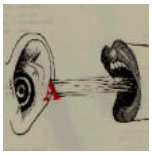
Lecture 12:
PCFG parsing,
Treebank Parsing: as good as gold?

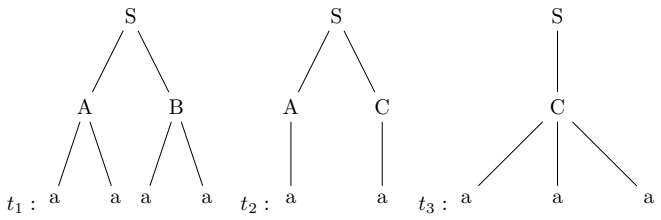


Professor Robert C. Berwick
berwick@csail.mit.edu

Menu

- Statistical Parsing with Treebanks II:
  - Some details on lexicalization: heads, generation model, and parsing
  - How to find features automatically for parsing
  - The state of the state of the art: from features to discriminative parsing
- Statistical Parsing with Treebanks II: do these systems *really* acquire 'knowledge of language'?

Why independence is a bad idea



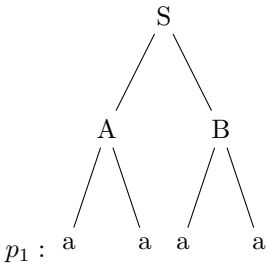
10x 20x 50x

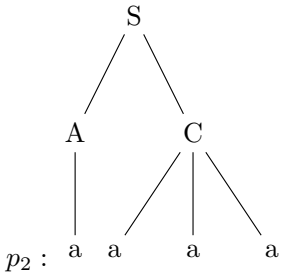
$$\frac{10}{10+20+50} = 0.125 \quad S \rightarrow A\ B$$
$$\frac{20}{10+20+50} = 0.25 \quad S \rightarrow A\ C$$
$$\frac{50}{10+20+50} = 0.625 \quad S \rightarrow C$$
$$\frac{10}{10+20} = 0.334 \quad A \rightarrow a\ a$$
$$\frac{20}{10+20} = 0.667 \quad A \rightarrow a$$
$$\frac{20}{20+50} = 0.285 \quad B \rightarrow a\ a$$
$$\frac{50}{20+50} = 0.714 \quad C \rightarrow a\ a\ a$$

$$\frac{10}{10+20+50} = 0.125 \quad S \rightarrow A\ B$$
$$\frac{20}{10+20+50} = 0.25 \quad S \rightarrow A\ C$$
$$\frac{50}{10+20+50} = 0.625 \quad S \rightarrow C$$
$$\frac{10}{10+20} = 0.334 \quad A \rightarrow a\ a$$
$$\frac{20}{10+20} = 0.667 \quad A \rightarrow a$$
$$\frac{20}{20+50} = 0.285 \quad B \rightarrow a\ a$$
$$\frac{50}{20+50} = 0.714 \quad C \rightarrow a\ a\ a$$

Parse of *a a a a*

And the other one?

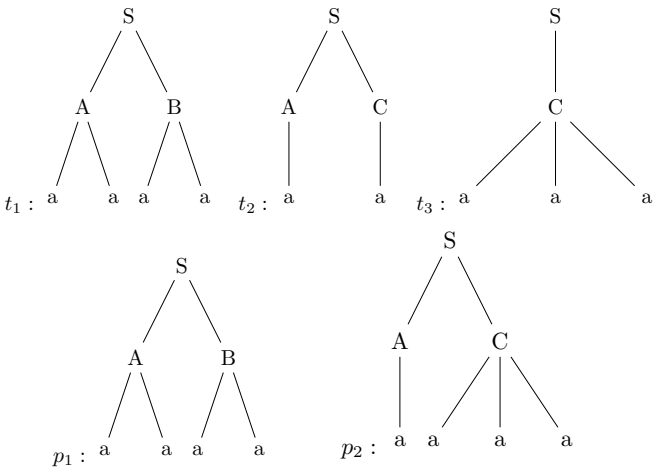




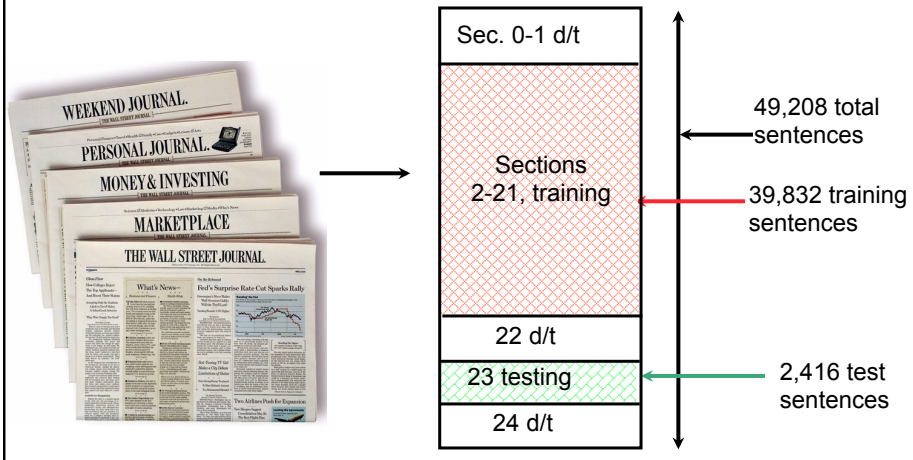
$p_1 = 0.125 \cdot 0.334 \cdot 0.285 = 0.01189$

$p_2 = 0.25 \cdot 0.667 \cdot 0.714 = 0.119$

What's the problem?



The "primary linguistic data": input is the Penn Treebank



PTB is a (particular) *linguistic theory* not a plain "corpus":
it assumes *some* assignment of (linguistic) structure to sentences
Question: What Knowledge of Language does/can such systems acquire?

What are 'heads' of phrases?

- Head of XP is 'X'

S →	NP VP	(VP is the head – nonstandard!)
VP →	Vt NP	(Vt is the head)
NP →	DT NN NN	(rightmost NN is head)

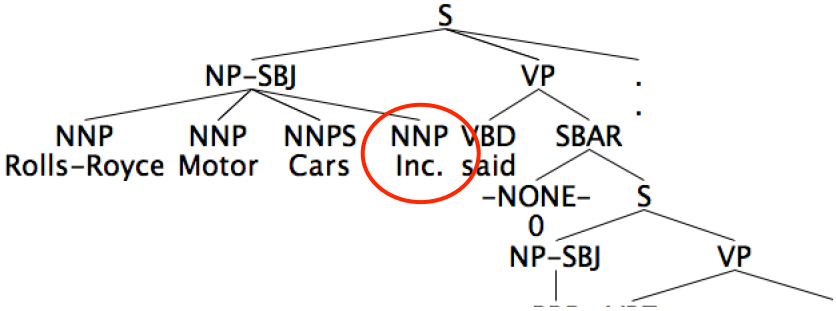
- Lexicalized parsers use deterministic rules to 'find' heads (not always in a linguistically justified way)
- The phrase receives its head annotation from the head 'child' below

Example head-finding rules

- For NP expansions:
  - **If** the right-hand side (rhs) of the rule contains NN, NNS, or NNP **then** select rightmost NN, NNS, or NNP
  - **Else if** the rhs contains an NP **then** select the leftmost **NP**
  - **Else if** the rhs contains a JJ **then** select the rightmost JJ
  - **Else if** the rhs contains a CD **then** select the rightmost CD
  - **Else** select the rightmost child

Example for NP

NP → DT NNP NN NP
NP → DT NN NN NNP NNP
NP → NP PP
NP → DT JJ
NP → DT



Lexicalized Chomsky normal form grammar definition

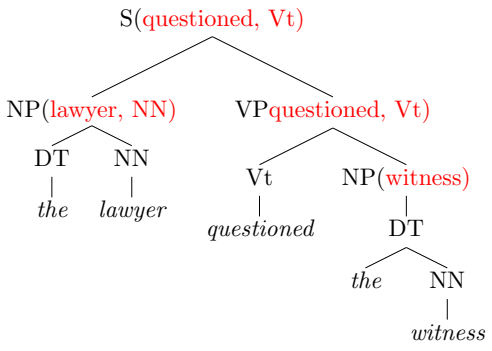
G is a 4-tuple: $(N, T, R, Start)$

- N is a finite set of non-terminal symbols
- T is a finite set of terminal symbols (words)
- R is a finite set of rule that are in one of these forms:
  - $X(h) \rightarrow Y_1(h)Y_2(w)$, for $X \in N$, and $Y_1, Y2 \in N$, and $h, w \in T$
  - $X(h) \rightarrow Y_1(w)Y_2(h)$, for $X \in N$, and $Y_1, Y2 \in N$, and $h, w \in T$
  - $X(h) \rightarrow h$, for $X \in N$, and $h \in T$
  - $X(w) \rightarrow w$, for $X \in N$, and $w \in T$
- $Start \in N$ is a distinguished start symbol

What does this do to grammar size?

- The grammar looks like Chomsky normal form, but it has potentially $O(|T|^2 \times |N|^3)$ rules
- So if we parse an n word sentence using the PCFG algorithm it might take $O(|n^3|T|^2|N|^3)$ time, and $|T|$ is huge (20K = 40K?)
- BUT in any one sentence $w_1 \ldots w_n$ of length n, at most $O(n^2 \times |N|^3)$ rules can be applicable, because any rules that contain a lexical item that is not in the sentence can be dismissed
- Parsing time is $(O(n^5|N|^3)$.

