CMP462: Natural Language Processing

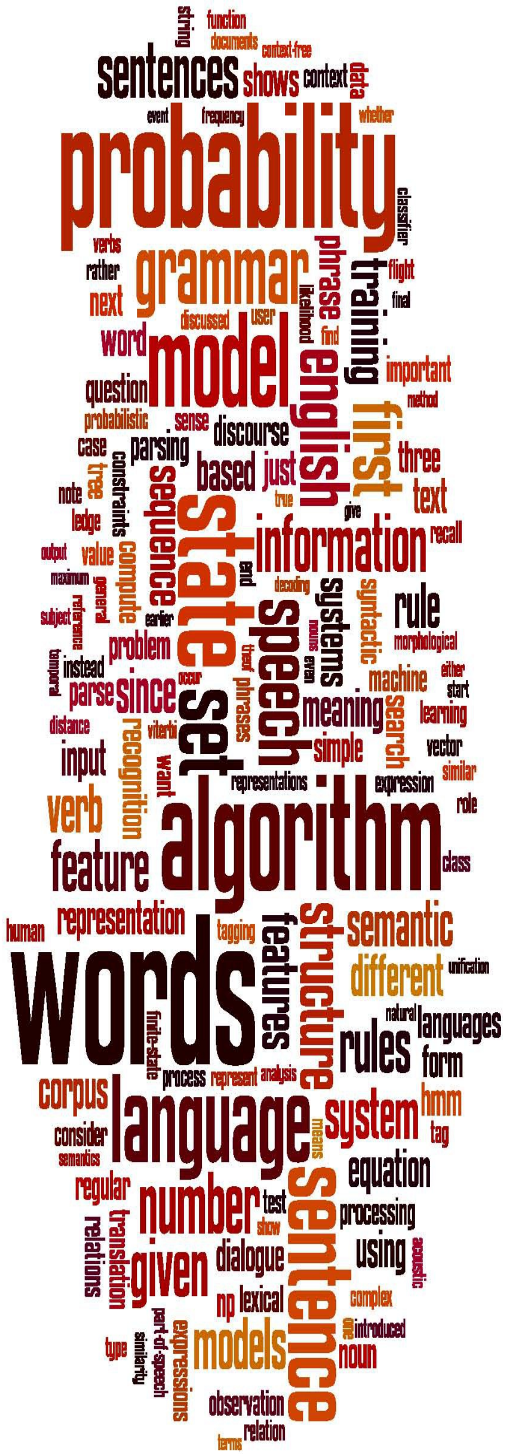
Lecture 10: Lexicalized Parsers

Mohamed Alaa El-Dien Aly
Computer Engineering Department
Cairo University
Spring 2013

Agenda

- Lexicalized Parsers
- Independence in PCFGs
- Unlexicalized Parsers

Acknowledgment:
Most slides adapted from Chris Manning and Dan Jurafsky's NLP class on Coursera.



Lexicalization of PCFGs

Introduction

Christopher Manning

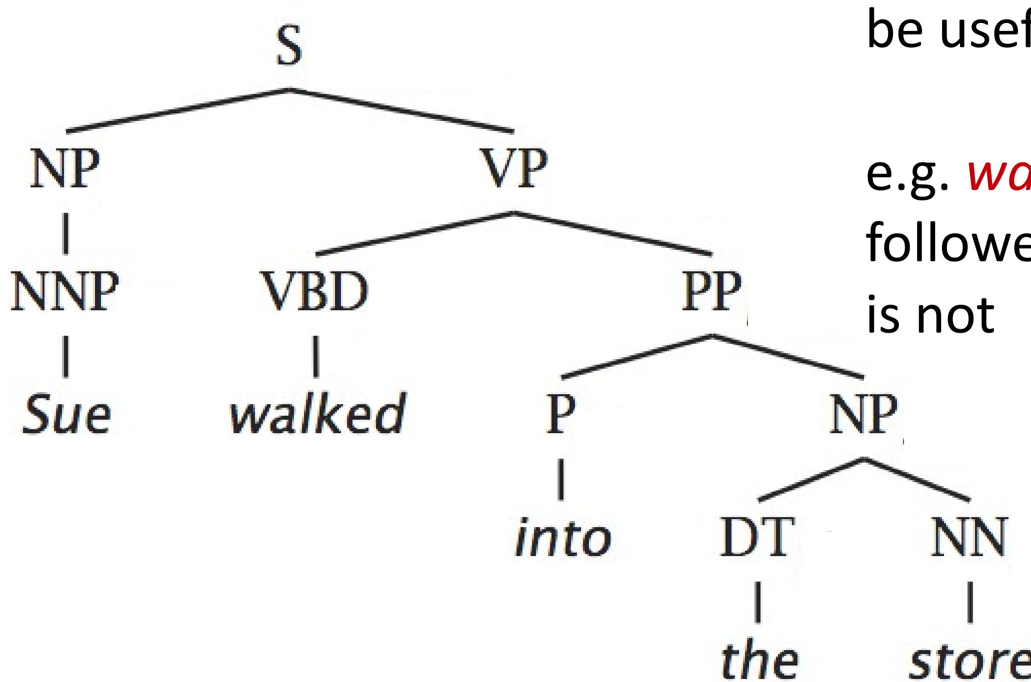
(Head) Lexicalization of PCFGs

[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

These rules do not mention the actual words, which might be useful

e.g. *walked* is likely to be followed by a PP, while *saw* is not



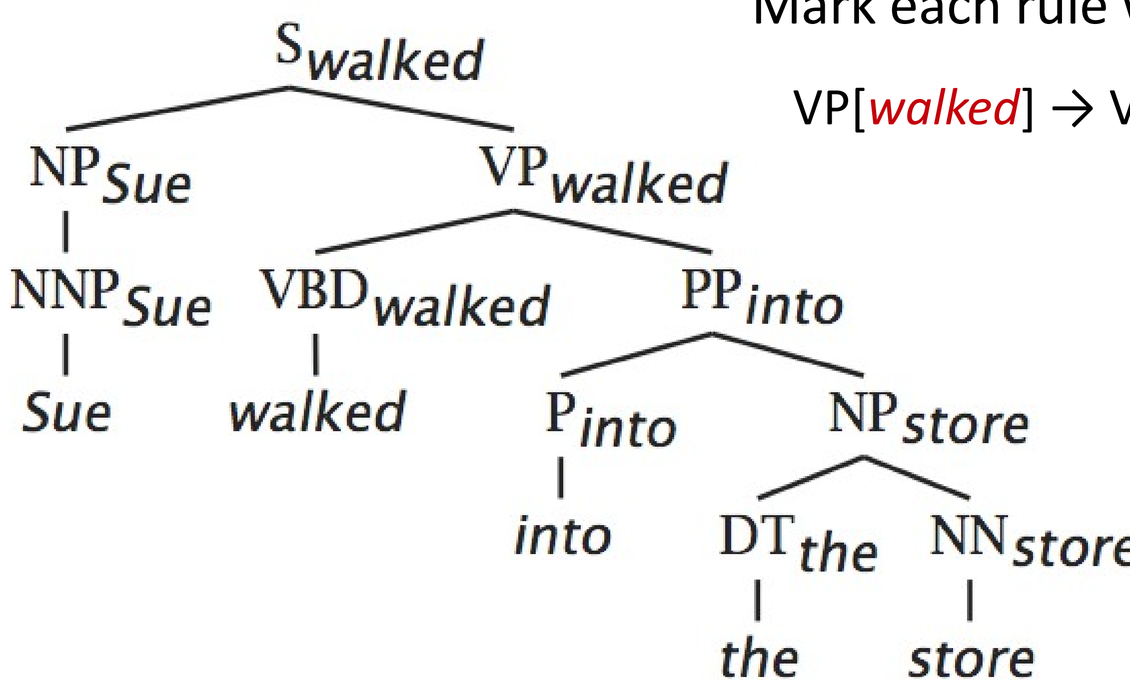
(Head) Lexicalization of PCFGs

[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
- Captures more information from the language into the grammar

