Chart-based RRG parsing  
using an automatically extracted  
RRG grammar with features

David Arps & Tatiana Bladier & Laura Kallmeyer

International Conference on Role and Reference Grammar  
19 August 2019  
University at Buffalo, The State University of New York
Overview

Design of RRG Grammars

Automatic RRG Grammar Extraction

Parsing experiments

Issues

Summary & Outlook
Introduction

1. RRG annotated treebank: RRRBank [2],

2. extract RRG elementary trees automatically,

3. use these elementary trees for RRG parsing.
Introduction

1. RRG annotated treebank: RRGBank [2],

2. extract RRG elementary trees automatically,

3. use these elementary trees for RRG parsing.
Introduction

1. RRG annotated treebank: RRGBank [2],

2. extract RRG elementary trees automatically,

3. use these elementary trees for RRG parsing.
Outline

Design of RRG Grammars

Automatic RRG Grammar Extraction

Parsing experiments

Issues

Summary & Outlook
RRGbank

★ Corpus of RRG annotated sentences [2] → automatically converted from Penn Treebank, → manually checked and validated;
★ 395 gold sentences, 1090 silver annotated sentences;
★ RRGbank and RRG annotation tool: rrgbank.phil.hhu.de.
Removing crossing branches

- we transform the RRG structures to remove crossing branches,
Removing crossing branches

★ we transform the RRG structures to remove crossing branches,
★ we mark the original position of the node with \([\text{OP=CL}]\),
Removing crossing branches

- we transform the RRG structures to remove crossing branches,
- we mark the original position of the node with \([OP=CL]\),
- original tree structure is easily recovered.
Operator projection and periphery can be recovered
Elementary trees in RRG Grammars

- We follow Kallmeyer et al. (2013) and Osswald & Kallmeyer (2018) [3, 4] in design of the elementary trees in our grammar.
Elementary trees in RRG Grammars

- We follow Kallmeyer et al. (2013) and Osswald & Kallmeyer (2018) [3, 4] in design of the elementary trees in our grammar.

- Three tree composition operations:
Elementary trees in RRG Grammars

- We follow Kallmeyer et al. (2013) and Osswald & Kallmeyer (2018) [3, 4] in design of the elementary trees in our grammar.

- Three tree composition operations:
  \[ \rightarrow substitution \text{ (argument slot filling)} \]
Elementary trees in RRG Grammars

- We follow Kallmeyer et al. (2013) and Osswald & Kallmeyer (2018) [3, 4] in design of the elementary trees in our grammar.

- Three tree composition operations:
  - $\rightarrow$ substitution (argument slot filling)
  - $\rightarrow$ wrapping substitution (displaced argument slot filling)
Elementary trees in RRG Grammars

- We follow Kallmeyer et al. (2013) and Osswald & Kallmeyer (2018) [3, 4] in design of the elementary trees in our grammar.

- Three tree composition operations:
  - $\rightarrow$ substitution (argument slot filling)
  - $\rightarrow$ wrapping substitution (displaced argument slot filling)
  - $\rightarrow$ sister adjunction (adding operators and periphery elements);

- Such RRG grammars capture long-distance dependencies
  - for example, WH-movement.
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Substitution and sister adjunction

Sentence: I have felt many aftershocks
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Substitution and sister adjunction

Sentence: *I have felt many aftershocks*
Combination operations: Wrapping substitution

Sentence:  *What does Asterix think Obelix drank*
Combination operations: Wrapping substitution

Sentence: *What does Asterix think Obelix drank*
Combination operations: Wrapping substitution

Sentence: What does Asterix think Obelix drank
Combination operations: Wrapping substitution

Sentence: *What does Asterix think Obelix drank*
Elementary trees with features

RRGbank

Extracted Elementary Trees

★ elementary trees are enhanced with features
Elementary trees with features

RRGbank

Extracted Elementary Trees

* elementary trees are enhanced with features
→ edge features
Elementary trees with features

RRGbank

Extracted Elementary Trees

* elementary trees are enhanced with features
  → edge features
  → node features
Edge features