Propagating both heads and part of speech tags

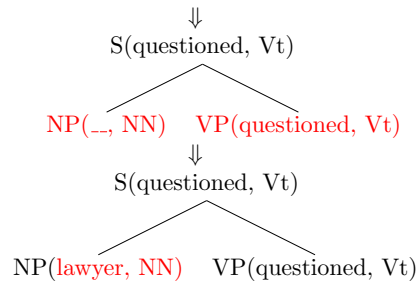


Two lexicalized parsing models with heads

- Charniak, 1997: head word annotation, P/R/F1 up to about 86.7%
- Collins, 1997: add POS tags, argument vs. adjuncts, subcategories for verbs, distance from head: 88.1%

Charniak generative model: decompose step by step (3 steps)

Rule: S → NP(NN) VP(Vt)
S(questioned, Vt)



Prob(NP(NN) VP(Vt) |S(questioned, Vt)

Prob(lawyer |S(questioned, Vt, VP, NP(NN)

Problem: note that the very first step requires an estimate based on counts of an entire rule
Such counts are *sparse*:
of 39, 400 training sentences in PTB, there are only 12, 409 rules
15% of all test data sentences have rule never seen

Charniak's answer: smoothed estimate, steps 1 & 2

$$p(\text{NP(NN) VP(Vt)}|\ \text{S(questioned, Vt)}) =$$

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

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

$$0 \leq \lambda_1, \lambda_2 \leq 1; \lambda_1 + \lambda_2 = 1$$

Interpolation between word and tag

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

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

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

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

$$0 \leq \lambda_3, \lambda_4, \lambda_5 \leq 1; \lambda_3 + \lambda_4 + \lambda_5 = 1$$

And one more smoothed estimate for last step, step 3

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

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

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

$$+\left(\lambda_3 \times \frac{Count(\text{lawyer}|\text{S(questioned, Vt), VP, NP(NN))})}{Count(\text{S(questioned, Vt), VP, NP(NN))})}\right.$$

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

$$\left.+\lambda_5 \times \frac{Count(\text{lawyer}|\text{NN})}{Count(\text{NN})}\right)$$

Many rules only occur a few times…

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

Refining the node expansion possibilities

- Charniak (1997) expands each phrase structure tree in a single step
- This works well to capture dependencies between children nodes
- But bad because of sparseness
- A pure dependency, one child at a time model is worse
- You can find the 'Goldilocks spot' by various 'in between' models, eg, generating children as a Markov process on both sides of the head

Using Markov chain in tree

- Step 1: generate Head of phrase
- Step 2: generate left modifiers as a Markov chain (until STOP symbol happens to be generated)
- Step 3: generate right modifiers as a Markov chain (until STOP symbol happens to be generated)

(For these, add in distance effect, argument/adjunct, and subcatgorization information)

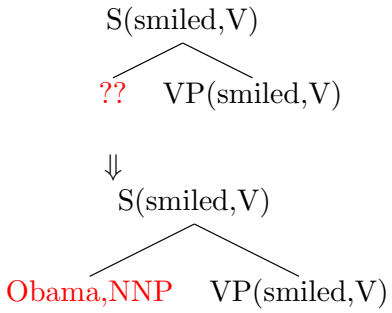
Using Markov processes in tree

- Another method: model rule productions as Markov processes
- Step 1: generate category of head child

$$\mathrm{S(smiled,\ V)} \Downarrow \mathrm{S(smiled,\ V)}$$

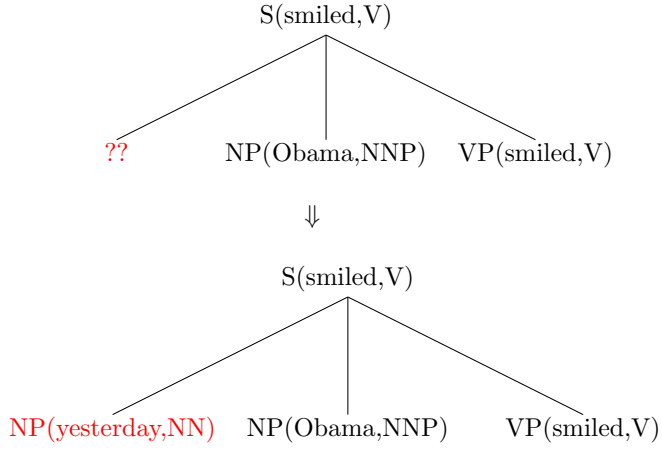
$$\text{Estimated as: } p_h(\mathrm{VP}|\mathrm{S}, \mathrm{smiled}, \mathrm{V}) \qquad | \\ \mathrm{VP(smiled,\ V)}$$

Step 2: Generate left modifiers in a Markov chain



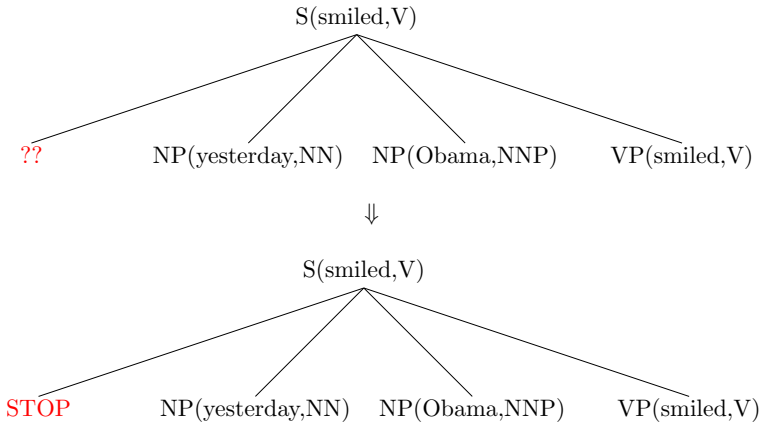
$p_h(\text{VP}|\text{S}, \text{smiled}, \text{V}) \times p_d(\text{NP(Obama, NNP)}|\text{S}, \text{VP}, \text{smiled}, \text{V}, \text{LEFT})$

Step 2: generate left modifiers



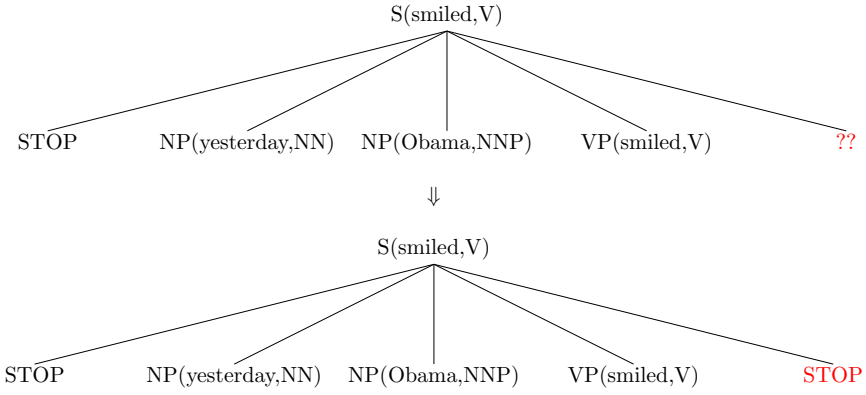
$p_h(\text{VP}|\text{S}, \text{smiled}, \text{V}) \times p_d(\text{NP}(\text{Obama}, \text{NNP})|\text{S}, \text{VP}, \text{smiled}, \text{V}, \text{LEFT}) \times$
$p_d(\text{NP(yesterday, NN)}|\text{S}, \text{VP}, \text{smiled}, \text{LEFT})$

Left mods generated until STOP is generated



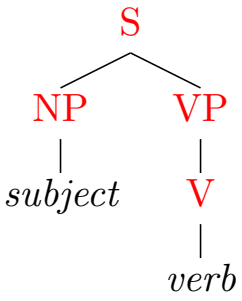
p_h(VP | S,smiled,V)×p_d(NP(Obama,NNP) | S,VP,smiled,V,LEFT)×
p_d(NP(yesterday,NN) | S,VP,smiled,LEFT) ×p_d (STOP | S,VP,smiled,V,LEFT)

Now generate right side the same way, from head to the right…

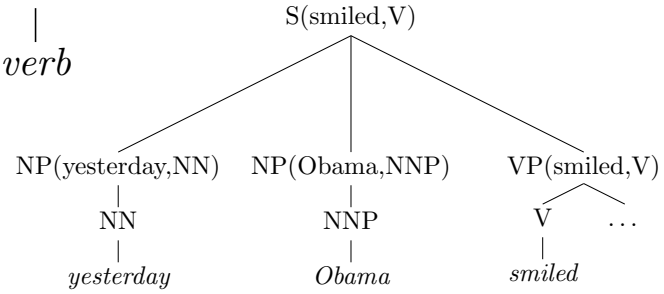


p_h(VP | S,smiled,V)×p_d(NP(Obama,NNP) | S,VP,smiled,V,LEFT)×
p_d(NP(yesterday,NN) | S,VP,smiled,LEFT)×p_d (STOP | S,VP,told,V,LEFT)×
p_d(STOP | S, VP, smiled, RIGHT)

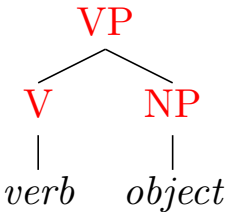
Adding an argument/adjunct feature

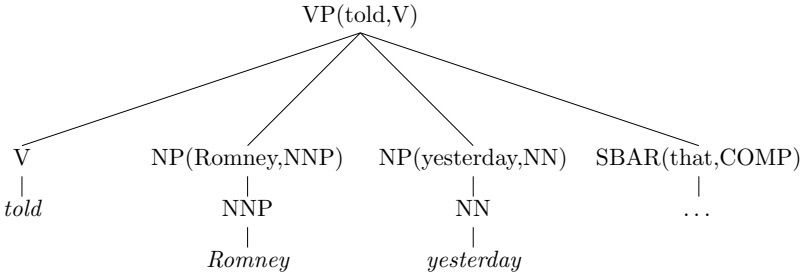


Q: what distinguishes *yesterday* from *Obama*?



