Multiword Expressions in Dependency Parsing

Joakim Nivre

Uppsala University
Linguistics and Philology
Overview of the Course

1. Introduction to dependency grammar and dependency parsing
2. Graph-based and transition-based dependency parsing
3. Multiword expressions in dependency parsing
4. Practical lab session (MaltParser)
Plan for this Lecture

- Multiword expressions in dependency parsing
  - Linguistic representations
  - Parsing techniques
  - Empirical studies
- Universal Dependencies
  - General principles
  - Multiword expressions
Linguistic Representations

- How do we represent MWEs in dependency trees?
- Do we need to modify the definition of a dependency tree?
- What about different classes of MWEs?
  - Fixed: by and large, in spite of
  - Semi-fixed: part(s) of speech, kick(s/ed) the bucket
  - Flexible: put off, look for, take a photo
- What about discontiguous MWEs?
The Spanning Tree Assumption

- Basic assumption in (current) dependency parsing:
  - Dependency graph for \( x = w_1, \ldots, w_n \) is a spanning tree in \( G_x \)
  - Every token is a node in the dependency tree (spanning)
  - Every node (except the root) has one incoming arc (tree)

- Possible variations:
  - Give up the tree assumption – allow general graphs
  - Give up the spanning assumption – tokens \( \neq \) nodes
## Tokens and Nodes

<table>
<thead>
<tr>
<th>Token</th>
<th>Node</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ordinary word tokens</td>
</tr>
</tbody>
</table>
### Tokens and Nodes

<table>
<thead>
<tr>
<th>Token</th>
<th>Node</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ordinary word tokens</td>
</tr>
<tr>
<td>1</td>
<td>&gt;1</td>
<td>Clitics, contractions</td>
</tr>
</tbody>
</table>

This requires a new type of dependency parser!
## Tokens and Nodes

<table>
<thead>
<tr>
<th>Token</th>
<th>Node</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ordinary word tokens</td>
</tr>
<tr>
<td>1</td>
<td>&gt;1</td>
<td>Clitics, contractions</td>
</tr>
<tr>
<td>&gt;1</td>
<td>1</td>
<td>Multiword expressions</td>
</tr>
</tbody>
</table>

This requires a new type of dependency parser!
## Tokens and Nodes

<table>
<thead>
<tr>
<th>Token</th>
<th>Node</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ordinary word tokens</td>
</tr>
<tr>
<td>1</td>
<td>&gt;1</td>
<td>Clitics, contractions</td>
</tr>
<tr>
<td>&gt;1</td>
<td>1</td>
<td>Multiword expressions</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Punctuation?</td>
</tr>
</tbody>
</table>
# Tokens and Nodes

<table>
<thead>
<tr>
<th>Token</th>
<th>Node</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ordinary word tokens</td>
</tr>
<tr>
<td>1</td>
<td>&gt;1</td>
<td>Clitics, contractions</td>
</tr>
<tr>
<td>&gt;1</td>
<td>1</td>
<td>Multiword expressions</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Punctuation?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Ellipsis?</td>
</tr>
</tbody>
</table>
## Tokens and Nodes

<table>
<thead>
<tr>
<th>Token</th>
<th>Node</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ordinary word tokens</td>
</tr>
<tr>
<td>1</td>
<td>&gt;1</td>
<td>Clitics, contractions</td>
</tr>
<tr>
<td>&gt;1</td>
<td>1</td>
<td>Multiword expressions</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Punctuation?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Ellipsis?</td>
</tr>
</tbody>
</table>

This requires a new type of dependency parser!
MWEs as Special Tokens

By and large multiword expressions are a pain in the neck

- Simplifies parsing if MWEs can be identified prior to parsing
- Limited to contiguous MWEs and awkward for flexible MWEs
- Common in treebanks (about half of the CoNLL-X data sets)
- What about part-of-speech tags?
MWEs as Dummy Dependency Structures

- Canonical structure without syntactic significance
- Special labels distinguish from real dependencies
- Part-of-speech tags may or may not reflect MWE-hood
MWEs as Dummy Dependency Structures

- Canonical structure without syntactic significance
- Special labels distinguish from real dependencies
- Part-of-speech tags may or may not reflect MWE-hood
MWEs as Real Dependency Structures

- Dependency structure reflects real internal structure
- Special labels may be used for subtypes (for example, LVCs)
So what representations should we use?

