PCFGs

A PCFG $G$ consists of the usual parts of a CFG

- A set of terminals, \( \{w^k\}, k = 1, \ldots, V \)
- A set of nonterminals, \( \{N^i\}, i = 1, \ldots, n \)
- A designated start symbol, $N^1$
- A set of rules, \( \{N^i \rightarrow \zeta^j\} \), (where $\zeta^j$ is a sequence of terminals and nonterminals)

and

- A corresponding set of probabilities on rules such that:

\[
\forall i \sum_j P(N^i \rightarrow \zeta^j) = 1
\]

PCFG Probability of a String

\[
P(w_1^n) = \sum_t P(w_1^n, t) \quad t \text{ a parse of } w_1^n
= \sum_{(t \text{yield}(t) = w_1^n)} P(t)
\]

A Simple PCFG (in CNF)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>1.0</td>
</tr>
<tr>
<td>$PP \rightarrow P \ NP$</td>
<td>1.0</td>
</tr>
<tr>
<td>$VP \rightarrow V \ NP$</td>
<td>0.7</td>
</tr>
<tr>
<td>$VP \rightarrow VP \ PP$</td>
<td>0.3</td>
</tr>
<tr>
<td>$P \rightarrow with$</td>
<td>1.0</td>
</tr>
<tr>
<td>$V \rightarrow saw$</td>
<td>1.0</td>
</tr>
<tr>
<td>$NP \rightarrow NP \ PP$</td>
<td>0.4</td>
</tr>
<tr>
<td>$NP \rightarrow astronomers$</td>
<td>0.1</td>
</tr>
<tr>
<td>$NP \rightarrow ears$</td>
<td>0.18</td>
</tr>
<tr>
<td>$NP \rightarrow saw$</td>
<td>0.04</td>
</tr>
<tr>
<td>$NP \rightarrow stars$</td>
<td>0.18</td>
</tr>
<tr>
<td>$NP \rightarrow telescopes$</td>
<td>0.1</td>
</tr>
<tr>
<td>$VP \rightarrow VP \ PP$</td>
<td>0.7</td>
</tr>
<tr>
<td>$NP \rightarrow with$</td>
<td>1.0</td>
</tr>
<tr>
<td>$NP \rightarrow stars$</td>
<td>0.18</td>
</tr>
<tr>
<td>$PP \rightarrow P \ NP$</td>
<td>0.7</td>
</tr>
<tr>
<td>$PP \rightarrow stars$</td>
<td>0.18</td>
</tr>
<tr>
<td>$PP \rightarrow with$</td>
<td>1.0</td>
</tr>
<tr>
<td>$PP \rightarrow ears$</td>
<td>0.18</td>
</tr>
</tbody>
</table>
Assumptions of PCFGs

1. Place invariance (like time invariance in HMM):
   \[ \forall k \ P(N_{j_{k(k+c)}}^I \rightarrow \zeta) \text{ is the same} \]
2. Context-free:
   \[ P(N_{j_{kl}}^I \rightarrow \zeta | \text{words outside } w_k \ldots w_l) = P(N_{j_{kl}}^I \rightarrow \zeta) \]
3. Ancestor-free:
   \[ P(N_{j_{kl}}^I \rightarrow \zeta | \text{ancestor nodes of } N_{j_{kl}}^I) = P(N_{j_{kl}}^I \rightarrow \zeta) \]

The sufficient statistics of a PCFG are thus simply counts of how often different local tree configurations occurred (= counts of which grammar rules were applied).

(Probabilistic) CKY algorithm

```java
function CKY(words, grammar) returns most probable parse/probability score = new double[#(words)+1][#(words)+1][#(nonterms)];
back = new Pair[#(words)+1][#(words)+1][#(nonterms)];
for i = 0; i < #(words); i++
for A in nonterms
    if A \rightarrow \text{words}[i] in grammar
        score[i][i+1][A] = P(A \rightarrow \text{words}[i])
// handle unaries
    boolean added = true
    while added
        added = false
        for A, B in nonterms
            if score[i][i+1][B] > 0 && A \rightarrow \text{B} in grammar
                prob = P(A \rightarrow \text{B}) * score[i][i+1][B]
                if (prob > score[i][i+1][A])
                    score[i][i+1][A] = prob
                    back[i][i+1][A] = B
                    added = true
for span = 2 to #(words)
    for begin = 0 to #(words) - span
        end = begin + span
        for split = begin + 1 to end - 1
            for A, B, C in nonterms
                prob = score[begin][split][B] * score[split][end][C] * P(A \rightarrow B C)
                if (prob > score[begin][end][A])
                    score[begin][end][A] = prob
                    back[begin][end][A] = new Triple(split,B,C)
// handle unaries
        boolean added = true
        while added
            added = false
            for A, B in nonterms
                if score[begin][end][B] > 0 && A \rightarrow \text{B} in grammar
                    prob = P(A \rightarrow \text{B}) * score[begin][end][B]
                    if (prob > score[begin][end][A])
                        score[begin][end][A] = prob
                        back[begin][end][A] = B
                        added = true
return buildTree(score, back)
```

