Parsing to Stanford Dependencies: Trade-offs between speed and accuracy

Daniel Cer, Marie-Catherine de Marneffe
Daniel Jurafsky, Christopher D. Manning
About the Representation
Widely used
Semantically-oriented
Slow to extract

Extraction Bottleneck
Stanford lexicalized phrase structure parser

Are There Faster and Better Approaches?
Dependency parsing algorithms
Alternate phrase structure parsers
Stanford Dependencies

Organization

Brief Review of Stanford Dependencies
Properties
Extraction pipeline

Experiments Comparing Parsing Approaches

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Phrase structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaltParser</td>
<td>Berkeley</td>
</tr>
<tr>
<td>MSTParser</td>
<td>Bikel</td>
</tr>
<tr>
<td></td>
<td>Charniak</td>
</tr>
</tbody>
</table>

Search Space Pruning with Charniak-Johnson
Performing **dependency parsing** using a phrase structure parser followed by rule based extraction is **more accurate** and, in some cases, **faster** than using statistical dependency parsing algorithms.
Performing **dependency parsing** using a phrase structure parser followed by rule based extraction is **more accurate** and, in some cases, **faster** than using statistical dependency parsing algorithms.

For English using the Stanford Dependency formalism
Performing **dependency parsing** using a phrase structure parser followed by rule based extraction is **more accurate** and, in some cases, **faster** than using statistical dependency parsing algorithms.

For English using the Stanford Dependency formalism

However, we suspect the results maybe more general
Capture relationships between **content words**

**Syntactic Dependencies**

**Stanford Dependencies**
Start out by extracting syntactic heads
Results in a \textbf{projective} dependency tree
Start out by extracting syntactic heads
Results in a **projective** dependency tree
Bills on ports and immigration
Bills on ports and immigration
Stanford Dependencies

Obtaining the Dependencies

Phrase Structure Parser

Sentence
Stanford Dependencies

Obtaining the Dependencies

Standard Pipeline

Phrase Structure Parser

Sentence

Constituent Parse Tree
Stanford Dependencies

Obtaining the Dependencies

Standard Pipeline

Phrase Structure Parser

Sentence

Constituent Parse Tree

Projective Basic Dependencies
Stanford Dependencies

Obtaining the Dependencies

Standard Pipeline

Phrase Structure Parser

Sentence

Constituent Parse Tree

Projective Basic Dependencies

Collapsed Dependencies
Stanford Dependencies

Obtaining the Dependencies

- Standard Pipeline
  - Phrase Structure Parser
    - Sentence
    - Constituent Parse Tree
    - Projective Basic Dependencies
  - Collapsed Dependencies
# Stanford Dependencies

## Obtaining the Dependencies

<table>
<thead>
<tr>
<th>Phrase Structure Parser</th>
<th>Dependency parser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence</td>
<td>Sentence</td>
</tr>
<tr>
<td>Constituent Parse Tree</td>
<td></td>
</tr>
<tr>
<td>Projective Basic</td>
<td></td>
</tr>
<tr>
<td>Dependencies</td>
<td></td>
</tr>
<tr>
<td>Collapsed Dependencies</td>
<td></td>
</tr>
</tbody>
</table>
# Stanford Dependencies

## Obtaining the Dependencies

<table>
<thead>
<tr>
<th>Phrase Structure Parser</th>
<th>Dependency parser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence</td>
<td>Sentence</td>
</tr>
<tr>
<td>Constituent Parse Tree</td>
<td>Projective Basic Dependencies</td>
</tr>
<tr>
<td>Projective Basic Dependencies</td>
<td>Projective Basic Dependencies</td>
</tr>
<tr>
<td>Collapsed Dependencies</td>
<td></td>
</tr>
</tbody>
</table>

- **Standard Pipeline**
- **Direct Pipeline**
Stanford Dependencies

Obtaining the Dependencies

<table>
<thead>
<tr>
<th>Phrase Structure Parser</th>
<th>Dependency parser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence</td>
<td>Sentence</td>
</tr>
<tr>
<td>Constituent Parse Tree</td>
<td></td>
</tr>
<tr>
<td>Projective Basic</td>
<td>Projective Basic</td>
</tr>
<tr>
<td>Dependencies</td>
<td>Dependencies</td>
</tr>
<tr>
<td>Collapsed Dependencies</td>
<td>Collapsed Dependencies</td>
</tr>
</tbody>
</table>
Experimental

RESULTS BY PIPELINE TYPE
<table>
<thead>
<tr>
<th>Method</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependency parsers</td>
<td>Penn Treebank Sections 2 through 21</td>
<td>Penn Treebank Section 22</td>
</tr>
<tr>
<td><strong>Malt Parser</strong></td>
<td>Nivre Eager, Nivre, Covington</td>
<td>Eisner</td>
</tr>
<tr>
<td><strong>MSTParser</strong></td>
<td>LibLinear, LibSVM</td>
<td>Factored MIRA</td>
</tr>
</tbody>
</table>