- Different types of MWEs require different representations
- At one end of the spectrum: *by and large*
  - No point in representing internal syntactic structure
  - Equivalent to a single node in dependency structure
  - Special token or dummy dependencies?
- At the other end: *take a photo*
  - Needs internal structure to allow modification and inflection
  - Real dependencies, special labels?
- What about everything in between?
Parsing Techniques

- Three main approaches:
  - Pre-processing – analyze MWEs before parsing
  - Post-processing – analyze MWEs after parsing
  - Joint processing – analyze MWEs during parsing

- Key question:
  - Does MWE identification help parsing or vice versa or both?
  - The answer may be different for different types of MWEs!
### Techniques and Representations

<table>
<thead>
<tr>
<th>Special tokens</th>
<th>Pre</th>
<th>Joint</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>
Techniques and Representations

<table>
<thead>
<tr>
<th></th>
<th>Pre</th>
<th>Joint</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special tokens</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Dummy dependencies</td>
<td>?</td>
<td>yes</td>
<td>?</td>
</tr>
</tbody>
</table>

If different types of MWEs require different representations, they may require different processing techniques as well!
# Techniques and Representations

<table>
<thead>
<tr>
<th></th>
<th>Pre</th>
<th>Joint</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special tokens</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Dummy dependencies</td>
<td>?</td>
<td>yes</td>
<td>?</td>
</tr>
<tr>
<td>Real dependencies</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

If different types of MWEs require different representations, they may require different processing techniques as well!
If different types of MWEs require different representations, they may require different processing techniques as well!
An Early Study [Nivre and Nilsson 2004]

- Swedish treebank with (limited) MWE annotation:
  - Function words like *in spite of*, *at large*
  - Names like Carl XVI Gustaf, Swedish Academy
  - Numerical expressions like $2 + 2 = 4$

1. Joint processing with dummy dependencies:

```
Ett skott kan på grund av terrängen få samma effekt.
```

2. Preprocessing with special tokens (gold input):

```
Ett skott kan på grund av terrängen få samma effekt.
```

Multiword Expressions in Dependency Parsing
Results

<table>
<thead>
<tr>
<th></th>
<th>MWE</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint</td>
<td>71.1</td>
<td>80.7</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>–</td>
<td>81.6</td>
</tr>
</tbody>
</table>

- Perfect MWE recognition improves parsing accuracy (slightly)
- Typical effects of failing to recognize MWEs:
  - Unusual part-of-speech patterns leads to distorted structure
    *(vad beträffar = as regards)*
  - Different attachment preferences for MWEs and compositional phrases
    *(i regel = as a rule)*
Results

<table>
<thead>
<tr>
<th></th>
<th>MWE</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint</td>
<td>71.1</td>
<td>80.7</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>–</td>
<td>81.6</td>
</tr>
</tbody>
</table>

- Perfect MWE recognition improves parsing accuracy (slightly)
- Typical effects of failing to recognize MWEs:
  - Unusual part-of-speech patterns leads to distorted structure (vad beträffar = as regards)
  - Different attachment preferences for MWEs and compositional phrases (i regel = as a rule)

Similar results observed later for Turkish [Eryiğit et al. 2011]
Regular and Irregular MWEs

[Candito and Constant 2014]

▶ French dependency treebank with dummy MWE dependencies:

▶ Alternative representations for regular MWEs:

▶ PoS patterns used to distinguish regular and irregular MWEs
## Processing Models

<table>
<thead>
<tr>
<th></th>
<th>Irregular</th>
<th>Regular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint</td>
<td>Parser</td>
<td>Parser</td>
</tr>
<tr>
<td>Joint-Reg</td>
<td>Pre</td>
<td>Pre</td>
</tr>
<tr>
<td>Joint-Irreg</td>
<td>Parser</td>
<td>Post</td>
</tr>
<tr>
<td>Pipeline</td>
<td>Pre</td>
<td>Post</td>
</tr>
</tbody>
</table>

