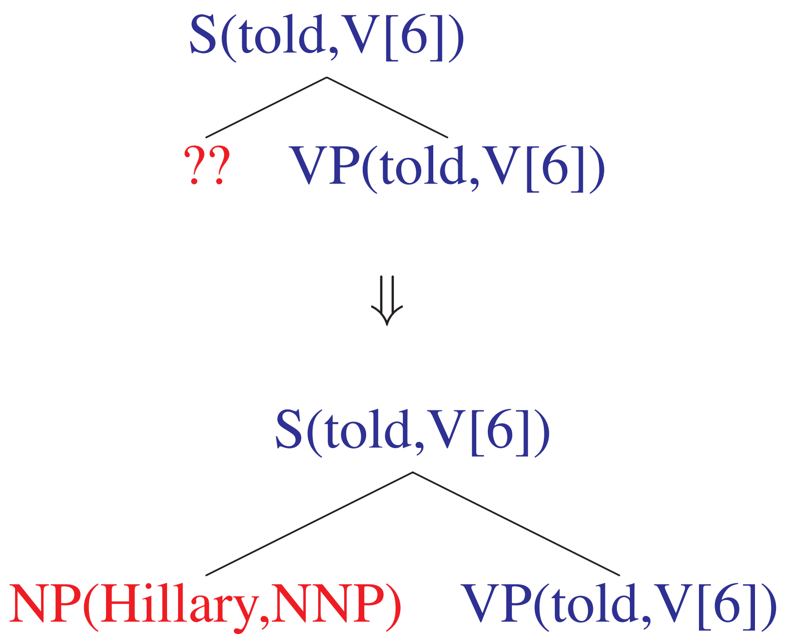
S(told,V[6])

VP(told,V[6])

$P_h(\text{VP} \mid \text{S, told, V[6]})$

# Modeling Rule Productions as Markov Processes

- Step 2: generate left modifiers in a Markov chain



S(told,V[6])

?? VP(told,V[6])

⇓

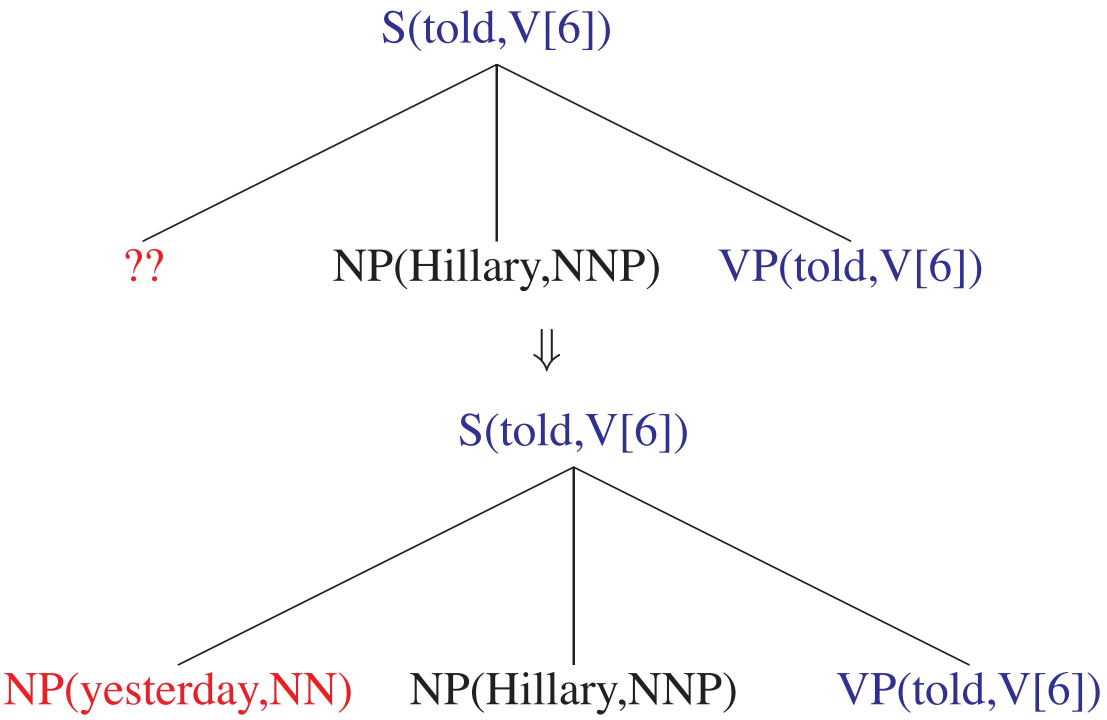
S(told,V[6])

