A simple pattern-matching algorithm for recovering empty nodes

Mark Johnson
Brown University

ACL’02, Philadelphia

Thanks to Eugene Charniak and fellow BLLIPers
NSF grants DMS 0074276 and ITR IIS 0085940
Talk outline

- Empty nodes in the Penn treebank representations
- A pattern-matching algorithm
- Evaluating empty node accuracy
- Evaluation on gold standard and parser trees
Empty nodes and co-indexation indicate non-local dependencies that are important for semantic interpretation.

Likely to be important for question-answering and machine translation.
The output of most modern statistical parsers only encode *local dependencies*

- Collins (1997) discusses recovering WH dependencies
- SUBGs typically encode non-local dependencies
Other previous work on empty nodes

**Generative syntax:** Non-local dependencies are a major theme

- Extremely complex theories
- Focuses on esoteric constructions
- Studies just a few kinds of non-local dependencies

**Psycholinguistics:** has studied interpretation of non-local dependencies

- Preferences for location of empty nodes
- How non-local dependencies affect complexity of sentence processing

- The *pattern-matching approach* described here is:
  - Theory neutral
  - Data-driven: trained from tree-bank*
  - Relatively straight-forward to implement
  - Can serve as a *base-line for more complex systems*
System architecture

Training

Treebank sections 2-21

Extract patterns

Empty node patterns

Parsing

Treebank section 23

Parser (Charniak)

Pattern matcher

Parse trees with LDDs
Empty node insertion via pattern-matching

Pattern

- Patterns extracted from Penn treebank training corpus (sections 2-21)
- Patterns matched against parser output
- A matching pattern suggests a long-distance dependency
## Summary of empty nodes in Penn trees

<table>
<thead>
<tr>
<th>Antecedent</th>
<th>Category</th>
<th>Label</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
</table>
| NP         | NP       | *     | 18,334| NP trace (Passive)  
> Sam was seen * |
| NP         | *        | 9,812 | NP PRO (implied subject)  
> * to sleep is nice |
| WHNP       | NP       | *T*   | 8,620 | WH trace (questions, relative clauses)  
> the woman who you saw *T* |
|            | *U*      | 7,478 | Empty units  
> $25 *U* |
|            | 0        | 5,635 | Empty complementizers  
> Sam said 0 Sasha snores |
### Summary of empty nodes in Penn trees

<table>
<thead>
<tr>
<th>Antecedent</th>
<th>Category</th>
<th>Label</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
</table>
| S          | S        | *T*   | 4,063 | Moved clauses  
  *Sam had to go, Sasha explained *T** |
| WHADVP     | ADVP     | *T*   | 2,492 | WH-trace  
  *Sam explained how to leave *T** |
|            | SBAR     |       | 2,033 | Empty clauses  
  *Sam had to go, Sasha explained (SBAR)* |
| WHNP       | 0        |       | 1,759 | Empty relative pronouns  
  *the woman 0 we saw* |
| WHADVP     | 0        |       | 575   | Empty relative pronouns  
  *no reason 0 to leave* |

- **Zipfian distribution of empty node types**
Two empty nodes in a long-distance dependency

```
NP
   /  
  NP  SBAR
     /    
    DT   NN
   /     /  
  the  man  WHNP-1
       /    
      S     
     /      
    NP  VP
   /    
  0    
   /   
  NNP  VBZ_t
  /    
Sam  likes  -NONE-
      /    
     *T*-1
```
Changes occurred, said Sam.
Extraposition and adjunction

```
S
/  \
NP-13  VP
/  \
NP  SBAR  VBD  VP
/  \
NNS -NONE- were  VBN NP  SBAR-2
/  \
conferences  *ICH*-2  held -NONE- WHNP-1
/  \
0 -NONE- NP  VP
/  \
*T*-1  TO  VP
/  \
to VB PP-CLR IN
/  
chew on
```
Auxiliary POS replacement: The POS of auxiliary verbs *is, being*, etc. are replaced by AUX, AUXG, etc. (Charniak)

Transitivity relabelling: The POS labels of transitive verbs are suffixed “_t”, e.g., *likes* is relabelled VBZ_t

- Transitivity is hypothesised to be a powerful cue to empty node placement
- Experiments on heldout data indicate this improves accuracy
- A verb is deemed transitive if it is followed by an NP with no function tag at least 50% of the time in the training corpus
- Morphological analysis may improve transitivity identification
Patterns and matchings

- A pattern is *the minimal set of local trees* that connects each empty node with the nodes coindexed with it.
- Indices are systematically renumbered*.
- The implementation deals with *adjunction* and overlapping long-distance dependencies.
  - Probably has a negligible effect on performance.
Empty node insertion

- Patterns are matched at each node in the tree
- Approximately 11,000 patterns
  - Pattern matching is speeded by indexing patterns on their topmost local tree
- Nodes in the tree to be matched are visited by a preorder traversal
  - Matching and insertion of deep pattern may destroy the context of a shallow one
  - Biases the algorithm in favor of deeper patterns
Overlapping patterns

The most common pattern  The third most common pattern

- The most common pattern will match every context that the third most common pattern matches (but not vice-versa)
- Preorder node traversal ensures that the third most common pattern gets a chance to match
Pattern extraction and selection

- Every pattern in training corpus is extracted
- For each pattern:
  - $c$: the number of times extracted
  - $m$: the number of times it matches some context in training corpus
    * Difficult to estimate because a larger pattern might destroy the context for a smaller one
  - If discounted success probability $< 1/2$ the pattern is discarded
    * Around 9,000 patterns remain after filtering
  - Patterns are sorted by depth (deep patterns first)
    * Exactly how patterns are sorted (e.g., frequency, discounted success probability) doesn’t seem to matter
## The most common patterns

