Lecture 12: PCFG parsing, Treebank Parsing: as good as gold?

Professor Robert C. Berwick
berwick@csail.mit.edu
Menu

• Statistical Parsing with Treebanks II:
  • Some details on lexicalization: heads, generation model, and parsing
  • How to find features automatically for parsing
  • The state of the state of the art: from features to discriminative parsing
• Statistical Parsing with Treebanks II: do these systems really acquire ‘knowledge of language’?
Why independence is a bad idea

\[
\begin{array}{c}
\text{S} \\
\text{A} & \text{B} \\
\text{a} & \text{a} & \text{a} & \text{a} \\
\text{S} \\
\text{A} & \text{C} \\
\text{a} & \text{a} & \text{a} \\
\text{S} \\
\text{C} \\
\text{a} & \text{a} & \text{a}
\end{array}
\]

\[
\begin{array}{ccc}
10x & 20x & 50x \\
\frac{10}{10+20+50} & = 0.125 & S \rightarrow A \ B \\
\frac{20}{10+20+50} & = 0.25 & S \rightarrow A \ C \\
\frac{50}{10+20+50} & = 0.625 & S \rightarrow C \\
\frac{10}{10+20} & = 0.334 & A \rightarrow a \ a \\
\frac{20}{10+20} & = 0.667 & A \rightarrow a \\
\frac{20}{20+50} & = 0.285 & B \rightarrow a \ a \\
\frac{50}{20+50} & = 0.714 & C \rightarrow a \ a \ a
\end{array}
\]
appropriate independence assumptions the usual approach in a statistical parser

The context-free nature of PCFGs, where a non-terminal can be expanded
likely parse tree does not even occur in the Treebank! And the reason for this

would be:

times and

If we assume that tree

The rule probabilities can be derived from a Treebank as we can observe

A

remove some independence assumptions by annotating each non-terminal with

B

appropriate independence assumptions the usual approach in a statistical parser

C

by any rule with that non-terminal in the left-hand side. In order to make

D

the following example. Consider a Treebank with three trees

E

above, the most plausible analysis gets the higher probability.

F

other. Since the probability for

G

NP PP

714 = 0

714 = 0

The Penn Treebank contains trees like in 5(a). The first approach was to

714 = 0

remove some independence assumptions by annotating each non-terminal with

714 = 0

appropriate independence assumptions the usual approach in a statistical parser

119. The parse tree

125

is the most likely tree for that input. The most

AC

there are two parses using the above PCFG:

AB

For input

AC

and

AB

there are two parses using the above PCFG:

AC

VP -> VP PP

AB

NP PP

P1:

P2:

(a)

(a)

p1:

p2:

p1 = 0.125 · 0.334 · 0.285 = 0.01189

p2 = 0.25 · 0.667 · 0.714 = 0.119

Parse of a a a a

And the other one?
What’s the problem?

The Penn Treebank contains trees like in 5(a). The first approach was to remove some independence assumptions by annotating each non-terminal with

\[ \text{NP \to NP PP} \]

The rule probabilities can be derived from a Treebank as we can observe:

\[ a \]

If we assume that tree \( t \) occurred 50 times then the PCFG we obtain from this Treebank has been set higher in the PCFG:

\[ a \]

Since all the other rules in one parse also occur in the Treebank, the most plausible analysis gets the higher probability.

Consider a Treebank with three trees above, the most plausible analysis gets the higher probability.

The probability for \( t \) would be:

\[ \frac{10}{10+20+50} \]

For input

\[ S \]

\[ A \]

\[ B \]

\[ C \]

\[ p \]

\[ A \]

\[ B \]

\[ C \]

\[ p \]

\[ 1 \]

\[ 2 \]

\[ 3 \]

\[ 6.863J/9.611J \] Fall 2012  Lecture 12
The “primary linguistic data”: input is the Penn Treebank

PTB is a (particular) *linguistic theory* not a plain “corpus”: it assumes *some* assignment of (linguistic) structure to sentences

Question: What Knowledge of Language does/can such systems acquire?
What are ‘heads’ of phrases?

• Head of XP is ‘X’

  \[
  S \to \text{NP VP} \quad \text{(VP is the head – nonstandard!)}
  \]
  \[
  \text{VP} \to \text{Vt NP} \quad \text{(Vt is the head)}
  \]
  \[
  \text{NP} \to \text{DT NN NN} \quad \text{(rightmost NN is head)}
  \]

• Lexicalized parsers use deterministic rules to ‘find’ heads (not always in a linguistically justified way)

• The phrase receives its head annotation from the head ‘child’ below
Example head-finding rules

• For NP expansions:
  • **If** the right-hand side (rhs) of the rule contains NN, NNS, or NNP **then** select rightmost NN, NNS, or NNP
  • **Else if** the rhs contains an NP **then** select the leftmost NP
  • **Else if** the rhs contains a JJ **then** select the rightmost JJ
  • **Else if** the rhs contains a CD **then** select the rightmost CD
  • **Else** select the rightmost child
Example for NP

