

SI485i : NLP

Set 9

Advanced PCFGs

Some slides from Chris Manning

Evaluating CKY

- How do we know if our parser works?
- Count the number of correct labels in your tree...the label and the span it dominates must both be correct.
 - [label, start, finish]
- Precision, Recall, F1 Score

Evaluation Metrics

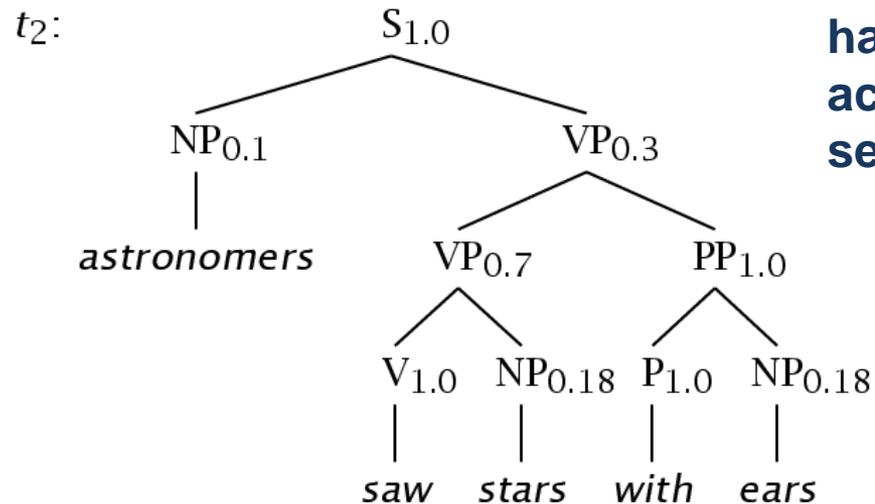
- C = number of correct non-terminals
- M = total number of non-terminals produced
- N = total number of non-terminals in the gold tree

- **Precision** = C / M
- **Recall** = C / N
- **F1 Score** (harmonic mean) = $2 * P * R / (P + R)$

Are PCFGs any good?

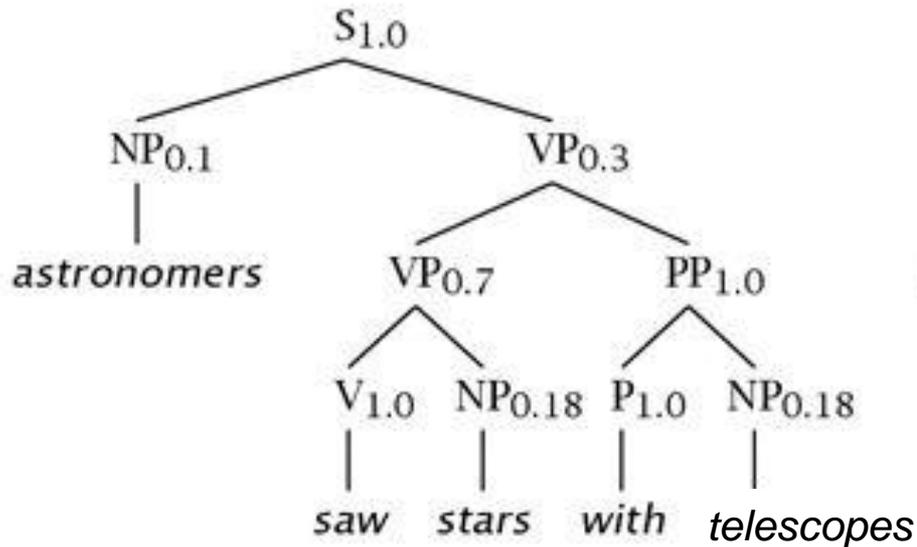
- Always produces *some* tree.
- Trees are reasonably good, giving a decent idea as to the correct structure.
- However, trees are rarely totally correct. Contain lots of errors.
- WSJ parsing accuracy = 73% F1

What's missing in PCFGs?