Arguments vs. adjuncts





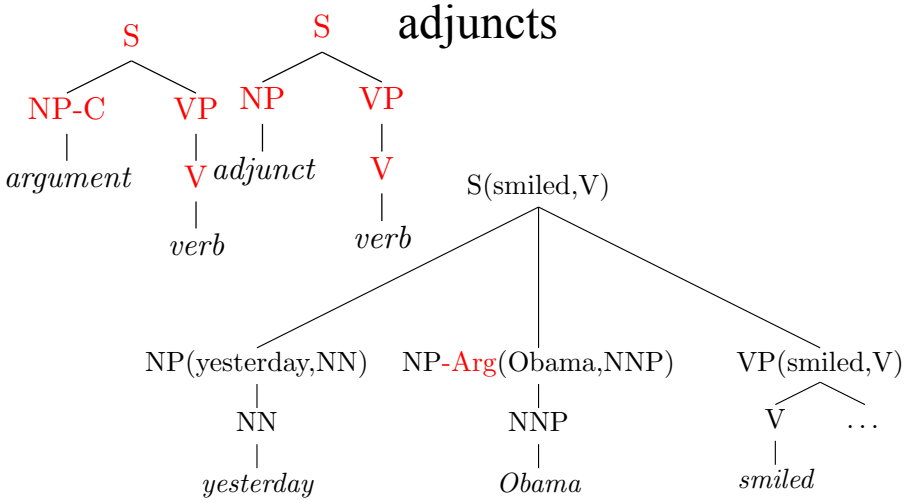
Arguments & Adjuncts

- Complements: to a first order approx, the required 'arguments' to a predicate
- Adjuncts *add* information, but aren't necessary
- Contrast: *I gave* vs. *I gave at the office*

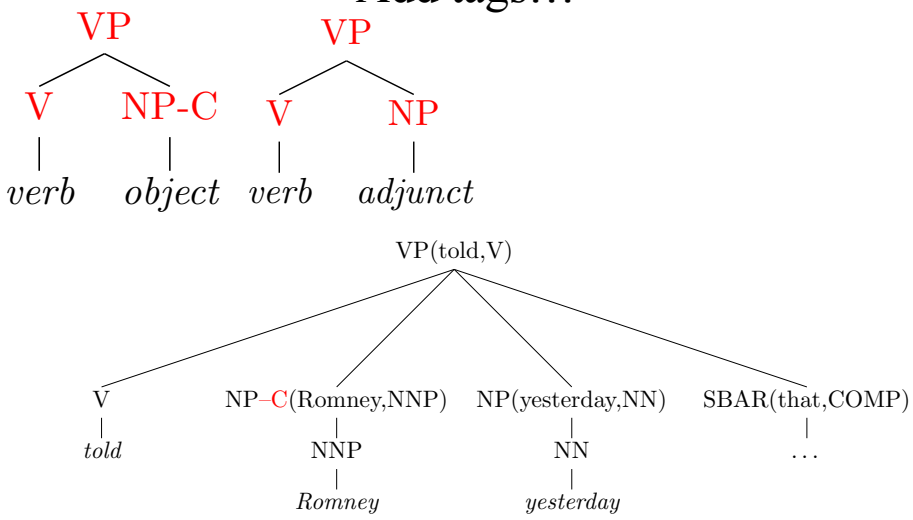
Subcategorization frames

- Given the 'who did what to whom' even structure
- Syntactic reflex of the predicate-argument relations - lexical semantics
- You've already seen them in primitive form as Verbs of 0, 1, 2 arguments
- But other details: some verbs take, e.g., a proposition as an argument (*I think that*....)
- So we add a probability model for this also

Add tags to distinguish arguments & adjuncts



Add tags…



Add probabilistic selection of a particular subcategorization frame (#/type args to verb)

S(told, V)

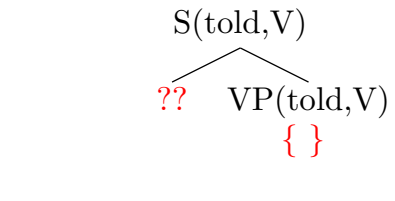
Step 2: generate head

$\Downarrow$

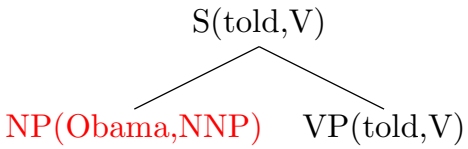
S(told, V)
|
VP(told, V)
NP-C

Was: $p_h(\text{VP}|\text{S}, \text{told}, \text{V})$
Now: $p_h(\text{VP}|\text{S}, \text{told}, \text{V}) \times p_{lc}(\text{ NP-C } | \text{ S,VP,told,V})$

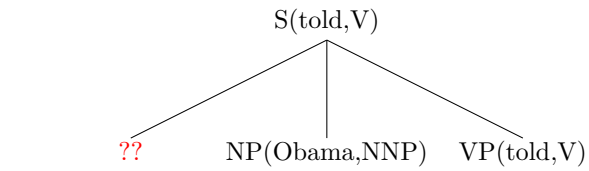
Generate left modifiers as Markov chain

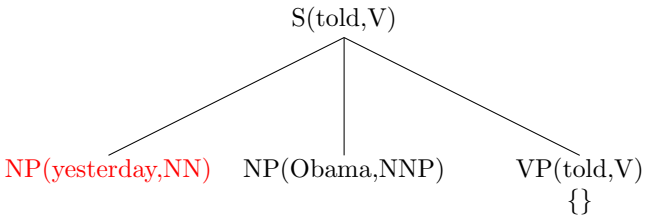


$\Downarrow$

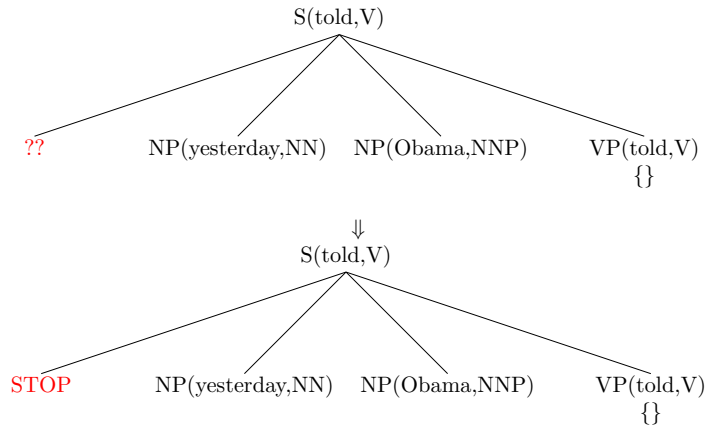


$p_h(\text{VP}|\text{S}, \text{told}, \text{V}) \times p_{lc}(\text{NP-C} \mid \text{S,VP,told,V}) \times$
$p_d(\text{NP(Obama, NNP)}|\text{S}, \text{VP}, \text{told}, \text{V}, \text{LEFT}, \{\text{NP-C}\})$





$p_h(\text{VP}|\text{S}, \text{told}, \text{V}) \times p_{lc}(\{\text{NP-C}\} \mid \text{S}, \text{VP}, \text{told}, \text{V}) \times$
$p_d(\text{NP(Obama,NNP)} \mid \text{S,VP,told,V,LEFT,}\{\text{NP-C}\}) \times$
$p_d(\text{NP(yesterday, NN)}|\text{S}, \text{VP}, \text{told}, \text{LEFT}\{\})$



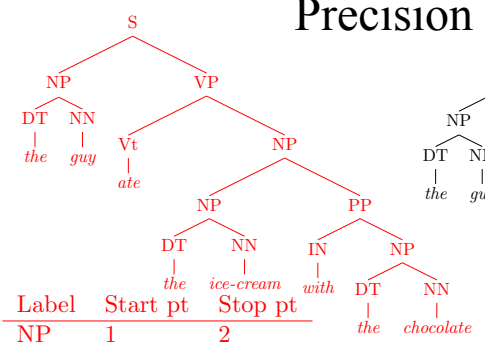
$p_h(\text{VP}|\text{S}, \text{told}, \text{V}) \times p_{lc}(\{\text{NP-C}\} \mid \text{S}, \text{VP}, \text{told}, \text{V}) \times$
$p_d(\text{NP(Obama,NNP)} \mid \text{S,VP,told,V,LEFT,}\{\text{NP-C}\}) \times$
$p_d(\text{NP(yesterday,NN)} \mid \text{S,VP,told,LEFT,}\{\}) \times$
$p_d(\text{STOP} \mid \text{S,VP,told,V,LEFT,}\{\})$

