Syntactic Dependencies (I)

CS 690N, Spring 2017
Advanced Natural Language Processing
http://people.cs.umass.edu/~brenocon/anlp2017/

Brendan O’Connor
College of Information and Computer Sciences
University of Massachusetts Amherst
Dependency parsing in action

Dependency parsing is used in many real-world applications, like question answering (Cui et al, 2005):
Dependency parsing in action

Dependency parsing is used in many real-world applications, like question answering (Cui et al, 2005):

What % of the nation’s cheese does Wisconsin produce?
Dependency parsing is used in many real-world applications, like question answering (Cui et al, 2005):

What % of the nation’s cheese does Wisconsin produce?

In Wisconsin, where farmers produce 28 % of the nation’s cheese, ...
Dependency parsing in action

Question answering works by searching for statements which match well against the query.

- In the surface form of the question, *produce* and % are six words apart.
- But in the dependency parse, they’re adjacent.

What % of the nation’s cheese does Wisconsin produce?

[Example: Jacob Eisenstein]
Projectivity

In **projective** dependency parsing, there are no crossing edges.

- Crossing edges are rare in English:

```
She ate a pizza yesterday which was vegetarian
```

\(^2\)figure from (Nivre 2007)
Projectivity

In **projective** dependency parsing, there are no crossing edges.

- Crossing edges are rare in English:
  
  She ate a pizza yesterday which was vegetarian

- They are more common in other languages, like Czech:

  \(^2\)figure from (Nivre 2007)

\[^2\text{figure from (Nivre 2007)}\]
Constits -> Deps

- Every phrase has a head word. It dominates all other words of that phrase in the dep. graph.
- Head rules: for every nonterminal in tree, choose one of its children to be its “head”. This will define head words.
- Every nonterminal type has a different head rule; e.g. from Collins (1997):

  - If parent is NP,
  - Search from right-to-left for first child that’s NN, NNP, NNPS, NNS, NX, JJR
  - Else: search left-to-right for first child which is NP
Adding Headwords to Trees

S
  NP
    DT the
    NN lawyer
  VP
    Vt questioned
    NP
      DT the
      NN witness

↓

S(questioned)
  NP(lawyer)
    DT the
    NN lawyer
  VP(questioned)
    Vt questioned
    NP(witness)
      DT the
      NN witness
• Dependencies tend to be less specific than constituent structure

(a) Flat

VP

V
ate

NP
dinner

PP
on the table

PP
with a fork

(b) Two-level (PTB-style)

VP

V
ate

NP
dinner

PP
on the table

PP
with a fork

(c) Chomsky adjunction

VP

VP

V
ate

NP
dinner

PP
on the table

PP
with a fork

(d) Dependency representation

ate
dinner
on the table
with a fork

[Example: Jacob Eisenstein]
Projectivity

- Projectivity: no crossing arcs. Corresponds to neatly nested constituencies
- Non-projective example:

She ate a pizza yesterday which was vegetarian

<table>
<thead>
<tr>
<th>Language</th>
<th>% non-projective edges</th>
<th>% non-projective sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>1.86%</td>
<td>22.42%</td>
</tr>
<tr>
<td>English</td>
<td>0.39%</td>
<td>7.63%</td>
</tr>
<tr>
<td>German</td>
<td>2.33%</td>
<td>28.19%</td>
</tr>
</tbody>
</table>

Table 12.1: Frequency of non-projective dependencies in three languages (Kuhlmann and Nivre, 2010)
Parsing to dependencies

• Constituents -> Dependency conversion is one approach

• Direct dependency parsing more common
  • Annotating dependencies is easier

• Algorithmic approaches
  • Graph-based: global CRF-style models
  • History-based: shift-reduce (Nivre)
Graph-based parsing

Edge scoring models

Inference: dynamic programming, (argmax) minimum spanning trees, (expectations) matrix tree theorem

Learning: structured perceptron/svm or crf loglik

[Slides: McDonald and Nivre, EACL 2014 tutorial]
Graph-based parsing

Higher order features: learn e.g. selectional restrictions
Decoding is more difficult

Inference: integer linear programs, gibbs sampling, easy-first ...
Learning: structured perceptron/svm or crf loglik
**Arc-Eager Transition System** [Nivre 2003]