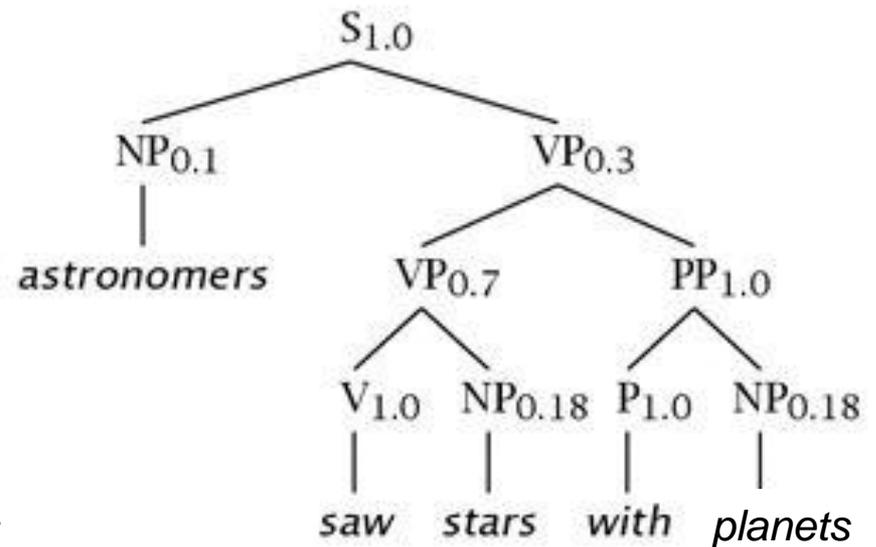


This choice of VP->VP PP has nothing to do with the actual words in the sentence.

Words barely affect structure.



Correct!!!



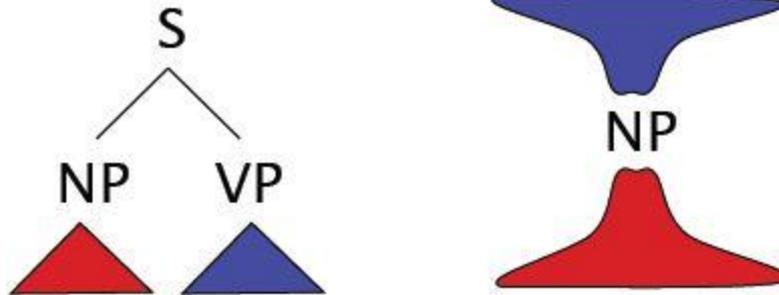
Incorrect

PCFGs and their words

- The words in a PCFG only link to their POS tags.
- The head word of a phrase contains a ton of information that the grammar does not use.
- Attachment ambiguity
 - “The astronomer saw the moon with the telescope.”
- Coordination
 - “The dogs in the house and the cats.”
- Subcategorization
 - “give” versus “jump”

PCFGs and their words

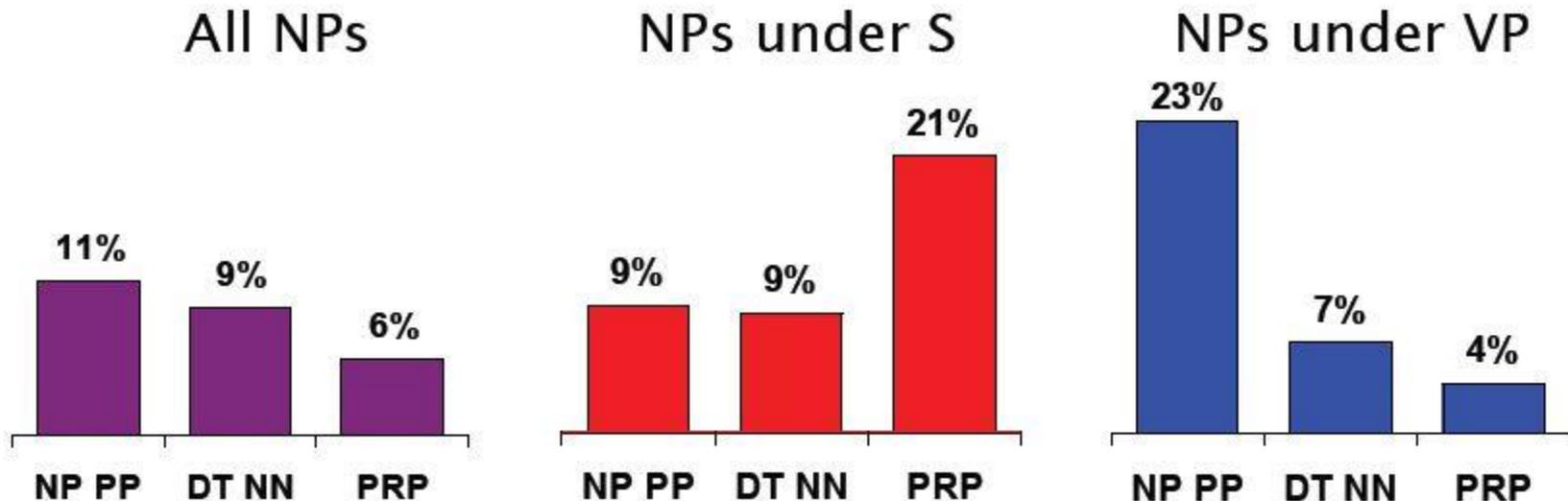
- The words are ignored due to our current independence assumptions in the PCFG.



