
• slides mostly taken from material prepared by B. Srinivas
Descriptions of Primitives

- Simple: likes/V
- Complex:

```
S
 NP VP
  V NP
  | likes

S
 NP S
  NP VP
  e V NP
  | likes
```

- Complexity of Descriptions
  - Complex constraints operate locally
  - Implications for statistical computations
**Extended Domain of Locality (EDL)**

1. Every elementary structure must contain all and only the arguments of the anchor.

2. There is one elementary structure for each syntactic environment a lexical item may appear in.
Factoring of Recursion

- Recursion is factored away from the domain for the statement of dependencies.
**Lexicalized Tree-Adjoining Grammars**

- Primary objects of LTAGs are Elementary Trees.
- Lexicalized, Extended Domain of Locality, Factoring of Recursion.
- Elementary Trees are of two types
  - Initial Trees and Auxiliary Trees
    
    \[
    S_r
    \]
    
    \[
    NP_{p}\downarrow
    \]
    
    \[
    V
    \]
    
    \[
    hit
    \]
    
    \[
    VP
    \]
    
    \[
    NP_p\downarrow
    \]
    
    \[
    A
    \]
    
    \[
    N_{r}\downarrow
    \]
    
    \[
    N_f^*
    \]
    
    purple

- Substitution and Adjunction are two combining operations.
**Example**

*who does Woody think Andy likes*
Example

who does Woody think Andy likes

- Derived Tree
Example

*who does Woody think Andy likes*

- Derivation Tree

```
α4 [likes]

α1 [who] (1)  β1 [think] (2)  α3 [Andy] (2.1)

β2 [does] (0)  α2 [Woody] (1)
```
Supertags

- Elementary trees are called Supertags.
- Localize head-complement and filler-gap dependencies.

- Supertags
  - more complex than part-of-speech tags
  - more supertags associated with word than part-of-speech tags
the purchase price includes two ancillary companies.
Supertagging

<table>
<thead>
<tr>
<th>Sent:</th>
<th>the purchase price includes two ancillary companies.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Assig.</td>
<td>$\beta_1$</td>
</tr>
<tr>
<td>Final Assig.</td>
<td>$\beta_1$</td>
</tr>
</tbody>
</table>

- Supertagging: Select most appropriate supertag for each word.
- Supertag disambiguation before parsing.
- Supertag disambiguation results in an “almost parse”. 
Models for Supertag Disambiguation

• N-gram models
  – Trigram model
  – Head trigram model

• Dependency based model (COLING 94)
  – More like full parsing
Trigram Model for Supertagging

- Find the most likely Supertag sequence for a given word sequence.
  \[ \hat{T} = \arg\max_T \Pr(T_1, T_2, \ldots, T_N | W_1, W_2, \ldots, W_N) \]

- By Bayes Rule
  \[ \hat{T} = \arg\max_T \frac{\Pr(W_1, W_2, \ldots, W_N | T_1, T_2, \ldots, T_N) \times \Pr(T_1, T_2, \ldots, T_N)}{\Pr(W_1, W_2, \ldots, W_N)} \]

- Since the word sequence is given
  \[ \hat{T} = \arg\max_T \Pr(W_1, W_2, \ldots, W_N | T_1, T_2, \ldots, T_N) \times \Pr(T_1, T_2, \ldots, T_N) \]
Trigram Model for Supertagging

- Contextual probability

\[
\Pr(T_1, T_2, \ldots, T_N) \approx \prod_{i=1}^N \Pr(T_i \mid T_{i-2}, T_{i-1})
\]

- Word Emit probability

\[
\Pr(W_1, W_2, \ldots, W_N \mid T_1, T_2, \ldots, T_N) \approx \prod_{i=1}^N \Pr(W_i \mid T_i)
\]

- Trigram Model

\[
\hat{T} = \arg\max_T \prod_{i=1}^N \Pr(T_i \mid T_{i-2}, T_{i-1}) \ast \Pr(W_i \mid T_i)
\]

where \(T_i\) is the supertag for word \(W_i\).

