# 6.891: Lecture 4 (September 15, 2003)
# Stochastic Parsing II

# Overview

- Heads in context-free rules
- Dependency representations of parse trees
- A first model for dependencies: (Charniak 1997)
- A second model for dependencies: (Collins 1997)

# 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: An Example of 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 | $\Rightarrow$ | DT | NNP | NN |
| NP | $\Rightarrow$ | DT | NN | NNP |
| NP | $\Rightarrow$ | NP | PP | |
| NP | $\Rightarrow$ | DT | JJ | |
| NP | $\Rightarrow$ | DT | | |

# Rules which Recover Heads: An Example of rules for VPs

**If** the rule contains Vi or Vt: Choose the leftmost Vi or Vt

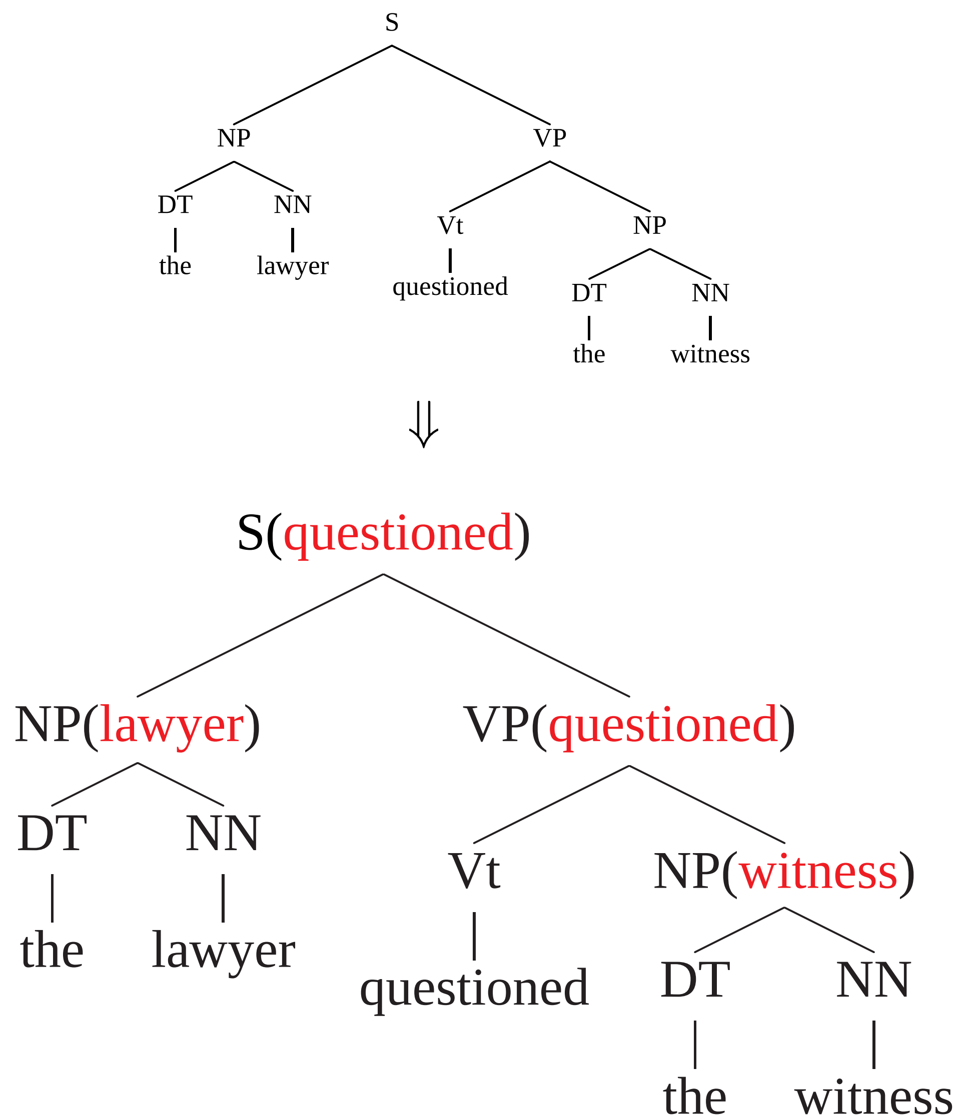
**Else If** the rule contains an VP: Choose the leftmost VP

**Else** Choose the leftmost child

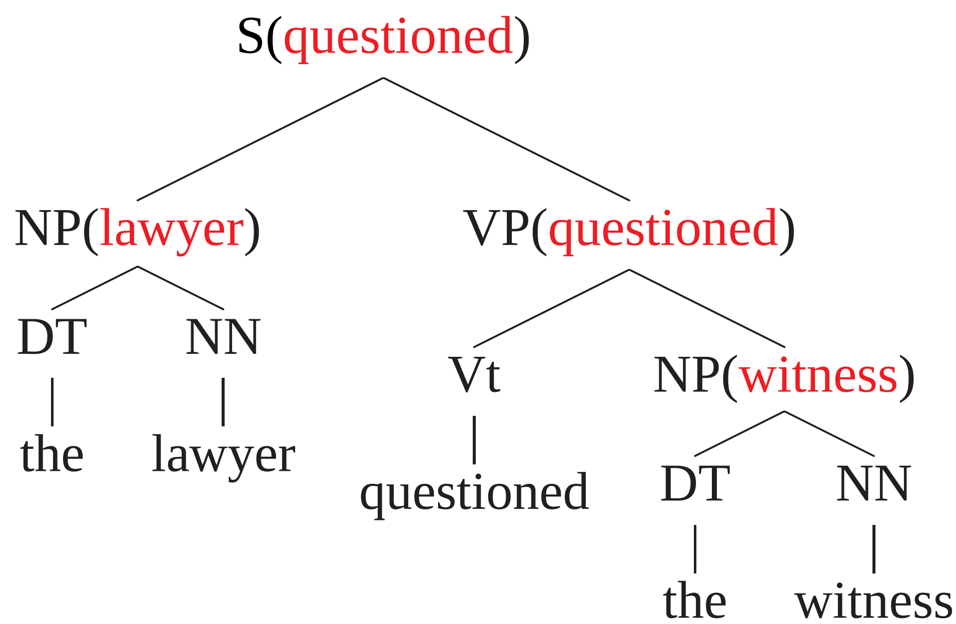
e.g.,

VP $\Rightarrow$ Vt NP
VP $\Rightarrow$ VP PP

# Adding Headwords to Trees



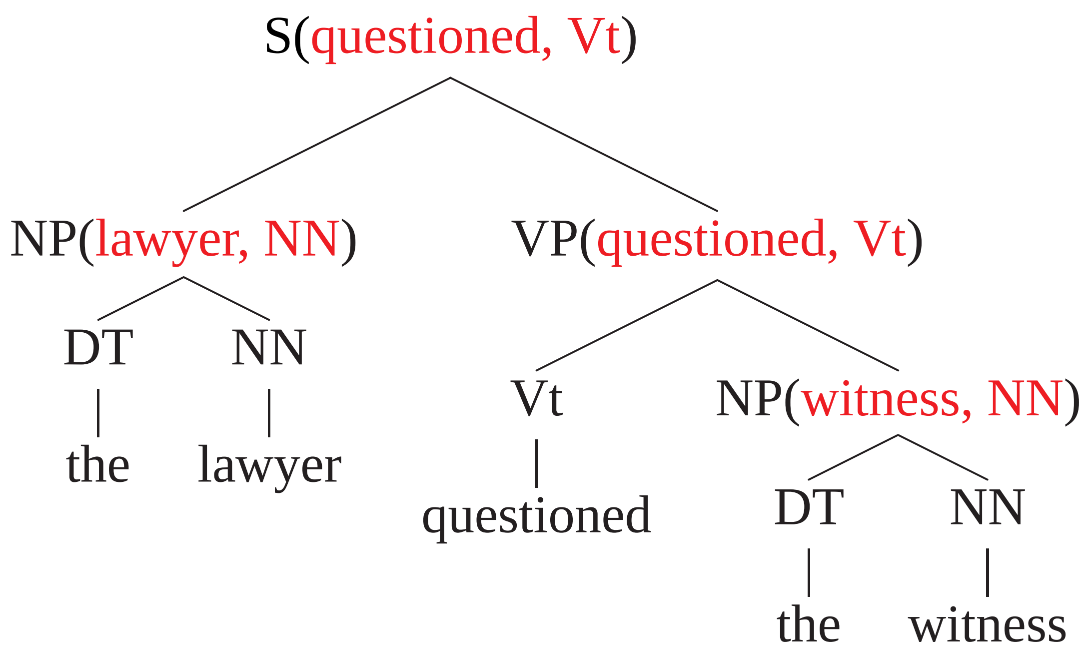
# Adding Headwords to Trees




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

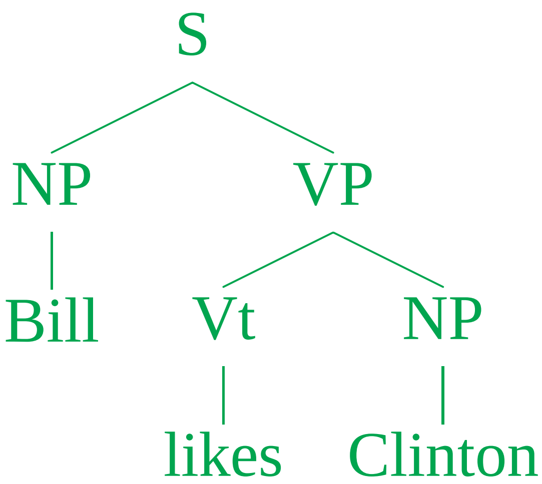
| | | | | | |
|---|---|---|---|---|---|
| S | $\Rightarrow$ | NP | VP | | (S receives headword from VP) |
| VP | $\Rightarrow$ | Vt | NP | | (VP receives headword from Vt) |
| NP | $\Rightarrow$ | DT | | NN | (NP receives headword from NN) |

# Adding Headtags to Trees




- Also propogate **part-of-speech tags** up the trees
  (We'll see soon why this is useful!)