- The words under the NP do not affect the VP.
- Any information that statistically connects above and below a node must flow through that node, so regions are independent given that central node.

PCFGs and independence

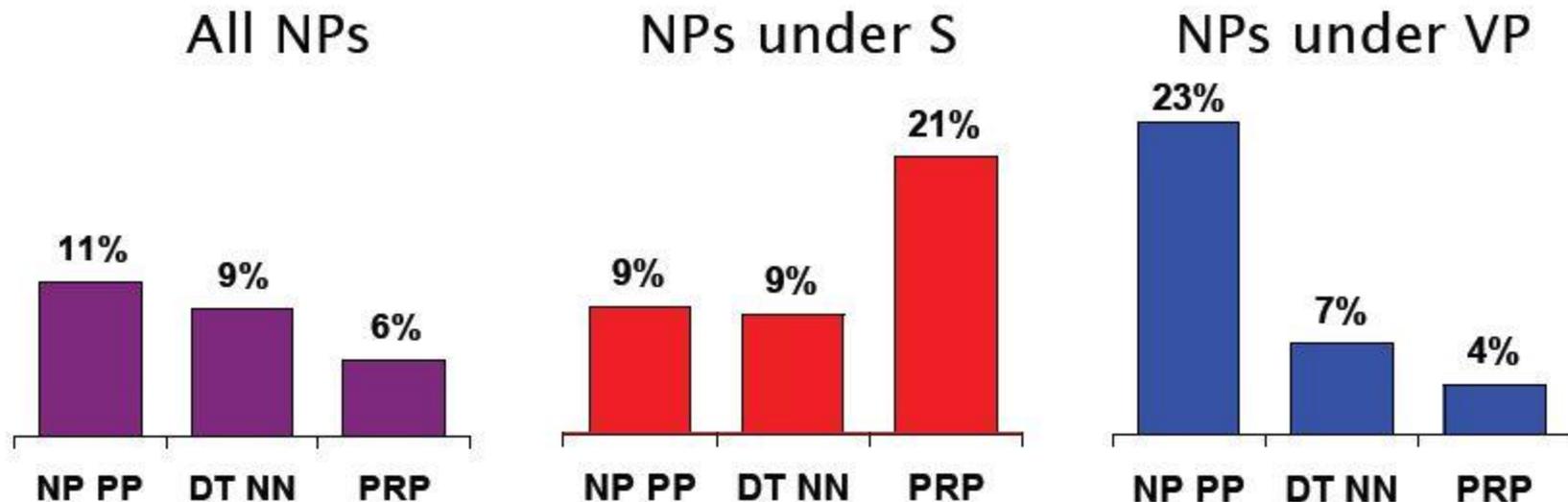
- Independence assumptions are too strong.



- The NPs under an S are typically what syntactic category? What about under a VP?

Relax the Independence

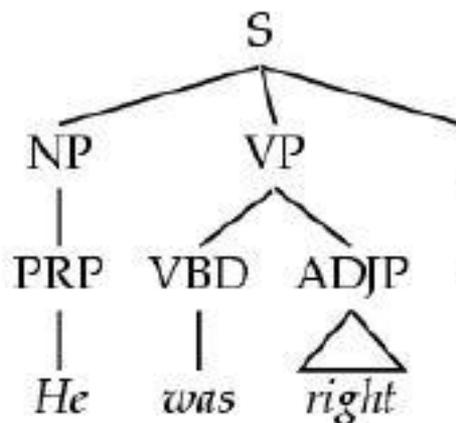
- **Thought question:** how could you change your grammar to encode these probabilities?



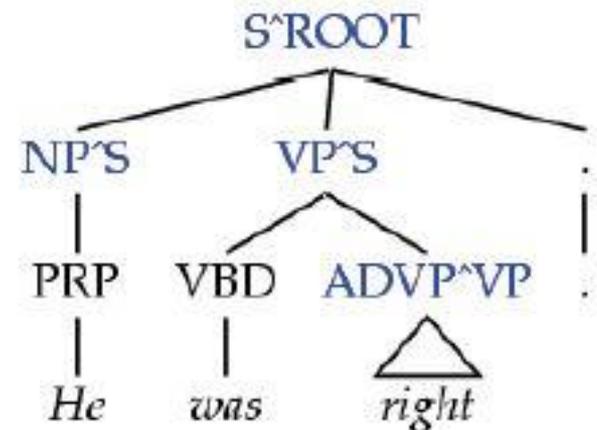
Vertical Markovization

- Expand the grammar
 - NP^S -> DT NN
 - NP^VP -> DT NN
 - NP^NP -> DT NN
 - etc.

