Lexicalization of PCFGs

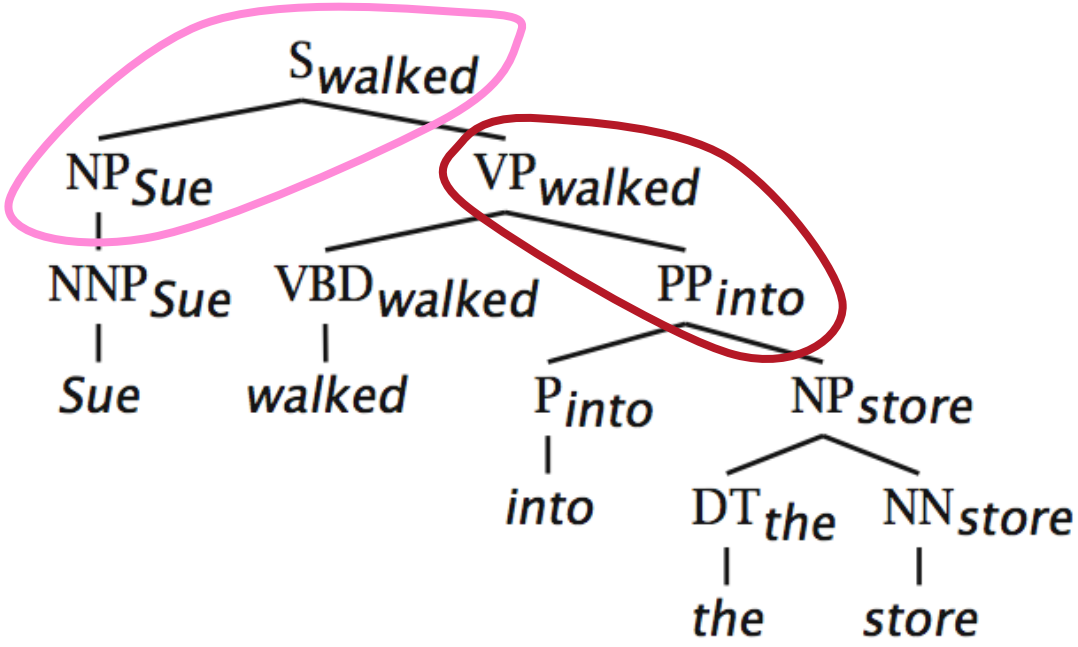
Introduction

Christopher Manning

(Head) Lexicalization of PCFGs

[Magerman 1995, Collins 1997; Charniak 1997]

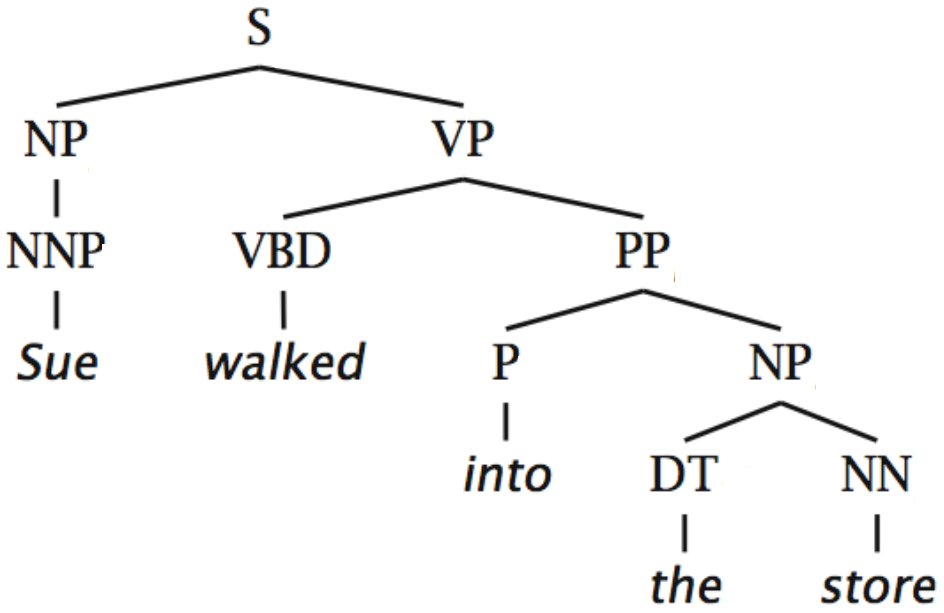
- The head word of a phrase gives a good representation of the phrase's structure and meaning
- Puts the properties of words back into a PCFG



(Head) Lexicalization of PCFGs

[Magerman 1995, Collins 1997; Charniak 1997]

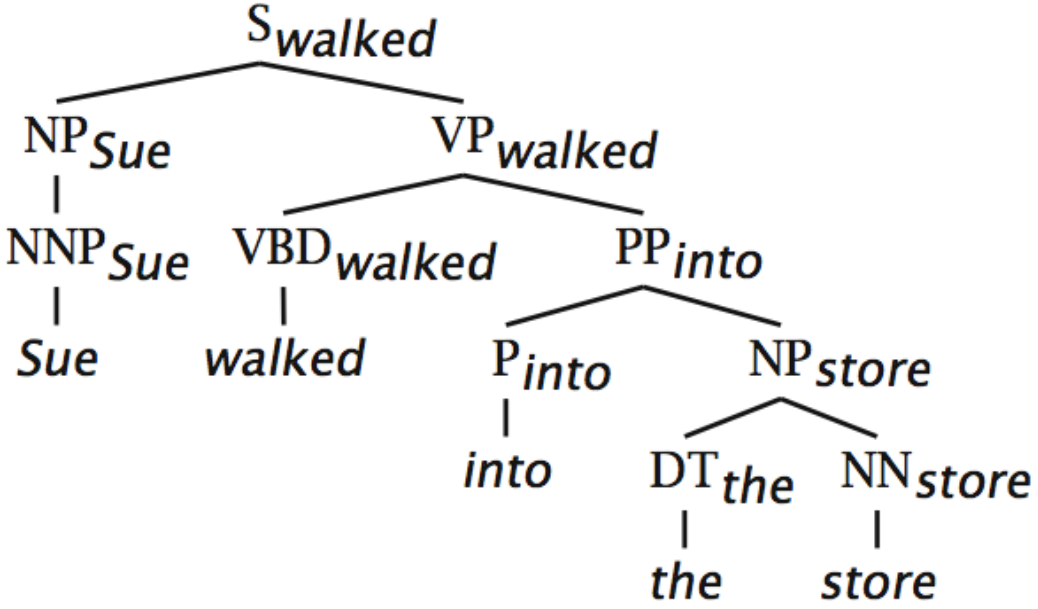
- The head word of a phrase gives a good representation of the phrase's structure and meaning
- Puts the properties of words back into a PCFG



(Head) Lexicalization of PCFGs

[Magerman 1995, Collins 1997; Charniak 1997]