NP(Hillary,NNP) VP(told,V[6])

$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT})$$

# Modeling Rule Productions as Markov Processes

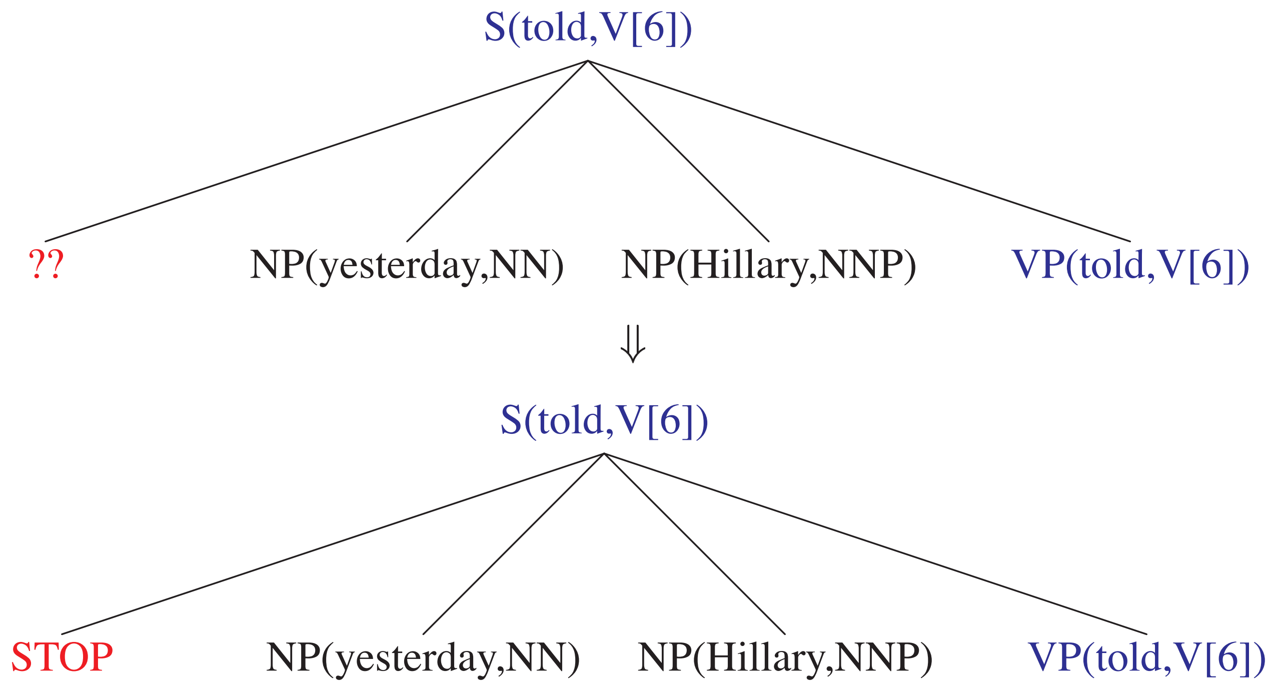
- Step 2: generate left modifiers in a Markov chain



$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$
$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT})$

# Modeling Rule Productions as Markov Processes

- Step 2: generate left modifiers in a Markov chain



S(told,V[6])

?? NP(yesterday,NN) NP(Hillary,NNP) VP(told,V[6])

