Constituency Parsing: CKY

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
Context-Free Grammar

- CFG describes a generative process for an (infinite) set of strings
  - 1. Nonterminal symbols
    - “S”: START symbol / “Sentence” symbol
  - 2. Terminal symbols: word vocabulary
  - 3. Rules (a.k.a. Productions). Practically, two types:

“Grammar”: one NT expands to >=1 NT
always one NT on left side of rule

Lexicon: NT expands to a terminal

\[
\begin{align*}
S & \rightarrow \text{NP VP} & \text{I + want a morning flight} \\
\text{NP} & \rightarrow \text{Pronoun} & \text{I} \\
& | \text{Proper-Noun} & \text{Los Angeles} \\
& | \text{Det Nominal} & \text{a + flight} \\
\text{Nominal} & \rightarrow \text{Nominal Noun} & \text{morning + flight} \\
& | \text{Noun} & \text{flights} \\
\text{VP} & \rightarrow \text{Verb} & \text{do} \\
& | \text{Verb NP} & \text{want + a flight} \\
& | \text{Verb NP PP} & \text{leave + Boston + in the morning} \\
& | \text{Verb PP} & \text{leaving + on Thursday} \\
\text{PP} & \rightarrow \text{Preposition NP} & \text{from + Los Angeles}
\end{align*}
\]

\[
\begin{align*}
\text{Noun} & \rightarrow \text{flights | breeze | trip | morning | ...} \\
\text{Verb} & \rightarrow \text{is | prefer | like | need | want | fly} \\
\text{Adjective} & \rightarrow \text{cheapest | non-stop | first | latest} \\
& | \text{other | direct | ...} \\
\text{Pronoun} & \rightarrow \text{me | I | you | it | ...} \\
\text{Proper-Noun} & \rightarrow \text{Alaska | Baltimore | Los Angeles} \\
& | \text{Chicago | United | American | ...} \\
\text{Determiner} & \rightarrow \text{the | a | an | this | these | that | ...} \\
\text{Preposition} & \rightarrow \text{from | to | on | near | ...} \\
\text{Conjunction} & \rightarrow \text{and | or | but | ...}
\end{align*}
\]
Constituent Parse Trees

\[
\begin{array}{c}
S \\
| \ NP \\
| | \ VP \\
| Pro \ Verb \\
| | | \ NP \\
| Det \ Nom \\
| | | | \ Noun \\
| | | | | \ Nom \\
| | | | | | \ flight \\
| | | | | | | \ morning \\
\end{array}
\]

Figure 12.4 The parse tree for “I prefer a morning flight” according to grammar \( \mathcal{L}_0 \).

Bracket notation

(12.2) \[ [S [NP [Pro I]] [VP [V prefer] [NP [Det a] [Nom [N morning] [Nom [N flight]]]]]] \]

<=> Set of non-terminal spans (start,end positions)
\{(NP, 0, 1), (VP, 1, 5), (NP, 2, 5), ...\}
Parsing with a CFG

• Task: given text and a CFG, answer:
  • Does there exist at least one parse?
  • Enumerate parses (backpointers)

• Problem: extremely high number of possible trees for a sentence, and even a large number of legal trees (licensed by the grammar) for a sentence
  • Many parsing algorithms have been invented to tackle this

• Cocke-Kasami-Younger algorithm (CKY)
  • Bottom-up dynamic programming:
    Find possible nonterminals for short spans of sentence, then possible combinations for higher spans
  • Maintains local ambiguity, representing many subtrees for each span. ("Packed forest" representation)
  • Provably finds all possible parse trees (legal derivations), and correctly says when none exist.
  • Requires converting to Chomsky Normal Form (binarization)
Chomsky Normal Form
For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..<j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j] (Recognizer)
    ... or ...
    add (A,B,C, k) to cell [i,j] (Parser)
For cell \([i,j]\) (loop through them bottom-up)

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 \([i,j]\)  \((\text{Recognizer})\)

... or ...