So far then

- Find heads of the rules to capture dependencies
- Break generation of parse tree (rule applications) into markov process steps
- Build dependencies back in through subcategorization, node annotation

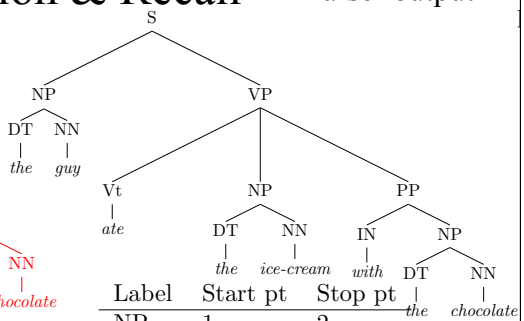
Precision & Recall

Gold standard



Label	Start pt	Stop pt
NP	1	2
NP	4	5
NP	4	8
PP	6	8
NP	7	8
VP	3	8
S	1	8

Parser output



Label	Start pt	Stop pt
NP	1	2
NP	4	5
PP	6	8
NP	7	8
VP	3	8
S	1	8

G=# constituents in gold standard = 7
P= # in parse output = 6
C= # correct = 6
Recall = *C*/*G* x 100 = 6/7 x 100=85.7%
Precision = *C*/*P* = 6/6 x 100 = 1--%

Basic Results

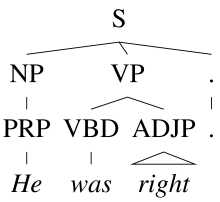
Method	Recall %	Precision %
PCFGs	70.6	74.8
Decision Trees	84.0	84.3
Lexicalization	85.3	85.7
Conditional, max entropy	86.3	87.5
Generative lexical, Charniak	86.7	86.6
Model 1 Collins generative lexical	87.5	87.7
Model 2 Collins w/ subcat	88.1	88.3
Stanford	89.1	88.9
Adaptive cats, Berkeley	91.2	90.4

How much does 'perfection' require?

Automatic Annotation Induction?

[Matsuzaki et. al '05, Prescher '05]

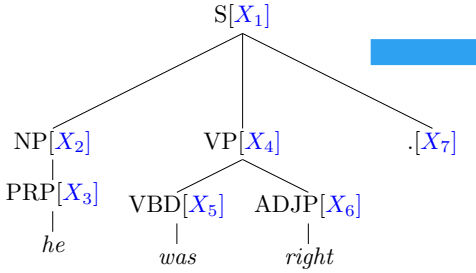
- Advantages:
  - Automatically learned:
    Label *all* nodes with latent variables.
    Same number k of subcategories for all categories.
- Disadvantages:
  - Grammar gets too large
  - Most categories are oversplit while others are undersplit.

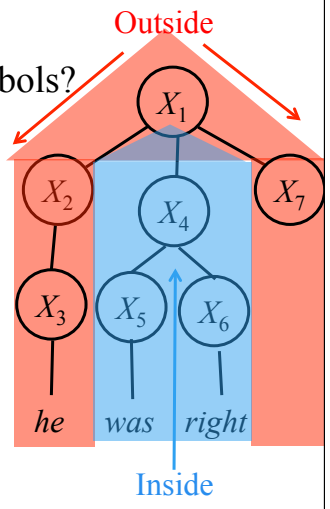


Model	*F1*
Klein & Manning '03	86.3
Matsuzaki et al. '05	86.7

Learning Latent Annotations (Petrov & Klein, 2006)

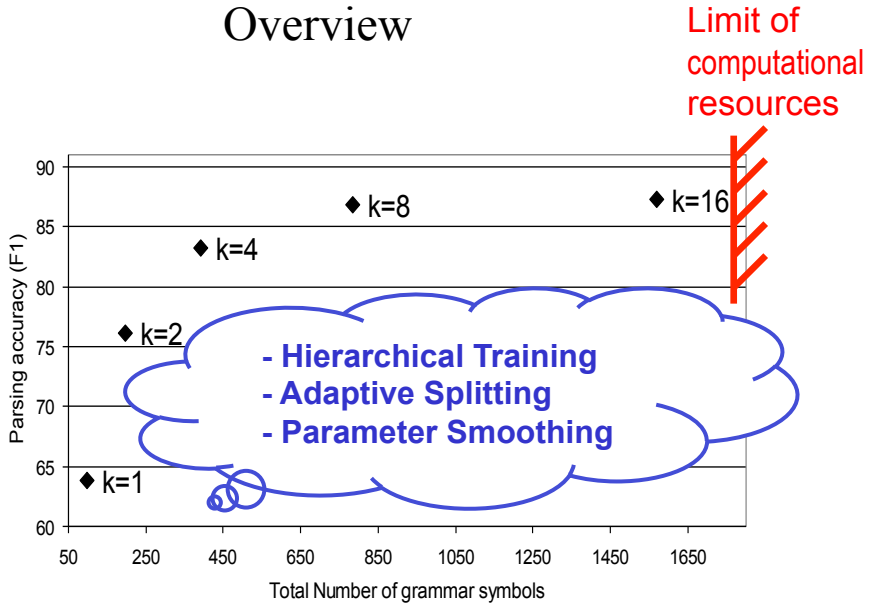
- Can you automatically find good symbols?
  - Brackets are known
  - Base categories are known
  - Induce subcategories
  - Split/merge refinement



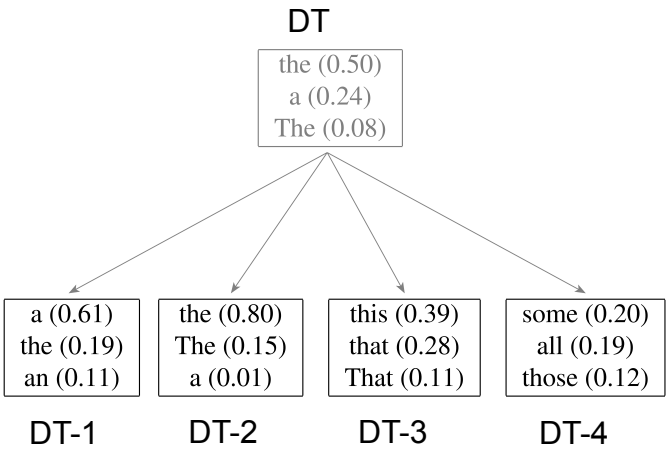


Uses EM for trees, as sketched before

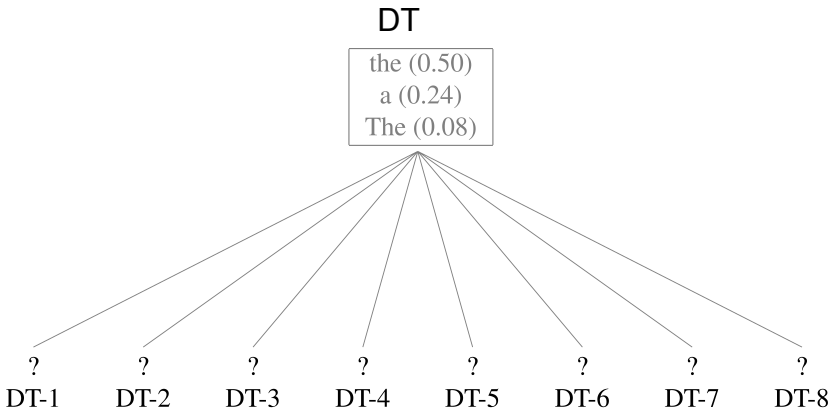
Overview



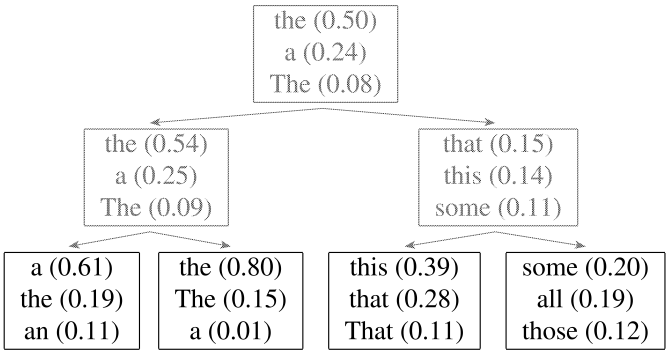
Refinement of the DT tag (like 6.034 decision trees)



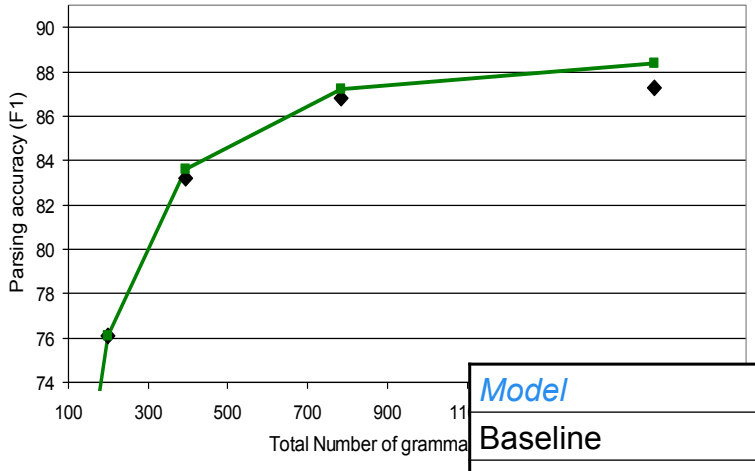
Too many categories? How do we know?



