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

Mark Johnson[1]

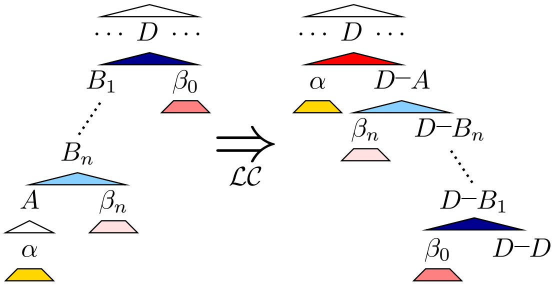
[1]Brown University
Providence, RI
Mark_Johnson@Brown.edu

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)

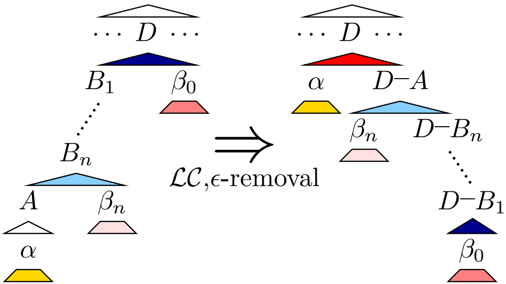
The selective left-corner transform



$$
\begin{array}{rcll}
D & \rightarrow & w\ D\!-\!w & \\
D & \rightarrow & \alpha\ D\!-\!A & \text{where } A \rightarrow \alpha \in P - L \\
D\!-\!B & \rightarrow & \beta\ D\!-\!C & \text{where } C \rightarrow B\,\beta \in L \\
D\!-\!D & \rightarrow & \epsilon &
\end{array}
$$

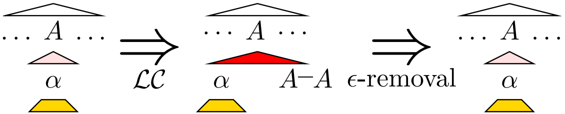
- The transformed grammar *is not a PCFG* because it isn't normalized (but it is equivalent to a PCFG)

Epsilon removal D–$D \to \epsilon$



$$
\begin{array}{lcll}
D & \to & w\ D\text{–}w & \\
D & \to & w\ D & \\
D & \to & \alpha\ D\text{–}A & \text{where } A \to \alpha \in P - L \\
D & \to & \alpha & \text{where } D \Rightarrow^{\star}_{L} A, A \to \alpha \in P - L \\
D\text{–}B & \to & \beta\ D\text{–}C & \text{where } C \to B\,\beta \in L \\
D\text{–}B & \to & \beta & \text{where } D \Rightarrow^{\star}_{L}, C \to B\,\beta \in L
\end{array}
$$

The effect of ϵ-removal on top-down rules



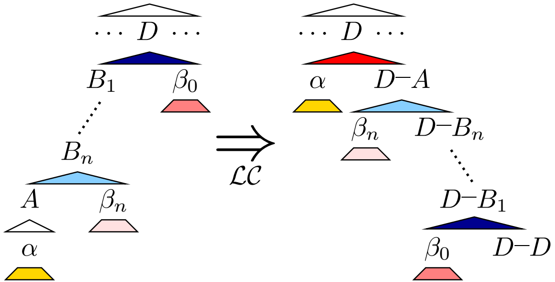
- Top-down rules in left-corner transform

$$\begin{aligned} D &\rightarrow \alpha\ D\text{–}A \quad \text{where } A \rightarrow \alpha \in P - L \\ D\text{–}D &\rightarrow \epsilon \end{aligned}$$

- After ϵ-removal

$$\begin{aligned} D &\rightarrow \alpha\ D\text{–}A \quad \text{where } A \rightarrow \alpha \in P - L \\ D &\rightarrow \alpha \quad\quad\quad \text{where } D \Rightarrow_L^\star A, A \rightarrow \alpha \in P - L \end{aligned}$$

Pruning useless rules — link constraints

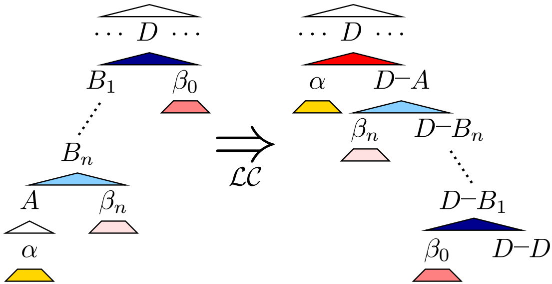


- A rule is *useless* if it is never used in a complete derivation
- *Link constraints* filter useless left-corner categories

$$D{-}X \text{ is useful} \quad \Leftrightarrow \quad D \Rightarrow_L^\star X\gamma \text{ for some } \gamma \in \{V \cup T\}^\star$$

(If we've applied ϵ-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 $A \to \alpha D \beta$, $\alpha \in \{V \cup T\}^+$