⇓

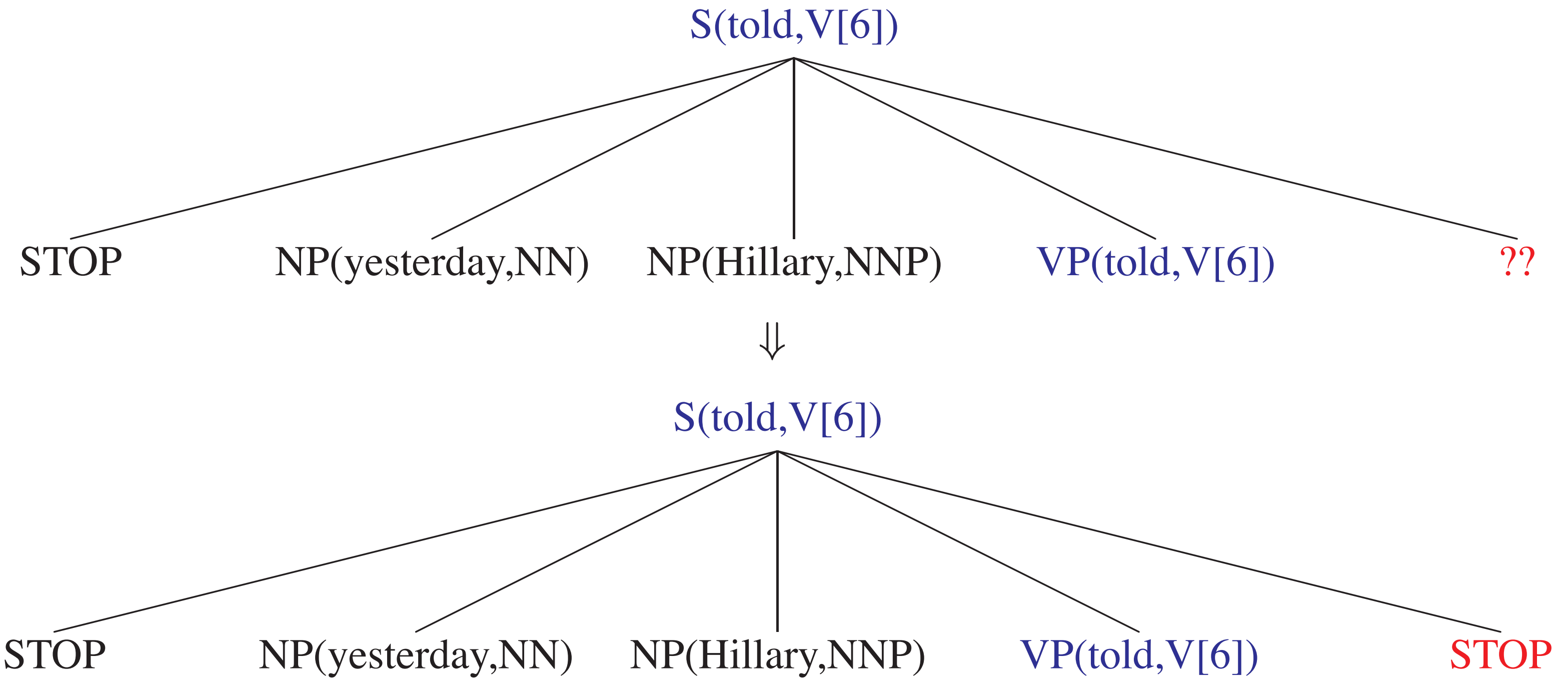
S(told,V[6])

STOP NP(yesterday,NN) NP(Hillary,NNP) VP(told,V[6])

$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$
$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT}) \times P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT})$