Calculation of Viterbi probabilities (CKY algorithm)

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \delta_{NP} = 0.1 )</td>
<td>( \delta_{VP} = 1.0 )</td>
<td>( \delta_{NP} = 0.18 )</td>
<td>( \delta_{VP} = 1.0 )</td>
<td>( \delta_{NP} = 0.18 )</td>
</tr>
<tr>
<td>2</td>
<td>( \delta_{NP} = 0.04 )</td>
<td>( \delta_{VP} = 0.126 )</td>
<td>( \delta_{NP} = 0.0009072 )</td>
<td>( \delta_{VP} = 0.0009072 )</td>
<td>( \delta_{NP} = 0.01296 )</td>
</tr>
<tr>
<td>3</td>
<td>( \delta_{NP} = 0.18 )</td>
<td>( \delta_{VP} = 1.0 )</td>
<td>( \delta_{NP} = 0.18 )</td>
<td>( \delta_{VP} = 1.0 )</td>
<td>( \delta_{NP} = 0.18 )</td>
</tr>
<tr>
<td>4</td>
<td>( \delta_{NP} = 0.18 )</td>
<td>( \delta_{VP} = 1.0 )</td>
<td>( \delta_{NP} = 0.18 )</td>
<td>( \delta_{VP} = 1.0 )</td>
<td>( \delta_{NP} = 0.18 )</td>
</tr>
<tr>
<td>5</td>
<td>( \text{astronomers} )</td>
<td>( \text{saw} )</td>
<td>( \text{stars} )</td>
<td>( \text{with} )</td>
<td>( \text{ears} )</td>
</tr>
</tbody>
</table>
```

The two parse trees' probabilities and the sentence probability

\[
P(t_1) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0009072
\]
\[
P(t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0006804
\]
\[
P(w_{15}) = P(t_1) + P(t_2) = 0.0015876
\]
Modern Statistical Parsers

- A greatly increased ability to do accurate, robust, broad coverage parsing (Charniak 1997; Collins 1997; Ratnaparkhi 1997b; Charniak 2000)
- Achieved by converting parsing into a classification task and using statistical/machine learning methods
- Statistical methods (fairly) accurately resolve structural and real-world ambiguities
- Much faster: rather than being cubic in the sentence length or worse, for modern statistical parsers parsing time is made linear (by using beam search)
- Provide probabilistic language models that can be integrated with speech recognition systems.

Supervised ML parsing

- Crucial resource has been treebanks of parses, especially the Penn Treebank (Marcus et al. 1993)
- From these train classifiers:
  - Mainly probabilistic models, but also:
  - Conventional decision trees
  - Decision lists/ transformation-based learning
- Possible only when extensive resources exist
- Somewhat uninteresting from Cog. Sci. viewpoint – which would prefer bootstrapping from minimal supervision

A Penn Treebank tree (POS tags not shown)

