Constituency Parsing (cont'd)  
+ PCFGs (if time)

CS 485, Fall 2023  
Applications of Natural Language Processing  
https://people.cs.umass.edu/~brenocon/cs485_f23/

Brendan O'Connor  
College of Information and Computer Sciences  
University of Massachusetts Amherst
Fill in the CYK dynamic programming table to parse the sentence below. In the bottom right corner, draw the two parse trees. Show the possible nonterminals in each cell. Optional: draw the backpointers too.

0
she  
1  eats  
2  fish  
3 with  
4 chopsticks  
5  

S \rightarrow NP VP  
NP \rightarrow she  
NP \rightarrow fish  
V \rightarrow eats  
P \rightarrow with  

Recursion dyn. prog.

\( \mathcal{O}(N^2) \) cells
For cell \([i,j]\)

For possible splitpoint \(k = (i+1) \ldots (j-1)\):

For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),

If exists rule \(A \rightarrow B \, C\),

\[\text{add}\, A\] to cell \([i,j]\)

How do we fill in \(C(1,2)\)?

[Example from Noah Smith]
How do we fill in \( C(1,2) \)?

Put together \( C(1,1) \) and \( C(2,2) \).

For cell \([i,j]\)

For possible splitpoint \( k=(i+1)\ldots(j-1) \):

For every \( B \) in \([i,k]\) and \( C \) in \([k,j]\),

If exists rule \( A \rightarrow B \ C \),

*add* \( A \) to cell \([i,j]\)

[Example from Noah Smith]
How do we fill in \( C(1,3) \)?

For cell \([i,j]\)

For possible splitpoint \( k = (i+1) \ldots (j-1) \):

For every \( B \) in \([i,k]\) and \( C \) in \([k,j]\),

If exists rule \( A \rightarrow B \ C \),

*add* \( A \) to cell \([i,j]\)

**Computational Complexity?**

[Example from Noah Smith]
For cell $[i,j]$
   For possible splitpoint $k=(i+1)\ldots(j-1)$:
      For every $B$ in $[i,k]$ and $C$ in $[k,j]$, 
      If exists rule $A \rightarrow B C$, 
      add $A$ to cell $[i,j]$

How do we fill in $C(1,3)$?

One way …

[Example from Noah Smith]
For cell \([i,j]\)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):  
For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),  
If exists rule \(A \rightarrow B C\),  
\textit{add} \(A\) to cell \([i,j]\)

How do we fill in \(C(1,3)\)?

\[\text{One way ...}\]
\[\text{Another way.}\]

[Example from Noah Smith]
How do we fill in $C(1,n)$?

For cell $[i,j]$
   For possible splitpoint $k=(i+1)\ldots(j-1)$:
      For every $B$ in $[i,k]$ and $C$ in $[k,j]$,  
      If exists rule $A \rightarrow B \cdot C$,  
      add $A$ to cell $[i,j]$ 

Computational Complexity?

[Example from Noah Smith]
How do we fill in $C(1, n)$?

$n - 1$ ways!

Computational Complexity?

$O(G \cdot n^3)$

$G = \text{grammar constant}$

[Example from Noah Smith]
Probabilistic CFGs

- Defines a probabilistic generative process for words in a sentence
- Can parse with a modified form of CKY
- How to learn? Fully supervised if you have a treebank

[J&M textbook]
PCFG as LM

- sample $p(w,y) = p(w|y) p(y)$
N-gram vs PCFG LM

• We also could sample from an n-gram (Markov) LM.... what differences should we expect?
PCFG as LM

- \( p(w,y) = \text{multiply all the expansion probabilities} \)

\[
\log p(w,y) = \text{sum of expansion 1-2 paths!}
\]
(P)CFG model, (P)CKY algorithm

• CKY: given CFG and sentence w
  • Does there exist at least one parse?
  • Enumerate parses (backpointers)

• Probabilistic CKY: given PCFG and sentence w
  • Most probable parse (“Viterbi parse”)
    \[ \hat{y} = \arg\max_y P(y \mid w) \]
  • Likelihood of sentence (“Inside algorithm”)
    \[ P(w) = \sum_y P(w \mid y) P(y) \]
• a PCFG with Penn Treebank’s nonterminals encodes overly strong conditional independence assumptions - big problems for both generation and parsing
• a bunch of tricks improve treebank-trained PCFGs to get better parsing performance
  • ~80% F1: "Treebank grammar" (PCFG directly trained on PTB)
  • ~90% F1: PCFG with clever non-terminal splitting
  • ~95% F1: state of the art (not PCFG)

Very sad 😞!
Better PCFG grammars

- Nonterminal splitting: because substitutability is too strong (e.g. “she” as subject vs object)

![Diagram of two grammatical trees comparing S to NP and VP attachments.]

Figure 11.5: A grammar that allows *she* to take the object position wastes probability mass on ungrammatical sentences.
Better PCFG grammars

- Parent annotation

Figure 11.8: Parent annotation in a CFG derivation
Better PCFG grammars

- Linguistically designed state splits

Figure 11.13: State-splitting creates a new non-terminal called NP-TMP, for temporal noun phrases. This corrects the PCFG parsing error in (a), resulting in the correct parse in (b).

[From Eisenstein (2017)]
Better PCFG grammars

- Lexicalization: encode semantic preferences

<table>
<thead>
<tr>
<th>Non-terminal</th>
<th>Direction</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>right</td>
<td>VP SBAR ADJP UCP NP</td>
</tr>
<tr>
<td>VP</td>
<td>left</td>
<td>VBD VBN MD VBZ TO VB VP VBG VBP ADJP NP</td>
</tr>
<tr>
<td>NP</td>
<td>right</td>
<td>N* EX $ CD QP PRP ...</td>
</tr>
<tr>
<td>PP</td>
<td>left</td>
<td>IN TO FW</td>
</tr>
</tbody>
</table>

Table 11.3: A fragment of head percolation rules

Figure 11.9: Lexicalization can address ambiguity on coordination scope (upper) and PP attachment (lower)
• a PCFG with Penn Treebank's nonterminals encodes overly strong conditional independence assumptions - big problems for both generation and parsing
• a bunch of tricks improve treebank-trained PCFGs to get better parsing performance
  • ~80% F1: "Treebank grammar" (PCFG directly trained on PTB)
  • ~90% F1: PCFG with clever non-terminal splitting
  • ~95% F1: state of the art (not PCFG)
• Parsing model accuracy: lots of ambiguity!!
  • PCFGs lack lexical information to resolve ambiguities (sneak in world knowledge?)
  • PCFGs that are successful parsers sneak in lexical information into the non-terminals ... but there are limits how much you can do
  • Next time: dependency parsing

• Practical guidance
  • $O(N)$ left-to-right incremental algorithms are more practical than CKY
  • Look carefully at parser's errors — are they tolerable for your application?