Mark each rule with its head word

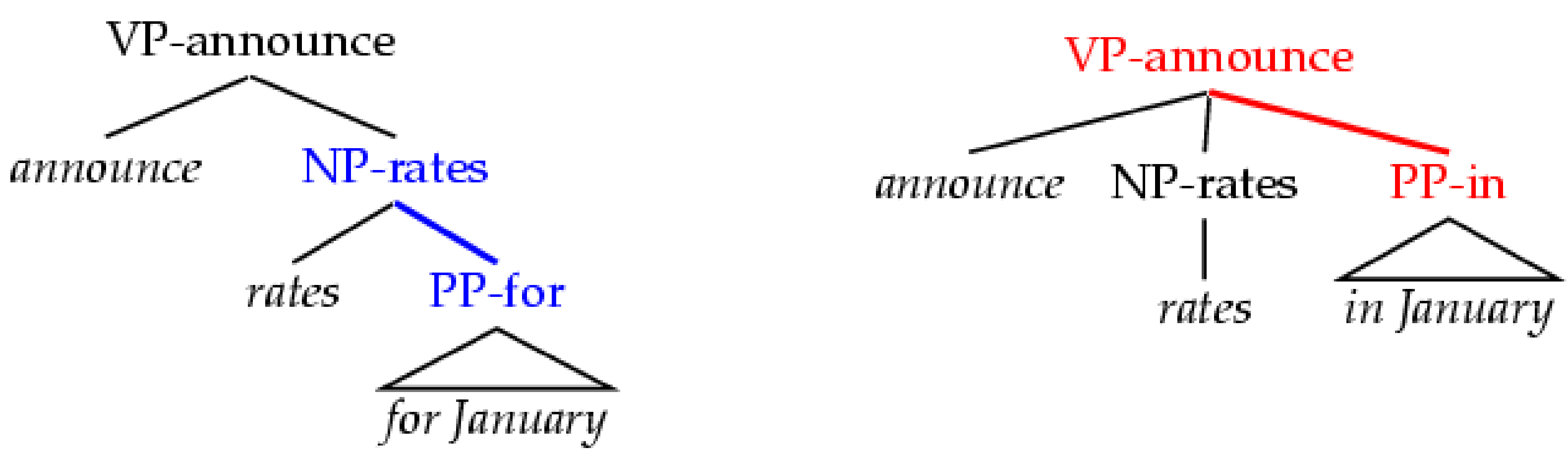
VP[*walked*] → VBD[*walked*] PP[*into*]



(Head) Lexicalization of PCFGs

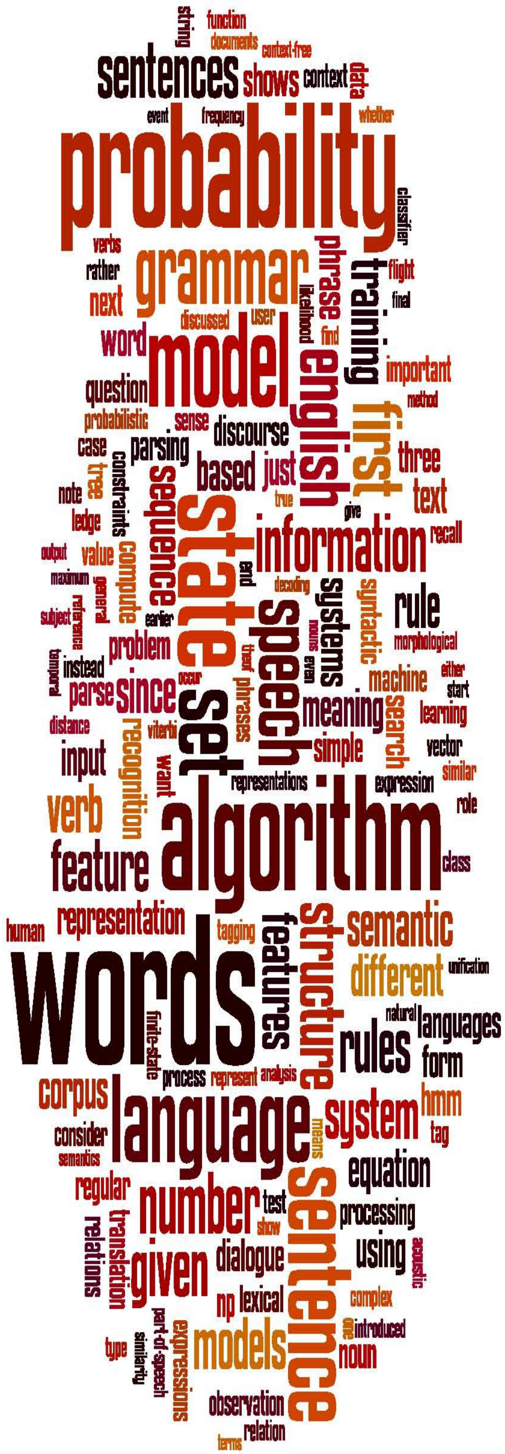
[Magerman 1995, Collins 1997; Charniak 1997]

- Word-to-word affinities are useful for certain ambiguities
  - PP attachment is now (partly) captured in a local PCFG rule.



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:
  - $p(\mathrm{VP} \rightarrow \mathrm{V\ NP\ NP}) = 0.00151$
  - $p(\mathrm{VP} \rightarrow \mathrm{V\ NP\ NP} \mid \mathrm{said}) = 0.00001$
  - $p(\mathrm{VP} \rightarrow \mathrm{V\ NP\ NP} \mid \mathrm{gave}) = 0.01980$ "
- Michael Collins, 2003 COLT tutorial: "Lexicalized Probabilistic Context-Free Grammars … perform vastly better than PCFGs (88% vs. 73% accuracy)"



Lexicalization of PCFGs

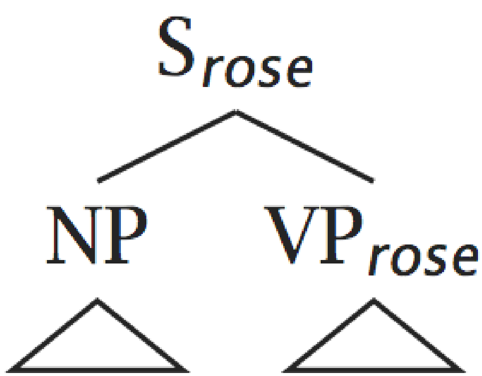
The model of Charniak (1997)

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
- Uses two probability distributions:
  - Probability of headwords
  - Probability of a rule

Charniak (1997) example

corporate profits rose



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

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

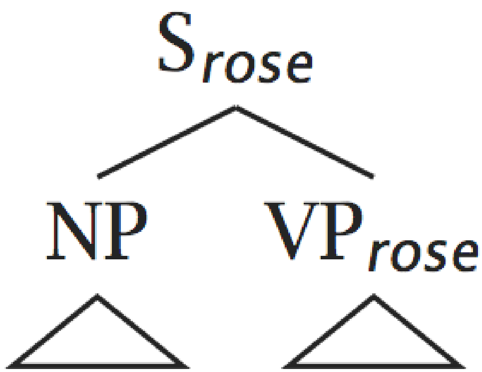
c. $P(h \mid ph, c, pc)$ Head word prob.

d. $P(r \mid h, c, pc)$ rule prob.

Find the most probable head word given parent head word (*rose*), current category (*NP*), and parent category (*S*)

Charniak (1997) example

corporate profits rose

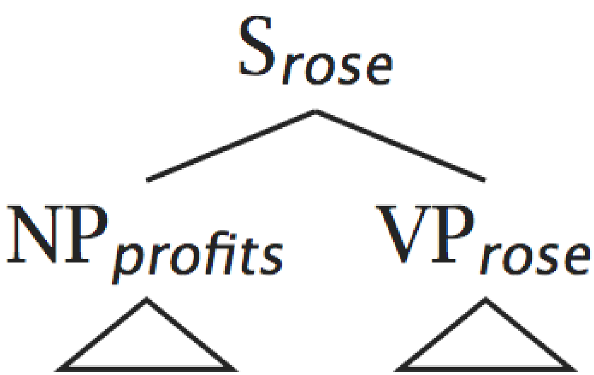


a. $h = profits;\ c = \text{NP}$

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

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

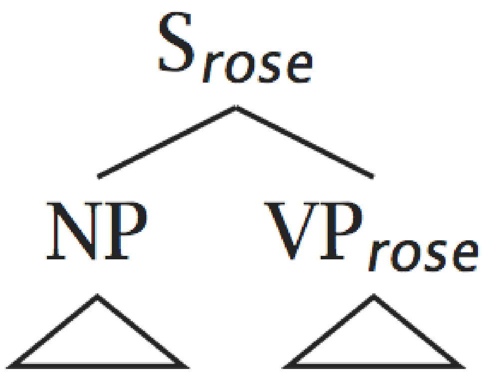
d. $P(r|h, c, pc)$



Find the best rule to expand NP given the current head word (*profits*), current category (*NP*), and parent category (*S*)

