Lecture 11:  
PCFGs: getting luckier all the time

Professor Robert C. Berwick
berwick@csail.mit.edu
Menu

• Statistical Parsing with Treebanks I: breaking independence assumptions
• How lucky can we get? Or: Whatasamatter U? (Stanford, Upenn, Berkeley)
• Staying in condition: The importance of grandparents
• Statistical Parsing with Treebanks II: lexicalization & discrimination: the state of the state of the art
How words matter for parsing decisions

• Consider the Prepositional Phrase (PP) attachment decisions again

• Words are good predictors even absent ‘total’ understanding
  • Children ate the cake with a spoon
  • Children ate the cake with frosting
OK, so we can try this

- Use a simple likelihood ratio test:

\[
LR(v, n, p) = \frac{L(p \mid v)}{L(p \mid n)}
\]

- \(p(\text{with} \mid \text{agreement}) = 0.15\)
- \(p(\text{with} \mid \text{breach}) = 0.02\)
- \(LR(\text{breach, with, agreement}) = 0.13 \rightarrow \text{choose noun attachment}\)
PP problems

• Doesn’t always work
• Chrysler confirmed that it would end its troubled venture with Maserati
• Should be a noun attachment, but stats says v:
  • w \hspace{1cm} count(w) \hspace{1cm} count(w, with)
  • end \hspace{1cm} 5156 \hspace{1cm} 607
  • venture \hspace{1cm} 1442 \hspace{1cm} 155

\[ p(\text{with}|\text{v}) = \frac{607}{5156} = 0.118 > p(\text{with} \mid \text{n}) = 0.107 \]
What is going wrong?

• If you see a V NP PP sequence, then for the PP to attach to the V, the NP can’t also have a PP modifier
• Use parsing to integrate these cues
Breaking up the symbols

- Relax independence assumptions by encoding dependencies into the PCFG rules
- Mark possessive NPs as a new nonterminal, break down independence barrier

What are the most useful features to encode?
Part of a General Strategy: Horizontal Markovization (what do vanilla PCFGs have?)

• This merges states

Reduces grammar size, first-order Markov improves scores by 1%
Can we do this automatically?
Horizontal Markovization

- Ordinary PCFG: first-order vertical (depends on lefthand side nonterminal)
- Parent annotation: 2\textsuperscript{nd} order; improves from 73 to 77.8\%
- Could do more: grandparent, etc, yields 78\% but # of symbols starts getting large
## Results

<table>
<thead>
<tr>
<th>Vertical Order</th>
<th>Horizontal Markov Order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$h = 0$</td>
</tr>
<tr>
<td>$v = 1$ No annotation</td>
<td>71.27</td>
</tr>
<tr>
<td></td>
<td>(854)</td>
</tr>
<tr>
<td>$v \leq 2$ Sel. Parents</td>
<td>74.75</td>
</tr>
<tr>
<td></td>
<td>(2285)</td>
</tr>
<tr>
<td>$v = 2$ All Parents</td>
<td>74.68</td>
</tr>
<tr>
<td></td>
<td>(2984)</td>
</tr>
<tr>
<td>$v \leq 3$ Sel. GParents</td>
<td>76.50</td>
</tr>
<tr>
<td></td>
<td>(4943)</td>
</tr>
<tr>
<td>$v = 3$ All GParents</td>
<td>76.74</td>
</tr>
<tr>
<td></td>
<td>(7797)</td>
</tr>
</tbody>
</table>
Why only children can go wrong: unary rules ‘transmpute’ categories (here S to VP) so that a high prob VP rule gets used...

Solution: Mark unary rules without parent by –U
Marking the only child

• \(^{\wedge}U\) (external unary) “I am the only child.”
• \(-U\) (internal unary) “I have only one child.”
• On the preterminal level (POS → word), external unary mark-up helps with
  • demonstratives (that, this) vs. articles (a, the) — both labeled as DT in Penn TreeBank
  • adverbs (e.g., also vs. as well).
Part of speech tag splitting: POS tags too coarse!

