The selective left-corner transform
(based on the Johnson and Roark (2000) Coling paper)

Mark Johnson\textsuperscript{1}

\textsuperscript{1}Brown University
Providence, RI
Mark.Johnson@Brown.edu

November 2009
Left-corner grammar and tree transforms

- Transforms left-recursion into right-recursion
- Top-down parser using left-corner transformed grammar simulates a left-corner parser with original grammar
- Defines an invertable mapping from parse trees of original grammar to parse trees of transformed grammar
- Left-corner *grammar transform*
  - new grammar defines *same distribution* over transformed trees as original grammar
  - reduces memory required (stack size)
- Left-corner *tree transform*
  - learn rule probabilities from *transformed trees*
  - defines *different distribution* from grammar estimated from original trees
  - makes some linguistic dependencies local (Manning and Carpenter 1997)
The transformed grammar is not a PCFG because it isn’t normalized (but it is equivalent to a PCFG)
Epsilon removal $D - D \rightarrow \epsilon$

$D \rightarrow w \ D - w$
$D \rightarrow w \ D$
$D \rightarrow \alpha \ D - A$ \quad \text{where} \ A \rightarrow \alpha \in P - L$
$D \rightarrow \alpha$ \quad \text{where} \ D \Rightarrow_{L}^{*} A, A \rightarrow \alpha \in P - L$
$D - B \rightarrow \beta \ D - C$ \quad \text{where} \ C \rightarrow B \ \beta \in L$
$D - B \rightarrow \beta$ \quad \text{where} \ D \Rightarrow_{L}^{*}, C \rightarrow B \ \beta \in L$
The effect of $\epsilon$-removal on top-down rules

- Top-down rules in left-corner transform

$$ D \to \alpha \; D\overline{A} \quad \text{where} \quad A \to \alpha \in P - L $$
$$ D\overline{D} \to \epsilon $$

- After $\epsilon$-removal

$$ D \to \alpha \; D\overline{A} \quad \text{where} \quad A \to \alpha \in P - L $$
$$ D \to \alpha \quad \text{where} \quad D \Rightarrow_L^* A, \; A \to \alpha \in P - L $$
Pruning useless rules — link constraints

- A rule is *useless* if it is never used in a complete derivation
- *Link constraints* filter useless left-corner categories

\[
D-X \text{ is useful } \iff D \Rightarrow_L^{*} X\gamma \text{ for some } \gamma \in \{V \cup T\}^*
\]

(If we’ve applied $\epsilon$-removal, then $\gamma \in \{V \cup T\}^+$)
Pruning useless rules — accessibility constraints

- **Accessibility constraints** restrict left-corner categories to those below a non-left child.
- $D^X$ is useful iff $D = S$ or the original grammar contains a rule $A \rightarrow \alpha D \beta$, $\alpha \in \{V \cup T\}^+$.
Choosing the set of left-corner rules

- The implementor chooses which rules are recognized top-down and which are recognized left-corner.
- The smallest set of rules that results in a non-left-recursive grammar is:
  \[ \{ A \rightarrow B\beta \in P : B \Rightarrow^*_P A \ldots \} \]
- If the preterminals are distinct from the non-terminals, then every terminal is recognized top-down.
Explosion in number of rules

\[ D \rightarrow w \ D^w \]
\[ D \rightarrow \alpha \ D^A \quad \text{where} \quad A \rightarrow \alpha \in P - L \]
\[ D-B \rightarrow \beta \ D-C \quad \text{where} \quad C \rightarrow B \beta \in L \]
\[ D-D \rightarrow \epsilon \]

- Even after pruning, the transformed grammar can be \emph{quadratically larger} than the original grammar
  - the transformed grammar can be huge
    - sparse data problems with tree transforms
- The transformed grammar contains a rule for each top-down rule \( A \rightarrow \alpha \) and each ancestor \( D \) in original grammar
- The transformed grammar contains a rule for each left-corner rule \( C \rightarrow B \beta \) and each ancestor \( D \) in original grammar
Top-down factorization