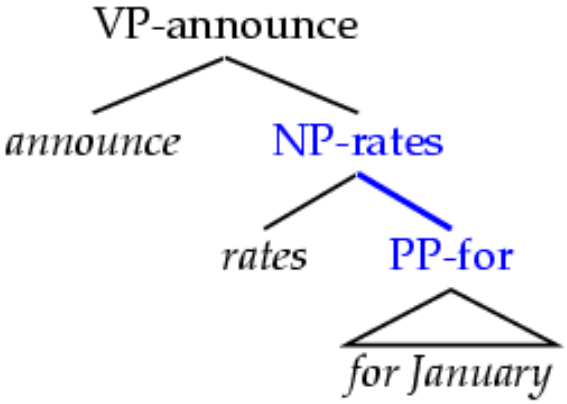
- The head word of a phrase gives a good representation of the phrase's structure and meaning
- Puts the properties of words back into a PCFG

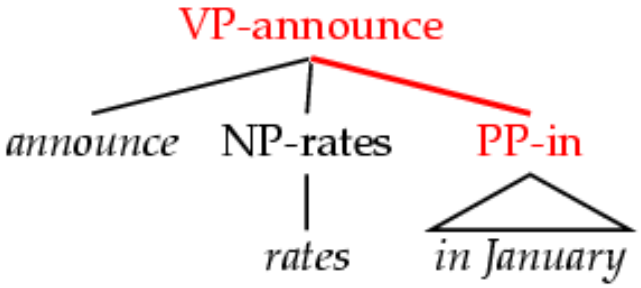


(Head) Lexicalization of PCFGs

[Magerman 1995, Collins 1997; Charniak 1997]

- Word-to-word affinities are useful for certain ambiguities
  - PP attachment is now (partly) captured in a local PCFG rule.
    - Think about: What useful information isn't captured?





  - Also useful for: coordination scope, verb complement patterns

Lexicalized parsing was seen as *the* parsing breakthrough of the late 1990s

- Eugene Charniak, 2000 JHU workshop: "To do better, it is necessary to condition probabilities on the actual words of the sentence. This makes the probabilities much tighter:
  - $p(\text{VP} \rightarrow \text{V NP NP})$ = 0.00151
  - $p(\text{VP} \rightarrow \text{V NP NP} \mid \text{said})$ = 0.00001
  - $p(\text{VP} \rightarrow \text{V NP NP} \mid \text{gave})$ = 0.01980 "
- Michael Collins, 2003 COLT tutorial: "Lexicalized Probabilistic Context-Free Grammars … perform vastly better than PCFGs (88% vs. 73% accuracy)"

Lexicalization of PCFGs

Introduction

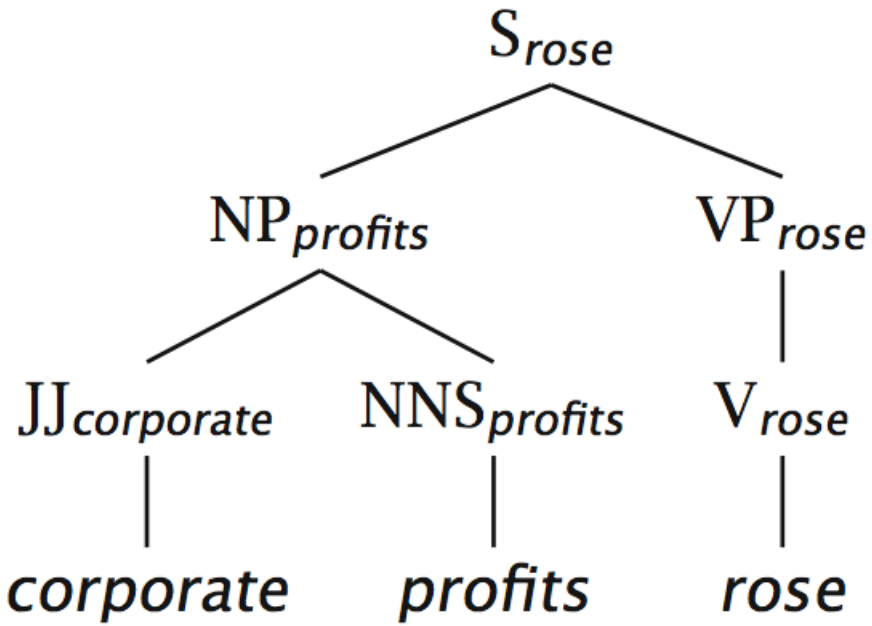
Christopher Manning

Lexicalization of PCFGs

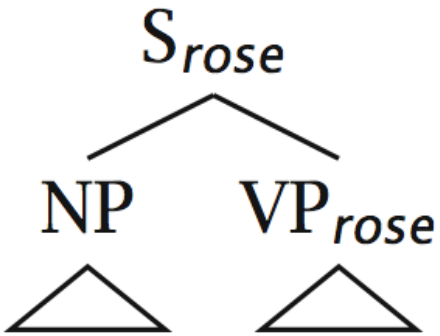
The model of Charniak (1997)

Charniak (1997)

- A very straightforward model of a lexicalized PCFG
- Probabilistic conditioning is "top-down" like a regular PCFG
  - But actual parsing is bottom-up, somewhat like the CKY algorithm we saw



Charniak (1997) example

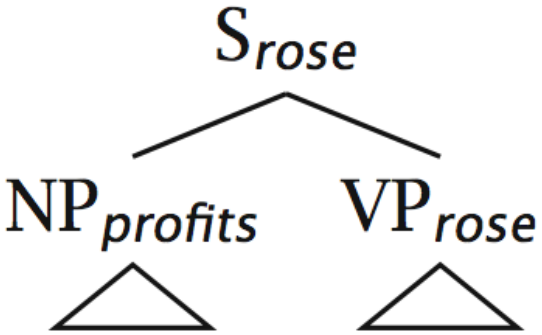


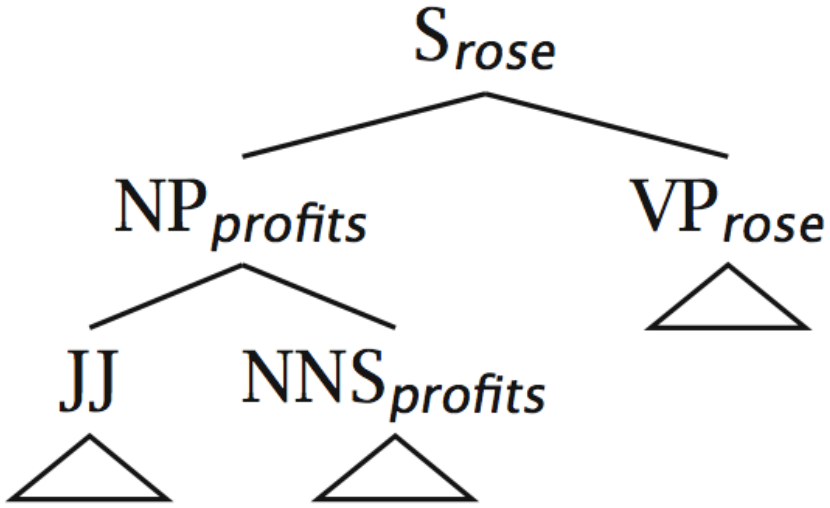
a. $h = profits$; $c = \text{NP}$

b. $ph = rose$; $pc = \text{S}$

c. $P(h|ph, c, pc)$

d. $P(r|h, c, pc)$





Lexicalization models argument selection by sharpening rule expansion probabilities

