

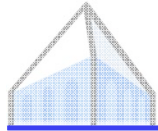
Natural Language Processing



Parsing III

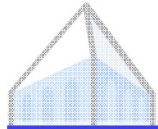
Dan Klein – UC Berkeley

Unsupervised Tagging



Unsupervised Tagging?

- AKA part-of-speech induction
- Task:
 - Raw sentences in
 - Tagged sentences out
- Obvious thing to do:
 - Start with a (mostly) uniform HMM
 - Run EM
 - Inspect results



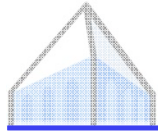
EM for HMMs: Process

- Alternate between recomputing distributions over hidden variables (the tags) and reestimating parameters
- Crucial step: we want to tally up how many (fractional) counts of each kind of transition and emission we have under current params:

$$\text{count}(w, s) = \sum_{i:w_i=w} P(t_i = s | \mathbf{w})$$

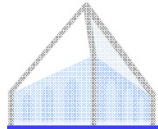
$$\text{count}(s \rightarrow s') = \sum_i P(t_{i-1} = s, t_i = s' | \mathbf{w})$$

- Same quantities we needed to train a CRF!



Merialdo: Setup

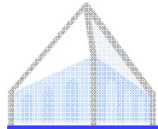
- Some (discouraging) experiments [Merialdo 94]
- Setup:
 - You know the set of allowable tags for each word
 - Fix k training examples to their true labels
 - Learn $P(w|t)$ on these examples
 - Learn $P(t|t_{-1}, t_{-2})$ on these examples
 - On n examples, re-estimate with EM
- Note: we know allowed tags but not frequencies



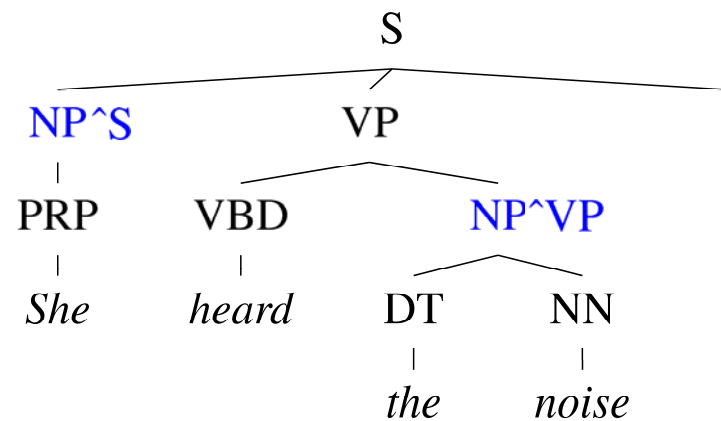
Merialdo: Results

Number of tagged sentences used for the initial model							
	0	100	2000	5000	10000	20000	all
Iter	Correct tags (% words) after ML on 1M words						
0	77.0	90.0	95.4	96.2	96.6	96.9	97.0
1	80.5	92.6	95.8	96.3	96.6	96.7	96.8
2	81.8	93.0	95.7	96.1	96.3	96.4	96.4
3	83.0	93.1	95.4	95.8	96.1	96.2	96.2
4	84.0	93.0	95.2	95.5	95.8	96.0	96.0
5	84.8	92.9	95.1	95.4	95.6	95.8	95.8
6	85.3	92.8	94.9	95.2	95.5	95.6	95.7
7	85.8	92.8	94.7	95.1	95.3	95.5	95.5
8	86.1	92.7	94.6	95.0	95.2	95.4	95.4
9	86.3	92.6	94.5	94.9	95.1	95.3	95.3
10	86.6	92.6	94.4	94.8	95.0	95.2	95.2

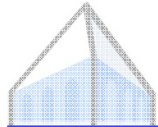
Latent Variable PCFGs



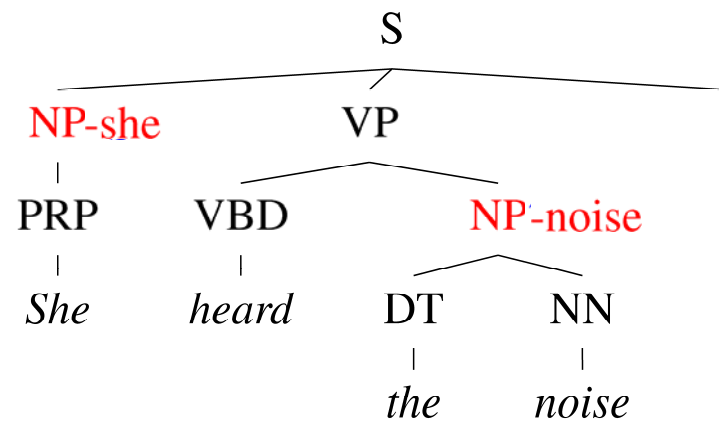
The Game of Designing a Grammar



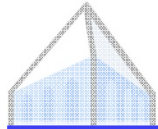
- Annotation refines base treebank symbols to improve statistical fit of the grammar
 - Parent annotation [Johnson '98]



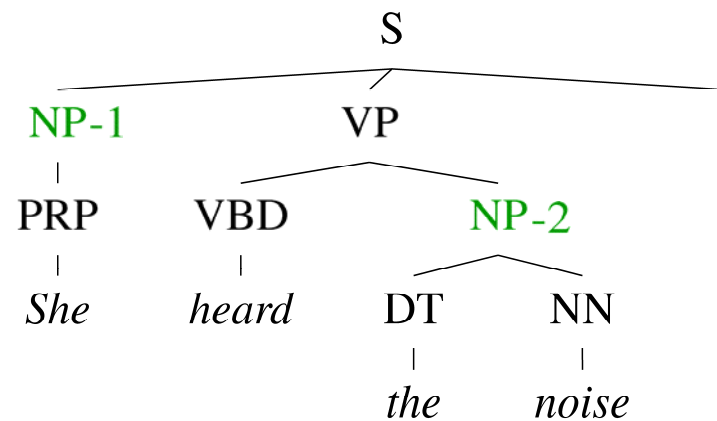
The Game of Designing a Grammar



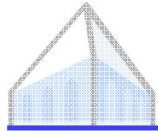
- Annotation refines base treebank symbols to improve statistical fit of the grammar
 - Parent annotation [Johnson '98]
 - Head lexicalization [Collins '99, Charniak '00]



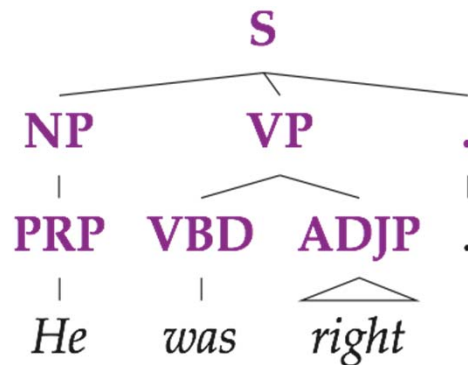
The Game of Designing a Grammar



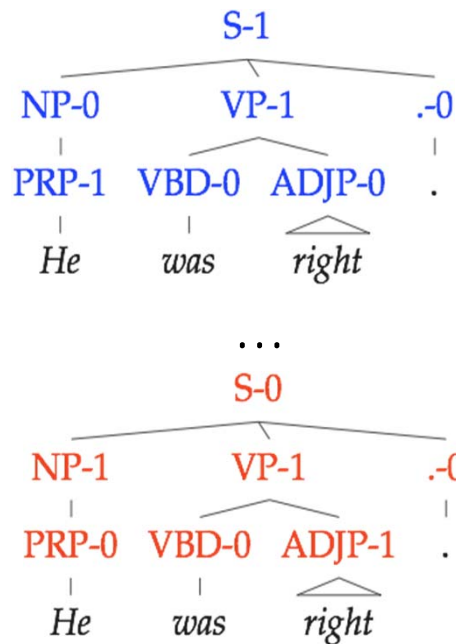
- Annotation refines base treebank symbols to improve statistical fit of the grammar
 - Parent annotation [Johnson '98]
 - Head lexicalization [Collins '99, Charniak '00]
 - Automatic clustering?



Latent Variable Grammars



Parse Tree T
Sentence w



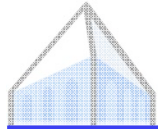
Derivations $t : T$



Grammar G		
$S_0 \rightarrow NP_0 VP_0$?	
$S_0 \rightarrow NP_1 VP_0$?	
$S_0 \rightarrow NP_0 VP_1$?	
$S_0 \rightarrow NP_1 VP_1$?	
$S_1 \rightarrow NP_0 VP_0$?	
...		
$S_1 \rightarrow NP_1 VP_1$?	
...		
$NP_0 \rightarrow PRP_0$?	
$NP_0 \rightarrow PRP_1$?	
...		

