Type-based MCMC for Sampling Tree Fragments from Forests
Xiaochang Peng, Daniel Gildea
University of Rochester

Overview

- We apply type-based MCMC to learning SCFG rules.
  - Assume fixed word alignment.
  - Learn SCFG rules consistent with alignment, each SCFG rule is a tree fragment in the phrase decomposition forest.
  - We assume fragment sizes (as in TSG learning) as well as bracketing structures (extending TSG learning)
- We investigate the impact of type-based method on the likelihood of the Markov Chain in this setting.
  - Token-based and block-based MCMC: do not deal with the coupling issue of variables.
  - Type-based MCMC: grouping strongly coupled variables as the same type.
- We present an innovative way of storing the type information.
  - Reduce the amount of bookkeeping by indexing on partial type information.
  - Additional steps to filter nodes with full type information.
- We replace the two-stage sampling schedule of Liang et al. (2010) with a simpler and faster one-stage method.
- Parallel programming with inexact type-based MCMC

Sample Tree Fragments from Forests

- Chung et al. (2014) present a schedule to learn Hiero-style SCFG rules from phrase decomposition forests
  - Build a phrase decomposition forest from bottom up
  - MCMC sampling from top down: sample cut, sample edge.

Type-based MCMC

- Two cut sites are of the same type if the composed rules we get are exactly the same when assigning same cut value to them:
  \[ \text{type}(1, n) \text{ def } (r_1, r_2, \ldots, r_n) \]
- We calculate the joint probability of an assignment having \( m \) cut sites:
  \[ P(z_j|N) \propto \prod_{j=1}^{n-m} P(r_j|N^{-1}) \prod_{j=1}^{m} P(c_j|N) P(z_j|N^{-1}) \text{ def } g(m) \]
- The posterior probability of all assignments having \( m \) cut sites is:
  \[ p(m|N) \propto \sum_{z_j} \sum_{c_j} p(z_j|N) = \frac{n}{m} g(m) \]
- We sample \( m \) according to this equation. Then we choose \( m \) sites among the \( n \) variables to be cut with uniform distribution.

Experimental Results

- Type-based sampling converges to a much better result than non-type-based top-down sampling and escapes some local optima that are hard for token-based methods to escape:
- The better likelihood of our Markov Chain using type-based MCMC also results in better transition:

<table>
<thead>
<tr>
<th>Sampling Schedule</th>
<th>Iteration</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-type-based</td>
<td>averaged (0-90)</td>
<td>25.62</td>
<td>24.98</td>
</tr>
<tr>
<td>Type-based</td>
<td>averaged (0-100)</td>
<td>25.88</td>
<td>25.20</td>
</tr>
<tr>
<td>Parallel Type-based</td>
<td>averaged (0-90)</td>
<td>25.75</td>
<td>25.04</td>
</tr>
</tbody>
</table>

Bookkeeping Strategy

- Unlike PTSG: the internal structure of each rule type is abstracted away.
- Strategy: build a small index at the cost of additional computation:
  - We only key on the rule types turned on in the current chosen derivation
  - We key on a single rule type and index only the root of each rule type

Optimization

- One-stage sampling schedule: build real \( m \) greedily.
  \[ P(z_j|N) = \prod_{j=1}^{n} P(r_j|N^{-1}) \]
- Inexact type-based MCMC with parallel programming:
  - Split the data into subsets, communicate local counts.
  - The local bookkeeping of each subset is not communicated.