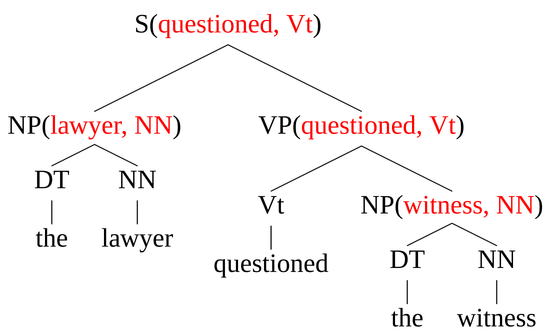
6.864 (Fall 2007): Lecture 5
Parsing and Syntax III

Recap: Adding Head Words/Tags to Trees



- We now have *lexicalized* context-free rules, e.g.,

$$\text{S(questioned,Vt)} \Rightarrow \quad \text{NP(lawyer,NN)} \quad \text{VP(questioned,Vt)}$$

Recap: Lexicalized PCFGs

- We now need to estimate rule probabilities such as

$$Prob(\text{S(questioned,Vt)} \Rightarrow \quad \text{NP(lawyer,NN)} \quad \text{VP(questioned,Vt)} \mid \text{S(questioned,Vt)})$$

- Sparse data is a problem. We have a **huge** number of non-terminals, and a **huge** number of possible rules. We have to work hard to estimate these rule probabilities...

- Once we have estimated these rule probabilities, we can find the highest scoring parse tree under the lexicalized PCFG using dynamic programming methods (see Problem set 1).

Recap: 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:

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

- For example,

$$Prob(\text{S(questioned,Vt)} \Rightarrow \quad \text{NP(lawyer,NN)} \quad \text{VP(questioned,Vt)} \mid \text{S(questioned,Vt)})$$

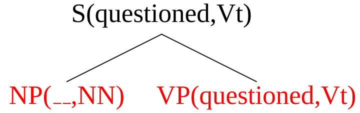
$$= Prob(\text{S(questioned,Vt)} \Rightarrow \quad \text{NP(NN)} \quad \text{VP(Vt)} \mid \text{S(questioned,Vt)})$$
$$\times Prob(\text{lawyer} \mid \text{S(questioned,Vt), VP, NP(NN)})$$

Motivation for Breaking Down Rules

- First step of decomposition of (Charniak 1997):

S(questioned,Vt)

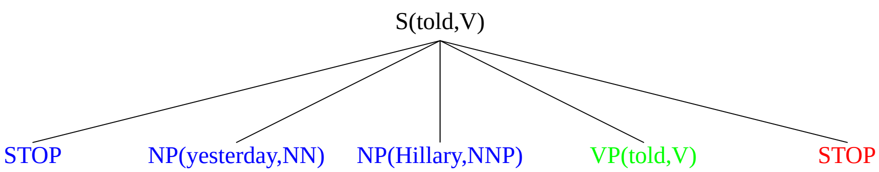
$\Downarrow$ $\quad P$(NP(NN) VP | S(questioned,Vt))



- 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

Modeling Rule Productions as Markov Processes

- Collins (1997), Model 1



We first generate the **head label** of the rule
Then generate the **left modifiers**
Then generate the **right modifiers**

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) $\times$
P_d(STOP | S,VP,told,V,LEFT) $\times$
P_d(STOP | S,VP,told,V,RIGHT)

The General Form of Model 1

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

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

- Collins model 1 decomposes the probability of each rule as:

$$\begin{aligned}
&P_h(H \mid X, h, t) \times \\
&\prod_{i=1}^{n} P_d(L_i(lw_i, lt_i) \mid X, H, h, t, \text{LEFT}) \times \\
&P_d(\text{STOP} \mid X, H, h, t, \text{LEFT}) \times \\
&\prod_{i=1}^{m} P_d(R_i(rw_i, rt_i) \mid X, H, h, t, \text{RIGHT}) \times \\
&P_d(\text{STOP} \mid X, H, h, t, \text{RIGHT})
\end{aligned}$$

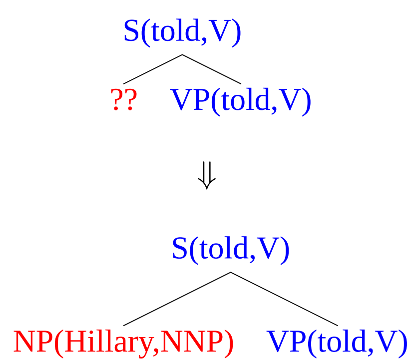
- P_h term is a head-label probability
- P_d terms are dependency probabilities
- Both the P_h and P_d terms are smoothed, using similar techniques to Charniak's model

Overview of Today's Lecture

- Refinements to Model 1
- Evaluating parsing models
- Extensions to the parsing models

A Refinement: Adding a Distance Variable

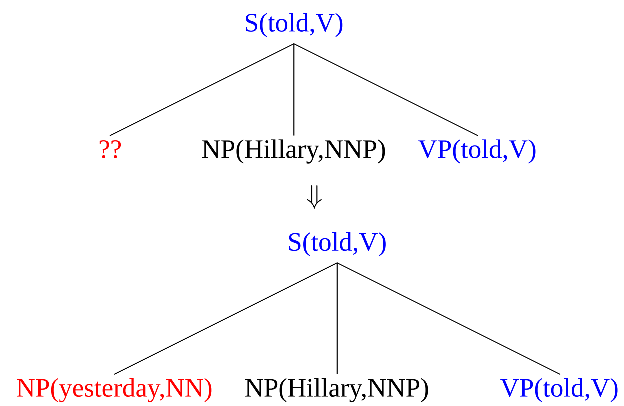
- $\Delta = 1$ if position is adjacent to the head, 0 otherwise



P_h(VP | S, told, V)$\times$
P_d(NP(Hillary,NNP) | S,VP,told,V,LEFT,$\Delta = 1$)