Lexicon		
$PRP_0 \rightarrow She$?	
$PRP_1 \rightarrow She$?	
...		
$VBD_0 \rightarrow was$?	
$VBD_1 \rightarrow was$?	
$VBD_2 \rightarrow was$?	
...		

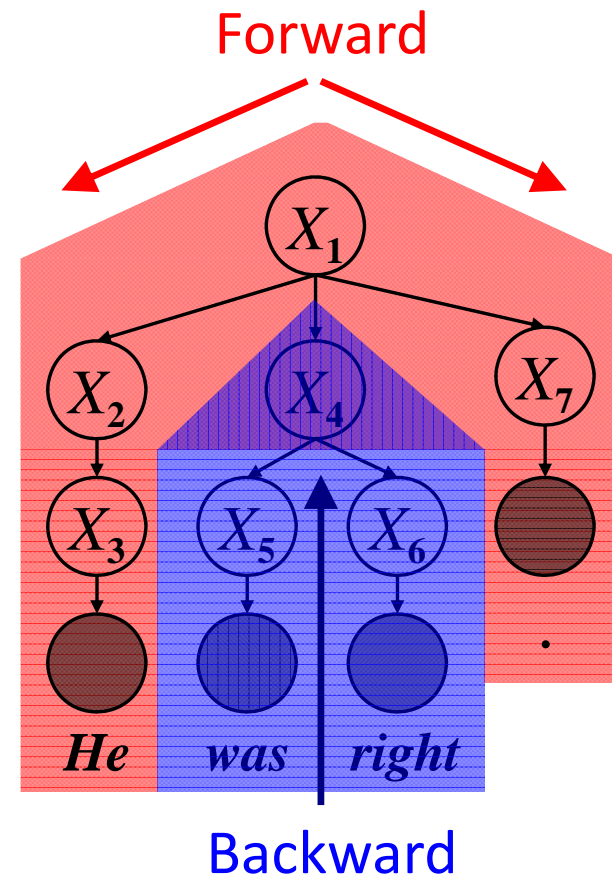
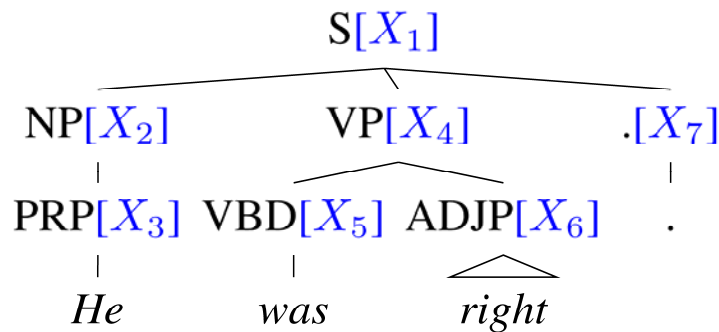
Parameters θ



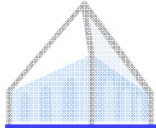
Learning Latent Annotations

EM algorithm:

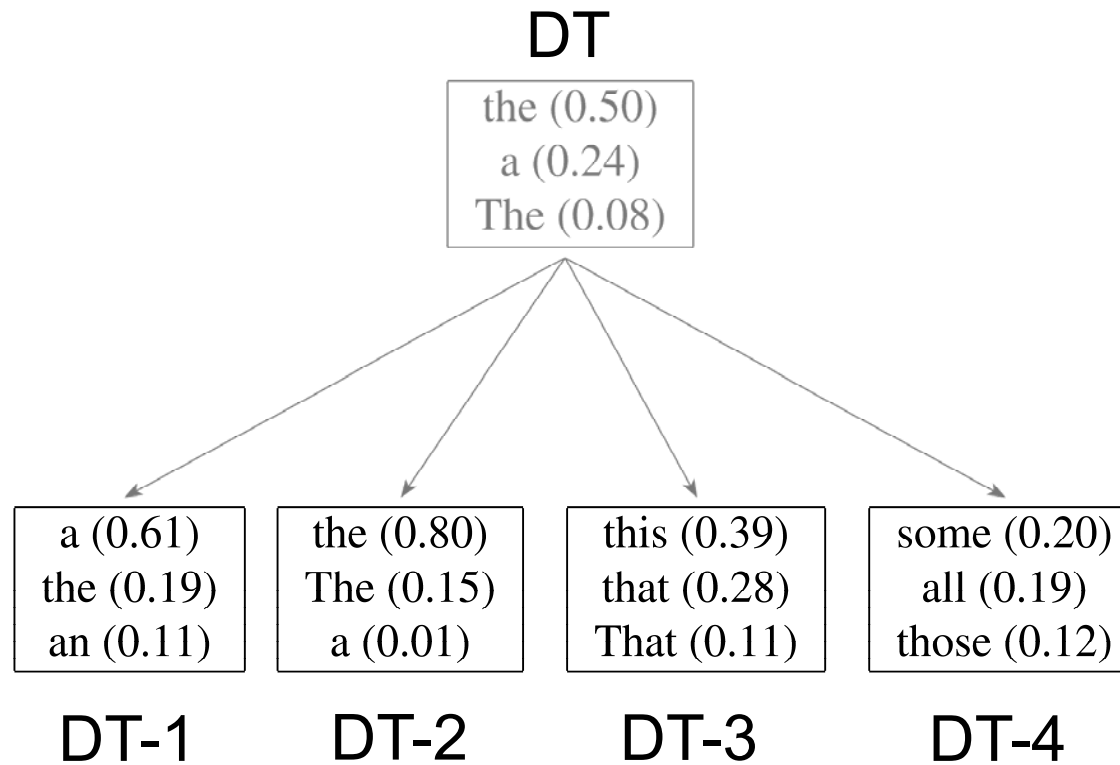
- Brackets are known
- Base categories are known
- Only induce subcategories

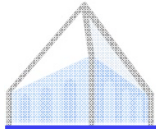


Just like Forward-Backward for HMMs.