**Configuration:** \((S, B, A)\) \(\text{[}S = \text{Stack, } B = \text{Buffer, } A = \text{Arcs}\text{]}\)

**Initial:** \(([[], [0, 1, \ldots, n]], \{\})\)

**Terminal:** \((S, [], A)\)

**Shift:** \((S, i|B, A) \Rightarrow (S|i, B, A)\)

**Reduce:** \((S|i, B, A) \Rightarrow (S, B, A)\)

**Right-Arc\((k)\):** \((S|i, j|B, A) \Rightarrow (S|i|j, B, A \cup \{(i, j, k)\})\)

**Left-Arc\((k)\):** \((S|i, j|B, A) \Rightarrow (S, j|B, A \cup \{(j, i, k)\})\) \(\neg h(i, A) \wedge i \neq 0\)

**Notation:**
- \(S|i\) = stack with top \(i\) and remainder \(S\)
- \(j|B\) = buffer with head \(j\) and remainder \(B\)
- \(h(i, A) = i\) has a head in \(A\)
Example Transition Sequence

\[ \text{ROOT}_S \quad \text{Economic, news, had, little, effect, on, financial, markets, .} \]

\[ \text{ROOT} \quad \text{Economic} \quad \text{news} \quad \text{had} \quad \text{little} \quad \text{effect} \quad \text{on} \quad \text{financial} \quad \text{markets} \quad . \]

adj  noun  verb  adj  noun  prep  adj  noun  .
Example Transition Sequence

\[ \text{ROOT, Economic}_S \quad \text{[news, had, little, effect, on, financial, markets, .]}_B \]

ROOT Economic news had little effect on financial markets .

adj noun verb adj noun prep adj noun .
Example Transition Sequence

\[ \text{ROOT}_{S} \quad \text{[news, had, little, effect, on, financial, markets, .]}_{B} \]

ROOT Economic news had little effect on financial markets .

adj noun verb adj noun prep adj noun .
Example Transition Sequence

\[ [\text{ROOT, news} ]_S \ [\text{had, little, effect, on, financial, markets, .} ]_B \]
Example Transition Sequence

\[ \text{ROOT}_S \quad \text{had, little, effect, on, financial, markets, .}_B \]

ROOT \quad \text{Economic} \quad \text{news} \quad \text{had} \quad \text{little} \quad \text{effect} \quad \text{on} \quad \text{financial} \quad \text{markets} \quad .

\text{amod} \quad \text{nsubj}
Example Transition Sequence

\[ [\text{ROOT, had}]_S \ [\text{little, effect, on, financial, markets, .}]_B \]
Example Transition Sequence

\[ [\text{ROOT, had, little}]_S \quad [\text{effect, on, financial, markets, .}]_B \]
Example Transition Sequence

\[ [\text{ROOT, had}]_S \quad [\text{effect, on, financial, markets, .}]_B \]
Example Transition Sequence

\[
\text{[ROOT, had, effect]}_S \quad \text{[on, financial, markets, .]}_B
\]
Example Transition Sequence

\[\text{[ROOT, had, effect, on, financial]_S} \quad \text{[markets, .]_B}\]
Example Transition Sequence

[ROOT, had, effect, on]ₜ [markets, .]ₜₜ

Recent Advances in Dependency Parsing

[Slides: McDonald and Nivre, EACL 2014 tutorial]
Example Transition Sequence

[ROOT, had, effect, on, markets]$_S$ [.]$_B$

**Diagram**

```
  root
  /   \\   \\
/       \   \
ROOT amod Economic adj  news noun had verb little adj effect noun on prep financial adj markets .
      |       |        |        |          |        |        |          |
      nsubj \       \
            dobj amod
                |  |    \\   \\
            prep  pmod amod
                |  |        |          |
            pmod
```

**Example Sentence**

Economic news had little effect on financial markets.
Example Transition Sequence
Example Transition Sequence

\[ [\text{ROOT, had, effect}]_S \quad [.]_B \]

