Natural Language Processing

Parsing IV

Dan Klein – UC Berkeley
Other Syntactic Models
Dependency Parsing

- Lexicalized parsers can be seen as producing *dependency trees*

  ![Dependency Tree Example](image)

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

- Each local binary tree corresponds to an attachment in the dependency graph
Dependency Parsing

- Pure dependency parsing is only cubic [Eisner 99]

- Some work on non-projective dependencies
  - Common in, e.g. Czech parsing
  - Can do with MST algorithms [McDonald and Pereira 05]

```
root John saw a dog yesterday which was a Yorkshire Terrier
```
Shift-Reduce Parsers

- Another way to derive a tree:

- Parsing
  - No useful dynamic programming search
  - Can still use beam search [Ratnaparkhi 97]
Tree Insertion Grammars

- Rewrite large (possibly lexicalized) subtrees in a single step

```
S
  /   \             /   \      /
 NP    Aux       VP     PP
   \          /           \    
    \        /             \   
     \      /               \  
      \  /                  \ 
       V                   
       |                   |
       |                    |
       NP                  NP
       /                   /     
       hold out          as incentives
       |                   |
       |                   |
       NP                NP
       /                 /     
       Conj             service concessions
       |                 |
       |                 |
       NP              NP
       /               /     
       discounts     and
       |             |
       |             |
       NP          NP
       /       /     
       as incentives
```

- Formally, a tree-insertion grammar
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable parse is NP-complete
TIG: Insertion

\( \phi \)

\( \psi \)

\( \phi' \)

\( \psi' \)

S
  NP \( \downarrow \)
   V
     NP \( \downarrow \)
       saw

S
  NP
     D \( \downarrow \)
       N
         man

S
  NP
     D \( \downarrow \)
       N
         man
     V
       NP \( \downarrow \)
         saw
Tree-adjoining grammars

- Start with *local trees*
- Can insert structure with *adjunction* operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- ... as well as other weird stuff that CFGs don’t capture well (e.g. cross-serial dependencies)
TAG: Long Distance

```
S
  V  S
   |   |
   does  NP  VP
         |     |
         Bill  V  S* 
          |     |
          think

S
  NP(wh)_i  S
   |       |
   who    NP  VP
          |     |
          Harry  V  NP_i
               |     |
               likes  ε
```
CCG Parsing

- **Combinatory Categorial Grammar**
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus (more later)
  - Can have spurious ambiguities (why?)

\[
\begin{align*}
  John & \vdash NP \\
  shares & \vdash NP \\
  buys & \vdash (S\backslash NP)/NP \\
  sleeps & \vdash S\backslash NP \\
  well & \vdash (S\backslash NP)\backslash(S\backslash NP)
\end{align*}
\]
Empty Elements
Empty Elements

- In the PTB, three kinds of empty elements:
  - Null items (usually complementizers)
  - Dislocation (WH-traces, topicalization, relative clause and heavy NP extraposition)
  - Control (raising, passives, control, shared argumentation)

- Need to reconstruct these (and resolve any indexation)
Example: English
Example: German
# Types of Empties

<table>
<thead>
<tr>
<th>Antecedent</th>
<th>POS</th>
<th>Label</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>NP</td>
<td>*</td>
<td>18,334</td>
<td>NP trace (e.g., <em>Sam was seen</em>)</td>
</tr>
<tr>
<td>NP</td>
<td>*</td>
<td>9,812</td>
<td></td>
<td>NP PRO (e.g., <em>to sleep is nice</em>)</td>
</tr>
<tr>
<td>WHNP</td>
<td>NP</td>
<td><em>T</em></td>
<td>8,620</td>
<td>WH trace (e.g., the woman who you saw <em>T</em>)</td>
</tr>
<tr>
<td></td>
<td><em>U</em></td>
<td>7,478</td>
<td></td>
<td>Empty units (e.g., $25 <em>U</em>)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>5,635</td>
<td></td>
<td>Empty complementizers (e.g., Sam said 0 Sasha snores)</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td><em>T</em></td>
<td>4,063</td>
<td>Moved clauses (e.g., Sam had to go, Sasha explained <em>T</em>)</td>
</tr>
<tr>
<td>WHADVP</td>
<td>ADVP</td>
<td><em>T</em></td>
<td>2,492</td>
<td>WH-trace (e.g., *Sam explained how to leave <em>T</em>)</td>
</tr>
<tr>
<td></td>
<td>SBAR</td>
<td></td>
<td>2,033</td>
<td>Empty clauses (e.g., Sam had to go, Sasha explained (SBAR))</td>
</tr>
<tr>
<td></td>
<td>WHNP</td>
<td>0</td>
<td>1,759</td>
<td>Empty relative pronouns (e.g., the woman 0 we saw)</td>
</tr>
<tr>
<td></td>
<td>WHADVP</td>
<td>0</td>
<td>575</td>
<td>Empty relative pronouns (e.g., no reason 0 to leave)</td>
</tr>
</tbody>
</table>

