Treebank Parsing in 20 sec

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn’t work well):

```
ROOT → S 1
S → NP VP . 1
NP → PRP 1
VP → VBD ADJP 1
```

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers without lexicalization.

Treebank Sentences

```
( (S (NP-SB) The move) (VP followed (NP (NP a round) (PP of (NP (NP similar increases) (PP by (NP other lenders)) (PP against (NP Arizona real estate loans))))))
, (S-ADV (NP-SB) -> (VP reflecting (NP (NP a continuing decline) (PP-LOC in (NP that market))))))
,)
```

Context-Free Grammars

- A context-free grammar is a tuple <N, T, S, R>
  - N: the set of non-terminals
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
  - T: the set of terminals (the words)
    - Not usually the sentence non-terminal S
  - S: the start symbol
    - Often written as ROOT or TOP
  - R: the set of rules
    - Of the form X → Y1 Y2 … Yk, with X, Yi ∈ N
    - Examples: S → NP VP, VP → VP CC VP
    - Also called rewrites, productions, or local trees

Example CFG

- Can just write the grammar (rules with non-terminal LHSs) and lexicon (rules with pre-terminal LHSs)

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT → S</td>
<td>JJ → new</td>
</tr>
<tr>
<td>S → NP VP</td>
<td>NN → art</td>
</tr>
<tr>
<td>VP → VBD</td>
<td>NNS → critics</td>
</tr>
<tr>
<td>VP → VBP NP</td>
<td>NNS → reviews</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>NNS → computers</td>
</tr>
<tr>
<td>PP → IN NP</td>
<td>VBP → write</td>
</tr>
</tbody>
</table>

N-Ary Rules, Grammar States

- Often we want to write grammar rules like
  - VP → VBD NP PP PP
which are not binary.

- We can work with these rules by introducing new intermediate symbols (states) into our grammar:
Treebank Grammar Scale

- Treebank grammars can be enormous!
  - As a set of FSTs, the raw grammar has ~10K states (why?).
  - Better parsers usually make the grammars larger, not smaller.

PCFGs and Independence

- Symbols in a PCFG define independence assumptions:

  \[ \begin{align*}
    S &\rightarrow NP \ VP \\
    NP &\rightarrow DT \ NN \\
    S &\rightarrow NP \ VP \\
    NP &\rightarrow DT \ NN
  \end{align*} \]

  - At any node, the material inside that node is independent of the material outside that node, given the label of that node.
  - Any information that statistically connects behavior inside and outside a node must flow through that node.

Non-Independence I

- Independence assumptions are often too strong.

  
<table>
<thead>
<tr>
<th></th>
<th>All NPs</th>
<th>NPs under S</th>
<th>NPs under VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP PP DT NN PRP</td>
<td>11% 9% 6%</td>
<td>9% 9%</td>
<td>21% 7% 4%</td>
</tr>
</tbody>
</table>

  - 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!

Non-Independence II

- Who cares?
  - NB, HMMs, all make false assumptions!
  - For generation, consequences would be obvious.
  - For parsing, does it impact accuracy?

  - Symptoms of overly strong assumptions:
    - Rewrites get used where they don’t belong.
    - Rewrites get used too often or too rarely.

Breaking Up the Symbols

- We can relax independence assumptions by encoding dependencies into the PCFG symbols:

  Parent annotation (Johnson 98)

    |          | Marking possessive NPs |
    |----------|-------------------------|
    | NP PP DT NN PRP | NPs under S |

  Annotations

  - Annotations split the grammar categories into sub-categories (in the original sense).
  - Conditioning on history vs. annotating
    - \( P(NP^S \rightarrow PRP) \) is a lot like \( P(NP \rightarrow PRP | S) \)
    - \( P(NP-POS \rightarrow NNP-POS) \) isn’t history conditioning.
  - Feature / unification grammars vs. annotation
    - Can think of a symbol like \( NP^NP-POS \) as \( NP [parent:NP, +POS] \)
  - After parsing with an annotated grammar, the annotations are then stripped for evaluation.

Annotations

<table>
<thead>
<tr>
<th></th>
<th>Marking possessive NPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP PP DT NN PRP</td>
<td>NPs under S</td>
</tr>
</tbody>
</table>

  Annotations split the grammar categories into sub-categories (in the original sense).

  - Conditioning on history vs. annotating
    - \( P(NP^S \rightarrow PRP) \) is a lot like \( P(NP \rightarrow PRP | S) \)
    - \( P(NP-POS \rightarrow NNP-POS) \) isn’t history conditioning.
  - Feature / unification grammars vs. annotation
    - Can think of a symbol like \( NP^NP-POS \) as \( NP [parent:NP, +POS] \)
  - After parsing with an annotated grammar, the annotations are then stripped for evaluation.
Lexicalization

- Lexical heads important for certain classes of ambiguities (e.g., PP attachment).
- Lexicalizing grammar creates a much larger grammar. (cf. next week)
  - Sophisticated smoothing needed
  - Smarter parsing algorithms
  - More data needed
- How necessary is lexicalization?
  - Bilexical vs. monolexical selection
  - Closed vs. open class lexicalization

Unlexicalized PCFGs

- What is meant by an “unlexicalized” PCFG?
  - Grammar not systematically specified to the level of lexical items
    - NP [stocks] is not allowed
    - NP^S-CC is fine
  - Closed vs. open class words (NP^S [the])
  - Long tradition in linguistics of using function words as features or markers for selection
  - Contrary to the bilexical idea of semantic heads
  - Open-class selection really a proxy for semantics
- Honesty checks:
  - Number of symbols: keep the grammar very small
  - No smoothing: over-annotating is a real danger
  - No smoothing is a bad idea – this use is rhetorical!

