Robust Dependency Parsing

Bernd Bohnet
University Stuttgart
Overview

• Graph-based Dependency Parsing
  – Hash Kernel
  – Parallel Parsing

• Robust Parsing
  – Error Analysis of Out-of-Domain Data (ST 2009)
  – Clusters
  – Adding addition data from the domain

• Conclusions
Graph-based Dependency Parsing

Peter bought a picture.

Build a graph of all words
Graph-based Dependency Parsing

Peter bought a picture

Build a graph of all words

Score edges: $p = w \times \phi(x,y)$

Search highest scoring tree: $\text{argmax} \sum (x,y)$
Graph-based Dependency Parsing

Peter bought a picture

Build a graph
Score edges: \( p = w \ast \phi(x,y) \)
Search highest scoring tree: \( \text{argmax} \sum (x,y) \)
Bottleneck of the Parser: Speed of the SVM

- **Problem**: the feature-index mapping of the SVM needs 80% of the parsing time

Task of the feature-Index mapping:
- maps features to indexes of weights
- has to filter out “negative features”
Hash Kernel

• **Hash Kernel**: feature-index mapping is replaced by a random function (hash function)
  – 3.5 times faster
  – Higher accuracy

⇒ Why do we get a higher accuracy?
Higher Accuracy with the Hash Kernel

<table>
<thead>
<tr>
<th>Language</th>
<th>Catalan</th>
<th>Chinese</th>
<th>Czech</th>
<th>English</th>
<th>German</th>
<th>Japanese</th>
<th>Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top CoNLL</td>
<td>87.86</td>
<td>79.19</td>
<td>80.38</td>
<td>89.88</td>
<td>87.48</td>
<td>92.57</td>
<td>87.64</td>
</tr>
<tr>
<td>Perceptron</td>
<td>86.35</td>
<td>76.11</td>
<td>80.11</td>
<td>89.88</td>
<td>87.48</td>
<td>92.21</td>
<td>87.19</td>
</tr>
<tr>
<td>Hash Kernel</td>
<td>87.45</td>
<td>76.99</td>
<td><strong>80.96</strong></td>
<td><strong>90.33</strong></td>
<td><strong>87.90</strong></td>
<td>92.47</td>
<td><strong>88.13</strong></td>
</tr>
<tr>
<td></td>
<td>2 (+1)</td>
<td>2</td>
<td>1 (+1)</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1 (+1)</td>
</tr>
</tbody>
</table>

- Hash Kernel improved the accuracy well
  ⇒ Average error reduction of about 10%
Improving Accuracy with „Negative“ Features

- Standard Approach: Collects first all „positive“ features from gold dependency trees (ca. 4 Mio. features)
- Our Approach: Use all features also features from “wrong” trees possible since the Hash Kernel can handle that amount well (4000 Mio. Features)

Wrong tree

```
subject+ John -> plays
```

Negative feature.

Correct tree

```
subject+plays->John
```

Feature from a gold tree.
Out-of-Domain

• Results on Out-of-Domain Data (Shared Task 2009)

<table>
<thead>
<tr>
<th>Rank</th>
<th>System</th>
<th>Average</th>
<th>Czech-ood</th>
<th>English-ood</th>
<th>German-ood</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P-columns</td>
<td>73.96</td>
<td>74.37</td>
<td>77.06</td>
<td>70.44</td>
</tr>
<tr>
<td>1</td>
<td>Bohnet</td>
<td>78.79</td>
<td>76.40</td>
<td>@ 82.64</td>
<td>@ 77.34</td>
</tr>
<tr>
<td>2</td>
<td>Merlo</td>
<td>78.01</td>
<td>@ 76.41</td>
<td>80.84</td>
<td>76.77</td>
</tr>
<tr>
<td>3</td>
<td>Chen</td>
<td>77.96</td>
<td>75.58</td>
<td>82.36</td>
<td>75.93</td>
</tr>
<tr>
<td>4</td>
<td>Che</td>
<td>77.90</td>
<td>76.03</td>
<td>81.57</td>
<td>76.11</td>
</tr>
<tr>
<td>5</td>
<td>Zhang</td>
<td>75.95</td>
<td>71.29</td>
<td>81.50</td>
<td>75.06</td>
</tr>
<tr>
<td>6</td>
<td>Lluís</td>
<td>75.09</td>
<td>72.11</td>
<td>80.92</td>
<td>72.25</td>
</tr>
</tbody>
</table>
Error Analysis