NP → DT NNP NN NP
NP → DT NN NN NNP NNP
NP → NP PP
NP → DT JJ
NP → DT

6.863J/9.611J Fall 2012  Lecture 12
Lexicalized Chomsky normal form grammar definition

$G$ is a 4-tuple: $(N, T, R, Start)$

- $N$ is a finite set of non-terminal symbols
- $T$ is a finite set of terminal symbols (words)
- $R$ is a finite set of rule that are in one of these forms:
  - $X(h) \rightarrow Y_1(h)Y_2(w)$, for $X \in N$, and $Y_1, Y_2 \in N$, and $h, w \in T$
  - $X(h) \rightarrow Y_1(w)Y_2(h)$, for $X \in N$, and $Y_1, Y_2 \in N$, and $h, w \in T$
  - $X(h) \rightarrow h$, for $X \in N$, and $h \in T$
  - $X(w) \rightarrow w$, for $X \in N$, and $w \in T$
- $Start \in N$ is a distinguished start symbol
What does this do to grammar size?

- The grammar looks like Chomsky normal form, but it has potentially $O(|T|^2 \times |N|^3)$ rules.

- So if we parse an $n$ word sentence using the PCFG algorithm it might take $O(|n^3|T|^2|N|^3)$ time, and $|T|$ is huge (20K = 40K?)

- BUT in any one sentence $w_1 \ldots w_n$ of length $n$, at most $O(n^2 \times |N|^3)$ rules can be applicable, because any rules that contain a lexical item that is not in the sentence can be dismissed.

- Parsing time is $O(n^5|N|^3)$. 

6.863J/9.611J Fall 2012  Lecture 12
Propagating both heads and part of speech tags

S(questioned, Vt)

NP(lawyer, NN)
  DT the
  NN lawyer

VP(questioned, Vt)
  Vt questioned
  NP(witness)
    DT the
    NN witness
Two lexicalized parsing models with heads

- Charniak, 1997: head word annotation, P/R/F1 up to about 86.7%
- Collins, 1997: add POS tags, argument vs. adjuncts, subcategories for verbs, distance from head: 88.1%
Charniak generative model: decompose step by step (3 steps)

Rule: $S \rightarrow NP(NN) \ VP(Vt)$
    $S(\text{questioned}, Vt)$

$\downarrow$

$S(\text{questioned}, Vt)$

$NP(\_, NN) \ VP(\text{questioned}, Vt)$

$\downarrow$

$S(\text{questioned}, Vt)$

$NP(\text{lawyer}, NN) \ VP(\text{questioned}, Vt)$

Problem: note that the very first step requires an estimate based on counts of an entire rule.

Such counts are *sparse*:

- of 39, 400 training sentences in PTB, there are only 12, 409 rules
- 15% of all test data sentences have rule never seen
Charniak’s answer: smoothed estimate, steps 1 & 2

\[ p(\text{NP(NN) VP(Vt)}| S(\text{questioned, Vt})) = \]

\[ \lambda_1 \times \frac{\text{Count}(S(\text{questioned, Vt}) \rightarrow \text{NP(NN)VP(Vt)})}{\text{Count}(S(\text{questioned, Vt}))} \]

\[ + \lambda_2 \times \frac{\text{Count}(\text{S(_, Vt)} \rightarrow \text{NP(NN)VP(Vt)})}{\text{Count}(\text{S(_, Vt)})} \]

\[ 0 \leq \lambda_1, \lambda_2 \leq 1; \lambda_1 + \lambda_2 = 1 \]

Interpolation between word and tag

\[ p(\text{lawyer }| S(\text{questioned, Vt, VP, NP(NN)}) = \]

\[ \lambda_3 \times \frac{\text{Count}(\text{lawyer}| S(\text{questioned, Vt), VP, NP(NN)})}{\text{Count}(S(\text{questioned, Vt), VP, NP(NN)))} \]

\[ + \lambda_4 \times \frac{\text{Count}(\text{lawyer}| S(_, Vt), VP, NP(NN)))}{\text{Count}(S(_, Vt), VP, NP(NN)))} \]

\[ + \lambda_5 \times \frac{\text{Count}(\text{lawyer}| \text{NN})}{\text{Count}(\text{NN})} \]

\[ 0 \leq \lambda_3, \lambda_4, \lambda_5 \leq 1; \lambda_3 + \lambda_4 + \lambda_5 = 1 \]
And one more smoothed estimate for last step, step 3

\[ p(NP(\text{lawyer, NN}), \text{VP}) | S(\text{questioned, Vt}) = \]

\[ \left( \lambda_1 \times \frac{\text{Count}(S(\text{questioned, Vt}) \rightarrow NP(\text{NN}) VP(\text{Vt}))}{\text{Count}(S(\text{questioned, Vt}))} \right) + \lambda_2 \times \frac{\text{Count}(S(\_, Vt) \rightarrow NP(\text{NN}) VP(\text{Vt}))}{\text{Count}(S(\_, Vt))} \]

\[ + \left( \lambda_3 \times \frac{\text{Count}(\text{lawyer}|S(\text{questioned, Vt}), \text{VP, NP(\text{NN}))}}{\text{Count}(S(\text{questioned, Vt}), \text{VP, NP(\text{NN}))}} \right) \]

\[ + \lambda_4 \times \frac{\text{Count}(\text{lawyer}|S(\_, Vt), \text{VP, NP(\text{NN}))}}{\text{Count}(S(\_, Vt), \text{VP, NP(\text{NN}))}} \]

\[ + \lambda_5 \times \frac{\text{Count}(\text{lawyer}|\text{NN})}{\text{Count}(\text{NN})} \]
Many rules only occur a few times...

