CoNLL-2005 shared task spotlight: Partial vs full parsing in SRL

Lluís Màrquez
TALP Research Center, Software Department
Universitat Politècnica de Catalunya

Ann Arbor, June 30, 2005
Motivation

- One of the conclusions from the 2004 edition:
  - State-of-the art SRL systems based on full parsing perform close to 80 ($F_1$), while systems based on partial parsing perform close to 70.

- Also, results from the 2005 edition are $\sim$10 points better than those of 2004

- How much of this difference is due to the transition from partial to full parsing?
Partial vs full parsing in SRL

Setting

- The marquez system at CoNLL-2005 presents a comparison between two single SRL systems:
  - \textbf{PP}: based on partial parsing (chunks + clauses)
  - \textbf{FP}_{CHA}: based on full parsing (Charniak)
  - Fixed setting for both systems
  - The only difference is input information
  - SRL is approached as a B-I-O sequential tagging task
Setting

- The marquez system at CoNLL-2005 presents a comparison between two single SRL systems:
  - Sequentialization performed by selecting the top-most syntactic elements in the sentence spans defined by clause boundaries
  - Rich set of features based on state-of-the-art
  - Most features common to PP and FP
  - But some complex features make sense only for FP
  - AdaBoost was used to train independent local classifiers for B, I, and O labels
Results

- Overall results of the systems on the development set

<table>
<thead>
<tr>
<th></th>
<th>Perfect props</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>47.38%</td>
<td>76.86%</td>
<td>70.55%</td>
<td>73.57</td>
</tr>
<tr>
<td>FPCHA</td>
<td>51.51%</td>
<td>78.08%</td>
<td>73.54%</td>
<td>75.75</td>
</tr>
</tbody>
</table>

- FPCHA results are better but not that much. Overall PP results are “only” 2.2 points below the best single system on the 2005 task

- FP results are structurally better (% of perfect props)
Results

- A more detailed analysis reveals that the two systems predict better different arguments

<table>
<thead>
<tr>
<th></th>
<th>FPCHA</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>84.03</td>
<td>79.79</td>
</tr>
<tr>
<td>A1</td>
<td>77.88</td>
<td>74.78</td>
</tr>
<tr>
<td>A2</td>
<td>62.36</td>
<td>65.10</td>
</tr>
<tr>
<td>A3</td>
<td>59.02</td>
<td>60.22</td>
</tr>
<tr>
<td>A4</td>
<td>67.86</td>
<td>67.27</td>
</tr>
</tbody>
</table>

- FP performs better in A0–A1 arguments. PP performs better in A2–A4 and some AM arguments.

- This is good for system combination
Results

• Some evidence: overall results of a combined system on the development set (a very naïve combination scheme was used)

<table>
<thead>
<tr>
<th></th>
<th>Perfect props</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>47.38%</td>
<td>76.86%</td>
<td>70.55%</td>
<td>73.57</td>
</tr>
<tr>
<td>$FP_{CHA}$</td>
<td>51.51%</td>
<td>78.08%</td>
<td>73.54%</td>
<td>75.75</td>
</tr>
<tr>
<td>Combin.</td>
<td>51.39%</td>
<td>78.39%</td>
<td>75.53%</td>
<td>76.93</td>
</tr>
</tbody>
</table>

• Recall improves 2%
Other observations

- FP generates shorter token sequences (21% less training examples than PP). Thus, FP trains faster using less memory.

- PP was expected to be more robust on the Brown corpus...
  - But, actually it is worse than FP :-(
  - Explanation: the clause splitter module degrades more than the full parser.