# Modeling Rule Productions as Markov Processes

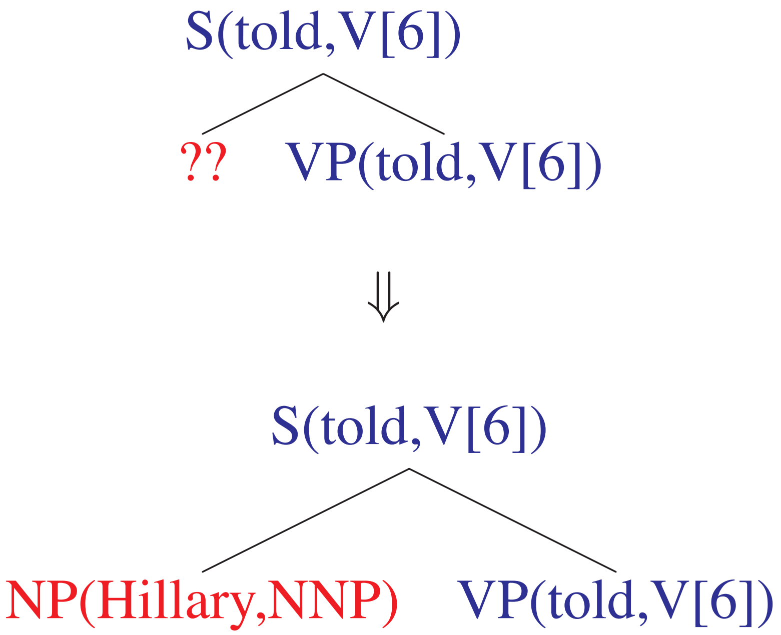
- Step 3: generate right modifiers in a Markov chain



$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$$
$$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT}) \times P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT}) \times$$
$$P_d(\text{STOP} \mid \text{S,VP,told,V[6],RIGHT})$$

# A Refinement: Adding a Distance Variable

- $\Delta = 1$ if position is adjacent to the head.



S(told,V[6])

?? VP(told,V[6])

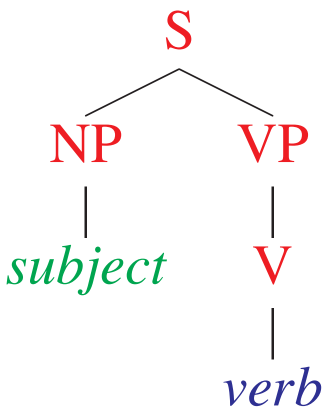
⇓

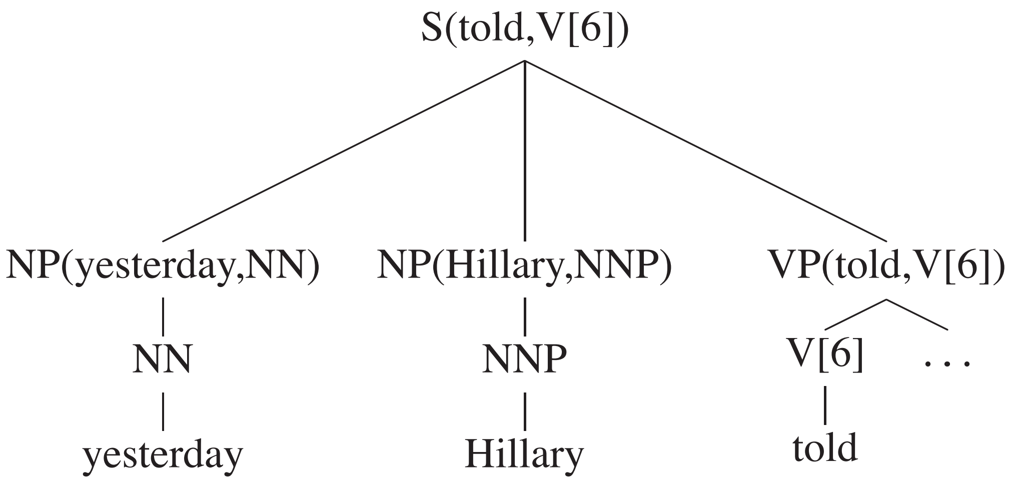
S(told,V[6])

NP(Hillary,NNP) VP(told,V[6])

$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times$$
$$P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 1)$$