<table>
<thead>
<tr>
<th>Rule count</th>
<th>No. of Rules by Type</th>
<th>Percentage by Type</th>
<th>No. of Rules by Token</th>
<th>Percentage by Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6765</td>
<td>54.52</td>
<td>6765</td>
<td>0.72</td>
</tr>
<tr>
<td>2</td>
<td>1688</td>
<td>13.60</td>
<td>3376</td>
<td>0.36</td>
</tr>
<tr>
<td>3</td>
<td>695</td>
<td>5.60</td>
<td>2085</td>
<td>0.22</td>
</tr>
<tr>
<td>4</td>
<td>457</td>
<td>3.68</td>
<td>1828</td>
<td>0.19</td>
</tr>
<tr>
<td>5</td>
<td>329</td>
<td>2.65</td>
<td>1645</td>
<td>0.18</td>
</tr>
<tr>
<td>6–10</td>
<td>835</td>
<td>6.73</td>
<td>6430</td>
<td>0.68</td>
</tr>
<tr>
<td>11–20</td>
<td>496</td>
<td>4.00</td>
<td>7219</td>
<td>0.77</td>
</tr>
<tr>
<td>21–50</td>
<td>501</td>
<td>4.04</td>
<td>15931</td>
<td>1.70</td>
</tr>
<tr>
<td>51–100</td>
<td>204</td>
<td>1.64</td>
<td>14507</td>
<td>1.54</td>
</tr>
<tr>
<td>&gt; 100</td>
<td>439</td>
<td>3.54</td>
<td>879596</td>
<td>93.64</td>
</tr>
</tbody>
</table>
Refining the node expansion possibilities

• Charniak (1997) expands each phrase structure tree in a single step
• This works well to capture dependencies between children nodes
• But bad because of sparseness
• A pure dependency, one child at a time model is worse
• You can find the ‘Goldilocks spot’ by various ‘in between’ models, eg, generating children as a Markov process on both sides of the head
Using Markov chain in tree

- Step 1: generate Head of phrase
- Step 2: generate left modifiers as a Markov chain (until STOP symbol happens to be generated)
- Step 3: generate right modifiers as a Markov chain (until STOP symbol happens to be generated)
(For these, add in distance effect, argument/adjunct, and subcatgorization information)
Using Markov processes in tree

- Another method: model rule productions as Markov processes
- Step 1: generate category of head child

\[
S(\text{smiled}, V) \quad \downarrow
\]

Estimated as: \( p_h(VP | S, \text{smiled}, V) \)
Step 2: Generate left modifiers in a Markov chain

\[ S(\text{smiled}, V) \]

\[ \text{?? VP(} \text{smiled}, V) \]

\[ \downarrow \]

\[ S(\text{smiled}, V) \]

\[ \text{Obama, NNP VP(} \text{smiled}, V) \]

\[ p_h(\text{VP}|S, \text{smiled}, V) \times p_d(\text{NP(Obama, NNP)}|S, \text{VP, smiled, V, LEFT}) \]
Step 2: generate left modifiers

\[
p_h(\text{VP}|\text{S, smiled, V}) \times p_d(\text{NP(Obama, NNP)}|\text{S, VP, smiled, V, LEFT}) \times p_d(\text{NP(yesterday, NN)}|\text{S, VP, smiled, LEFT})
\]
Left mods generated until STOP is generated

\[
p_h(\text{VP} | \text{S,smiled,V}) \times p_d(\text{NP(Obama,NNP)} | \text{S,VP,smiled,V,LEFT}) \times
\]
\[
p_d(\text{NP(yesterday,NN)} | \text{S,VP,smiled,LEFT}) \times p_d(\text{STOP} | \text{S,VP,smiled,V,LEFT})
\]
Now generate right side the same way, from head to the right…

\[
p_h(\text{VP} | \text{S}, \text{smiled}, \text{V}) \times p_d(\text{NP(Obama, NNP)} | \text{S, VP, smiled, V, LEFT}) \times p_d(\text{NP(yesterday, NN)} | \text{S, VP, smiling, LEFT}) \times p_d(\text{STOP | S, VP, told, V, LEFT}) \times p_d(\text{STOP | S, VP, smiled, RIGHT})
\]
Adding an argument/adjunct feature