Typical Experimental Setup

- Corpus: Penn Treebank, WSJ
  - Training: sections 02-21
  - Development: section 22 (here, first 20 files)
  - Test: section 23
- Accuracy – F1: harmonic mean of per-node labeled precision and recall.
- Here: also size – number of symbols in grammar.
  - Passive / complete symbols: NP, NP^S
  - Active / incomplete symbols: NP → NP CC

Multiple Annotations

- Each annotation done in succession
  - Order does matter
  - Too much annotation and we’ll have sparsity issues (where?).

Horizontal Markovization

- Order 1
- Order \( \infty \)

Vertical Markovization

- Order 1
- Order 2
Vertical and Horizontal

Examples:
- Raw treebank: $v=1$, $h=\infty$
- Johnson 98: $v=2$, $h=\infty$
- Collins 99: $v=2$, $h=2$
- Best F1: $v=3$, $h=2v$

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>77.8</td>
<td>7.5K</td>
</tr>
</tbody>
</table>

Unary Splits

- Problem: unary rewrites used to transmute categories so a high probability rule can be used.
- Solution: Mark unary rewrite sites with -U

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>77.8</td>
<td>7.5K</td>
</tr>
<tr>
<td>UNARY</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
</tbody>
</table>

Tag Splits

- Problem: Treebank tags are too coarse.
- Example: Sentential, PP, and other prepositions are all marked IN.
- Partial Solution:
  - Subdivide the IN tag.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
<tr>
<td>SPLIT-IN</td>
<td>80.3</td>
<td>8.1K</td>
</tr>
</tbody>
</table>

Other Tag Splits

- UNARY-DT: mark demonstratives as DT^U ("the X" vs. "those")
- UNARY-RB: mark phrasal adverbs as RB^U ("quickly" vs. "very")
- TAG-PA: mark tags with non-canonical parents ("not" is an RB^VP)
- SPLIT-AUX: mark auxiliary verbs with –AUX [cf. Charniak 97]
- SPLIT-CC: separate "but" and "&" from other conjunctions
- SPLIT-%: "%" gets its own tag.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>81.7</td>
<td>9.1K</td>
</tr>
<tr>
<td>SPLIT-CC</td>
<td>81.8</td>
<td>9.3K</td>
</tr>
</tbody>
</table>

Treebank Splits

- The treebank comes with some annotations (e.g., -LOC, -SUBJ, etc).
- Whole set together hurt the baseline.
- One in particular is very useful (NP-TMP) when pushed down to the head tag (why?).
- Can mark gapped S nodes as well.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>81.8</td>
<td>9.3K</td>
</tr>
<tr>
<td>NP-TMP</td>
<td>82.2</td>
<td>9.6K</td>
</tr>
<tr>
<td>GAPPED-S</td>
<td>82.3</td>
<td>9.7K</td>
</tr>
</tbody>
</table>

Yield Splits

- Problem: sometimes the behavior of a category depends on something inside its future yield.
- Examples:
  - Possessive NPs
  - Finite vs. infinite VPs
  - Lexical heads
- Solution: annotate future elements into nodes.
  - Lexicalized grammars do this (in very careful ways – why?).

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>82.3</td>
<td>9.7K</td>
</tr>
<tr>
<td>POSS-NP</td>
<td>83.1</td>
<td>9.8K</td>
</tr>
<tr>
<td>SPLIT-VP</td>
<td>85.7</td>
<td>10.5K</td>
</tr>
</tbody>
</table>
Distance / Recursion Splits

- Problem: vanilla PCFGs cannot distinguish attachment heights.
- Solution: mark a property of higher or lower sites:
  - Contains a verb.
  - Is (non)-recursive.
  - Base NPs [cf. Collins 99]
  - Right-recursive NPs

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>85.7</td>
<td>10.5K</td>
</tr>
<tr>
<td>BASE-NP</td>
<td>86.0</td>
<td>11.7K</td>
</tr>
<tr>
<td>DOMINATES-V</td>
<td>86.9</td>
<td>14.1K</td>
</tr>
<tr>
<td>RIGHT-REC-NP</td>
<td>87.0</td>
<td>15.2K</td>
</tr>
</tbody>
</table>

Some Test Set Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>LP</th>
<th>LR</th>
<th>F1</th>
<th>CB</th>
<th>0 CB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magerman 95</td>
<td>84.9</td>
<td>84.6</td>
<td><strong>84.7</strong></td>
<td>1.26</td>
<td>56.6</td>
</tr>
<tr>
<td>Collins 96</td>
<td>86.3</td>
<td>85.8</td>
<td><strong>86.0</strong></td>
<td>1.14</td>
<td>59.9</td>
</tr>
<tr>
<td>Unlexicalized</td>
<td>86.9</td>
<td>85.7</td>
<td><strong>86.3</strong></td>
<td>1.10</td>
<td>60.3</td>
</tr>
<tr>
<td>Charniak 97</td>
<td>87.4</td>
<td>87.5</td>
<td><strong>87.4</strong></td>
<td>1.00</td>
<td>62.1</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.7</td>
<td>88.6</td>
<td><strong>88.6</strong></td>
<td>0.90</td>
<td>67.1</td>
</tr>
</tbody>
</table>

- Beats "first generation" lexicalized parsers.
- Lots of room to improve – more complex models next.