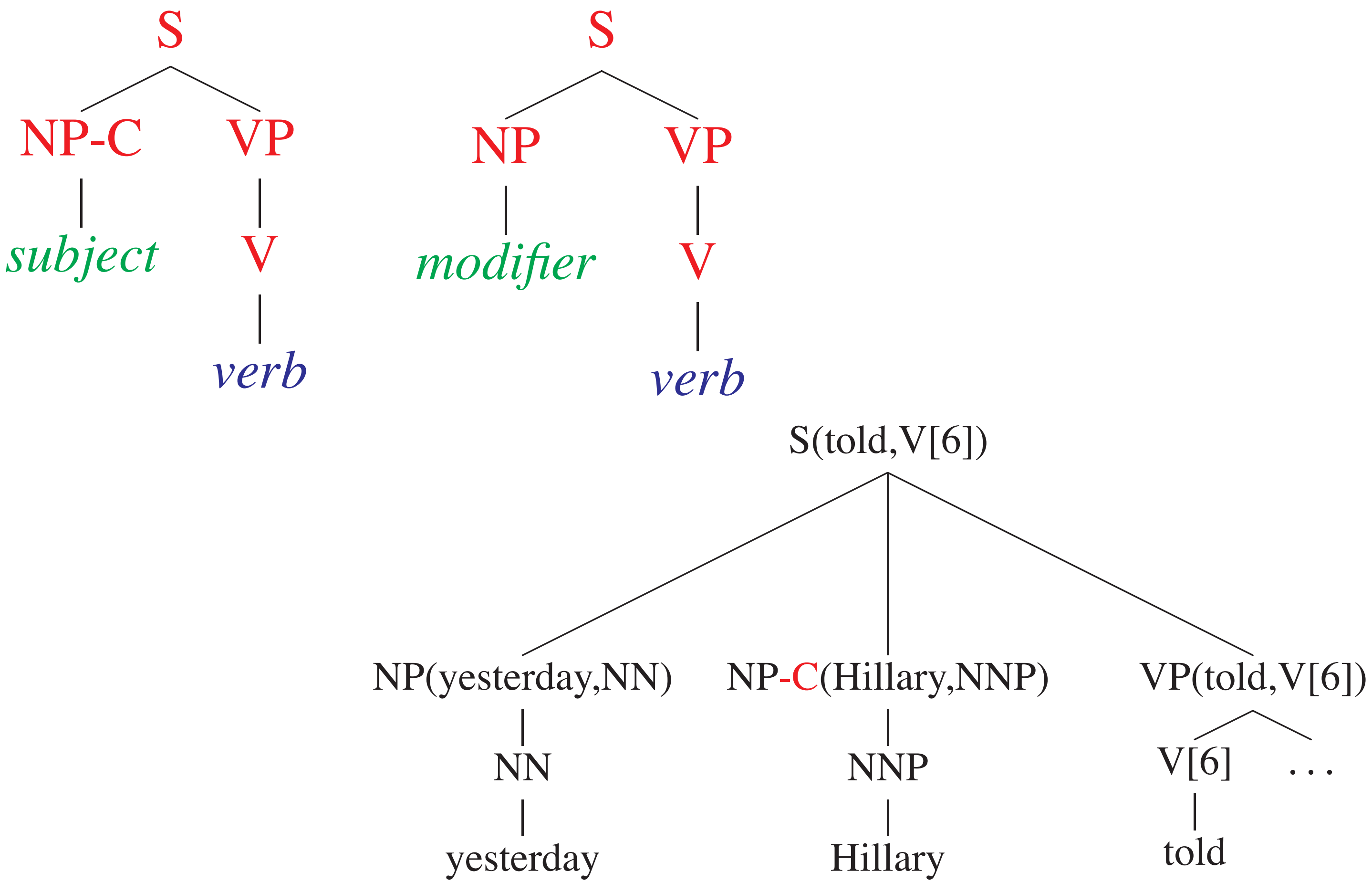
# Adding the Complement/Adjunct Distinction