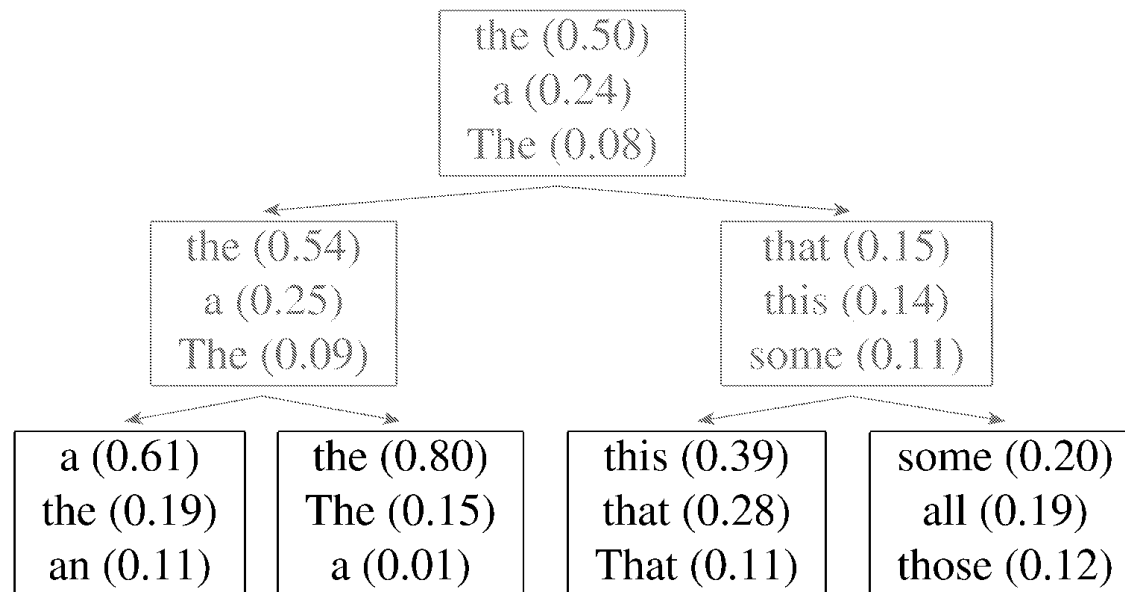


Refinement of the DT tag



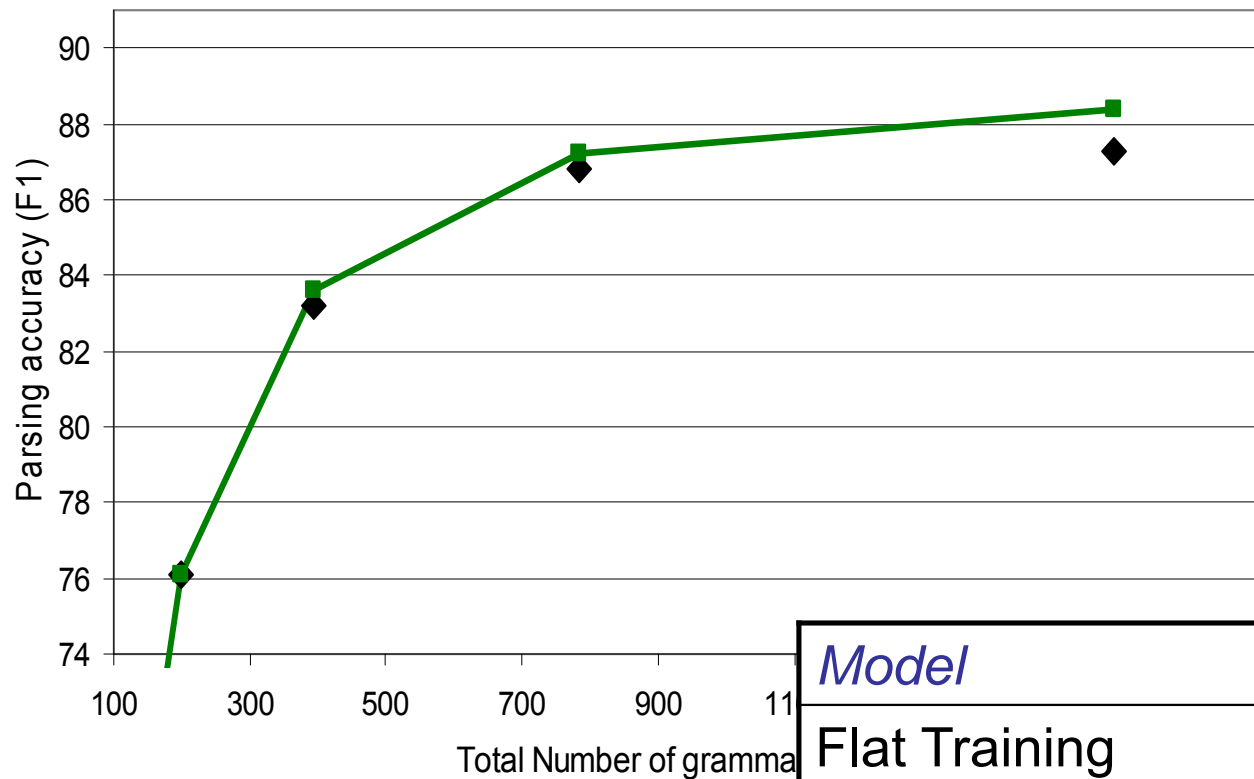


Hierarchical refinement

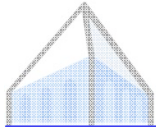




Hierarchical Estimation Results

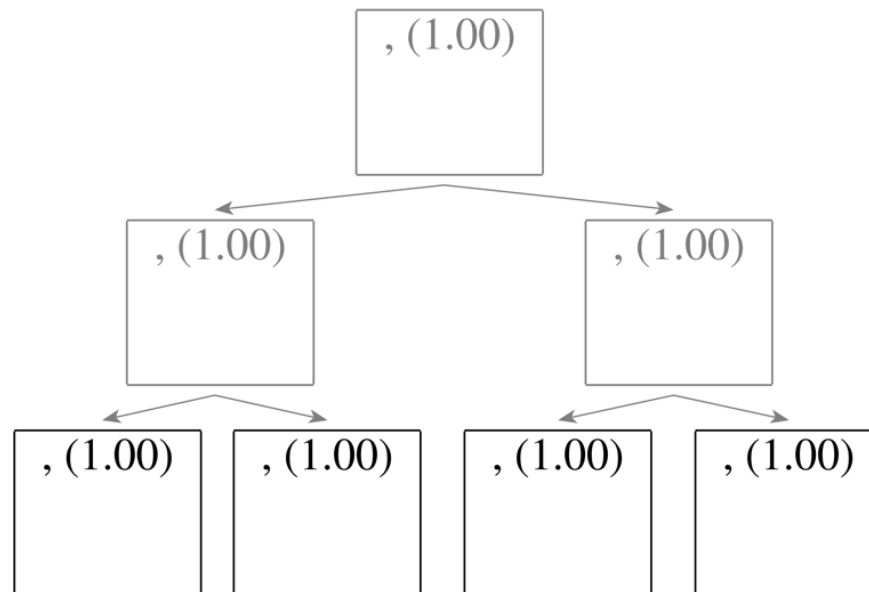


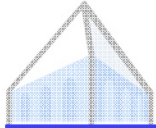
<i>Model</i>	<i>F1</i>
Flat Training	87.3
Hierarchical Training	88.4



Refinement of the , tag

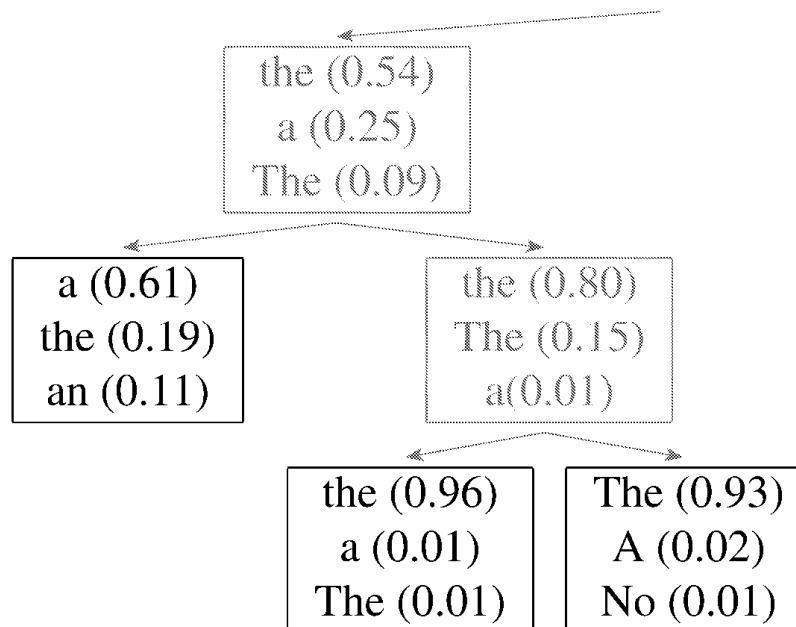
- Splitting all categories equally is wasteful:

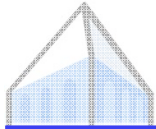




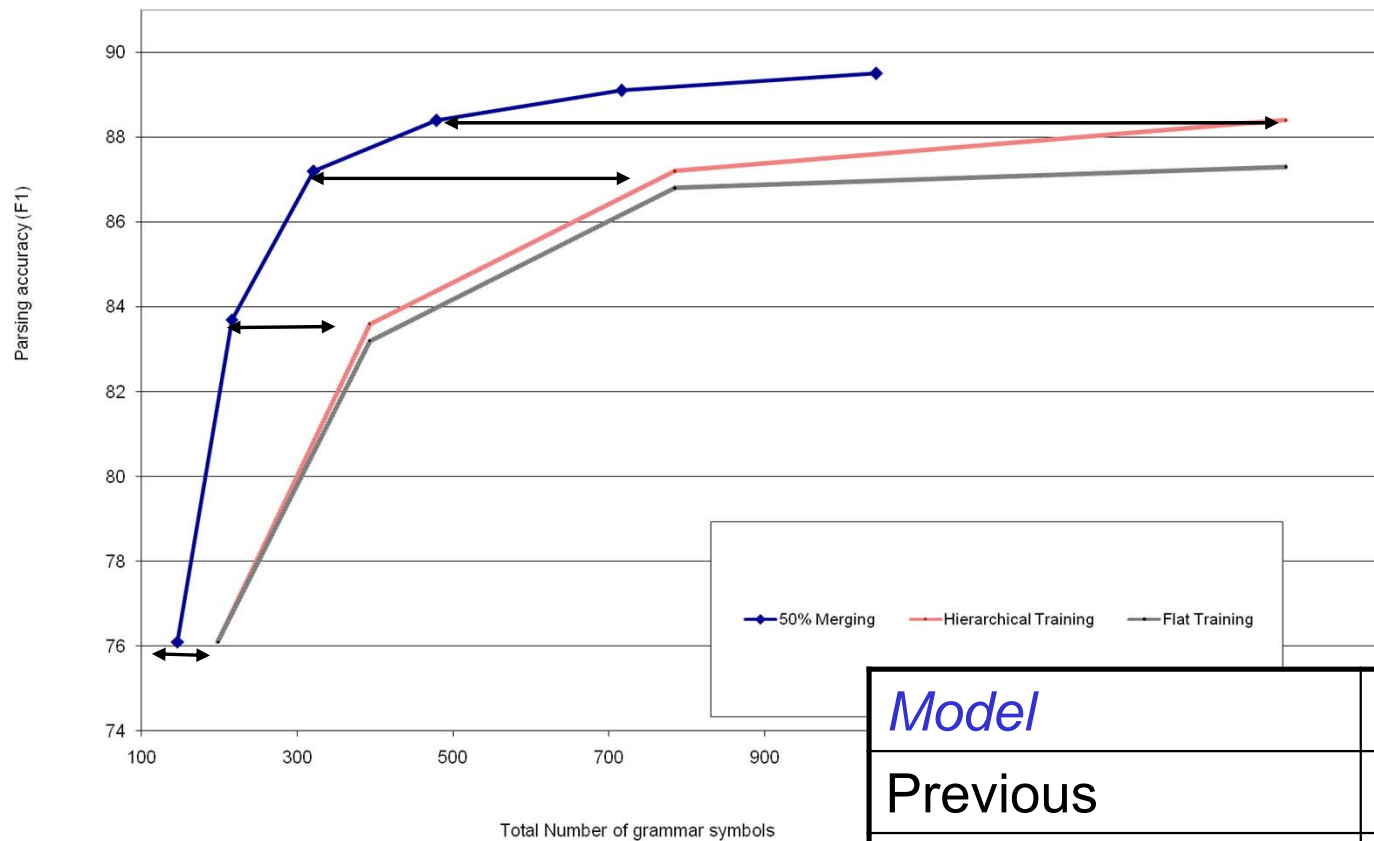
Adaptive Splitting

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

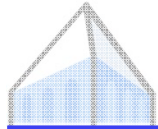




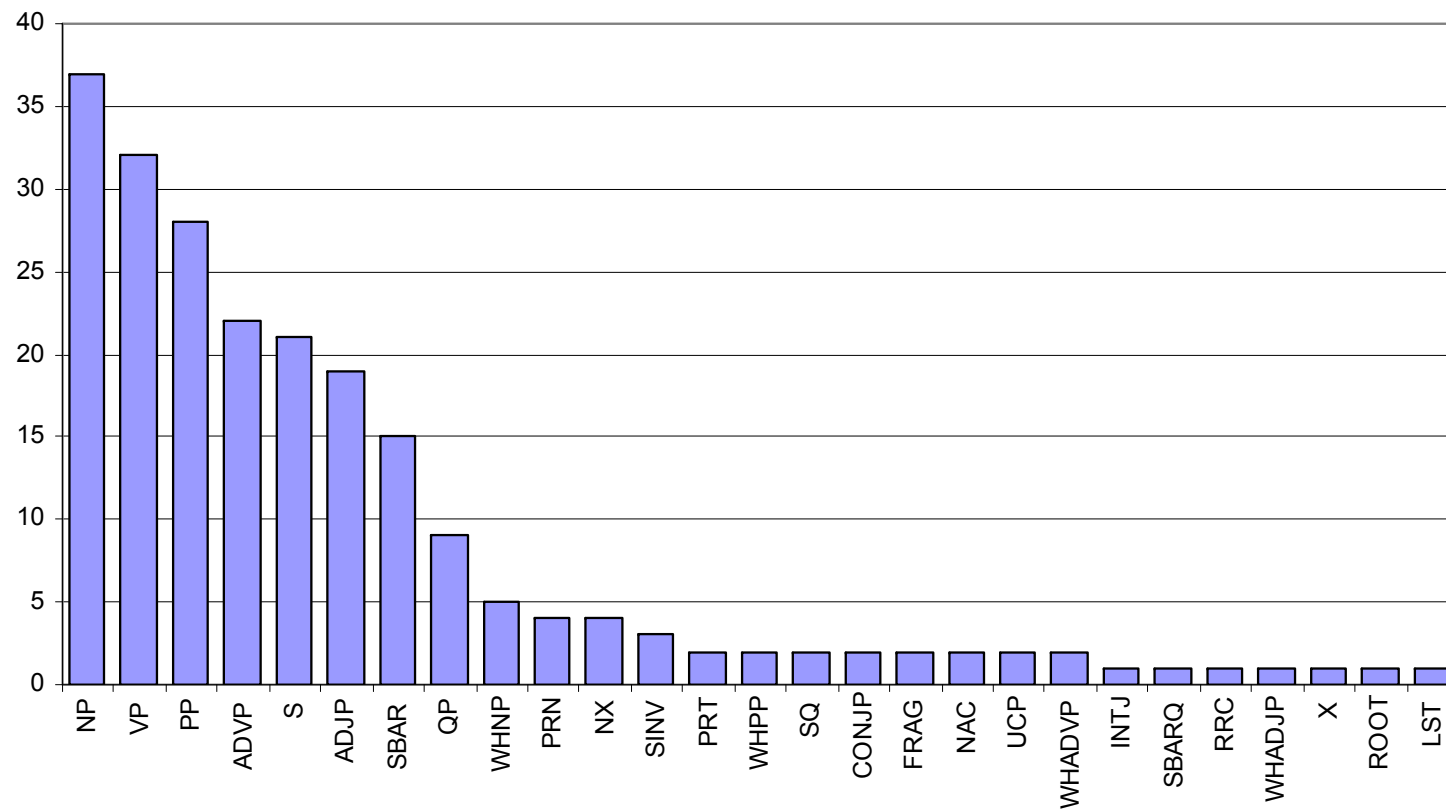
Adaptive Splitting Results

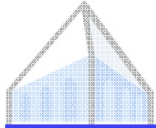


<i>Model</i>	<i>F1</i>
Previous	88.4
With 50% Merging	89.5

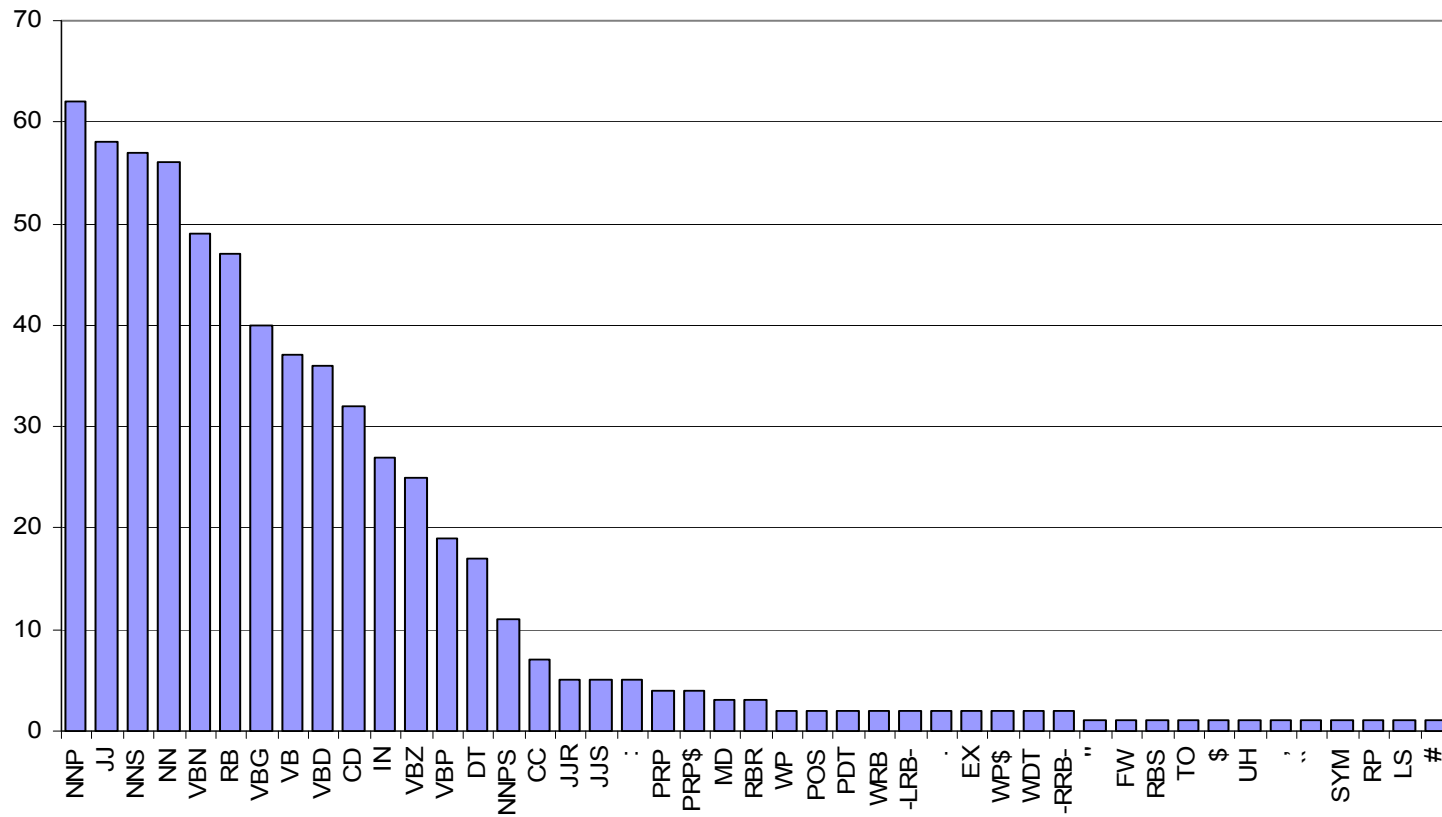


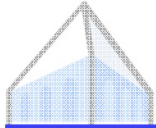
Number of Phrasal Subcategories





Number of Lexical Subcategories





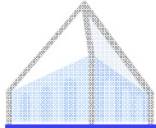
Learned Splits

- 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