Charniak (1997) example

corporate profits rose

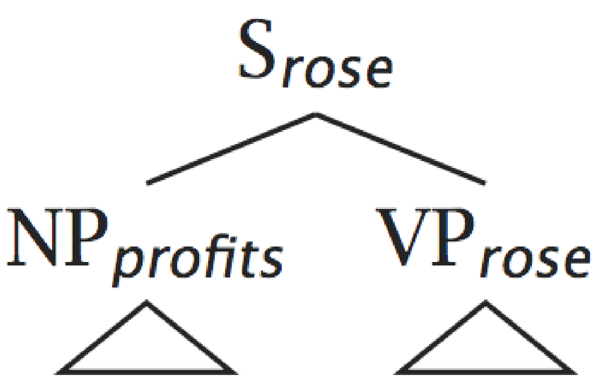


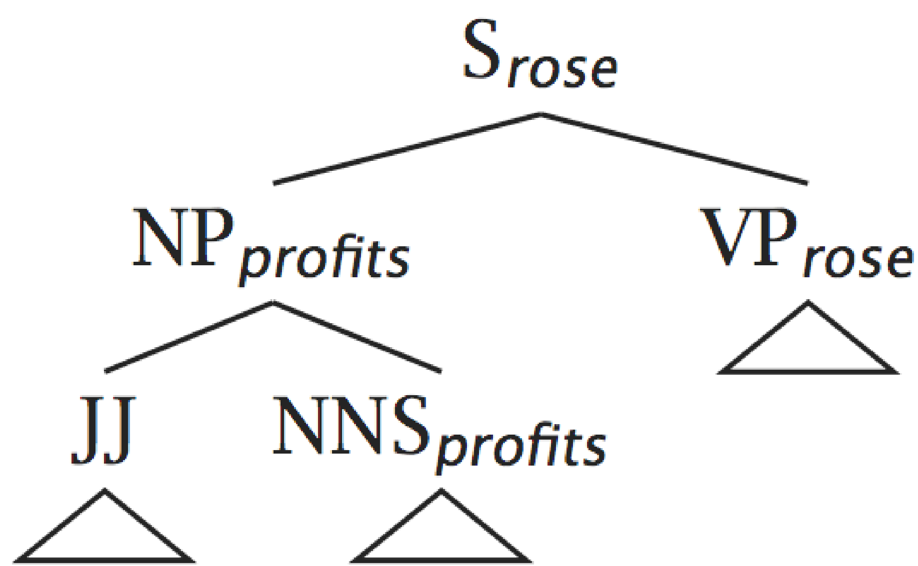
a. $h = profits$; $c = \mathrm{NP}$

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

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

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

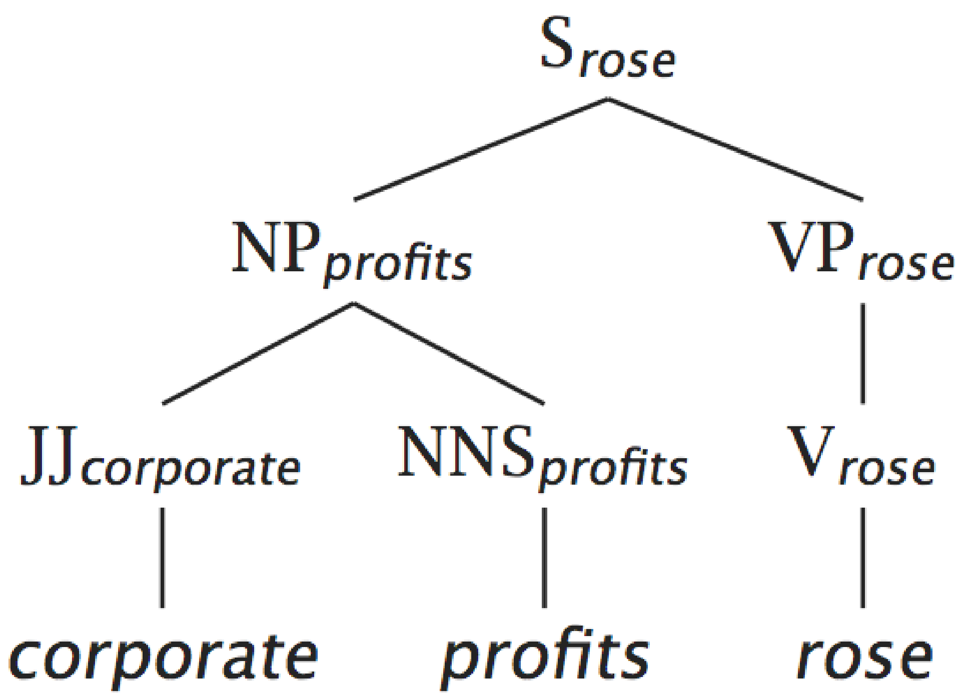




Repeat!

Charniak (1997) example

corporate profits rose



Lexicalization models argument selection by sharpening rule expansion probabilities

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

Frequencies of different rules and head 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%

Lexicalization sharpens probabilities: Predicting heads

Having more "context" sharpens the 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

Can we actually estimate all these probabilities?

Charniak (1997) linear interpolation/shrinkage

we have all information

leave out parent headword & use its category

leave out parent headword

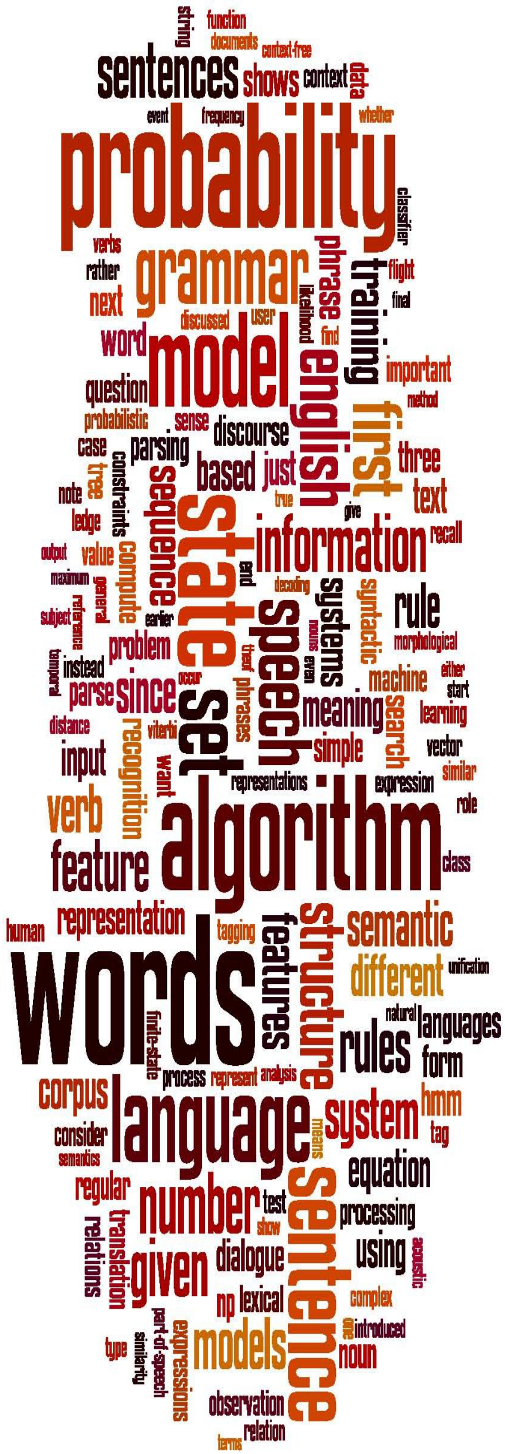
$$\begin{aligned}\hat{P}(h|ph,c,pc) \;=\; & \lambda_1(e)P_{\mathrm{MLE}}(h|ph,c,pc) \\ & +\lambda_2(e)P_{\mathrm{MLE}}(h|C(ph),c,pc) \\ & +\lambda_3(e)P_{\mathrm{MLE}}(h|c,pc)+\lambda_4(e)P_{\mathrm{MLE}}(h|c)\end{aligned}$$

- $\lambda_i(e)$ 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

	$P(\text{prft}\mid\text{rose},\text{NP},\text{S})$	$P(\text{corp}\mid\text{prft},\text{JJ},\text{NP})$
$P(h\mid ph,c,pc)$	$\boxed{0}$	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.



