The grammar: Binary, no epsilons,

```
S → NP VP 0.9  N → people 0.5
S → VP 0.1   N → fish 0.2
VP → V NP 0.5  V → rods 0.1
VP → V NP 0.1  NP → people 0.1
VP → V NP 0.1  V → fish 0.6
VP → V NP 0.1  V → tanks 0.3
PP → P NP 1.0  P → with 1.0
```

The grammar:

```
S → NP VP 0.9  N → people 0.5
S → VP 0.1   N → fish 0.2
VP → V NP 0.5  V → rods 0.1
VP → V NP 0.1  NP → people 0.1
VP → V NP 0.1  V → fish 0.6
VP → V NP 0.1  V → tanks 0.3
PP → P NP 1.0  P → with 1.0
```
Evaluating constituency parsing

Gold standard brackets:
S-(0:11), NP-(0:2), VP-(2:9), VP-(3:10), NP-(4:8), PP-(6:9), NP-(7:3), NP-(9:10)

Candidate brackets:
S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:8), PP-(6:10), NP-(7,10)

Labeled Precision: 3/7 = 42.9%
Labeled Recall: 3/8 = 37.5%
PPOS Tagging Accuracy: 11/11 = 100.0%

How good are PCFGs?

• Penn WSJ parsing accuracy: about 73% LP/LR F1
• Robust
  • Usually admit everything, but with low probability
  • Partial solution for grammar ambiguity
    • A PCFG gives some idea of the plausibility of a parse
      • But not so good because the independence assumptions are too strong
    • Give a probabilistic language model
      • But in the simple case it performs worse than a trigram model
    • The problem seems to be that PCFGs lack the lexicalization of a trigram model

(Head) Lexicalization of PCFGs

[Magarman 1995, Collins 1997; Charniak 1997]

• The head word of a phrase gives a good representation of the phrase’s structure and meaning
• Puts the properties of words back into a PCFG

(Head) Lexicalization of PCFGs

[Magarman 1995, Collins 1997; Charniak 1997]

• The head word of a phrase gives a good representation of the phrase’s structure and meaning
• Puts the properties of words back into a PCFG

(Head) Lexicalization of PCFGs

[Magarman 1995, Collins 1997; Charniak 1997]

• The head word of a phrase gives a good representation of the phrase’s structure and meaning
• Puts the properties of words back into a PCFG

(Head) Lexicalization of PCFGs

[Magarman 1995, Collins 1997; Charniak 1997]

• Word-to-word affinities are useful for certain ambiguities
  • PP attachment is now (partly) captured in a local PCFG rule.
    • Think about: What useful information isn’t captured?
  • Also useful for: coordination scope, verb complement patterns
Lexicalized parsing was seen as the parsing breakthrough of the late 1990s

- Eugene Charniak, 2000 JHU workshop: “To do better, it is necessary to condition probabilities on the actual words of the sentence. This makes the probabilities much tighter:
  - $p(VP \to V \text{ NP} = 0.00151$
  - $p(VP \to V \text{ NP} \mid \text{ said}) = 0.00001$
  - $p(VP \to V \text{ NP} \mid \text{ gave}) = 0.01980$

- Michael Collins, 2003 COLT tutorial: “Lexicalized Probabilistic Context-Free Grammars … perform vastly better than PCFGs (88% vs. 73% accuracy)”

Lexicalization of PCFGs: Charniak (1997)

- A very straightforward model of a lexicalized PCFG
- Probabilistic conditioning is “top-down” like a regular PCFG
  - But actual parsing is bottom-up, somewhat like the CKY algorithm we saw

Charniak (1997) example

Lexicalization models argument selection by sharpening rule expansion probabilities

- The probability of different verbal complement frames (i.e., “subcategorizations”) depends on the verb:

<table>
<thead>
<tr>
<th>Local Tree</th>
<th>comp</th>
<th>take</th>
<th>think</th>
<th>want</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP \to V</td>
<td>0.95</td>
<td>0.04</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>VP \to VP</td>
<td>0.05</td>
<td>0.15</td>
<td>0.20</td>
<td>0.60</td>
</tr>
<tr>
<td>VP \to V NP</td>
<td>0.00</td>
<td>0.20</td>
<td>0.80</td>
<td>0.00</td>
</tr>
<tr>
<td>VP \to V PP</td>
<td>0.05</td>
<td>0.70</td>
<td>0.00</td>
<td>0.25</td>
</tr>
<tr>
<td>VP \to V SBAR</td>
<td>0.01</td>
<td>0.20</td>
<td>0.79</td>
<td>0.00</td>
</tr>
</tbody>
</table>

  “Nonexistent” probabilities

Lexicalization sharpens probabilities:
Predicting heads

“Bilexical probabilities”

- $P(\text{prices} \mid \text{n-plural}) = .013$
- $P(\text{prices} \mid \text{n-plural, NP}) = .013$
- $P(\text{prices} \mid \text{n-plural, NP, S}) = .025$
- $P(\text{prices} \mid \text{n-plural, NP, S, v-past}) = .052$
- $P(\text{prices} \mid \text{n-plural, NP, S, v-past, fell}) = .146$

Charniak (1997) linear interpolation/shrinkage

- $P(\text{hph}, c, pc) = \lambda_1(\text{e})P_{\text{unl}}(\text{hph}, c, pc) + \lambda_2(\text{e})P_{\text{lad}}(\text{h}, \text{c}, pc) + \lambda_3(\text{e})P_{\text{lad}}(\text{h}, \text{c}, \text{pc})$
- $\lambda_1(\text{e})$ is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- $C(\text{ph})$ is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction
Charniak (1997) shrinkage example

\[
P(h(\text{prf}(\text{rose}, \text{NP}), S)) = P(\text{corp}(\text{prf}, \text{JJ}), \text{NP})
\]

- \(P(h(c)) = 0.000537, P(h(c,c)) = 0.000418\)

- Allows utilization of richly conditioned estimates, but smoothes when sufficient data is unavailable
- One can't just use MLEs; one commonly sees previously unseen events, which would have probability 0.

Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations ("arrows") called dependencies. The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.).

Methods of Dependency Parsing

1. Dynamic programming (like in the CKY algorithm) You can do it similarly to lexicalized PCFG parsing: an O(n^3) algorithm Eisner (1996) gives a clever algorithm that reduces the complexity to O(n^2), by producing parse items with heads at the ends rather than in the middle.

2. Graph algorithms You create a maximum spanning tree for a sentence. McDonald et al.'s (2005) MSTParser scores dependencies independently using a ML classifier (he uses MIRA, for online learning, but it could be MaxEnt).

3. Constraint Satisfaction Edges are eliminated that don't satisfy hard constraints. Karlsson (1990), etc.


Relation between phrase structure and dependency structure

- A dependency grammar has a notion of a head. Officially, CFGs don't.
- But modern linguistic theory and all modern statistical parsers (Charniak, Collins, Stanford, ...) do, via hand-written phrase "head rules":
  1. The head of a Noun Phrase is a noun/number/adj/....
  2. The head of a Verb Phrase is a verb/modal/....
- The head rules can be used to extract a dependency parse from a CFG parse.

Dependency Conditioning Preferences

What are the sources of information for dependency parsing?

1. Bilexical affinities
2. Dependency distance
3. Intervening material
4. Valency of heads