Adaptor grammars – a framework for NPB models

Word \rightarrow \text{Stem Suffix}

Stem \rightarrow \text{Chars}

Suffix \rightarrow \text{Chars}

Chars \rightarrow \text{Char}

Chars \rightarrow \text{Char Chars}

Char \rightarrow a | b | c | \ldots

• CFG rules define possible trees
• To generate an adapted nonterminal
  ▶ return a previously generated tree \( t \) with prob. \( \propto n(t) \)
  ▶ generate a tree \( t \) using CFG rules with prob. \( \propto \alpha X P'(t) \)

\[
P_{\text{Word}} \sim DP(\alpha_{\text{Word}}, P_{\text{Word}'})
\]

\[
P_{\text{Word}'}(w) = \sum_{s,t:s \cdot t = w} P_{\text{Stem}}(s)P_{\text{Suffix}}(t)
\]

\[
P_{\text{Stem}} \sim DP(\alpha_{\text{Stem}}, P_{\text{Stem}'})
\]

\[
P_{\text{Stem}'}(w) \propto p^{\vert w \vert}
\]
Software for using adaptor grammars

- Intended to ease exploration of Goldwater-style NPB models
  - general enough to express e.g., LDA-ish topic models
- Metropolis-within-Gibbs sampler
- Proposal distribution is *PCFG approximation* based on sample parses of other sentences
  - one production for each cached subtree
  - dynamic programming algorithm for producing proposals
- Software implements Pitman-Yor processes (not just CRPs)
- Sampling PY hyperparameters sometimes improves model accuracy
- *Table label resampling* vastly speeds convergence
- Nonidentifiability of latent states can be a potential problem
  \[\Rightarrow\] evaluate a single sample of parses for all sentences
Adaptor grammar derivation (0)

**Word** restaurant
Word → Stem Suffix

**Stem** restaurant
Stem → Phoneme*

**Suffix** restaurant
Suffix → Phoneme+
Adaptor grammar derivation (1a)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → Phoneme*

Suffix restaurant
Suffix → Phoneme+
Adaptor grammar derivation (1b)

Word restaurant
Word → Stem Suffix

Stem restaurant
Stem → Phoneme

Suffix restaurant
Suffix → Phoneme

...
Adaptor grammar derivation (1c)

**Word** restaurant
Word → Stem Suffix

**Stem** restaurant
Stem → Phoneme*

**Suffix** restaurant
Suffix → Phoneme+
Adaptor grammar derivation (1d)

**Word** restaurant
Word → Stem Suffix

**Stem** restaurant
Stem → Phoneme

**Suffix** restaurant
Suffix → Phoneme
Adaptor grammar derivation (2a)

**Word** restaurant
Word → Stem Suffix

**Stem** restaurant
Stem → Phoneme*

**Suffix** restaurant
Suffix → Phoneme+
Adaptor grammar derivation (2b)

**Word** restaurant  
Word → Stem Suffix

**Stem** restaurant  
Stem → Phoneme*  

**Suffix** restaurant  
Suffix → Phoneme+
Adaptor grammar derivation (2c)

Word \textit{restaurant}

\begin{itemize}
  \item Word → Stem Suffix
  \begin{itemize}
    \item Stem → Phoneme^*
      \begin{itemize}
        \item Suffix → Phoneme^+
      \end{itemize}
  \end{itemize}
\end{itemize}
Adaptor grammar derivation (2d)

**Word restaurant**
Word → Stem Suffix

**Stem restaurant**
Stem → Phoneme*

**Suffix restaurant**
Suffix → Phoneme+
Adaptor grammar derivation (3)

**Word** restaurant
Word → Stem Suffix

**Stem** restaurant
Stem → Phoneme*

**Suffix** restaurant
Suffix → Phoneme+
Adaptor grammar derivation (4a)

**Word** restaurant  
Word $\rightarrow$ Stem Suffix

**Stem** restaurant  
Stem $\rightarrow$ Phoneme$^*$

**Suffix** restaurant  
Suffix $\rightarrow$ Phoneme$^+$
Adaptor grammar derivation (4b)

**Word restaurant**  
Word → Stem Suffix

**Stem restaurant**  
Stem → Phoneme*  

**Suffix restaurant**  
Suffix → Phoneme+
Adaptor grammar derivation (4c)

**Word** restaurant
Word → Stem Suffix

**Stem** restaurant
Stem → Phoneme*

**Suffix** restaurant
Suffix → Phoneme⁺
Adaptor grammar derivation (4d)

**Word** restaurant
Word → Stem Suffix

**Stem** restaurant
Stem → Phoneme*

**Suffix** restaurant
Suffix → Phoneme
Table label resampling improves mobility

- Metropolis-within-Gibbs: resample $T_i$ given $W_i$ and $\vec{T}_{-i}$
- Table label resampling resamples the labels on each table
  - can change parses for many sentences at once

**Word** restaurant

**Stem** restaurant

**Suffix** restaurant

<table>
<thead>
<tr>
<th>Word</th>
<th>Stem</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy</td>
<td>b</td>
<td>y</td>
</tr>
<tr>
<td>run</td>
<td>r</td>
<td>n</td>
</tr>
<tr>
<td>by</td>
<td>b</td>
<td>y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stem</th>
<th>Phonoem*</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy</td>
<td>b</td>
</tr>
<tr>
<td>run</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Phoneme*</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>s</td>
</tr>
</tbody>
</table>

| | | |
| | | |
| | | |
Table label resampling with Colloc grammar

- log posterior probability

Iteration

0 200 400 600 800 1000

-185000 -190000 -195000 -200000 -205000 -210000 -215000 -220000

no resampling
resampling
resampling to iteration 100