Szeged at EPE 2017: First Experiments in a Generalized Syntactic Parsing Framework

Zsolt Szántó, Richárd Farkas
Introduction

• Motivations
  – Utilize the graph based input format
  – Different parser models to different tasks
• 3 methods
  – Quick experiments
Parse Distribution as Input for Downstream Applications

- Traditional way: one parse for one document
- Distribution of the edges
- K-best parsing
- Similar to the product parsing techniques in constituent parsing
Parse Distribution as Input for Downstream Applications

- 10-best parsing
- Keep all edges in graph format
- Count of same edges added as new property
- For all experiments: mate parser
- MST-Parser 10-best (Unidep 2.0)
- The downstream applications only handled one edge between two nodes

→ We couldn’t improve our scores
Constituents in the Relational Representation

• Several applications might prefer constituent parses instead of dependency trees
  – Scope detection

• Many ways to convert constituents to the EPE’s graph format
Constituents in the Relational Representation

- Create new nodes for constituents
- Edges: child–parent relations
Constituents in the Relational Representation

- Create new nodes for constituents
- Edges: Constituent - generated words
Constituents in the Relational Representation

• Create new nodes for constituents
• Assign their scope to them
Constituents in the Relational Representation

- We used the combination of 2nd and 3rd method

- Mate + Berkeley Parser
Constituents in the Relational Representation

- Results on negation detection task:

<table>
<thead>
<tr>
<th></th>
<th>mate</th>
<th>mate + berkeley</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
<td>test</td>
</tr>
<tr>
<td>Scope Match</td>
<td>78.42</td>
<td>80.00</td>
</tr>
<tr>
<td>Scope Tokens</td>
<td>86.64</td>
<td>89.17</td>
</tr>
<tr>
<td>Event Match</td>
<td>75.47</td>
<td>67.90</td>
</tr>
<tr>
<td>Full Negation</td>
<td>62.15</td>
<td>61.98</td>
</tr>
</tbody>
</table>
Label Set Adjustment Driven by Downstream Applications

- Different downstream applications might utilize different types of grammatical patterns
- Collapse labels by downstream applications
Label Set Adjustment Driven by Downstream Applications

- Replace one label with another
- Run the downstream application
- Calculate scores for all possible replacements
  - Complete graph
  - Nodes: dependency label
  - Edges: score for the application
- Keep the outcoming edges with maximum weight for all nodes
Label Set Adjustment Driven by Downstream Applications

- Started replacing the nodes from the highest edge weight to the lowest
- Calculated scores at each step
Label Set Adjustment Driven by Downstream Applications

• Experiments
  – Turku Event Extraction System
  – Replacement in prediction step
  – Best result after collapsing of 3 labels

<table>
<thead>
<tr>
<th></th>
<th>Event Extraction</th>
<th>Negation Resolution</th>
<th>Opinion Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>mate</td>
<td>47.84</td>
<td>61.98</td>
<td>65.87</td>
</tr>
<tr>
<td>mate + label adjusment</td>
<td>47.37</td>
<td>60.53</td>
<td>66.33</td>
</tr>
</tbody>
</table>
Summary

• 3 methods
  – Used k-best parse by the downstream application
  – Constituent and dependency trees in general framework
  – Collapsed edge labels according to downstream application

• All methods require future work and detailed analysis