**RRGbank**

- NP
  - DEF-OP: the
  - CORE-N
  - NUC-N
  - N: average

**Extracted Elementary Trees**

- NP
  - NP*: [DEF +]
    - [DEF +] [OPS [NP +]]
    - [DEF -] [OPS [NP -]]
  - DEF-OP: [OP=NP]
    - the
- NUC-N
  - CORE-N
  - N: average

**Left and right edge feature structures:**
Edge features

RRGbank

Extracted Elementary Trees

Left and right edge feature structures:

☆ unify adjacent structures in the derived tree
Edge features

RRGbank

Extracted Elementary Trees

Left and right edge feature structures:

★ unify adjacent structures in the derived tree
★ model ordering constraints
Edge features

RRGbank

Extracted Elementary Trees

Left and right edge feature structures:

★ unify adjacent structures in the derived tree
★ model ordering constraints
★ percolate upwards until phrasal nodes
Node features

RRGb ank

Extracted Elementary Trees

NP
DEF-OP
the
CORE-N
NUC-N
N
average

NP
DEF-OP

NP*
[DEF +]

[NP +]
[DEF +]

OPS [NP +]
[NP -]

NUC-N

DEF-OP

[OP=NP]

the

CORE-N

NUC-N

N

average

★ One feature structure per node:  
- unify during tree composition  
- store syntactic or syn-sem interface information.

★ Unification successful → accept parse tree
Outline

Design of RRG Grammars

Automatic RRG Grammar Extraction

Parsing experiments

Issues

Summary & Outlook
RRG Grammar extraction algorithm (1)

- Top-down extraction of elementary trees.
- Heuristics from head-modifier percolation tables.
- We use RRG structures from RRGbank for automatic grammar induction
  → rrgbank.phil.hhu.de.
RRG Grammar extraction algorithm (2)

The average dropped 27 points

ROOT
CLAUSE
CORE

NP

CORE_N

NUC_N

DEF-OP

N

TNS-OP [OP=CL]

V

CD

N

NP

CORE_N

NUC_N

DEF-OP [OP = NP, DEF = +]

The

average

NP

CORE_N

NUC_N

TNS-OP [OP = CL, TNS = past]

had

dropped

27

points
Outline

Design of RRG Grammars

Automatic RRG Grammar Extraction

Parsing experiments

Issues

Summary & Outlook
Chart-based RRG Parser TuLiPA

- input: set of elementary trees and sentences to parse;
Chart-based RRG Parser TuLiPA

- input: set of elementary trees and sentences to parse;
- output: all derivations that can be derived by combining the elementary trees;
Chart-based RRG Parser TuLiPA

- input: set of elementary trees and sentences to parse;
- output: all derivations that can be derived by combining the elementary trees;
- standard CYK algorithm;
Chart-based RRG Parser TuLiPA

- input: set of elementary trees and sentences to parse;
- output: all derivations that can be derived by combining the elementary trees;
- standard CYK algorithm;
- bottom-up, left-to-right traversal of the derived tree;
Chart-based RRG Parser TuLiPA

- **input:** set of elementary trees and sentences to parse;
- **output:** all derivations that can be derived by combining the elementary trees;
- **standard CYK algorithm;**
- **bottom-up, left-to-right traversal of the derived tree;**
- **software:** TuLiPA RRG parser [1] (https://github.com/spetitjean/TuLiPA-frames)

TuLiPA = Tübingen Linguistic Parsing Architecture;
Chart-based RRG Parser TuLiPA

- input: set of elementary trees and sentences to parse;
- output: all derivations that can be derived by combining the elementary trees;
- standard CYK algorithm;
- bottom-up, left-to-right traversal of the derived tree;
- software: TuLiPA RRG parser [1] (https://github.com/spetitjean/TuLiPA-frames)
  TuLiPA = Tübingen Linguistic Parsing Architecture;
- suitable for hand-crafted precision RRG grammars;
Chart-based RRG Parser TuLiPA