```
( (S (NP-SBJ 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-SBJ *)) (VP reflecting (NP (NP a continuing decline) (PP-LOC in (NP that market)))))
```

Probabilistic models for parsing

- Conditional/Parsing model: We estimate directly the probability of parses of a sentence
  \[
  \hat{t} = \arg \max_t P(t|s, G) \quad \text{where} \quad \sum_t P(t|s, G) = 1
  \]
- We don’t learn from the distribution of sentences we see (but nor do we assume some distribution for them)
- Generative/Joint/Language model: \[
  \sum_t \{t: \text{yield}(t) \in L\} P(t) = 1
  \]
  - Most likely tree
    \[
    \hat{t} = \arg \max_s P(t|s) = \arg \max_s \frac{P(t|s)}{P(s)} = \arg \max_s P(t, s)
    \]
- (Collins 1997; Charniak 1997, 2000)

Generative/Derivational model = Chain rule

\[
P(t) = \sum_{d: d \text{ is a derivation of } t} P(d)
\]
Or: \[
P(t) = P(d) \quad \text{where } d \text{ is the canonical derivation of } t
\]
\[
d = P(S \sqcup \alpha_1 \oplus \ldots \oplus \alpha_m = S) = \prod_{i=1}^{m} P(r_i|r_1, \ldots r_{i-1})
\]
- History-based grammars
  \[
P(d) = \prod_{i=1}^{m} P(r_i|\pi(h_i))
  \]

Enriching a PCFG

- A naive PCFG with traditional nonterminals (NP, PP, etc.) works quite poorly due to the independence assumptions it embodies (Charniak 1996)
- Fix: encode more information into the nonterminal space
  - Structure sensitivity (Manning and Carpenter 1997; Johnson 1998b)
    - Expansion of nodes depends a lot on their position in the tree (independent of lexical content)
    - E.g., enrich nodes by also recording their parents:
      - NP is different to VP
      - S NP is different to VP NP
Enriching a PCFG (2)

- (Head) Lexicalization (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

Parsing via classification decisions:

Charniak (1997)

- A very simple, conservative model of lexicalized PCFG
- Probabilistic conditioning is “top-down” (but actual computation is bottom-up)

Charniak (1997) example

\[
\begin{align*}
S_{\text{rose}} &\quad \text{a. } h = \text{profits}; c = \text{NP} \\
\text{NP} &\quad \text{b. } ph = \text{rose}; pc = S \\
\text{VP} &\quad \text{c. } P(h|ph,c,pc) \\
\text{S} &\quad \text{d. } P(r|h,c,pc)
\end{align*}
\]

Charniak (1997) linear interpolation/shrinkage

\[
\hat{P}(h|ph,c,pc) = \lambda_1(e)P_{\text{MLE}}(h|ph,c,pc) + \lambda_2(e)P_{\text{MLE}}(h|C(ph),c,pc) + \lambda_3(e)P_{\text{MLE}}(h|c,pc) + \lambda_4(e)P_{\text{MLE}}(h|c)
\]

- \( \lambda_i(e) \) is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- \( C(ph) \) is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction

Charniak (1997) shrinkage example

\[
\begin{align*}
P(h|ph,c,pc) &\quad 0.00352 &\quad 0.000557 \\
P(h|C(ph),c,pc) &\quad 0.00627 &\quad 0.00352 \\
P(h|c,pc) &\quad 0.000557 &\quad 0.00352
\end{align*}
\]

- Allows utilization of rich highly 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.

Sparseness & the Penn Treebank

- The Penn Treebank - 1 million words of parsed English
- WSJ - has been a key resource (because of the widespread reliance on supervised learning)
- But 1 million words is like nothing:
  - 965,000 constituents, but only 66 WHADJP, of which only 6 aren’t how much or how many, but there is an infinite space of these (how clever/original/incompetent (at risk assessment and evaluation))
- Most of the probabilities that you would like to compute, you can’t compute
Sparseness & the Penn Treebank (2)

- Most intelligent processing depends on bilexical statistics: likelihoods of relationships between pairs of words.
- Extremely sparse, even on topics central to the WSJ:
  - stocks plummeted 2 occurrences
  - stocks stabilized 1 occurrence
  - stocks skyrocketed 0 occurrences
  - #stocks discussed 0 occurrences
- So far there has been very modest success augmenting the Penn Treebank with extra unannotated materials or using semantic classes or clusters (cf. Charniak 1997, Charniak 2000) – as soon as there are more than tiny amounts of annotated training data.

Probabilistic parsing

- Charniak (1997) expands each phrase structure tree in a single step.
- This is good for capturing dependencies between child nodes
- But it is bad because of data sparseness
- A pure dependency, one child at a time, model is worse
- But one can do better by in between models, such as generating the children as a Markov process on both sides of the head (Collins 1997; Charniak 2000)

Parser results

- Parsers are normally evaluated on the relation between individual postulated nodes and ones in the gold standard tree (Penn Treebank, section 23)
- Normally people make systems balanced for precision/recall
- Normally evaluate on sentences of 40 words or less
- Magerman (1995): about 85% labeled precision and recall
- Charniak (2000) gets 90.1% labeled precision and recall
- Good performance. Steady progress in error reduction
- At some point size of and errors in treebank must become the limiting factor
  - (Some thought that was in 1997, when several systems were getting 87.x%, but apparently not.)