6.864 (Fall 2007): Lecture 4
Parsing and Syntax II

Overview

- Heads in context-free rules
- The anatomy of lexicalized rules
- Dependency representations of parse trees
- Two models making use of dependencies
  - Charniak (1997)
  - Collins (1997)

Heads in Context-Free Rules

Add annotations specifying the “head” of each rule:

S	$\Rightarrow$	NP	**VP**
VP	$\Rightarrow$	**Vi**	
VP	$\Rightarrow$	**Vt**	NP
VP	$\Rightarrow$	**VP**	PP
NP	$\Rightarrow$	DT	**NN**
NP	$\Rightarrow$	**NP**	PP
PP	$\Rightarrow$	**IN**	NP

Vi	$\Rightarrow$	sleeps
Vt	$\Rightarrow$	saw
NN	$\Rightarrow$	man
NN	$\Rightarrow$	woman
NN	$\Rightarrow$	telescope
DT	$\Rightarrow$	the
IN	$\Rightarrow$	with
IN	$\Rightarrow$	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	$\Rightarrow$	NP	**VP**		(VP is the head)
VP	$\Rightarrow$	**Vt**	NP		(Vt is the head)
NP	$\Rightarrow$	DT	NN	**NN**	(NN is the head)

- A core idea in syntax (e.g., see 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

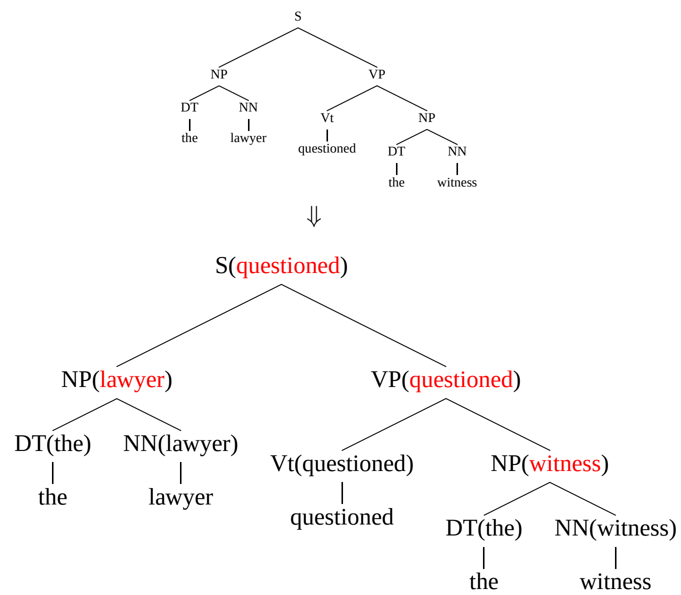
Else If the rule contains an VP: Choose the leftmost VP

Else Choose the leftmost child

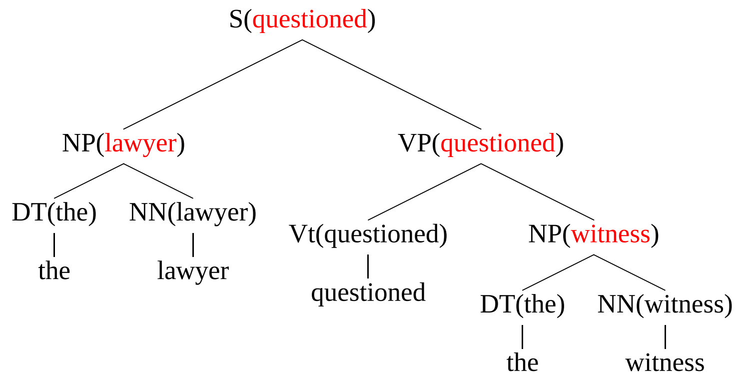
e.g.,

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)

Chomsky Normal Form

A context free grammar $G = (N, \Sigma, R, S)$ in Chomsky Normal Form is as follows

- N is a set of non-terminal symbols
- Σ is a set of terminal symbols
- R is a set of rules which take one of two forms:
  - $X \rightarrow Y_1 Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
  - $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$
- $S \in N$ is a distinguished start symbol

We can find the highest scoring parse under a PCFG in this form, in $O(n^3|R|)$ time where n is the length of the string being parsed, and $|R|$ is the number of rules in the grammar (see the dynamic programming algorithm in the previous notes)

