ADVANCED PARSING

David Kauchak
CS159 – Spring 2019

some slides adapted from Dan Klein

Admin

Assignment 3?

Assignment 4 (A and B)

Lab on Wednesday

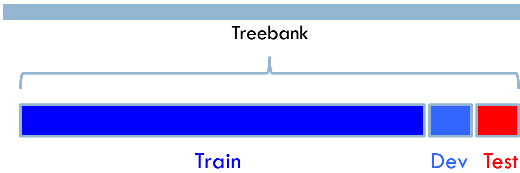
Parsing evaluation

You've constructed a parser

You want to know how good it is

Ideas?

Parsing evaluation



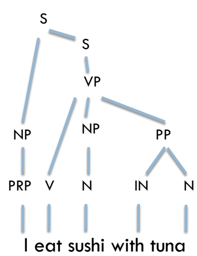
Learn a model using the training set

Parse the test set without looking at the "correct" trees

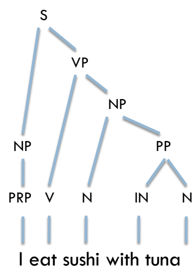
Compare our generated parse tree to the "correct" tree

Comparing trees

Computed Tree P



Correct Tree T



Ideas?

Comparing trees

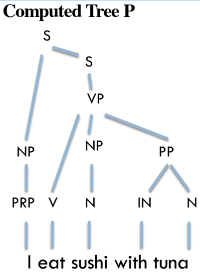
Idea 1: see if the trees match exactly

- Problems?
  - Will have a low number of matches (people often disagree)
  - Doesn't take into account getting it *almost* right

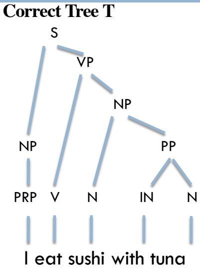
Idea 2: compare the constituents

Comparing trees

Computed Tree P



Correct Tree T



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

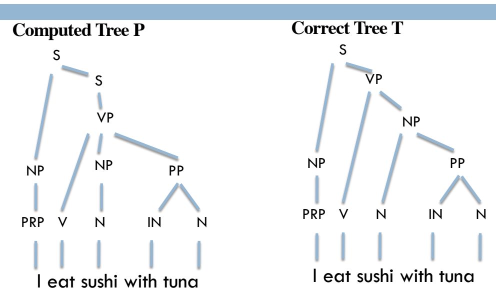
$$\frac{\text{\# of correct constituents}}{\text{\# of constituents in the correct tree}}$$

F1

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

What does this favor?

Comparing trees



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

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

Parsing evaluation

Corpus: Penn Treebank, WSJ

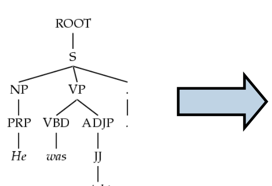
Training:	sections	02-21
Development:	section	22 (first 20 files)
Test:	section	23

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

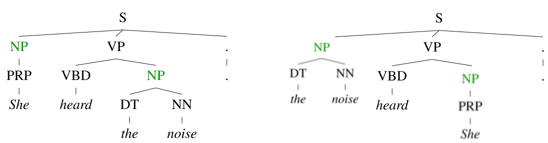
Generic PCFG Limitations

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

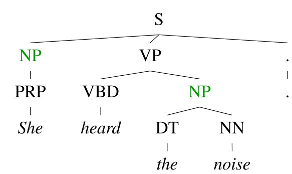
Conditional Independence?



Will a PCFG differentiate between these?

What's the problem?

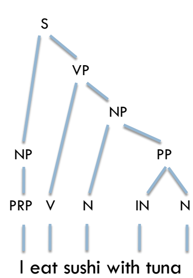
Conditional Independence?



It treats all NPs as equivalent… but they're not!

- A grammar with symbols like "NP" won't be context-free
- Statistically, conditional independence too strong

Stong independence assumption

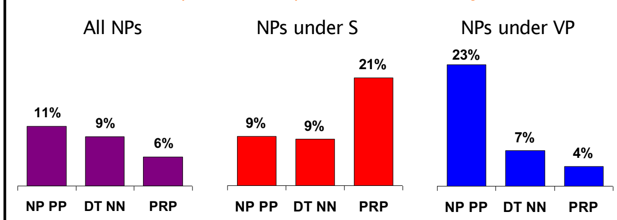


S -> NP VP
NP -> PRP
PRP -> I
VP -> V NP
V -> eat
NP -> N PP
N -> sushi
PP -> IN N
IN -> with
N -> tuna

We're making a strong independence assumption here!