- Unseen events

  - Good-Turing discounting with Katz’s Back-off Model.
Training and Test Data

• Training Set A:
  – 200,000 word-supertag pairs
  – collected by bootstrapping and hand correction.
  – WSJ sections 15 through 18

• Training Set B:
  – 1,000,000 word-supertag pairs
  – collected by heuristically mapping from Penn Treebank
  – WSJ sections 0-19 and 21-24

• Test Set: section 20 of WSJ.
Performance of Trigram Supertagger

- Performance of the supertagger on the WSJ corpus

- Correct supertag implies that a word is assigned the same supertag as it would be in the correct parse of the sentence.

<table>
<thead>
<tr>
<th>Size of training corpus</th>
<th>Size of test corpus</th>
<th># of words correctly supertagged</th>
<th>% correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>47,000</td>
<td>35,391</td>
<td>75.3%</td>
</tr>
<tr>
<td>200,000</td>
<td>47,000</td>
<td>42,723</td>
<td>90.9%</td>
</tr>
<tr>
<td>1 Million</td>
<td>47,000</td>
<td>43,334</td>
<td>92.2%</td>
</tr>
</tbody>
</table>

- Errors:
  - PP attachment
  - Verbs with more than two complements.
**Head Trigram Model for Supertagging**

- **Head Trigram Model**

\[
\hat{T} = \arg\max_T \prod_{i=1}^{N} \Pr(T_i \mid T_{H_{i-2}}, T_{H_{i-1}}) \times \Pr(W_i \mid T_i)
\]

- ...saw the big man with ...

  Trigram Model computes: \(\Pr(\text{with} \mid T) \times \Pr(T \mid T_{\text{man}}, T_{\text{big}})\)

  Head Trigram Model computes: \(\Pr(\text{with} \mid T) \times \Pr(T \mid T_{\text{man}}, T_{\text{saw}})\)

- **Head identification**

- **Head Propagation**

\[(1)\]

  Initialize: \((H_{-2}, H_{-1}) = (-2, -1)\)

  Update: \((H_{i-1}, H_i) = (H_{i-2}, H_{i-1})\) if \(W_i\) is not a head word

  \(= (H_{i-1}, i)\) if \(W_i\) is a head word
Head Trigram Model for Supertagging

- Head-word tagger: Identify the head words given a sentence

<table>
<thead>
<tr>
<th></th>
<th>Size of training corpus</th>
<th>Size of test corpus</th>
<th># of words correctly supertagged</th>
<th>% correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>47,000</td>
<td></td>
<td>38,258</td>
<td>81.4%</td>
</tr>
<tr>
<td>1 Million</td>
<td>47,000</td>
<td></td>
<td>42,864</td>
<td>91.2%</td>
</tr>
</tbody>
</table>

- Performance of the head trigram supertagger:

<table>
<thead>
<tr>
<th></th>
<th>Size of training corpus</th>
<th>Size of test corpus</th>
<th># of words correctly supertagged</th>
<th>% correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>47,000</td>
<td></td>
<td>35,391</td>
<td>75.3%</td>
</tr>
<tr>
<td>1 Million</td>
<td>47,000</td>
<td></td>
<td>40,890</td>
<td>87%</td>
</tr>
</tbody>
</table>
Chunking using Supertagged output

• Noun chunks
  – Non-recursive noun phrases
• Scan right to left starting with the noun initial supertag and collect all functors of a noun or a noun modifier.
• Examples:
  – New Jersey Turnpike Authority
  – its increasingly rebellious citizens
  – two $ 400 million real estate mortgage investment conduits
Chunking using Supertagged output


<table>
<thead>
<tr>
<th>System</th>
<th>Training Size</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>R&amp;M</td>
<td>Baseline</td>
<td>81.9%</td>
<td>78.2%</td>
</tr>
<tr>
<td>R&amp;M</td>
<td>200K</td>
<td>92.3%</td>
<td>91.8%</td>
</tr>
<tr>
<td>Supertags</td>
<td>Baseline</td>
<td>74.0%</td>
<td>58.4%</td>
</tr>
<tr>
<td>Supertags</td>
<td>200K</td>
<td>93.0%</td>
<td>91.8%</td>
</tr>
<tr>
<td>Supertags</td>
<td>1000K</td>
<td>93.8%</td>
<td>92.5%</td>
</tr>
</tbody>
</table>