A New Form of Grammar

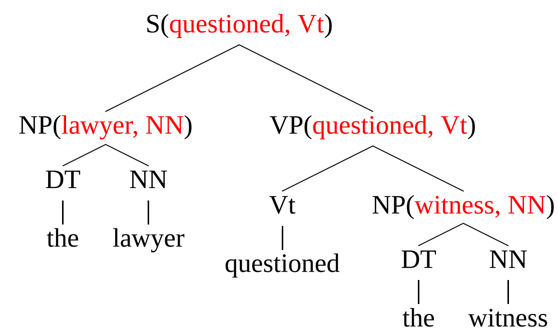
We define the following type of "lexicalized" grammar:
(we'll call this is a lexicalized Chomsky normal form grammar)

- N is a set of non-terminal symbols
- Σ is a set of terminal symbols
- R is a set of rules which take one of three forms:
  - $X(h) \rightarrow Y_1(h)\ Y_2(w)$ for $X \in N$, and $Y_1, Y_2 \in N$, and $h, w \in \Sigma$
  - $X(h) \rightarrow Y_1(w)\ Y_2(h)$ for $X \in N$, and $Y_1, Y_2 \in N$, and $h, w \in \Sigma$
  - $X(h) \rightarrow h$ for $X \in N$, and $h \in \Sigma$
- $S \in N$ is a distinguished start symbol

A New Form of Grammar

- The new form of grammar looks just like a Chomsky normal form CFG, but with potentially $O(|\Sigma|^2 \times |N|^3)$ possible rules.
- Naively, parsing an n word sentence using the dynamic programming algorithm will take $O(n^3|\Sigma|^2|N|^3)$ time. But $|\Sigma|$ can be huge!!
- Crucial observation: at most $O(n^2 \times |N|^3)$ rules can be applicable to a given sentence $w_1, w_2, \ldots w_n$ of length n. This is because any rules which contain a lexical item that is not one of $w_1 \ldots w_n$, can be safely discarded.
- The result: we can parse in $O(n^5|N|^3)$ time.

Adding Headtags to Trees



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

Overview

- Heads in context-free rules
- The anatomy of lexicalized rules
- Dependency representations of parse trees
- Two models making use of dependencies
  - Charniak (1997)
  - Collins (1997)

Non-terminals in Lexicalized rules

An example lexicalized rule:

VP(told,V) $\Rightarrow$ V(told,V)   NP(Clinton,NNP)   SBAR(that,COMP)

- Each **non-terminal** is a triple consisting of:
  1. A label
  2. A word
  3. A tag (i.e., a part-of-speech tag)
- E.g., for VP(told,V): label = VP, word = told, tag = V

  for V(told,V): label = V, word = told, tag = V

The Parent of a Lexicalized Rule

An example lexicalized rule:

VP(told,V) $\Rightarrow$ V(told,V)   NP(Clinton,NNP)   SBAR(that,COMP)

- The **parent** of the rule is the non-terminal on the left-hand-side (LHS) of the rule
- e.g., VP(told,V) in the above example
- We will also refer to the **parent label**, **parent word**, and **parent tag**. In this case:
  1. Parent label is VP
  2. Parent word is told
  3. Parent tag is V

The Head of a Lexicalized Rule

An example lexicalized rule:

VP(told,V) $\Rightarrow$ V(told,V)   NP(Clinton,NNP)   SBAR(that,COMP)

- The **head** of the rule is a single non-terminal on the right-hand-side (RHS) of the rule
- e.g., V(told,V) is the head in the above example.
- We will also refer to the **head label**, **head word**, and **head tag**. In this case:
  1. Head label is V
  2. Head word is told
  3. Head tag is V

- Note: we **always** have
  - parent word = head word
  - parent tag = head tag

The Left-Modifiers of a Lexicalized Rule

An example lexicalized rule:

VP(told,V) $\Rightarrow$ V(told,V)   NP(Clinton,NNP)   SBAR(that,COMP)

- The **left-modifiers** of the rule are any non-terminals appearing to the left of the head
- In this example there are no left-modifiers
- In general there can be any number (0 or greater) of left-modifiers

The Left-Modifiers of a Lexicalized Rule

Another example lexicalized rule:

S(told,V) $\Rightarrow$ NP(yesterday,NN) NP(Hillary,NNP) VP(told,V)