- input: set of elementary trees and sentences to parse;
- output: all derivations that can be derived by combining the elementary trees;
- standard CYK algorithm;
- bottom-up, left-to-right traversal of the derived tree;
- software: TuLiPA RRG parser [1] (https://github.com/spetitjean/TuLiPA-frames) TuLiPA = Tübingen Linguistic Parsing Architecture;
- suitable for hand-crafted precision RRG grammars;
- suitable for automatically extracted RRG grammars.
Extracted RRG grammar

- removed punctuation
- exhaustive parsing (i.e. not probabilistic, overgenerating a lot)
- 2 versions:
  1. no features
  2. edge features for operators model adjunction constraints
- do feature structures eliminate parse trees that contradict linguistic intuitions?
## Parsing experiments

<table>
<thead>
<tr>
<th></th>
<th>Gold Grammar</th>
<th>Silver Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w feats</td>
<td>w/o feats</td>
</tr>
<tr>
<td>Sentences</td>
<td>395</td>
<td>1480</td>
</tr>
<tr>
<td>avg. sentence length</td>
<td>6.1</td>
<td>8.0</td>
</tr>
<tr>
<td>token-supertag pairs</td>
<td>1526</td>
<td>6288</td>
</tr>
<tr>
<td>avg. number of parses</td>
<td>6.9</td>
<td>1166</td>
</tr>
<tr>
<td><strong>savings</strong></td>
<td>45.1%</td>
<td>39.7%</td>
</tr>
</tbody>
</table>

Features decrease number of results by $\approx 45\%$
Sentence: By 9:45, the industrial average had dropped 27 points.
Sentence: *By 9:45, the industrial average had dropped 27 points.*
Outline

Design of RRG Grammars

Automatic RRG Grammar Extraction

Parsing experiments

Issues

Summary & Outlook
Attachment ambiguities

```
CORE*  \nPP-PERI  \nCORE_P  \nNUC_P  ↓  CORE↓
| P  |
in
NP*  \nPP-PERI  \nCORE_P  \nNUC_P  ↓  NP↓
| P  |
in
CORE_N*  \nPP-PERI  \nCORE_P  \nNUC_P  ↓  NP↓
| P  |
in
CORE*  \nPP-PERI  \nCORE_P  \nNUC_P  ↓  CORE↓
| P  |
in
```

```
CORE  ↓  NP*
NP  ↓  NP
CORE-N  ↓  CORE-N
NUC-N  ↓  N
| N  |
bank

X*  ↓  N
PP-PERI  \nCORE_P  \nTexas

NUC_P  ↓  X↓
| P  |
in
```

syntactic information needed that might not be in RRGBank
## Size of the grammar

<table>
<thead>
<tr>
<th></th>
<th>Gold Grammar</th>
<th></th>
<th>Silver Grammar</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentences</td>
<td>395</td>
<td>1480</td>
<td></td>
<td></td>
</tr>
<tr>
<td>avg. sentence length</td>
<td>6.1</td>
<td>8.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>token-supertag pairs</td>
<td>1526</td>
<td>1497</td>
<td>6288</td>
<td>6044</td>
</tr>
<tr>
<td>avg. number of parses</td>
<td>6.9</td>
<td>12.7</td>
<td>1166</td>
<td>2939</td>
</tr>
</tbody>
</table>

The number of parses per sentence increases with the size of the grammar.
Extraction of wrapping substitution trees

- discontinuous constituents are marked with traces in PTB;
- no special marking in RRGBank;
- transfer traces from PTB to RRG trees in RRGBank?
Outline

Design of RRG Grammars

Automatic RRG Grammar Extraction

Parsing experiments

Issues

Summary & Outlook
Summary

- automatically extracted elementary trees from RRGBank;
- experiments with exhaustive parsing of 395/1480 sentences;
- parsing w/o edge features → too many results;
- some edge features already rule out bad results.
Future Work

- adding more edge features = better results;
- extract wrapping elementary trees;
- 100s of results → not satisfying;
- ambiguity and annotation/extraction mistakes have bad consequences;
- use unlexicalized elementary trees (= supertags);
- probabilistic grammar and parsing → A* parsing algorithm ParTAGe by Waszczuk (2017) [5];
- Web GUI.
Thank you!

THANK YOU VERY MUCH FOR YOUR ATTENTION!
References