**RelEx** CMU Link grammar parser in Stanford compatibility mode
Results by Pipeline Type

Method

**Train** Penn Treebank Sections 2 through 21
**Test** Penn Treebank Section 22

Phrase Structure Parsers

Charniak
Charniak Johnson Reranking
Bikel
Berkeley
Stanford
Results by Pipeline Type

Phrase Structure Parser Accuracy

<table>
<thead>
<tr>
<th>Parser</th>
<th>Labeled Attachment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak Johnson</td>
<td>89</td>
</tr>
<tr>
<td>Berkeley</td>
<td>87</td>
</tr>
<tr>
<td>Bikel</td>
<td>84</td>
</tr>
<tr>
<td>Stanford</td>
<td>83</td>
</tr>
</tbody>
</table>

Best: Charniak Johnson Reranking Parser
Results by Pipeline Type

Phrase Structure Parser Speed

- Berkeley
- Stanford
- Charniak
- Johnson
- Bikel

Parse times similar except for Bikel
Results by Pipeline Type

Dependency Parser Accuracy

<table>
<thead>
<tr>
<th>Pipeline Type</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nivre Eager LibSVM</td>
<td>81</td>
</tr>
<tr>
<td>MSTParser Eisner</td>
<td>78</td>
</tr>
<tr>
<td>Nivre Eager LibLinear</td>
<td>75</td>
</tr>
</tbody>
</table>

Best: Nivre Eager LibSVM
Results by Pipeline Type

Dependency Parser Speed

<table>
<thead>
<tr>
<th>Pipeline Type</th>
<th>Sentences/Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nivre Eager LibLinear</td>
<td>120</td>
</tr>
<tr>
<td>Nivre Eager LibSVM</td>
<td>10</td>
</tr>
<tr>
<td>MSTParser Eisner</td>
<td>1</td>
</tr>
</tbody>
</table>

Fastest: Nivre Eager LibLinear
Comparison of
SPEED AND ACCURACY TRADE-OFFS
Worst Phrase Structure Parser vs. Best Dependency Parser
Speed and Accuracy Trade-Offs

Worst vs. Best Accuracy

<table>
<thead>
<tr>
<th></th>
<th>Phrase Structure</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bikel</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>Stanford</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>Nivre LibSVM</td>
<td>82</td>
<td>80</td>
</tr>
<tr>
<td>MSTParser Eisner</td>
<td>80</td>
<td>78</td>
</tr>
</tbody>
</table>

Worst phrase structure parser better than Best dependency parser
Speed and Accuracy Trade-Offs

Worst vs. Best Accuracy

Labeled Attachment

Phrase Structure

F1

Bikel

86

84

82

80

78

76

Dependency

+3

Stanford

86

84

82

80

78

76

Nivre Eager LibSVM

MSTParser Eisner

Worst phrase structure parser better than Best dependency parser
Best Phrase Structure Parser vs. Best Dependency Parser
Speed and Accuracy Trade-Offs

Best vs. Best Accuracy

<table>
<thead>
<tr>
<th></th>
<th>Phrase Structure</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak</td>
<td>90</td>
<td>82</td>
</tr>
<tr>
<td>Johnson</td>
<td>88</td>
<td>+8</td>
</tr>
<tr>
<td>Berkeley</td>
<td>88</td>
<td>80</td>
</tr>
<tr>
<td>Nivre Eager</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>LibSVM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSTParser Eisner</td>
<td>84</td>
<td></td>
</tr>
</tbody>
</table>

Eight point difference between Best phrase structure and Best dependency parser
Worst vs. Best Speed

**Worst** Dependency Parser vs. **Best** Phrase Structure Parser
Speed and Accuracy Trade-Offs

Worst vs. Best Speed

Worst dependency parser better than Best phrase structure parser
Speed and Accuracy Trade-Offs

Worst vs. Best Speed

Worst dependency parser better than Best phrase structure parser

<table>
<thead>
<tr>
<th>Dependency Parsing</th>
<th>Phrase Structure Parsing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nivre Eager LibSVM</td>
<td>+2</td>
</tr>
<tr>
<td>MSTParser Eisner</td>
<td></td>
</tr>
<tr>
<td>Berkeley</td>
<td></td>
</tr>
<tr>
<td>Stanford</td>
<td></td>
</tr>
</tbody>
</table>
Best Dependency Parser

vs.

Best Phrase Structure Parser
Speed and Accuracy Trade-Offs

Best vs. Best Speed

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Phrase Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nivre Eager LibLinear</td>
<td>103 sentences/second</td>
</tr>
<tr>
<td>Nivre Eager LibSVM</td>
<td></td>
</tr>
<tr>
<td>Berkeley</td>
<td></td>
</tr>
<tr>
<td>Stanford</td>
<td></td>
</tr>
</tbody>
</table>

103 sentences/second difference between Best dependency and Best phrase structure parser
Speed and Accuracy Trade-Offs

Best vs. Best Speed

<table>
<thead>
<tr>
<th></th>
<th>Dependency</th>
<th>Phrase Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nivre Eager</td>
<td>103</td>
<td>+103</td>
</tr>
<tr>
<td>LibLinear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nivre Eager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LibSVM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Berkeley</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stanford</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