- Parent annotation also for preterminal tags
- Splitting of IN tags into 6 linguistically motivated groups (prepositions vs. conjunctions vs. complementizers (that..); noun-modifying vs. primarily verb-modifying prepositions (of vs. as))
- Distinction between auxiliaries have and be. Special conjunction class containing but/But
- &. % get their own tags
A winner with this example

(a)

(b)
Bottom line?

- Using all these tricks, without lexicalization: F1 of 86.3%

- What does it mean to have an ‘unlexicalized’ grammar anyway?
- Can we find the right features ‘automatically’?
- Ans: Yes!
- Answer again: latent feature ‘discovery’ (to come): toss in the kitchen sink & let statistical method find some combo that works best
S-walked

NP-Sue

NNP-Sue

Sue

VP-walked

VBD-walked

walked

PP-into

IN-into

into

NP-store

DT

the

NN-store

store
Lexicalization of PCFGs (de Marcken, 1995, Magerman 95; Collins 96)

- Word-word affinities are useful for resolving some parsing ambiguities
- What PP modifies is partly captured in a PCFG rule – what isn’t captured?
So, this is what the work had done in the 1990s up to 2000

- Charniak (2000): To do better, we have to condition probabilities on the actual words of the sentence; this makes the probabilities tighter”

  \[ p(\text{VP} \rightarrow \text{V NP NP}) = 0.00151 \]

  \[ p(\text{VP} \rightarrow \text{V NP NP} | \text{said}) = 0.00001 \]

  \[ p(\text{VP} \rightarrow \text{V NP NP} | \text{give}) = 0.01980 \]

- From 73% boost to 88% precision/recall accuracy
• Simple lexicalized PCFG
• Probabilistic conditioning is ‘top down’ (as in generating with a PCFG), but actual computation is bottom-up
Charniak example

a. $h(\text{ead}) = \text{profits}$; $c(\text{ategory}) = \text{NP}$
b. $\text{ph} (\text{parent head}) = \text{rose}$;
   $\text{pc(parent category)} = \text{S}$
c. $p(h | \text{ph, c, pc})$
d. $P(r | h, c, pc)$
How lexicalization sharpens probabilities

- Different rule expansions (verb subcategories)

<table>
<thead>
<tr>
<th>Local Tree</th>
<th>Come</th>
<th>Take</th>
<th>Think</th>
<th>Want</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP → V</td>
<td>9.5%</td>
<td>2.6%</td>
<td>4.6%</td>
<td>5.7%</td>
</tr>
<tr>
<td>VP → V NP</td>
<td>1.1%</td>
<td>32.1</td>
<td>0.2</td>
<td>13.9</td>
</tr>
<tr>
<td>VP → V PP</td>
<td>34.5</td>
<td>3.1</td>
<td>7.1</td>
<td>0.3</td>
</tr>
<tr>
<td>VP → V Sbar</td>
<td>6.6</td>
<td>0.3</td>
<td>73.0</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → V S</td>
<td>2.2</td>
<td>1.3</td>
<td>4.8</td>
<td>70.8</td>
</tr>
<tr>
<td>VP → V NP S</td>
<td>0.1</td>
<td>5.7</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>VP → V PRT NP</td>
<td>0.3</td>
<td>5.8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VP → V PRT PP</td>
<td>6.1</td>
<td>1.5</td>
<td>0.3</td>
<td>0</td>
</tr>
</tbody>
</table>
Bi-lexical probabilities

