Admin

- Assignment 3?
- Assignment 4 (A and B)
- Lab next Tuesday

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

Correct Tree T

Computed Tree P

I eat sushi with tuna

I eat sushi with tuna

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

How many constituents match?
How can we turn this into a score?

Evaluation measures

Precision

Recall

F1

$2 \cdot \text{Precision} \cdot \text{Recall}$

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

What does this favor?
Comparing trees

<table>
<thead>
<tr>
<th>Correct Tree T</th>
<th>Computed Tree P</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → VP NP</td>
<td>S → VP NP</td>
</tr>
<tr>
<td>S → IN NP</td>
<td>S → IN NP</td>
</tr>
<tr>
<td>VP → V IN</td>
<td>VP → V IN</td>
</tr>
<tr>
<td>NP → PRP</td>
<td>NP → PRP</td>
</tr>
<tr>
<td>VP → VBD ADJP</td>
<td>VP → VBD ADJP</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>72.0</td>
</tr>
</tbody>
</table>

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

Strong independence assumption

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
  - Too little -> can’t discriminate (PCFG)

17

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

19

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

Horizontal Markovization

Horizontal Markov order: rewrites depend on past $k$ sibling nodes

Order 1 is most common: condition on a single sibling

F1 performance

# of non-terminals
Horizontal Markovization

F1 performance

Problems with PCFGs

What's different between basic PCFG scores here?

Example of Importance of Lexicalization

Sentence 1: John put the dog in the carrier
Sentence 2: John knew the dog in the carrier

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 VP
    - Take leftmost VP
    - Take right child

Lexicalized PCFGs?

Problem: we now have to estimate probabilities like

\[ \text{VP(put)} \rightarrow \text{VBD(put) NP(dog) PP(in)} \]

How would we estimate the probability of this rule?

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

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.

\[ \text{LHS} \rightarrow L_1 L_2 \ldots L_i H R_1 R_2 \ldots R_m \]
One approach

LHS → L₁L₂...LₙH R₁R₂...Rₘ

First generate the head (H) given the parent
Then repeatedly generate left symbols (Lᵢ) until the beginning is reached
Then right (Rᵢ) symbols until the end is reached

Sample Production Generation

VP₁ → VBD₁ NP₁ PP₁

VP₂ →

Sample Production Generation

VP₁ → VBD₁ NP₁ PP₁

VP₂ →

Sample Production Generation

VP₁ → VBD₁ NP₁ PP₁

VP₂ → STOP L₁ H

Pₜ(STOP | VPₙ)

Sample Production Generation

VP₁ → VBD₁ NP₁ PP₁

VP₂ → STOP L₁ H

Pₜ(STOP | VPₙ)
Note: Penn treebank tends to have fairly flat parse trees that produce long productions.
Estimating Production Generation Parameters

Estimate $P_{\text{PP}}$, $P_{\text{NP}}$, and $P_{\text{R}}$ parameters from treebank data

\[
P_{\text{PP}}(\text{right of head in a VP PUT production}) = \frac{\text{Count}(\text{PP right of head in a VP PUT production})}{\text{Count}(\text{symbol right of head in a VP PUT production})}
\]

\[
P_{\text{NP}}(\text{right of head in a VP PUT}) = \frac{\text{Count}(\text{NP right of head in a VP PUT production})}{\text{Count}(\text{symbol right of head in a VP PUT production})}
\]

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

\[
s_{\text{PP}}(\text{right of head in a VP PUT}) = \lambda_1 P_{\text{PP}}(\text{right of head in a VP PUT}) + (1 - \lambda_1) P_{\text{PP}}(\text{right of head in a VP PUT})
\]

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 $P(X_{[i,j]})$
  □ 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.

Other Tag Splits

UNARY-DT: mark demonstratives as DT^U (“the X” vs. “those”)

UNARY-RB: mark phrasal adverbs as RB^U (“quickly” vs. “very”)

TAG-PA: mark tags with non-canonical parents (“not” is on RB^VP)

SPLIT-AUX: mark auxiliary verbs with –AUX [cf. Charniak 97]

SPLIT-CC: separate “but” and “&” from other conjunctions

SPLIT-%: “%” gets its own tag.

Learning good splits:

Latent Variable Grammars

<table>
<thead>
<tr>
<th>Grammar G</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP, VP, ?</td>
<td></td>
</tr>
<tr>
<td>S → NP, VP, ?</td>
<td></td>
</tr>
<tr>
<td>S → NP, VP, ?</td>
<td></td>
</tr>
<tr>
<td>S → NP, VP, ?</td>
<td></td>
</tr>
<tr>
<td>S → NP, VP, ?</td>
<td></td>
</tr>
<tr>
<td>S → NP, VP, ?</td>
<td></td>
</tr>
<tr>
<td>S → NP, VP, ?</td>
<td></td>
</tr>
<tr>
<td>S → NP, VP, ?</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parse Tree</th>
<th>Derivations t : T</th>
<th>Parameters θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence</td>
<td>T</td>
<td>w</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
<tr>
<td>SPLIT-IN</td>
<td>80.3</td>
<td>8.1K</td>
</tr>
</tbody>
</table>
Refinement of the DT tag

DT

- the (0.50)
- a (0.24)
- The (0.08)
- a (0.61)
- the (0.19)
- an (0.11)
- this (0.39)
- that (0.28)
- some (0.20)
- all (0.19)
- those (0.12)

DT-1 DT-2 DT-3 DT-4

Learned Splits

Proper Nouns (NNP):
- 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

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-11: million, billion, trillion
- CD-0: 1, 50, 100
- CD-3: 1, 30, 31
- CD-9: 78, 58, 34

Learned Splits

Final Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 ≤ 40 words</th>
<th>F1 all words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
<td>85.7</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
<td>86.1</td>
</tr>
<tr>
<td>Collins ’99</td>
<td>88.6</td>
<td>88.2</td>
</tr>
<tr>
<td>Charniak &amp; Johnson ’05</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td>Petrov et. al. 06</td>
<td>90.2</td>
<td>89.7</td>
</tr>
</tbody>
</table>
Human Parsing

How do humans do it?

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

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.

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

The prime number few.
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 barked.
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 soft 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.