Hierarchical refinement of the DT tag



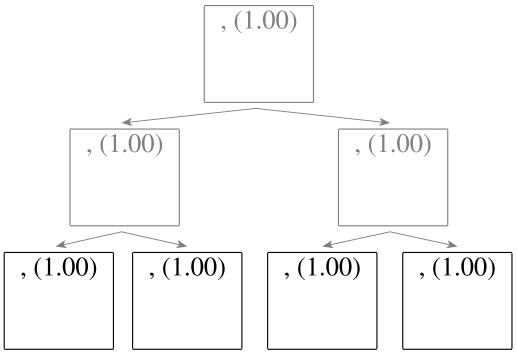
Hierarchical Estimation Results



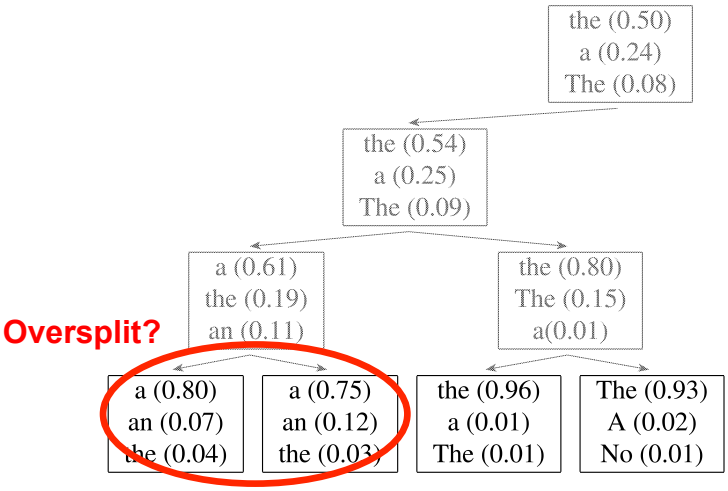
Model	*F1*
Baseline	87.3
Hierarchical Training	88.4

Refinement of the , tag

- Splitting all categories the same amount is wasteful:

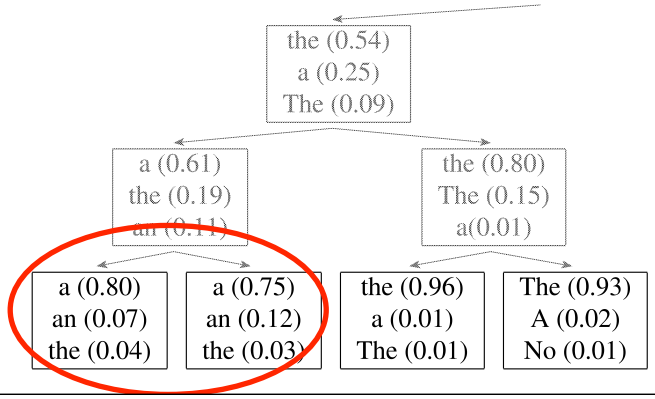


The DT tag revisited



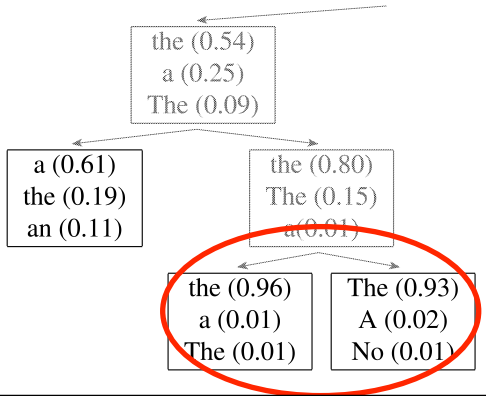
Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful



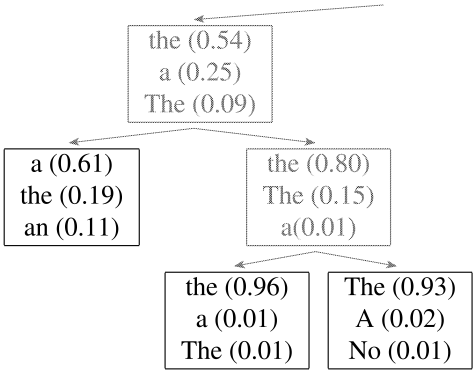
Adaptive splitting: Goldilocks principle

- If we want to split complex categories more
- Split everything and roll back splits that were least useful



Adaptive splitting – Goldilocks principle

- If we want to split complex categories more
- Split everything and roll back splits that were least useful

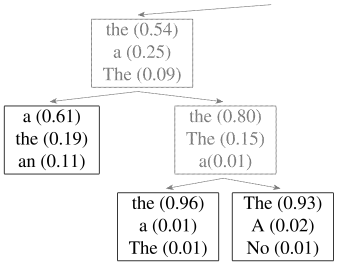


Adaptive Splitting – ah, just right

- Evaluate loss in likelihood from removing each split =

$$\frac{\textit{Data likelihood with split reversed}}{\textit{Data likelihood with split}}$$

- No loss in accuracy when 50% of the splits are reversed.

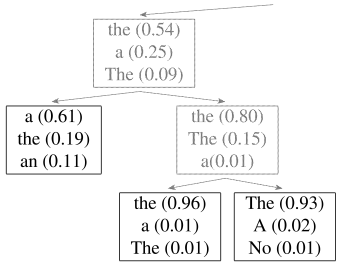


Adaptive Splitting

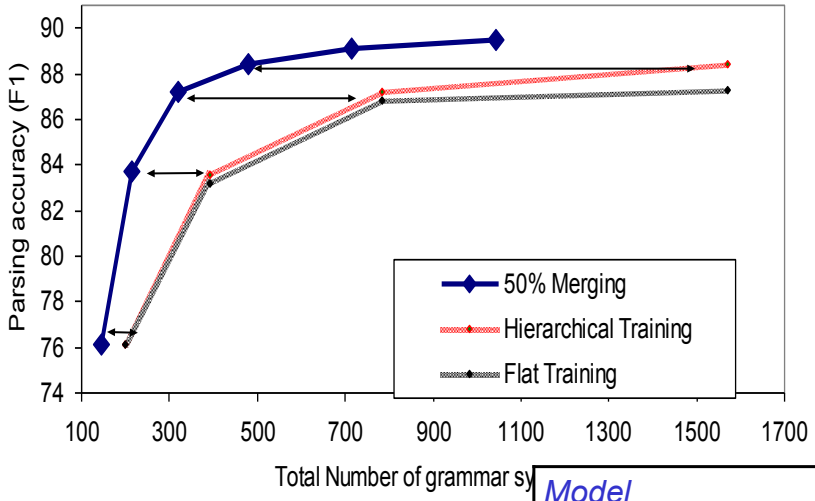
- Evaluate loss in likelihood from removing each split =

$$\frac{\textit{Data likelihood with split reversed}}{\textit{Data likelihood with split}}$$

- No loss in accuracy when 50% of the splits are reversed.