- $p(\text{prices} \mid \text{n-plural}) = 0.013$
- $p(\text{prices} \mid \text{n-plural}, \text{NP}) = 0.013$
- $p(\text{prices} \mid \text{n-plural}, \text{NP}, \text{S}) = 0.025$
- $p(\text{prices} \mid \text{n-plural}, \text{NP}, \text{S}, \text{v-past}) = 0.052$
- $p(\text{prices} \mid \text{n-plural}, \text{NP}, \text{S}, \text{v-past}, \text{fell}) = 0.146$
Building the model: the issue of sparseness in the Penn Treebank – linear interpolation smoothing

\[ \hat{p}(h|ph, c, pc) = \lambda_1(e)p_{\text{MLE}}(h|ph, c, pc) \]
\[ = \lambda_2(e)p_{\text{MLE}}(h|C(ph), c, pc) \]
\[ = \lambda_3(e)p_{\text{MLE}}(h|c, pc) + \lambda_4(e)p_{\text{MLE}}(h|c) \]

Here, \( \lambda_i(e) \) is a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, and so on

\( C(ph) \) is the semantic class of the parent headword

You must use techniques like this to handle data sparseness
A concrete example from PTB

\[ p(\text{profits}|\text{rose}, \text{NP}, \text{S}) \quad p(\text{corp} | \text{profits}, \text{JJ}, \text{NP}) \]

\[
\begin{array}{lcccc}
p(h|ph,c,pc) & 0 & 0.245 \\
p(h|C(ph),c,pc) & 0.00352 & 0.0150 \\
p(h|c,pc) & 0.000627 & 0.00533 \\
p(h|c) & 0.000557 & 0.00418 \\
\end{array}
\]

- Uses highly conditioned estimates, smooths where necessary
- One can’t just use MLE estimates, too many 0’s
Sparserness and the PTB

- PTB: 1 million words of parsed text, a key resource
- But 1 million words is too little!
  965,000 phrases, but only 66 wh-phrases, and only 6 aren’t how many or how much
  how clever/original/sharp (at risk management)
- Most of the probabilities you want to compute, you can’t compute
Sparseness and PTB - bilexical

- Parse preferences depend on bi-lexical statistics: likelihoods of relationships between pairs of words (compound nouns, PP attachments)
- But these are very sparse, even on topics central to the Wall Street Journal:
  - Stocks plummeted 2 occurrences
  - Stocks stabilized 1 occurrence
  - Stocks skyrocketed 0 occurrences
  - Stocks discussed 0 occurrences
Complexity of Lexicalized PCFG parsing

Time charged:
- \( i, j, k \Rightarrow n^3 \)
- \( A, B, C \Rightarrow |G|^3 \)
- Naively, \( |G| \) becomes huge
- Any word in \( S \) as head
  \[ \Rightarrow |G| \rightarrow |G| \cdot n \]

Naïve version: running time is \( O(|G|^3 \times n^5) \)

This is what we actually have for Charniak, Collins, \( O(|G|^3 \times n^5) \), but use heuristics (beam search) to be faster
Initialization
1 for $i = 1 \ldots n$, $X \in N$
2 \[ \pi[i, i, i, X] = P(X(w_i) \rightarrow w_i | X(w_i)) \] // part-of-speech tagging
3 end for

Filling the table
4 for $\ell = 1 \ldots n - 1$, $i = 1 \ldots n - \ell$, $h = i \ldots i + \ell$, $X \in N$ // iterate over the table
5 \[ j = i + \ell; \quad p_{\text{max}} = 0 \]
6 for $s = h \ldots j$, $m = s + 1 \ldots j$, $Y \in N$, $Z \in N$ // left-headed case
7 \[ p = P(X(w_h) \rightarrow Y(w_m)Z | X(w_h)) \]
8 \[ \times \pi[i, s, h, Y] \times \pi[s + 1, j, m, Z] \]
9 if $p > p_{\text{max}}$ then
10 \[ p_{\text{max}} = p \]
11 \[ \text{split}[i, j, h, X] = (\text{left}, s, m, Y, Z) \]
12 end if
13 end for
14 for $s = i \ldots h - 1$, $m = i \ldots s$, $Y \in N$, $Z \in N$ // right-headed case
15 \[ p = P(X(w_h) \rightarrow Y(w_m)Z | X(w_h)) \]
16 \[ \times \pi[i, s, m, Y] \times \pi[s + 1, j, h, Z] \]
17 if $p > p_{\text{max}}$ then
18 \[ p_{\text{max}} = p \]
19 \[ \text{split}[i, j, h, X] = (\text{right}, s, m, Y, Z) \]
20 end if
21 end for
22 $\pi[i, j, h, X] = p_{\text{max}}$
23 end for