Q: what distinguishes *yesterday* from *Obama*?
Arguments vs. adjuncts

VP

V  NP

verb  object

VP(told,V)

V

told

NP(Romney,NNP)

NNP

Romney

NP(yesterday,NN)

NN

yesterday

SBAR(that,COMP)

...
Arguments & Adjuncts

• Complements: to a first order approx, the required ‘arguments’ to a predicate
• Adjuncts add information, but aren’t necessary
• Contrast: *I gave* vs. *I gave at the office*
Subcategorization frames

- Given the ‘who did what to whom’ even structure
- Syntactic reflex of the predicate-argument relations - lexical semantics
- You’ve already seen them in primitive form as Verbs of 0, 1, 2 arguments
- But other details: some verbs take, e.g., a proposition as an argument (*I think that*....)
- So we add a probability model for this also
Add tags to distinguish arguments & adjuncts
Add tags…

```
VP
  V
  verb
  NP-C
  object

VP
  V
  verb
  NP
  adjunct

VP(told,V)
  V
  told
  NP-C(Romney,NNP)
  NP(yesterday,NN)

SBAR(that,COMP)
```

```
VP
  V
  that
  COMP
```

6.863J/9.611J Fall 2012 Lecture 12
Add probabilistic selection of a particular subcategorization frame (#/type args to verb)

\[ S(told, V) \]

Step 2: generate head

\[ \downarrow \]

\[ S(told, V) \]
\[ \uparrow \]
\[ VP(told, V) \]
\[ NP-C \]

Was: \( p_h(VP|S, told, V) \)
Now: \( p_h(VP|S, told, V) \times p_{lc}(NP-C | S, VP, told, V) \)
Generate left modifiers as Markov chain

\[ S(told,V) \]

\[ \begin{array}{c}
?? \\
\end{array} \]

\[ VP(told,V) \]

\[ \{ \} \]

\[ \downarrow \]

\[ S(told,V) \]

\[ NP(Obama,NNP) \]

\[ VP(told,V) \]

\[
\begin{align*}
\rho_h(VP|S,\text{told},V) \times \rho_{lc}(\text{NP-C} | S,VP,\text{told},V) \times \\
\rho_d(\text{NP}(Obama,NNP)|S,VP,\text{told},V,\text{LEFT},\{\text{NP-C}\})
\end{align*}
\]
\[ p_h(VP|S, \text{told}, V) \times p_{lc}([\text{NP-C}] | S, \text{VP, told, V}) \times \\
 p_d(\text{NP(Obama, NNP)} | S, \text{VP, told, V, LEFT, [NP-C]}) \times \\
 p_d(\text{NP(yesterday, NN)} | S, \text{VP, told, LEFT{}}) \]
\[ p_h(VP|S,\text{told}, V) \times p_{lc}({\{NP-C}\} | S, \text{VP, told, V}) \times \]
\[ p_d(\text{NP(Obama,NNP)} | S,\text{VP,told,V,LEFT,\{}\{NP-C\}\}) \times \]
\[ p_d(\text{NP(yesterday,NN)} | S,\text{VP,told,LEFT,\{}\}) \times \]
\[ p_d(\text{STOP} | S,\text{VP,told,V,LEFT,\{}\}) \]
So far then

- Find heads of the rules to capture dependencies
- Break generation of parse tree (rule applications) into markov process steps
- Build dependencies back in through subcategorization, node annotation
Evaluation of treebank parsers

[Diagram showing a tree structure with the following nodes and labels:
S, NP, VP, DT, NN, Vt, PP, IN, DT, NN, the, guy, ate, ice-cream, with, chocolate, the, chocolate.

Label Start pt Stop pt
NP 1 2
NP 4 5
NP 4 8
PP 6 8
NP 7 8
VP 3 8
S 1 8]
Precision & Recall

**Gold standard**

```
S
  NP
    DT NN
      the guy
  VP
    Vt
      ate
  PP
    DT NN
      the ice-cream
  IN
    with
  NP
    DT NN
      the chocolate
```

<table>
<thead>
<tr>
<th>Label</th>
<th>Start pt</th>
<th>Stop pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>NP</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>NP</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>PP</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>NP</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>VP</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

\[ G = \# \text{ constituents in gold standard} = 7 \]

**Parser output**

```
S
  NP
    DT NN
      the guy
  VP
    Vt
      ate
  PP
    DT NN
      the ice-cream
  IN
    with
  NP
    DT NN
      the chocolate
```

<table>
<thead>
<tr>
<th>Label</th>
<th>Start pt</th>
<th>Stop pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>NP</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>PP</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>NP</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>VP</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

\[ P = \# \text{ in parse output} = 6 \]

\[ C = \# \text{ correct} = 6 \]

Recall = \( \frac{C}{G} \times 100 = \frac{6}{7} \times 100 = 85.7\% \)

