CMPT-413
Computational Linguistics

Anoop Sarkar
http://www.cs.sfu.ca/~anoop

March 17, 2011
Probabilistic CFG (PCFG)

\[
\begin{align*}
S & \rightarrow NP \ VP & 1 \\
VP & \rightarrow V \ NP & 0.9 \\
VP & \rightarrow VP \ PP & 0.1 \\
PP & \rightarrow P \ NP & 1 \\
NP & \rightarrow NP \ PP & 0.25 \\
NP & \rightarrow Calvin & 0.25 \\
NP & \rightarrow monsters & 0.25 \\
NP & \rightarrow school & 0.25 \\
V & \rightarrow imagined & 1 \\
P & \rightarrow in & 1
\end{align*}
\]

\[
P(input) = \sum_{tree} P(tree | input)
\]

\[
P(Calvin \ imagined \ monsters \ in \ school) = ?
\]

Notice that \( P(VP \rightarrow V \ NP) + P(VP \rightarrow VP \ PP) = 1.0 \)
Probabilistic CFG (PCFG)

\[ P(Calvin \text{ imagined monsters in school}) =? \]

\[
(S \ (NP \ Calvin) \\
(VP \ (V \ imagined) \\
  (NP \ (NP \ monsters) \\
   (PP \ (P \ in) \\
    (NP \ school)))))
\]

\[
(S \ (NP \ Calvin) \\
(VP \ (VP \ (V \ imagined) \\
  (NP \ monsters)) \\
  (PP \ (P \ in) \\
   (NP \ school)))))
\]
Probabilistic CFG (PCFG)

(S (NP Calvin)
 (VP (V imagined)
  (NP (NP monsters)
   (PP (P in)
    (NP school)))))

\[
P(tree_1) = P(S \rightarrow NP VP) \times P(NP \rightarrow Calvin) \times P(VP \rightarrow V NP) \times P(V \rightarrow imagined) \times P(NP \rightarrow NP PP) \times P(NP \rightarrow monsters) \times P(PP \rightarrow P NP) \times P(P \rightarrow in) \times P(NP \rightarrow school)
\]
\[
= 1 \times 0.25 \times 0.9 \times 1 \times 0.25 \times 0.25 \times 1 \times 1 \times 0.25 = .003515625
\]
Probabilistic CFG (PCFG)

\[
(S \ (NP \ Calvin) \\
\quad (VP \ (VP \ (V \ imagined) \\
\quad \quad (NP \ monsters))) \\
\quad (PP \ (P \ in) \\
\quad \quad (NP \ school))))
\]

\[
P(tree_2) = P(S \rightarrow NP \ VP) \times P(NP \rightarrow Calvin) \times P(VP \rightarrow VP \ PP) \times \ P(VP \rightarrow V \ NP) \times P(V \rightarrow imagined) \times P(NP \rightarrow monsters) \times \ P(PP \rightarrow P \ NP) \times P(P \rightarrow in) \times P(NP \rightarrow school)
\]
\[
= 1 \times 0.25 \times 0.1 \times 0.9 \times 1 \times 0.25 \times 1 \times 1 \times 0.25 = .00140625
\]
Probabilistic CFG (PCFG)

\[ P(\text{Calvin imagined monsters in school}) = P(\text{tree}_1) + P(\text{tree}_2) \]
\[ = 0.003515625 + 0.00140625 \]
\[ = 0.004921875 \]

Most likely tree is \( \text{tree}_1 \) = \( \arg \max_{\text{tree}} P(\text{tree} | \text{input}) \)

(S (NP Calvin)
  (VP (V imagined)
    (NP (NP monsters)
      (PP (P in)
        (NP school)))))

(S (NP Calvin)
  (VP (VP (V imagined)
    (NP monsters))
  (PP (P in)
    (NP school)))))
Central condition: $\sum_{\alpha} P(A \rightarrow \alpha) = 1$

Called a proper PCFG if this condition holds

Note that this means $P(A \rightarrow \alpha) = P(\alpha \mid A) = \frac{f(A,\alpha)}{f(A)}$

$P(T \mid S) = \frac{P(T,S)}{P(S)} = P(T,S) = \prod_i P(RHS_i \mid LHS_i)$
What is the PCFG that can be extracted from this single tree:

(S (NP (Det the) (NP man))
   (VP (VP (V played)
       (NP (Det a) (NP game)))
   (PP (P with)
       (NP (Det the) (NP dog)))))

How many different rhs $\alpha$ exist for $A \rightarrow \alpha$ where $A$ can be $S$, $NP$, $VP$, $PP$, $Det$, $N$, $V$, $P$
We can do this with multiple trees. Simply count occurrences of CFG rules over all the trees.

A repository of such trees labelled by a human is called a TreeBank.
Ambiguity

- Part of Speech ambiguity
  - saw → noun
  - saw → verb

- Structural ambiguity: Prepositional Phrases
  - I saw (the man) with the telescope
  - I saw (the man with the telescope)

- Structural ambiguity: Coordination
  - a program to promote safety in ((trucks) and (minivans))
  - a program to promote ((safety in trucks) and (minivans))
  - ((a program to promote safety in trucks) and (minivans))
Ambiguity ← attachment choice in alternative parses

(a program) to promote (safety in trucks and minivans)

(a program) to promote (safety in trucks)