- Personal pronouns (PRP):

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



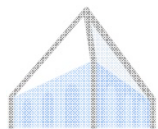
Learned Splits

- 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

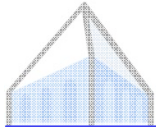


Final Results (Accuracy)

		≤ 40 words F1	all F1
ENG	Charniak&Johnson '05 (generative)	90.1	89.6
	Split / Merge	90.6	90.1
GER	Dubey '05	76.3	-
	Split / Merge	80.8	80.1
CHN	Chiang et al. '02	80.0	76.6
	Split / Merge	86.3	83.4

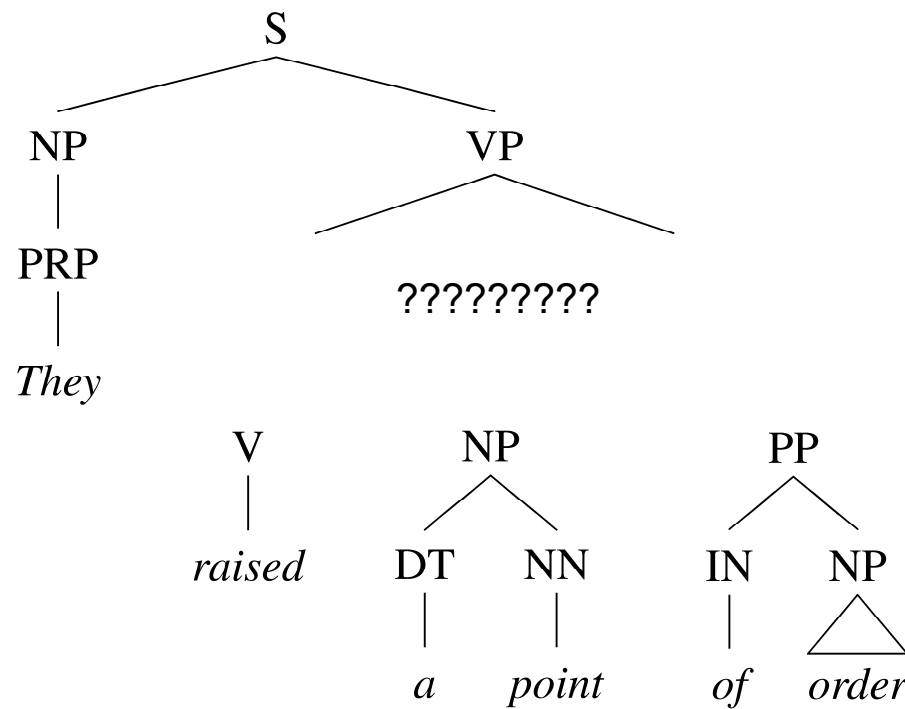
Still higher numbers from reranking / self-training methods

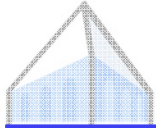
Efficient Parsing for Hierarchical Grammars