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:

$$\{A \rightarrow B\beta \in P \ : \ B \Rightarrow_P^\star A \ldots\}$$

- If the preterminals are distinct from the non-terminals, then every terminal is recognized top-down

Explosion in number of rules

$$\begin{array}{rcll} D & \rightarrow & w\ D\text{–}w & \\ D & \rightarrow & \alpha\ D\text{–}A & \text{where } A \rightarrow \alpha \in P - L \\ D\text{–}B & \rightarrow & \beta\ D\text{–}C & \text{where } C \rightarrow B\,\beta \in L \\ D\text{–}D & \rightarrow & \epsilon & \end{array}$$

- Even after pruning, the transformed grammar can be *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:

$$D \;\rightarrow\; \alpha \; D\text{–}A \quad \text{where } A \rightarrow \alpha \in P - L$$

⇒ Introduce new nonterminal intervening between D and A

- Resulting rule schemata:

$$\begin{array}{lcll} D & \rightarrow & A' \; D\text{–}A & \text{where } A' \text{ is a "new" nonterminal} \\ A' & \rightarrow & \alpha & \text{where } A \rightarrow \alpha \in P - L \end{array}$$

Left-corner factorization

- Problematic rule schema:

$$D\text{–}B \;\rightarrow\; \beta\ D\text{–}C \qquad \text{where } C \rightarrow B\,\beta \in L$$

⇒ Introduce a new nonterminal intervening between D and B

- Resulting rule schemata:

$$\begin{array}{lll} D\text{–}B & \rightarrow & C\backslash B\ D\text{–}C \qquad \text{where } C\backslash B \text{ is a “new” nonterminal} \\ C\text{–}B & \rightarrow & \beta \qquad\qquad\qquad \text{where } C \rightarrow B\,\beta \in L \end{array}$$

- These transformations can also be used in tree-transformations

Sizes of PCFGs without epsilon removal

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

- 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

	rule factoring			
	none	(td)	(lc)	(td, lc)
G	15,040			
$\mathcal{LC}_P$	564,430		38,489	
$\mathcal{LC}_N$	563,295	176,644	411,986	25,335
$\mathcal{LC}_{L_0}$	505,435	157,899	371,102	23,566
$\mathcal{T}_P$	22,035		17,398	
$\mathcal{T}_N$	21,589	16,688	20,696	15,795
$\mathcal{T}_{L_0}$	21,061	16,566	20,168	15,673

- 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

Transform	none	(td)	(lc)	(td,ld)
none	514			
$\mathcal{T}_P$	665		535	
$\mathcal{T}_N$	664	543	639	518
T_{L_0}	640	547	615	522
$\mathcal{T}_{P,\epsilon}$	719		539	
$\mathcal{T}_{N,\epsilon}$	718	554	685	521
$\mathcal{T}_{L_0,\epsilon}$	706	561	666	521

Labelled precision and recall on section 23

Transform	none	(td)	(lc)	(td,ld)
none	70.8,75.3			
$\mathcal{T}_{P,\epsilon}$	75.8,77.7		74.8,76.9	
$\mathcal{T}_{N,\epsilon}$	75.8,77.6	73.8,75.8	75.5,77.8	72.8,75.4
$\mathcal{T}_{L_0,\epsilon}$	75.8,77.4	73.0,74.7	75.6,77.8	72.9,75.4

Binarization and left-corner parsing

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

$$
\begin{aligned}
&X \to w\ X\text{–}w \\
&X\text{–}X \to \epsilon \\
&X\text{–}B_1 \to X\text{–}A\ B_2\ \ldots\ B_n \quad \text{where } A \to B_1\ \ldots\ B_n \in P
\end{aligned}
$$

- Left binarization and left-corner transform:

$$
\begin{aligned}
&X \to wX\text{–}w \\
&X\text{–}X \to \epsilon \\
&X\text{–}\beta \to X\text{–}A \qquad \text{where } A \to \beta \in P \\
&X\text{–}\beta \to B\ X\text{–}\beta B
\end{aligned}
$$

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

Binarization with left-corner factoring

- Left-corner factoring grammar

$$
\begin{array}{ll}
X \to w\ X{-}w & \\
X{-}X \to \epsilon & \\
X{-}B \to A\backslash B\ X{-}A & \\
A\backslash B \to \beta & \text{where } A \to B\ \beta \in P
\end{array}
$$

  - predicts entire RHS after 1st child
- Binarized left-corner factoring grammar

$$
\begin{array}{ll}
X \to w\ X{-}w & \\
X{-}X \to \epsilon & \\
X{-}B \to A\backslash B\ X{-}A & \\
A\backslash\beta \to \epsilon & \text{where } A \to \beta \in P \\
A\backslash\beta \to B\ A\backslash\beta B & \text{filter: } A \to \beta\, B\, \gamma \in P
\end{array}
$$

  - incrementally enumerates children on RHS

Binarization with left-corner factoring