Prefix Probabilities for Linear Indexed Grammars

TAG+ 4, IRCS/University of Pennsylvania

Mark-Jan Nederhof  
(DFKI)  

Anoop Sarkar  
(UPenn)  

Giorgio Satta  
(Univ. di Padova)
Prefix Probabilities

• Language model: given a string $a_1, \ldots, a_{i-1}$, $a_i$ can be any word in the vocabulary $\Sigma$, what is $P(a_i | a_1, \ldots, a_{i-1})$?

• Standard techniques use trigram models: $P(a_i | a_{i-2}, a_{i-1})$

• A stochastic grammar can be used by computing the prefix probability:

$$
\sum_{w \in \Sigma^*} P(a_1, \ldots, a_n, w)
$$
Stochastic Tree Adjoining Grammars

\[
\begin{align*}
\alpha & \rightarrow a \quad \beta_1 \quad \phi(\beta_1, X) \\
& \quad \rightarrow X \quad c \\
& \quad \rightarrow b \\
\beta_2 & \rightarrow d \quad e \\
& \quad \rightarrow a \quad \rightarrow c \\
& \quad \rightarrow b \quad \rightarrow d \quad \rightarrow e \\
& \quad \rightarrow f
\end{align*}
\]
Let prefix = $abd$
Problem

- Derivations are a combination of two kinds of subderivations:
  
  1. potentially unbounded subderivations, independent of input
  
  2. bounded subderivations, depend on input symbols

- Problem: how to partition derivations uniquely into subderivations.

- Without unique partitions, algorithm will return incorrect probabilities.
TAG Derivations and LIGs

- CFGs with stack symbols, Indexed Grammars: $A[..\eta_r] \rightarrow B[..\eta_1]C[..\eta_2]$

- Unique rhs gets stack = Linear Indexed Grammars: $A[..\eta_r] \rightarrow B[..\eta_1]C[\$\eta_2]$

- LIGs and TAGs are weakly equivalent.

- Given a TAG, a strongly equivalent LIG can be built.

- LIGs offer a convenient way to denote TAG derivations.
TAG Derivations and LIGs

\[ b[..root] \rightarrow t[$X]t[..Y] \]

\[ t[..X] \rightarrow b[..X] \]

\[ t[..X] \rightarrow t[..X root] \]

\[ b[..foot] \rightarrow b[..] \]
vertical subderivation
$ver_1$

$ver_2$

$ver_3$

horizontal subderivation
Summary

- LIGs offer a convenient way to denote TAG derivations.

- Potentially unbounded subderivations, independent of input. Probability is computed off-line.

- Bounded subderivations, depends on input symbols. Inside probability computed as prefix is recognized.

- Algorithm partitions derivations uniquely into subderivations.