# Heads and Semantics




$\Rightarrow$ *like(Bill, Clinton)*

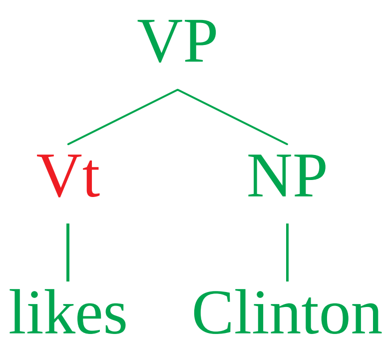
**Syntactic structure** $\Rightarrow$
**Semantics/Logical form/Predicate-argument structure**

# Adding Predicate Argument Structure to our Grammar

- Identify words with lambda terms:

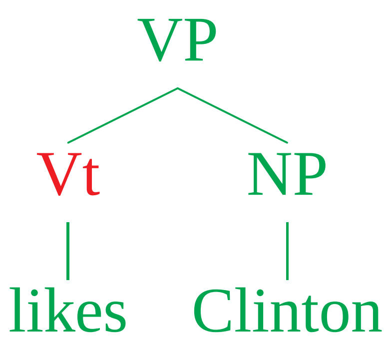
| | |
|---|---|
| likes | $\lambda y, x \quad like(x, y)$ |
| Bill | $Bill$ |
| Clinton | $Clinton$ |

- Semantics for an entire constituent is formed by applying semantics of head (predicate) to the other children (arguments)

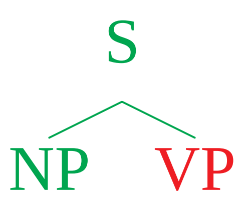



$$\Rightarrow \quad [\lambda y, x \quad like(x, y)] \, [Clinton]$$
$$= [\lambda x \quad like(x, Clinton)]$$

# Adding Predicate-Argument Structure to our Grammar




$$\Rightarrow \quad [\lambda y, x \quad like(x, y)]\ [Clinton]$$
$$= [\lambda x \quad like(x, Clinton)]$$




$$\Rightarrow \quad [\lambda x \quad like(x, Clinton)]\ [Bill]$$
$$= [like(Bill, Clinton)]$$

Note that $like$ is the predicate for both the VP and the S, and provides the head for both rules

# Headwords and Dependencies

- A new representation: a tree is represented as a set of *dependencies*, not a set of *context-free rules*

# Headwords and Dependencies

- A **dependency** is an 8-tuple:

| | |
|---|---|
| (headword, | headtag, |
| modifer-word, | modifer-tag, |
| parent non-terminal, | head non-terminal, |
| modifier non-terminal, | direction) |

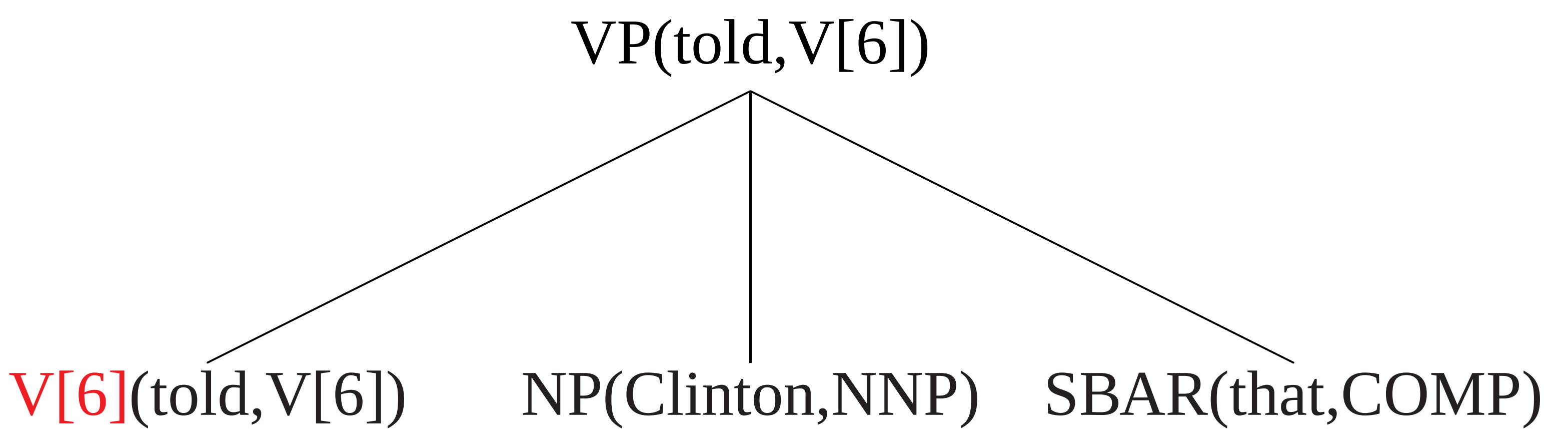
- Each rule with $n$ children contributes $(n-1)$ dependencies.

VP(questioned,Vt) $\Rightarrow$ Vt(questioned,Vt) NP(lawyer,NN)

$\Downarrow$

(questioned, Vt, lawyer, NN, VP, Vt, NP, RIGHT)

# Headwords and Dependencies

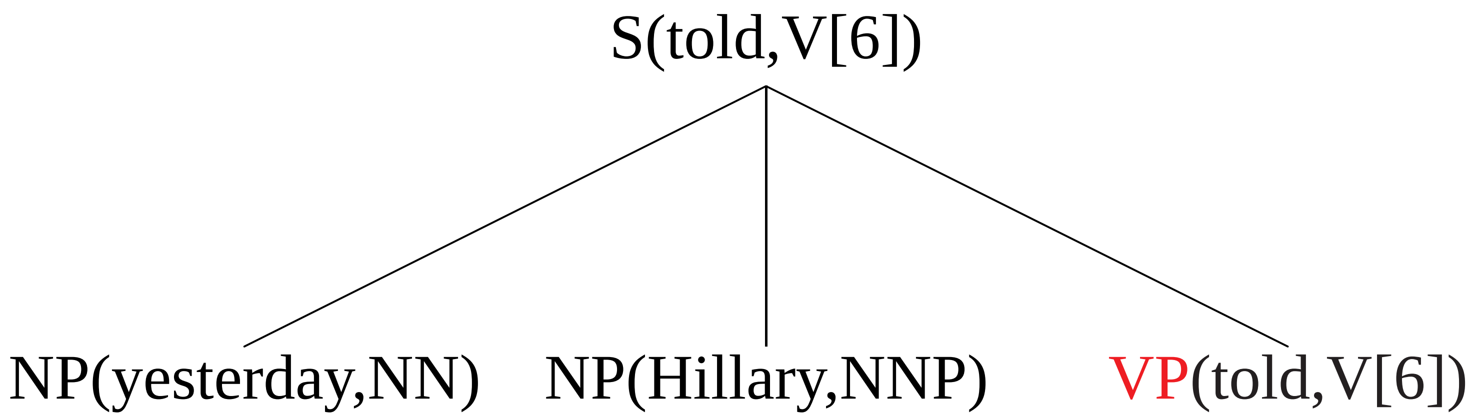



$\Downarrow$

(told, V[6], Clinton, NNP, VP, V[6], NP, RIGHT)

(told, V[6], that, COMP, VP, V[6], SBAR, RIGHT)

# Headwords and Dependencies




⇓

(told, V[6], yesterday, NN, S, VP, NP, LEFT)