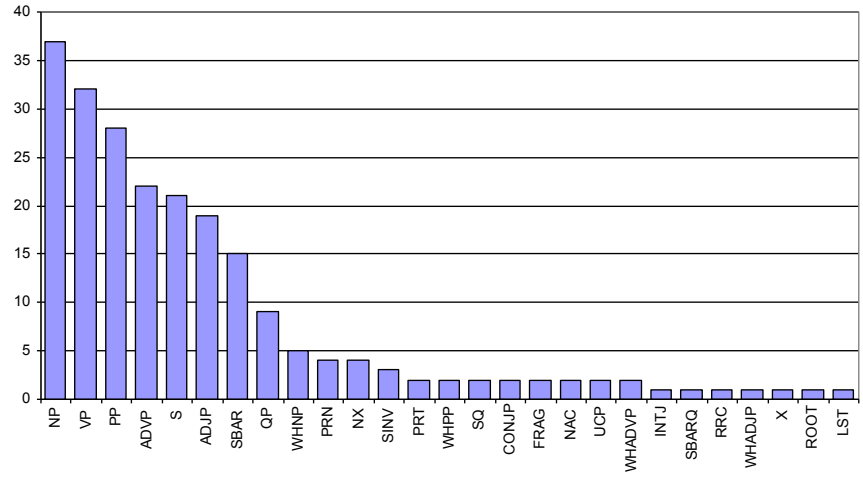


Adaptive Splitting Results

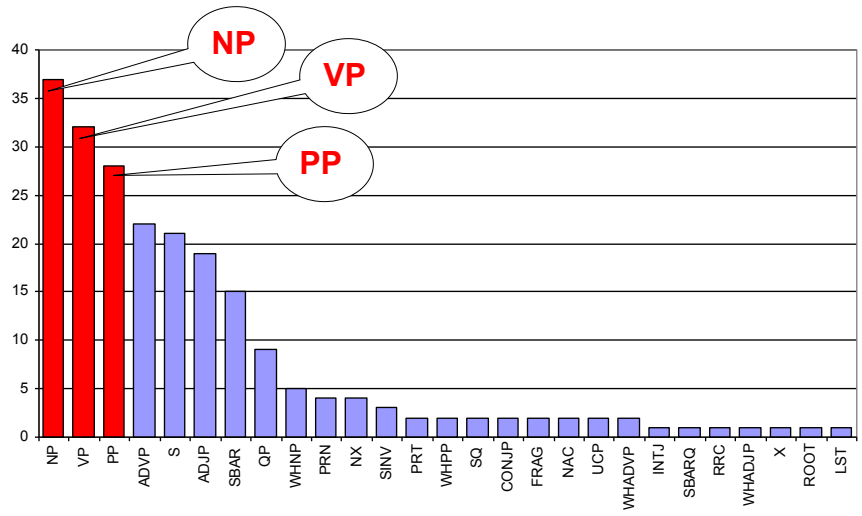


Model	*F1*
Previous	88.4
With 50% Merging	89.5

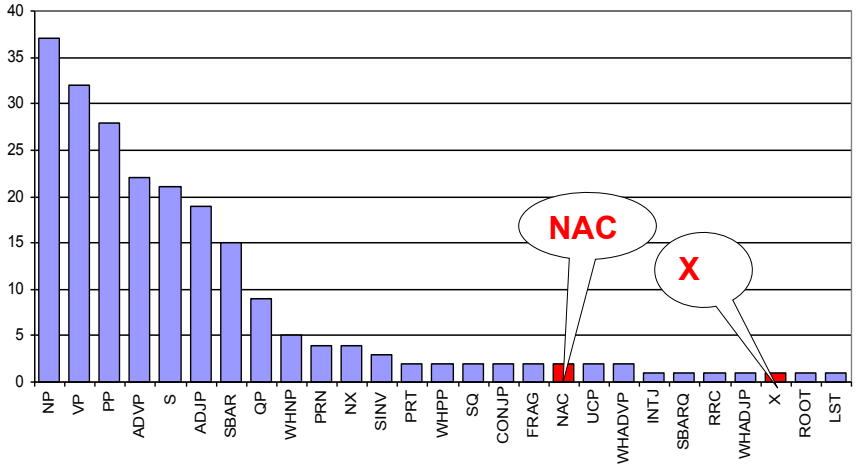
Number of phrasal categories

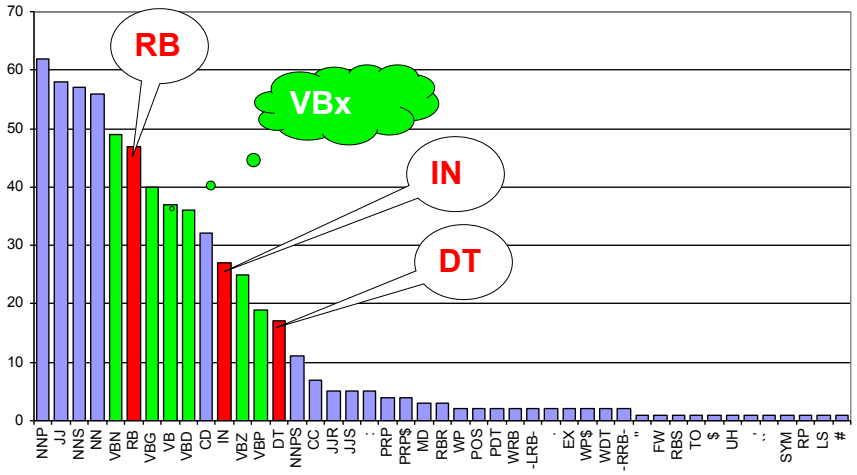


Number of phrasal subcategories

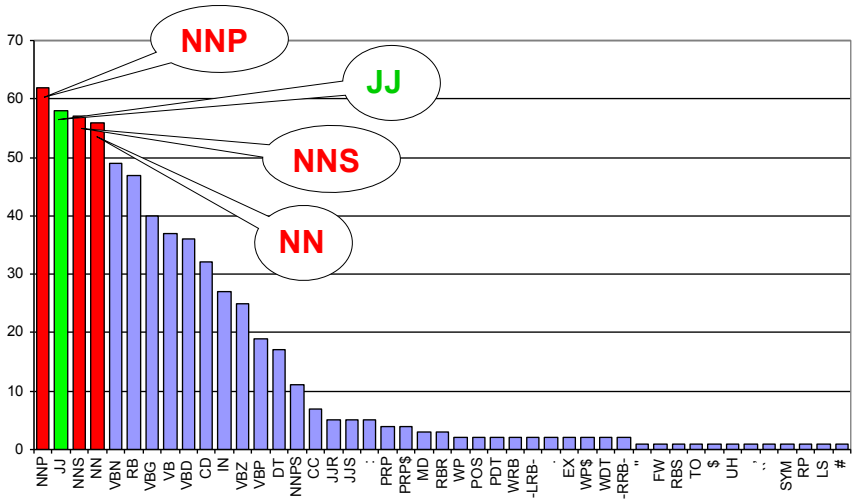


Number of Phrasal Subcategories



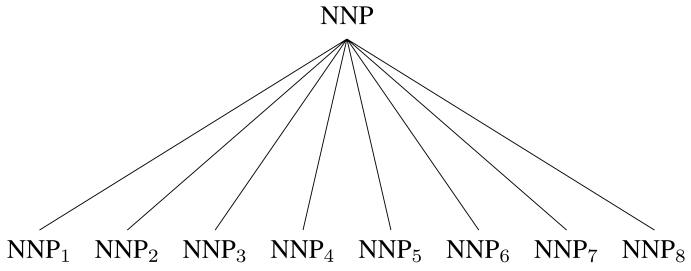


Number of Lexical Subcategories



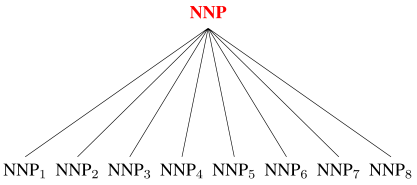
Smoothing

- Heavy splitting can lead to overfitting
- Smoothing can fix this.



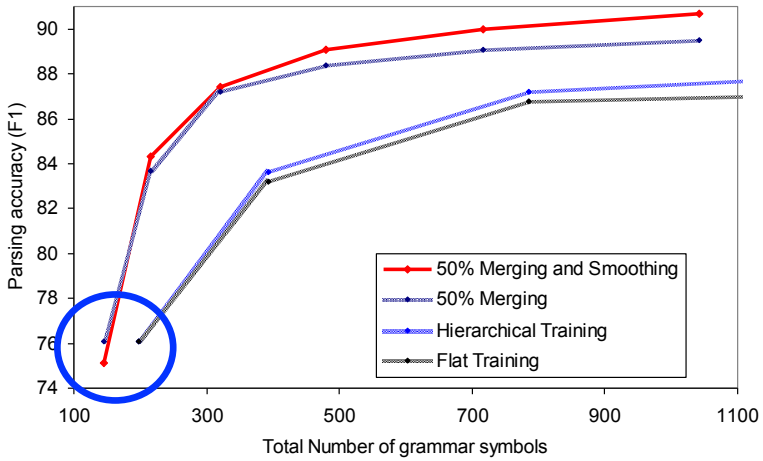
Linear smoothing over rules

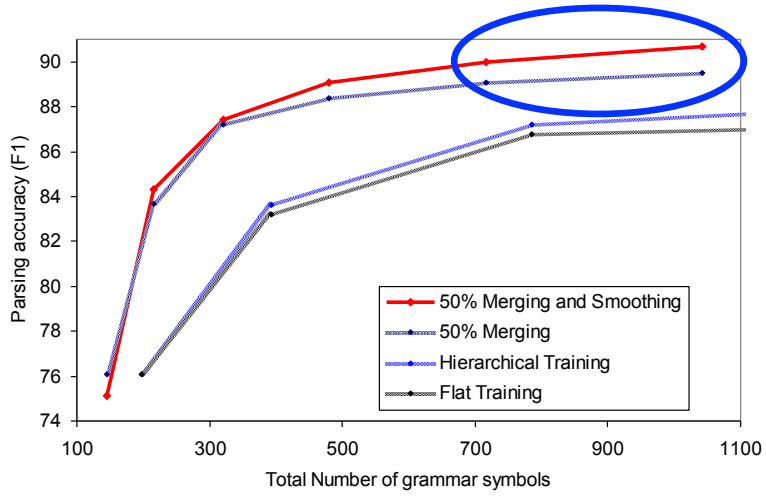
$$p_x = \mathrm{P}(A_x \rightarrow BC)$$

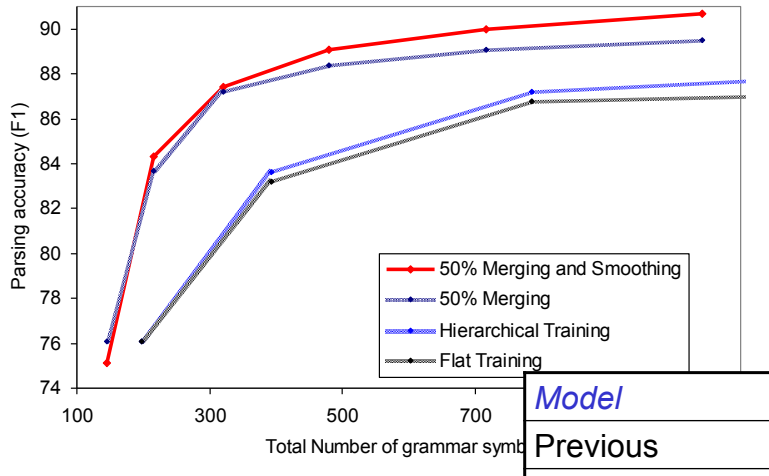


$$p'_x = (1-\alpha)p_x + \alpha\bar{p}$$

where $\bar{p} = \frac{1}{n}\sum_x p_x$







Model	*F1*
Previous	89.5
With Smoothing	90.7

The bottom line

Parser	*F1 ≤ 40 words*	*F1 all words*
Klein & Manning '03	86.3	85.7
Matsuzaki et al. '05	86.7	86.1
Collins '99	88.6	88.2
Charniak & Johnson '05	90.1	89.6
Auto learning cats	**90.2**	**89.7**

Do the categories make sense?

