Hand video

- http://www.youtube.com/watch?v=-KxjVlaLBmk

PARSING 3

David Kauchak
CS159 – Spring 2011

some slides adapted from Dan Klein

Admin

- Assignment 3 out
  - Due Friday at 6pm

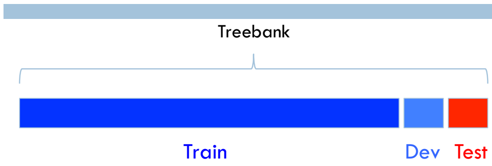
- How are things going?
- Where we've been

- Where we're going

Parsing evaluation

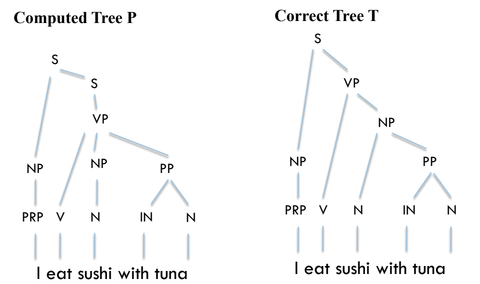
- You've constructed a parser
- You want to know how good it is
- Ideas?

Parsing evaluation



- Learn a model using the training set
- Parse the test set without looking at the "correct" trees
- Compare our generated parse tree to the "correct" tree

Comparing trees

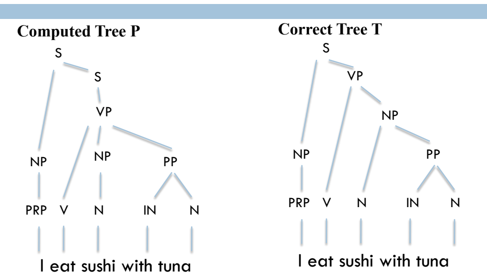


Ideas?

Comparing trees

- Idea 1: see if the trees match exactly
  - Problems?
    - Will have a low number of matches (people often disagree)
    - Doesn't take into account getting it *almost* right
- Idea 2: compare the constituents

Comparing trees



How many constituents match?

How can we turn this into a score?

Evaluation measures

- Precision

$$\frac{\text{\# of correct constituents}}{\text{\# of constituents in the computed tree}}$$

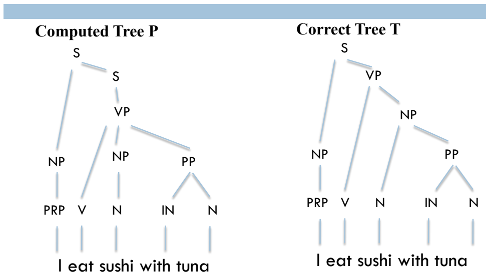
- Recall

$$\frac{\text{\# of correct constituents}}{\text{\# of constituents in the correct tree}}$$

- F1

$$\frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Comparing trees



Constituents: 11 # Correct Constituents: 9 # Constituents: 10

Precision: 9/11 Recall: 9/10 F1: 0.857

Parsing evaluation

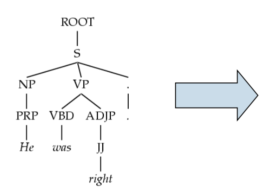
- Corpus: Penn Treebank, WSJ

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

- Parsing has been fairly standardized to allow for easy comparison between systems

Treebank PCFGs

- Use PCFGs for broad coverage parsing
- Can take a grammar right off the trees (doesn't work well):



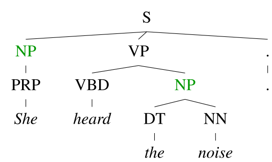
ROOT → S
S → NP VP .
NP → PRP
VP → VBD ADJP
.....

Model	F1
Baseline	72.0

Generic PCFG Limitations

- PCFGs do not use any information about where the current constituent is in the tree
- PCFGs do not rely on specific words or concepts, only general structural disambiguation is possible (e.g. prefer to attach PPs to Nominals)
- MLE estimates are not always the best

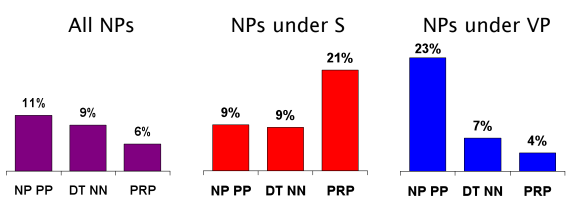
Conditional Independence?



- Not every NP expansion can fill every NP slot
  - A grammar with symbols like "NP" won't be context-free
  - Statistically, conditional independence too strong

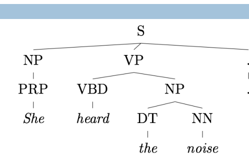
Non-Independence

- Independence assumptions are often too strong.



