Improving MT Quality Prediction with Syntactic Tree Kernels

Christian Hardmeier

Uppsala universitet

2011-05-31
Confidence estimation for MT

- MT output is not perfect.
- When it is too bad, post-editors just waste time discarding it and have to translate from scratch anyway.
- Productivity could be increased by discarding sentences automatically if they are unlikely to be good translations.
Confidence estimation for MT

- *Confidence estimation* (or *quality estimation*) aims at predicting the quality of MT output based on input, output, models etc.

- We present results on sentence-level MT confidence estimation with Support Vector Machine classification.

- Using *tree kernels* drastically reduces the effort required to create a confidence estimation system while delivering quite good results.
Subtitle dataset

- English-Swedish datasets
- about 4,000 subtitles from different TV series
- post-edited and annotated by professional translators
- manual quality judgments on a scale from 1 to 4
Quality annotation scheme

1. MT output unusable.
   Subtitle needs to be retranslated from scratch.

2. Post-editing quicker than retranslation.
   “I needed to think about whether or not the MT output was usable.”

3. Only quick post-editing required.
   “I could see almost immediately what I had to change.”

4. MT output fit for purpose, no changes required.

1 and 2: negative class
3 and 4: positive class
Europarl datasets

- Europarl test sets annotated for translation quality
- Published by Lucia Specia et al. (LREC 2010)
- Annotated on a 1–4 scale similar to ours
- 4,000 sentences translated by 4 different MT systems
- Results for confidence estimation with this data set presented by Specia et al. in *Machine Translation* 24 (2010) and two conference papers (EAMT 2009, MT Summit 2009)
Explicit vs. implicit features

Explicit features

designed in a manual feature engineering process,
extracted with special-purpose tools
labour-intensive but specific

Implicit features

automatically extracted by a general-purpose method
e. g. tree kernels
Explicit features

For all systems:
- number of words, length ratio
- type-token ratio
- number of tokens matching particular patterns such as punctuation, short and long words etc.
- source and target language model scores
- OOV ratio
- word frequencies in training corpus

Only for subtitle system:
- some more specific token counts
- short output indicator
- word alignment types in phrases
Tree kernels

- In SVM learning, features can be represented implicitly by using kernel functions.
- Kernel functions can be defined over structures such as parse trees.
- Tree kernels measure similarities between trees by counting common substructures.
Parse trees

- Constituency parses (Stanford parser):
  - English

- Dependency parses (MaltParser):
  - English
  - Swedish (subtitle test set)
  - Spanish (Europarl test sets)

- Experiments used
  - either *constituency parses* for the MT *input only*
  - or *dependency parses* for both MT *input and output*
The Subset Tree Kernel counts substructure that correspond to *complete productions* in a constituency tree.

- If a fragment contains one child of a node, it must contain them all.
- used for *constituency trees*
Tree kernels for dependency trees

Handle POS tags and edge labels by putting them as nodes into the tree.
Experiments

- binary SVM classifiers
- 3rd degree polynomial kernel for explicit features
- Baseline: majority class classifier, accepts everything
Results: Subtitle system

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>majority class</td>
<td>50.2</td>
<td>100.0</td>
<td>66.8</td>
</tr>
<tr>
<td>explicit features</td>
<td>69.5</td>
<td>58.3</td>
<td>63.3</td>
</tr>
<tr>
<td>constituency (S)</td>
<td>64.8</td>
<td>67.0</td>
<td>65.9</td>
</tr>
<tr>
<td>dependency (S+T)</td>
<td>64.9</td>
<td>65.7</td>
<td>65.3</td>
</tr>
<tr>
<td>all + constituency (S)</td>
<td>67.5</td>
<td>66.3</td>
<td>66.8</td>
</tr>
<tr>
<td>all + dependency (S+T)</td>
<td>68.7</td>
<td>68.8</td>
<td>68.8</td>
</tr>
</tbody>
</table>
### Results: Europarl system 2

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>majority class</td>
<td>54.6</td>
<td>100.0</td>
<td>70.6</td>
</tr>
<tr>
<td>explicit features</td>
<td>67.1</td>
<td>82.7</td>
<td>74.0</td>
</tr>
<tr>
<td>constituency (S)</td>
<td>69.0</td>
<td>69.9</td>
<td>69.4</td>
</tr>
<tr>
<td>dependency (S+T)</td>
<td>69.5</td>
<td>68.2</td>
<td>68.8</td>
</tr>
<tr>
<td>all + constituency (S)</td>
<td>74.4</td>
<td>73.3</td>
<td>73.9</td>
</tr>
<tr>
<td>all + dependency (S+T)</td>
<td>74.1</td>
<td>76.2</td>
<td>75.1</td>
</tr>
</tbody>
</table>
Results: Europarl systems 1–3

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>majority class</td>
<td>83.0</td>
<td>70.6</td>
<td>68.3</td>
</tr>
<tr>
<td>explicit features</td>
<td>83.5</td>
<td>74.0</td>
<td>70.4</td>
</tr>
<tr>
<td>constituency (S)</td>
<td></td>
<td>69.4</td>
<td>66.9</td>
</tr>
<tr>
<td>dependency (S+T)</td>
<td></td>
<td>68.8</td>
<td>68.1</td>
</tr>
<tr>
<td>all + constituency (S)</td>
<td><strong>85.1</strong></td>
<td>73.9</td>
<td>73.4</td>
</tr>
<tr>
<td>all + dependency (S+T)</td>
<td>84.8</td>
<td><strong>75.1</strong></td>
<td><strong>73.9</strong></td>
</tr>
</tbody>
</table>
## Results: Accuracy

|                      | Europarl |      |      | sub-
|----------------------|----------|------|------|-------
|                      | 1        | 2    | 3    | titles|
| majority class       | 71.0     | 54.6 | 51.8 | 50.2  |
| Specia et al., MT 24 (2010) | 76.8     | 66.0 | 69.8 |       |
| explicit features    | 72.6     | 68.7 | 70.3 | 66.4  |
| constituency tree kernel (S) | 66.4     | 66.9 | 64.7 |       |
| dependency tree kernel (S+T) | 66.4     | 67.8 | 65.0 |       |
| explicit + constituency (S) | **77.8** | 71.1 | 72.5 | 66.7  |
| explicit + dependency (S+T) | 76.7     | **72.4** | **72.8** | **68.3** |
Conclusions

- Tree kernels alone achieve only slightly lower performance for most test sets at reduced development effort.
- Combining tree kernels with explicit features led to a small improvement for all test sets.
- Use tree kernels when you start building a confidence estimation system, then add more features to improve performance...
- ...or improve tree kernel approach to use more information.