- Proper Nouns (NNP):

NNP-14	Oct.	Nov.	Sept.
NNP-12	John	Robert	James
NNP-2	J.	E.	L.
NNP-1	Bush	Noriega	Peters
NNP-15	New	San	Wall
NNP-3	York	Francisco	Street

- Proper Names (PRP)

PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

Linguistic Candy

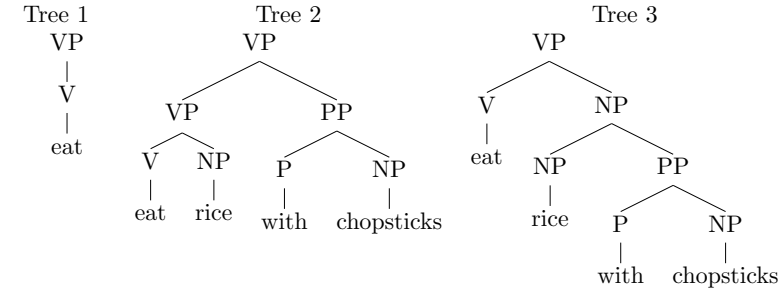
- Relative adverbs (RBR):

RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

- Cardinal Numbers (CD):

CD-7	one	two	Three
CD-4	1989	1990	1988
CD-11	million	billion	trillion
CD-0	1	50	100
CD-3	1	30	31
CD-9	78	58	34

The dangers of eating with chopsticks

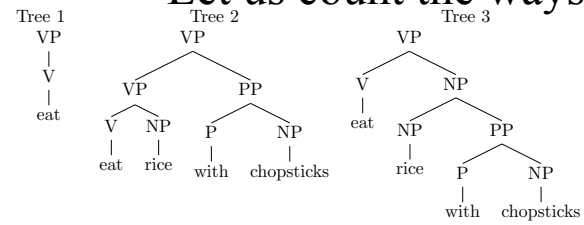


108 times 6 times 2 times

Now let's calculate the odds… what will the PCFG trained on this data return for *eat rice with chopsticks*?

Tree #2? No! It will return Tree #3. Why?

Let us count the ways…



108 times 6 times 2 times

VP→V count: 108
VP→V NP count: 6 + 2 = 8
VP→VP PP count: 6
Total VP→ =122

p(VP→VP PP) = 6/122 = 0.0492

NP→rice/chopsticks count: 12+4=16
NP→V NP count: 2
Total NP→ =18

p(NP→NP PP) = 2/9 = 0.111… > p(VP→VP PP) Oops!

What to do?

- Be more *discriminating*

The latest bake-off

System	F	P	R	Exact	Speed
ENHANCED TRAINING / SYSTEMS					
Charniak-SR	92.07	92.44	91.70	44.87	1.8
Charniak-R	91.41	91.78	91.04	44.04	1.8
Charniak-S	91.02	91.16	90.89	40.77	1.8
STANDARD PARSERS					
Berkeley	90.06	90.30	89.81	36.59	4.2
Charniak	89.71	89.88	89.55	37.25	1.8
SSN	89.42	89.96	88.89	32.74	1.8
BUBS	88.50	88.57	88.43	31.62	27.6
Bikel	88.16	88.23	88.10	32.33	0.8
Collins-3	87.66	87.82	87.50	32.22	2.0
Collins-2	87.62	87.77	87.48	32.51	2.2
Collins-1	87.09	87.29	86.90	30.35	3.3
Stanford-L	86.42	86.35	86.49	27.65	0.7
Stanford-U	85.78	86.48	85.09	28.35	2.7

To what degree is syntactic analysis a solved problem?

- PTB F1: 0.84 Magerman (1995)→ 0.92 Charniak (2006)
- But: single aggregate score misleading (sentence accuracy ~10–25%)
- Nonlocal dependencies, e.g., *What did you buy* - zilch
- How well do these systems actually work in recovering "who did what to whom"?

A more thorough job (Bender et al, 2011)

- Select ten 'hard' syntactic phenomena, local and non-local
- Find 100 'suitable' sentences per phenomenon in Wikipedia;
- Dual-annotate and reconcile for 'relevant' dependencies
- Run seven off-the-shelf parsers on this data (the strings)
- Design parser-specific patterns for automated evaluation

Example syntactic patterns

1. Nonlocal 'bare relatives':
   A classic example Schumacher provides ___ is that of education
2. Nonlocal 'tough' adjectives:
   Original copies are very hard to find ___
3. Right-node raising:
   He also played for ___ and managed Kilmarnock
4. 'It' expletives (not arguments):
   Crew negligence is blamed, and it is suggested that the flight crew were drunk
5. Absolutives (local):
   the format consisted of 12 games, each team facing the other teams twice
6. 'Controlled' arguments: *Alfred ... Continued __ to paint full time.*

Summary of phenomena types

1. barerel: bare relative clauses
2. tough: tough phrases
3. rnr: right-node raising
4. itexpl: "It" expletives
5. vpart: verb-particles ("threw out")
6. ned: noun with *ed* modified by another noun
7. absol: NP followed by non-finite predicate
8. vger: verbal gerund (in NP position) – "*accessing the website…*"
9. argadj: displaced argument – "*the story shows through flashbacks…*
10. control

The test examples

- Select from English Wikipedia (Wikiwoods)
- 900 million tokens
- Random selection of candidates
- Dual-vetted: skip too simple, too complex, false positives
- Result: 1000 sentences covering the 10 phenomena

Is this fair play?

- Are these phenomena represented in the PTB?
- Bare relatives, verb-particles, absolutives directly represented
- 'tough' construction reliably annotated (missing argument linked to *wh* head)
- Right-node raising (rnr) & vger (gerundive verbs), explicit, though a few false + for rnr; for vger, pos tag is sometimes nominal (instead of verb)

Fair Play?

- For remaining 4, there are some issues:
  1. Control: ok, but includes others (*ate the meat raw*)
  2. Itexpl: annotation –NONE- *EXP* is ok, but omits some
  3. Argadj: interleave args/adjuncts – PTB doesn't distinguish between post-verbal modifiers & verbal arguments; eg, PP-loc, but not consistently appied
  4. Ned: not mentioned; e.g., *gritty-eyed* left as JJ

47 Million Wikipedia sentences, 900M tokens

Phenomenon	Frequency	Candidates
barerel	2.12%	546
tough	0.07%	175
rnr	0.69%	1263
itexpl	0.13%	402
vpart	4.07%	765
ned	1.18%	349
absol	0.51%	963
vger	5.16%	679
argadj	3.60%	1346
control	3.78%	124

Annotation

- Specify target scheme; parallel annotation by two expert linguists
- Initial agreement: 79% (full sentences); all mismatches reconciled

Example annotation

The Act having been passed in that year, Jessop withdrew, and Whitworth carried on with the assistance of his son

Item	Type	Annotation
1011079100200	ABSOL	having|been|passed ARG act
1011079100200	ABSOL	withdrew MOD having|been|passed
1011079100200	ABSOL	carried+on MOD having|been|

The 'localness' or not of the examples

Phenomenon	Head	Type	Dependent	Distance
Bare relatives	gapped predicate in relative	ARG2/MOD	modified noun	3.0 (8)
(barerel)	modified noun	MOD	top predicate of relative	3.3 (8)
Tough adjectives	*tough* adjective	ARG2	*to*-VP complement	1.7 (5)
(tough)	gapped predicate in *to*-VP	ARG2	subject/modifiee of adjective	6.4 (21)
Right Node Raising	verb/prep2	ARG2	shared noun	2.8 (9)
(rnr)	verb/prep1	ARG2	shared noun	6.1 (12)
Expletive It	*it*-subject taking verb	!ARG1	*it*	1.2 (3)
(itexpl)	raising-to-object verb	!ARG2	*it*	–
Verb+particle constructions	particle	!ARG2	complement	2.7 (9)
(vpart)	verb+particle	ARG2	complement	3.7 (10)
Adj/Noun2 + Noun1-*ed*	head noun	MOD	Noun1-*ed*	2.4 (17)
(ned)	Noun1-*ed*	ARG1/MOD	Adj/Noun2	1.0 (1.5)
Absolutives	absolutive predicate	ARG1	subject of absolutive	1.7 (12)
(absol)	main clause predicate	MOD	absolutive predicate	9.8 (26)
Verbal gerunds	selecting head	ARG[1,2]	gerund	1.9 (13)
(vger)	gerund	ARG2/MOD	first complement/modifier of gerund	2.3 (8)
Interleaved arg/adj	selecting verb	MOD	interleaved adjunct	1.2 (7)
(argadj)	selecting verb	ARG[2,3]	displaced complement	5.9 (26)
Control	"upstairs" verb	ARG[2,3]	"downstairs" verb	2.4 (23)
(control)	"downstairs" verb	ARG1	shared argument	4.8 (17)

Table 3: Dependencies labeled for each phenomenon type, including average and maximum surface distances.