- The **left-modifiers** of the rule are any non-terminals appearing to the left of the head
- In this example there are two left-modifiers:
  - NP(yesterday,NN)
  - NP(Hillary,NNP)

The Right-Modifiers of a Lexicalized Rule

An example lexicalized rule:

VP(told,V) $\Rightarrow$ V(told,V)   NP(Clinton,NNP)   SBAR(that,COMP)

- The **right-modifiers** of the rule are any non-terminals appearing to the right of the head
- In this example there are two right-modifiers:
  - NP(Clinton,NNP)
  - SBAR(that,COMP)
- In general there can be any number (0 or greater) of right-modifiers

The General Form of a Lexicalized Rule

- The general form of a lexicalized rule is as follows:

$$X(h,t) \Rightarrow L_n(lw_n, lt_n) \ldots L_1(lw_1, lt_1)\ H(h,t)\ R_1(rw_1, rt_1) \ldots R_m(rw_m, rt_m)$$

- $X(h,t)$ is the parent of the rule
- $H(h,t)$ is the head of the rule
- There are n left modifiers, $L_i(lw_i, lt_i)$ for $i = 1 \ldots n$
- There are m right-modifiers, $R_i(rw_i, rt_i)$ for $i = 1 \ldots m$
- There can be zero or more left or right modifiers: i.e., $n \geq 0$ and $m \geq 0$

- X, H, L_i for $i = 1 \ldots n$ and R_i for $i = 1 \ldots m$ are **labels**
- h, lw_i for $i = 1 \ldots n$ and rw_i for $i = 1 \ldots m$ are **words**
- t, lt_i for $i = 1 \ldots n$ and rt_i for $i = 1 \ldots m$ are **tags**

Overview

- Heads in context-free rules
- The anatomy of lexicalized rules
- Dependency representations of parse trees
- Two models making use of dependencies
  - Charniak (1997)
  - Collins (1997)

Headwords and Dependencies

- A new representation: a tree is represented as a set of *dependencies*, not a set of *context-free rules*
- A **dependency** is an 8-tuple:

(head-word,	head-tag,
modifer-word,	modifer-tag,
parent-label,	head-label,
modifier-label,	direction)

- Each rule with n children contributes $(n-1)$ dependencies. **There is one dependency for each left or right modifier**

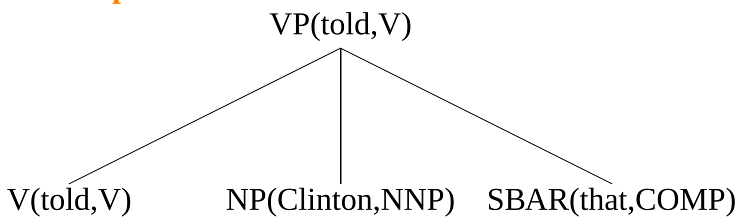
VP(questioned,Vt) $\Rightarrow$ Vt(questioned,Vt) NP(lawyer,NN)

$\Downarrow$

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

Headwords and Dependencies

An example rule:



This rule contributes two dependencies:

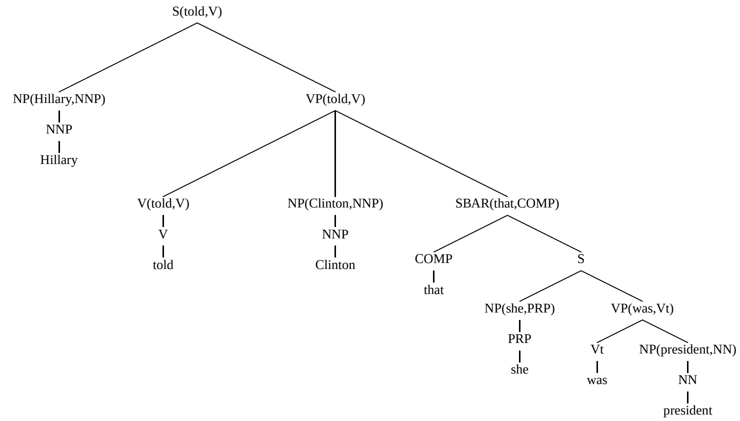
head-word	head-tag	mod-word	mod-tag	parent-label	head-label	mod-label	direction
told	V	Clinton	NNP	VP	V	NP	RIGHT
told	V	that	COMP	VP	V	SBAR	RIGHT