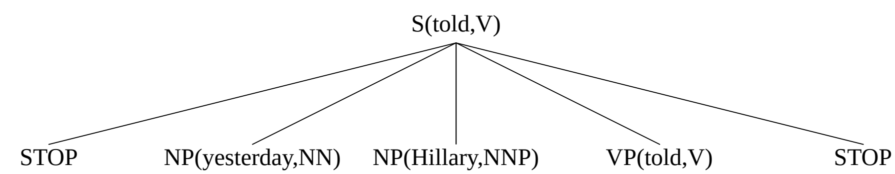
A Refinement: Adding a Distance Variable

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



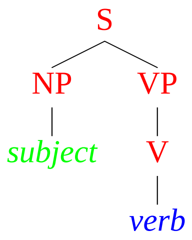
P_h(VP | S, told, V) $\times$ P_d(NP(Hillary,NNP) | S,VP,told,V,LEFT)$\times$
P_d(NP(yesterday,NN) | S,VP,told,V,LEFT,$\Delta = 0$)

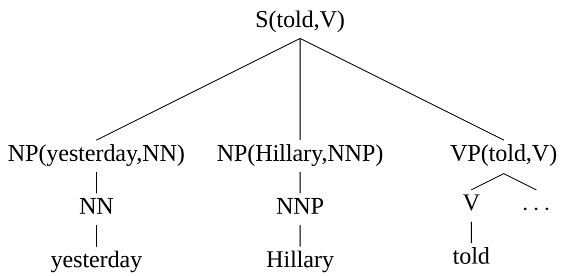
The Final Probabilities



P_h(VP | S, told, V)$\times$
P_d(NP(Hillary,NNP) | S,VP,told,V,LEFT,$\Delta = 1$)$\times$
P_d(NP(yesterday,NN) | S,VP,told,V,LEFT,$\Delta = 0$)$\times$
P_d(STOP | S,VP,told,V,LEFT,$\Delta = 0$)$\times$
P_d(STOP | S,VP,told,V,RIGHT,$\Delta = 1$)

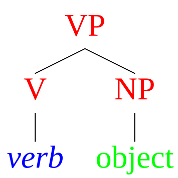
Adding the Complement/Adjunct Distinction

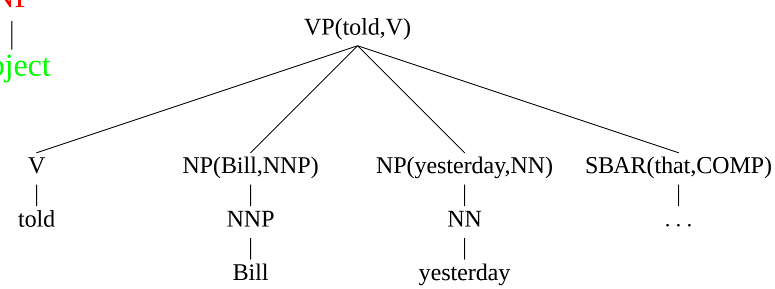




- *Hillary* is the subject
- *yesterday* is a temporal modifier
- **But nothing to distinguish them.**

Adding the Complement/Adjunct Distinction





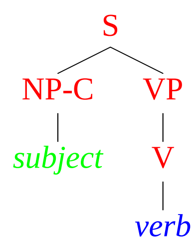
- *Bill* is the object
- *yesterday* is a temporal modifier
- **But nothing to distinguish them.**

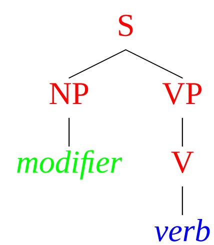
Complements vs. Adjuncts

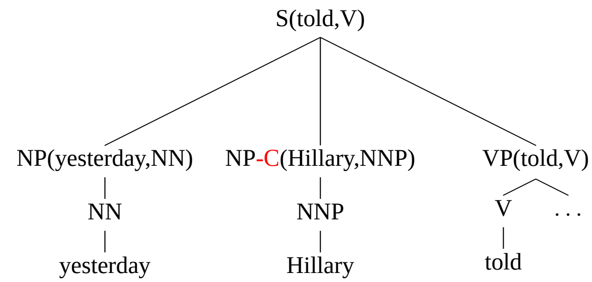
- Complements are closely related to the head they modify, adjuncts are more indirectly related
- Complements are usually arguments of the thing they modify
  yesterday Hillary told . . . $\Rightarrow$ *Hillary* is doing the *telling*
- Adjuncts add modifying information: time, place, manner etc.
  yesterday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

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

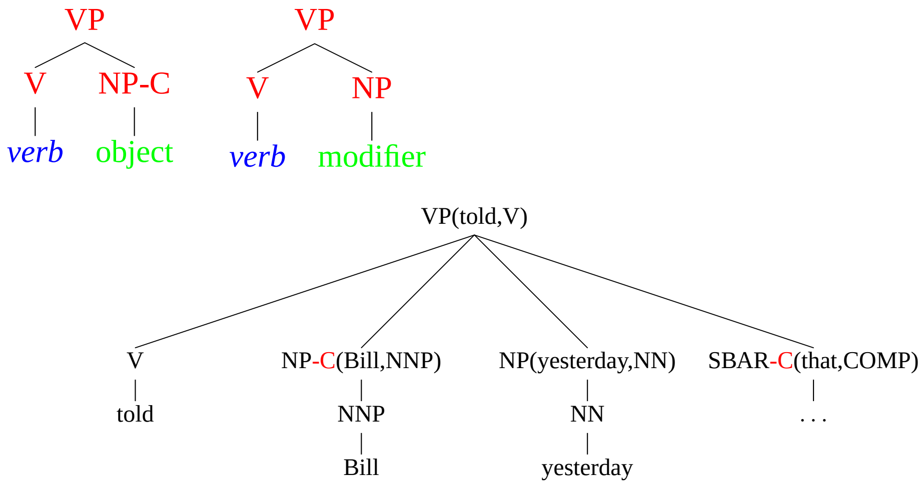
Adding Tags Making the Complement/Adjunct Distinction





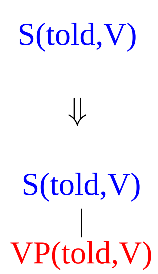


Adding Tags Making the Complement/Adjunct Distinction



Adding Subcategorization Probabilities

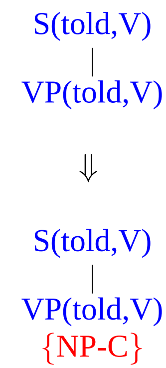
- Step 1: generate category of head child