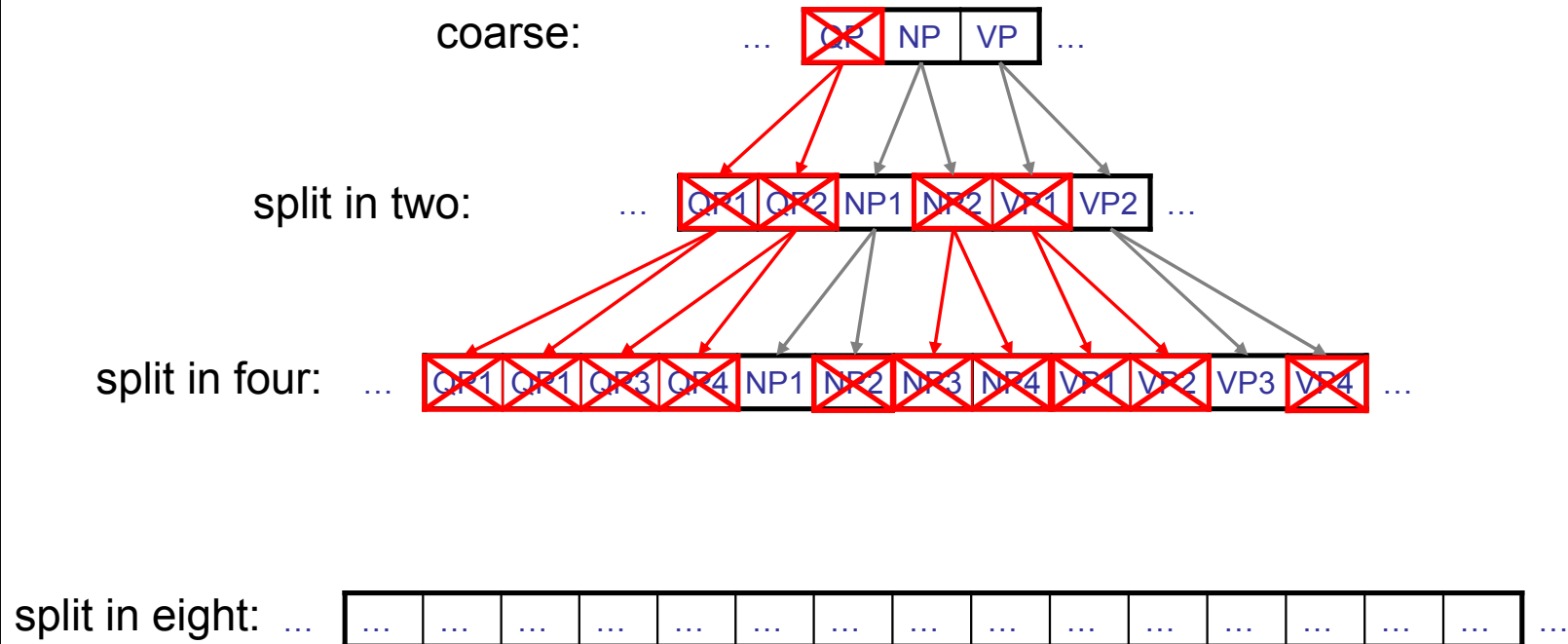
Coarse-to-Fine Inference

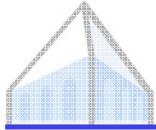
- Example: PP attachment



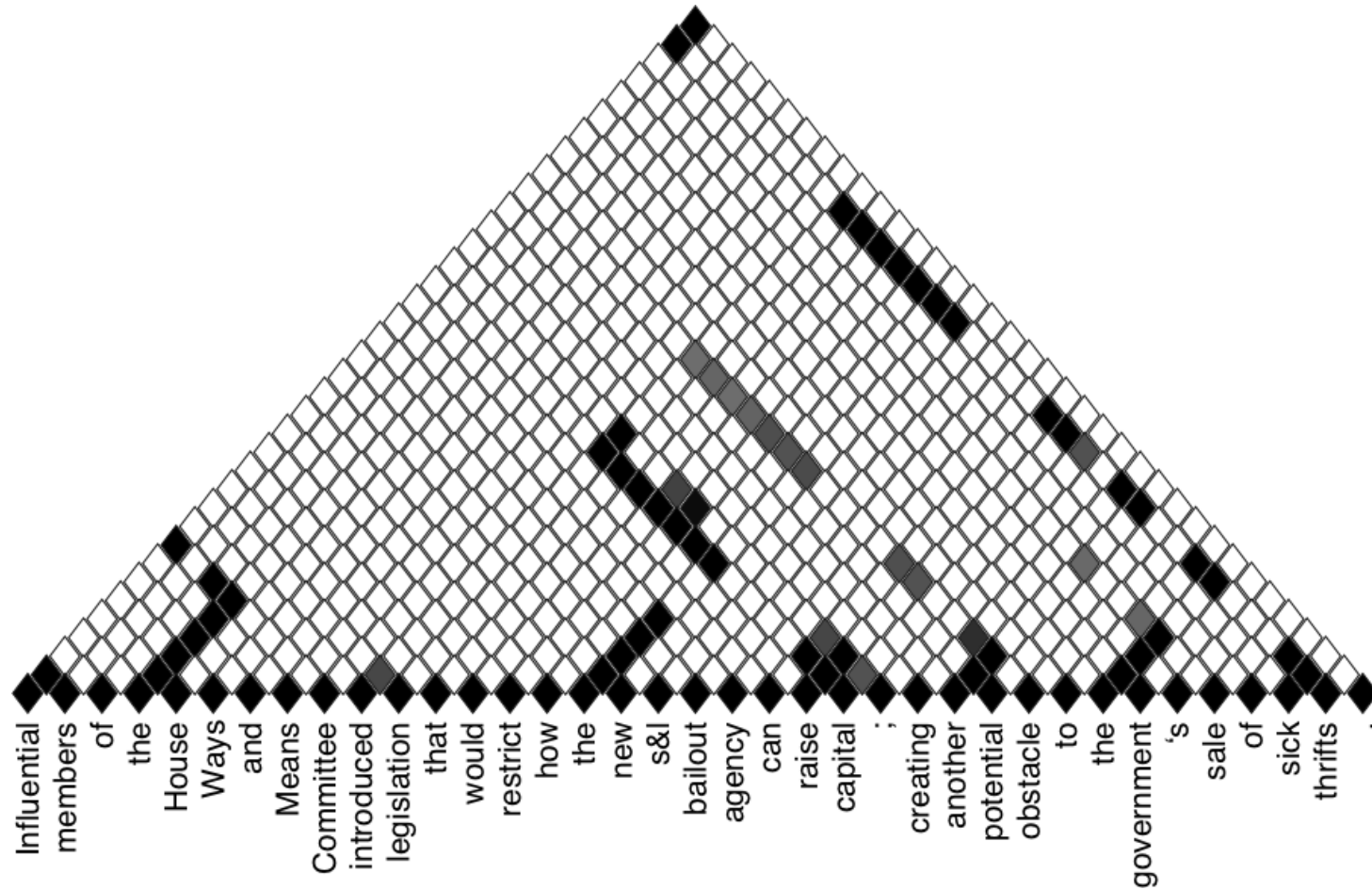


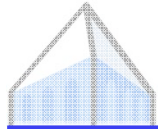
Hierarchical Pruning





Bracket Posteriors





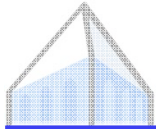
1621 min

111 min

35 min

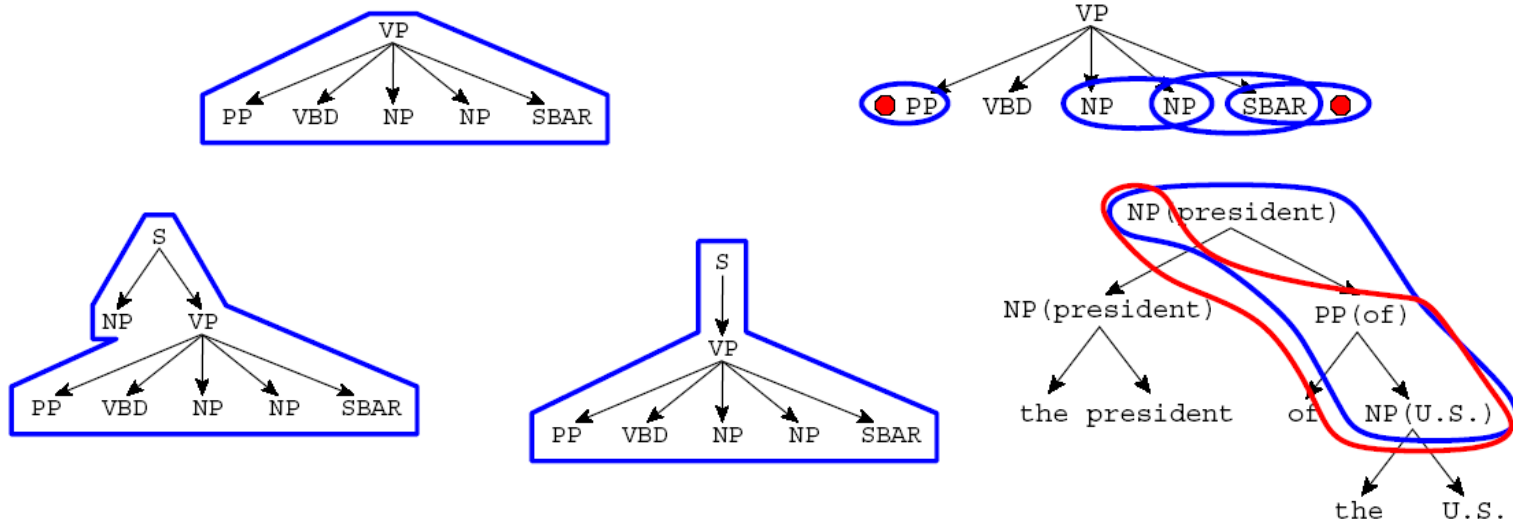
15 min
(no search error)

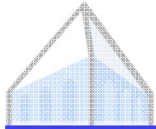
Other Syntactic Models



Parse Reranking

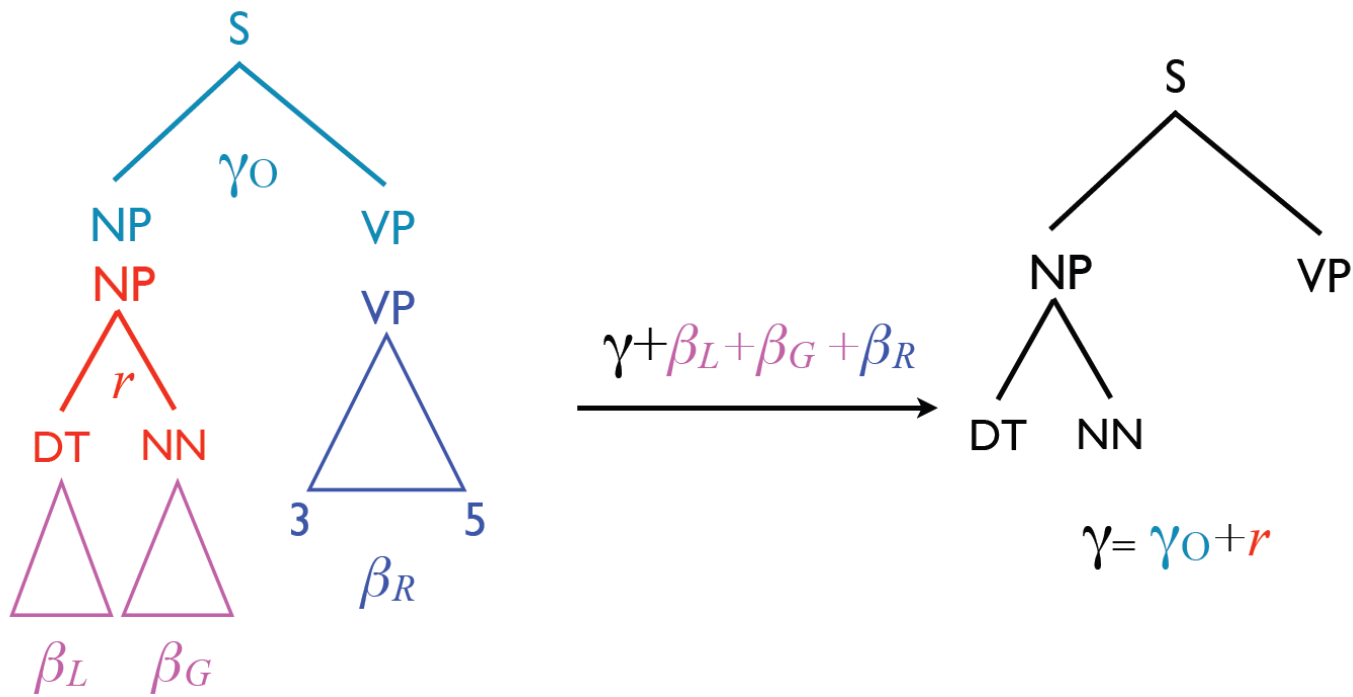
- Assume the number of parses is very small
- We can represent each parse T as an arbitrary feature vector $\phi(T)$
 - Typically, all local rules are features
 - Also non-local features, like how right-branching the overall tree is
 - [Charniak and Johnson 05] gives a rich set of features

