Issues Concerning Decoding with SCFG
Tagyoung Chung  Licheng Fang  Daniel Gildea
University of Rochester

Summary
- Although other solutions are available, modifying the SCFG extraction process such that it does not extract any unary rules is a good solution to the problem caused by unary rules.
- Both filtering rules (Hopkins and Langmead, 2010) and binarizing SCFG rules are effective methods for efficient SCFG decoding, but the two methods have different trade-offs.
- A set of glue rules that allows reordering, especially when training data size is small, is an effective way to provide more flexibility to SCFG rules.

Modifying grammar extraction
- Conversion to CNF removes unary rules but bloats grammar.
- Following modification prevents unary rules:
  - Extract a rule only if its nonterminal covers a larger span than in the source language than the nonterminal.

Unary rules
- A rule that have exactly one nonterminal and no terminals on the source side:
  - Example: NP → NP (Chinese side)
  - the NP (English side).
- Common when extracted using the GHKM extraction method (Galley et al., 2004).
- Problematic because they may cause:
  - Endless chains.

Decoding with unary rules
- Unary cycle detection:
  - Backtrack on a unary chain to see if a newly generated item has been generated before.
  - Running time is constrained by the pruning constant.
- Ban negative cost unary rules
  - May happen during parameter tuning.

Glue rules
- An SCFG may fail to parse sentences at test time.
- Monotonic glue rules are used in Hiero (Chiang, 2005):
  - S → C, C
  - S → S, C, S
  - S is the goal state and C is the glue nonterminal.
- ABC glue rules (Wu, 1997) allows constituents to be reordered and produces only left-heavy parses:
  - S → S C, S C
  - S → S A A
  - A A
  - ⟨ B C ⟩
  - ⟨ A C ⟩

Adding phrases
- Valid GHKM rules are constrained by both alignments and target-side parse trees.
- Adding phrases from phrase-based translation (Koehn et al., 2003) with glue nonterminal may provide more flexibility.

Results

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline + monotonic glue rules</td>
<td>20.99</td>
</tr>
<tr>
<td>No-unary + monotonic glue rules</td>
<td>23.83</td>
</tr>
<tr>
<td>No-unary + ABC glue rules</td>
<td>23.94</td>
</tr>
<tr>
<td>No-unary (scope-filtered) + monotonic</td>
<td>23.99</td>
</tr>
<tr>
<td>No-unary (scope-filtered) + ABC glue rules</td>
<td>24.09</td>
</tr>
<tr>
<td>No-unary + ABC glue rules + phrases</td>
<td>23.43</td>
</tr>
</tbody>
</table>

- 250K sentence training data, 392-sentence development set for tuning, and 428-sentence test set (four references).
- No-unary rules produce better results (statistically significant) but ABC glue rules produce only slightly better translation (statistically insignificant).
- Scope-3 pruned grammar produced slightly better translation (statistically insignificant) but overall slightly slower.
- Adding phrases was slightly worse: Once phrases are used, the decoder needs to rely on glue rules to finish parsing.

Handling virtual nonterminals
- The grammar either needs to be binarized (Zhang et al., 2006) or pruned (Hopkins and Langmead, 2010) to ensure $O(n^3)$ time parsing.
- Binarization creates new rules with virtual nonterminals. We assign the remaining cost of the original rule as an outside-estimate to avoid the situation where new rules are unfairly advantaged when competing with regular rules.

\[
\text{cost heuristic} = \begin{cases} 
0 & \text{if } \text{virtual nonterminal} \\
-\log(\rho) & \text{if } \text{regular nonterminal}
\end{cases}
\]
- Example:
  - $A \rightarrow BCD: -\log(\rho) \Rightarrow V \rightarrow BC: 0 - \log(\rho)$
  - $A \rightarrow VD: -\log(\rho) 0$