Selecting a root
24 $p_{\text{tree}} = 0$; $h_{\text{tree}} = \text{nil}$
25 for $h = 1 \ldots n$
26 \[ p = P_S(S(w_h)) \times \pi[1, n, h, S] \]
27 if $p > p_{\text{tree}}$ then
28 \[ p_{\text{tree}} = p; \quad h_{\text{tree}} = h \]
29 end if
30 end for
31 return $(p_{\text{tree}}, h_{\text{tree}}, \text{split})$
Initialization
1 for $i = 1 \ldots n$, $X \in N$
2 $\pi[i, i, i, X] = P(X(w_i) \rightarrow w_i \mid X(w_i))$ \hfill // part-of-speech tagging
3 end for

Filling the table
4 for $\ell = 1 \ldots n-1$, $i = 1 \ldots n-\ell$, $h = i \ldots i+\ell$, $X \in N$ \hfill // iterate over the table
5 $j = i + \ell$; $p_{max} = 0$
6 for $s = h \ldots j$, $m = s+1 \ldots j$, $Y \in N$, $Z \in N$ \hfill // left-headed case
7 $p = P(X(w_h) \rightarrow Y(w_h)Z(w_m) \mid X(w_h))$
8 $\quad \times \pi[i, s, h, Y] \times \pi[s+1, j, m, Z]$
9 if $p > p_{max}$ then
10 $\quad p_{max} = p$
11 $\quad split[i, j, h, X] = (\text{left}, s, m, Y, Z)$
12 end if
13 end for
Can one do better?

- Yes: can get back to $O(n^3)$, if we put some constraints on the grammars (more later)
Refining the node expansion possibilities

• Charniak (1997) expands each phrase structure tree in a single step
• This works well to capture dependencies between children nodes
• But bad because of sparseness
• A pure dependency, one child at a time model is worse
• You can find the ‘Goldilocks spot’ by various ‘in between’ models, eg, generating children as a Markov process on both sides of the head

- Generative models
- Underlying lexicalized PCFG has rules of the form:
  \[ P \rightarrow L_j L_{j-1} \ldots L_1 H R_1 \ldots R_{k-1} R_k \]
- Really a two-fold \( n \)-gram model: left- and right- of head
- A more elaborated set of grammar transforms and factorizations to deal with data sparseness and linguistic properties (not so much the latter)
- Each child is generated in turn: given that \( P \) has been generated, generate \( H \), then generate the modifying nonterminals from head-adjacent outwards, with some limited conditioning
Collins model overview

$L_i$ generated

$P(ARENT)(t_k, w_h)$ \hspace{1em} \text{[t=tag, w=word]}

\[ L_i \quad L_{i-1} \quad \ldots \quad L_1 \{\text{subcat}_L\} H(t_k, w_h) \]

$\Lambda$
M(odifying) nonterminals are generated in 2 steps

\[ P_M \quad S(VBD-sat) \]
\[ NP(NNP-\text{John}) \quad VP(VBD-sat) \]
\[ P_{Mw} \quad P_H \]
Smoothing for head words of modifying nonterminals

<table>
<thead>
<tr>
<th>Back off level</th>
<th>$P_{M_w}(w_{M_i} \ldots)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$M_i, t_{w_i}, \text{coord}, \text{punct}, P, H, w_k, t_k, \Lambda_M, \text{subcat}_\text{side}$</td>
</tr>
<tr>
<td>1</td>
<td>$M_i, t_{w_i}, \text{coord}, \text{punct}, P, H, t_k, \Lambda_M, \text{subcat}_\text{side}$</td>
</tr>
<tr>
<td>2</td>
<td>$t_w$</td>
</tr>
</tbody>
</table>

- Uses backoff, rather than interpolation
- Other parameter classes have similar or more elaborate backoff methods
Including ‘linguistics’?