**Diagram:**

```
NP
  | NP
  | S
  | DT NN WHNP-1 S
  | the man -NONE- NP
  | 0 NNP VBD\_t NP
  | Sam likes -NONE- *T*\_1

SIN
  \_ S-1 , VP NP
  \_ NP VBD SBAR NNP
  \_ NNS VBD said -NONE- S Sam
  \_ changesoccurred 0 -NONE- *T*\_1
```
A Pattern-Matching Approach

- [Johnson 02]

Diagram:

```
NP
  NP  SBAR
    DT  NN  WHNP-1  S
      the  man  -NONE-  NP  VP
                0  NNP  VBZ-t  NP
                  Sam  likes  -NONE-  *T*-1
```

```
SBAR
  WHNP-1  S
    -NONE-  NP  VP
      0  VBZ-t  NP
        -NONE-  *T*-1
```

```
NP
  SBAR
    DT  NN  S
      the  man  NP  VP
         NNP  VBZ-t
           Sam  likes
```
Pattern-Matching Details

- Something like transformation-based learning
- Extract patterns
  - Details: transitive verb marking, auxiliaries
  - Details: legal subtrees
- Rank patterns
  - Pruning ranking: by correct / match rate
  - Application priority: by depth
- Pre-order traversal
- Greedy match
## Top Patterns Extracted

<table>
<thead>
<tr>
<th>Count</th>
<th>Match</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>5816</td>
<td>6223</td>
<td>(S (NP (<del>NONE</del> *) VP)</td>
</tr>
<tr>
<td>5605</td>
<td>7895</td>
<td>(SBAR (<del>NONE</del> 0) S)</td>
</tr>
<tr>
<td>5312</td>
<td>5338</td>
<td>(SBAR WHNP-1 (S (NP (<del>NONE</del> <em>T</em>-1)) VP))</td>
</tr>
<tr>
<td>4434</td>
<td>5217</td>
<td>(NP QP (<del>NONE</del> <em>U</em>))</td>
</tr>
<tr>
<td>1682</td>
<td>1682</td>
<td>(NP $ CD (<del>NONE</del> <em>U</em>))</td>
</tr>
<tr>
<td>1327</td>
<td>1593</td>
<td>(VP VBN_t (NP (<del>NONE</del> *)) PP)</td>
</tr>
<tr>
<td>700</td>
<td>700</td>
<td>(ADJP QP (<del>NONE</del> <em>U</em>))</td>
</tr>
<tr>
<td>662</td>
<td>1219</td>
<td>(SBAR (WHNP-1 (<del>NONE</del> 0)) (S (NP (<del>NONE</del> <em>T</em>-1)) VP))</td>
</tr>
<tr>
<td>618</td>
<td>635</td>
<td>(S S-1, NP (VP VBD (SBAR (<del>NONE</del> 0) (S (<del>NONE</del> <em>T</em>-1)))) .)</td>
</tr>
<tr>
<td>499</td>
<td>512</td>
<td>(SINV &quot; S-1&quot;, (VP VBZ (S (<del>NONE</del> <em>T</em>-1))) NP .)</td>
</tr>
<tr>
<td>361</td>
<td>369</td>
<td>(SINV &quot; S-1&quot;, (VP VBD (S (<del>NONE</del> <em>T</em>-1))) NP .)</td>
</tr>
<tr>
<td>352</td>
<td>320</td>
<td>(S NP-1 (VP VBZ (S (NP (<del>NONE</del> *)) VP)) )</td>
</tr>
<tr>
<td>346</td>
<td>273</td>
<td>(S NP-1 (VP AUX (VP VBN_t (NP (<del>NONE</del> *)) PP)) )</td>
</tr>
<tr>
<td>322</td>
<td>467</td>
<td>(VP VBD_t (NP (<del>NONE</del> *)) PP)</td>
</tr>
<tr>
<td>269</td>
<td>275</td>
<td>(S &quot; S-1&quot;, NP (VP VBD (S (<del>NONE</del> <em>T</em>-1)) .)</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Empty node POS</th>
<th>Label</th>
<th>Section 23</th>
<th>Parser output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$P$</td>
<td>$R$</td>
</tr>
<tr>
<td>(Overall)</td>
<td></td>
<td>0.93</td>
<td>0.83</td>
</tr>
<tr>
<td>NP</td>
<td>*</td>
<td>0.95</td>
<td>0.87</td>
</tr>
<tr>
<td>NP</td>
<td><em>T</em></td>
<td>0.93</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.94</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td><em>U</em></td>
<td>0.92</td>
<td>0.98</td>
</tr>
<tr>
<td>S</td>
<td><em>T</em></td>
<td>0.98</td>
<td>0.83</td>
</tr>
<tr>
<td>ADVP</td>
<td><em>T</em></td>
<td>0.91</td>
<td>0.52</td>
</tr>
<tr>
<td>SBAR</td>
<td></td>
<td>0.90</td>
<td>0.63</td>
</tr>
<tr>
<td>WHNP</td>
<td>0</td>
<td>0.75</td>
<td>0.79</td>
</tr>
</tbody>
</table>
Semantic Roles
Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:

  \[ \text{Judge She } \text{blames} \text{ [Evaluate the Government ] [Reason for failing to do enough to help ] .} \]

  Holman would characterise this as \text{blaming} \text{ [Evaluate the poor ] .}

  The letter quotes Black as saying that \[ \text{Judge white and Navajo ranchers } \text{misrepresent their livestock losses and blame} \text{ [Reason everything ] [Evaluate on coyotes ] .} \]

- Says more than which NP is the subject (but not much more):
- Relations like \text{subject} are syntactic, relations like \text{agent} or \text{message} are semantic
- Typical pipeline:
  - Parse, then label roles
  - Almost all errors locked in by parser
  - Really, SRL is quite a lot easier than parsing
SRL Example

He heard the sound of liquid sloshing in a metal container as Farrell approached him from behind.

[Diagram showing parse tree with tagging and SRL annotations]
FrameNet: roles shared between verbs
PropBank: each verb has its own roles
PropBank more used, because it’s layered over the treebank (and so has greater coverage, plus parses)
Note: some linguistic theories postulate fewer roles than FrameNet (e.g. 5-20 total: agent, patient, instrument, etc.)
PropBank Example

fall.01  
sense: move downward
roles:  
Arg1: thing falling
Arg2: extent, distance fallen
Arg3: start point
Arg4: end point

Sales fell to $251.2 million from $278.7 million.
arg1: Sales
rel: fell
arg4: to $251.2 million
arg3: from $278.7 million
**PropBank Example**

**rotate.02**

- sense: shift from one thing to another
- roles:
  - Arg0: causer of shift
  - Arg1: thing being changed
  - Arg2: old thing
  - Arg3: new thing

Many of Wednesday’s winners were losers yesterday as investors quickly took profits and rotated their buying to other issues, traders said. (wsj_1723)

- arg0: investors
- rel: rotated
- arg1: their buying
- arg3: to other issues
PropBank Example

aim.01  sense: intend, plan
        roles:  Arg0: aimer, planner
               Arg1: plan, intent

        The Central Council of Church Bell Ringers aims *trace* to improve relations with vicars.
        (wsj_0089)
        arg0:  The Central Council of Church Bell Ringers
        rel:  aims
        arg1:  *trace* to improve relations with vicars

aim.02  sense: point (weapon) at
        roles:  Arg0: aimer
               Arg1: weapon, etc.
               Arg2: target

        Banks have been aiming packages at the elderly.
        arg0:  Banks
        rel:  aiming
        arg1:  packages
        arg2:  at the elderly
Shared Arguments

(NP-SBJ (JJ massive) (JJ internal) (NN debt))
(VP (VBZ has))
(VP (VBN forced))
(S
  (NP-SBJ-1 (DT the) (NN government)))
(VP
  (VP (TO to))
  (VP (VB borrow))
  (ADVP-MNR (RB massively))...
Path Features

Path | Description
--- | ---
VB↑VP↑PP | PP argument/adjunct
VB↑VP↑S↑NP | subject
VB↑VP↑NP | object
VB↑VP↑VP↑S↑NP | subject (embedded VP)
VB↑VP↑ADVP | adverbial adjunct
NN↑NP↑NP↑PP | prepositional complement of noun
## Results

- **Features:**
  - Path from target to filler
  - Filler’s syntactic type, headword, case
  - Target’s identity
  - Sentence voice, etc.
  - Lots of other second-order features

- **Gold vs parsed source trees**
  - SRL is fairly easy on gold trees
  - Harder on automatic parses

<table>
<thead>
<tr>
<th></th>
<th>CORE</th>
<th>ArgM</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>92.2</td>
<td>89.9</td>
</tr>
<tr>
<td>Acc.</td>
<td>80.7</td>
<td>71.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CORE</th>
<th>ArgM</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>84.1</td>
<td>81.4</td>
</tr>
<tr>
<td>Acc.</td>
<td>66.5</td>
<td>55.6</td>
</tr>
</tbody>
</table>
Empties and SRL

S

NP
NN Housing
NNS lobbies

VP
VBD persuaded

VP
NP NNP Congress

S

TO to

VP

NP NP raise

PP to $124,875