Precision = \( \frac{C}{P} = \frac{6}{6} \times 100 = 100\% \)
### Basic Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall %</th>
<th>Precision %</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCFGs</td>
<td>70.6</td>
<td>74.8</td>
</tr>
<tr>
<td>Decision Trees</td>
<td>84.0</td>
<td>84.3</td>
</tr>
<tr>
<td>Lexicalization</td>
<td>85.3</td>
<td>85.7</td>
</tr>
<tr>
<td>Conditional, max entropy</td>
<td>86.3</td>
<td>87.5</td>
</tr>
<tr>
<td>Generative lexical, Charniak</td>
<td>86.7</td>
<td>86.6</td>
</tr>
<tr>
<td>Model 1 Collins generative lexical</td>
<td>87.5</td>
<td>87.7</td>
</tr>
<tr>
<td>Model 2 Collins w/ subcat</td>
<td>88.1</td>
<td>88.3</td>
</tr>
<tr>
<td>Stanford</td>
<td>89.1</td>
<td>88.9</td>
</tr>
<tr>
<td>Adaptive cats, Berkeley</td>
<td>91.2</td>
<td>90.4</td>
</tr>
</tbody>
</table>

How much does ‘perfection’ require?
True parse

Parser’s output

Much syntax is still hard…
Complexity of Lexicalized PCFG parsing

Time charged:
- $i, j, k \Rightarrow n^3$
- $A, B, C \Rightarrow |G|^3$
- Naively, $|G|$ becomes huge
- Any word in $S$ as head
  $\Rightarrow |G| \rightarrow |G| \cdot n$

Naïve version: running time is $O(|G|^3 \times n^5)$

This is what we actually have for Charniak, Collins, $O(|G|^3 \times n^5)$, but use heuristics (beam search) to be faster
Initialization

1. for $i = 1 \ldots n$, $X \in N$
2. \[ \pi[i, i, X] = P(X(w_i) \to w_i | X(w_i)) \] // part-of-speech tagging
3. end for

Filling the table

4. for $\ell = 1 \ldots n - 1$, $i = 1 \ldots n - \ell$, $h = i \ldots i + \ell$, $X \in N$ // iterate over the table
5. $j = i + \ell$; $p_{\text{max}} = 0$
6. for $s = h \ldots j$, $m = s + 1 \ldots j$, $Y \in N$, $Z \in N$ // left-headed case
7. \[ p = P(X(w_h) \to Y(w_h)Z(w_m) | X(w_h)) \]
   \[ \times \pi[i, s, h, Y] \times \pi[s + 1, j, m, Z] \]
8. if $p > p_{\text{max}}$ then
9. \[ p_{\text{max}} = p \]
10. \[ \text{split}[i, j, h, X] = (\text{left}, s, m, Y, Z) \]
11. end if
12. end for
13. for $s = i \ldots h - 1$, $m = i \ldots s$, $Y \in N$, $Z \in N$ // right-headed case
14. \[ p = P(X(w_h) \to Y(w_m)Z(w_h) | X(w_h)) \]
   \[ \times \pi[i, s, m, Y] \times \pi[s + 1, j, h, Z] \]
15. if $p > p_{\text{max}}$ then
16. \[ p_{\text{max}} = p \]
17. \[ \text{split}[i, j, h, X] = (\text{right}, s, m, Y, Z) \]
18. end if
19. end for
20. \[ \pi[i, j, h, X] = p_{\text{max}} \]
21. end for

Selecting a root

22. $p_{\text{tree}} = 0$; $h_{\text{tree}} = \text{nil}$
23. for $h = 1 \ldots n$
24. \[ p = P_S(S(w_h)) \times \pi[1, n, h, S] \]
25. if $p > p_{\text{tree}}$ then
26. \[ p_{\text{tree}} = p; \quad h_{\text{tree}} = h \]
27. end if
28. end for
29. return $(p_{\text{tree}}, h_{\text{tree}}, \text{split})$
Initialization

for $i = 1 \ldots n$, $X \in N$

$$\pi[i, i, i, X] = P(X(w_i) \rightarrow w_i | X(w_i))$$

// part-of-speech tagging

end for

Filling the table

for $\ell = 1 \ldots n - 1$, $i = 1 \ldots n - \ell$, $h = i \ldots i + \ell$, $X \in N$  // iterate over the table

$j = i + \ell$;  $p_{\text{max}} = 0$

for $s = h \ldots j$, $m = s + 1 \ldots j$, $Y \in N$, $Z \in N$  // left-headed case

$$p = P(X(w_h) \rightarrow Y(w_h)Z(w_m) | X(w_h))$$

$$\times \pi[i, s, h, Y] \times \pi[s + 1, j, m, Z]$$

if $p > p_{\text{max}}$ then

$p_{\text{max}} = p$

split$[i, j, h, X] = (\text{left}, s, m, Y, Z)$

end if

end for
Can one do better?

• Yes: can get back to $O(n^3)$, if we put some constraints on the grammars (more later)
Automatic Annotation Induction?

[Matuzaki et. al ’05, Prescher ’05]

• **Advantages:**
  
  • **Automatically learned:**
    
    Label *all* nodes with latent variables.
    