- **root**
  - **amod**
  - **nsubj**
- **ROOT**
- **Economic**
- **adj**
- **news**
- **noun**
- **had**
- **verb**
- **little**
- **adj**
- **effect**
- **noun**
- **on**
- **prep**
- **financial**
- **adj**
- **markets**
- **.**
- **pmod**
  - **amod**
Example Transition Sequence

\[\text{ROOT}, \text{had}]_S \ [.]_B

Recent Advances in Dependency Parsing

[Slides: McDonald and Nivre, EACL 2014 tutorial]
Example Transition Sequence

\[ [\text{ROOT, had, .}]_S \quad [\quad]_B \]

Recent Advances in Dependency Parsing

[Slides: McDonald and Nivre, EACL 2014 tutorial]
Greedy Inference

- Given an oracle $o$ that correctly predicts the next transition $o(c)$, parsing is deterministic:

\[
\text{Parse}(w_1, \ldots, w_n)
\]

1. $c \leftarrow ([], [0, 1, \ldots, n]_B, \{\})$
2. while $B_c \neq []$
3. $t \leftarrow o(c)$
4. $c \leftarrow t(c)$
5. return $G = ([0, 1, \ldots, n], A_c)$

- Complexity given by upper bound on number of transitions
- Parsing in $O(n)$ time for the arc-eager transition system
From Oracles to Classifiers

- An oracle can be approximated by a (linear) classifier:
  \[ o(c) = \underset{t}{\text{argmax}} \ w \cdot f(c, t) \]
- History-based feature representation \( f(c, t) \)
- Weight vector \( w \) learned from treebank data
Feature Representation

- Features over input tokens relative to $S$ and $B$

**Configuration**

```
(ROOT, had, effect)S  [on, financial, markets, .]B
```

**Features**

- $\text{word}(S_2) = \text{ROOT}$
- $\text{word}(S_1) = \text{had}$
- $\text{word}(S_0) = \text{effect}$
- $\text{word}(B_0) = \text{on}$
- $\text{word}(B_1) = \text{financial}$
- $\text{word}(B_2) = \text{markets}$
Feature Representation

- Features over input tokens relative to $S$ and $B$
- Features over the (partial) dependency graph defined by $A$

**Configuration**

**Features**

- $\text{dep}(S_1) = \text{root}$
- $\text{dep}(\text{lc}(S_1)) = \text{nsubj}$
- $\text{dep}(\text{rc}(S_1)) = \text{dobj}$
- $\text{dep}(S_0) = \text{dobj}$
- $\text{dep}(\text{lc}(S_0)) = \text{amod}$
- $\text{dep}(\text{rc}(S_0)) = \text{NIL}$
Feature Representation

- Features over input tokens relative to $S$ and $B$
- Features over the (partial) dependency graph defined by $A$
- Features over the (partial) transition sequence

Configuration

```
[ROOT, had, effect]_S  [on, financial, markets, ]_B
```

Features

$t_{i-1} = \text{Right-Arc(dobj)}$

$t_{i-2} = \text{Left-Arc(amod)}$

$t_{i-3} = \text{Shift}$

$t_{i-4} = \text{Right-Arc(root)}$

$t_{i-5} = \text{Left-Arc(nsubj)}$

$t_{i-6} = \text{Shift}$
Feature Representation

- Features over input tokens relative to $S$ and $B$
- Features over the (partial) dependency graph defined by $A$
- Features over the (partial) transition sequence

Configuration

$$\text{ROOT, had, effect}_S \quad \text{on, financial, markets, .}_B$$

Features

$$t_{i-1} = \text{Right-Arc(dobj)}$$
$$t_{i-2} = \text{Left-Arc(AMod)}$$
$$t_{i-3} = \text{Shift}$$
$$t_{i-4} = \text{Right-Arc(root)}$$
$$t_{i-5} = \text{Left-Arc(nsubj)}$$
$$t_{i-6} = \text{Shift}$$

- Feature representation unconstrained by parsing algorithm
Local Learning

- Given a treebank:
  - Reconstruct oracle transition sequence for each sentence
  - Construct training data set $D = \{(c, t) \mid o(c) = t\}$
  - Maximize accuracy of local predictions $o(c) = t$
- Any (unstructured) classifier will do (SVMs are popular)
- Training is local and restricted to oracle configurations
Linear vs neural features