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

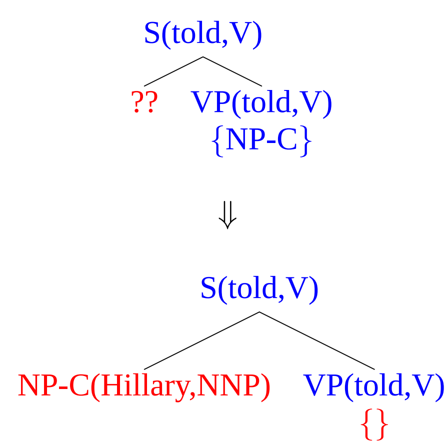
Adding Subcategorization Probabilities

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

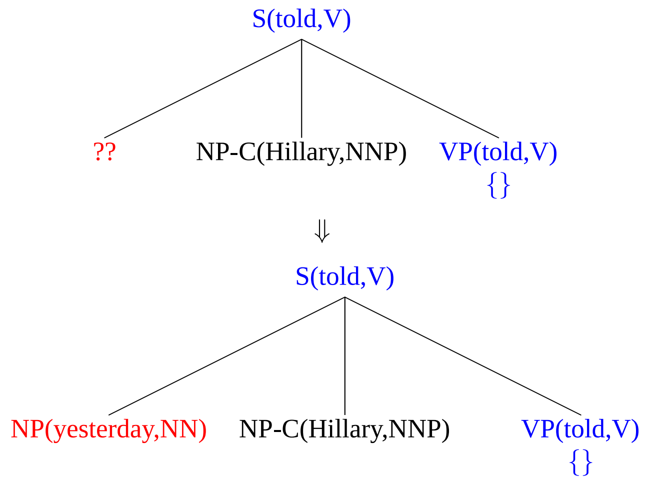


$P_h(\text{VP} \mid \text{S, told, V}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V})$

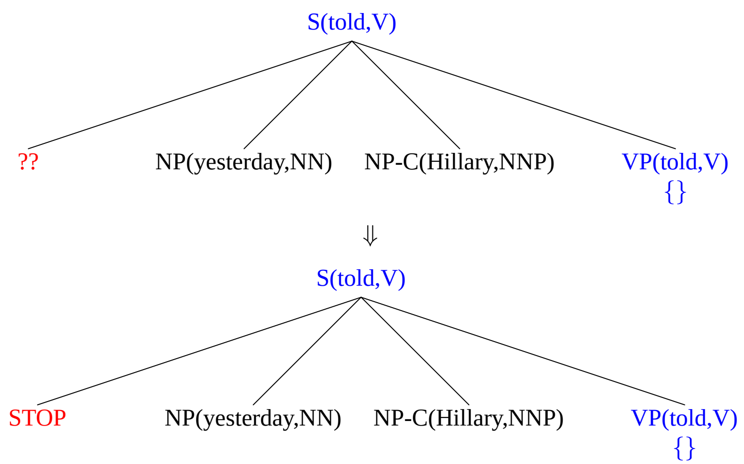
- Step 3: generate left modifiers in a Markov chain



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

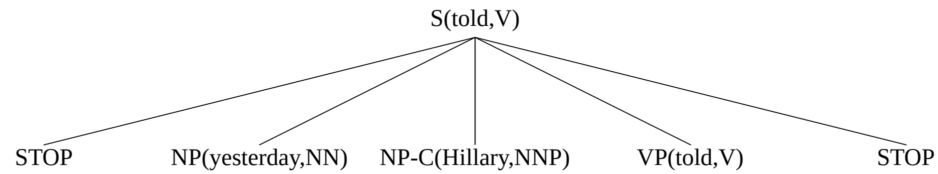


$P_h(\text{VP} \mid \text{S, told, V}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V})$
$P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V,LEFT,}\{\text{NP-C}\}) \times$
$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V,LEFT,}\{\})$



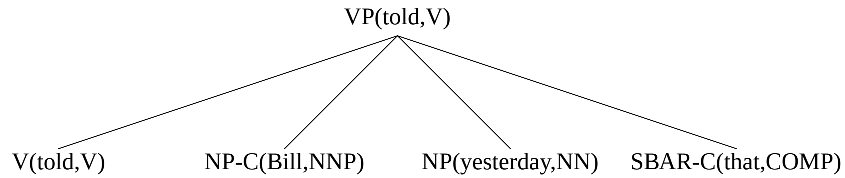
$P_h(\text{VP} \mid \text{S, told, V}) \times P_{lc}(\{\text{NP-C}\} \mid \text{S, VP, told, V})$
$P_d(\text{NP-C(Hillary,NNP)} \mid \text{S,VP,told,V,LEFT,}\{\text{NP-C}\}) \times$
$P_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,V,LEFT,}\{\}) \times$
$P_d(\text{STOP} \mid \text{S,VP,told,V,LEFT,}\{\})$

The Final Probabilities



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

Another Example



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

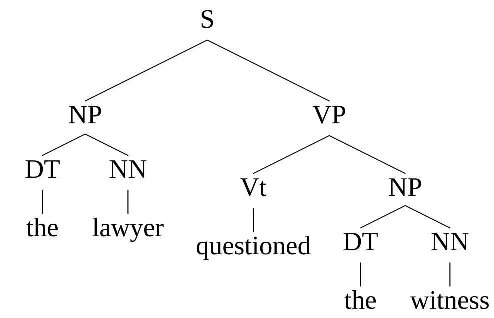
Summary

- Identify heads of rules ⇒ 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)

Overview of Today's Lecture

- Refinements to Model 1
- Evaluating parsing models
- Extensions to the parsing models