- **Pre** = MWEs pre-recognized and merged to single tokens
- **Post** = MWEs recognized after parsing
## Results

<table>
<thead>
<tr>
<th></th>
<th>Dummy MWE</th>
<th>Dummy Overall</th>
<th>Real MWE</th>
<th>Real Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint</td>
<td>73.5</td>
<td>84.5</td>
<td>81.4</td>
<td>86.9</td>
</tr>
<tr>
<td>Joint-Reg</td>
<td>73.3</td>
<td>84.2</td>
<td>80.4</td>
<td>86.6</td>
</tr>
<tr>
<td>Joint-Irreg</td>
<td>75.4</td>
<td>84.4</td>
<td>82.1</td>
<td>87.0</td>
</tr>
<tr>
<td>Pipeline</td>
<td>74.4</td>
<td>83.9</td>
<td>80.4</td>
<td>86.5</td>
</tr>
</tbody>
</table>

- Real dependencies better than dummy dependencies
- Irregular MWEs benefit most from joint processing
- Regular MWEs better identified after parsing?
Light Verb Constructions [Vincze et al. 2013]

- Hungarian dependency treebank with LVC annotation:

  Holnap (Tomorrow) → nagyon (very) → fontos (important) → döntést (decision) → kell (will-have-to) → hoznunk (make-we)

- Can a parser learn to identify light verb constructions?
- How is overall parsing accuracy affected?
Results

<table>
<thead>
<tr>
<th></th>
<th>LVC</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parser plain</td>
<td>–</td>
<td>90.6</td>
</tr>
<tr>
<td>Parser LVC</td>
<td>75.6</td>
<td>90.4</td>
</tr>
<tr>
<td>Post dictionary</td>
<td>21.3</td>
<td>–</td>
</tr>
<tr>
<td>Post C4.5</td>
<td>74.5</td>
<td>–</td>
</tr>
</tbody>
</table>

- Parser improves LVC identification with a marginal drop in overall labeled attachment score
- Parser significantly better than post-classifier on discontiguous LVCs (64.0 > 60.0)
Conclusion

- We have only scratched the surface . . .
- Complex interaction between several factors:
  - MWE types
  - Linguistic representations
  - Processing techniques
- Tentative conclusions:
  - MWE identification can benefit from syntactic context
  - Regular MWEs should be assigned regular syntactic structure
Universal Dependencies

- **Background:**
  - Treebank annotation schemes vary across languages
  - Hard to compare results across languages [Nivre et al. 2007]
  - Hard to evaluate cross-lingual learning [McDonald et al. 2013]
  - Hard to build multilingual systems

- **Universal Dependencies** (http://universaldependencies.github.io/docs/):
  - Stanford universal dependencies [de Marneffe et al. 2014]
  - Google universal part-of-speech tags [Petrov et al. 2012]
  - Interset morphological features [Zeman 2008]
Universal Dependencies

- **Background:**
  - Treebank annotation schemes vary across languages
  - Hard to compare results across languages [Nivre et al. 2007]
  - Hard to evaluate cross-lingual learning [McDonald et al. 2013]
  - Hard to build multilingual systems

- **Universal Dependencies** (http://universaldependencies.github.io/docs/):
  - Stanford universal dependencies [de Marneffe et al. 2014]
  - Google universal part-of-speech tags [Petrov et al. 2012]
  - Interset morphological features [Zeman 2008]

First guidelines released Oct 1, 2014
Universal Dependencies

- **Background:**
  - Treebank annotation schemes vary across languages
  - Hard to compare results across languages [Nivre et al. 2007]
  - Hard to evaluate cross-lingual learning [McDonald et al. 2013]
  - Hard to build multilingual systems

