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

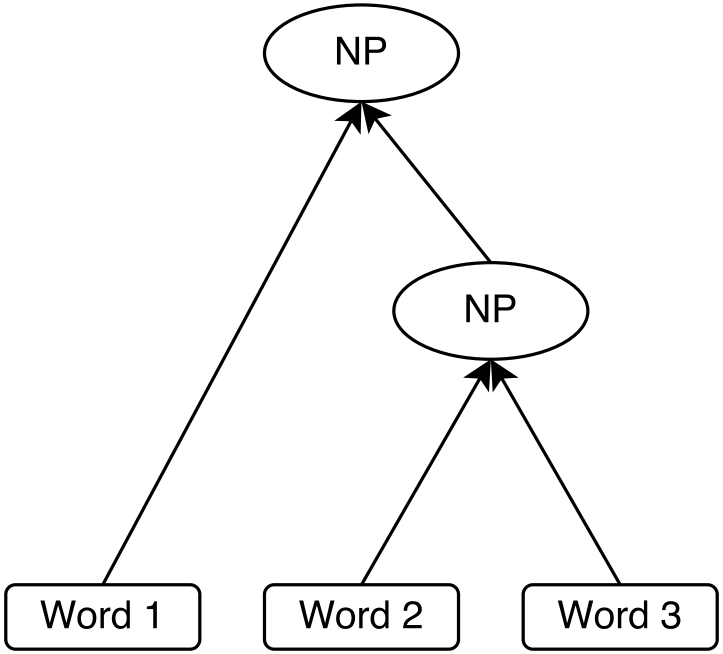
- **10-best parsing**
- **Keep all edges in graph format**
- **Count of same edges added as new property**
- **For all experiments: mate parser**
- **MST-Parser 10-best (Unidep 2.0)**
- **The downstream applications only handled one edge between two nodes –> We couldn’t improve our scores**

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**

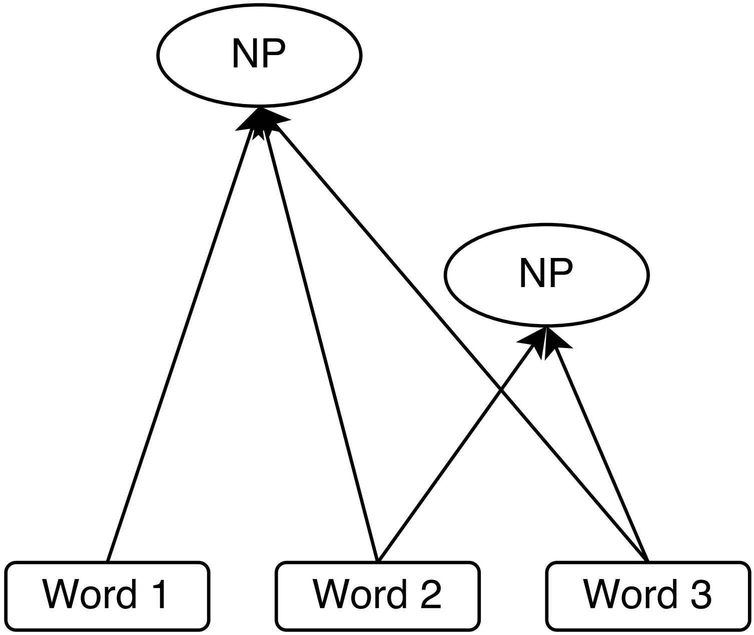
Constituents in the Relational Representation

- **Create new nodes for constituents**
- **Edges: child–parent relations**



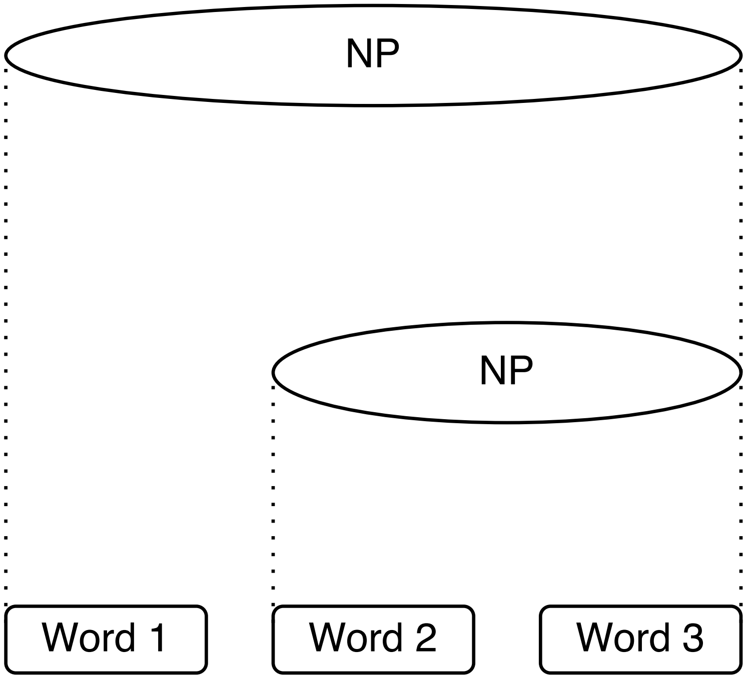
Constituents in the Relational Representation