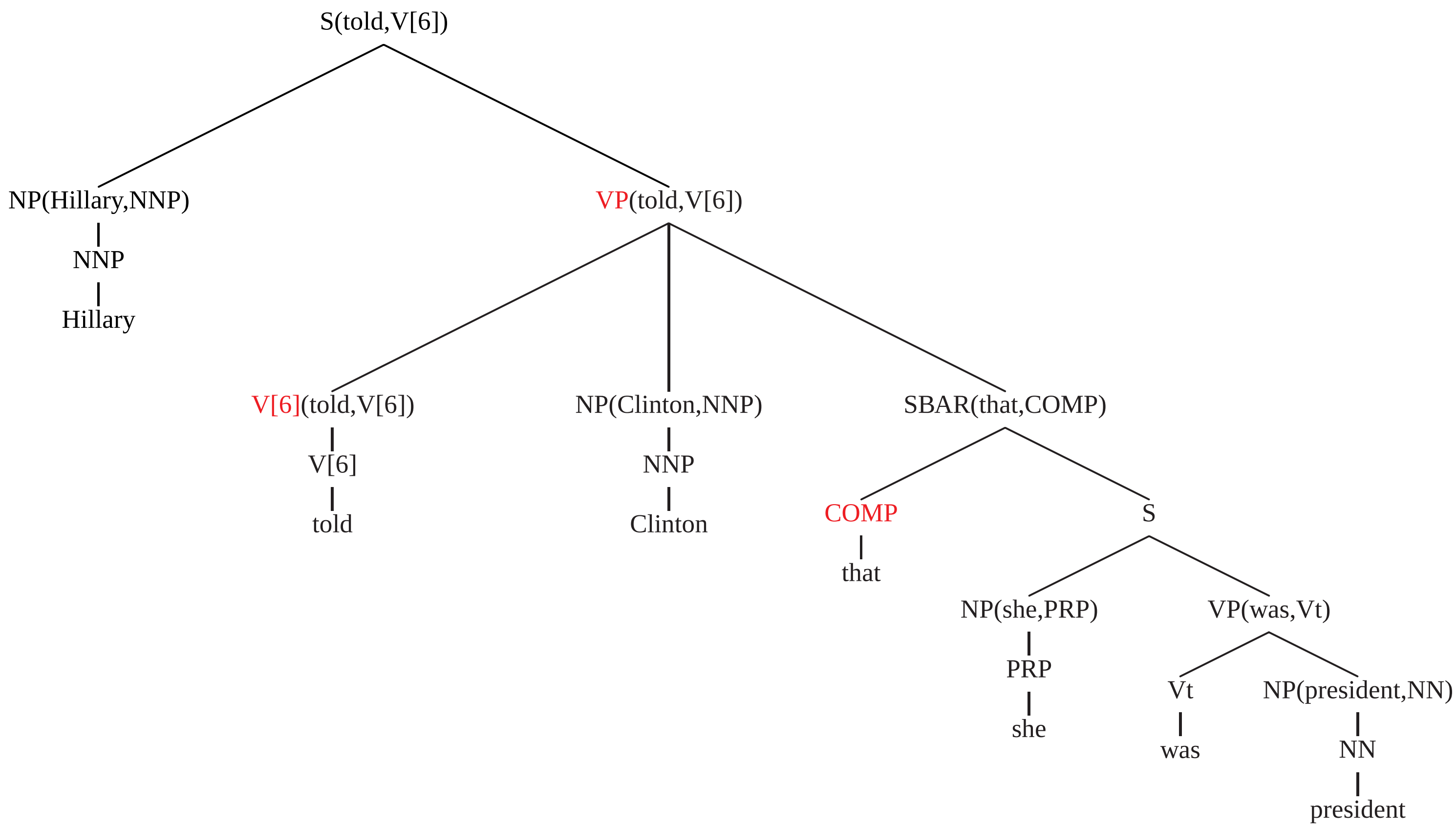
(told, V[6], Hillary, NNP, S, VP, NP, LEFT)

# A Special Case: the Top of the Tree

TOP
|
S(told,V[6])

$\Downarrow$

(__, __, told, V[6], TOP, S, __, SPECIAL)




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| (__ | __ | told | V[6] | TOP | S | __ | SPECIAL) |
| (told | V[6] | Hillary | NNP | S | VP | NP | LEFT) |
| (told | V[6] | Clinton | NNP | VP | V[6] | NP | RIGHT) |
| (told | V[6] | that | COMP | VP | V[6] | SBAR | RIGHT) |
| (that | COMP | was | Vt | SBAR | COMP | S | RIGHT) |
| (was | Vt | she | PRP | S | VP | NP | LEFT) |
| (was | Vt | president | NP | VP | Vt | NP | RIGHT) |

## Charniak (1997)

S(questioned,Vt)

⇓ $P($NP(__,NN) VP $|$ S(questioned,Vt)$)$

S(questioned,Vt)

NP(__,NN) VP(questioned,Vt)

⇓ $P($lawyer $|$ S,VP,NP,NN, questioned,Vt)$)$

S(questioned,Vt)

NP(lawyer,NN) VP(questioned,Vt)

# Smoothed Estimation

$$P(\text{NP(\_\_,NN) VP} \mid \text{S(questioned,Vt)}) =$$

$$\lambda_1 \times \frac{Count(\text{S(questioned,Vt)}\rightarrow\text{NP(\_\_,NN) VP})}{Count(\text{S(questioned,Vt)})}$$

$$+\lambda_2 \times \frac{Count(\text{S(\_\_,Vt)}\rightarrow\text{NP(\_\_,NN) VP})}{Count(\text{S(\_\_,Vt)})}$$

- Where $0 \leq \lambda_1, \lambda_2 \leq 1$, and $\lambda_1 + \lambda_2 = 1$

# Smoothed Estimation

$$P(\text{lawyer} \mid \text{S,VP,NP,NN,questioned,Vt}) =$$

$$\lambda_1 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,questioned,Vt})}{Count(\text{S,VP,NP,NN,questioned,Vt})}$$

$$+\lambda_2 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,Vt})}{Count(\text{S,VP,NP,NN,Vt})}$$

$$+\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{NN})}{Count(\text{NN})}$$

- Where $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$

$$P(\text{NP(lawyer,NN) VP} \mid \text{S(questioned,Vt)}) =$$

$$\left(\lambda_1 \times \frac{Count(\text{S(questioned,Vt)}\rightarrow\text{NP(\_\_,NN) VP})}{Count(\text{S(questioned,Vt)})}\right.$$

$$\left.+\lambda_2 \times \frac{Count(\text{S(\_\_,Vt)}\rightarrow\text{NP(\_\_,NN) VP})}{Count(\text{S(\_\_,Vt)})}\right)$$

$$\times \left(\lambda_1 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,questioned,Vt})}{Count(\text{S,VP,NP,NN,questioned,Vt})}\right.$$

$$+\lambda_2 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,Vt})}{Count(\text{S,VP,NP,NN,Vt})}$$

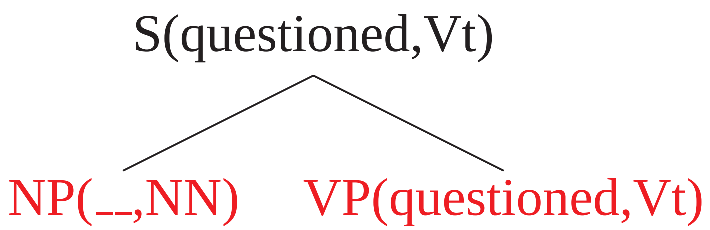
$$\left.+\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{NN})}{Count(\text{NN})}\right)$$

# Motivation for Breaking Down Rules

- First step of decomposition of (Charniak 1997):

S(questioned,Vt)

$\Downarrow$ $\quad P($NP(__,NN) VP | S(questioned,Vt)$)$




---

- Relies on counts of entire rules
- These counts are *sparse*:
  - 40,000 sentences from Penn treebank have 12,409 rules.
  - 15% of all test data sentences contain a rule never seen in training

# Motivation for Breaking Down Rules

| Rule Count | No. of Rules by Type | Percentage by Type | No. of Rules by token | Percentage by token |
|---|---|---|---|---|
| 1 | 6765 | 54.52 | 6765 | 0.72 |
| 2 | 1688 | 13.60 | 3376 | 0.36 |
| 3 | 695 | 5.60 | 2085 | 0.22 |
| 4 | 457 | 3.68 | 1828 | 0.19 |
| 5 | 329 | 2.65 | 1645 | 0.18 |
| 6 ... 10 | 835 | 6.73 | 6430 | 0.68 |
| 11 ... 20 | 496 | 4.00 | 7219 | 0.77 |
| 21 ... 50 | 501 | 4.04 | 15931 | 1.70 |
| 51 ... 100 | 204 | 1.64 | 14507 | 1.54 |
| $>$ 100 | 439 | 3.54 | 879596 | 93.64 |

Statistics for rules taken from sections 2-21 of the treebank
(Table taken from my PhD thesis).

# Modeling Rule Productions as Markov Processes

- Step 1: generate category of head child

---

S(told,V[6])

$\Downarrow$

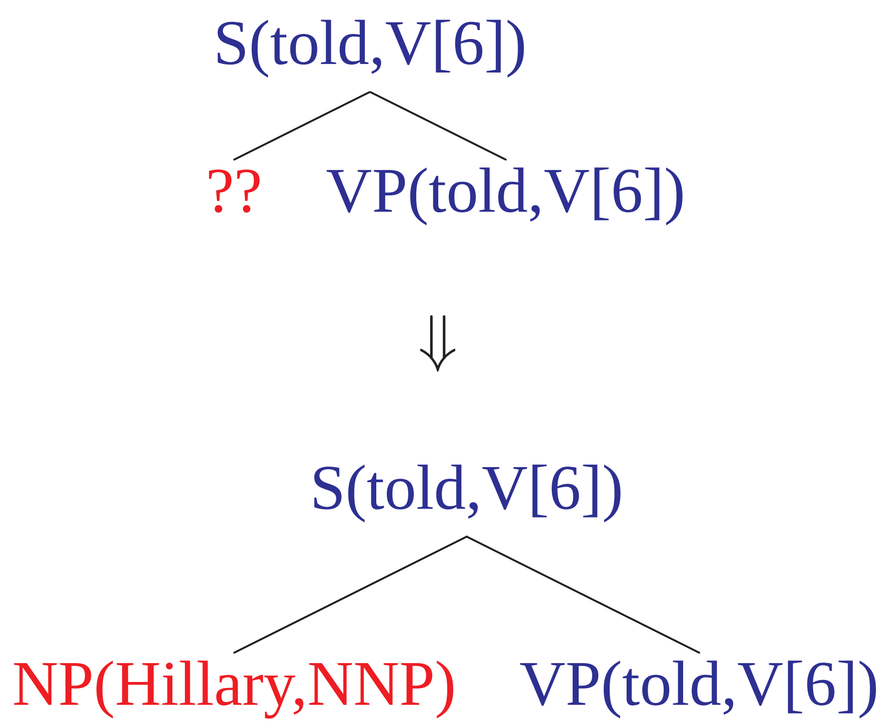
S(told,V[6])
|
VP(told,V[6])