- The probability of different verbal complement frames (i.e., "subcategorizations") depends on the verb:

Local Tree	*come*	*take*	*think*	*want*
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%
VP → V NP S	0.1%	5.7%	0.0%	0.3%
VP → V PRT NP	0.3%	5.8%	0.0%	0.0%
VP → V PRT PP	6.1%	1.5%	0.2%	0.0%

"monolexical" probabilities

Lexicalization sharpens probabilities: Predicting heads

"Bilexical probabilities"

- P(prices | n-plural) = .013
- P(prices | n-plural, NP) = .013
- P(prices | n-plural, NP, S) = .025
- P(prices | n-plural, NP, S, v-past) = .052
- P(prices | n-plural, NP, S, v-past, fell) = .146

Charniak (1997) linear interpolation/ shrinkage

$$\begin{aligned}\hat{P}(h|ph,c,pc) &= \lambda_1(e)P_{\text{MLE}}(h|ph,c,pc) \\ &\quad +\lambda_2(e)P_{\text{MLE}}(h|C(ph),c,pc) \\ &\quad +\lambda_3(e)P_{\text{MLE}}(h|c,pc) + \lambda_4(e)P_{\text{MLE}}(h|c)\end{aligned}$$

- $\lambda_i(e)$ is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- $C(ph)$ is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction

Charniak (1997) shrinkage example

	$P(\text{prft}\mid\text{rose}, \text{NP}, \text{S})$	$P(\text{corp}\mid\text{prft}, \text{JJ}, \text{NP})$
$P(h\mid ph, c, pc)$	0	0.245
$P(h\mid C(ph), c, pc)$	0.00352	0.0150
$P(h\mid c, pc)$	0.000627	0.00533
$P(h\mid c)$	0.000557	0.00418

- Allows utilization of rich highly conditioned estimates, but smoothes when sufficient data is unavailable
- One can't just use MLEs: one commonly sees previously unseen events, which would have probability 0.

Lexicalization of PCFGs

The model of Charniak (1997)

Sparseness & the Penn Treebank

- The Penn Treebank – 1 million words of parsed English WSJ – has been a key resource (because of the widespread reliance on supervised learning)
- But 1 million words is like nothing:
  - 965,000 constituents, but only 66 WHADJP, of which only 6 aren't *how much* or *how many*, but there is an infinite space of these
    - *How clever/original/incompetent (at risk assessment and evaluation) …*
- Most of the probabilities that you would like to compute, you can't compute

Quiz question!

- Classify each of the italic red phrases as a:

  WHNP WHADJP WHADVP WHPP

1. That explains *why* she is succeeding.
2. *Which student* scored highest on the assignment?
3. Nobody knows *how deep* the recession will be.
4. *During which class* did the slide projection not work?
5. *Whose iPhone* was stolen?

Sparseness & the Penn Treebank (2)

- Many parse preferences depend on bilexical statistics: likelihoods of relationships between pairs of words (compound nouns, PP attachments, …)
- Extremely sparse, even on topics central to the WSJ:
  - *stocks plummeted* 2 occurrences
  - *stocks stabilized* 1 occurrence
  - *stocks skyrocketed* 0 occurrences
  - #*stocks discussed* 0 occurrences
- There has been only modest success in augmenting the Penn Treebank with extra unannotated materials or using semantic classes – given a reasonable amount of annotated training data.
  - Cf. Charniak 1997, Charniak 2000
  - But McClosky et al. 2006 doing self-training and Koo and Collins2008 semantic classes are rather more successful!

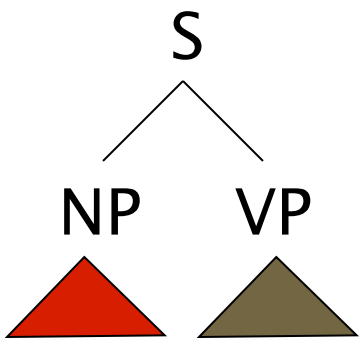
PCFG Independence Assumptions

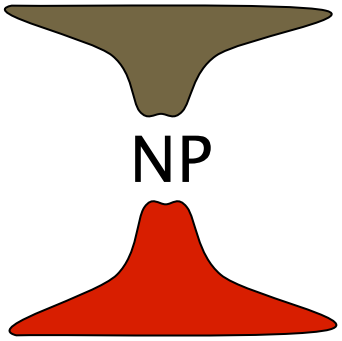
PCFGs and Independence

- The symbols in a PCFG define independence assumptions:

S → NP VP

NP → DT NN

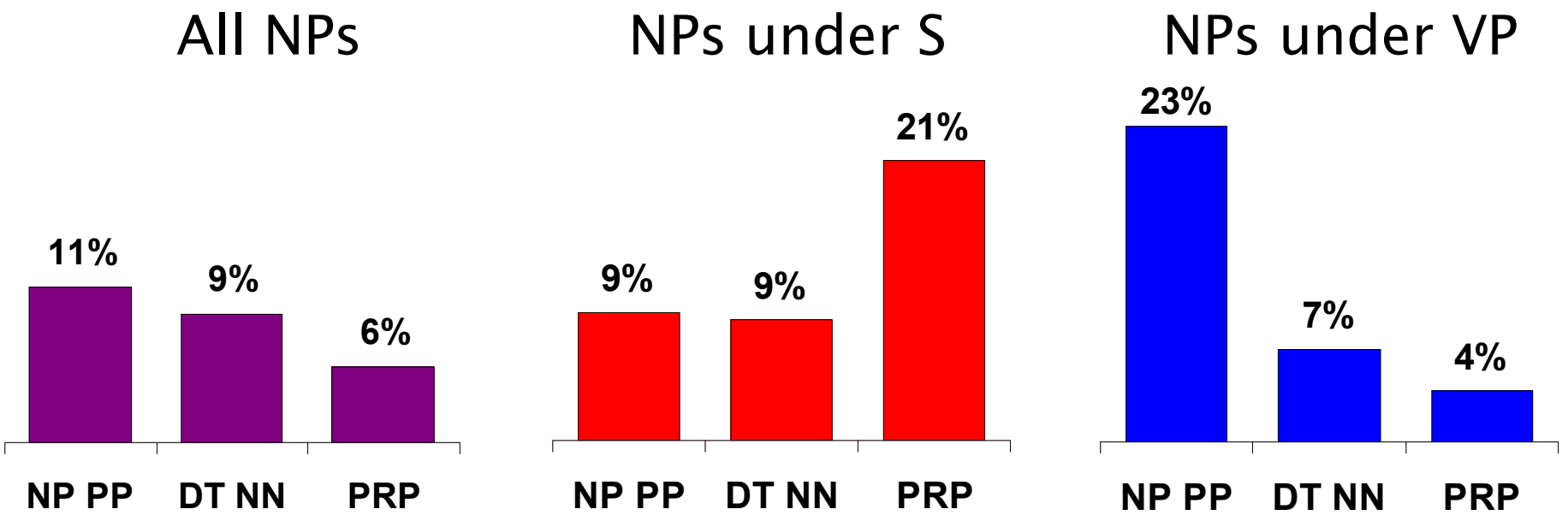




- At any node, the material inside that node is independent of the material outside that node, given the label of that node
- Any information that statistically connects behavior inside and outside a node must flow through that node's label