- Internal structure of the noun phrases.
Chunking using Supertagged output

• Verbs chunks
  – Sequence of verbs or verbal modifiers.
• Scan left to right starting with the verb or verbal modifier supertag and collect all functors of a verb or a verb modifier.
• Examples
  – would not have been stymied
  – did n’t even care
  – just beginning to collect
Chunking using Supertagged output


<table>
<thead>
<tr>
<th>System</th>
<th>Training Size</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>R&amp;M</td>
<td>Baseline</td>
<td>60.0%</td>
<td>47.8%</td>
</tr>
<tr>
<td>R&amp;M</td>
<td>200K</td>
<td>88.5%</td>
<td>87.7%</td>
</tr>
<tr>
<td>Supertags</td>
<td>Baseline</td>
<td>76.3%</td>
<td>67.9%</td>
</tr>
<tr>
<td>Supertags</td>
<td>200K</td>
<td>86.5%</td>
<td>91.4%</td>
</tr>
</tbody>
</table>

• Differences in verb groups
  – (has involved simply buying) (and then holding)
  – predicatives

• Internal structure of the Verb phrases.
  – Sentential complement information.
Lightweight Dependency Analyzer

- Information associated with supertags:
  - Slots: substitution and foot nodes
- Fillers of substitution nodes are argument words and fillers of foot nodes are modified words.
- Two pass algorithm:
  - Establish dependencies for auxiliary supertags
  - Mark all the words that serve as arguments as unavailable for the next pass
  - Establish dependencies for initial supertags.
- Establish dependencies – local search
  - first supertag with root node same as the argument type.
Lightweight Dependency Analyzer

The implicit interior state of the iteration over the hash table entries has dynamic extent

<table>
<thead>
<tr>
<th>Pos</th>
<th>Word</th>
<th>Supertag</th>
<th>Slot req.</th>
<th>Pass 1</th>
<th>Pass 2</th>
<th>Dep Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The</td>
<td>α₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>implicit</td>
<td>β₂</td>
<td>+N*</td>
<td>2*</td>
<td>2*</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>interior</td>
<td>β₂</td>
<td>+N*</td>
<td>3*</td>
<td>3*</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>state</td>
<td>α₂</td>
<td>-D.</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>of</td>
<td>β₁</td>
<td>-NP* +NP.</td>
<td>3* 6.</td>
<td>3* 6.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>the</td>
<td>α₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>iteration</td>
<td>α₂</td>
<td>-D.</td>
<td>5.</td>
<td>5.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>over</td>
<td>β₁</td>
<td>-NP* +NP.</td>
<td>6* 11.</td>
<td>6* 11.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>the</td>
<td>α₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>hash</td>
<td>β₃</td>
<td>+N*</td>
<td>10*</td>
<td>10*</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>table</td>
<td>β₃</td>
<td>+N*</td>
<td>11*</td>
<td>11*</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>entries</td>
<td>α₂</td>
<td>-D.</td>
<td>8.</td>
<td>8.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>has</td>
<td>α₃</td>
<td>+NP. -NP.</td>
<td>3. 14.</td>
<td>3. 14.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>dynamic</td>
<td>β₂</td>
<td>+N*</td>
<td>14*</td>
<td>14*</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>extent</td>
<td>α₄</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lightweight Dependency Analyzer

• Trigram supertagger trained on one million supertagged WSJ words.

• Performance on pairwise dependency links
  – A link in output must be in gold standard

<table>
<thead>
<tr>
<th>Corpus</th>
<th># of dependency links</th>
<th># produced by LDA</th>
<th># correct</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>140,280</td>
<td>126,493</td>
<td>112,420</td>
<td>80.1%</td>
<td>88.8%</td>
</tr>
<tr>
<td>WSJ</td>
<td>47,333</td>
<td>41,009</td>
<td>38,480</td>
<td>82.3%</td>
<td>93.8%</td>
</tr>
</tbody>
</table>
Lightweight Dependency Analyzer

- Test corpus was parsed using the XTAG system
- Performance on pairwise dependency links