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

# Modeling Rule Productions as Markov Processes

- Step 2: generate left modifiers in a Markov chain

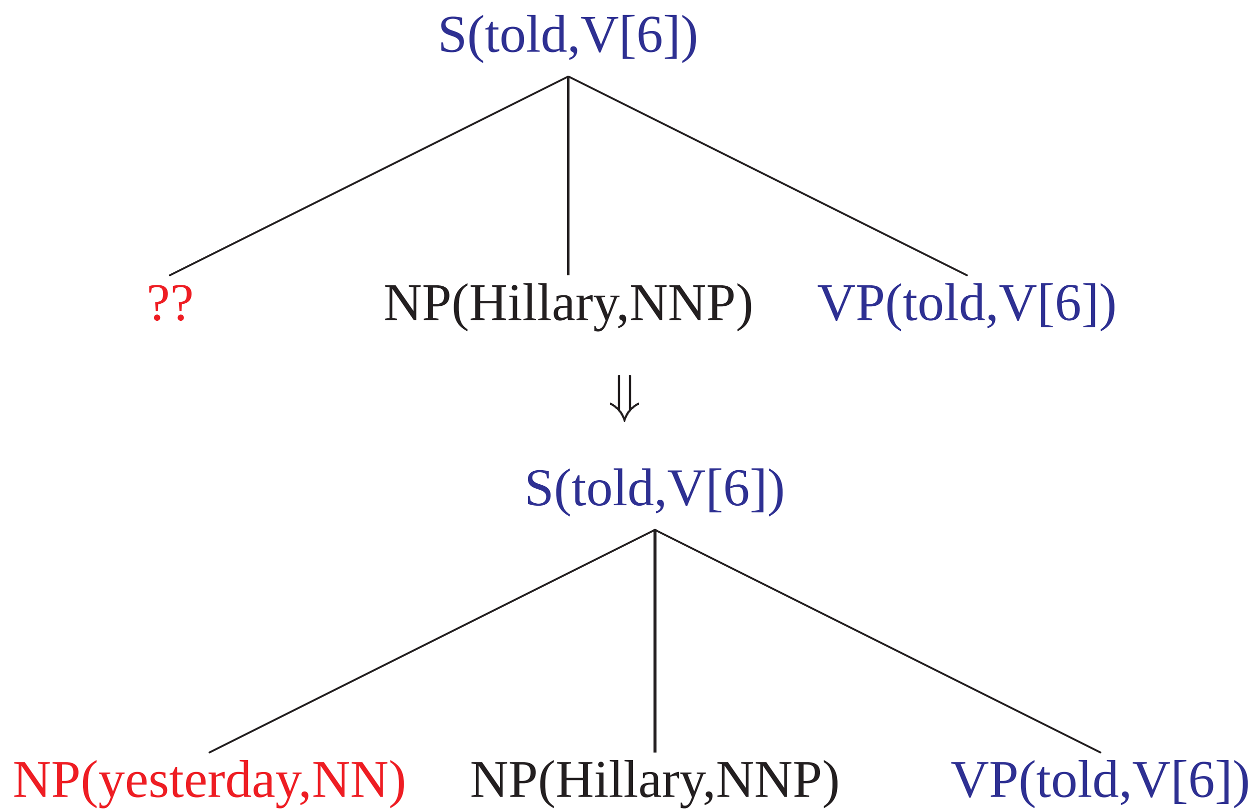
---




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

# Modeling Rule Productions as Markov Processes

- Step 2: generate left modifiers in a Markov chain

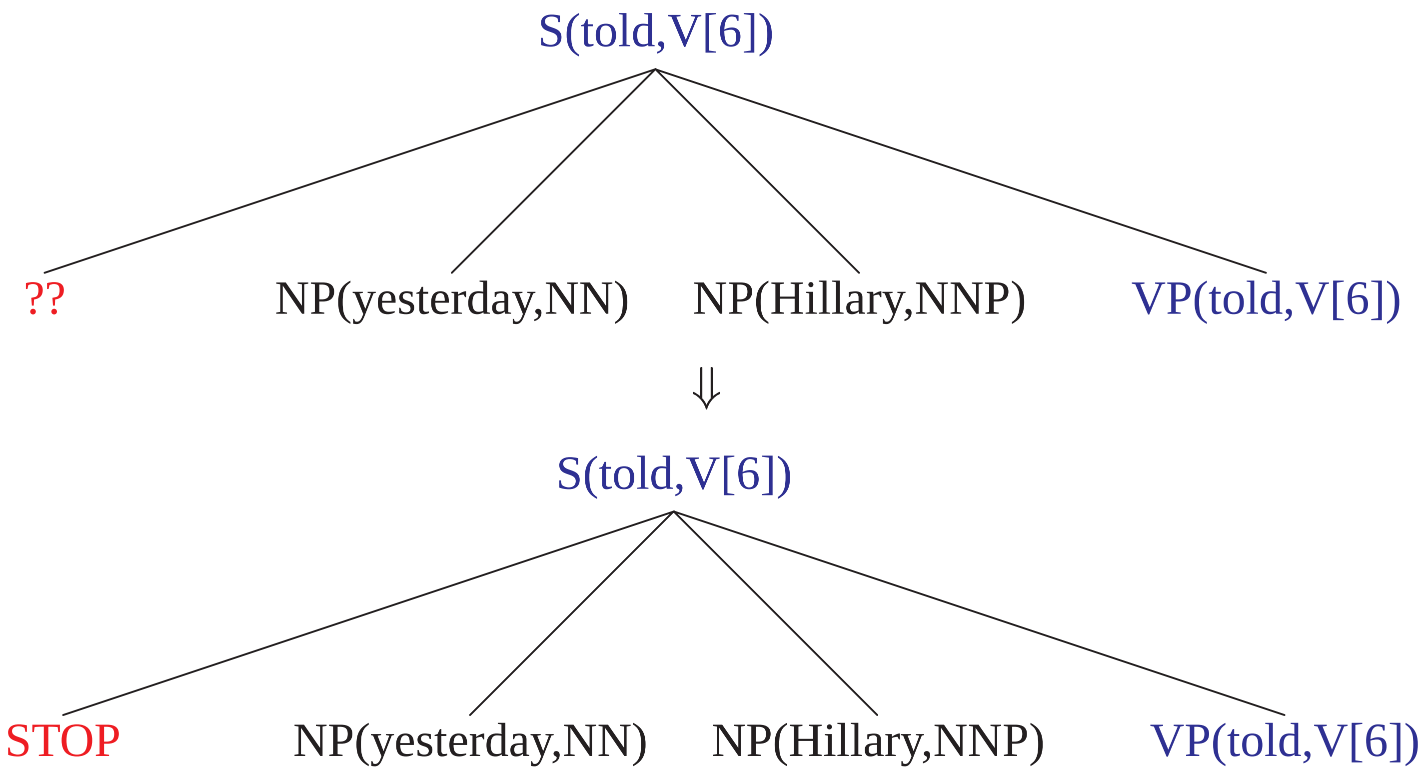



$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$$
$$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT})$$