\(\text{add } (A,B,C,\ k)\) to cell \([i,j]\)  \((\text{Parser})\)

\(Grammar\):

\(Adj \rightarrow \) yummy

\(NP \rightarrow \) foods

\(NP \rightarrow \) store

\(NP \rightarrow \) NP NP

\(NP \rightarrow \) Adj NP

\(Recognizer\): per span, record list of possible nonterminals

\(Parser\): per span, record possible ways the nonterminal was constructed.
For cell $[i,j]$ (loop through them bottom-up)
For possible split point $k=(i+1)..(j-1)$:
For every $B$ in $[i,k]$ and $C$ in $[k,j]$,
If exists rule $A \rightarrow B C$,
  add $A$ to cell $[i,j]$  (Recognizer)
... or ...
  add $(A,B,C, \ k)$ to cell $[i,j]$  (Parser)

**Recognizer**: per span, record list of possible nonterminals

**Parser**: per span, record possible ways the nonterminal was constructed.

**Grammar**
- Adj $\rightarrow$ yummy
- NP $\rightarrow$ foods
- NP $\rightarrow$ store
- NP $\rightarrow$ NP NP
- NP $\rightarrow$ Adj NP
she eats fish with chopsticks
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.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>eats</td>
<td>fish</td>
<td>with</td>
<td>chopsticks</td>
<td>S → NP VP, NP → NP PP, VP → V NP, VP → VP PP, PP → P NP, NP → she, NP → fish, NP → fork, NP → chopsticks, V → eats, V → fish, P → with</td>
</tr>
</tbody>
</table>

Brendan O’Connor and Andrew McCallum, UMass Amherst
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 \( C(1,2) \)?
How do we fill in $C(1,2)$?

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

For cell $[i,j]$

For possible splitpoint $k=(i+1) ...(j-1)$:

For every $B$ in $[i,k]$ and $C$ in $[k,j]$,

If exists rule $A \rightarrow B \ C$, 

add $A$ to cell $[i,j]$
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 \(C(1,3)\)?
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 \(A \rightarrow B\ C\),

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

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

\textbf{One way …}
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 \(C(1,3)\)?

One way …
Another 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 \(A \rightarrow B C\),
add \(A\) to cell \([i,j]\)

Computational Complexity?

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 \(A \rightarrow B \ C\),  
\textit{add} \(A\) to cell \([i,j]\)

\begin{align*}
\text{How do we fill in } C(1,n) ? \\
& \text{ } \text{n - 1 ways!}
\end{align*}