Non-Independence I

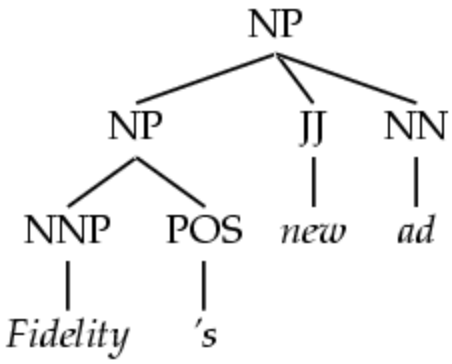
- The independence assumptions of a PCFG are often too strong

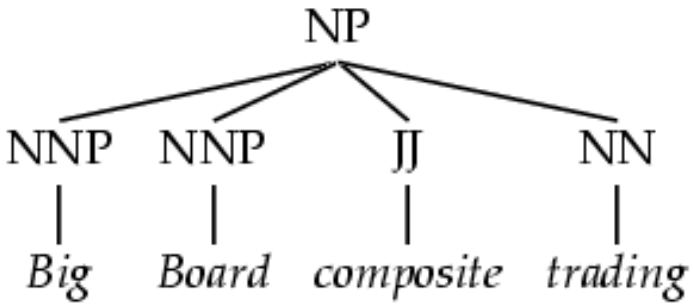


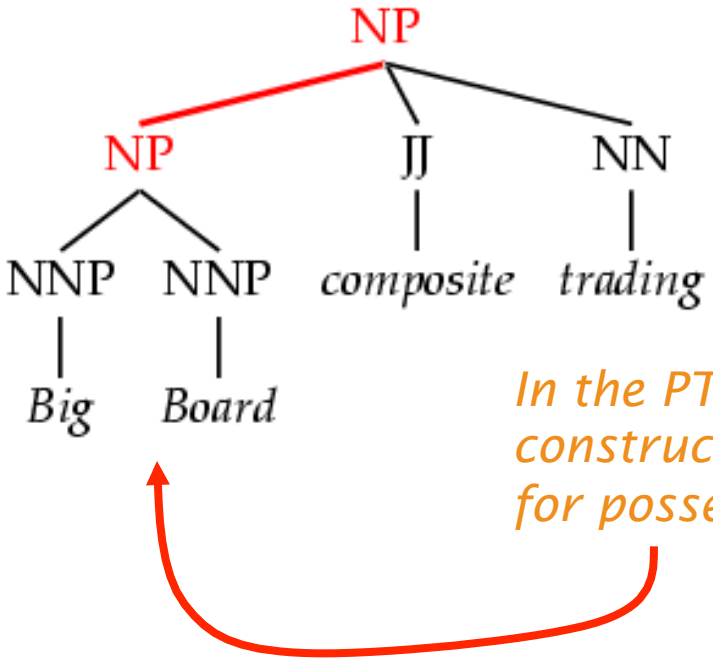
- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects)

Non-Independence II

- Symptoms of overly strong assumptions:
  - Rewrites get used where they don't belong



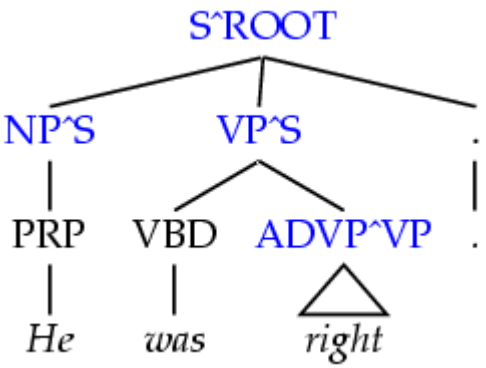




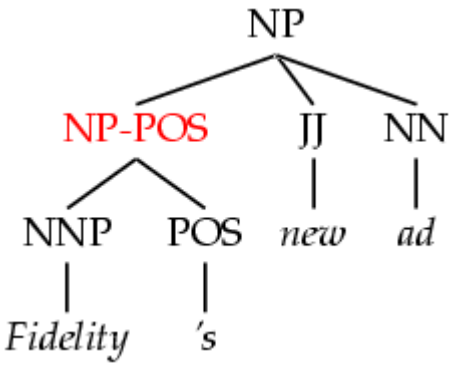
Refining the Grammar Symbols

- We can relax independence assumptions by encoding dependencies into the PCFG symbols, by state splitting:

Parent annotation
[Johnson 98]



Marking
possessive NPs



- Too much state-splitting ➔ sparseness (no smoothing used!)
- What are the most useful features to encode?

PCFG Independence Assumptions

Annotations

- Annotations split the grammar categories into sub-categories.
- Conditioning on history vs. annotating
  - P(NP^S → PRP) is a lot like P(NP → PRP | S)
  - P(NP-POS → NNP POS) isn't history conditioning.
- Feature grammars vs. annotation
  - Can think of a symbol like NP^NP-POS as NP [parent:NP, +POS]
- After parsing with an annotated grammar, the annotations are then stripped for evaluation.

The Return of Unlexicalized PCFGs

Accurate Unlexicalized Parsing

[Klein and Manning 1993]

- What do we mean by an "unlexicalized" PCFG?
  - Grammar rules are not systematically specified down to the level of lexical items
    - NP-stocks is not allowed
    - NP^S-CC is fine
  - Closed vs. open class words
    - Long tradition in linguistics of using function words as features or markers for selection (VB-have, SBAR-if/whether)
    - Different to the bilexical idea of semantic heads
    - Open-class selection is really a proxy for semantics
- Thesis
  - Most of what you need for accurate parsing, and much of what lexicalized PCFGs actually capture *isn't* lexical selection between content words but just basic grammatical features, like verb form, finiteness, presence of a verbal auxiliary, etc.