NP

NP

a program

NP

to

VP

VP

promote

NP

NP

safety

in

PP

trucks and minivans

NP

NP

and

NP

safety

PP

minivans

in

trucks
Parsing as a machine learning problem

- $S =$ a sentence
  $T =$ a parse tree
  A statistical parsing model defines $P(T \mid S)$
- Find best parse: $\arg \max_T P(T \mid S)$
- $P(T \mid S) = \frac{P(T,S)}{P(S)} = P(T, S)$
- Best parse: $\arg \max_T P(T, S)$
- e.g. for PCFGs: $P(T, S) = \prod_{i=1 \ldots n} P(\text{RHS}_i \mid \text{LHS}_i)$
Adding Lexical Information to PCFG
Adding Lexical Information to PCFG (Collins 99, Charniak 00)

\[ P_h(\text{VB} \mid \text{VP, indicated}) \times P_l(\text{STOP} \mid \text{VP, VB, indicated}) \times P_r(\text{NP(difference)} \mid \text{VP, VB, indicated}) \times P_r(\text{PP(in)} \mid \text{VP, VB, indicated}) \times P_r(\text{STOP} \mid \text{VP, VB, indicated}) \]
Evaluation of Parsing

▶ Consider a candidate parse to be evaluated against the truth (or gold-standard parse):

candidate: (S (A (P this) (Q is)) (A (R a) (T test)))
gold: (S (A (P this)) (B (Q is) (A (R a) (T test))))

▶ In order to evaluate this, we list all the constituents

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,4,S)</td>
<td>(0,4,S)</td>
</tr>
<tr>
<td>(0,2,A)</td>
<td>(0,1,A)</td>
</tr>
<tr>
<td>(2,4,A)</td>
<td>(1,4,B)</td>
</tr>
<tr>
<td></td>
<td>(2,4,A)</td>
</tr>
</tbody>
</table>

▶ Skip spans of length 1 which would be equivalent to part of speech tagging accuracy.

▶ Precision is defined as $\frac{\#\text{correct}}{\#\text{proposed}} = \frac{2}{3}$ and recall as $\frac{\#\text{correct}}{\#\text{in gold}} = \frac{2}{4}$.

▶ Another measure: crossing brackets,

candidate: [ an [incredibly expensive] coat ] (1 CB)
gold: [ an [incredibly [expensive coat]]]
Evaluation of Parsing

Bracketing recall $R = \frac{\text{num of correct constituents}}{\text{num of constituents in the goldfile}}$

Bracketing precision $P = \frac{\text{num of correct constituents}}{\text{num of constituents in the parsed file}}$

Complete match = % of sents where recall & precision are both 100%

Average crossing = $\frac{\text{num of constituents crossing a goldfile constituent}}{\text{num of sents}}$

No crossing = % of sents which have 0 crossing brackets

2 or less crossing = % of sents which have $\leq 2$ crossing brackets
### Statistical Parsing Results

<table>
<thead>
<tr>
<th>System</th>
<th>( \leq 40\text{wds} )</th>
<th>( \leq 100\text{wds} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>(Magerman 95)</td>
<td>84.9</td>
<td>84.6</td>
</tr>
<tr>
<td>(Collins 99)</td>
<td>88.5</td>
<td>88.7</td>
</tr>
<tr>
<td>(Charniak 97)</td>
<td>87.5</td>
<td>87.4</td>
</tr>
<tr>
<td>(Ratnaparkhi 97)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Charniak 99)</td>
<td>90.1</td>
<td>90.1</td>
</tr>
<tr>
<td>(Collins 00)</td>
<td>90.1</td>
<td>90.4</td>
</tr>
<tr>
<td>Voting (HB99)</td>
<td>92.09</td>
<td>89.18</td>
</tr>
</tbody>
</table>
Practical Issues: Beam Thresholding and Priors

- Probability of nonterminal $X$ spanning $j \ldots k$: $N[X, j, k]$
- Beam Thresholding compares $N[X, j, k]$ with every other $Y$ where $N[Y, j, k]$
- But what should be compared?
- Just the *inside probability*: $P(X \Rightarrow t_j \ldots t_k)$?
  written as $\beta(X, j, k)$
- Perhaps $\beta(\text{FRAG}, 0, 3) > \beta(\text{NP}, 0, 3)$, but NPs are much more likely than FRAGs in general
The correct estimate is the *outside probability*:

\[ P(S \Rightarrow t_1 \ldots t_{j-1} X t_{k+1} \ldots t_n) \]

written as \( \alpha(X, j, k) \)

- Unfortunately, you can only compute \( \alpha(X, j, k) \) efficiently after you finish parsing and reach \((S, 0, n)\)
Practical Issues: Beam Thresholding and Priors

- To make things easier we multiply the prior probability $P(X)$ with the inside probability $\beta(X, j, k)$.
- In beam Thresholding we compare every new insertion of $X$ for span $j, k$ as follows:
  Compare $P(X) \cdot \beta(X, j, k)$ with the most probable $Y$
  $P(Y) \cdot \beta(Y, j, k)$
- Assume $Y$ is the most probable entry in $j, k$, then we compare
  \[ \text{beam} \cdot P(Y) \cdot \beta(Y, j, k) \quad (1) \]
  \[ P(X) \cdot \beta(X, j, k) \quad (2) \]
- If (2) $< (1)$ then we prune $X$ for this span $j, k$
- beam is set to a small value, say 0.001 or even 0.01.
- As the beam value increases, the parser speed increases (since more entries are pruned).
- A simpler (but not as effective) alternative to using the beam is to keep only the top $K$ entries for each span $j, k$. 
Experiments with Beam Thresholding

![Graph showing parsing time vs. sentence length for different beam thresholds.](image-url)