- Non-stateful approaches
  - Nivre (~2003 & others), “MALT”: linear SVM to make shift-reduce decisions, trained on oracle decisions
  - Chen and Manning (2014): neural softmax, trained on oracle decisions
  - Andors et al. (2016), “SyntaxNet”: similar but with global normalization (CRF-ish)
- Stateful: Stack LSTM over state transitions (Dyer et al., like last week)
### Greedy, Local, Transition-Based Parsing

- **Advantages:**
  - Highly efficient parsing – linear time complexity with constant time oracles and transitions
  - Rich history-based feature representations – no rigid constraints from inference algorithm

- **Drawback:**
  - Sensitive to search errors and error propagation due to greedy inference and local learning

- The major question in transition-based parsing has been how to **improve learning and inference**, while maintaining high efficiency and rich feature models
 Beam Search

- Maintain the $k$ best hypotheses [Johansson and Nugues 2006]:

$$
\text{Parse}(w_1, \ldots, w_n)
$$

1. Beam $\leftarrow \{(S, [0, 1, \ldots, n])_B, \{ \} \}$
2. while $\exists c \in \text{Beam} \ [B_c \neq []]$
3. foreach $c \in \text{Beam}$
4. foreach $t$
5. Add($t(c)$, NewBeam)
6. Beam $\leftarrow \text{Top}(k, \text{NewBeam})$
7. return $G = ([0, 1, \ldots, n], A_{\text{Top}(1, \text{Beam})})$

- Note:
  - $\text{Score}(c_0, \ldots, c_m) = \sum_{i=1}^{m} w \cdot f(c_{i-1}, t_i)$
  - Simple combination of locally normalized classifier scores
  - Marginal gains in accuracy

(Beam search can even hurt, since training only on oracle paths)
State of the art

- Unlabeled attachment scores: Accuracy of choose-the-parent
- As of 2014, on old CoNLL 2006 data (variable quality)

<table>
<thead>
<tr>
<th>Language</th>
<th>Best Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>81.12 (MS11)</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>94.02 (ZH13)</td>
</tr>
<tr>
<td>Chinese</td>
<td>92.68 (LX14)</td>
</tr>
<tr>
<td>Czech</td>
<td>91.04 (ZL14)</td>
</tr>
<tr>
<td>Danish</td>
<td>92.00 (ZH13)</td>
</tr>
<tr>
<td>Dutch</td>
<td>86.47 (ZL14)</td>
</tr>
<tr>
<td>English</td>
<td>93.22 (MA13)</td>
</tr>
<tr>
<td>German</td>
<td>92.41 (MA13)</td>
</tr>
<tr>
<td>Japanese</td>
<td>93.74 (LX14)</td>
</tr>
<tr>
<td>Portuguese</td>
<td>93.03 (KR10)</td>
</tr>
<tr>
<td>Slovene</td>
<td>86.95 (MS11)</td>
</tr>
<tr>
<td>Spanish</td>
<td>88.24 (ZL14)</td>
</tr>
<tr>
<td>Swedish</td>
<td>91.62 (ZH13)</td>
</tr>
<tr>
<td>Turkish</td>
<td>77.55 (KR10)</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>89.58</strong></td>
</tr>
</tbody>
</table>

Results table from Zhang et al.

- Labeled attachment scores
- On newer “Universal Dependencies” data (higher quality??) with stack LSTM shift-reduce model

<table>
<thead>
<tr>
<th>Language</th>
<th>de</th>
<th>en</th>
<th>es</th>
<th>fr</th>
<th>it</th>
<th>pt</th>
<th>sv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bulgarian</td>
<td></td>
<td>79.3</td>
<td>85.9</td>
<td>83.7</td>
<td>81.7</td>
<td>88.7</td>
<td>85.7</td>
</tr>
<tr>
<td>Chinese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Czech</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Danish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dutch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japanese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portuguese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slovene</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spanish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swedish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turkish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>89.58</td>
<td>83.7</td>
<td>81.7</td>
<td>88.7</td>
<td>85.7</td>
<td>83.5</td>
<td></td>
</tr>
</tbody>
</table>

Results from Ammar et al.

- CoNLL 2017 shared task right now...
  http://universaldependencies.org/conll17/