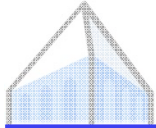




K-Best Parsing

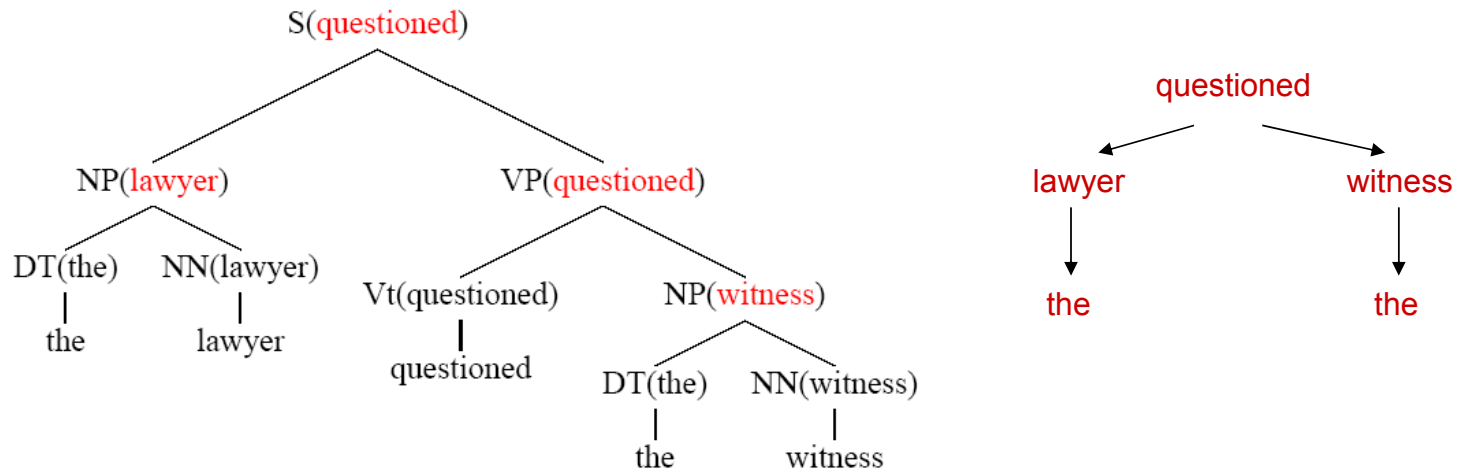
[Huang and Chiang 05,
Pauls, Klein, Quirk 10]



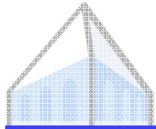


Dependency Parsing

- Lexicalized parsers can be seen as producing *dependency trees*

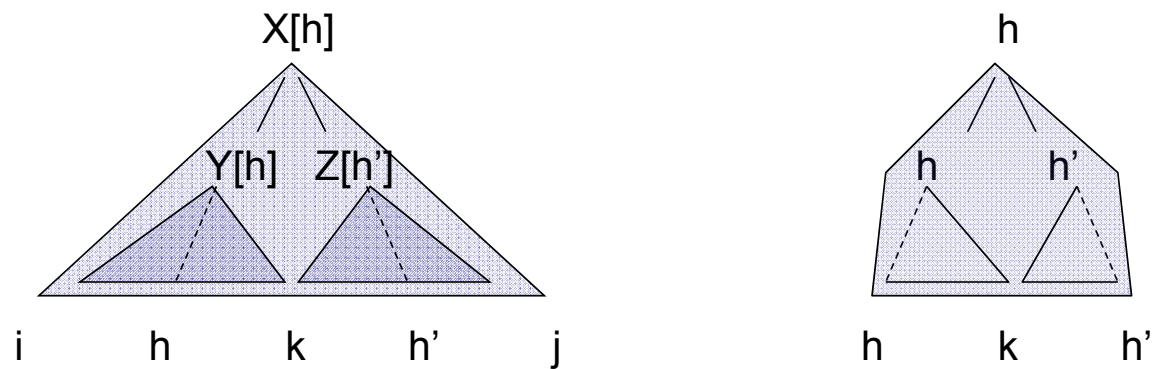


- Each local binary tree corresponds to an attachment in the dependency graph

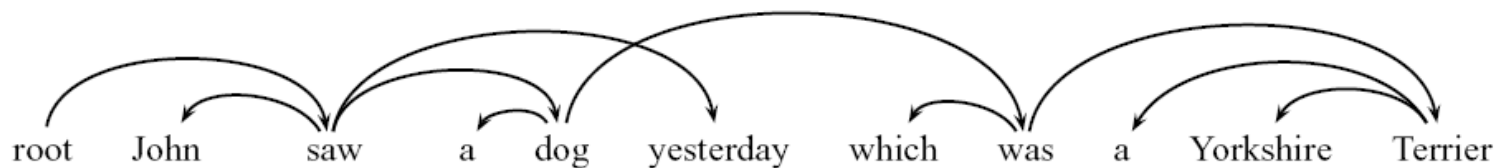


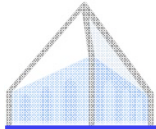
Dependency Parsing

- Pure dependency parsing is only cubic [Eisner 99]



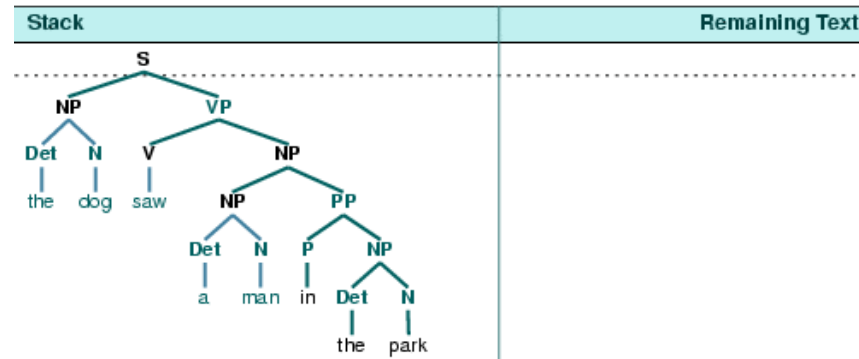
- Some work on *non-projective* dependencies
 - Common in, e.g. Czech parsing
 - Can do with MST algorithms [McDonald and Pereira 05]



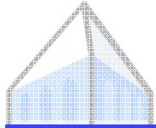


Shift-Reduce Parsers

- Another way to derive a tree:

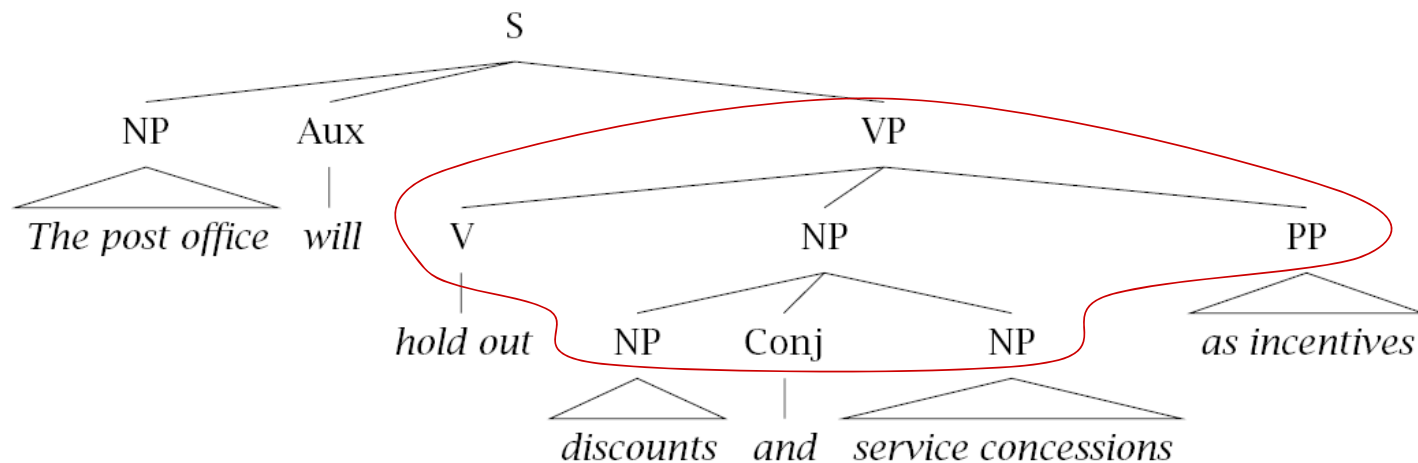


- Parsing
 - No useful dynamic programming search
 - Can still use beam search [Ratnaparkhi 97]

