Syntactic Dependencies

CS 585, Fall 2015
Introduction to Natural Language Processing
http://people.cs.umass.edu/~brenocon/inlp2015/

Brendan O’Connor
College of Information and Computer Sciences
University of Massachusetts Amherst
• today:
  • syntactic dependencies
  • start on coreference

• Longer distance graphs among words and phrases in a text.
Dependencies
(vs. Constituents)
Disambiguation with lexical information

• (P)CFG structural information doesn’t tell us much about which is more likely
• Lexical knowledge might help? (Or other knowledge?)
  • dessert -> with -> fork
  • ate -> with -> fork
• Intuitively: a notion of *modification* or *argument structure*. 
Constits -> Deps

• Syntactic theory: Every phrase has a head word. It carries the primary syntactic (semantic?) properties of the phrase.
• Head rules: for every nonterminal in tree, choose one of its children to be its “head”.
• Very simple example:

```
• NP -> Adj NP*
• NP -> NP* PP
• PP -> Prep* NP
```
Head rules

\[ S \rightarrow NP\ VP^* \]
\[ VP \rightarrow V^*\ NP \]
\[ NP \rightarrow \text{Det}\ NP^* \]

Rules more complicated for nonbinary expansions, allowing multiple non-heads, e.g.
\[ VP \rightarrow V^*\ PP\ PP \]
Head rules can be used to add words into PCFG nonterminals (“lexicalized PCFGs”)
- Helps a lot for attachment disambiguation
  eat-with-fork vs dessert-with-fork
- Or -- why not use dependency graph directly?
  - Grammatical relations are between individual words
  - Graph is acyclic, connected, with a single root.

Intuitions:
- Syntactic structure is complete (Connectedness).
- Syntactic structure is hierarchical (Acyclicity).
- Every word has at most one syntactic head (Single-Head).

Connectedness can be enforced by adding a special root node.

ROOT
- Economic news had little effect on financial markets.
  adj noun verb adj noun prep adj noun

Recent Advances in Dependency Parsing
Head rules

\[
S \rightarrow NP \ VP^*
\]

\[
VP \rightarrow V^* \ NP
\]

\[
NP \rightarrow \text{Det} \ NP^*
\]

Graph conversion: (12.7.1):
the head of each non-head-child is subordinate to the head of the head-child.
Two ways to parse to dependencies:

- Run a constit parser, then run a (typically rule-based) constit-\(\rightarrow\)deps converter
- Direct dependency parsing

Dependencies useful for many applications

Dependency annotations are available for more languages ... perhaps better suited for a wider variety of languages (e.g. free word order)
Bell, based in Los Angeles, makes and distributes electronic, computer and building products.

- Edges between core words
- DAG (sometimes tree). Options to expand coordination, etc.

**nsubj**: nominal subject

**dobj**: direct object

**prep_X**: prepositional argument

**amod**: adjective modifier

**...**

Using the graph: word-relation-word edges, paths, subgraphs...
Dependency paths

- Information extraction with long(er)-distance connections. Skip over modifiers and subclauses.

officials ←nsubj← meet ←prep→ with ←pobj→ counterparts

British ←amod← (NP) ←nsubj← meet ←prep→ with ←pobj→ (NP) ←pobj→ Iranian
Dependency paths

- Information extraction with long(er)-distance connections. Skip over modifiers and subclauses.

Task: get features which describe the “X was born in Y” semantic relation

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Left window</th>
<th>NE1</th>
<th>Middle</th>
<th>NE2</th>
<th>Right window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical</td>
<td>[]</td>
<td>PER</td>
<td>[was/VERB born/VERB in/CLOSED]</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Lexical</td>
<td>[Astronomer]</td>
<td>PER</td>
<td>[was/VERB born/VERB in/CLOSED]</td>
<td>LOC</td>
<td>[., Missouri]</td>
</tr>
<tr>
<td>Lexical</td>
<td>[#PAD#, Astronomer]</td>
<td>PER</td>
<td>[was/VERB born/VERB in/CLOSED]</td>
<td>LOC</td>
<td>[., Missouri]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Edwin Hubble \psi_{lex-mod}]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Astronomer \psi_{lex-mod}]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[\psi_{lex-mod},]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Edwin Hubble \psi_{lex-mod}]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[\psi_{lex-mod},]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Astronomer \psi_{lex-mod}]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[\psi_{lex-mod},]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[\psi_{inside Missouri}]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Edwin Hubble \psi_{lex-mod}]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[\psi_{inside Missouri}]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Astronomer \psi_{lex-mod}]</td>
<td>PER</td>
<td>[↑s was ↓pred born ↓mod in ↓pcomp-n]</td>
<td>LOC</td>
<td>[\psi_{inside Missouri}]</td>
</tr>
</tbody>
</table>

Table 3: Features for ‘Astronomer Edwin Hubble was born in Marshfield, Missouri’.
Dependency paths

- Rule-based semantic relation extraction

Luke watches as Vader kills Kenobi

Luke runs away

The soldiers shoot at him
Dependency paths

- Rule-based semantic relation extraction

- e.g. assume a verb’s subjects and objects denote arguments in an event

Luke watches as Vader kills Kenobi

Luke runs away

The soldiers shoot at him
Dependency paths

- Rule-based semantic relation extraction
- e.g. assume a verb’s subjects and objects denote arguments in an event
- But gets complicated (syntax-semantics interface)

Luke watches as Vader kills Kenobi

Luke runs away

The soldiers shoot at him
Dependency paths

- Rule-based semantic relation extraction

- e.g. assume a verb’s subjects and objects denote arguments in an event

- But gets complicated (syntax-semantics interface)

- “the Death Star’s destruction”
Should you use a parser in your project?

• Dependency n-grams as features
  • e.g. dep bigrams (word, REL, word)

• Parsers performance and efficiency varies
  • “Shift-reduce” or “incremental” dependency parsers: tend to be fastest, currently
  • Performance: is your data similar to newswire text? (The usual training data)