- Problematic rule schema:

\[ D \rightarrow \alpha D\overline{A} \text{ where } A \rightarrow \alpha \in P - L \]

⇒ Introduce new nonterminal intervening between \( D \) and \( A \)

- Resulting rule schemata:

\[ D \rightarrow A' D\overline{A} \text{ where } A' \text{ is a “new” nonterminal} \]

\[ A' \rightarrow \alpha \text{ where } A \rightarrow \alpha \in P - L \]
Left-corner factorization

- Problematic rule schema:

\[ D - B \rightarrow \beta \ D - C \] where \( C \rightarrow B \beta \in L \)

\[ \Rightarrow \text{Introduce a new nonterminal intervening between } D \text{ and } B \]

- Resulting rule schemata:

\[ D - B \rightarrow C \setminus B \ D - C \] where \( C \setminus B \) is a “new” nonterminal

\[ C - B \rightarrow \beta \] where \( C \rightarrow B \beta \in L \)

- These transformations can also be used in tree-transformations
Sizes of PCFGs without epsilon removal

<table>
<thead>
<tr>
<th></th>
<th>none</th>
<th>(td)</th>
<th>(lc)</th>
<th>(td, lc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G)</td>
<td>15,040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\mathcal{LC}_P)</td>
<td>346,344</td>
<td>30,716</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\mathcal{LC}_N)</td>
<td>345,272</td>
<td>113,616</td>
<td>254,067</td>
<td>22,411</td>
</tr>
<tr>
<td>(\mathcal{LC}_{L_0})</td>
<td>314,555</td>
<td>103,504</td>
<td>232,415</td>
<td>21,364</td>
</tr>
<tr>
<td>(T_P)</td>
<td>20,087</td>
<td></td>
<td>17,146</td>
<td></td>
</tr>
<tr>
<td>(T_N)</td>
<td>19,619</td>
<td>16,349</td>
<td>19,002</td>
<td>15,732</td>
</tr>
<tr>
<td>(T_{L_0})</td>
<td>18,945</td>
<td>16,126</td>
<td>18,437</td>
<td>15,618</td>
</tr>
</tbody>
</table>

- \(P\) is the set of all productions in \(G\) (i.e., the standard left-corner transform),
- \(N\) is the set of all productions in \(P\) which do not begin with a POS tag, and
- \(L_0\) is the set of left-recursive productions.
Sizes of PCFGs with epsilon removal

<table>
<thead>
<tr>
<th></th>
<th>none</th>
<th>(td)</th>
<th>(lc)</th>
<th>(td, lc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>15,040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{LC}_P$</td>
<td>564,430</td>
<td>38,489</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{LC}_N$</td>
<td>563,295</td>
<td>176,644</td>
<td>411,986</td>
<td>25,335</td>
</tr>
<tr>
<td>$\mathcal{LC}_{L_0}$</td>
<td>505,435</td>
<td>157,899</td>
<td>371,102</td>
<td>23,566</td>
</tr>
<tr>
<td>$T_P$</td>
<td>22,035</td>
<td>17,398</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_N$</td>
<td>21,589</td>
<td>16,688</td>
<td>20,696</td>
<td>15,795</td>
</tr>
<tr>
<td>$T_{L_0}$</td>
<td>21,061</td>
<td>16,566</td>
<td>20,168</td>
<td>15,673</td>
</tr>
</tbody>
</table>

- $P$ is the set of all productions in $G$ (i.e., the standard left-corner transform),
- $N$ is the set of all productions in $P$ which do not begin with a POS tag, and
- $L_0$ is the set of left-recursive productions.
Rules in section 23 not seen in 2–21

