Bottom-Up Transfer in EBMT

Vincent Vandeghinste & Scott Martens
Centre for Computational Linguistics
Katholieke Universiteit Leuven
Project: PaCo-MT

Parse and Corpus-based Machine Translation

• 3 year project (2008-2011), 500K€

• Sponsored by STEVIN program of the Dutch Language Union

• NL ↔ EN  NL ↔ FR

• Consortium partners
  – CCL – KULeuven
  – Alfa-Informatics – RUGroningen
  – OneLiner bvba Translation Services
Advanced Multimedia Alignment and Structured Summarization

• 4 year project (2007-2010),
• Sponsored by IWT: Innovation and Technology Institute of Flanders
• CCL provides translational components
• Consortium partners
  – CCL – KULeuven
  – ESAT-Visics KULeuven
  – LIIR – KULeuven
  – EDM – University of Hasselt
System Description

- **Hybrid MT**: Stochastic Example-based Transfer System
- Automatic *transfer rule induction* based on parallel treebanks
- Automatic *dictionary extraction* (lexical rules) from parallel treebanks
- Reusing existing tools as much as possible
Similar Approaches

- **Data-Oriented Translation** (Poutsma, Hearne)
- Transfer Rules resemble Galley et al. (2004;2006), but no explicit rule extraction: virtual rules
- Synchronous CFG (Ambati et al. 2009)
- Synchronous Tree-Substitution Grammars (Zhang et al. 2007)
PaCo-MT

Diagram:

- SL parse tree
- Syntactic Analysis
- source language sentence
- transfer
- dictionary
- parallel data
- TL data
- Generation
- target language sentence
- TL bags of bags
Syntactic Analysis

- **Dutch**
  - Alpino parser (van Noord 2006)
  - Phrase structure + dependencies
- **English**
  - Stanford parser (Klein & Manning 2003)
  - Phrase structure + dependencies
- **French**
  - Not in this paper
Requires

– A parallel corpus (Europarl, DGT, TMs)
– Alignment at the sentence level
– Alignment at the word level (Giza++)
– Source language parser
– Target language parser
Alignment at the tree level: Sub-sentential alignment (node alignment)

– Lexicalized: Each tree pair, sub-tree pair, word pair
  • an example translation pair
  • dictionary entry

– Not lexicalized: Each tree pair, sub-tree pair
  • an example translation rule
  • a transfer instance

– Tiedemann & Kotzé (2009): A Discriminative Approach to Tree Alignment (RANLP09)
Example Alignment
Bottom-Up Transfer

- Top Down transfer does not work: cf. Vandeghinste & Martens (2009)
- Bottom-up:
  - Starting with translations of words and phrases
  - Structural translations on the basis of translations discovered at the bottom
- Confidently translate words and phrases,
- Use those translations to constrain the choice of structures above.
- Errors propagate upwards, not downwards
Virtual Rules

• Consult the treebank on the fly
• Lexicalized translations: An entire phrase from the source parse tree appears in the treebank:
  – Trees are reordered so that the children of each node in the tree appear in fixed lexicographic order (ignoring original word order)
  – Trees are rewritten as strings (depth first order)
  – If subtrees in SL parse are identical to subtrees in treebank then there is a substring in the converted treebank that is identical
Bottom-up Subtree

The diagram illustrates a bottom-up subtree in a syntactic analysis. The subtree is represented with a NP (noun phrase) at the top, with branches that include NNP (proper noun), IN (preposition or particle), DT (determiner), and NNPS (plural noun). The path from the root NP to the leaf NNP (Approval) is shown with solid lines, and the path to the leaf NNPS (Minutes) is shown with dashed lines.
Bottom-up Subtree Matching

• Similar to subsentential translation memory:
  – Each match is to a linguistically motivated phrase
  – When a match is found, the aligned target language subtree is used in the translation

• Finding string matches: suffix array
  – Identifies matches in indexed string in sublinear time
  – Converting the subtree discovery problem into a string matching problem
Transfer of the upper portions of the tree

- Generalization of the rule construction method from Vandeghinste & Martens (2009)
- Converting trees into strings in a breadth-first method
Breadth-first String Representation