Evaluation: Representing Trees as Constituents



⇓

Label	Start Point	End Point
NP	1	2
NP	4	5
VP	3	5
S	1	5

Precision and Recall

Label	Start Point	End Point
NP	1	2
NP	4	5
NP	4	8
PP	6	8
NP	7	8
VP	3	8
S	1	8

Label	Start Point	End Point
NP	1	2
NP	4	5
PP	6	8
NP	7	8
VP	3	8
S	1	8

- G = number of constituents in gold standard = 7
- P = number in parse output = 6
- C = number correct = 6

$$\text{Recall} = 100\% \times \frac{C}{G} = 100\% \times \frac{6}{7} \qquad \text{Precision} = 100\% \times \frac{C}{P} = 100\% \times \frac{6}{6}$$

Results

Method	Recall	Precision
PCFGs (Charniak 97)	70.6%	74.8%
Conditional Models – Decision Trees (Magerman 95)	84.0%	84.3%
Generative Lexicalized Model (Charniak 97)	86.7%	86.6%
Model 1 (no subcategorization)	87.5%	87.7%
Model 2 (subcategorization)	88.1%	88.3%

Effect of the Different Features

MODEL	A	V	R	P
Model 1	NO	NO	75.0%	76.5%
Model 1	YES	NO	86.6%	86.7%
Model 1	YES	YES	87.8%	88.2%
Model 2	NO	NO	85.1%	86.8%
Model 2	YES	NO	87.7%	87.8%
Model 2	YES	YES	88.7%	89.0%

Results on Section 0 of the WSJ Treebank. Model 1 has no subcategorization, Model 2 has subcategorization. A = YES, V = YES mean that the adjacency/verb conditions respectively were used in the distance measure. **R/P** = recall/precision.

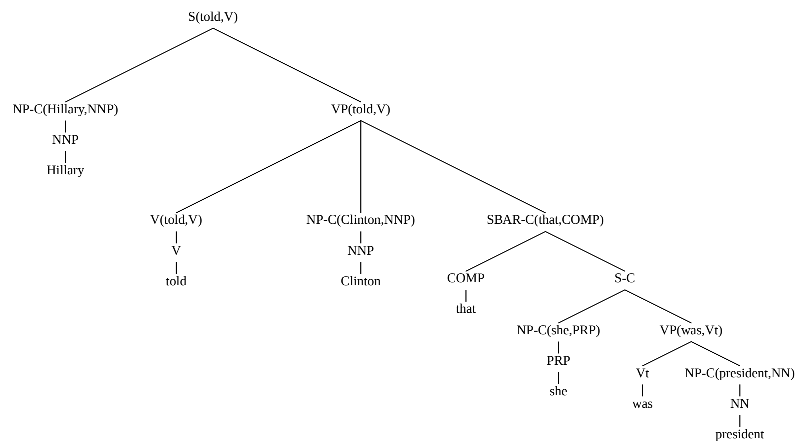
Weaknesses of Precision and Recall

Label	Start Point	End Point
NP	1	2
NP	4	5
NP	4	8
PP	6	8
NP	7	8
VP	3	8
S	1	8

Label	Start Point	End Point
NP	1	2
NP	4	5
PP	6	8
NP	7	8
VP	3	8
S	1	8

NP attachment:
(S (NP The men) (VP dumped (NP (NP large sacks) (PP of (NP the substance)))))

VP attachment:
(S (NP The men) (VP dumped (NP large sacks) (PP of (NP the substance))))



(--	--	told	V	TOP	S	--	SPECIAL)
(told	V	Hillary	NNP	S	VP	NP-C	LEFT)
(told	V	Clinton	NNP	VP	V	NP-C	RIGHT)
(told	V	that	COMP	VP	V	SBAR-C	RIGHT)
(that	COMP	was	Vt	SBAR-C	COMP	S-C	RIGHT)
(was	Vt	she	PRP	S-C	VP	NP-C	LEFT)
(was	Vt	president	NN	VP	Vt	NP-C	RIGHT)

Dependency Accuracies

- All parses for a sentence with n words have n dependencies
  Report a single figure, dependency accuracy

- Model 2 with all features scores 88.3% dependency accuracy
  (91% if you ignore non-terminal labels on dependencies)

- Can calculate precision/recall on particular dependency types
  e.g., look at all subject/verb dependencies $\Rightarrow$
  all dependencies with label **(S,VP,NP-C,LEFT)**

$$\text{Recall} = \frac{\text{number of subject/verb dependencies correct}}{\text{number of subject/verb dependencies in gold standard}}$$

$$\text{Precision} = \frac{\text{number of subject/verb dependencies correct}}{\text{number of subject/verb dependencies in parser's output}}$$

R	CP	P	Count	Relation	Rec	Prec
1	29.65	29.65	11786	NPB TAG TAG L	94.60	93.46
2	40.55	10.90	4335	PP TAG NP-C R	94.72	94.04
3	48.72	8.17	3248	S VP NP-C L	95.75	95.11
4	54.03	5.31	2112	NP NPB PP R	84.99	84.35
5	59.30	5.27	2095	VP TAG NP-C R	92.41	92.15
6	64.18	4.88	1941	VP TAG VP-C R	97.42	97.98
7	68.71	4.53	1801	VP TAG PP R	83.62	81.14
8	73.13	4.42	1757	TOP TOP S R	96.36	96.85
9	74.53	1.40	558	VP TAG SBAR-C R	94.27	93.93
10	75.83	1.30	518	QP TAG TAG R	86.49	86.65
11	77.08	1.25	495	NP NPB NP R	74.34	75.72
12	78.28	1.20	477	SBAR TAG S-C R	94.55	92.04
13	79.48	1.20	476	NP NPB SBAR R	79.20	79.54
14	80.40	0.92	367	VP TAG ADVP R	74.93	78.57
15	81.30	0.90	358	NPB TAG NPB L	97.49	92.82
16	82.18	0.88	349	VP TAG TAG R	90.54	93.49
17	82.97	0.79	316	VP TAG SG-C R	92.41	88.22

Accuracy of the 17 most frequent dependency types in section 0 of the treebank, as recovered by model 2. R = rank; CP = cumulative percentage; P = percentage; Rec = Recall; Prec = precision.