<table>
<thead>
<tr>
<th>Training Size (words)</th>
<th>Test Size (words)</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>200,000</td>
<td>12,000</td>
<td>83.6%</td>
<td>83.5%</td>
</tr>
<tr>
<td>1,000,000</td>
<td>12,000</td>
<td>85.0%</td>
<td>85.0%</td>
</tr>
</tbody>
</table>

- Performance at the sentence level
  (Matching against XTAG derivation trees)

<table>
<thead>
<tr>
<th></th>
<th>% sentences with 0 errors</th>
<th>% sentences with ≤1 error</th>
<th>% sentences with ≤2 errors</th>
<th>% sentences with ≤3 errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>200K</td>
<td>35%</td>
<td>60.3%</td>
<td>78%</td>
<td>89.8%</td>
</tr>
<tr>
<td>1M</td>
<td>40%</td>
<td>63.0%</td>
<td>80.1%</td>
<td>91.0%</td>
</tr>
</tbody>
</table>
Dependency Based Model

• Data Representation

<table>
<thead>
<tr>
<th>(P.O.S, Supertag)</th>
<th>Direction of Dependent Supertag</th>
<th>Ordinal position</th>
<th>Dependent Supertag</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D, $\alpha_1$)</td>
<td>()</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(N, $\alpha_{13}$)</td>
<td>()</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(N, $\alpha_2$)</td>
<td>($-$)</td>
<td>$-1$</td>
<td>$\alpha_1$</td>
<td>0.975</td>
</tr>
<tr>
<td>(V, $\alpha_{15}$)</td>
<td>($-,$ $+$)</td>
<td>$-1$</td>
<td>$\alpha_{13}$</td>
<td>0.700</td>
</tr>
<tr>
<td>(V, $\alpha_{15}$)</td>
<td>($-,$ $+$)</td>
<td>1</td>
<td>$\alpha_{13}$</td>
<td>0.420</td>
</tr>
</tbody>
</table>

• For example, the fourth entry reads
  – the supertag $\alpha_{15}$, anchored by a verb (V)
  – has a left and a right dependent ($-,$ $+$)
  – the first word to the left ($-1$) with the supertag $\alpha_{13}$ serves as a dependent and
  – the strength of this association is represented by the probability 0.700
### Dependency Based Model

**Sent:** the purchase price includes two ancillary companies.

<table>
<thead>
<tr>
<th>POS:</th>
<th>D</th>
<th>N</th>
<th>N</th>
<th>V</th>
<th>D</th>
<th>A</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\beta_1$</td>
<td>$\alpha_5$</td>
<td>$\alpha_6$</td>
</tr>
<tr>
<td>Assig.</td>
<td>$\alpha_7$</td>
<td>$\beta_2$</td>
<td>$\alpha_8$</td>
<td>$\alpha_9$</td>
<td>$\alpha_{10}$</td>
<td>$\beta_3$</td>
<td>$\alpha_{11}$</td>
</tr>
<tr>
<td></td>
<td>$\alpha_{12}$</td>
<td>$\alpha_{13}$</td>
<td>$\alpha_{14}$</td>
<td>$\alpha_{15}$</td>
<td>$\alpha_{16}$</td>
<td>$\alpha_{17}$</td>
<td>$\alpha_{18}$</td>
</tr>
<tr>
<td>Final Assig.</td>
<td>$\alpha_1$</td>
<td>$\beta_2$</td>
<td>$\alpha_3$</td>
<td>$\alpha_{15}$</td>
<td>$\alpha_{10}$</td>
<td>$\beta_3$</td>
<td>$\alpha_6$</td>
</tr>
</tbody>
</table>

- Every anchor must find its dependents.
- Every dependent must be linked to a anchor.
- No two dependency arcs may cross one another.
Dependency Based Model

- Performance results on Wall Street Journal (WSJ) sentences

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Total number</th>
<th>Number correct</th>
<th>% correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supertags</td>
<td>915</td>
<td>707</td>
<td>77.26%</td>
</tr>
<tr>
<td>Dependency links</td>
<td>815</td>
<td>620</td>
<td>76.07%</td>
</tr>
</tbody>
</table>

- Issues:
  - Needs a parsed corpus as training material
  - Attempts at getting a complete linkage
  - Worst-case complexity: $O(n^3)$
  - Lots of parameters to train: $O(S^{2*DA})$
  - More like parsing than not