To evaluate, extraction patterns using about 400 regex patterns

The players

Trained 'Directly' on the (WSJ Portion of the) PTB

- **Stanford** (Klein & Manning, 2003) factored model; GR output;
- **C&J** (Charniak & Johnson, 2005) Stanford GR post-processor;
- **MST** (McDonald et al., 2005) second-order projective model.

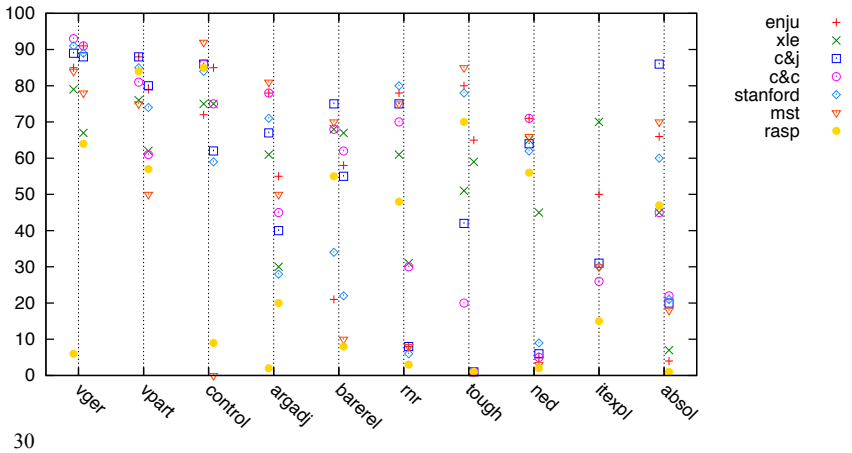
Trained Indirectly on the (WSJ Portion of the) PTB

- **Enju** (Miyao et al., 2004) HPSG; predicate–argument outputs;
- **C&C** (Clark & Curran, 2007) CCG; grammatical relation outputs.

(Partly) Analytically Engineered

- **RASP** (Briscoe et al., 2006) PoS 'tag sequence grammar'; GRs;
- **XLE** (Kaplan et al., 2004) hand-built LFG and lexicon; f-structures.

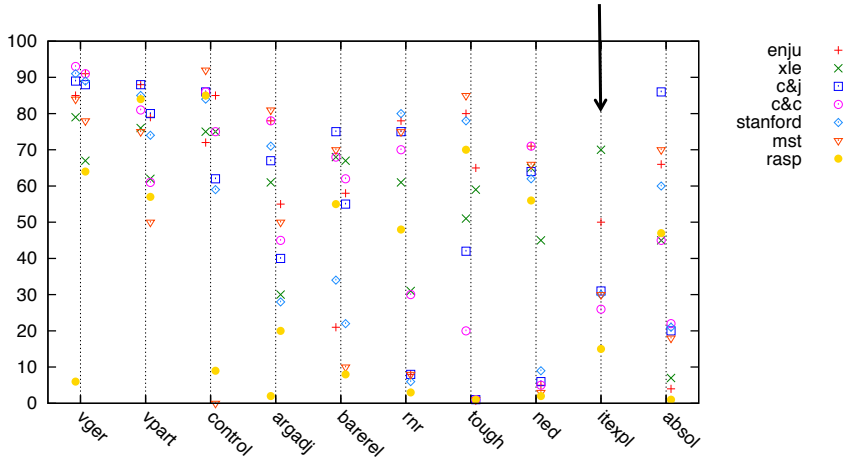
Individual dependency recall



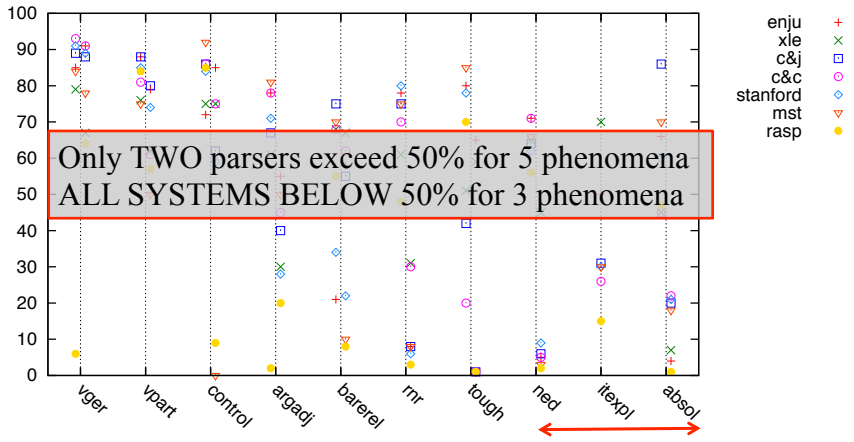
30

Good Recovery of Some Phenomena: VGER, VPART, CONTROL.

Predictable: ITEXPL requires lexical knowledge (not in 'PTB').

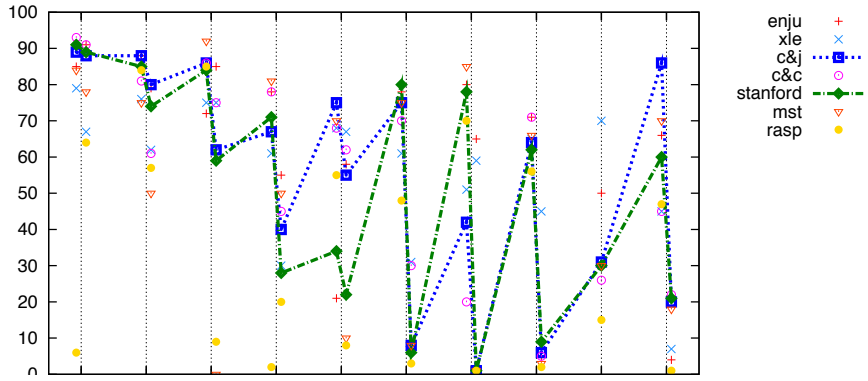


Some Dependencies Lost on Most Parsers: RNR, NED, ABSOL.

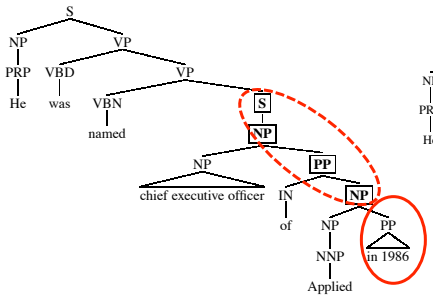


Perhaps more work needs to be done?

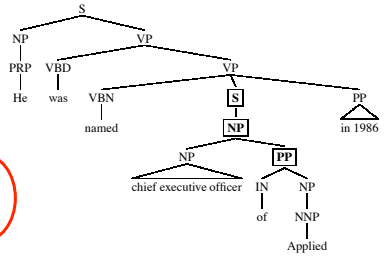
C&J vs. Stanford: Average 56 % vs. 52 %.



Propagated Errors: PP attachment



(a) Parser output



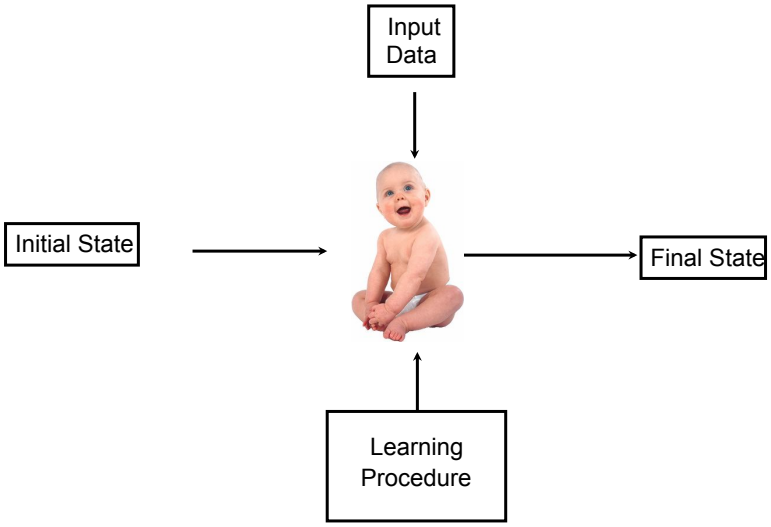
(b) Gold tree

Error Types (1+6 more)

1. (PP attachment: high vs. low)
2. NP attachment: high vs. low)
3. Modifier attachment
4. Clause attachment
5. Unary error: S, NP, etc.
6. Conjunction
7. NP internal structure

What happens when we move to a new domain?

The usual setup for language acquisition



What is a treebank analog?

So are we done???

= $2500/baby

???

Question: How well does this do, from a *cognitive* perspective?

The lessons

- Cognitive 'Turing tests' for treebank parsers - How does their acquired knowledge hold up as a model for *cognitively accurate* language acquisition?
- Three lessons

1. Jackendoff's lesson: *can computers learn to read?* – the Linguistic Society of America's pamphlet on *Why can't computers use English?*
2. Levin's Lesson: Robustness, sensitivity, and lexical semantics: *can computers learn to drink milk with cookies?*
3. The Las Vegas lesson: *what happens in Las Vegas stays in Las Vegas – computers can count, and people cannot*