# Modeling Rule Productions as Markov Processes

- Step 2: generate left modifiers in a Markov chain

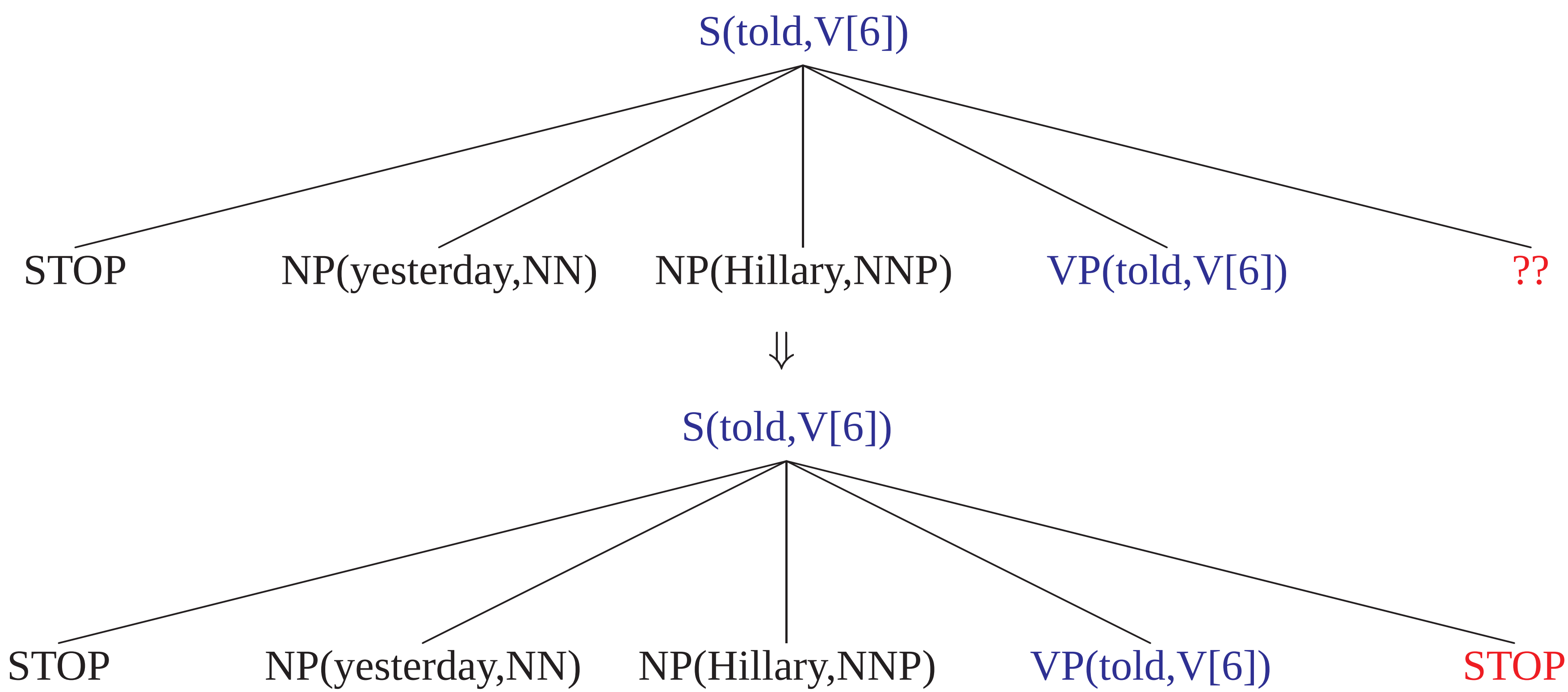
---




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

# Modeling Rule Productions as Markov Processes

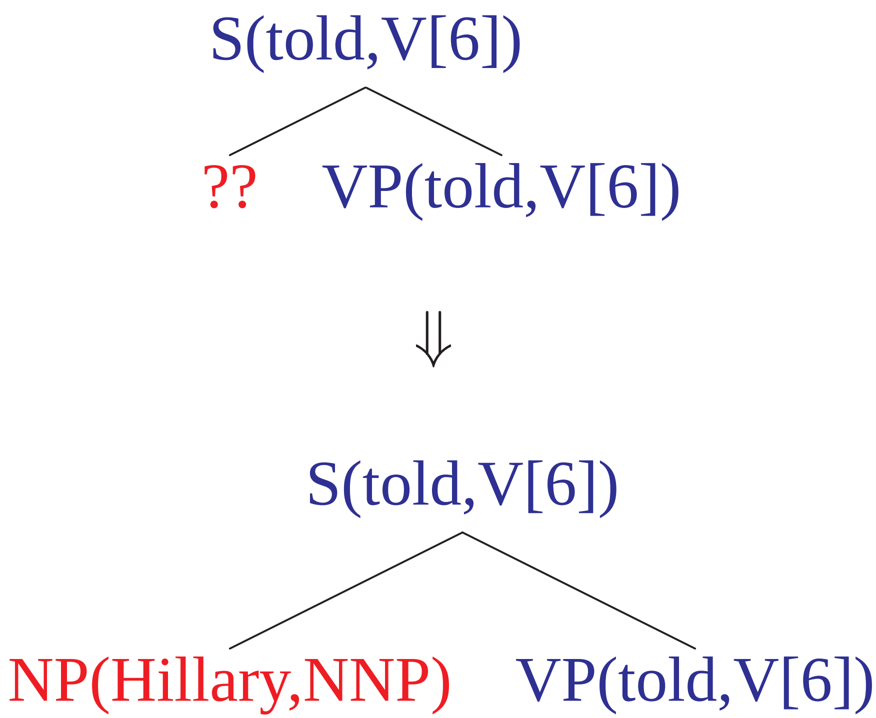
- Step 3: generate right modifiers in a Markov chain




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

# A Refinement: Adding a Distance Variable

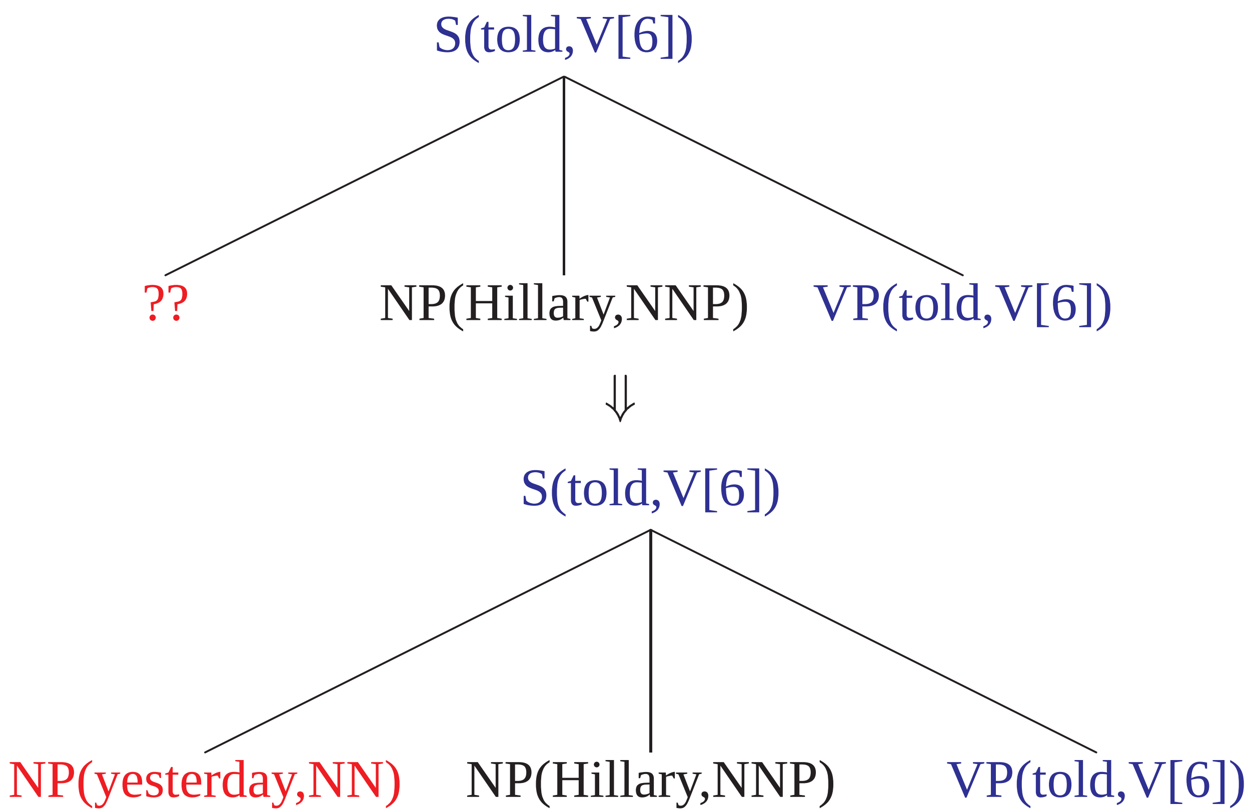
- $\Delta = 1$ if position is adjacent to the head.




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

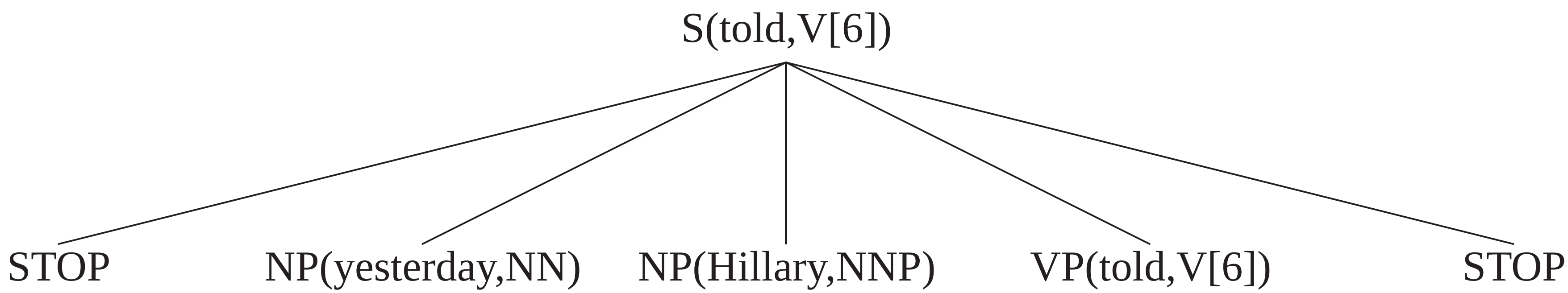
# A Refinement: Adding a Distance Variable

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



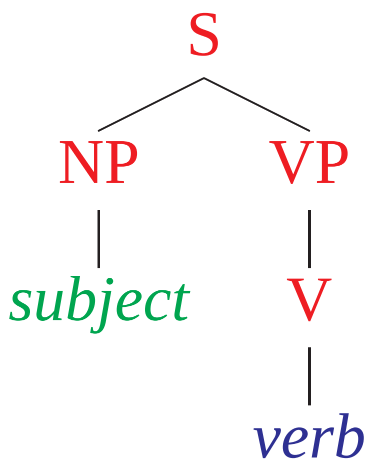

$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT}) \times$
$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 0)$

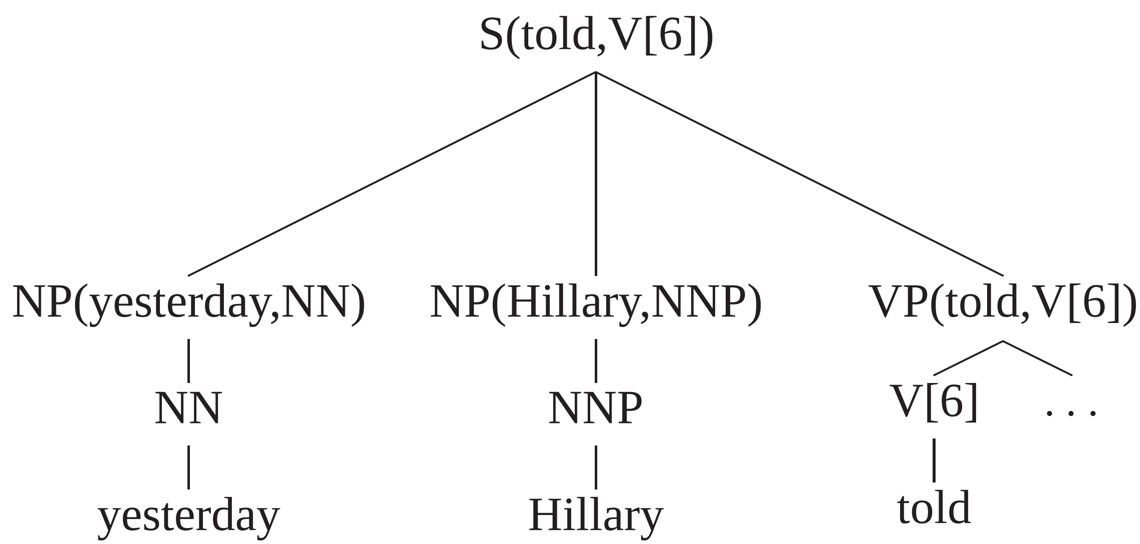
# The Final Probabilities




$$
\begin{aligned}
&P_h(\text{VP} \mid \text{S, told, V[6]}) \times \\
&P_d(\text{NP(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 1) \times \\
&P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 0) \times \\
&P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 0) \times \\
&P_d(\text{STOP} \mid \text{S,VP,told,V[6],RIGHT,}\Delta = 1)
\end{aligned}
$$