Type	Sub-type	Description	Count	Recall	Precision
Complement to a verb	S VP NP-C L	Subject	3248	95.75	95.11
	VP TAG NP-C R	Object	2095	92.41	92.15
6495 = 16.3% of all cases	VP TAG SBAR-C R		558	94.27	93.93
	VP TAG SG-C R		316	92.41	88.22
	VP TAG S-C R		150	74.67	78.32
	S VP S-C L		104	93.27	78.86
	S VP SG-C L		14	78.57	68.75
	...				
	TOTAL		6495	93.76	92.96
Other complements	PP TAG NP-C R		4335	94.72	94.04
	VP TAG VP-C R		1941	97.42	97.98
7473 = 18.8% of all cases	SBAR TAG S-C R		477	94.55	92.04
	SBAR WHNP SG-C R		286	90.56	90.56
	PP TAG SG-C R		125	94.40	89.39
	SBAR WHADVP S-C R		83	97.59	98.78
	PP TAG PP-C R		51	84.31	70.49
	SBAR WHNP S-C R		42	66.67	84.85
	SBAR TAG SG-C R		23	69.57	69.57
	PP TAG S-C R		18	38.89	63.64
	SBAR WHPP S-C R		16	100.00	100.00
	S ADJP NP-C L		15	46.67	46.67
	PP TAG SBAR-C R		15	100.00	88.24
	...				
	TOTAL		7473	94.47	94.12

Type	Sub-type	Description	Count	Recall	Precision
PP modifi cation	`NP NPB PP R`		2112	84.99	84.35
	`VP TAG PP R`		1801	83.62	81.14
4473 = 11.2% of all cases	`S VP PP L`		287	90.24	81.96
	`ADJP TAG PP R`		90	75.56	78.16
	`ADVP TAG PP R`		35	68.57	52.17
	`NP NP PP R`		23	0.00	0.00
	`PP PP PP L`		19	21.05	26.67
	`NAC TAG PP R`		12	50.00	100.00
	...				
	TOTAL		4473	82.29	81.51
Coordination	`NP NP NP R`		289	55.71	53.31
	`VP VP VP R`		174	74.14	72.47
763 = 1.9% of all cases	`S S S R`		129	72.09	69.92
	`ADJP TAG TAG R`		28	71.43	66.67
	`VP TAG TAG R`		25	60.00	71.43
	`NX NX NX R`		25	12.00	75.00
	`SBAR SBAR SBAR R`		19	78.95	83.33
	`PP PP PP R`		14	85.71	63.16
	...				
	TOTAL		763	61.47	62.20

Type	Sub-type	Description	Count	Recall	Precision
Mod'n within BaseNPs	`NPB TAG TAG L`		11786	94.60	93.46
	`NPB TAG NPB L`		358	97.49	92.82
12742 = 29.6% of all cases	`NPB TAG TAG R`		189	74.07	75.68
	`NPB TAG ADJP L`		167	65.27	71.24
	`NPB TAG QP L`		110	80.91	81.65
	`NPB TAG NAC L`		29	51.72	71.43
	`NPB NX TAG L`		27	14.81	66.67
	`NPB QP TAG L`		15	66.67	76.92
	...				
	TOTAL		12742	93.20	92.59
Mod'n to NPs	`NP NPB NP R`	Appositive	495	74.34	75.72
	`NP NPB SBAR R`	Relative clause	476	79.20	79.54
1418 = 3.6% of all cases	`NP NPB VP R`	Reduced relative	205	77.56	72.60
	`NP NPB SG R`		63	88.89	81.16
	`NP NPB PRN R`		53	45.28	60.00
	`NP NPB ADVP R`		48	35.42	54.84
	`NP NPB ADJP R`		48	62.50	69.77
	...				
	TOTAL		1418	73.20	75.49

Type	Sub-type	Description	Count	Recall	Precision
Sentential head	`TOP TOP S R`		1757	96.36	96.85
	`TOP TOP SINV R`		89	96.63	94.51
1917 = 4.8% of all cases	`TOP TOP NP R`		32	78.12	60.98
	`TOP TOP SG R`		15	40.00	33.33
	...				
	TOTAL		1917	94.99	94.99
Adjunct to a verb	`VP TAG ADVP R`		367	74.93	78.57
	`VP TAG TAG R`		349	90.54	93.49
2242 = 5.6% of all cases	`VP TAG ADJP R`		259	83.78	80.37
	`S VP ADVP L`		255	90.98	84.67
	`VP TAG NP R`		187	66.31	74.70
	`VP TAG SBAR R`		180	74.44	72.43
	`VP TAG SG R`		159	60.38	68.57
	`S VP TAG L`		115	86.96	90.91
	`S VP SBAR L`		81	88.89	85.71
	`VP TAG ADVP L`		79	51.90	49.40
	`S VP PRN L`		58	25.86	48.39
	`S VP NP L`		45	66.67	63.83
	`S VP SG L`		28	75.00	52.50
	`VP TAG PRN R`		27	3.70	12.50
	`VP TAG S R`		11	9.09	100.00
	...				
	TOTAL		2242	75.11	78.44

Some Conclusions about Errors in Parsing

- "Core" sentential structure (complements, NP chunks) recovered with over 90% accuracy.

- Attachment ambiguities involving adjuncts are resolved with much lower accuracy ($\approx 80\%$ for PP attachment, $\approx 50-60\%$ for coordination).

Overview of Today's Lecture

- Refinements to Model 1
- Evaluating parsing models
- Extensions to the parsing models

Trigram Language Models (from Lecture 2)

Step 1: The chain rule (note that $w_{n+1} = \text{STOP}$)

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n+1} P(w_i \mid w_1 \ldots w_{i-1})$$

Step 2: Make Markov independence assumptions:

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n+1} P(w_i \mid w_{i-2}, w_{i-1})$$

For Example

$$\begin{aligned} P(\text{the, dog, laughs}) &= P(\text{the} \mid \text{START}) \times P(\text{dog} \mid \text{START, the}) \\ &\quad \times P(\text{laughs} \mid \text{the, dog}) \times P(\text{STOP} \mid \text{dog, laughs}) \end{aligned}$$

Parsing Models as Language Models

- Generative models assign a probability $P(T, S)$ to each tree/sentence pair
- Say sentence is S, set of parses for S is $\mathcal{T}(S)$, then

$$P(S) = \sum_{T \in \mathcal{T}(S)} P(T, S)$$

- Can calculate perplexity for parsing models

A Quick Reminder of Perplexity

- We have some test data, n sentences

$$S_1, S_2, S_3, \ldots, S_n$$

- We could look at the probability under our model $\prod_{i=1}^{n} P(S_i)$. Or more conveniently, the *log probability*

$$\log \prod_{i=1}^{n} P(S_i) = \sum_{i=1}^{n} \log P(S_i)$$

- In fact the usual evaluation measure is *perplexity*

$$\text{Perplexity} = 2^{-x} \quad \text{where} \quad x = \frac{1}{W} \sum_{i=1}^{n} \log P(S_i)$$

and W is the total number of words in the test data.