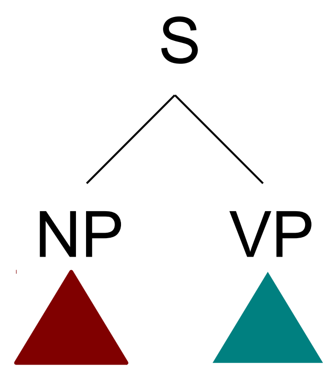
PCFG Independence Assumptions

PCFGs and Independence

- The symbols in a PCFG define independence assumptions:

$S \rightarrow NP\ VP$

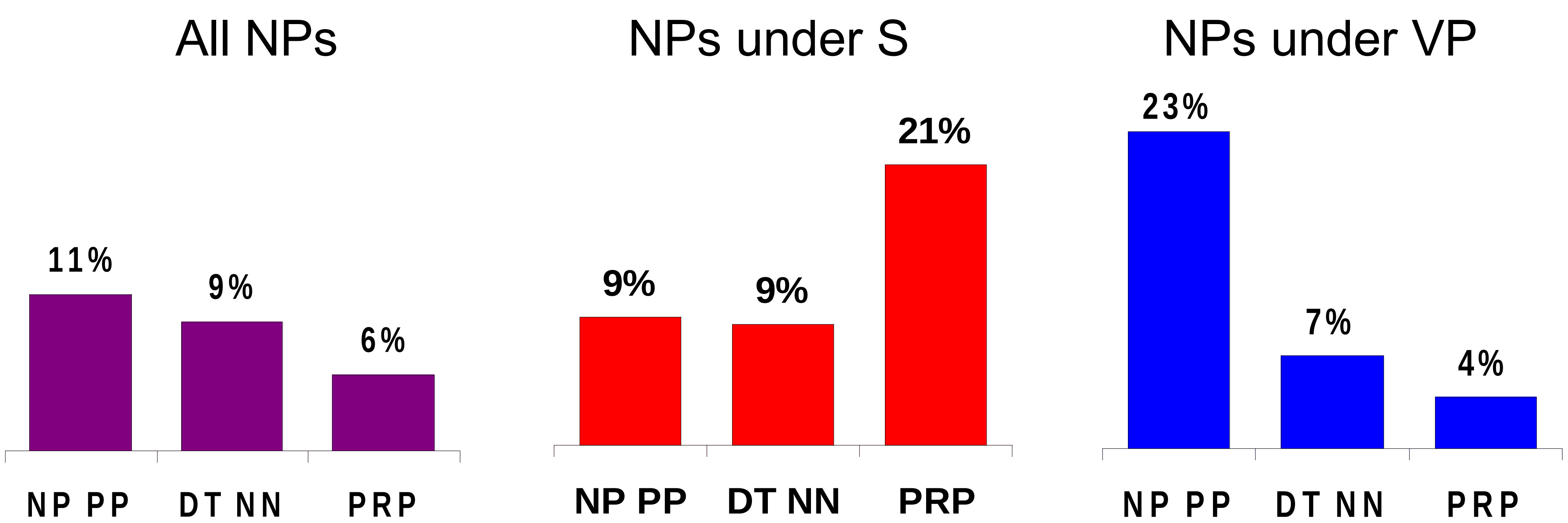
$NP \rightarrow DT\ NN$



  - 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
  - We can parse any subtree independently of any other part of the tree

Non-Independence I

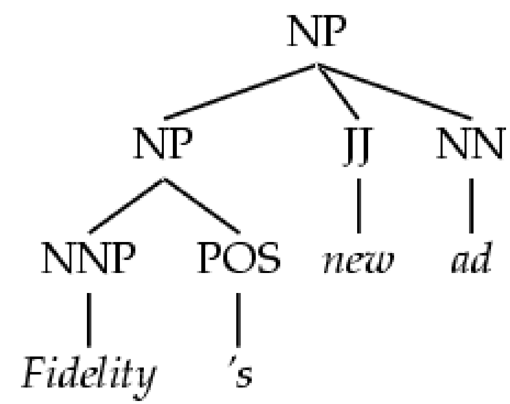
- The independence assumptions of a PCFG are often too strong



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

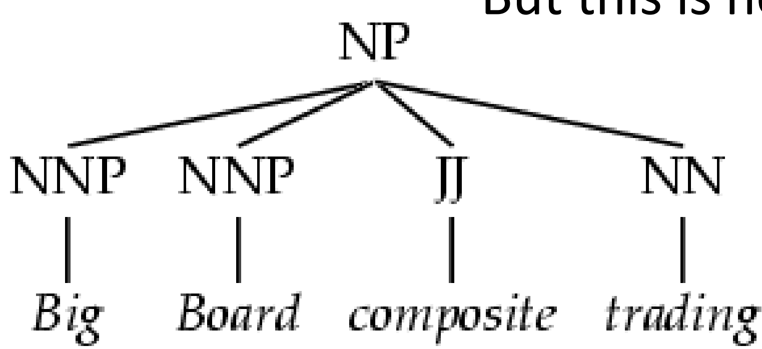
Non-Independence II

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

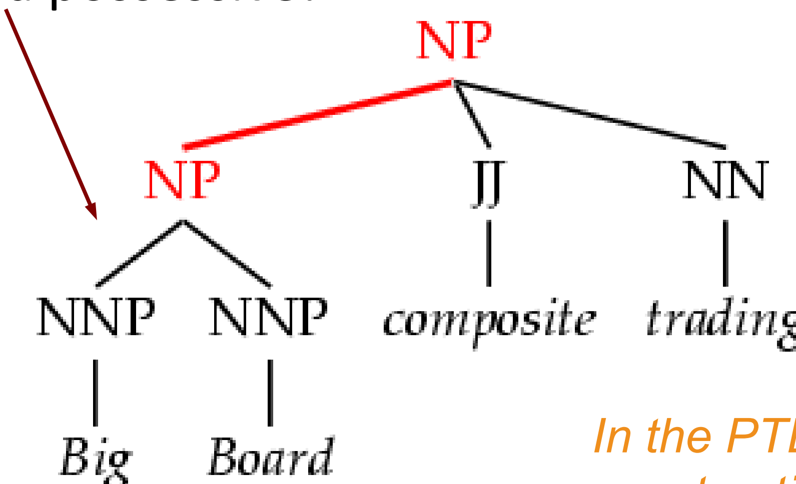


Because the of expansion NP → NNP NNP is independent of NP → NP JJ NN

But this is not a possessive!



Correct parse



In the PTB, this construction is for possessives

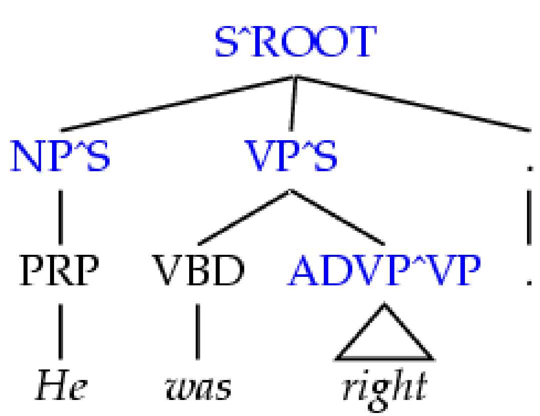
Why didn't it learn the correct one?

Incorrect parse

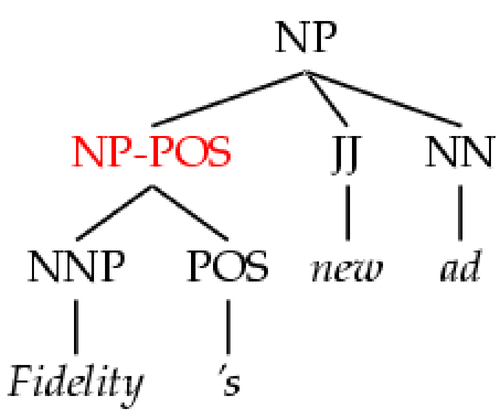
Refining the Grammar Symbols

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

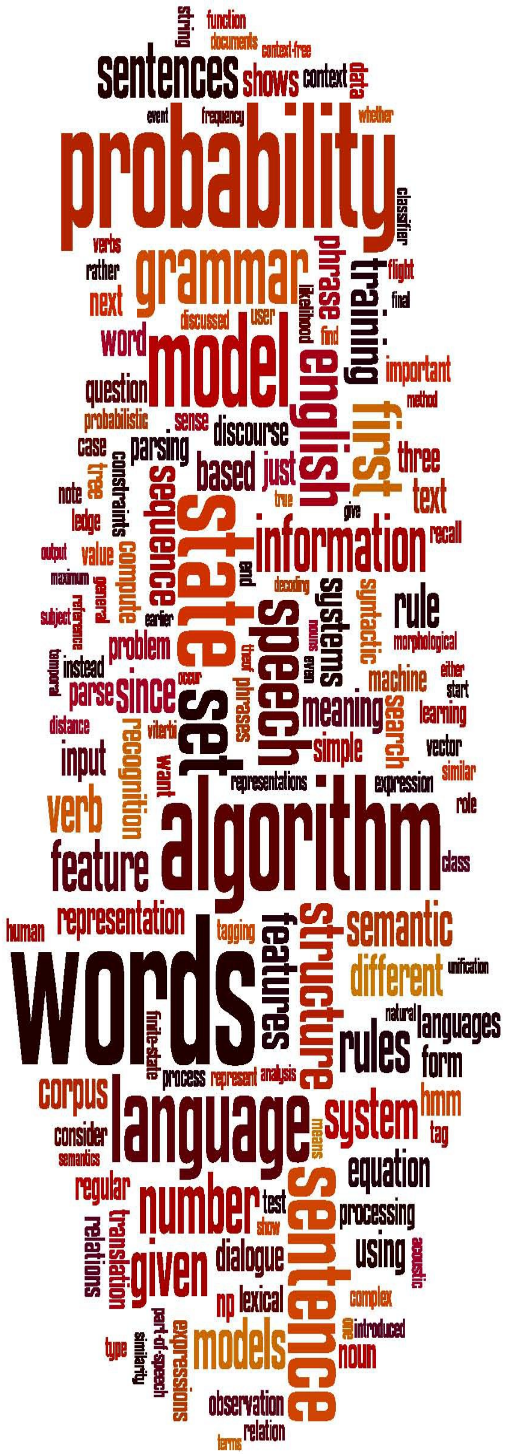
Parent annotation [Johnson 98]