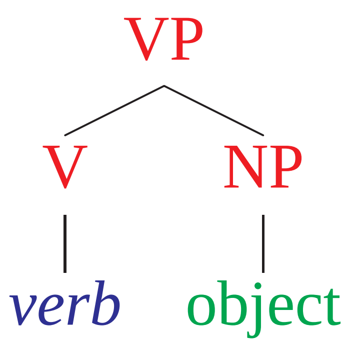
# Adding the Complement/Adjunct Distinction

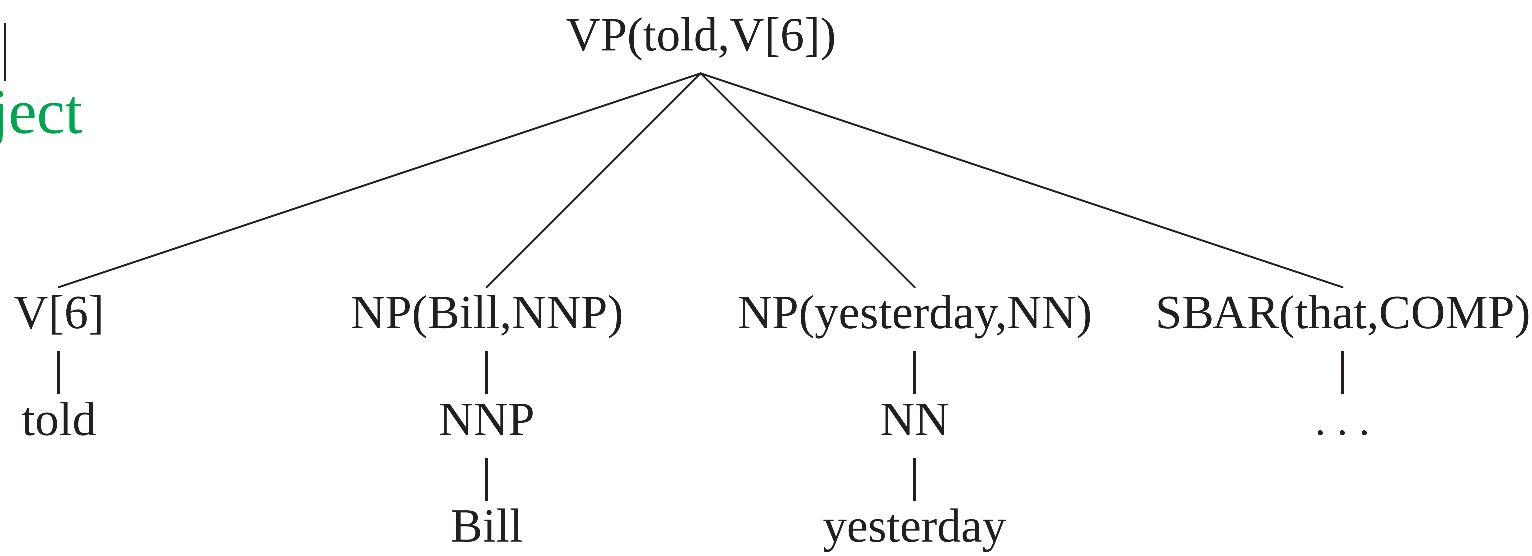



- *Hillary* is the subject
- *yesterday* is a temporal modifier
- **But nothing to distinguish them.**

# Adding the Complement/Adjunct Distinction



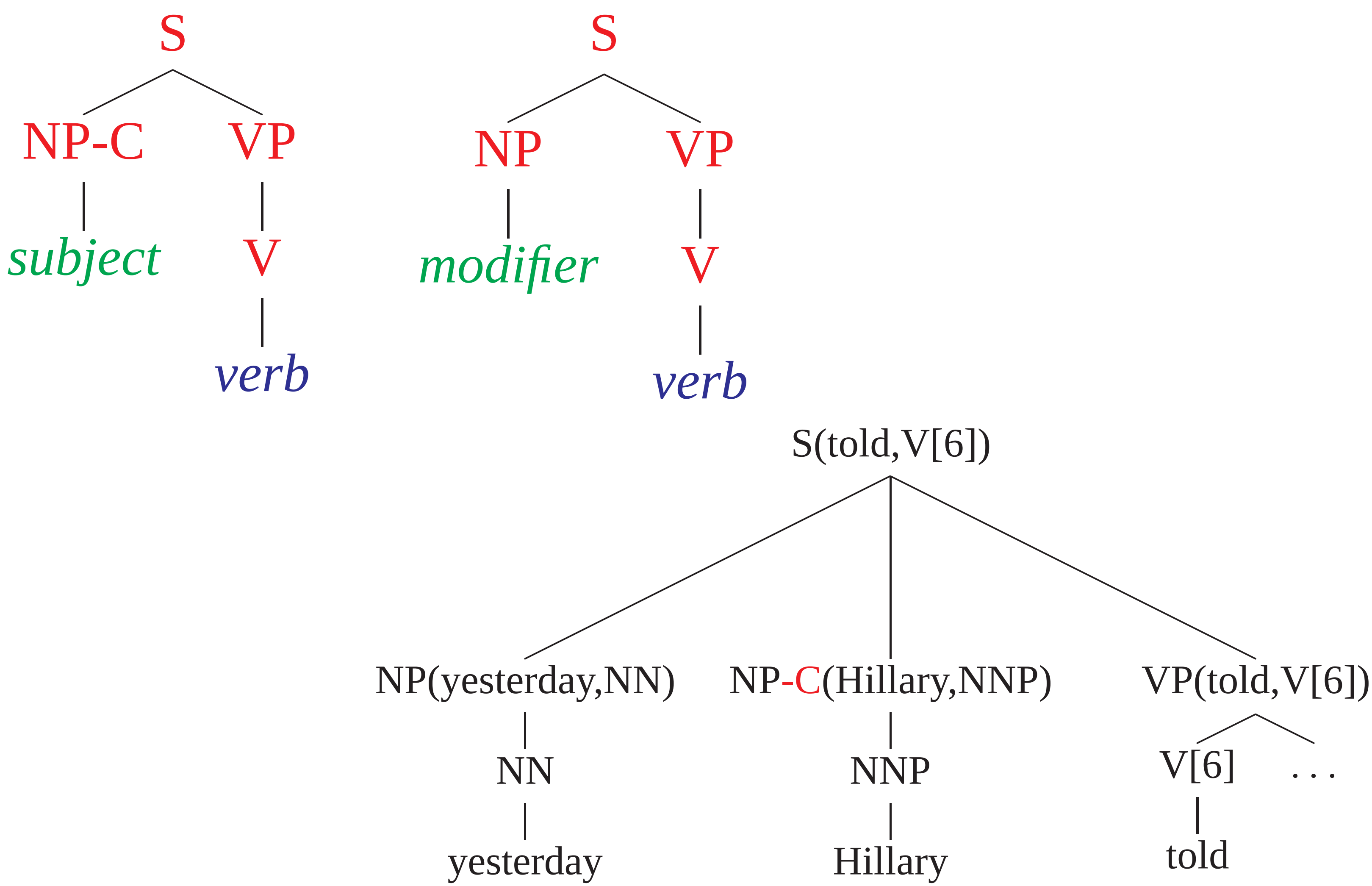

- *Bill* is the object
- *yesterday* is a temporal modifier
- **But nothing to distinguish them.**

# Complements vs. Adjuncts

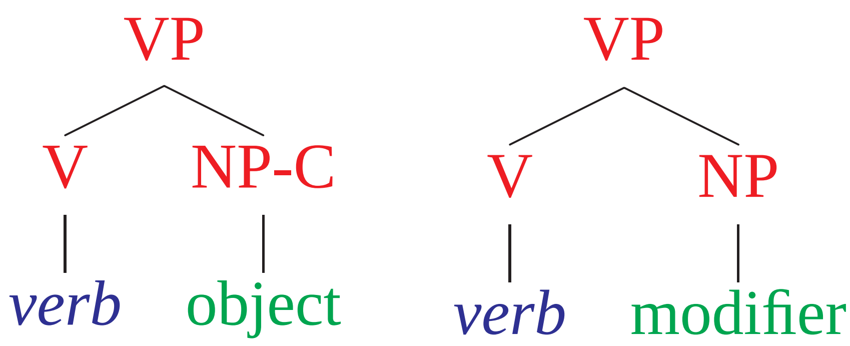
- Complements are closely related to the head they modify, adjuncts are more indirectly related
- Complements are usually arguments of the thing they modify
  yesterday Hillary told . . . ⇒ *Hillary* is doing the *telling*
- Adjuncts add modifying information: time, place, manner etc.
  yesterday Hillary told . . . ⇒ *yesterday* is a *temporal modifier*
- Complements are usually required, adjuncts are optional