- **Create new nodes for constituents**
- **Edges: Constituent - generated words**



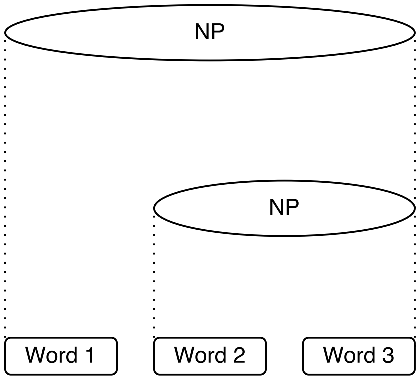
Constituents in the Relational Representation

- **Create new nodes for constituents**
- **Assign their scope to them**

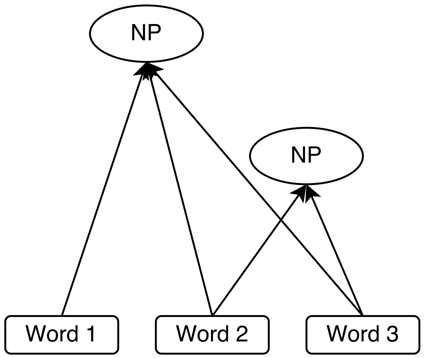


Constituents in the Relational Representation

- **We used the combination of 2nd an 3rd method**



- **Mate + Berkeley Parser**



Constituents in the Relational Representation

- **Results on negation detection task:**

	mate		**mate + berkeley**	
	dev	test	dev	test
Scope Match	**78.42**	80.00	77.98	**81.14**
Scope Tokens	86.64	89.17	**87.38**	**89.27**
Event Match	**75.47**	**67.90**	72.90	65.20
Full Negation	**62.15**	**61.98**	59.91	61.26

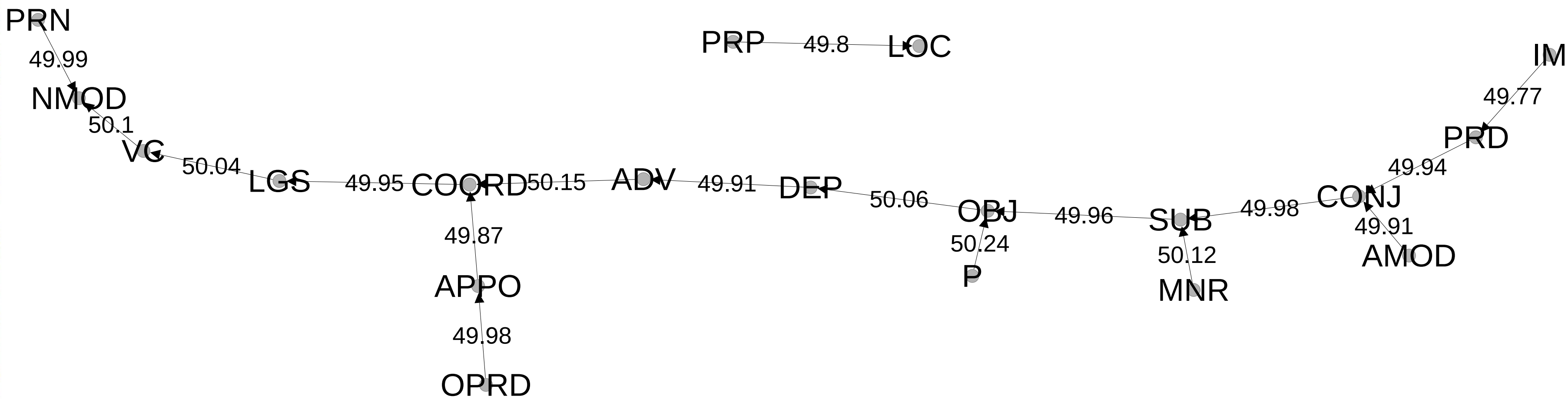
Label Set Adjustment Driven by Downstream Applications

- **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**

	Event Extraction	Negation Resolution	Opinion Analysis
mate	**47.84**	**61.98**	65.87
mate + label adjusment	47.37	60.53	**66.33**

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