Dependency tree projection across parallel texts

David Mareček
marecek@ufal.mff.cuni.cz

Charles University in Prague
Institute of Formal and Applied Linguistics

FEAST
Saarbrücken, Germany, October 18, 2010
Motivation

- For many languages we don’t have any manually annotated data for training statistical parsers

- But, for many of these languages, there exists some form of parallel corpus
  - often with English

- Our goal is:
  - make a word-alignment on this parallel corpus
  - run a statistical dependency parser on the English side
  - transfer the dependencies from English to our language using the alignment
  - train the parser on the resulting trees
Outline

- Word alignment
  - uni-directional alignment
  - symmetrization methods

- Algorithm for projecting dependencies using alignment
  - projecting tags in case we don’t have any tagger

- Training and evaluating MST parser
  - using tagger trained on manually annotated corpus
  - tags projected from English across our parallel corpus

- Ways how to filter out the noise from training data
  - recognition of the bad trees
**Word alignment**

- GIZA++ toolkit [Och and Ney, 2003]

- assymetric output:
  - For each word in one language a counterpart from the other language is found

- GIZA++ is run in both the directions and then it can be symmetrized
  - English-to-X
  - X-to-English
  - Intersection symmetrization
  - Grow-diag-final-and symmetrization
Coordination of fiscal policies indeed, can be counterproductive.

Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein.
Coordination of fiscal policies indeed, can be counterproductive.

Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein.
Word alignment - example

- “Intersection” symmetrization
  - intersection of previous two unidirectional alignments

Coordination of fiscal policies indeed, can be counterproductive.

Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein.
Word alignment - example

- “Ground-diag-final-and” symmetrization
  - links from intersection
  - links where one or two of its ends are neighbouring with some already added link

Coordination of fiscal policies indeed, can be counterproductive.

Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein.
Alignment links used for the projection

- We use only such links that appeared in unidirectional X-to-English alignment
  - we need to find some parent for each token in the language X
  - we don’t care about not aligned English words

- We recognize three weights of links
  - 1: links that appeared only in X-to-English alignment (blue)
  - 2: links that appeared also in “grow-diag-final-and” symmetrization (yellow)
  - 3: links that appeared in “intersection” symmetrization (red)

Coordination of fiscal policies indeed, can be counterproductive.

Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein.
The Algorithm for dependency projection

e_root = technical root of English parse tree;
f_root = technical root of foreign parse tree;
build_subtree(e_root, f_root);

function build_subtree(e_node, f_node);
begin
    for e_child in e_node->get_children do begin
        links = all alignment links leading from e_child;
        if not links then
            build_subtree(e_child, f_node);
        else begin
            main_link = the link with the highest weight (or the first one of them);
            main_f_child = the node which is connected to e_child by main_link;
            other_f_children = nodes connected to e_child by other links;
            main_f_child->set_parent(f_node);
            main_f_child->set_tag(e_child->get_tag);
            for f_child in other_f_children do f_child->set_parent(main_f_child);
            build_subtree(e_child, main_f_child);
        end;
    end;
end;
Algorithm - example

Coordination of fiscal policies indeed, can be counterproductive.

Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein.
Training a parser – tagging

- We need some tags for training the parser.

- If we have a tagger for our language,
  - we can use it and tag our data.

- In case we don’t have any tagger and any human-annotated corpus,
  - we can make a projection of the English tags into our language
  - we assign a special tag ‘??’ to tokens that haven’t got any tag by the projection
Projecting tags

Coordination of fiscal policies indeed, can be counterproductive.
Filtering the training data

- Why filtering? A lot of noise in the data.
  - non-parallel sentences
  - very free translations
  - very strange trees

- Training a parser on the whole corpus would take a lot of time
  - hours, days, weeks...

- We will filter out such trees that have wrong alignment (alignment sparseness)

- We will filter out such trees that have a lot of non-projective dependencies
  - often caused by wrong alignment
Alignment sparseness limit

- The sentence is not good if there are not many intersection links related to the length of the sentences.

\[ S = 1 - \frac{\#\text{links}}{(e\_\text{length} + f\_\text{length}) / 2} \]

- We filter out all the sentences with sparseness greater than some limit \( S_{\text{max}} \).
The sentence is not good if there are many non-projective edges in the projected tree.

We filter out all sentences that have more non-projectivities than some limit.

Mesured for $S=0.25$
Experimental setup

- Languages used:
  - Czech
  - German

- Parallel Corpora
  - Project Syndicate (news-commentaries from WMT10 translation task)
  - about 100,000 parallel sentences

- Treebanks
  - dtest sets from CoNLL shared task 2006/2007

- Parser
  - Maximum spanning tree parser [McDonald, 2005]

- Tagger
  - Morce tagger for English [Spoustova, 2007]
  - Tree-tagger [Schmidt, 1994]
Results

- The best results were achieved with the following filtering:
  - We filter out all the trees with at least one non-projective dependency
  - We filter out all trees where the alignment sparseness $S$ was greater than 0.25.

<table>
<thead>
<tr>
<th>Language</th>
<th>Tags</th>
<th>Sentences</th>
<th>Accuracy</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>by tagger</td>
<td>15,762</td>
<td>62.0 %</td>
<td>10.7 %</td>
</tr>
<tr>
<td>German</td>
<td>by tagger</td>
<td>17,368</td>
<td>55.7 %</td>
<td>14.9 %</td>
</tr>
<tr>
<td>Czech</td>
<td>projected</td>
<td>15,762</td>
<td>53.5 %</td>
<td>7.14 %</td>
</tr>
<tr>
<td>German</td>
<td>projected</td>
<td>17,368</td>
<td>54.2 %</td>
<td>11.7 %</td>
</tr>
</tbody>
</table>
Conclusions

- We proved that it’s possible to create a dependency parser without having a manually annotated treebank.
- The unlabeled accuracy is about 60%.
- We tested it on languages for which we have some treebank.
- The problem of testing is in a different annotation guidelines for each treebank.
Thank you for your attention