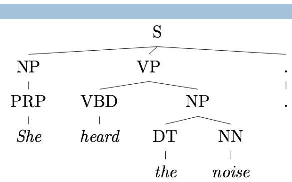
- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
- Also: the subject and object expansions are correlated

Grammar Refinement



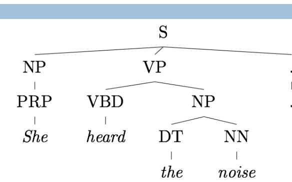
- PCFG would treat these two NPs the same… but they're not!
- We can't exchange them: "the noise heard she"
- Idea: expand/refine our grammar
- Challenges:
  - Must refine in ways that facilitate disambiguation
  - Too much refinement -> sparsity problems
  - To little -> can't discriminate (PCFG)

Grammar Refinement



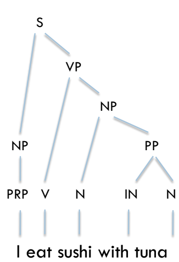
Ideas?

Grammar Refinement



- Structure Annotation [Johnson '98, Klein&Manning '03]
  - Differentiate constituents based on their local context
- Lexicalization [Collins '99, Charniak '00]
  - Differentiate constituents based on the spanned words
- Constituent splitting [Matsuzaki et al. 05, Petrov et al. '06]
  - Cluster/group words into sub-constituents

Less independence

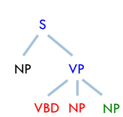


S -> NP VP
NP -> PRP
PRP -> I
VP -> V NP
V -> eat
NP -> N PP
N -> sushi
PP -> IN N
IN -> with
N -> tuna

We're making a strong independence assumption here!

Markovization

- Except for the root node, every node in a parse tree has:
  - A vertical history/context
  - A horizontal history/context

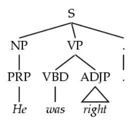


Traditional PCFGs use the full horizontal context and a vertical context of 1

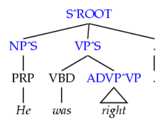
Vertical Markovization

- Vertical Markov order: rewrites depend on past k ancestor nodes.
- Order 1 is most common: aka parent annotation

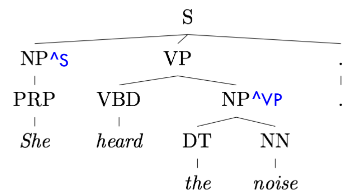
Order 1



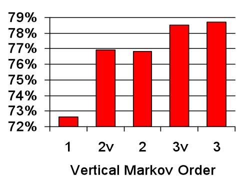
Order 2

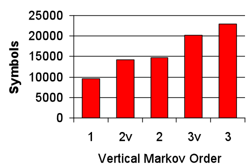


Allows us to make finer grained distinctions



Vertical Markovization



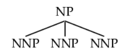


F1 performance

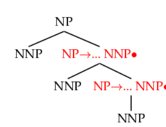
# of non-terminals

Horizontal Markovization

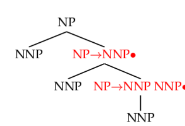
- Horizontal Markov order: rewrites depend on past k ancestor nodes
- Order 1 is most common: condition on a single sibling



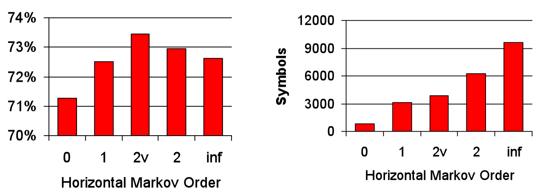
Order 1



Order ∞



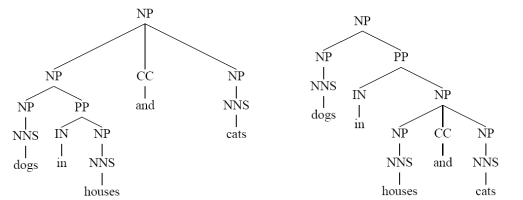
Horizontal Markovization



F1 performance

# of non-terminals

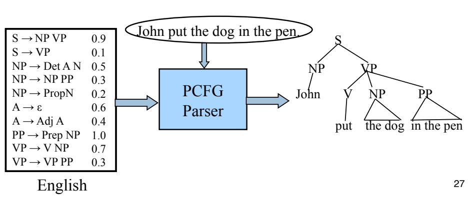
Problems with PCFGs



- What's different between basic PCFG scores here?

Example of Importance of Lexicalization

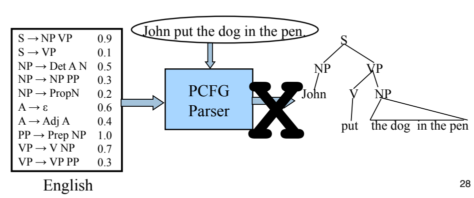
- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG.
- But the desired preference can depend on specific words.



27

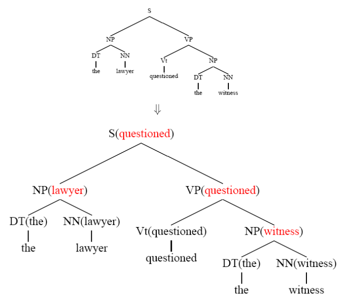
Example of Importance of Lexicalization

- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG.
- But the desired preference can depend on specific words.