Non-Independence

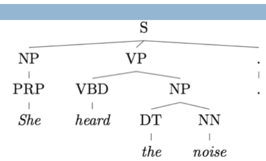
Independence assumptions are often too strong



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

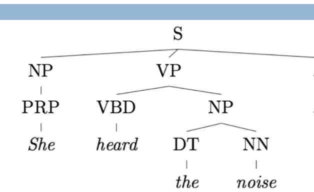


Idea: expand/refine our grammar

Challenges:

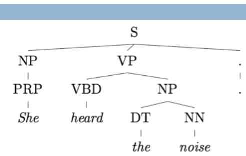
- Must refine in ways that facilitate disambiguation
- Must trade-offs between too little and too much refinement.
  - Too much refinement -> sparsity problems
  - To little -> can't discriminate (PCFG)

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

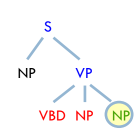
Constituent splitting [Matsuzaki et al. 05, Petrov et al. '06]

- Cluster/group words into sub-constituents

Markovization

Except for the root node, every node in a parse tree has:

- A vertical history/context
- A horizontal history/context



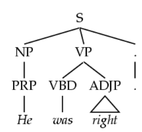
Traditional PCFGs use the full horizontal context and a vertical context of 1

Vertical Markovization

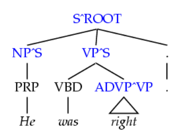
Vertical Markov order: rewrites depend on past k ancestor nodes.

Order 1 is most common: aka parent annotation

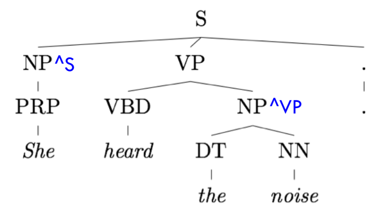
Order 1



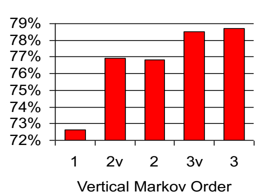
Order 2



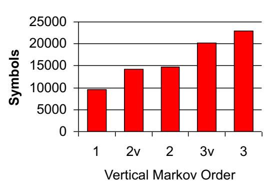
Allows us to make finer grained distinctions



Vertical Markovization



F1 performance

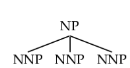


# of non-terminals

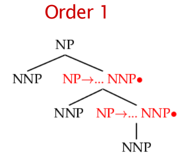
Horizontal Markovization

Horizontal Markov order: rewrites depend on past k sibling nodes

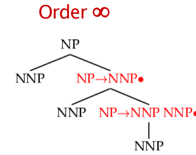
Order 1 is most common: condition on a single sibling



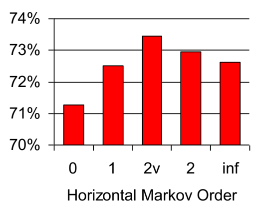
Order 1

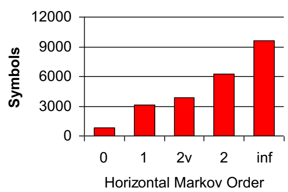


Order ∞



Horizontal Markovization

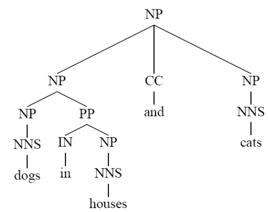


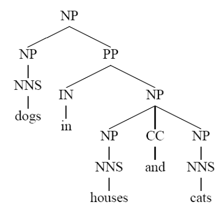


F1 performance

# of non-terminals

Problems with PCFGs



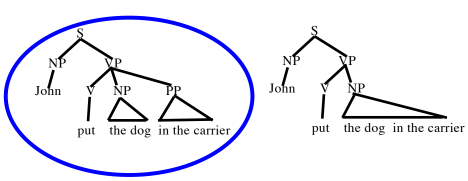


What's different between basic PCFG scores here?

Example of Importance of Lexicalization

A general preference for attaching PPs to NPs rather than VPs can be learned by an ordinary PCFG

But the desired preference can depend on specific words

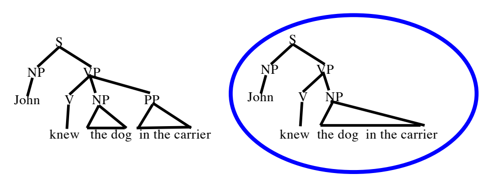


Which is correct?

Example of Importance of Lexicalization

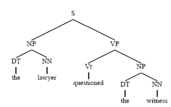
A general preference for attaching PPs to NPs rather than VPs can be learned by an ordinary PCFG

But the desired preference can depend on specific words



Which is correct?

Lexicalized Trees

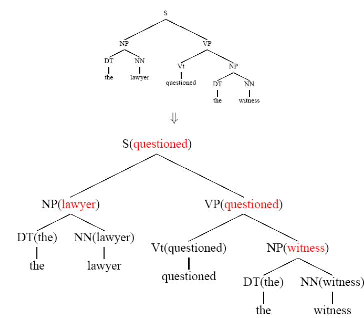


How could we lexicalize the grammar/tree?

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?

