MT with Tree-to-tree Transfer

Ondřej Bojar
obo@cuni.cz

May 5, 2008
Summary

• Implemented a decoder for tree-to-tree transfer:
  – Based on Synchronous Tree Substitution Grammar.
  – Main goal: MT with transfer at the tectogrammatical layer.

Transfer at t-layer should be easier than direct translation:

• Reduced structure size (auxiliary words disappear).
• Reduced vocabulary size (Czech morphological complexity).
• Czech and English t-trees structurally more similar
  ⇒ less parallel data might be sufficient (but more monolingual).
• Dependency context more relevant than adjacency context.
  – Plus long-distance dependencies (non-projectivites) solved at t-layer.
• Word order ignored / interpreted as information structure (given/new).
• Ready for fancy t-layer features: e.g. co-reference.
Tree-to-tree Transfer

Synchronous Tree Substitution Grammar (Čmejrek, 2006) applicable at or across various layers of annotation:

Given an input dependency tree:

- decompose it into known treelets,
- replace treelets by their treelet translations,
- join output treelets and produce output final tree,
  - for a-tree, read off the sequence of words,
  - for t-tree, run rule-based Czech sentence generation (Ptáček, 2005)
Idea: Observe a Pair of Dependency Trees

# Asociace uvedla, že domácí poptávka v září stoupla.

# The association said domestic demand grew in September.
Idea: Decompose Trees into Treelets

Asociace uvedla, že domácí poptávka v září stoupla.

The association said domestic demand grew in September.
Idea: Collect Dictionary of Treelet Pairs

Internal nodes (●) introduce words. Frontiers (⊙) allow to attach other treelets (there is no adjunction in STSG).
Treelet Pair Formally, Synchronous Derivation

A **treelet pair** $t_{1:2}$ is a tuple $(t_1, t_2, m)$ where:

- $t_1$ and $t_2$ are little trees for source and target languages ($L_1$ and $L_2$) and states ($Q_1$ and $Q_2$),
- $m$ is a 1-1 **mapping** of frontier nodes in $t_1$ and $t_2$.

Unlike Čmejrek (2006), I require all frontier nodes mapped, i.e. equal number of left and right frontier nodes.

From a starting **synchronous state** $Start_{1:2} \in Q_1 \times Q_2$, a **synchronous derivation** $\delta$ constructs a pair of dependency trees by:

- attaching treelet pairs $t_{1:2}^0, \ldots, t_{1:2}^k$ at corresponding frontier nodes, and
- ensuring that the root states $q_{1:2}^0, \ldots, q_{1:2}^k$ of the attached treelet pairs $t_{1:2}^0, \ldots, t_{1:2}^k$ match the frontier states of the corresponding frontier nodes.

Can define probability of a derivation: $p(\delta) = p(t_{1:2}^0|Start_{1:2}) \times \prod_{i=1}^k p(t_{1:2}^k|q_{1:2}^k)$
Decoding STSG

• Find target tree such that the synchronous derivation $\delta$ is most likely.
• Implemented as two-step top-down beam-search similar to Moses:

1. Prepare **translation options table**:
   - For every source node consider every subtree rooted at that node.
   - If the subtree matches the source treelet in a treelet pair, we’ve got a translation option.
   - Keep only best $\tau$ translation options at a node.

2. Gradually **expand partial hypotheses**:
   - Starting at root use translation options to cover source tree.
   - Keep only best $\sigma$ partial hypotheses of a given size (input nodes covered).
The association said demand grew. 

Sample translation options at root:  

⇒ # _Pred _AuxK  

⇒ # _Pred .

Sample translation options at 'said':  

⇒ _Sb uvedla , že _Pred  

Sample translation options at '.':  

⇒ .
Hypothesis Expansion Example

The association said demand grew.

Sample Derivation:

- \( h_0 \) # ⇒ #
- \( h_1 \) #
- \( h_2 \) #
- \( h_3 \) #

Linearized output:

- ⇒ _#_
- ⇒ # _Pred_
- ⇒ # _Sb uvedla , že _Pred_
- ⇒ # _Sb uvedla , že _Sb stoupla .

Ondřej Bojar MT with Tree-to-tree Transfer May 5, 2008
## Empirical Evaluation

<table>
<thead>
<tr>
<th>Tree-based Transfer</th>
<th>LM Type</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>eaca</td>
<td>(n)-gram</td>
<td>8.8±0.6</td>
</tr>
<tr>
<td>eaca</td>
<td>none</td>
<td>6.6±0.5</td>
</tr>
<tr>
<td>etca</td>
<td>(n)-gram</td>
<td>6.3±0.6</td>
</tr>
<tr>
<td>etcct factored, preserving structure</td>
<td>binode</td>
<td>5.6±0.5</td>
</tr>
<tr>
<td>etcct factored, preserving structure</td>
<td>none</td>
<td>5.3±0.5</td>
</tr>
<tr>
<td>eact, target side atomic</td>
<td>binode</td>
<td>3.0±0.3</td>
</tr>
<tr>
<td>etcct, nodes atomic, all node attributes</td>
<td>binode</td>
<td>2.6±0.3</td>
</tr>
<tr>
<td>etcct, nodes atomic, all node attributes</td>
<td>none</td>
<td>1.6±0.3</td>
</tr>
<tr>
<td>etcct, nodes atomic, just t-lemmas</td>
<td>none</td>
<td>0.7±0.2</td>
</tr>
</tbody>
</table>

Phrase-based (Moses) as reported by Bojar (2007)

<table>
<thead>
<tr>
<th></th>
<th>LM Type</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla</td>
<td>(n)-gram</td>
<td>12.9±0.6</td>
</tr>
<tr>
<td>Factored to improve target morphology</td>
<td>(n)-gram</td>
<td>14.2±0.7</td>
</tr>
</tbody>
</table>
Discussion and Future Research

Why target t-layer performs so poorly?

• errors accumulate (noisy input parse, noisy transfer, noisy generation),
• training data loss due to incompatible parses and node alignments,
• combinatorial explosion of t-node attributes,
  Stacks filled up with variations of less relevant attributes, different lexical choices pushed away.
• rule-based generation does not make use of $n$-gram language model,
  BLEU disfavours methods without language models.
• too many parameters $\Rightarrow$ minimum error rate training does not converge.

Future:

• Improve preprocessing pipeline (use more recent taggers, parsers. . . ).
• Allow customization of the search: choose words and forms first, translate fine-grained attributes later.
References