Experimental Approach

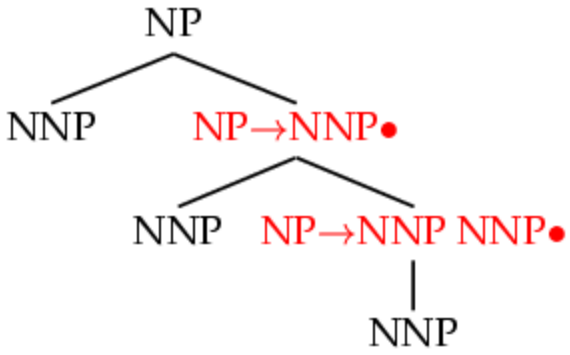
- Corpus: Penn Treebank, WSJ; iterate on small dev set

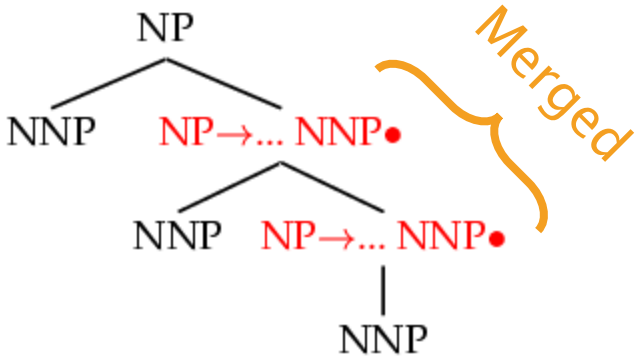
Training:	sections	02-21
Development:	section	22 (first 20 files) ←
Test:	section	23

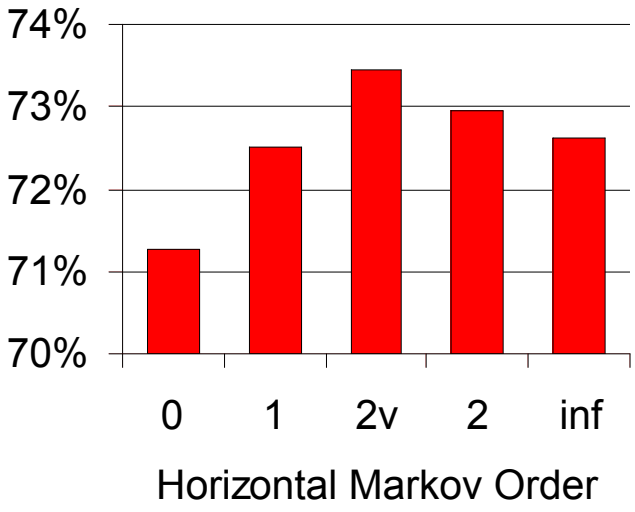
- Size – number of symbols in grammar.
  - Passive / complete symbols: NP, NP^S
  - Active / incomplete symbols: @NP_NP_CC [from binarization]
- We state-split as sparingly as possible
  - Highest accuracy with fewest symbols
  - Error-driven, manual hill-climb, one annotation at a time

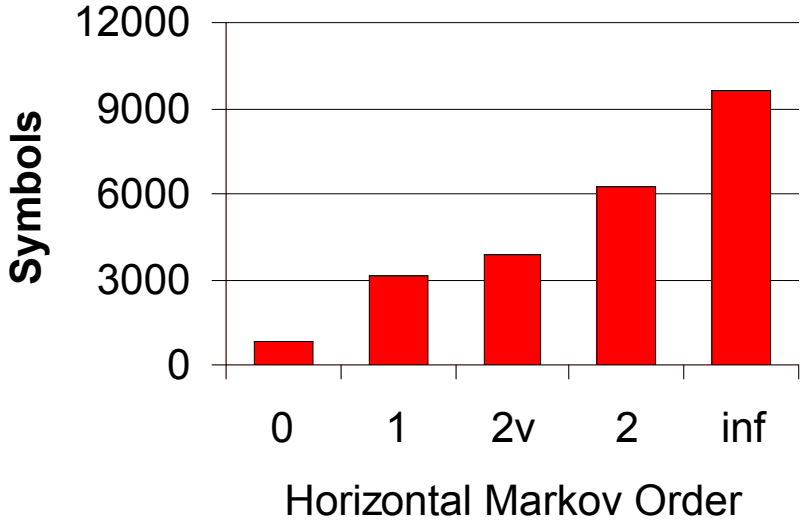
Horizontal Markovization

- Horizontal Markovization: Merges States







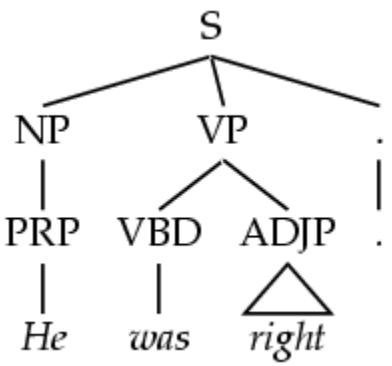


Vertical Markovization

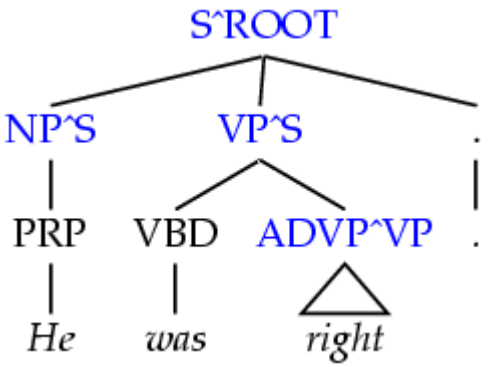
- Vertical Markov order: rewrites depend on past *k* ancestor nodes.

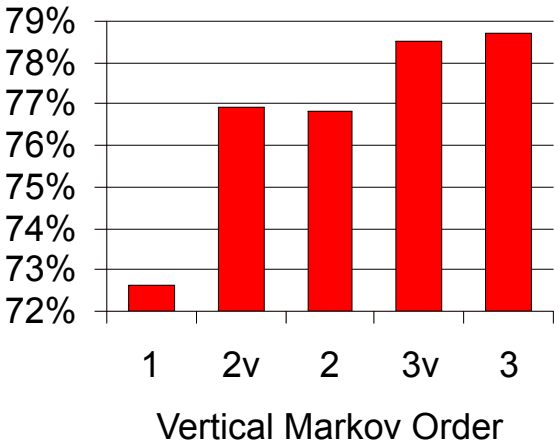
  (i.e., parent annotation)