Trigrams Can't Capture Long-Distance Dependencies

Actual Utterance: He is a resident of the U.S. and of the U.K.

Recognizer Output: He is a resident of the U.S. and *that* the U.K.

- Bigram *and that* is around 15 times as frequent as *and of*
  ⇒ Bigram model gives over 10 times greater probability to incorrect string

- Parsing models assign 78 times higher probability to the correct string

Examples of Long-Distance Dependencies

Subject/verb dependencies

Microsoft, the world's largest software company, acquired . . .

Object/verb dependencies

. . . acquired the New-York based software company . . .

Appositives

Microsoft, the world's largest software company, acquired . . .

Verb/Preposition Collocations

I put the coffee mug on the table

The USA elected the son of George Bush Sr. as president

Coordination

She said that . . . and that . . .

Work on Parsers as Language Models

- "The Structured Language Model". Ciprian Chelba and Fred Jelinek, see also recent work by Peng Xu, Ahmad Emami and Fred Jelinek.

- "Probabilistic Top-Down Parsing and Language Modeling". Brian Roark.

- "Immediate Head-Parsing for Language Models". Eugene Charniak.

Some Perplexity Figures from (Charniak, 2000)

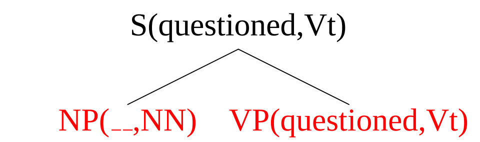
Model	Trigram	Grammar	Interpolation
Chelba and Jelinek	167.14	158.28	148.90
Roark	167.02	152.26	137.26
Charniak	167.89	144.98	133.15

- *Interpolation* is a mixture of the trigram and grammatical models

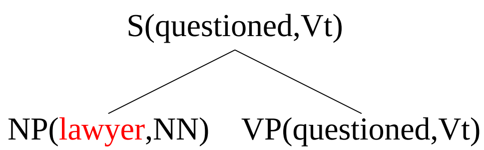
- Chelba and Jelinek, Roark use trigram information in their grammatical models, Charniak doesn't!

- Note: Charniak's parser in these experiments is as described in (Charniak 2000), and makes use of Markov processes generating rules (a shift away from the Charniak 1997 model).

Extending Charniak's Parsing Model



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



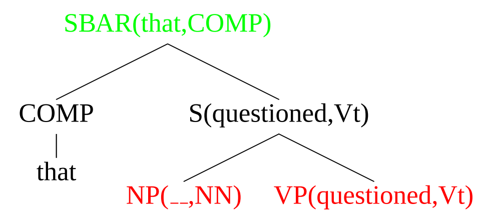
Extending Charniak's Parsing Model

She said that the lawyer questioned him

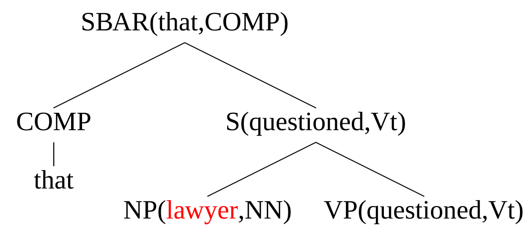
$\Rightarrow$ bigram lexical probabilies

$P(\text{questioned} \mid \text{SBAR,COMP,S,Vt, that,COMP})$
$P(\text{lawyer} \mid \text{S,VP,NP,NN, questioned,Vt})$
$P(\text{him} \mid \text{VP,Vt,NP,PRP, questioned,Vt})$ ...

Adding Syntactic Trigrams



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



Extending Charniak's Parsing Model

She said that the lawyer questioned him

$\Rightarrow$ trigram lexical probabilies

$P(\text{questioned} \mid \text{SBAR,COMP,S,Vt, that,COMP, said})$
$P(\text{lawyer} \mid \text{S,VP,NP,NN, questioned,Vt, that})$
$P(\text{him} \mid \text{VP,Vt,NP,PRP, questioned,Vt,that})$ ...

Some Perplexity Figures from (Charniak, 2000)

Model	Trigram	Grammar	Interpolation
Chelba and Jelinek	167.14	158.28	148.90
Roark	167.02	152.26	137.26
Charniak (Bigram)	167.89	144.98	133.15
Charniak (Trigram)	167.89	130.20	126.07

Model 3: A Model of Wh-Movement

- Examples of Wh-movement:

Example 1 The person (SBAR who TRACE bought the shoes)

Example 2 The shoes (SBAR that I bought TRACE last week)

Example 3 The person (SBAR who I bought the shoes from TRACE)

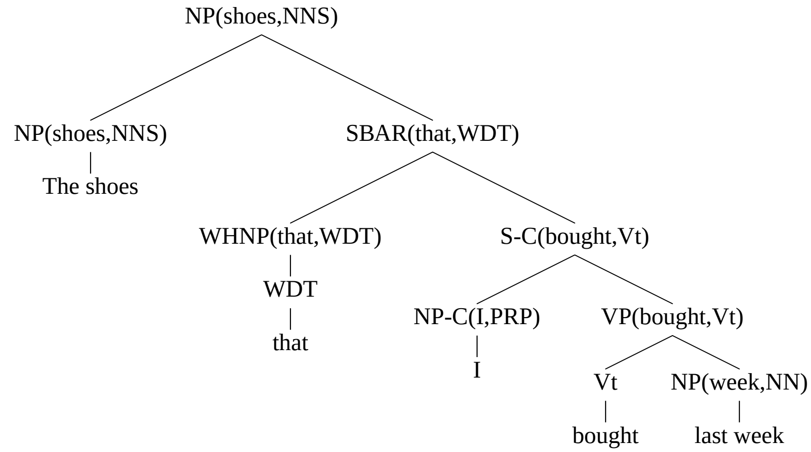
Example 4 The person (SBAR who Jeff said I bought the shoes from TRACE)

