Terminal-Aware Synchronous Binarization

Licheng Fang, Tagyoung Chung and Daniel Gildea • University of Rochester

Synchronous binarization
- Factorizes synchronous context-free grammars rules
- Makes machine translation decoding with SCFG faster
- Facilitates language model integration during decoding

\[ X \rightarrow A B C, A C B \]

different source-side binarizations

Room for improvement
- Vanilla synchronous binarization (Zhang et al, 2006) always chooses the rightmost binarizable point when many legal synchronous binarizations exist, which is common

Cost function \( b \) - synchronous binarization
- Enforces synchronous binarization
- Non-synchronously-binarizable rules are discarded

Cost function \( n \) - maximize nonterminal sharing
- Binarization trees that generate more unseen nonterminals are discouraged

Cost function \( c \) - early source-side terminal matching
- A simple probabilistic model based on corpus statistics
- Prefers to match infrequent source-side terminals first

Late target-side terminal attachment
- Conventional wisdom is attaching target-side terminals as low as possible in binarization tree
- But LM discourages long translation hypotheses. Unfairly pruning away translation hypotheses generated by rules with many target-side terminals
- We attach target-side terminals as high as possible in the binarization tree

Binarization as parsing
- Formulated as parsing the source side of a rule
- Chooses a best binarization tree based on cost functions
- Uses CYK-like algorithm
- We define 3 cost functions

The conventional method considers only nonterminals:
- always chooses right-most binarizable point for terminals on the source side
- attaches target-side terminals as low as possible in the binarization tree

Cost function \( \gamma \) - target-side terminal attachment
- Kills duplicates of terminals
- Maximizes source-side terminal attachment

Results
- Chinese-to-English translation task on dataset of 250K training sentence pairs, and a 329-sentence test set
- Early source-side terminal matching is better than rightmost binarization
- Late target-side terminal attachment is better than early target-side terminal attachment

Keys: (b, n)-late means we use the cost function \( b \) to select the best binarization tree, and ties are broken by the cost function \( n \). Late target-side terminal attachment is also applied.