Marking possessive NPs



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



The Return of Unlexicalized PCFGs

Accurate Unlexicalized Parsing

[Klein and Manning 2003]

- 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)
    - 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

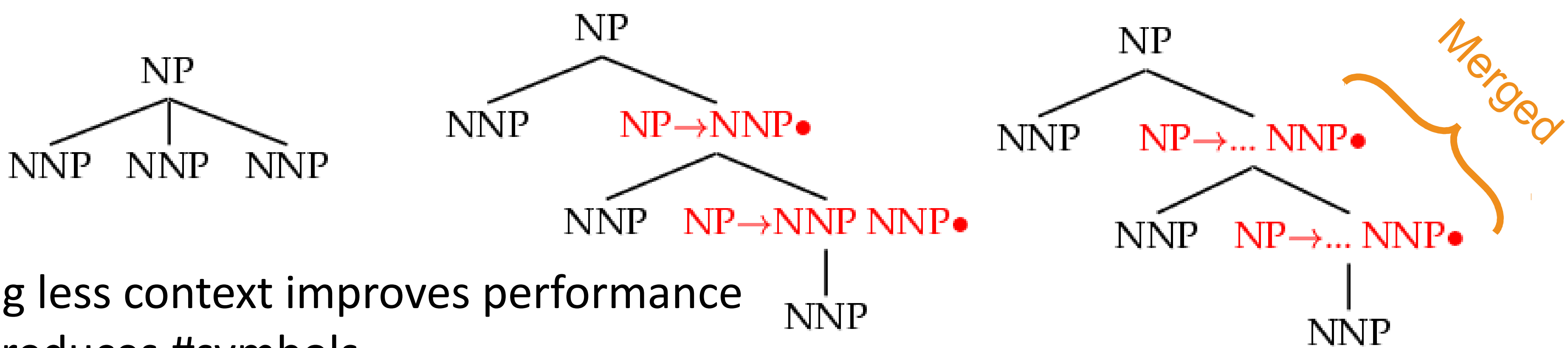
Training:	sections	02-21
Development:	section	22 (first 20 files) ←
Test:	section	23

- Performance – P/R/F1
- 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

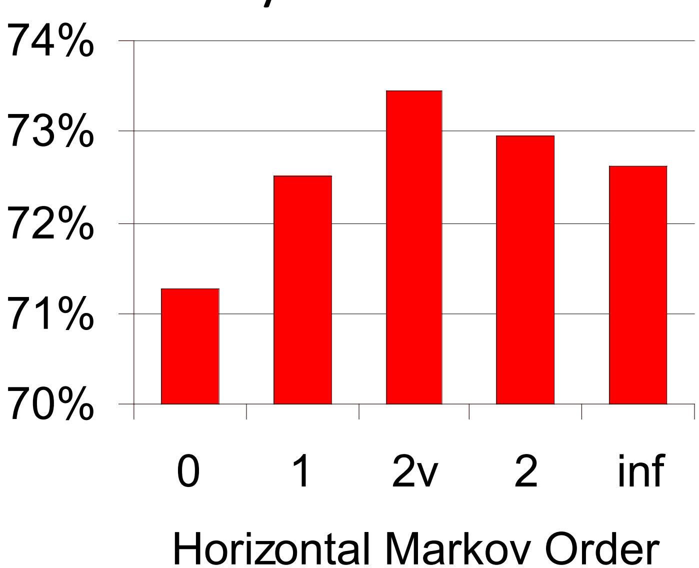
Horizontal Markovization

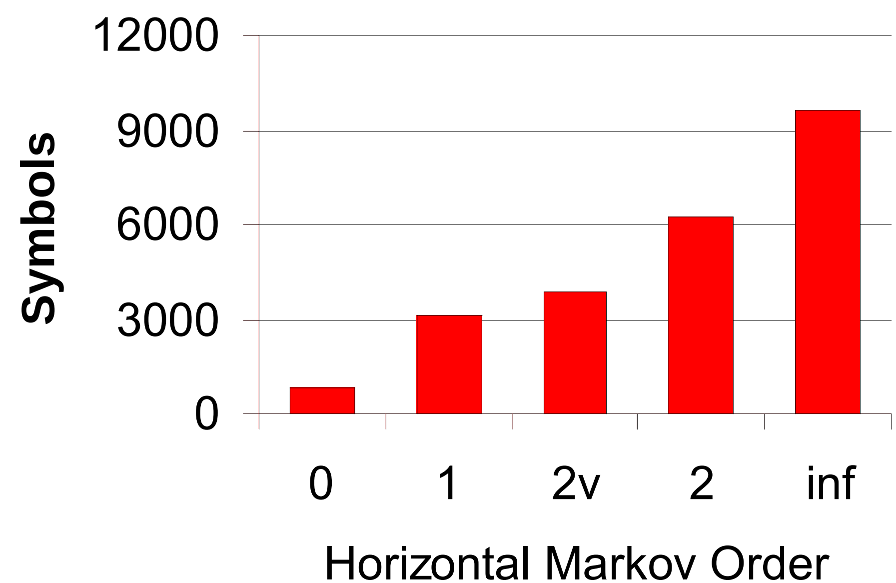
- Horizontal Markovization: Merges States

Condition on fixed amount of context/history



Using less context improves performance and reduces #symbols

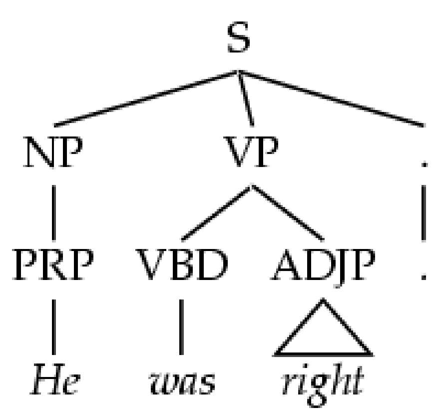




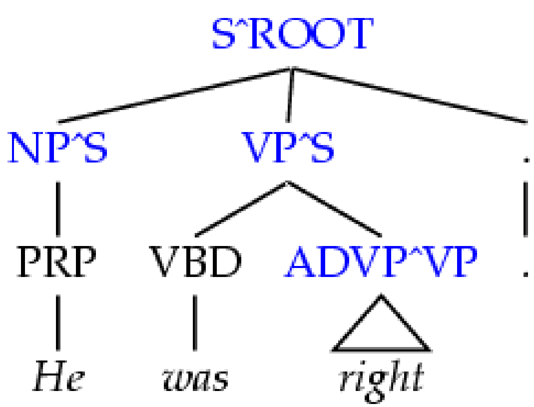
Vertical Markovization

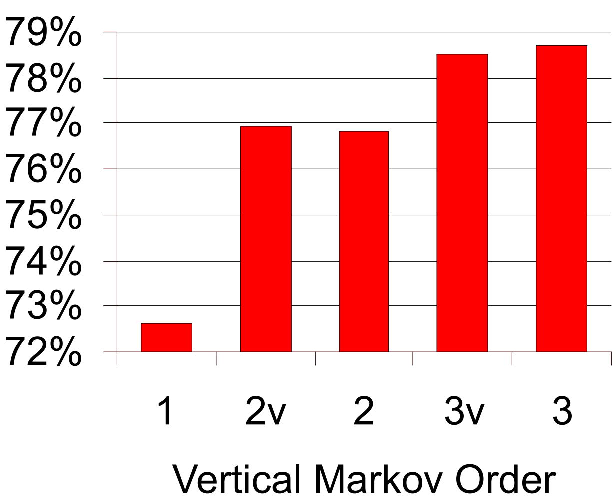
- Vertical Markov order: rewrites depend on past *k* ancestor nodes.
  (i.e., parent annotation)