\text{Computational} \\
\text{Complexity ?}

\text{O}(G \ n^3) \\
G = \text{grammar constant}

\text{[Example from Noah Smith]}
Probabilistic CFGs

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S \rightarrow NP VP</td>
<td>[.80]</td>
</tr>
<tr>
<td>S \rightarrow Aux NP VP</td>
<td>[.15]</td>
</tr>
<tr>
<td>S \rightarrow VP</td>
<td>[.05]</td>
</tr>
<tr>
<td>NP \rightarrow Pronoun</td>
<td>[.35]</td>
</tr>
<tr>
<td>NP \rightarrow Proper-Noun</td>
<td>[.30]</td>
</tr>
<tr>
<td>NP \rightarrow Det Nominal</td>
<td>[.20]</td>
</tr>
<tr>
<td>NP \rightarrow Nominal</td>
<td>[.15]</td>
</tr>
<tr>
<td>Nominal \rightarrow Noun</td>
<td>[.75]</td>
</tr>
<tr>
<td>Nominal \rightarrow Nominal Noun</td>
<td>[.20]</td>
</tr>
<tr>
<td>Nominal \rightarrow Nominal PP</td>
<td>[.05]</td>
</tr>
<tr>
<td>VP \rightarrow Verb</td>
<td>[.35]</td>
</tr>
<tr>
<td>VP \rightarrow Verb NP</td>
<td>[.20]</td>
</tr>
<tr>
<td>VP \rightarrow Verb NP PP</td>
<td>[.10]</td>
</tr>
<tr>
<td>VP \rightarrow Verb PP</td>
<td>[.15]</td>
</tr>
<tr>
<td>VP \rightarrow Verb NP NP</td>
<td>[.05]</td>
</tr>
<tr>
<td>VP \rightarrow VP PP</td>
<td>[.15]</td>
</tr>
<tr>
<td>PP \rightarrow Preposition NP</td>
<td>[1.0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Det \rightarrow that</td>
<td>[.10]</td>
</tr>
<tr>
<td>Det \rightarrow a</td>
<td>[.30]</td>
</tr>
<tr>
<td>Det \rightarrow the</td>
<td>[.60]</td>
</tr>
<tr>
<td>Noun \rightarrow book</td>
<td>[.10]</td>
</tr>
<tr>
<td>Noun \rightarrow flight</td>
<td>[.30]</td>
</tr>
<tr>
<td>Noun \rightarrow meal</td>
<td>[.15]</td>
</tr>
<tr>
<td>Noun \rightarrow money</td>
<td>[.05]</td>
</tr>
<tr>
<td>Noun \rightarrow flights</td>
<td>[.40]</td>
</tr>
<tr>
<td>Noun \rightarrow dinner</td>
<td>[.10]</td>
</tr>
<tr>
<td>Verb \rightarrow book</td>
<td>[.30]</td>
</tr>
<tr>
<td>Verb \rightarrow include</td>
<td>[.30]</td>
</tr>
<tr>
<td>Verb \rightarrow prefer</td>
<td>[.40]</td>
</tr>
<tr>
<td>Pronoun \rightarrow I</td>
<td>[.40]</td>
</tr>
<tr>
<td>Pronoun \rightarrow she</td>
<td>[.05]</td>
</tr>
<tr>
<td>Pronoun \rightarrow me</td>
<td>[.15]</td>
</tr>
<tr>
<td>Pronoun \rightarrow you</td>
<td>[.40]</td>
</tr>
<tr>
<td>Proper-Noun \rightarrow Houston</td>
<td>[.60]</td>
</tr>
<tr>
<td>Proper-Noun \rightarrow TWA</td>
<td>[.40]</td>
</tr>
<tr>
<td>Aux \rightarrow does</td>
<td>[.60]</td>
</tr>
<tr>
<td>Aux \rightarrow can</td>
<td>[.40]</td>
</tr>
<tr>
<td>Preposition \rightarrow from</td>
<td>[.30]</td>
</tr>
<tr>
<td>Preposition \rightarrow to</td>
<td>[.30]</td>
</tr>
<tr>
<td>Preposition \rightarrow on</td>
<td>[.20]</td>
</tr>
<tr>
<td>Preposition \rightarrow near</td>
<td>[.15]</td>
</tr>
<tr>
<td>Preposition \rightarrow through</td>
<td>[.05]</td>
</tr>
</tbody>
</table>

- Defines a probabilistic generative process for words in a sentence
- (How to learn? Fully supervised with a treebank...)
General Electric Co. said it signed a contract with the developers of the Ocean State Power project for the second phase of an independent power plant, for $400 million, which is being built in Burrillville, R.I.
PCFG as LM
Is a PCFG a good LM? Yes...
Is a PCFG a good LM? No...
(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 $w$
  - Likelihood of sentence $P(w)$
  - Most probable parse ("Viterbi parse")
    \[
    \text{argmax}_y P(y \mid w) = \text{argmax}_y P(y, w)
    \]
• Parsing model accuracy: lots of ambiguity!!
  • PCFGs lack lexical information to resolve ambiguities (sneak in world knowledge?)
  • Modern constituent parsers: enrich PCFG with lexical information and fine-grained nonterminals
  • Modern dependency parsers: effectively the same trick

• Parsers’ computational efficiency
  • Grammar constant; pruning & heuristic search
  • $O(N^3)$ for CKY (ok? depends...)
  • $O(N)$ left-to-right incremental algorithms

• What was the syntactic training data?