103 sentences/second difference between Best dependency and Best phrase structure parser
Accuracy
Use Phrase Structure Parsers
Best choice: Charniak Johnson Reranking

Speed
Use Dependency Parsers
Best Choice: Nivre Eager* with LibLinear

* Actually, any parser in the MaltParser package will do.
Making Use Of

CHARNIAK JOHNSON
SEARCH SPACE PRUNING
Example Search

Best First Search
Best First Search
Charniak Johnson Search Space Pruning

Example Search

Best First Search
Charniak Johnson Search Space Pruning

Example Search

Best First Search

[Diagram showing the process of Best First Search with nodes and arrows]
Charniak Johnson Search Space Pruning

Example Search

Best First Search
Best First Search
Charniak Johnson Search Space Pruning

Example Search

First Complete Parse!
After the First Complete Parse
Count edges expanded so far

Then Expand
Edge count $\times$ Pruning constant more edges

Pruning constant = $T$ parameter /10
Expand edge count x Constant more edges
Charniak Johnson Search Space Pruning

Example Search

**Expand** edge count \( \times \) Constant more edges
Example Search

Expand edge count x Constant more edges
Charniak Johnson Search Space Pruning

Example Search

Expand edge count \( \times \) Constant more edges
Charniak Johnson Search Space Pruning

**Pruning Effects on Accuracy**

Minimal loss of accuracy for moderate pruning

Labeled Attachment

- Default T210
- T50
- T10
Charniak Johnson Search Space Pruning

Pruning Effects on Accuracy

Minimal loss of accuracy for moderate pruning
Pruning Effects on Speed

3x speed gains with moderate pruning
Charniak Johnson Search Space Pruning

Pruning Effects on Speed

3x speed gains with moderate pruning
Speed and Accuracy Trade-Offs

Best vs. Best Accuracy

Best Phrase Structure Parser vs. Best Dependency Parser

with Pruning
Pruning Speed and Accuracy Trade-Offs

Best vs. Best Accuracy

<table>
<thead>
<tr>
<th>Charniak Johnson</th>
<th>Berkeley</th>
<th>Nivre Eager LibSVM</th>
<th>MSTParser Eisner</th>
</tr>
</thead>
<tbody>
<tr>
<td>88 F1</td>
<td>82</td>
<td>82</td>
<td>78 F1</td>
</tr>
</tbody>
</table>

Remember this?
Let’s insert Charniak Johnson T50
**Pruning** Speed and Accuracy Trade-Offs

Best vs. Best Accuracy

- **Phrase Structure**: About as good as Berkeley
- **Dependency**: Charniak Johnson T50

Labeled Attachment

<table>
<thead>
<tr>
<th></th>
<th>Phrase Structure</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak Default</td>
<td>90</td>
<td>-1</td>
</tr>
<tr>
<td>Berkeley</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>Charniak Johnson T50</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Nivre Eager LibSVM</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>MSTParser Eisner</td>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>

About as good as Berkeley
**Pruning** Speed and Accuracy Trade-Offs

Best vs. Best Accuracy

- **Phrase Structure**
  - Charniak Johnson Default: 90
  - Berkeley: 88
  - **Charniak Johnson T50**: 88

- **Dependency**
  - Nivre Eager LibSVM: 80
  - MSTParser Eisner: 74

Much better than best dependency parser
Speed and Accuracy Trade-Offs

Best vs. Best Speed

Best Dependency Parser vs. Best Phrase Structure Parser

with Pruning
<table>
<thead>
<tr>
<th></th>
<th>Sentences/Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nivre Eager LibLinear</td>
<td>120</td>
</tr>
<tr>
<td>Nivre Eager LibSVM</td>
<td>0</td>
</tr>
<tr>
<td>Berkeley</td>
<td>+103</td>
</tr>
<tr>
<td>Stanford</td>
<td>0</td>
</tr>
</tbody>
</table>

Remember this?
Let's insert Charniak Johnson T50
Speed and Accuracy Trade-Offs

Best vs. Best Speed

Let's insert Charniak Johnson T50

...Excluding the 100+ sentences/second parser
Speed and Accuracy Trade-Offs

Best vs. Best Speed

<table>
<thead>
<tr>
<th>Sentences/Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependency</td>
</tr>
<tr>
<td>Phrase Structure</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

- Nivre Eager LibSVM
- Charniak Johnson T50
- Berkeley
- Stanford

Nearly as fast as Nivre Eager using LibSVM
**Pruning Summary**

**Charniak Johnson T50**

**Accuracy** similar to best phrase structure parsers

**Speed** similar to most dependency parsers
Conclusion

For Accuracy
Charniak Johnson
89.1 Labeled Attachment F1

For Speed
Nivre Eager with LibLinear
+100 Sentences/second

For A Good Balance
Charniak Johnson T50
87.6 Labeled Attachment F1
Speed similar to most dependency parsers

Thanks to
Joakim Nivre
Mihai Surdeanu