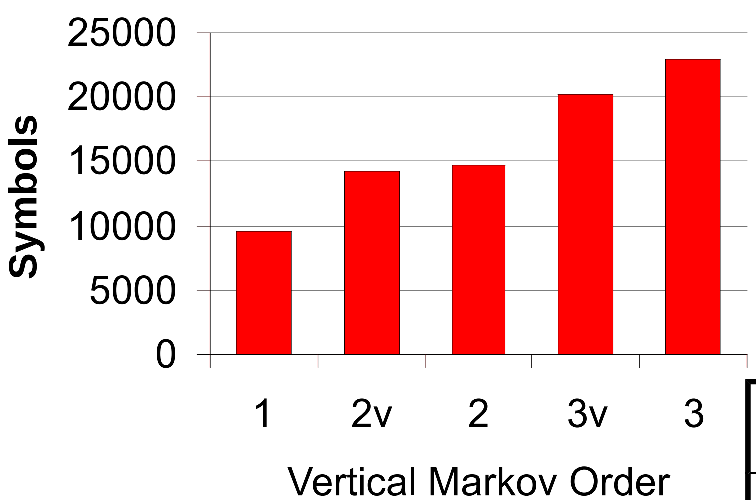
Order 1



Order 2



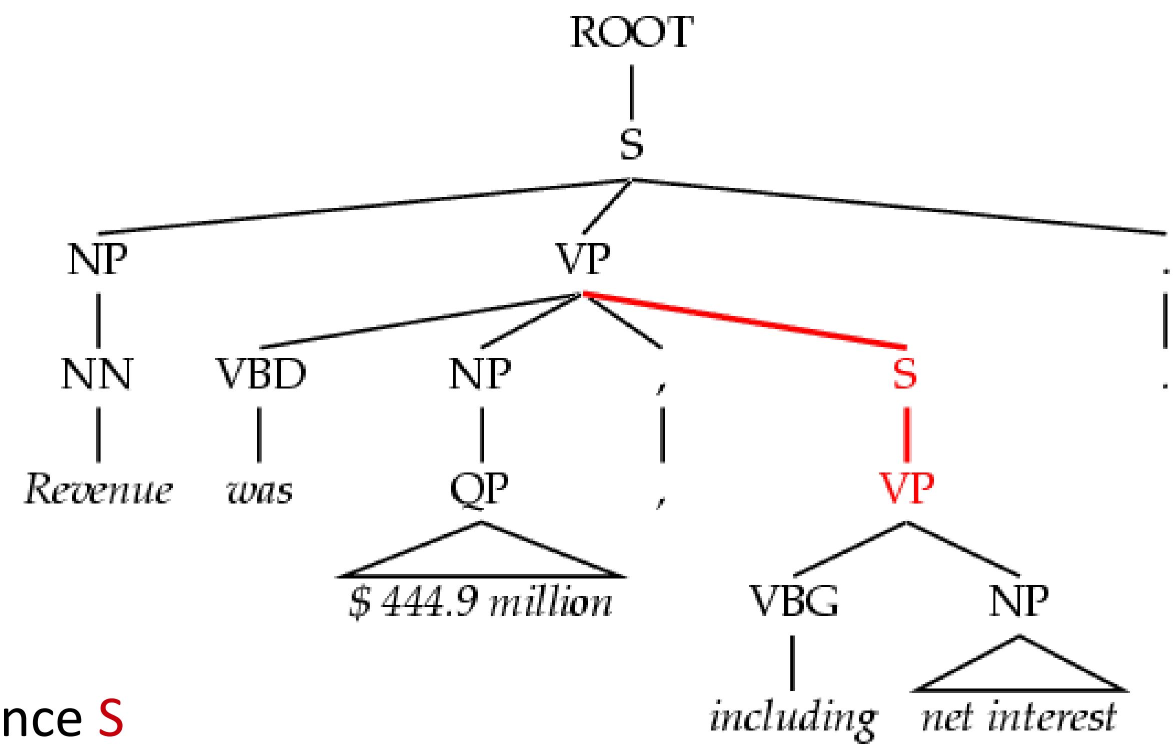




Model	F1	Size
v=h=2v	77.8	7.5K

Unary Splits

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

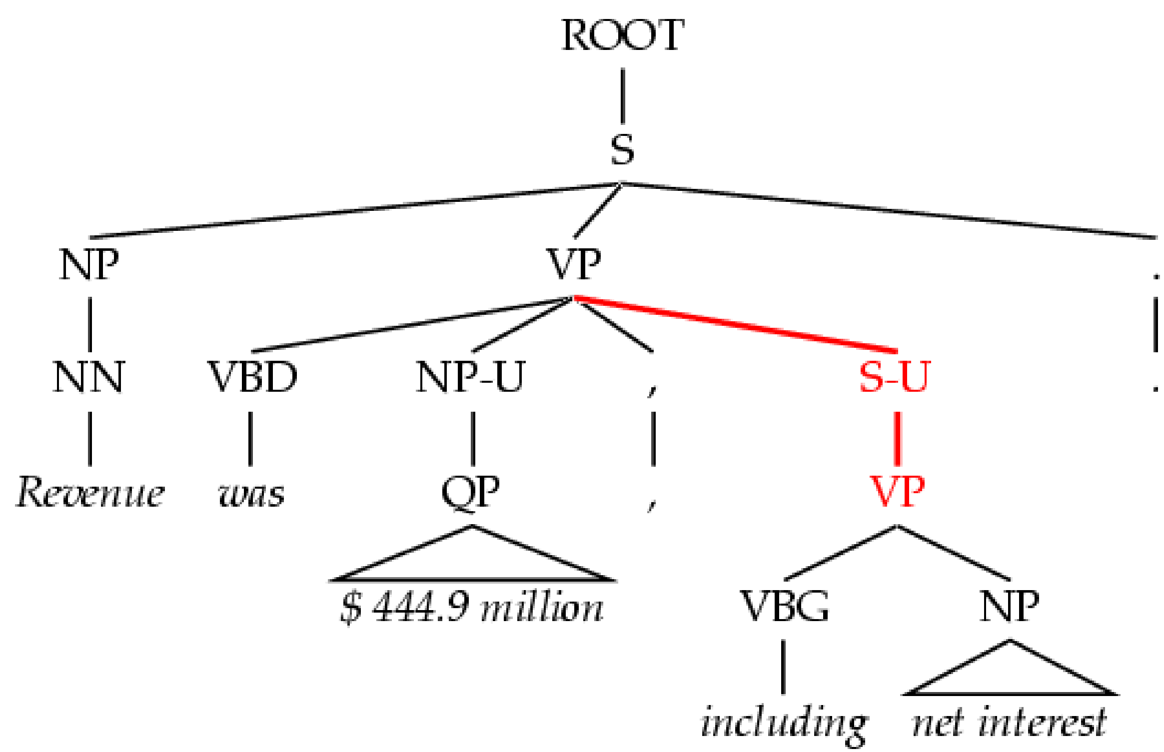


Here we are expecting a sentence S and not just a VP, but the rule S → VP has high probability

Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

Unary Splits

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

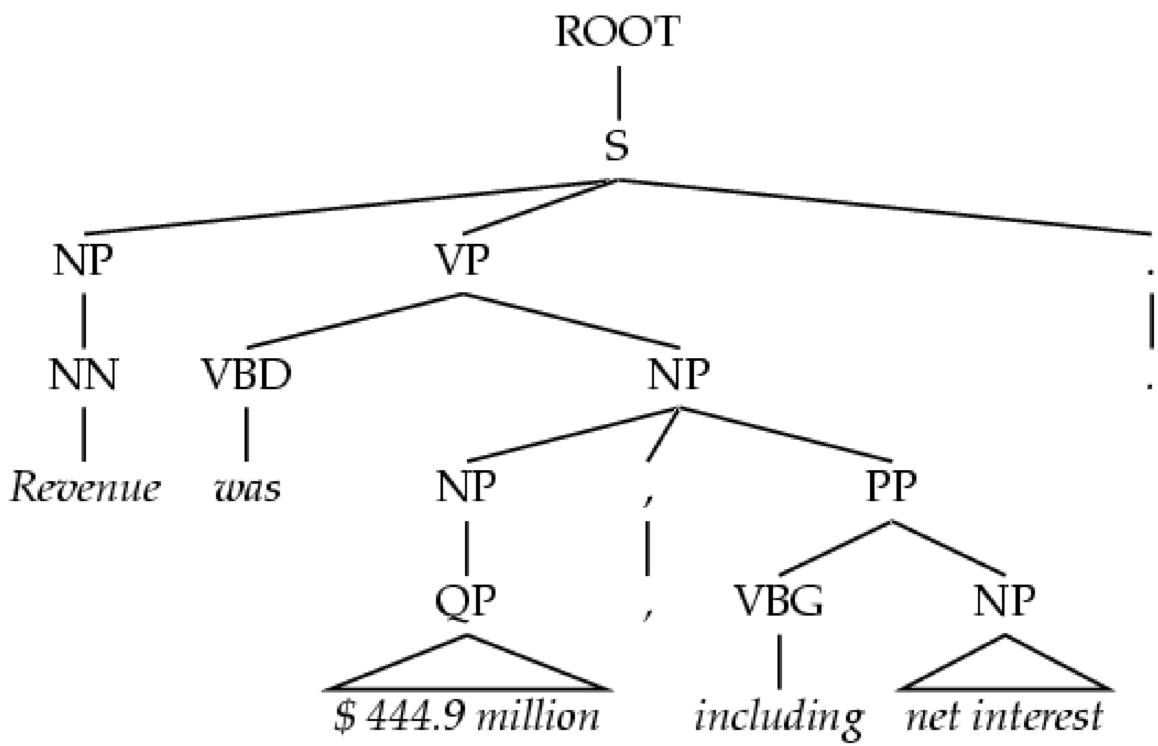


- Solution: Mark unary rewrite sites with -U in the training data

Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

Unary Splits

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

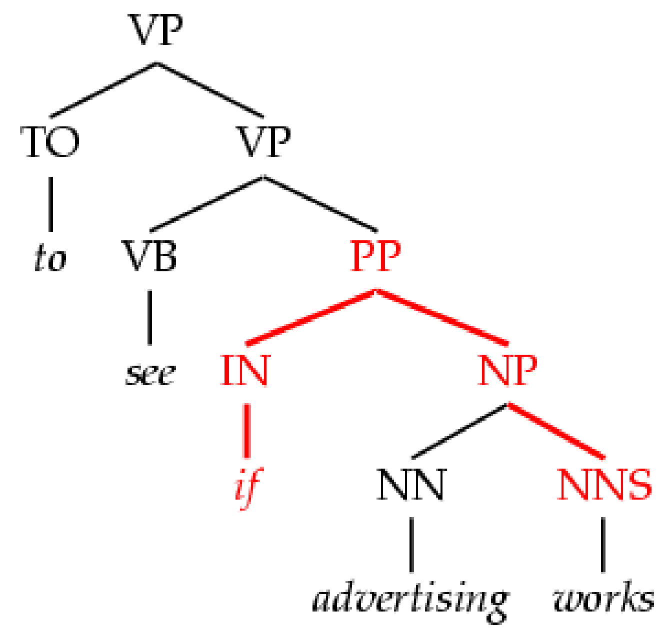


- Solution: Mark unary rewrite sites with -U in the training data

Annotation	F1	Size
Base	77.8	7.5K
UNARY	78.3	8.0K

Tag Splits

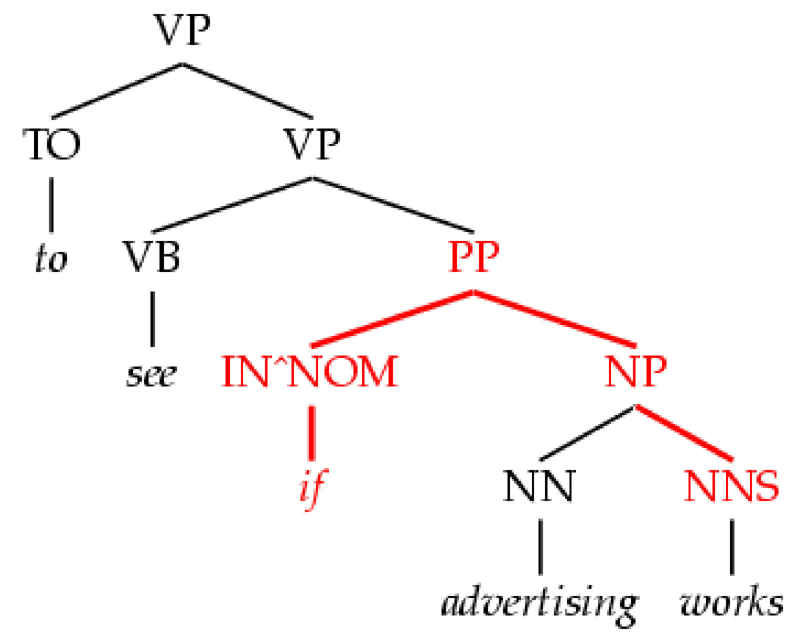
- Problem: Treebank tags are too coarse.
- Example: SBAR sentential complementizers (*that, whether, if)*, subordinating conjunctions (*while, after)*, and true prepositions (*in, of, to)* are all tagged IN.
- Partial Solution:
  - Subdivide the IN tag.



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

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.



Add tag IN^NOM that expects a NP as its complement, and learn that *if* is not an example of this

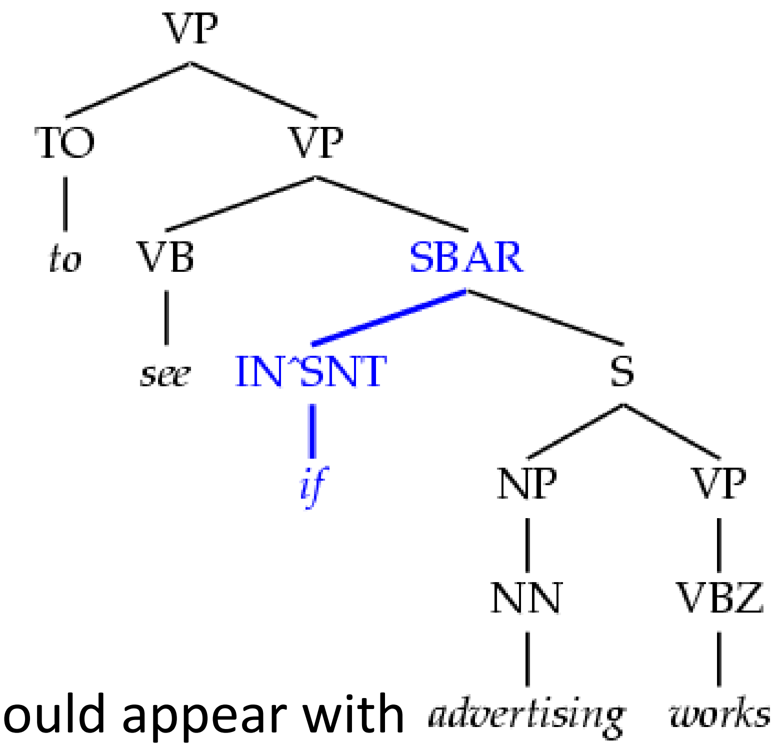
- Partial Solution:
  - Subdivide the IN tag.

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

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.



if actually should appear with with a sentence complement

- Partial Solution:
  - Subdivide the IN tag.

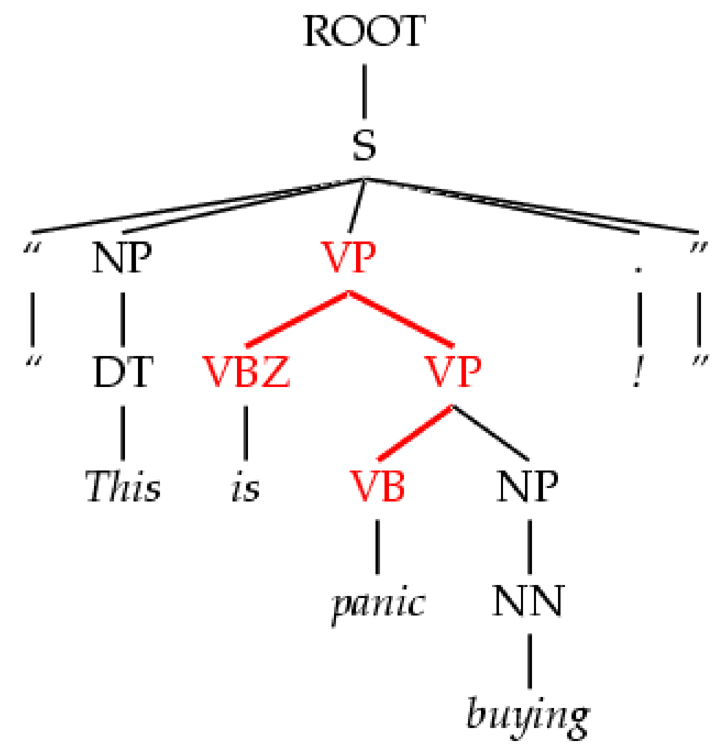
Annotation	F1	Size
Previous	78.3	8.0K
SPLIT-IN	80.3	8.1K

Yield Splits

- Problem: sometimes the behavior of a category depends on something inside its future yield (subtree).

- Examples:
  - Possessive NPs
  - Finite vs. infinite VPs
  - Lexical heads!

- Solution: annotate future elements into nodes.