A Special Case: the Top of the Tree

TOP
|
S(told,V)

⇓

(__, __, told, V, TOP, S, __, SPECIAL)



(__	__	told	V	TOP	S	__	SPECIAL)
(told	V	Hillary	NNP	S	VP	NP	LEFT)
(told	V	Clinton	NNP	VP	V	NP	RIGHT)
(told	V	that	COMP	VP	V	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)

Overview

- Heads in context-free rules
- The anatomy of lexicalized rules
- Dependency representations of parse trees
- Two models making use of dependencies
  - Charniak (1997)
  - Collins (1997)

A Model from Charniak (1997)

S(questioned,Vt)

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

S(questioned,Vt)

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

$\Downarrow$ $\quad$ $Prob$(lawyer | S(questioned,Vt),VP,NP(NN))

S(questioned,Vt)

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

The General Form of Charniak's Model

- The general form of a lexicalized rule is as follows:

$$X(h,t) \Rightarrow L_n(lw_n,lt_n) \ldots L_1(lw_1,lt_1)\ H(h,t)\ R_1(rw_1,rt_1) \ldots R_m(rw_m,rt_m)$$

- Charniak's model decomposes the probability of each rule as:

$$Prob(X(h,t) \Rightarrow L_n(lt_n) \ldots L_1(lt_1) H(t) R_1(rt_1) \ldots R_m(rt_m) \mid X(h,t))$$
$$\times \prod_{i=1}^{n} Prob(lw_i \mid X(h,t), H, L_i(lt_i))$$
$$\times \prod_{i=1}^{m} Prob(rw_i \mid X(h,t), H, R_i(rt_i))$$

Dissecting Charniak's Model: Rule Probabilities

- First term of Charniak's model:

$$Prob(X(h,t) \Rightarrow L_n(lt_n) \ldots L_1(lt_1) H(t) R_1(rt_1) \ldots R_m(rt_m) \mid X(h,t))$$

- This corresponds to a choice of context-free rule, at this stage no modifier words are generated
- For our old example rule,

  VP(told,V) $\Rightarrow$ V(told,V) NP(Clinton,NNP) SBAR(that,COMP)

  we would have

  P(VP(told,V) $\Rightarrow$ V(V) NP(NNP) SBAR(COMP) | VP(told,V))

Dissecting Charniak's Model: Modifier Probabilities

- For each right modifier, there is a term

$$Prob(rw_i \mid X(h,t), H, R_i(rt_i))$$

- This corresponds to generating the modifier word rw_i for the i'th right modifier.
- This probability is conditioned on
  1. the head-word h,
  2. the labels X, H, and R_i
  3. the tags t and rt_i.
- We now have a probability that is sensitive to the **dependency** between rw_i and h
- There is a similar probability for each left modifier

Smoothed Estimation

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

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

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

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

Smoothed Estimation

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

$$\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{S(questioned,Vt), VP, NP(NN)})}{Count(\text{S(questioned,Vt), VP, NP(NN)})}$$

$$+\lambda_4 \times \frac{Count(\text{lawyer} \mid \text{S(\_\_,Vt), VP, NP(NN)})}{Count(\text{S(\_\_,Vt), VP, NP(NN)})}$$

$$+\lambda_5 \times \frac{Count(\text{lawyer} \mid \text{NN})}{Count(\text{NN})}$$

- Where $0 \leq \lambda_3, \lambda_4, \lambda_5 \leq 1$, and $\lambda_3 + \lambda_4 + \lambda_5 = 1$

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

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

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

$$\times \; (\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{S(questioned,Vt), VP, NP(NN)})}{Count(\text{S(questioned,Vt), VP, NP(NN)})}$$

$$+\lambda_4 \times \frac{Count(\text{lawyer} \mid \text{S(\_\_,Vt), VP, NP(NN)})}{Count(\text{S(\_\_,Vt), VP, NP(NN)})}$$

$$+\lambda_5 \times \frac{Count(\text{lawyer} \mid \text{NN})}{Count(\text{NN})}$$

Motivation for Breaking Down Rules

- First step of decomposition of (Charniak 1997):

S(questioned,Vt)

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

S(questioned,Vt)