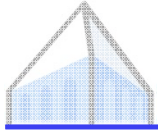


Data-oriented parsing:

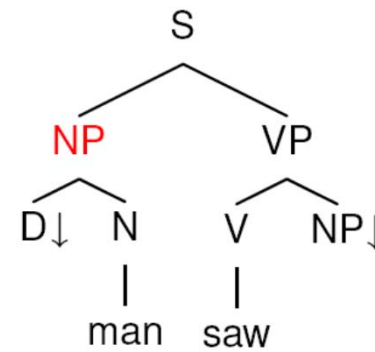
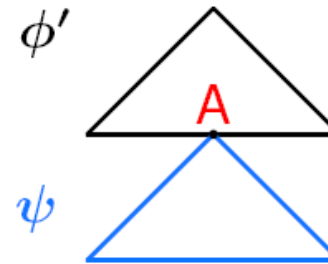
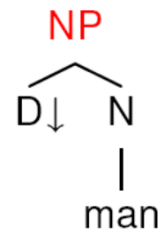
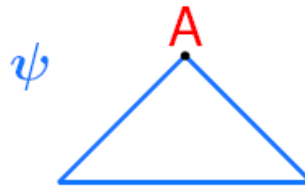
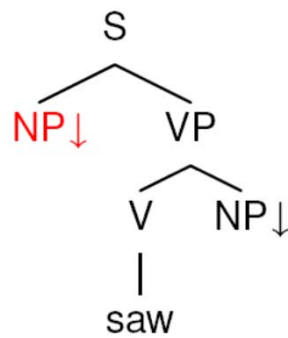
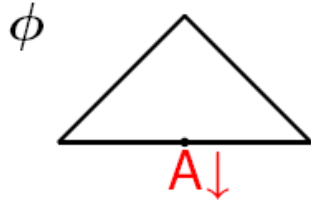
- Rewrite large (possibly lexicalized) subtrees in a single step

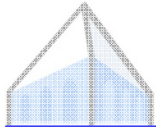


- Formally, a *tree-insertion grammar*
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable *parse* is NP-complete



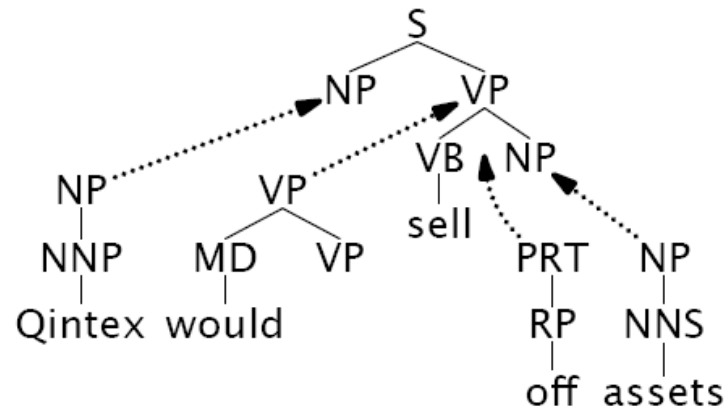
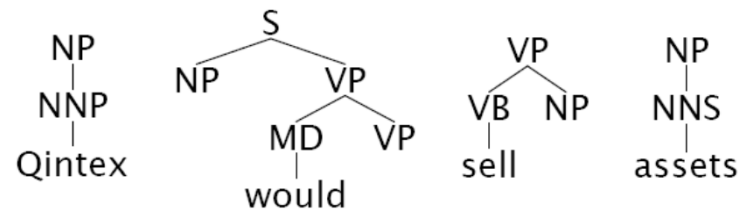
TIG: Insertion

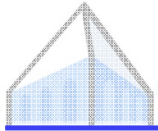




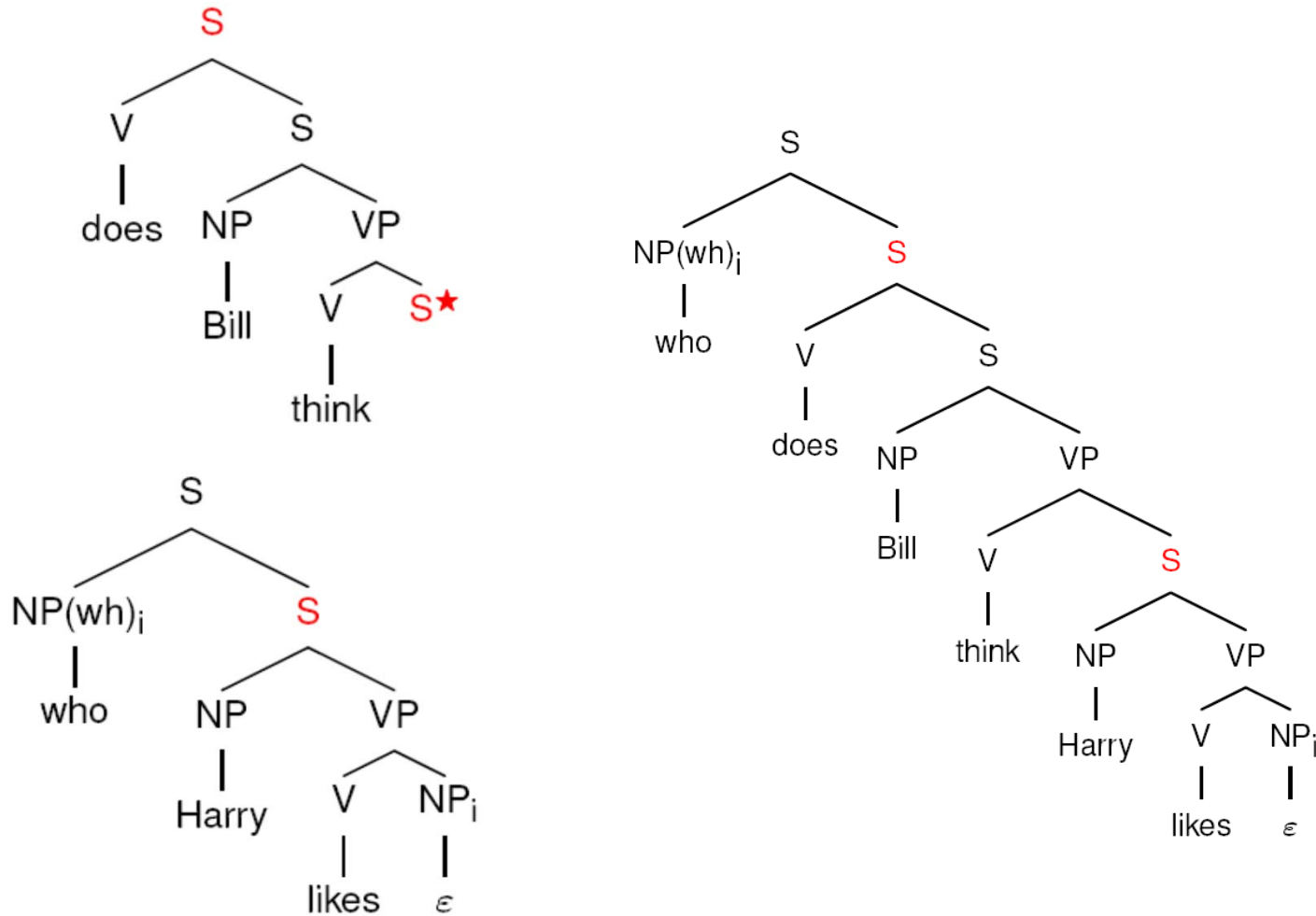
Tree-adjoining grammars

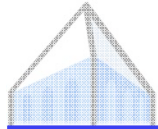
- Start with *local trees*
- Can insert structure with *adjunction* operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- ... as well as other weird stuff that CFGs don't capture well (e.g. cross-serial dependencies)





TAG: Long Distance





CCG Parsing

- **Combinatory
Categorial Grammar**
 - Fully (mono-) lexicalized grammar
 - Categories encode argument sequences
 - Very closely related to the lambda calculus (more later)
 - Can have spurious ambiguities (why?)

John \vdash NP

shares \vdash NP

buys \vdash (S\NP)/NP

sleeps \vdash S\NP

well \vdash (S\NP)\(S\NP)