• Successively more info about ‘linguistics’
  • Distance measure favors ‘close attachment’ (eg, *I said my dog died yesterday*)
  • Also punctuation
  • Category splitting: Distinguish base NP from full NP with post-modifier
  • Coordination feature
  • Model of subcategorization: arguments vs. adjuncts (required argument vs. optional: eg, *I gave John a book/I gave John a book on Friday*)
• What didn’t work: *What did John eat ___*
Bilexical statistics: is the use of maximal context of $P_{Mw}$ useful?

• Brain surgery…
• Suggestive: removed the bi-lexical information, got only a small drop in performance…
• Not convincing – was tried on a ‘black box’ emulation of the original ‘brain’
Choice of heads

- If not bilexical statistics, then surely the choice of heads is important to parser performance.
- Parsers performed decently even when all head rules were of the form, “if parent is $X$, choose left/rightmost child at random.”
- This emulation gives: 88.55% labeled precision and 88.80% labeled recall on section 00 of PTB, sentence length < 40 (compared to 89.9 & 90.1).
Use of maximal context of $P_{Mw}$

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Precision</th>
<th>Crossed Brackets</th>
<th>0 Crossed Brackets</th>
<th>&lt;3 Cross Brackets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Model</td>
<td>89.9</td>
<td>90.1</td>
<td>0.78</td>
<td>68.8</td>
<td>89.2</td>
</tr>
<tr>
<td>No bigrams</td>
<td>89.5</td>
<td>90.0</td>
<td>0.80</td>
<td>68.0</td>
<td>88.8</td>
</tr>
</tbody>
</table>

Test on section 00 of PTB, sentence length $< 40$
How often is the rich context for $P_{M_w}$ actually used?

<table>
<thead>
<tr>
<th>Back-off Level</th>
<th>Number of accesses</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3,257,309</td>
<td>1.49</td>
</tr>
<tr>
<td>1 (no bilexical)</td>
<td>24,294,084</td>
<td>11.0</td>
</tr>
<tr>
<td>2 (tags)</td>
<td>191,527,387</td>
<td>87.4</td>
</tr>
<tr>
<td>Total</td>
<td>219,078,780</td>
<td>100</td>
</tr>
</tbody>
</table>

# of times parsing engine was able to deliver a probability for the various back-off levels of the modifier-word generation model $P_M$, testing on sec. 00, trained on 02-21
Bilexical stats *are* used (Bikel, 2004)

- The 1.49% use of bilexical dependencies suggests they don’t play much of a role in parsing, but...
- The parser pursues many, many incorrect theories
- So, ask how often the parser uses bigram probabilities when pursuing its *top* scoring theory
- Answer this by doing the following:
  - Train as normal, on sections 2-21
  - Parse section 00
  - Feed these parse trees in as *constraints*
- Now bilexical use zooms up to 28.8%
- So: used often, but don’t affect parsing accuracy
- Why? One answer:
  
  Distributions that include head words usually so similar to ones that do not, so as to make almost no diff in accuracy
Automatic Annotation Induction?

[Matsuzaki et. al ’05, Prescher ’05]

• Advantages:
  • **Automatically learned:**
    Label *all* nodes with latent variables.
    Same number $k$ of subcategories for all categories.

• Disadvantages:
  • Grammar gets too large
  • Most categories are oversplit while others are undersplit.

<table>
<thead>
<tr>
<th>Model</th>
<th>$F1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
</tr>
</tbody>
</table>
Learning Latent Annotations (Petrov & Klein, 2006)

- Can you automatically find good symbols?
  - Brackets are known
  - Base categories are known
  - Induce subcategories
  - Split/merge refinement

Uses EM for trees, as sketched before
Overview

Limit of computational resources

- Hierarchical Training
- Adaptive Splitting
- Parameter Smoothing