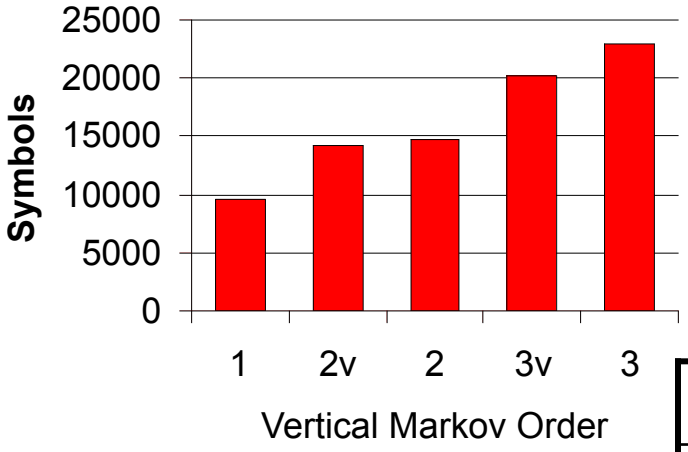
Order 1



Order 2



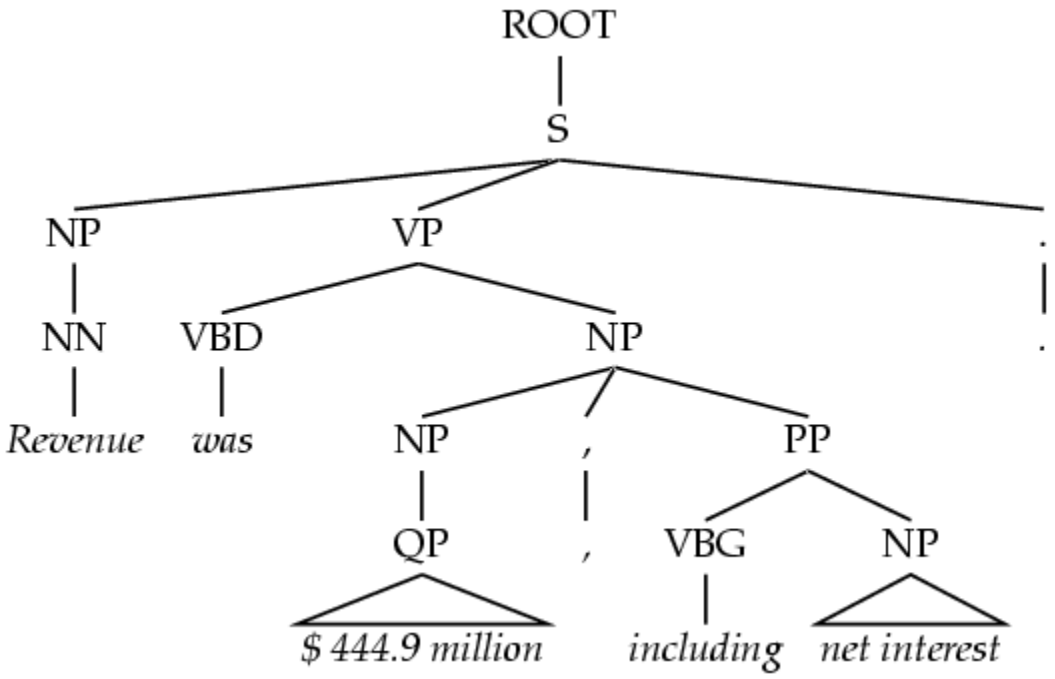




Model	F1	Size
v=h=2v	77.8	7.5K

Unary Splits

- Problem: unary rewrites are used to transmute categories so a high-probability rule can be used.

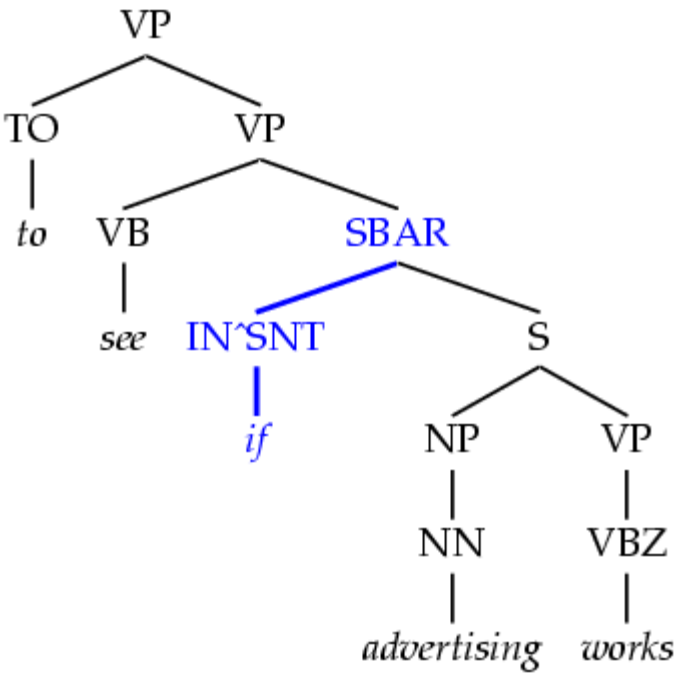


- Solution: Mark unary rewrite sites with -U

Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

Tag Splits

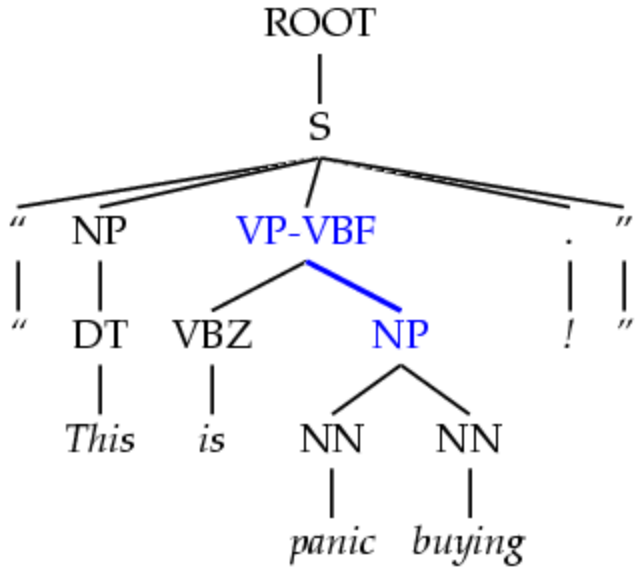
- Problem: Treebank tags are too coarse.
- Example: SBAR sentential complementizers (*that, whether, if)*, subordinating conjunctions (*while, after)*, and true prepositions (*in, of, to)* are all tagged IN.
- Partial Solution:
  - Subdivide the IN tag.



Annotation	F1	Size
Previous	78.3	8.0K
SPLIT-IN	80.3	8.1K

Yield Splits

- Problem: sometimes the behavior of a category depends on something inside its future yield.
- Examples:
  - Possessive NPs
  - Finite vs. infinite VPs
  - Lexical heads!
- Solution: annotate future elements into nodes.