    Same number *k* of subcategories for all categories.
  
• **Disadvantages:**

  • Grammar gets too large
  
  • Most categories are oversplit while others are undersplit.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
</tr>
</tbody>
</table>
Learning Latent Annotations (Petrov & Klein, 2006)

- Can you automatically find good symbols?
  - Brackets are known
  - Base categories are known
  - Induce subcategories
  - Split/merge refinement

Uses EM for trees, as sketched before
Overview

Limit of computational resources

- Hierarchical Training
- Adaptive Splitting
- Parameter Smoothing

Total Number of grammar symbols

Parsing accuracy (F1)
Refinement of the DT tag (like 6.034 decision trees)
Too many categories? How do we know?

DT

the (0.50)
a (0.24)
The (0.08)

DT-1  DT-2  DT-3  DT-4  DT-5  DT-6  DT-7  DT-8
Hierarchical refinement of the DT tag

- the (0.50)
  - a (0.24)
  - The (0.08)

- that (0.15)
  - this (0.14)
  - some (0.11)

- a (0.61)
  - the (0.19)
  - an (0.11)

- the (0.80)
  - a (0.01)

- this (0.39)
  - that (0.28)
  - That (0.11)

- some (0.20)
  - all (0.19)
  - those (0.12)
Hierarchical Estimation Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>87.3</td>
</tr>
<tr>
<td>Hierarchical Training</td>
<td>88.4</td>
</tr>
</tbody>
</table>
Refinement of the , tag

• Splitting all categories the same amount is wasteful:

![Diagram showing the refinement of the , tag]
The DT tag revisited

Oversplit?
Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful
Adaptive splitting: Goldilocks principle

- If we want to split complex categories more
- Split everything and roll back splits that were least useful
Adaptive splitting – Goldilocks principle

- If we want to split complex categories more
- Split everything and roll back splits that were least useful
Adaptive Splitting – ah, just right

• Evaluate loss in likelihood from removing each split

\[ = \]

\textit{Data likelihood with split reversed}

\textit{Data likelihood with split}

• No loss in accuracy when 50% of the splits are reversed.
Adaptive Splitting

• Evaluate loss in likelihood from removing each split
  =

  Data likelihood with split reversed

  Data likelihood with split

• No loss in accuracy when 50% of the splits are reversed.
Adaptive Splitting Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>88.4</td>
</tr>
<tr>
<td>With 50% Merging</td>
<td>89.5</td>
</tr>
</tbody>
</table>
Number of phrasal categories

[Bar chart showing the number of occurrences for various phrasal categories including NP, VP, PP, ADVP, S, ADJP, SBAR, QP, WHNP, PRN, NX, SINV, PRT, WHPP, SQ, CONJP, FRAG, NAC, UCP, WHADVP, INTJ, SBARQ, RRC, WHADJP, X, ROOT, LST.]
Number of phrasal subcategories
Number of Phrasal Subcategories

6.863J/9.611J Fall 2012  Lecture 12
Smoothing

- Heavy splitting can lead to overfitting
- Smoothing can fix this.
Linear smoothing over rules

\[ p_x = P(A_x \rightarrow BC) \]

\[ p'_x = (1 - \alpha)p_x + \alpha \overline{p} \]

where \( \bar{p} = \frac{1}{n} \sum_x p_x \)
Total Number of grammar symbols

Parsing accuracy (F1)

50% Merging and Smoothing
50% Merging
Hierarchical Training
Flat Training

6.863J/9.611J Fall 2012 Lecture 12
Total Number of grammar symbols

50% Merging and Smoothing
50% Merging
Hierarchical Training
Flat Training
Total Number of grammar symbols

Parsing accuracy (F1)

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>89.5</td>
</tr>
<tr>
<td>With Smoothing</td>
<td>90.7</td>
</tr>
</tbody>
</table>
The bottom line

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 ≤ 40 words</th>
<th>F1 all words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
<td>85.7</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
<td>86.1</td>
</tr>
<tr>
<td>Collins ’99</td>
<td>88.6</td>
<td>88.2</td>
</tr>
<tr>
<td>Charniak &amp; Johnson ’05</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td>Auto learning cats</td>
<td><strong>90.2</strong></td>
<td><strong>89.7</strong></td>
</tr>
</tbody>
</table>
Do the categories make sense?