Order 1



Order 2



Vertical Markovization

- Markovization can use k ancestors, not just $k=1$.
 - $NP^{\wedge}VP^{\wedge}S \rightarrow DT NN$
- The best distance in early experiments was $k=3$.
- **WARNING**: doesn't this explode the size of the grammar? Yes. But the algorithm is $O(n^3)$, so a bigger grammar (not n) doesn't have to hurt that much and the gain in performance can be worth it.

Horizontal Markovization

- Similar to vertical.
- Don't label with the parents, but now label with the left siblings in your immediate tree.
- This takes into context where you are in your local tree structure.

Markovization Results

Vertical Order		Horizontal Markov Order				
		$h = 0$	$h = 1$	$h \leq 2$	$h = 2$	$h = \infty$
$v = 1$	No annotation	71.27 (854)	72.5 (3119)	73.46 (3863)	72.96 (6207)	72.62 (9657)
$v \leq 2$	Sel. Parents	74.75 (2285)	77.42 (6564)	77.77 (7619)	77.50 (11398)	76.91 (14247)
$v = 2$	All Parents	74.68 (2984)	77.42 (7312)	77.81 (8367)	77.50 (12132)	76.81 (14666)
$v \leq 3$	Sel. GParents	76.50 (4943)	78.59 (12374)	79.07 (13627)	78.97 (19545)	78.54 (20123)
$v = 3$	All GParents	76.74 (7797)	79.18 (15740)	79.74 (16994)	79.07 (22886)	78.72 (22002)

Figure 2: Markovizations: F_1 and grammar size.

Figure from Klein & Manning (2003)

More Context in the Grammar

- Markovization is just the beginning. You can label non-terminals with all kinds of other useful information
 - Label nodes dominating verbs
 - Label NP as NP-POSS that has a possessive child (his dog)
 - Split IN tags into 6 categories!
 - Label CONJ tags if they are *but* or *and*
 - Give % its own tag
 - Etc.

Annotated Grammar Results

Annotation	Cumulative			Indiv.
	Size	F_1	ΔF_1	ΔF_1
Baseline ($v \leq 2, h \leq 2$)	7619	77.77	–	–
UNARY-INTERNAL	8065	78.32	0.55	0.55
UNARY-DT	8066	78.48	0.71	0.17
UNARY-RB	8069	78.86	1.09	0.43
TAG-PA	8520	80.62	2.85	2.52
SPLIT-IN	8541	81.19	3.42	2.12
SPLIT-AUX	9034	81.66	3.89	0.57
SPLIT-CC	9190	81.69	3.92	0.12
SPLIT-%	9255	81.81	4.04	0.15
TMP-NP	9594	82.25	4.48	1.07
GAPPED-S	9741	82.28	4.51	0.17
POSS-NP	9820	83.06	5.29	0.28
SPLIT-VP	10499	85.72	7.95	1.36
BASE-NP	11660	86.04	8.27	0.73
DOMINATES-V	14097	86.91	9.14	1.42
RIGHT-REC-NP	15276	87.04	9.27	1.94

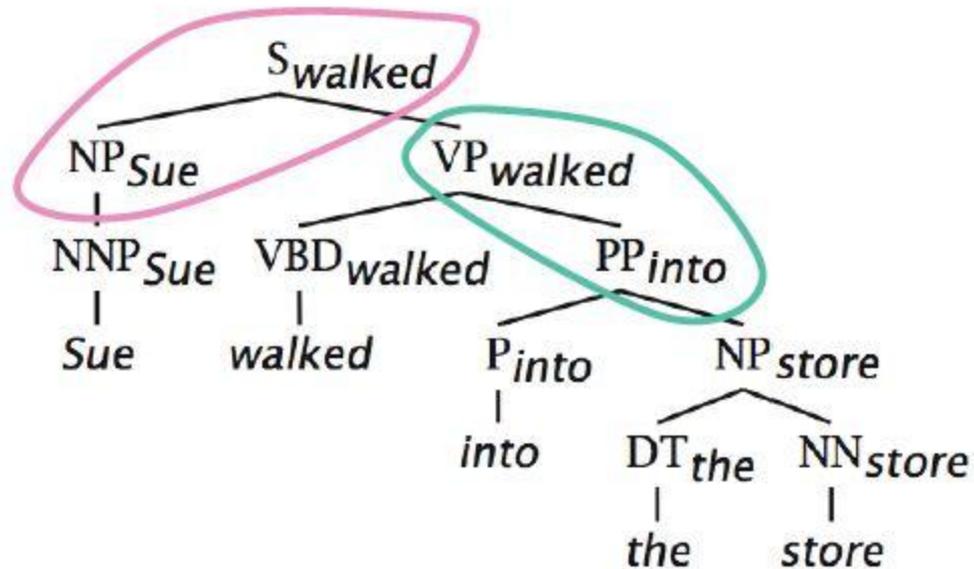
Figure from Klein & Manning (2003)