28

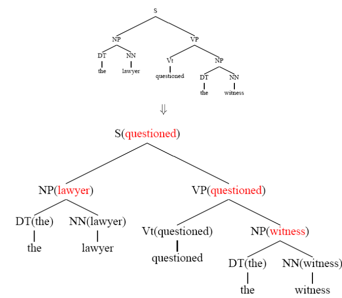
Lexicalized Trees



How could we lexicalize the grammar/tree?

Lexicalized Trees

- Add “headwords” to each phrasal node
  - Syntactic vs. semantic heads
  - Headship not in (most) treebanks
  - Usually *use head rules*, e.g.:
    - NP:
      - Take leftmost NP
      - Take rightmost N*
      - Take rightmost JJ
      - Take right child
    - VP:
      - Take leftmost VB*
      - Take leftmost VP
      - Take left child



Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

  VP(put) → VBD(put) NP(dog) PP(in)

- How would we estimate the probability of this rule?

$$\frac{\text{Count(VP(put) -> VBD(put) NP(dog) PP(in))}}{\text{Count(VP (put))}}$$

- Never going to get these automically off of a treebank
- Ideas?

One approach

- Combine this with some of the markovization techniques we saw
- Collins' (1999) parser
  - Models productions based on context to the left and the right of the head daughter.
    - $LHS \rightarrow L_n L_{n-1} \ldots L_1 H\ R_1 \ldots R_{m-1} R_m$
  - First generate the head (H) and then repeatedly generate left (L_i) and right (R_i) context symbols until the symbol STOP is generated.

Sample Production Generation

$VP_{put} \rightarrow VBD_{put}\ NP_{dog}\ PP_{in}$

Note: Penn treebank tends to have fairly flat parse trees that produce long productions.

$VP_{put} \rightarrow$	STOP	VBD_{put}	NP_{dog}	PP_{in}	STOP
	L_1	H	R_1	R_2	R_3

$$P_L(STOP \mid VP_{put}) * P_H(VBD \mid Vp_{put}) * P_R(NP_{dog} \mid VP_{put}) * P_R(PP_{in} \mid VP_{put}) * P_R(STOP \mid PP_{in})$$

Estimating Production Generation Parameters

- Estimate P_H, P_L, and P_R parameters from treebank data.

$$P_R(PP_{in} \mid VP_{put}) = \frac{\text{Count}(PP_{in} \text{ right of head in a } VP_{put} \text{ production})}{\text{Count}(\text{symbol right of head in a } VP_{put\text{-}VBD})}$$

$$P_R(NP_{dog} \mid VP_{put}) = \frac{\text{Count}(NP_{dog} \text{ right of head in a } VP_{put} \text{ production})}{\text{Count}(\text{symbol right of head in a } VP_{put})}$$

- Smooth estimates by combining with simpler models conditioned on just POS tag or no lexical info

$$smP_R(PP_{in} \mid VP_{put\text{-}}) = \lambda_1 P_R(PP_{in} \mid VP_{put}) + (1-\lambda_1)\,(\lambda_2 P_R(PP_{in} \mid VP_{VBD}) + (1-\lambda_2)\, P_R(PP_{in} \mid VP))$$

Problems with lexicalization

- We've solved the estimation problem
- There's also the issue of performance
- Lexicalization causes the size of the number of grammar rules to explode!
- Our parsing algorithms take too long too finish

- Ideas?

Pruning during search

- We can no longer keep all possible parses around
- We can no longer guarantee that we actually return the most likely parse
- Beam search [Collins 99]
  - In each cell only keep the **K** most likely hypothesis
  - Disregard constituents over certain spans (e.g. punctuation)
  - F1 of 88.6!

Pruning with a PCFG

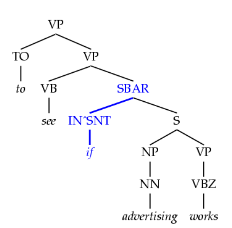
- The Charniak parser prunes using a two-pass approach [Charniak 97+]
  - First, parse with the base grammar
  - For each X:[i,j] calculate P(X|i,j,s)
    - This isn't trivial, and there are clever speed ups
  - Second, do the full $O(n^5)$ CKY
    - Skip any X :[i,j] which had low (say, < 0.0001) posterior
  - Avoids almost all work in the second phase!
- F1 of 89.7!

Tag splitting

- Lexicalization is an extreme case of splitting the tags to allow for better discrimination
- Idea: what if rather than doing it for all words, we just split some of the tags

Tag Splits

- Problem: Treebank tags are too coarse
- Example: Sentential, PP, and other prepositions are all marked IN
- Partial Solution:
  - Subdivide the IN tag



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

Other Tag Splits

	F1	Size
UNARY-DT: mark demonstratives as DT^U ("the X" vs. "those")	80.4	8.1K
UNARY-RB: mark phrasal adverbs as RB^U ("quickly" vs. "very")	80.5	8.1K
TAG-PA: mark tags with non-canonical parents ("not" is an RB^VP)	81.2	8.5K
SPLIT-AUX: mark auxiliary verbs with –AUX [cf. Charniak 97]	81.6	9.0K
SPLIT-CC: separate "but" and "&" from other conjunctions	81.7	9.1K
SPLIT-%: "%" gets its own tag.	81.8	9.3K

Learning good splits: Latent Variable Grammars