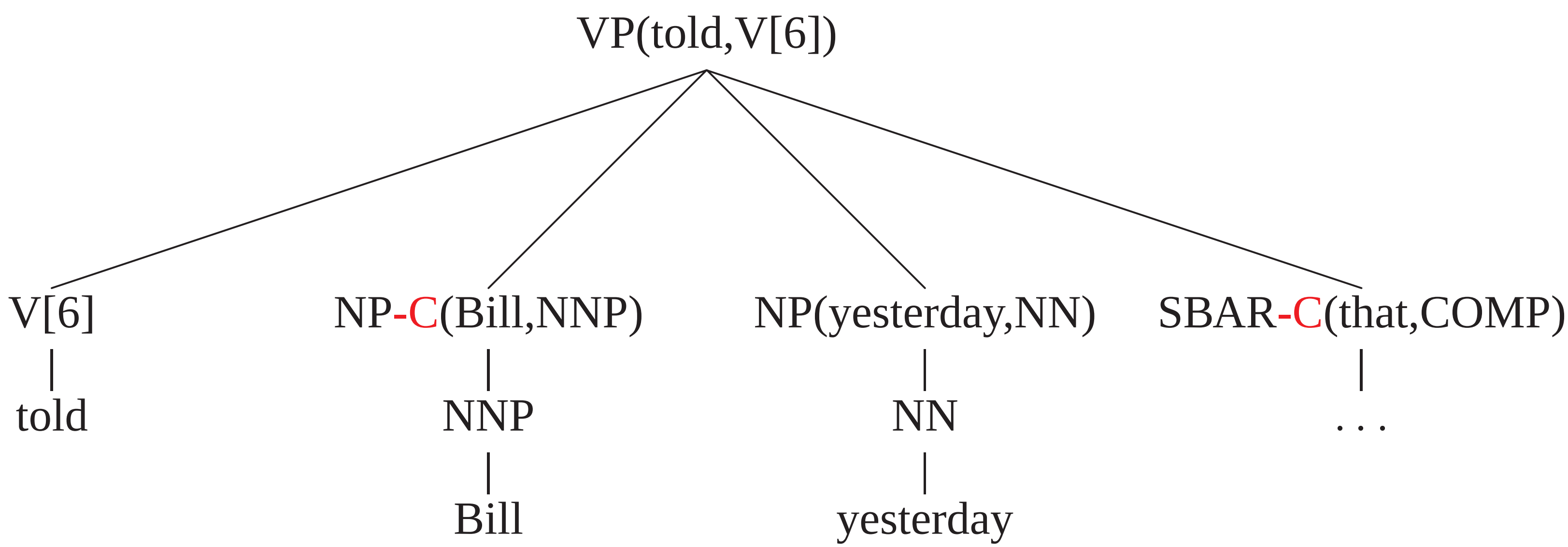
yesterday Hillary told . . . (grammatical)
vs. Hillary told . . . (grammatical)
vs. yesterday told . . . (ungrammatical)

# Adding Tags Making the Complement/Adjunct Distinction



# Adding Tags Making the Complement/Adjunct Distinction



# Adding Subcategorization Probabilities

- Step 1: generate category of head child

---

S(told,V[6])

$\Downarrow$

S(told,V[6])
|
VP(told,V[6])

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

# Adding Subcategorization Probabilities

- Step 2: choose left **subcategorization frame**

---

S(told,V[6])

|

VP(told,V[6])

$\Downarrow$

S(told,V[6])

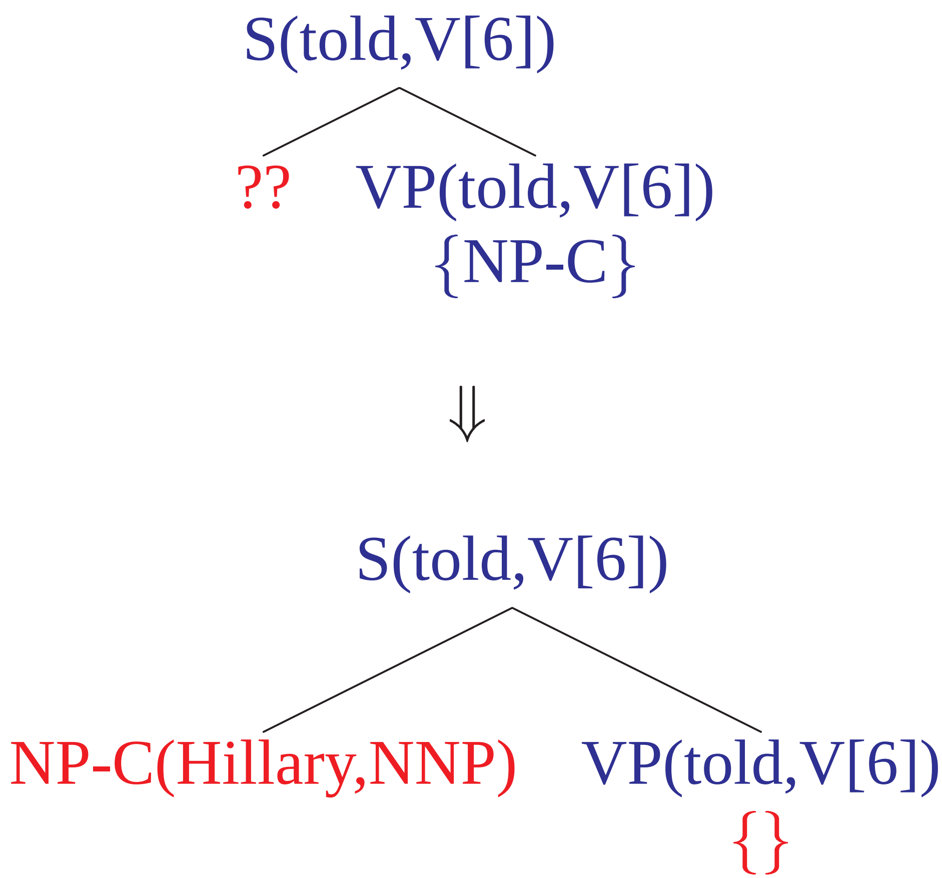
|

VP(told,V[6])