- **Universal Dependencies** ([http://universaldependencies.github.io/docs/](http://universaldependencies.github.io/docs/)):
  - Stanford universal dependencies [de Marneffe et al. 2014]
  - Google universal part-of-speech tags [Petrov et al. 2012]
  - Interset morphological features [Zeman 2008]

First guidelines released Oct 1, 2014
First 10 treebanks released Jan 15, 2015
Universal Dependencies
Universal Dependencies

▶ Syntactic words – explicit splitting of clitics and contractions
Universal Dependencies

- Syntactic words – explicit splitting of clitics and contractions
- Universal part-of-speech tags + morphological features
Syntactic words – explicit splitting of clitics and contractions

Universal part-of-speech tags + morphological features

Dependency tree + augmented dependencies (not shown)
Guiding Principles

- Maximize parallelism
  - Don’t annotate the same thing in different ways
  - Don’t make different things look the same
- But don’t overdo it
  - Don’t annotate things that are not there
  - Languages select from a universal pool of categories
  - Allow language-specific extensions
Keeping content words as heads promotes parallelism

Function words often correlate with morphology
## Dependency Relations [de Marneffe et al. 2014]

- **Core dependents of clausal predicates**
  - Nominal dep: nsubj, nsubjpass, dobj, iobj
  - Predicate dep: cs subj, cs subjpass, xcomp

- **Non-core dependents of clausal predicates**
  - Nominal dep: nmod
  - Predicate dep: advcl
  - Modifier word: advmod, neg

- **Compounding and unanalyzed**
  - Compound: compound
  - Name: name, foreign
  - Goes with: goeswith

- **Special clausal dependents**
  - Nominal dep: vocative, discourse
  - Auxiliary: aux, auxpass
  - Other: mark, punct

- **Coordination**
  - Conj: conj, cc, punct

- **Case-marking, prepositions, possessive**
  - Case

- **Loose joining relations**
  - List: list, dislocated
  - Parataxis: parataxis
  - Remnant: remnant, reparandum

- **Other**
  - Sentence head: root

- **Unspecified dependency**
  - Dep

- **Taxonomy of 42 universal grammatical relations, broadly supported across many languages in language typology**

- **Language specific subtypes can be added**
Morphology

- Taxonomy of 17 universal part-of-speech tags, based on the Google Universal Tagset [Petrov et al. 2012]
- Standardized inventory of morphological features, based on the Interset system [Zeman 2008]
MWEs in Universal Dependencies

- UD does not allow merged tokens: *in spite of* \(\not\rightarrow\) *in_spite_of*
- MWEs have to be encoded with (dummy or real) dependencies
- Three relations currently used:
  - **mwe**: fixed grammaticized expressions
  - **compound**: lexical compounds of any category
  - **name**: multiword proper names
The **mwe** relation

- **Used for** fixed grammaticized expressions that behave like function words or short adverbials
- **Annotated** in a flat, head-initial structure, where all words in the expression modify the first one using the **mwe** label
The **compound** relation

- Used for lexical compounds, including nominal compounds and particle verbs
- Annotated to reflect headness properties
The name relation

- Used for proper nouns constituted of multiple nominal elements, but not for phrasal or clausal names (The King of Sweden, Gone with the Wind)
- Annotated in a flat, head-initial structure, where all words in the name modify the first one using the name label
Language-Specific Subtypes

- We can define language-specific subtypes of universal relations
- This holds for MWE-type relations as well
- Examples in the first release:
  - `compound:prt` used for verb particles in several languages
  - `nsubj:lvc`, `dobj:lvc` used for LVCs in Hungarian
Want to Build a Better Mousetrap?

- Universal Dependencies is an open and evolving standard
  - Version 1 of the guidelines released Oct 1, 2014
  - Will be kept stable for at least a year
  - We need your help to improve the next version
  - We also need data from more languages
Coming Up Next

1. Introduction to dependency grammar and dependency parsing
2. Graph-based and transition-based dependency parsing
3. Multiword expressions in dependency parsing
4. Practical lab session (MaltParser)
   - Choose a language from the first UD release
   - Train and evaluate a dependency parser
   - Analyze parsing performance with respect to MWEs
References and Further Reading