• Proper Nouns (NNP):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NNP-12</td>
<td>John</td>
<td>Robert</td>
<td>James</td>
</tr>
<tr>
<td>NNP-2</td>
<td>J.</td>
<td>E.</td>
<td>L.</td>
</tr>
<tr>
<td>NNP-1</td>
<td>Bush</td>
<td>Noriega</td>
<td>Peters</td>
</tr>
<tr>
<td>NNP-15</td>
<td>New</td>
<td>San</td>
<td>Wall</td>
</tr>
<tr>
<td>NNP-3</td>
<td>York</td>
<td>Francisco</td>
<td>Street</td>
</tr>
</tbody>
</table>

• Proper Names (PRP)

<table>
<thead>
<tr>
<th>PRP-0</th>
<th>it</th>
<th>He</th>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRP-1</td>
<td>it</td>
<td>he</td>
<td>they</td>
</tr>
<tr>
<td>PRP-2</td>
<td>it</td>
<td>them</td>
<td>him</td>
</tr>
</tbody>
</table>
### Linguistic Candy

- **Relative adverbs (RBR):**

<table>
<thead>
<tr>
<th>RBR-0</th>
<th>further</th>
<th>lower</th>
<th>higher</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBR-1</td>
<td>more</td>
<td>less</td>
<td>More</td>
</tr>
<tr>
<td>RBR-2</td>
<td>earlier</td>
<td>Earlier</td>
<td>later</td>
</tr>
</tbody>
</table>

- **Cardinal Numbers (CD):**

<table>
<thead>
<tr>
<th>CD-7</th>
<th>one</th>
<th>two</th>
<th>Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-4</td>
<td>1989</td>
<td>1990</td>
<td>1988</td>
</tr>
<tr>
<td>CD-11</td>
<td>million</td>
<td>billion</td>
<td>trillion</td>
</tr>
<tr>
<td>CD-0</td>
<td>1</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>CD-3</td>
<td>1</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>CD-9</td>
<td>78</td>
<td>58</td>
<td>34</td>
</tr>
</tbody>
</table>
The dangers of eating with chopsticks

Now let’s calculate the odds… what will the PCFG trained on this data return for *eat rice with chopsticks*?

Tree #2? No! It will return Tree #3. Why?
Let us count the ways...

108 times

6 times

2 times

VP→V count: 108
VP→V NP count: 6 + 2 = 8
VP→VP PP count: 6
Total VP→ = 122

p(VP→VP PP) = 6/122 = 0.0492

NP→rice/chopsticks count: 12+4=16
NP→V NP count: 2
Total NP→ = 18

p(NP→NP PP) = 2/9 = 0.111… > p(VP→VP PP) Oops!
What to do?

• Be more *discriminating*
The latest bake-off

<table>
<thead>
<tr>
<th>System</th>
<th>F</th>
<th>P</th>
<th>R</th>
<th>Exact</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ENHANCED TRAINING / SYSTEMS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charniak-SR</td>
<td>92.07</td>
<td>92.44</td>
<td>91.70</td>
<td>44.87</td>
<td>1.8</td>
</tr>
<tr>
<td>Charniak-R</td>
<td>91.41</td>
<td>91.78</td>
<td>91.04</td>
<td>44.04</td>
<td>1.8</td>
</tr>
<tr>
<td>Charniak-S</td>
<td>91.02</td>
<td>91.16</td>
<td>90.89</td>
<td>40.77</td>
<td>1.8</td>
</tr>
<tr>
<td><strong>STANDARD PARSERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Berkeley</td>
<td>90.06</td>
<td>90.30</td>
<td>89.81</td>
<td>36.59</td>
<td>4.2</td>
</tr>
<tr>
<td>Charniak</td>
<td>89.71</td>
<td>89.88</td>
<td>89.55</td>
<td>37.25</td>
<td>1.8</td>
</tr>
<tr>
<td>SSN</td>
<td>89.42</td>
<td>89.96</td>
<td>88.89</td>
<td>32.74</td>
<td>1.8</td>
</tr>
<tr>
<td>BUBS</td>
<td>88.50</td>
<td>88.57</td>
<td>88.43</td>
<td>31.62</td>
<td>27.6</td>
</tr>
<tr>
<td>Bikel</td>
<td>88.16</td>
<td>88.23</td>
<td>88.10</td>
<td>32.33</td>
<td>0.8</td>
</tr>
<tr>
<td>Collins-3</td>
<td>87.66</td>
<td>87.82</td>
<td>87.50</td>
<td>32.22</td>
<td>2.0</td>
</tr>
<tr>
<td>Collins-2</td>
<td>87.62</td>
<td>87.77</td>
<td>87.48</td>
<td>32.51</td>
<td>2.2</td>
</tr>
<tr>
<td>Collins-1</td>
<td>87.09</td>
<td>87.29</td>
<td>86.90</td>
<td>30.35</td>
<td>3.3</td>
</tr>
<tr>
<td>Stanford-L</td>
<td>86.42</td>
<td>86.35</td>
<td>86.49</td>
<td>27.65</td>
<td>0.7</td>
</tr>
<tr>
<td>Stanford-U</td>
<td>85.78</td>
<td>86.48</td>
<td>85.09</td>
<td>28.35</td>
<td>2.7</td>
</tr>
</tbody>
</table>
To what degree is syntactic analysis a solved problem?