- Key ungrammatical examples:

Example 1 The person (SBAR who Fran and TRACE bought the shoes)
(derived from *Fran and Jeff bought the shoes*)

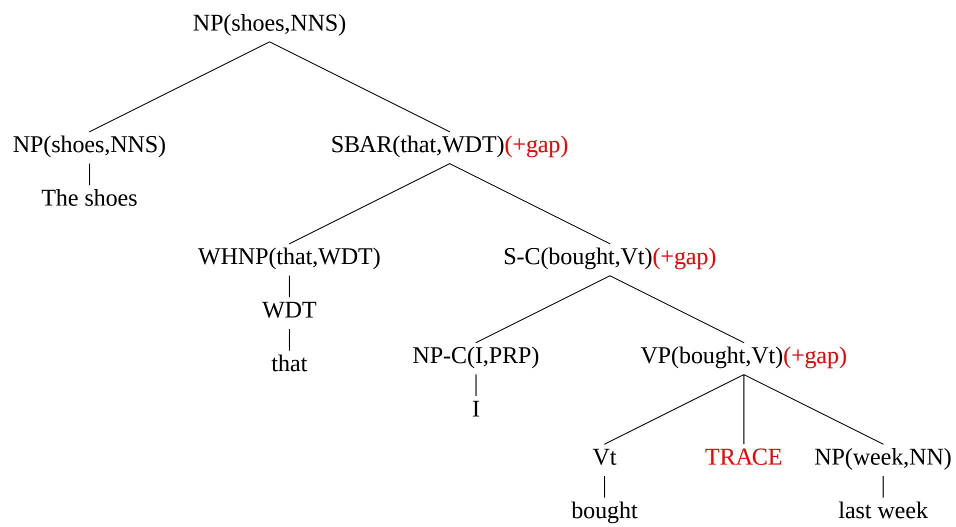
Example 2
The store (SBAR that Jeff bought the shoes because Fran likes TRACE)
(derived from *Jeff bought the shoes because Fran likes the store*)

The Parse Trees at this Stage



It’s difficult to recover “shoes” as the object of “bought”

Adding Gaps and Traces

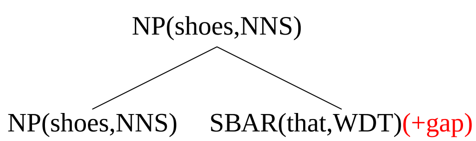


It’s easy to recover “shoes” as the object of “bought”

Adding Gaps and Traces

- This information can be recovered from the treebank
- Doubles the number of non-terminals (with/without gaps)
- Similar to treatment of Wh-movement in GPSG (generalized phrase structure grammar)
- If our parser recovers this information, it's easy to recover syntactic relations

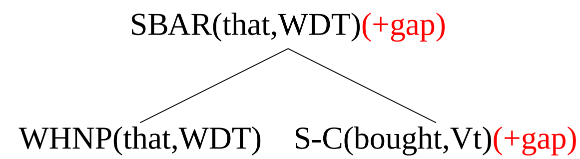
New Rules: Rules that Generate Gaps



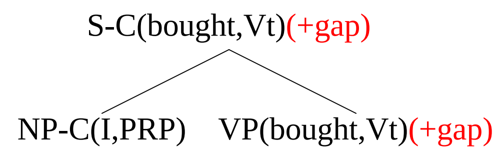
- Modeled in a very similar way to previous rules

New Rules: Rules that Pass Gaps down the Tree

- Passing a gap to a modifier

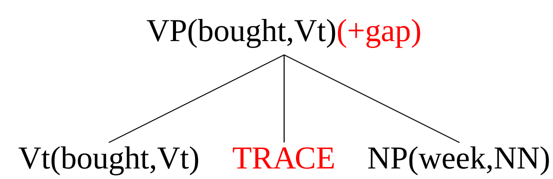


- Passing a gap to the head



New Rules: Rules that Discharge Gaps as a Trace

- Discharging a gap as a TRACE



Adding Gap Propagation (Example 1)

- Step 1: generate category of head child

SBAR(that,WDT)(+gap)

$\Downarrow$

SBAR(that,WDT)(+gap)
|
WHNP(that,WDT)

P_h(WHNP | SBAR, that, WDT)

Adding Gap Propagation (Example 1)

- Step 2: choose to propagate the gap to the head, or to the left or right of the head

SBAR(that,WDT)(+gap)
|
WHNP(that,WDT)

$\Downarrow$

SBAR(that,WDT)(+gap)
|
WHNP(that,WDT)

P_h(WHNP | SBAR, that, WDT) $\times$ P_g(RIGHT | SBAR, that, WDT)

- In this case left modifiers are generated as before

Adding Gap Propagation (Example 1)

- Step 3: choose right subcategorization frame

SBAR(that,WDT)(+gap)
|
WHNP(that,WDT)

$\Downarrow$

SBAR(that,WDT)(+gap)
|
WHNP(that,WDT)
{S-C,+gap}

P_h(WHNP | SBAR, that, WDT) $\times$ P_g(RIGHT | SBAR, that, WDT)$\times$
P_{rc}({S-C} | SBAR, WHNP, that, WDT)

Adding Gap Propagation (Example 1)

- Step 4: Generate right modifiers

SBAR(that,WDT)(+gap)

WHNP(that,WDT) ??
{S-C,+gap}

$\Downarrow$

SBAR(that,WDT)(+gap)

WHNP(that,WDT) S-C(bought,Vt)(+gap)
{}

P_h(WHNP | SBAR, that, WDT) $\times$ P_g(RIGHT | SBAR, that, WDT)$\times$
P_{rc}({S-C} | SBAR, WHNP, that, WDT)$\times$
P_d(S-C(bought,Vt)(+gap) | SBAR, WHNP, that, WDT, RIGHT, {S-C,+gap})

