Natural Language Processing

PCFG Parsing
The CKY algorithm

• We are interested in an efficient algorithm that can compute a most probable parse tree for the sentence \( w \) given the grammar \( G \).

• This task is also known as Viterbi parsing.

• The CKY algorithm solves this problem in time \( O(|G||w|^3) \).

• Presupposes a Chomsky normal form grammar:
  
  \[ C \rightarrow w_i \text{ (preterminal)} \]
  
  \[ C \rightarrow C_1 C_2 \text{ (binary)} \]
Dynamic programming

General idea:
Do not work with individual objects, but group objects into classes that share relevant properties, and manipulate only these classes.
The CKY algorithm

Signatures

covers all words between $min$ and $max$

$[min, max, C]$
Fencepost positions

We view the sentence $w$ as a fence with $n$ holes, one hole for each word $w_i$, and we number the fenceposts from 0 till $n$. 
The problem, reformulated

- **Original problem:**
  Find a most probable parse tree \( t \)
  for the sentence \( w \) according to the grammar \( G \).

- **Reformulated problem:**
  Find a most probable parse tree \( t \)
  with signature \([0, |w|, S]\).
The CKY algorithm

Central property

• Populate a table (chart) that maps signatures to parse trees.

• Ensure that chart[min][max][C] = t if and only if t is a most probable parse tree with signature [min, max, C].

• Return the entry chart[0][|w|][S].
Treebank grammars

• Given a treebank, we can
  • extract a CFG from the trees
  • estimate rule probabilities using MLE

• Given a plain text corpus and a CFG, we can
  • estimate rule probabilities using EM
Problems with treebank grammars

- Insensitive to structural context

<table>
<thead>
<tr>
<th>Tree context</th>
<th>NP → NP PP</th>
<th>NP → DT NN</th>
<th>NP → PRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anywhere</td>
<td>11%</td>
<td>9%</td>
<td>6%</td>
</tr>
<tr>
<td>NP under S</td>
<td>9%</td>
<td>9%</td>
<td>21%</td>
</tr>
<tr>
<td>NP under VP</td>
<td>23%</td>
<td>7%</td>
<td>4%</td>
</tr>
</tbody>
</table>

- Insensitive to lexical relations

*book a flight to LA vs drive a car to LA*
Treebank transformations

- Problems arise from a mismatch between standard treebank representations and the independence assumptions of the PCFG model
- Transform treebanks to fit the PCFG model
  - Parent annotation – incorporate structural context in nonterminal symbols
  - Lexicalization – incorporate lexical information in nonterminal symbols
Parent annotation

• Replace A by $A^B$ when B is parent of A

• For example:

\[
\begin{align*}
\text{NP}^\text{S} & \rightarrow \text{Det}^\text{NP} \text{ Noun}^\text{NP} \\
\text{NP}^\text{S} & \rightarrow \text{Pro}^\text{NP} \\
\text{NP}^\text{VP} & \rightarrow \text{Det}^\text{NP} \text{ Noun}^\text{NP} \\
\text{NP}^\text{S} & \rightarrow \text{Pro}^\text{NP}
\end{align*}
\]
Lexicalization

• Replace A by A(w) when w is the lexical head of A
• For example:

  VP(book) → Verb(book) NP(flight)
  VP(book) → Verb(book) NP(flight) PP(to)
  Nom(flight) → Noun(flight) PP(to)
  VP(drive) → Verb(drive) NP(car)
  VP(drive) → Verb(drive) NP(car) PP(to)
  Nom(car) → Noun(car) PP(to)
The state of the art

• Evaluation of PCFG parsers:
  • Recall and precision of phrases
  • Manually annotated gold standard
• Empirical results on WSJ for English:
  • Plain treebank grammars ~ 75% R/P
  • Add parent annotation ~ 80% R/P
  • Add lexicalization ~ 85% R/P
  • Add a few more tricks ~ 90% R/P