<table>
<thead>
<tr>
<th>Transform</th>
<th>none</th>
<th>(td)</th>
<th>(lc)</th>
<th>(td,ld)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>514</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{T}_P$</td>
<td>665</td>
<td>535</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{T}_N$</td>
<td>664</td>
<td>543</td>
<td>639</td>
<td>518</td>
</tr>
<tr>
<td>$\mathcal{T}_{L_0}$</td>
<td>640</td>
<td>547</td>
<td>615</td>
<td>522</td>
</tr>
<tr>
<td>$\mathcal{T}_{P,\epsilon}$</td>
<td>719</td>
<td>539</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{T}_{N,\epsilon}$</td>
<td>718</td>
<td>554</td>
<td>685</td>
<td>521</td>
</tr>
<tr>
<td>$\mathcal{T}_{L_0,\epsilon}$</td>
<td>706</td>
<td>561</td>
<td>666</td>
<td>521</td>
</tr>
</tbody>
</table>
Labelled precision and recall on section 23

<table>
<thead>
<tr>
<th>Transform</th>
<th>none</th>
<th>(td)</th>
<th>(lc)</th>
<th>(td,ld)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>70.8, 75.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{P,\epsilon}$</td>
<td>75.8, 77.7</td>
<td>74.8, 76.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{N,\epsilon}$</td>
<td>75.8, 77.6</td>
<td>73.8, 75.8</td>
<td>75.5, 77.8</td>
<td>72.8, 75.4</td>
</tr>
<tr>
<td>$T_{L_0,\epsilon}$</td>
<td>75.8, 77.4</td>
<td>73.0, 74.7</td>
<td>75.6, 77.8</td>
<td>72.9, 75.4</td>
</tr>
</tbody>
</table>
Binarization and left-corner parsing

- Basic idea: *delay decisions as long as possible*
- In standard left-corner parsing ⇒ *left binarization*
- Standard left-corner grammar transform:

  \[X \rightarrow w \, Xw\]
  \[X-X \rightarrow \epsilon\]
  \[X-B_1 \rightarrow X-A \, B_2 \, \ldots \, B_n\] where \(A \rightarrow B_1 \, \ldots \, B_n \in P\)

- Left binarization and left-corner transform:

  \[X \rightarrow wXw\]
  \[X-X \rightarrow \epsilon\]
  \[X-\beta \rightarrow X-A\] where \(A \rightarrow \beta \in P\)
  \[X-\beta \rightarrow B \, X-\beta B\]

- But this explodes the number of rules, and left-corner factorization does not help!
Binarization with left-corner factoring

- Left-corner factoring grammar

\[
\begin{align*}
X & \rightarrow w \ X\bar{w} \\
X\bar{X} & \rightarrow \epsilon \\
X\bar{B} & \rightarrow A\backslash B \ X\bar{A} \\
A\backslash B & \rightarrow \beta \\
\text{where } A & \rightarrow B \ \beta \in P
\end{align*}
\]

▶ predicts entire RHS after 1st child

- Binarized left-corner factoring grammar

\[
\begin{align*}
X & \rightarrow w \ X\bar{w} \\
X\bar{X} & \rightarrow \epsilon \\
X\bar{B} & \rightarrow A\backslash B \ X\bar{A} \\
A\backslash \beta & \rightarrow \epsilon \\
A\backslash \beta & \rightarrow B \ A\backslash \beta B \\
\text{where } A & \rightarrow \beta \in P \\
\text{filter: } A & \rightarrow \beta B \ \gamma \in P
\end{align*}
\]

▶ incrementally enumerates children on RHS
Binarization with left-corner factoring

\[
\begin{array}{c}
\ldots X \ldots \\
A \\
B_1 \quad B_2 \quad \ldots \quad B_n \\
\gamma_1 \quad \gamma_2 \quad \ldots \quad \gamma_n \\
w_1
\end{array}
\Rightarrow
\begin{array}{c}
\ldots X \ldots \\
w_1 \quad X-w_1 \\
B_1 \quad X-B_1 \\
\gamma_1 \quad A \backslash B_1 \\
B_2 \quad X-A \\
\gamma_2 \\
A \backslash B_1 B_2 \\
B_n \quad X-X \\
\gamma_n \\
A \backslash B_1 \ldots B_{n-1} \\
B_n \quad A \backslash B_1 \ldots B_n
\end{array}
\]