NP(__,NN) VP(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

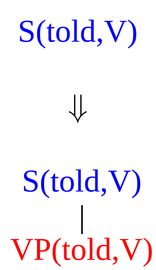
Motivation for Breaking Down Rules

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

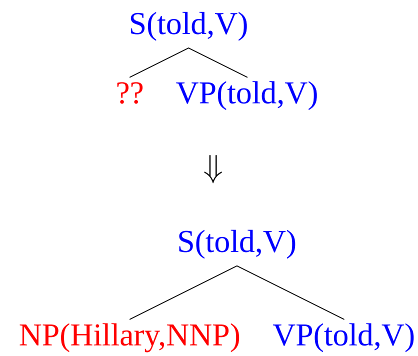
- Step 1: generate category of head child



P_h(VP | S, told, V)

Modeling Rule Productions as Markov Processes

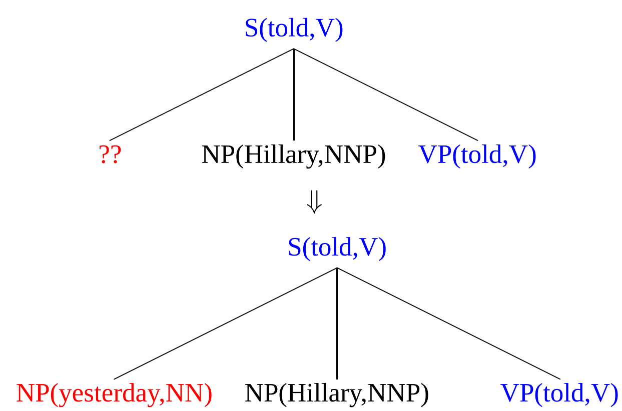
- Step 2: generate left modifiers in a Markov chain



P_h(VP | S, told, V) $\times$ P_d(NP(Hillary,NNP) | S,VP,told,V,LEFT)

Modeling Rule Productions as Markov Processes

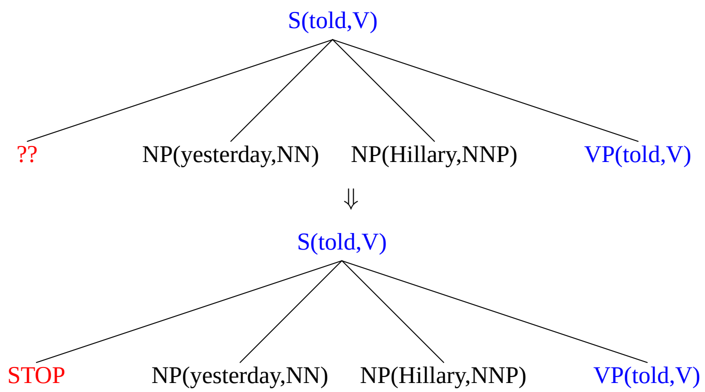
- Step 2: generate left modifiers in a Markov chain



P_h(VP | S, told, V) $\times$ P_d(NP(Hillary,NNP) | S,VP,told,V,LEFT)$\times$
P_d(NP(yesterday,NN) | S,VP,told,V,LEFT)

Modeling Rule Productions as Markov Processes

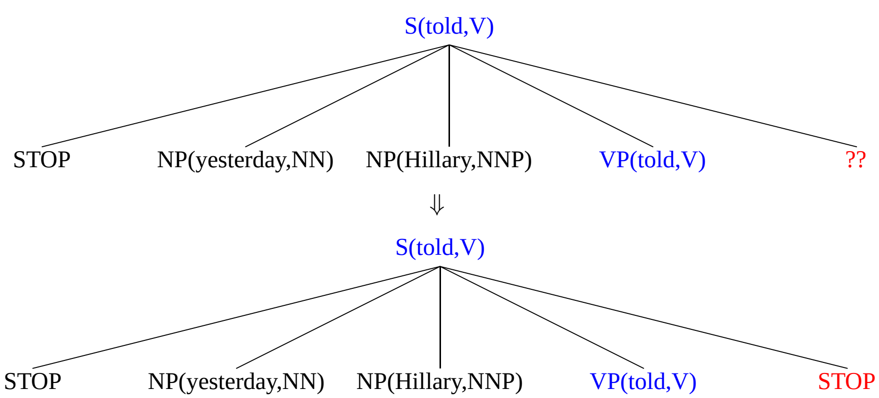
- Step 2: generate left modifiers in a Markov chain



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

Modeling Rule Productions as Markov Processes

- Step 3: generate right modifiers in a Markov chain



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