Lexicalization

- Markovization and all of these grammar additions relax the independence assumptions between “neighbor” nodes.
- We still haven’t used the words yet.
- **Lexicalization** is the process of adding the main word of the subtree to its non-terminal parent.

Lexicalization

- The *head word* of a phrase is the main content-bearing word.
- Use the *head word* to label non-terminals.



Lexicalization Benefits

- PP-attachment problems are better modeled
 - “announced rates in january”
 - “announced in january rates”
 - The VP-announce will prefer having “in MONTH” as its child
- Subcategorization frames are now used!
 - VP-give expects two NP children
 - VP-sit expects no NP children, maybe one PP
- And many others...

Lexicalization and Frames

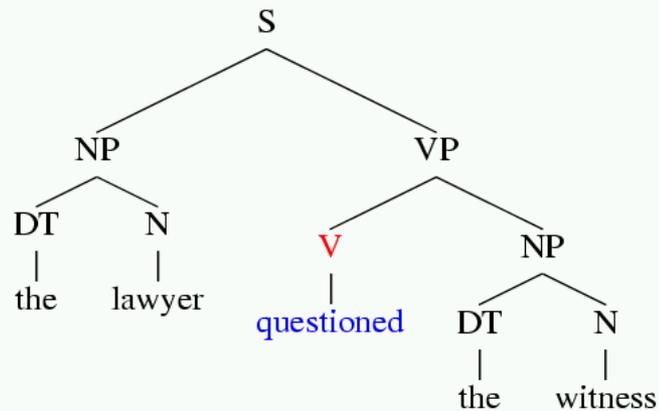
- Different probabilities of each VP rule if lexicalized with each of these four verbs:

<i>Local Tree</i>	<i>come</i>	<i>take</i>	<i>think</i>	<i>want</i>
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

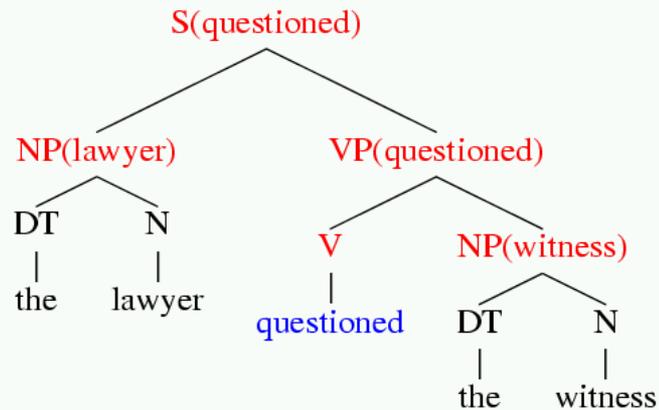
Independence Assumptions

- PCFGs



73% Accuracy

- Lexicalized PCFGs



88% Accuracy

Exercise!

The plane flew heavy cargo with its big engines.

1. Draw the parse tree. Binary rules not required.
2. Add lexicalization to the grammar rules.
3. Add 2nd order vertical markovization.

Putting it all together

- Lexicalized rules give you a massive gain. This was a big breakthrough in the 90's.
- You can combine lexicalized rules with markovization and all other features.
- Grammars explode.
- Lexicalization ... there are lots of details and backoff models that are required to make this work in reasonable time (not covered in this class).

State of the Art

- Parsing doesn't have to use these PCFG models.
- **Discriminative Learning** has been used to get the best gains. Instead of computing probabilities from MLE counts, it **weights** each rule through optimization techniques that we do not cover in this class.
- The best parsers output multiple trees, and then use a different algorithm to **rank** those possibilities.
- Best F1 performance: low-mid 90's.

Key Ideas

1. Parsing evaluation: precision/recall/F1
2. Independence assumptions of non-terminals
3. Markovization of grammar rules
4. Adding misc. features to rules
5. Lexicalization of grammar rules