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

Never going to get these automatically 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 child.

$$\text{LHS} \rightarrow L_n L_{n-1} \ldots L_1 H\ R_1 \ldots R_{m-1} R_m$$

One approach

$LHS \rightarrow L_n L_{n-1} \ldots L_1 H\ R_1 \ldots R_{m-1} R_m$

First generate the head (H) given the parent

Then repeatedly generate left symbols (L_i) until the beginning is reached

Then right (R_i) symbols until the end is reached

Sample Production Generation

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

$VP_{put} \rightarrow$

Sample Production Generation

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

$VP_{put} \rightarrow$ VBD_{put} (H)

$P_H(VBD \mid VP_{put})$

Sample Production Generation

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

$VP_{put} \rightarrow$ STOP (L_1) VBD_{put} (H)

$P_L(STOP \mid VP_{put})$

Sample Production Generation

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

$VP_{put} \rightarrow$ STOP (L_1) VBD_{put} (H) NP_{dog} (R_1)

$P_R(NP_{dog} \mid VP_{put})$

Sample Production Generation

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

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

$P_R(PP_{in} \mid VP_{put})$

Sample Production Generation

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

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

$P_R(STOP \mid PP_{in})$

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 (L_1) VBD_{put} (H) NP_{dog} (R_1) PP_{in} (R_2) STOP (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

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

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

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

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

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 hypotheses
- 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 (non-lexicalized) 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 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

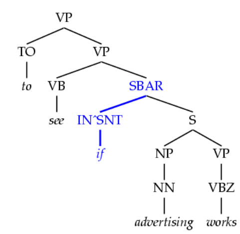
Problem: Treebank tags are too coarse

- We even saw this with the variety of tagsets

Example: Sentential, PP, and other prepositions are all marked IN

Partial Solution:

- Subdivide the IN tag

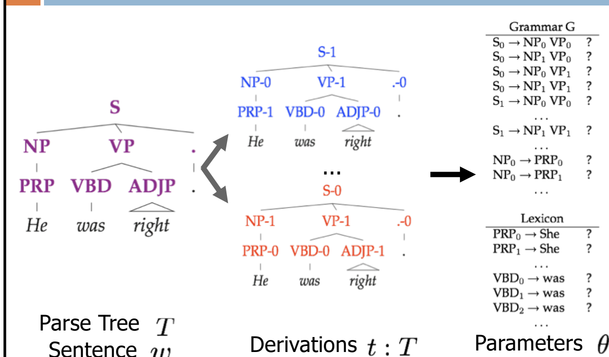


Annotation	F1	Size
Previous	78.3	8.0K
SPLIT-IN	80.3	8.1K

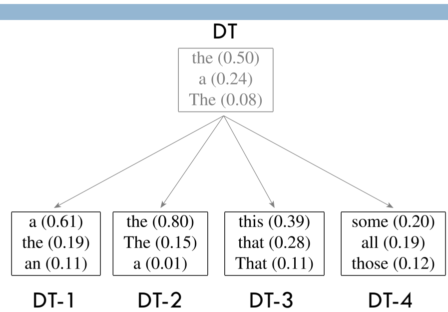
Other Tag Splits

	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



Refinement of the DT tag



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

Parser	*F1 ≤ 40 words*	*F1 all words*
Klein & Manning ’03	86.3	85.7
Matsuzaki et al. ’05	86.7	86.1
Collins ’99	88.6	88.2
Charniak & Johnson ’05	90.1	89.6
Petrov et. al. 06	**90.2**	**89.7**

Human Parsing

How do humans do it?

How might you try and figure it out computationally/experimentally?

Human Parsing

Read these sentences

Which one was fastest/slowest?

John put the dog in the pen with a lock.

John carried the dog in the pen with a bone in the car.

John liked the dog in the pen with a bone.

Human Parsing

Computational parsers can be used to predict human reading time as measured by tracking the time taken to read each word in a sentence.

Psycholinguistic studies show that words that are more probable given the preceding lexical and syntactic context are read faster.

- John put the dog in the pen with a lock.
- John carried the dog in the pen with a bone in the car.
- John liked the dog in the pen with a bone.

Modeling these effects requires an ***incremental*** statistical parser that incorporates one word at a time into a continuously growing parse tree.

Garden Path Sentences

People are confused by sentences that seem to have a particular syntactic structure but then suddenly violate this structure, so the listener is "lead down the garden path".

- The horse raced past the barn fell.
  - vs. The horse raced past the barn broke his leg.
- The complex houses married students.
- The old man the sea.
- While Anna dressed the baby spit up on the bed.

Incremental computational parsers can try to predict and explain the problems encountered parsing such sentences.

More garden sentences

http://www.fun-with-words.com/ambiguous_garden_path.html

The prime number few.
Fat people eat accumulates.
The cotton clothing is usually made of grows in Mississippi.
Until the police arrest the drug dealers control the street.
The man who hunts ducks out on weekends.
When Fred eats food gets thrown.
Mary gave the child the dog bit a bandaid.
The girl told the story cried.
I convinced her children are noisy.
Helen is expecting tomorrow to be a bad day.
The horse raced past the barn fell.
I know the words to that song about the queen don't rhyme.
She told me a little white lie will come back to haunt me.
The dog that I had really loved bones.
That Jill is never here hurts.
The man who whistles tunes pianos.
The old man the boat.
Have the students who failed the exam take the supplementary.
The raft floated down the river sank.
We painted the wall with cracks.
The tycoon sold the offshore oil tracts for a lot of money wanted to kill JR.