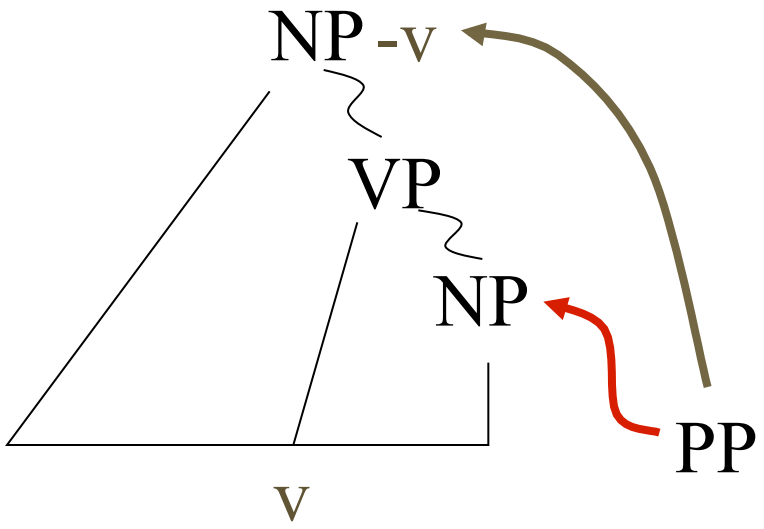


Annotation	F1	Size
tag splits	82.3	9.7K
POSS-NP	83.1	9.8K
SPLIT-VP	85.7	10.5K

Distance / Recursion Splits

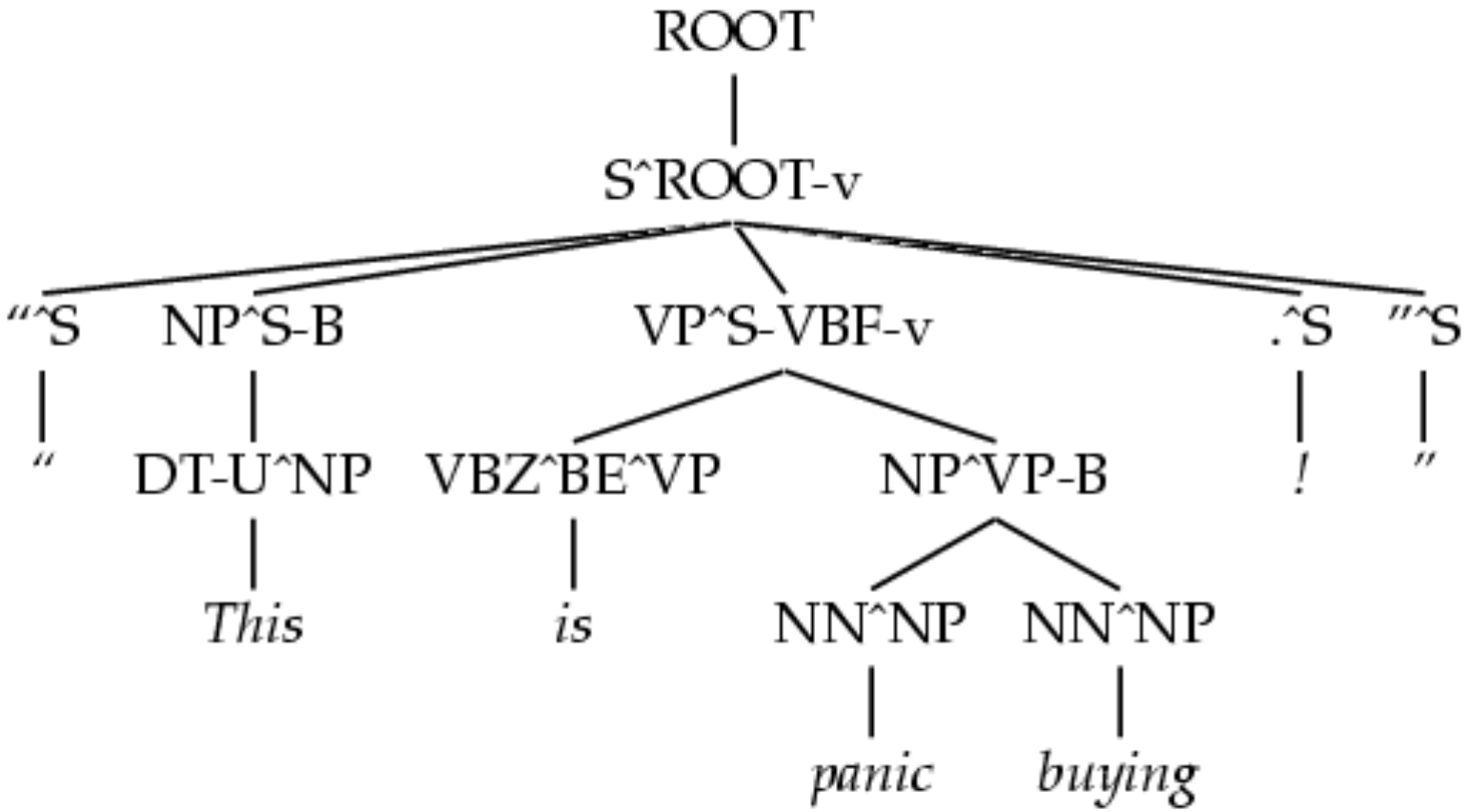
- Problem: vanilla PCFGs cannot distinguish attachment heights.

- Solution: mark a property of higher or lower sites:
  - Contains a verb.
  - Is (non)-recursive.
    - Base NPs [cf. Collins 99]
    - Right-recursive NPs



Annotation	F1	Size
Previous	85.7	10.5K
BASE-NP	86.0	11.7K
DOMINATES-V	86.9	14.1K
RIGHT-REC-NP	87.0	15.2K

A Fully Annotated Tree



Final Test Set Results

Parser	LP	LR	**F1**
Magerman 95	84.9	84.6	**84.7**
Collins 96	86.3	85.8	**86.0**
Klein & Manning 03	86.9	85.7	**86.3**
Charniak 97	87.4	87.5	**87.4**
Collins 99	88.7	88.6	**88.6**

- Beats "first generation" lexicalized parsers

The Return of Unlexicalized PCFGs

Latent Variable PCFGs

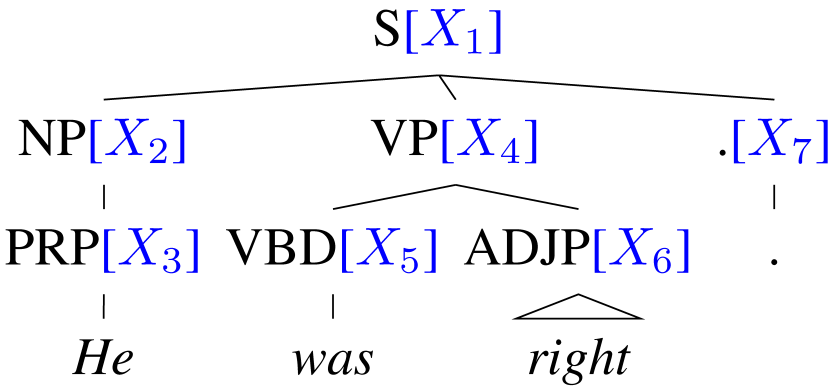
Extending the idea to induced syntactico-semantic classes