Adding Gap Propagation (Example 2)

- Step 1: generate category of head child

S-C(bought,Vt)(+gap)

$\Downarrow$

S-C(bought,Vt)(+gap)
|
VP(bought,Vt)

P_h(VP | S-C, bought, Vt)

Adding Gap Propagation (Example 2)

- Step 2: choose to propagate the gap to the head, or to the left or right of the head

---

S-C(bought,Vt)(+gap)
|
VP(bought,Vt)

$\Downarrow$

S-C(bought,Vt)(+gap)
|
VP(bought,Vt)(+gap)

P_h(VP | S-C, bought, Vt) $\times$ P_g(HEAD | S-C, VP, bought, Vt)

- In this case we're done: rest of rule is generated as before

Adding Gap Propagation (Example 3)

- Step 1: generate category of head child

VP(bought,Vt)(+gap)

$\Downarrow$

VP(bought,Vt)(+gap)
|
Vt(bought,Vt)

P_h(Vt | VP, bought, Vt)

Adding Gap Propagation (Example 3)

- Step 2: choose to propagate the gap to the head, or to the left or right of the head

---

VP(bought,Vt)(+gap)
|
VP(bought,Vt)

$\Downarrow$

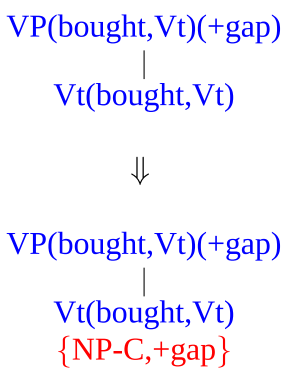
VP(bought,Vt)(+gap)
|
VP(bought,Vt)

P_h(Vt | SBAR, that, WDT) $\times$ P_g(RIGHT | VP, Vt, bought, Vt)

- In this case left modifiers are generated as before

Adding Gap Propagation (Example 3)

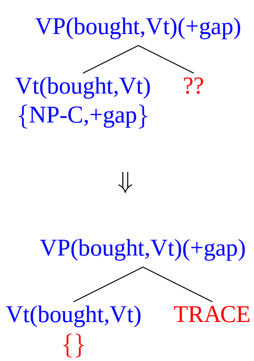
- Step 3: choose right subcategorization frame



P_h(Vt | SBAR, that, WDT) × P_g(RIGHT | VP, Vt, bought, Vt)×
P_{rc}({NP-C} | VP, Vt, bought, Vt)

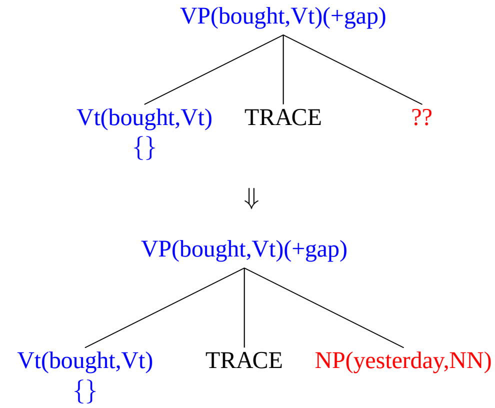
Adding Gap Propagation (Example 3)

- Step 4: generate right modifiers



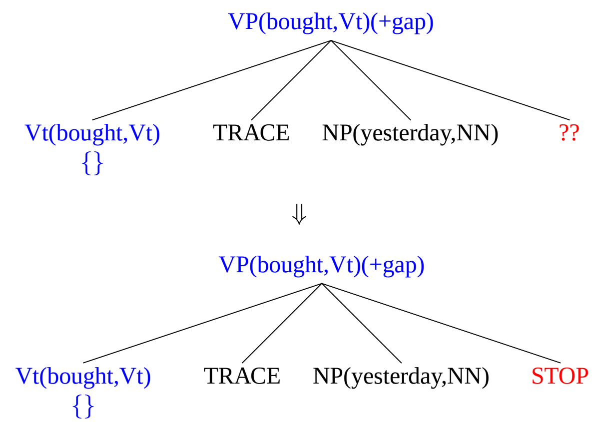
P_h(Vt | SBAR, that, WDT) × P_g(RIGHT | VP, Vt, bought, Vt)×
P_{rc}({NP-C} | VP, Vt, bought, Vt)×
P_d(TRACE | VP, Vt, bought, Vt, RIGHT, {NP-C,+gap})

Adding Gap Propagation (Example 3)



P_h(Vt | SBAR, that, WDT) × P_g(RIGHT | VP, Vt, bought, Vt)×
P_{rc}({NP-C} | VP, Vt, bought, Vt)×
P_d(TRACE | VP, Vt, bought, Vt, RIGHT, {NP-C,+gap})×
P_d(NP(yesterday,NN) | VP, Vt, bought, Vt, RIGHT, {})

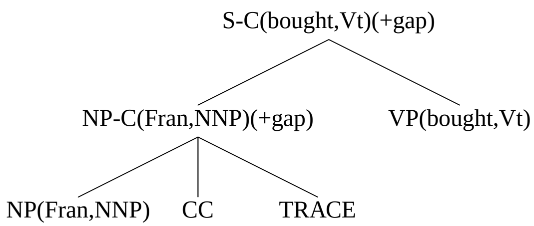
Adding Gap Propagation (Example 3)



P_h(Vt | SBAR, that, WDT) × P_g(RIGHT | VP, Vt, bought, Vt)×
P_{rc}({NP-C} | VP, Vt, bought, Vt)×
P_d(TRACE | VP, Vt, bought, Vt, RIGHT, {NP-C,+gap})×
P_d(NP(yesterday,NN) | VP, Vt, bought, Vt, RIGHT, {})×
P_d(STOP | VP, Vt, bought, Vt, RIGHT, {})

Ungrammatical Cases Contain Low Probability Rules

Example 1 The person (SBAR who Fran and TRACE bought the shoes)



Example 2 The store (SBAR that Jeff bought the shoes because Fran likes TRACE)