- *Hillary* is the subject
- *yesterday* is a temporal modifier
- **But nothing to distinguish them.**

# Adding Tags Making the Complement/Adjunct Distinction



# Adding dependency on structure

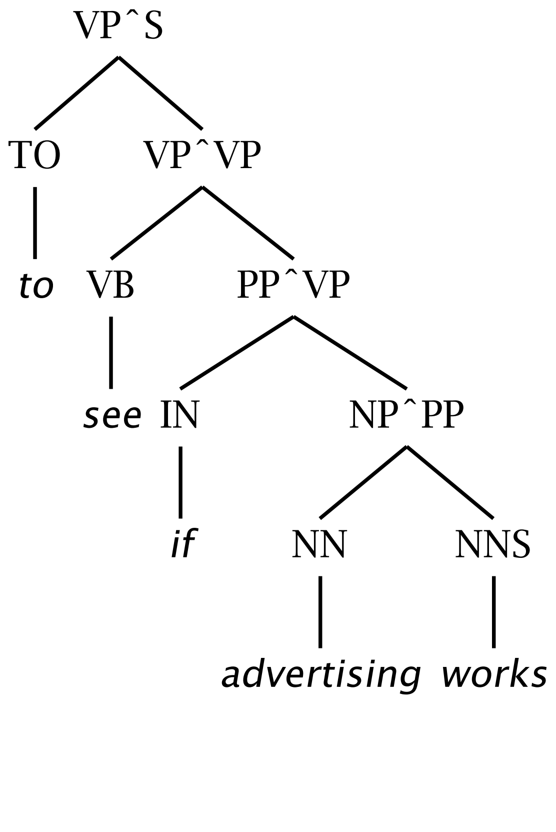
# Weakening the independence assumption of PCFG

Probabilities dependent on structural context

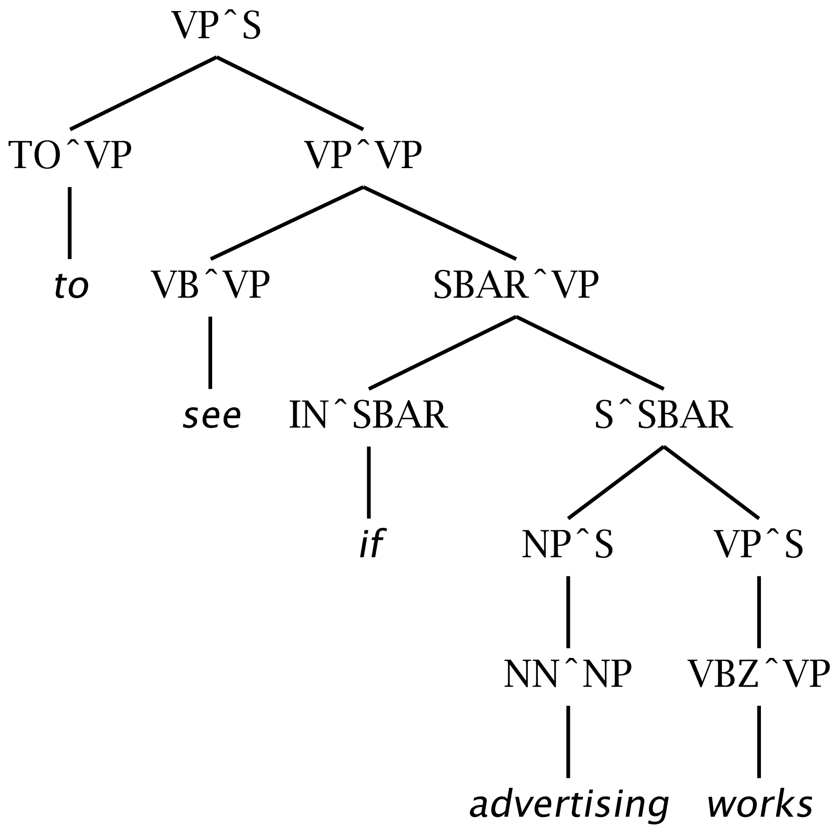
- PCFGs are also deficient on purely structural grounds too
- Really context independent?