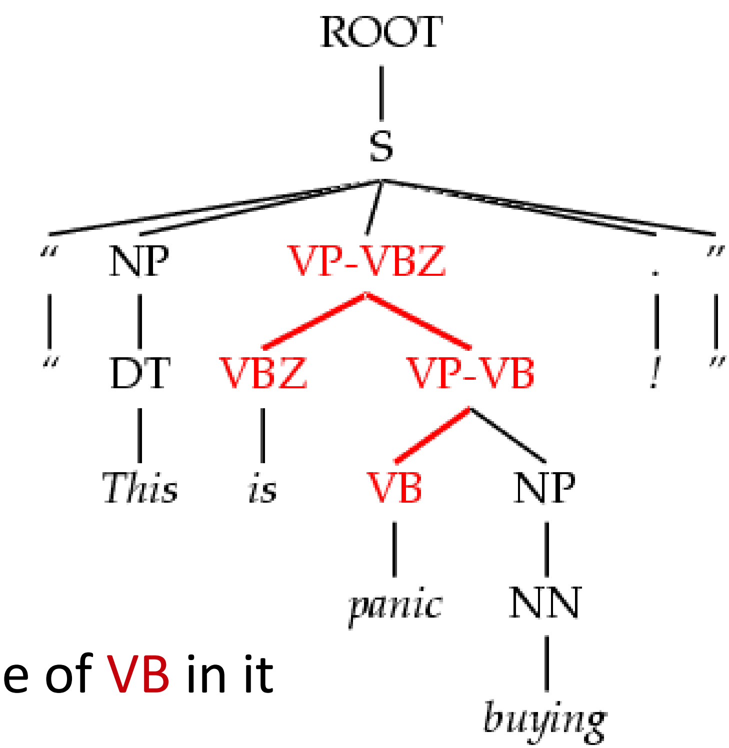


Annotation	F1	Size
tag splits	82.3	9.7K
POSS-NP	83.1	9.8K
SPLIT-VP	85.7	10.5K

Yield Splits

- Problem: sometimes the behavior of a category depends on something inside its future yield (subtree).

- Examples:
  - Possessive NPs
  - Finite vs. infinite VPs
  - Lexical heads!



Mark VP with the type of VB in it

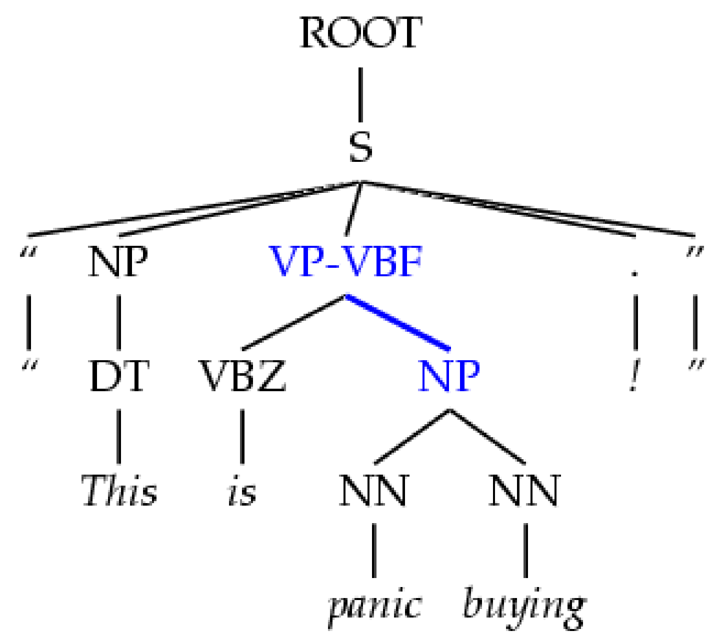
- Solution: annotate future elements into nodes.

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

Yield Splits

- Problem: sometimes the behavior of a category depends on something inside its future yield (subtree).

- Examples:
  - Possessive NPs
  - Finite vs. infinite VPs
  - Lexical heads!



Helps find the right parse

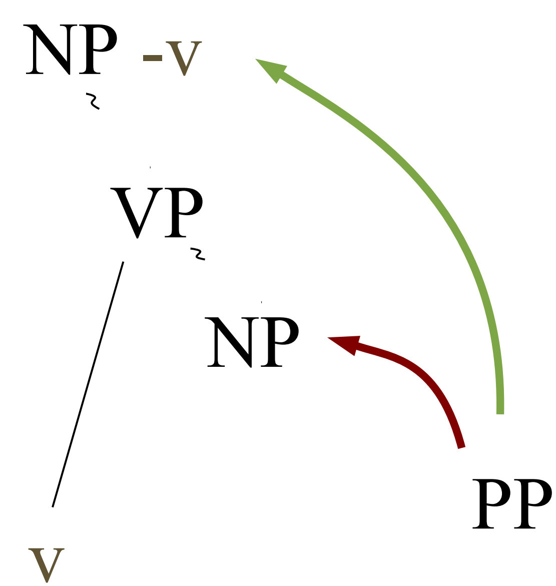
- Solution: annotate future elements into nodes.

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

Distance / Recursion Splits

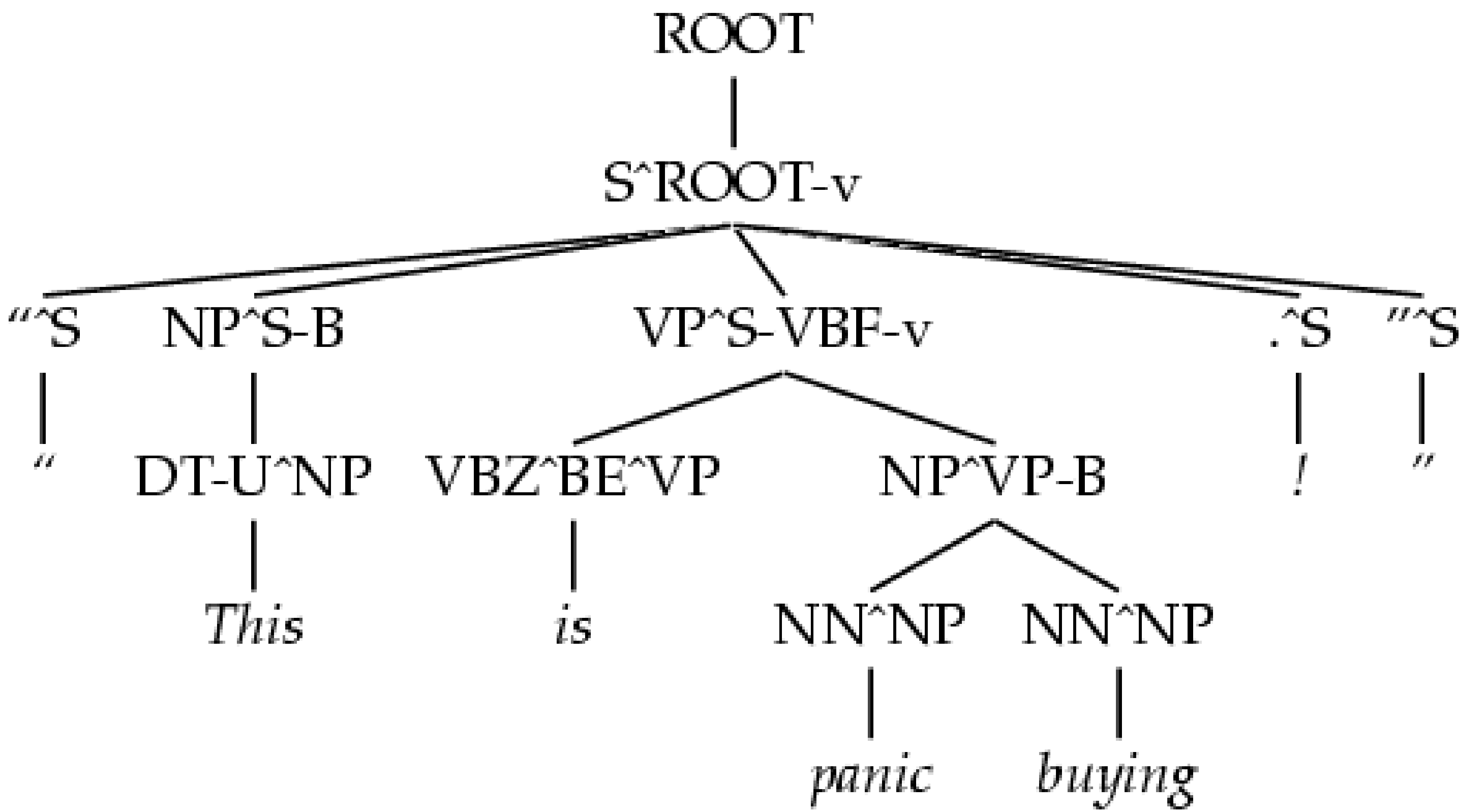
- Problem: vanilla PCFGs cannot distinguish attachment heights. Some tags tend to appear high while others appear low in the tree.

- 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

Summary

- Lexicalized parsers:
  - Use lexical information to annotate the grammar rules
  - Induces some “semantic” information into the parser
- Unlexicalized parsers:
  - Use some “context” to annotate the grammar rules
  - Deals with some problems of the independence assumptions of PCFGs
  - No use of lexical information
  - Comparable results to early lexicalized parsers

Recap

- Lexicalized Parsers
- Independence in PCFGs
- Unlexicalized Parsers