- # indicates exhaustion of the children of some node
- $ indicates exhaustion of the nodes at a particular depth in the tree

- One-to-one correspondence of strings with subtrees
- If any two subtrees are identical from the root down to some depth, these string representations share a common prefix
- By sorting them, we can quickly match any subtree in a new parse tree down to a fixed depth
Matching the source tree with the examples

- Bottom-up matching finds all phrases and words that have matches in the treebank
Matching the source tree with the examples

- Top-down matching looks for structures in the source language treebank matching the remaining part of the translation
Matching the source tree with the examples

- Each top-down match is finally connected to the bottom-up matches
Target Language Generation

- In transfer rules: no ordering of children
- **Optimal surface ordering using a large target language treebank**
  
  Vandeghinste (2009). Tree-Based Target Language Modeling. EAMT

- Additional lexical selection
Target Language Generation

• On a large TL corpus, we extract CFG rules on different abstraction levels
  – Dependency relations (REL)
  – Syntactic Category labels / Parts-of-speech (CAT)
  – CAT + REL
  – CAT + REL + Token

• For every node in the TL tree, we check if we find a rewrite rule at the most concrete level, cascading down to more abstract levels if we don’t find a solution

• Still in the process of determining the optimal ordering from concrete to abstract
Target Language Generation

• We estimate the probability of different orderings, selecting the most probable, by looking at the frequency of occurrence in the training data
• When \( f = 0 \), generate all permutations
• Recursively ordering all the nodes in the tree to generate several surface forms of the unordered target language tree
Experiment

• Test set of 500 Dutch sentences with 2 reference translations
• Independent variables:
  – Dummy transfer (only lexical transfer rules)
  – Small vs. large beam
• Compared with top-down (Vdg & M 2009)
• Compared with Moses without punct.
Results

- 52.7 % relative improvement compared to Top-down
- PER metric marginally worse than Moses: lexical selection is ‘good’
- Dummy gives an indication of the influence of structural transfer
- Difference in beam width in TLG is neglectable

<table>
<thead>
<tr>
<th>Condition</th>
<th>BLEU</th>
<th>NIST</th>
<th>WER</th>
<th>CER</th>
<th>PER</th>
<th>TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-down</td>
<td>13.53</td>
<td>5.70</td>
<td>76.20</td>
<td>61.91</td>
<td>52.39</td>
<td>70.36</td>
</tr>
<tr>
<td>Dummy</td>
<td>12.49</td>
<td>6.01</td>
<td>78.75</td>
<td>63.83</td>
<td>50.05</td>
<td>70.69</td>
</tr>
<tr>
<td>Smallbeam</td>
<td>20.65</td>
<td>6.44</td>
<td>70.34</td>
<td>55.37</td>
<td>48.96</td>
<td>63.72</td>
</tr>
<tr>
<td>Largebeam</td>
<td>20.59</td>
<td>6.43</td>
<td>70.10</td>
<td>55.12</td>
<td>48.98</td>
<td>63.54</td>
</tr>
<tr>
<td>Moses No Punct.</td>
<td>26.72</td>
<td>6.94</td>
<td>60.53</td>
<td>45.65</td>
<td>47.82</td>
<td>58.07</td>
</tr>
</tbody>
</table>
Conclusions & Future

• **Lexical selection is good**
  – We have solutions for some of the problems of our system which are not yet reflected in these results

• **Influence of structural transfer is large and positive**
  – Partial subtree matching
  – Different parameter settings
  – Should improve the coverage of the induced rules

• **Improvements to the virtual transfer rule system**
  – Too slow now
  – Using sampling in many cases
  – Subtree indexing to reduce this time
Conclusions & Future

• Regular Tree Grammar
  – Weakly equivalent in generation capacity to a CFG
  – Tree Adjoining Grammar and other subsets of tree grammars are available and might be better

• Improvement in Alignment Quality
  – Realigning the data is computationally heavy

• Try out different language pairs
  – EN -> NL
  – NL -> FR
  – FR -> NL

• Enlarging the treebanks