| Expansion | % as Subj | % as Obj |
|---|---:|---:|
| NP → PRP | 13.7% | 2.1% |
| NP → NNP | 3.5% | 0.9% |
| NP → DT NN | 5.6% | 4.6% |
| NP → NN | 1.4% | 2.8% |
| NP → NP SBAR | 0.5% | 2.6% |
| NP → NP PP | 5.6% | 14.1% |

# Weakening the independence assumption of PCFG



(a)



(b)

# Faster parsing with beam search

# Pruning for Speed

- Heuristically throw away constituents that probably won't make it into a complete parse.
- Use probabilities to decide which ones.
  - So probs are useful for speed as well as accuracy!
- Both safe and unsafe methods exist
  - Throw x away if $p(x) < 10^{-200}$ (and lower this threshold if we don't get a parse)
  - Throw x away if $p(x) < 100 * p(y)$ for some y that spans the same set of words
  - Throw x away if $p(x)*q(x)$ is small, where $q(x)$ is an estimate of probability of all rules needed to combine x with the other words in the sentence

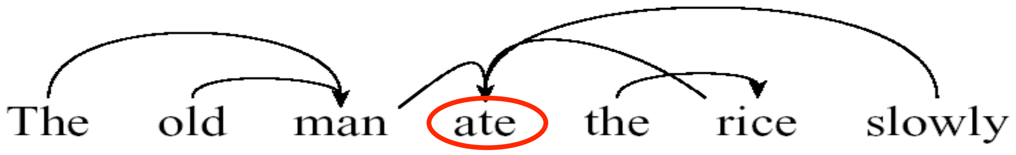
# Dependency Parsing

# Phrase Structure Grammars and Dependency Grammars

Phrase Structure Grammar describes the structure of sentences with phrase structure tree

Alternatively, a Dependency grammar describes the structure with dependencies between words

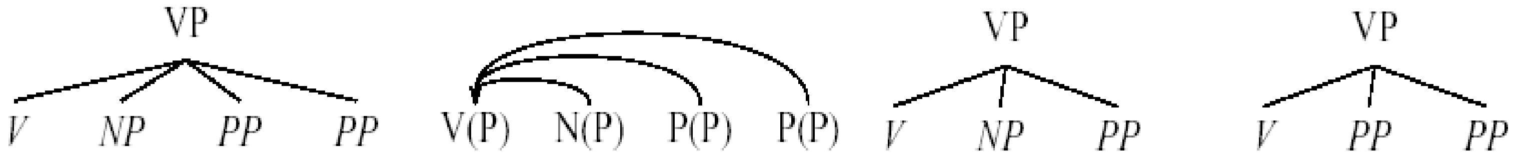
- One word is the head of a sentence and All other words are dependent on that word
- Dependent on some other word which connects to the headword through a sequence of dependencies



# Phrase Structure Grammars and Dependency Grammars

Two key advantages of Dependency grammar are

- Easy to use lexical information
  - Disambiguation decisions are being made directly with words
  - No need to build a large superstructure
  - Not necessary to worry about how to lexicalize a PS tree
- Dependencies are one way of decomposing PS rules
  - Lots of rare flat trees in Penn Treebank → Sparse Data
  - Can get reasonable probabilistic estimate if we decompose it



# Evaluation

| Method | Recall | Precision |
|---|---|---|
| PCFGs (Charniak 97) | 70.6% | 74.8% |
| Decision trees (Magerman 95) | 84.0% | 84.3% |
| Lexicalized with backoff (Charniak 97) | 86.7% | 86.6% |
| Lexicalized with Markov (Collins 97 M1) | 87.5% | 87.7% |
| “ with subcategorization (Collins 97 M2) | 88.1% | 88.3% |
| MaxEnt-inspired (Charniak 2000) | 90.1% | 90.1% |