- Corpus: Europal
- Drop for German from 87.48 LAS to 77.06 LAS
  ⇒ accuracy drop of 10.4 points!
- Errors are cause by
  - "Spelling" errors (ü,ö,ä,ß -> ue, oe,etc. )
  - Tagging accuracy went down to 86 (95.5 in-domain)
  - Unknown words
- Retagging brings accuracy back on track
  - Correcting the spelling errors and retagging gives us 96 %
    tagging accuracy and 84.7 LAS
  - Unknown words remain
Cluster improve Results

• E.g. Brown-cluster – one of the simplest methods
• The input is a large corpus of words. The output is a tree of words, each with an assigned bit-string. An example output is shown below in list form.

<table>
<thead>
<tr>
<th>Word</th>
<th>Bit-string</th>
</tr>
</thead>
<tbody>
<tr>
<td>lawyer</td>
<td>1000001101000</td>
</tr>
<tr>
<td>newspaperman</td>
<td>100000110100100</td>
</tr>
<tr>
<td>stewardess</td>
<td>100000110100101</td>
</tr>
<tr>
<td>toxicologist</td>
<td>100000110100111</td>
</tr>
<tr>
<td>slang</td>
<td>100000110101000</td>
</tr>
<tr>
<td>babysitter</td>
<td>100000110101100</td>
</tr>
<tr>
<td>conspirator</td>
<td>100000110110100</td>
</tr>
<tr>
<td>womanizer</td>
<td>100000110110111</td>
</tr>
<tr>
<td>mailman</td>
<td>100000111000100</td>
</tr>
<tr>
<td>salesman</td>
<td>100000111011000</td>
</tr>
<tr>
<td>bookkeeper</td>
<td>100000111010000</td>
</tr>
<tr>
<td>troubleshooter</td>
<td>100000111010010</td>
</tr>
<tr>
<td>bouncer</td>
<td>100000111010011</td>
</tr>
<tr>
<td>technician</td>
<td>100000111010001</td>
</tr>
<tr>
<td>janitor</td>
<td>100000111010101</td>
</tr>
<tr>
<td>saleswoman</td>
<td>100000111010110</td>
</tr>
</tbody>
</table>

...
Impact of Cluster

• Unknown words are one of the major reasons for the drop of accuracy
  ⇒ Cluster capture for these cases

• Clusters have a good effect on tagging and parsing performance
  – Tagging German: 97.25 $\Rightarrow$ 97.6
  – Parsing German: 88.3 $\Rightarrow$ 88.4 LAS
## Clusters and Tagging

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cluster</th>
<th>Base line</th>
</tr>
</thead>
<tbody>
<tr>
<td>german dev set:</td>
<td>0.975</td>
<td>0.946</td>
</tr>
<tr>
<td>OOD-Shared Task 2009:</td>
<td>0.963</td>
<td>-</td>
</tr>
<tr>
<td>smultron economy:</td>
<td>0.913</td>
<td>-</td>
</tr>
<tr>
<td>smultron sophies world:</td>
<td>0.964</td>
<td>-</td>
</tr>
</tbody>
</table>
Adding Domain Data

• Data sets available, e.g. OntoNotes, Hungarian
• General observations:
  – Best results with training only in the domain
  – Adding data form domain helps to newspaper part helps
  – We saw for Hungarian a drop on one data set when we added to the newspaper texts other data
Conclusion

• Techniques are available in order to improve accuracy on out-of-domain data
  – Clusters
  – Spelling correction
  – Additional training data of the domain
• Many open question and ideas
  – What is the reason for the drop (using a distinct tool)?
  – Optimization of the feature set?
  – Removing partly the lexicalization of the tagger and parser improves the quality for unknown words
Thank you!