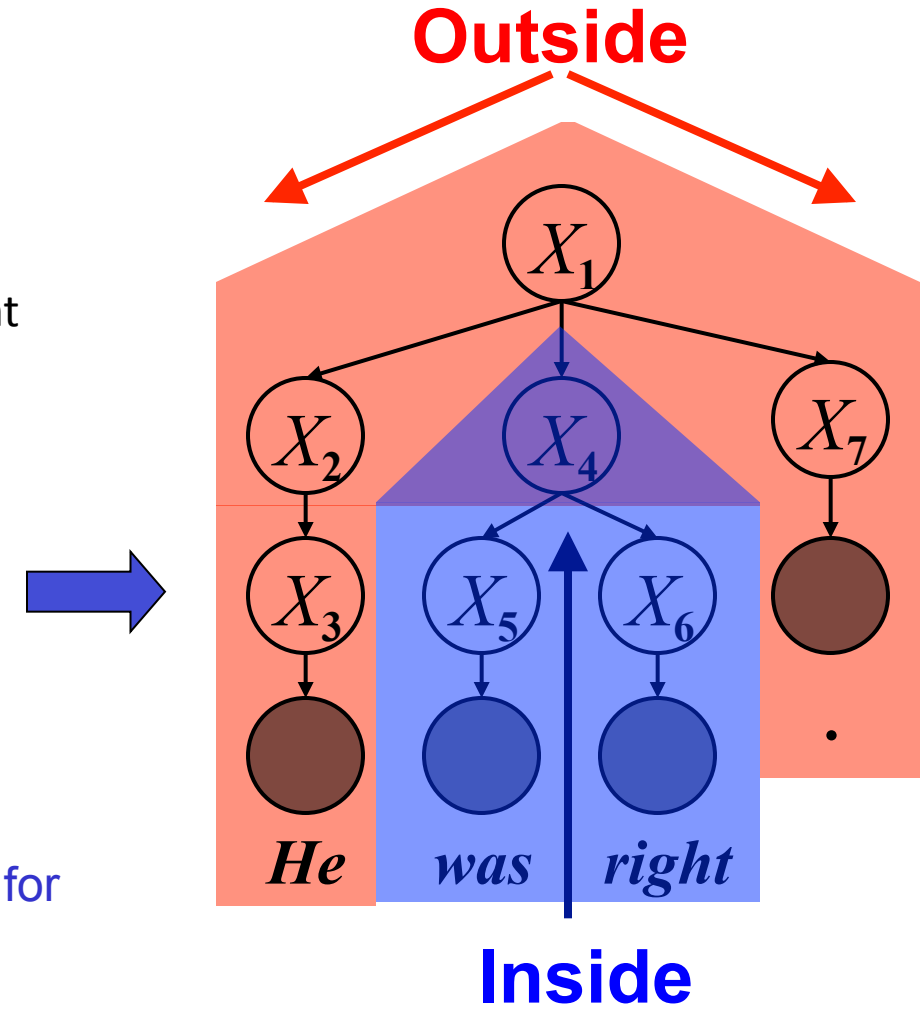
Learning Latent Annotations

[Petrov and Klein 2006, 2007]

Can you automatically find good symbols?

- Brackets are known
- Base categories are known
- Induce subcategories
- Clever split/merge category refinement





EM algorithm, like Forward-Backward for HMMs, but constrained by tree

POS tag splits' commonest words: effectively a semantic class-based model

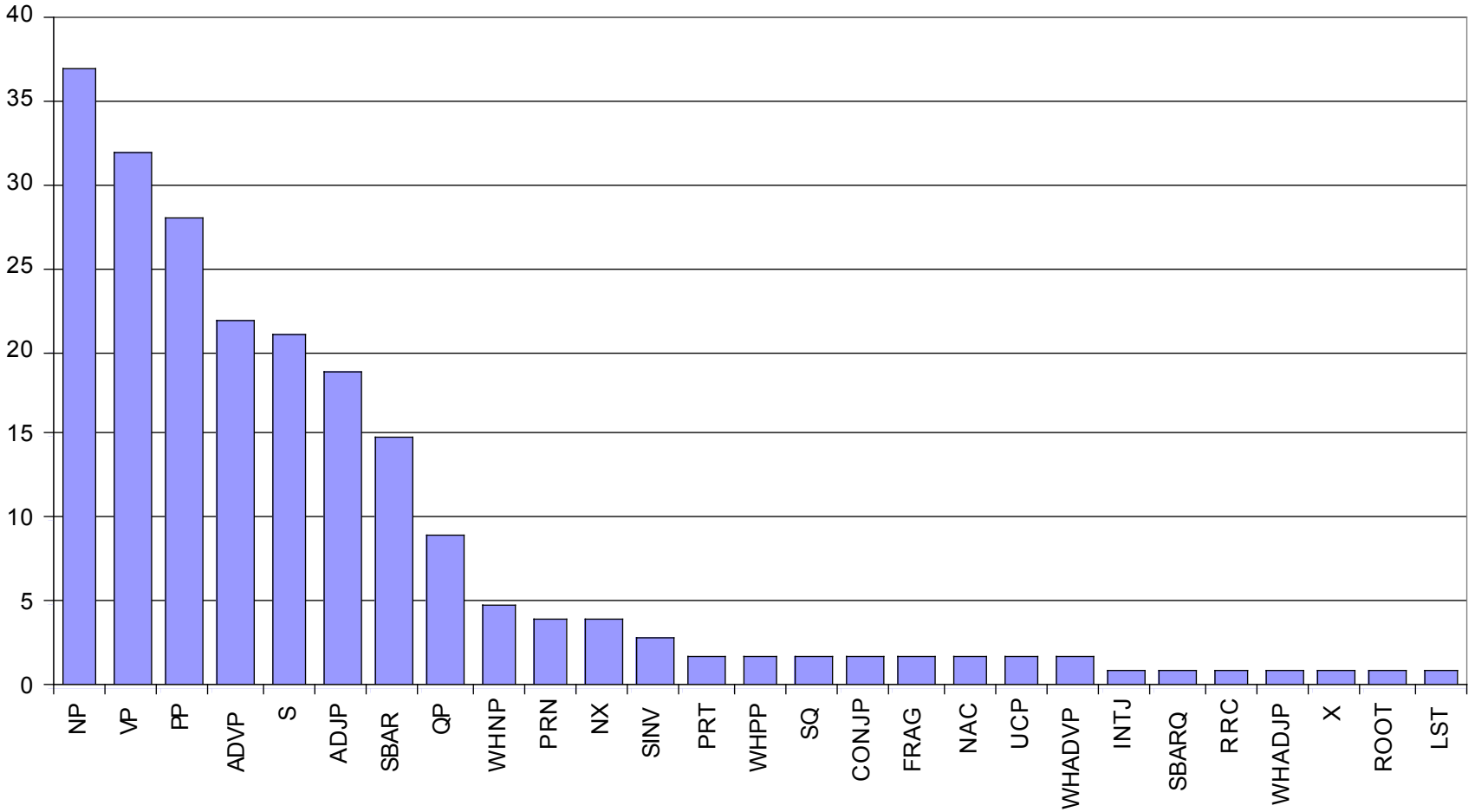
- Proper Nouns (NNP):

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

Number of phrasal subcategories



The Latest Parsing Results... (English PTB3 WSJ train 2-21, test 23)

Parser	*F1 ≤ 40 words*	*F1 all words*
Klein & Manning unlexicalized 2003	86.3	85.7
Matsuzaki et al. simple EM latent states 2005	86.7	86.1
Charniak generative, lexicalized ("maxent inspired") 2000	90.1	89.5
Petrov and Klein NAACL 2007	90.6	90.1
Charniak & Johnson discriminative reranker 2005	92.0	91.4
Fossum & Knight 2009 combining constituent parsers		**92.4**

Latent Variable PCFGs

Extending the idea to induced syntactico-semantic classes