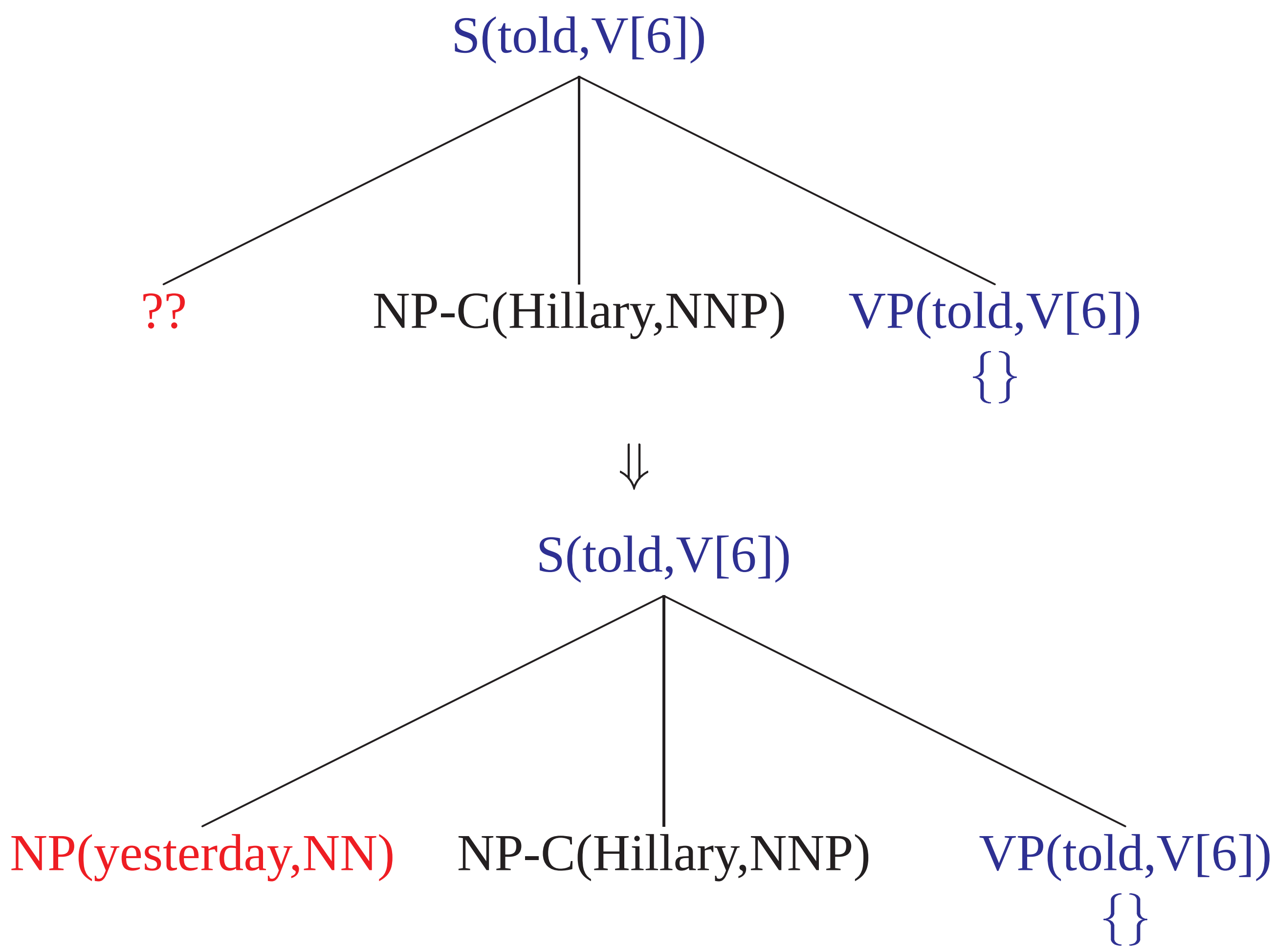
{NP-C}

$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]})$$

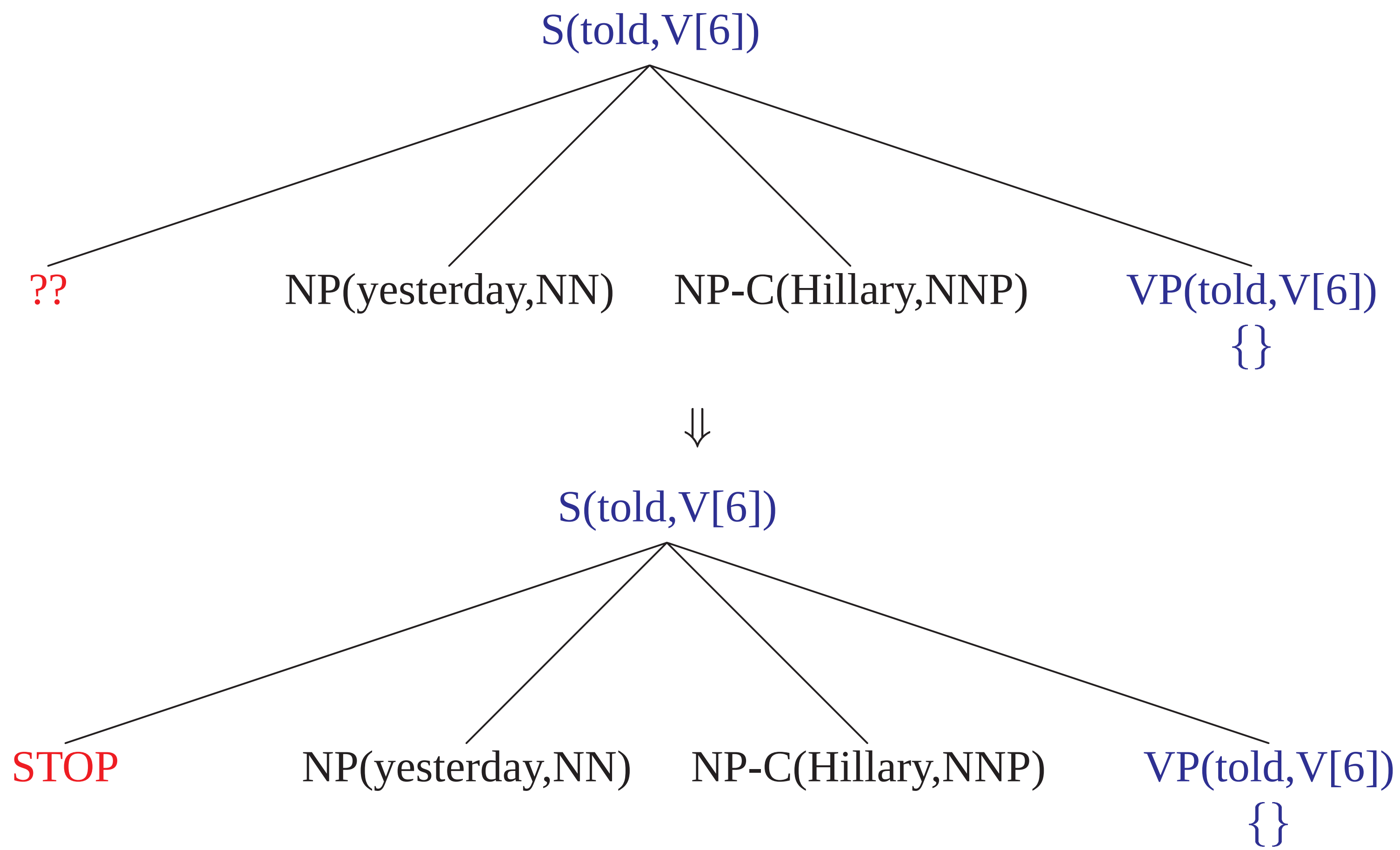
- Step 3: generate left modifiers in a Markov chain




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

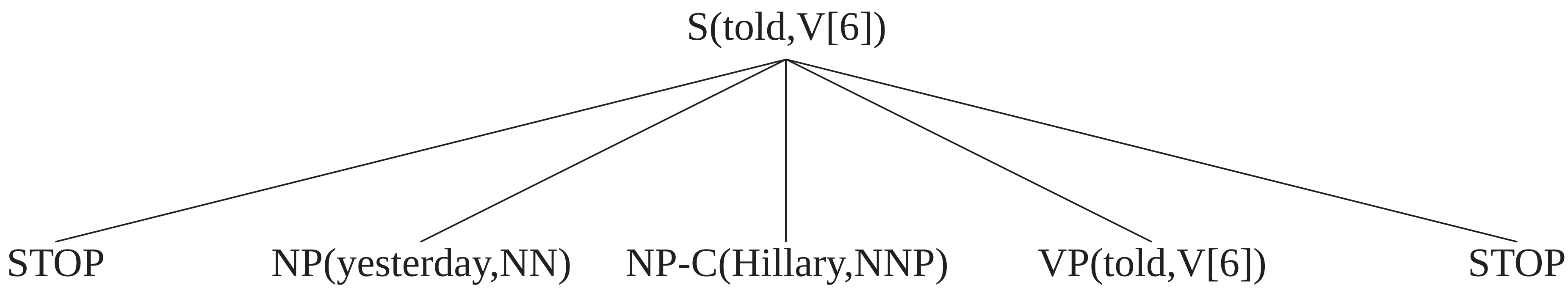



$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]})$$
$$P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\{\text{NP-C}\})\times$$
$$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT,}\{\})$$



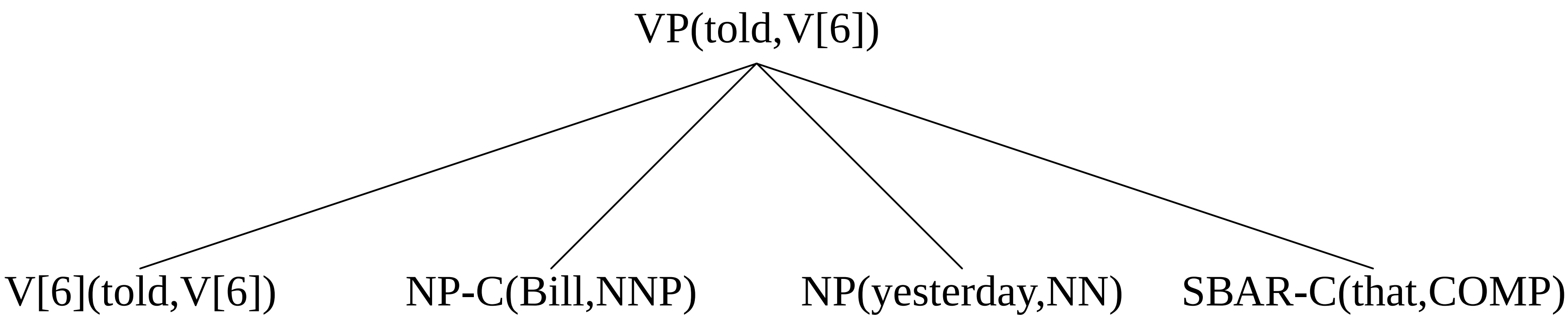

$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]})$
$P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\{\text{NP-C}\})\times$
$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT,}\{\})\times$
$P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT,}\{\})$

# The Final Probabilities




$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times$$
$$P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V[6]}) \times$$
$$P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 1, \{\text{NP-C}\}) \times$$
$$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 0, \{\}) \times$$
$$P_d(\text{STOP} \mid \text{S,VP,told,V[6],LEFT,}\Delta = 0, \{\}) \times$$
$$P_{rc}(\{\} \mid \text{S, VP, told, V[6]}) \times$$
$$P_d(\text{STOP} \mid \text{S,VP,told,V[6],RIGHT,}\Delta = 1, \{\})$$

# Another Example




$$P_h(\text{V[6]} \mid \text{VP, told, V[6]})\times$$
$$P_{lc}(\{\} \mid \text{VP, V[6], told, V[6]})\times$$
$$P_d(\text{STOP} \mid \text{VP,V[6],told,V[6],LEFT},\Delta = 1,\{\})\times$$
$$P_{rc}(\{\text{NP-C, SBAR-C}\} \mid \text{VP, V[6], told, V[6]})\times$$
$$P_d(\text{NP-C(Bill,NNP)} \mid \text{VP,V[6],told,V[6],RIGHT},\Delta = 1,\{\text{NP-C, SBAR-C}\})\times$$
$$P_d(\text{NP(yesterday,NN)} \mid \text{VP,V[6],told,V[6],RIGHT},\Delta = 0,\{\text{SBAR-C}\})\times$$
$$P_d(\text{SBAR-C(that,COMP)} \mid \text{VP,V[6],told,V[6],RIGHT},\Delta = 0,\{\text{SBAR-C}\})\times$$
$$P_d(\text{STOP} \mid \text{VP,V[6],told,V[6],RIGHT},\Delta = 0,\{\})$$

# Summary

- Identify heads of rules $\Rightarrow$ dependency representations

- Presented two variants of PCFG methods applied to *lexicalized grammars*.
  - Break generation of rule down into small (markov process) steps
  - Build dependencies back up (distance, subcategorization)

- Next: we'll talk about the effectiveness of these parsers