<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 (-NONE- *)) VP)</td>
</tr>
<tr>
<td>5605</td>
<td>7895</td>
<td>(SBAR (-NONE- 0) S)</td>
</tr>
<tr>
<td>5312</td>
<td>5338</td>
<td>(SBAR WHNP-1 (S (NP (-NONE- <em>T</em>-1)) VP))</td>
</tr>
<tr>
<td>4434</td>
<td>5217</td>
<td>(NP QP (-NONE- <em>U</em>))</td>
</tr>
<tr>
<td>1682</td>
<td>1682</td>
<td>(NP $ CD (-NONE- <em>U</em>))</td>
</tr>
<tr>
<td>1327</td>
<td>1593</td>
<td>(VP VBN_t (NP (-NONE- *)) PP)</td>
</tr>
<tr>
<td>700</td>
<td>700</td>
<td>(ADJP QP (-NONE- <em>U</em>))</td>
</tr>
<tr>
<td>662</td>
<td>1219</td>
<td>(SBAR (WHNP-1 (-NONE- 0)) (S (NP (-NONE- <em>T</em>-1)) VP))</td>
</tr>
<tr>
<td>618</td>
<td>635</td>
<td>(S S-1 , NP (VP VBD (SBAR (-NONE- 0) (S (-NONE- <em>T</em>-1)))))</td>
</tr>
<tr>
<td>499</td>
<td>512</td>
<td>(SINV “ S-1 ,” (VP VBZ (S (-NONE- <em>T</em>-1))) NP .)</td>
</tr>
<tr>
<td>361</td>
<td>369</td>
<td>(SINV “ S-1 ,” (VP VBD (S (-NONE- <em>T</em>-1))) NP .)</td>
</tr>
</tbody>
</table>
Empty node recovery evaluation

- Two different evaluation methods
  - *Standard Parseval evaluation:* evaluates empty node location, but not coindexation
  - *Extended evaluation:* evaluates both empty node location and coindexation
- Evaluate on *test trees without empty nodes* and on *parser output*

**Standard Parseval evaluation:** Nodes identified by a triple $\langle \text{cat, left, right} \rangle$ (note $\text{left} = \text{right}$ for empty nodes)
- $G =$ set of empty nodes identified in gold-standard trees
- $T =$ set of trees produced by parser*

\[ P = \frac{|G \cap T|}{|T|} \quad R = \frac{|G \cap T|}{|G|} \quad f = \frac{2 \cdot P \cdot R}{P + R} \]
## Empty node identification results

<table>
<thead>
<tr>
<th>Empty node Category</th>
<th>Empty node 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>
Evaluation of empty nodes and their antecedents

- Each empty node is identified by a set of triples \( \langle \text{cat}, \text{left}, \text{right} \rangle \) corresponding to
  - the empty node itself
  - each node co-indexed with the empty node

- In order to “get the empty node right”, the category and location of each of its antecedents must be recovered
  - Most empty nodes have zero or one antecedents
  - Stringent requirement, which also evaluates parser accuracy
  - Other measures (e.g., which only require identification of the head of the antecedent) yield very similar results
## Empty node and antecedent identification results

<table>
<thead>
<tr>
<th>Empty node</th>
<th>Antecedant</th>
<th>POS</th>
<th>Label</th>
<th>Section 23</th>
<th>Parser output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Overall)</td>
<td></td>
<td></td>
<td>$P$</td>
<td>$R$</td>
</tr>
<tr>
<td>NP</td>
<td>NP</td>
<td>*</td>
<td></td>
<td>0.80</td>
<td>0.70</td>
</tr>
<tr>
<td>WHNP</td>
<td>NP</td>
<td><em>T</em></td>
<td></td>
<td>0.93</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>NP</td>
<td>*</td>
<td></td>
<td>0.45</td>
<td>0.77</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>0.94</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td><em>U</em></td>
<td></td>
<td></td>
<td>0.92</td>
<td>0.98</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td><em>T</em></td>
<td></td>
<td>0.98</td>
<td>0.83</td>
</tr>
<tr>
<td>WHADVP</td>
<td>ADVP</td>
<td><em>T</em></td>
<td></td>
<td>0.91</td>
<td>0.52</td>
</tr>
<tr>
<td>SBAR</td>
<td>WHNP</td>
<td>0</td>
<td></td>
<td>0.75</td>
<td>0.79</td>
</tr>
</tbody>
</table>
Discussion

- **Empty node identification** can be performed with reasonable accuracy
  - Performance drop-off on parser trees
  - Precision $\gg$ recall $\Rightarrow$ patterns may be too specialized
    * **Skeletal patterns** trade precision for recall, but leave f-score unchanged

- **Antecedent recovery** is considerably harder
  - Only half of the bound NP PRO are recovered!
    * Requires semantic/pragmatic information about interpretation
    * 10 pages of rules/examples about NP PRO indexing in tagging guidelines!
    * **Lexicalized patterns** ought to help, but didn’t
    * More sophisticated classifiers (boosted decision stubs) had very similar performance to simple pattern matcher
  - Many long distance dependencies (e.g., WH-dependencies) can on average be reliably identified
Conclusions and Future Work

- This paper proposed two Parseval-style measures to evaluate empty node identification and antecedent identification
  - Restricted to Penn treebank style representation of long distance dependencies

- A simple pattern-matching post-processing approach to long-distance dependency identification works reasonably well

- Provides a baseline against which to evaluate more sophisticated systems

- Performance drop-off when using parser trees
  - a single system that integrates parsing and long distance dependency identification may perform better