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

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

\[
\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

What does this favor?
Comparing trees

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

I eat sushi with tuna

# 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?

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)

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

![Diagram of Vertical Markovization](image1)

---

**Allow us to make finer grained distinctions**

![Example of Vertical Markovization](image2)

---

**Horizontal Markovization**

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

Order 1 is most common: condition on a single sibling

![Diagram of Horizontal Markovization](image3)
Horizontal Markovization

Problems with PCFGs

Symbols

F1 performance # of non-terminals

What's different between basic PCFG scores here?

Example of Importance of Lexicalization

Example of Importance of Lexicalization

Which is correct?

Which is correct?
Lexicalized Trees

How could we lexicalize the grammar/tree?

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
  - Take rightmost JJ
  - Take right child

- VP:
  - Take leftmost VP
  - Take leftmost NP
  - Take left child

Lexicalized PCFGs?

Problem: we now have to estimate probabilities like

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

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.

\[ LHS \rightarrow L_{l_{n-1}} \ldots L_{l_1} H R_{r_{m}} \ldots R_{r_1} \]
One approach

LHS → L₁L₂...LᵢH...Rₜ⁻¹...Rₘ

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

Sample Production Generation

VPₜₜₜ → VBDₜₜₜ NPₜₜₜ PPₜₜₜ

VPₜₜₜ →

Sample Production Generation

VPₜₜₜ → VBDₜₜₜ NPₜₜₜ PPₜₜₜ

VPₜₜₜ →

Sample Production Generation

VPₜₜₜ → VBDₜₜₜ NPₜₜₜ PPₜₜₜ

VPₜₜₜ → STOP

Pₛ(VBD | VPₜₜₜ)

Sample Production Generation

VPₜₜₜ → VBDₜₜₜ NPₜₜₜ PPₜₜₜ

VPₜₜₜ → STOP

Pₛ(STOP | VPₜₜₜ)
Sample Production Generation

Sample Production Generation

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

Estimate $P_{PH}$, $P_L$, and $P_R$ parameters from treebank data

$P_{PH}(\text{PP}\rightarrow \text{VP}_{put}) = \frac{\text{Count}(\text{said right of head in a VP_{put} production})}{\text{Count}(\text{symbol right of head in a VP_{put})}}$

$P_R(NP\rightarrow \text{VP}_{put}) = \frac{\text{Count}(\text{NP dog right of head in a VP_{put} production})}{\text{Count}(\text{NP dog left of head in a VP_{put}})}$

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

$smP_R(\text{PP}\rightarrow \text{VP}_{put}) = \lambda_1 P_{PH}(\text{PP}\rightarrow \text{VP}_{put}) + (1-\lambda_1) P_{PH}(\text{PP}\rightarrow \text{VP})$

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 $P(X[i,j]|p)$
  - 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!
### 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

**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 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.
http://www.funwith-words.com/ambiguous_garden_path.html