- PTB F1: 0.84 Magerman (1995) → 0.92 Charniak (2006)
- But: single aggregate score misleading (sentence accuracy ~10–25%)
- Nonlocal dependencies, e.g., *What did you buy* - zilch
- How well do these systems actually work in recovering “who did what to whom”?
A more thorough job (Bender et al, 2011)

- Select ten ‘hard’ syntactic phenomena, local and non-local
- Find 100 ‘suitable’ sentences per phenomenon in Wikipedia;
- Dual-annotate and reconcile for ‘relevant’ dependencies
- Run seven off-the-shelf parsers on this data (the strings)
- Design parser-specific patterns for automated evaluation
Example syntactic patterns

1. Nonlocal ‘bare relatives’:
   A classic example Schumacher provides ___ is that of education.

2. Nonlocal ‘tough’ adjectives:
   Original copies are very hard to find ___

3. Right-node raising:
   He also played for ___ and managed Kilmarnock.

4. ‘It’ expletives (not arguments):
   Crew negligence is blamed, and it is suggested that the flight crew were drunk.

5. Absolutives (local):
   the format consisted of 12 games, each team facing the other teams twice.

6. ‘Controlled’ arguments: Alfred ... Continued __ to paint full time.
Summary of phenomena types

1. barerel: bare relative clauses
2. tough: tough phrases
3. rnr: right-node raising
4. itexpl: “It” expletives
5. vpart: verb-particles (“threw out”)
6. ned: noun with ed modified by another noun
7. absol: NP followed by non-finite predicate
8. vger: verbal gerund (in NP position) – “accessing the website…”
9. argadj: displaced argument – “the story shows through flashbacks…"
10. control
The test examples

- Select from English Wikipedia (Wikiwoods)
- 900 million tokens
- Random selection of candidates
- Dual-vetted: skip too simple, too complex, false positives
- Result: 1000 sentences covering the 10 phenomena
Is this fair play?

• Are these phenomena represented in the PTB?
• Bare relatives, verb-particles, absolutes directly represented
• ‘tough’ construction reliably annotated (missing argument linked to wh head)
• Right-node raising (rnr) & vger (gerundive verbs), explicit, though a few false + for rnr; for vger, pos tag is sometimes nominal (instead of verb)
Fair Play?

• For remaining 4, there are some issues:
  1. Control: ok, but includes others (*ate the meat raw*)
  2. Itexpl: annotation –NONE- *EXP* is ok, but omits some
  3. Argadj: interleave args/adjuncts – PTB doesn’t distinguish between post-verbal modifiers & verbal arguments; eg, PP-loc, but not consistently applied
  4. Ned: not mentioned; e.g., *gritty-eyed* left as JJ
47 Million Wikipedia sentences, 900M tokens

<table>
<thead>
<tr>
<th>Phenomenon</th>
<th>Frequency</th>
<th>Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>barerel</td>
<td>2.12%</td>
<td>546</td>
</tr>
<tr>
<td>tough</td>
<td>0.07%</td>
<td>175</td>
</tr>
<tr>
<td>rnr</td>
<td>0.69%</td>
<td>1263</td>
</tr>
<tr>
<td>itexpl</td>
<td>0.13%</td>
<td>402</td>
</tr>
<tr>
<td>vpart</td>
<td>4.07%</td>
<td>765</td>
</tr>
<tr>
<td>ned</td>
<td>1.18%</td>
<td>349</td>
</tr>
<tr>
<td>absol</td>
<td>0.51%</td>
<td>963</td>
</tr>
<tr>
<td>vger</td>
<td>5.16%</td>
<td>679</td>
</tr>
<tr>
<td>argadj</td>
<td>3.60%</td>
<td>1346</td>
</tr>
<tr>
<td>control</td>
<td>3.78%</td>
<td>124</td>
</tr>
</tbody>
</table>
Annotation

• Specify target scheme; parallel annotation by two expert linguists
• Initial agreement: 79% (full sentences); all mismatches reconciled
Example annotation

The Act having been passed in that year, Jessop withdrew, and Whitworth carried on with the assistance of his son

<table>
<thead>
<tr>
<th>Item</th>
<th>Type</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1011079100200</td>
<td>ABSOL</td>
<td>having</td>
</tr>
<tr>
<td>1011079100200</td>
<td>ABSOL</td>
<td>withdrew MOD having</td>
</tr>
<tr>
<td>1011079100200</td>
<td>ABSOL</td>
<td>carried+on MOD having</td>
</tr>
</tbody>
</table>
The ‘localness’ or not of the examples

<table>
<thead>
<tr>
<th>Phenomenon</th>
<th>Head</th>
<th>Type</th>
<th>Dependent</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare relatives (barerel)</td>
<td>gapped predicate in relative modified noun</td>
<td>ARG2/MOD</td>
<td>modified noun</td>
<td>3.0 (8)</td>
</tr>
<tr>
<td></td>
<td>modified noun</td>
<td>MOD</td>
<td>top predicate of relative</td>
<td>3.3 (8)</td>
</tr>
<tr>
<td>Tough adjectives (tough)</td>
<td>tough adjective</td>
<td>ARG2</td>
<td>to-VP complement</td>
<td>1.7 (5)</td>
</tr>
<tr>
<td></td>
<td>gapped predicate in to-VP</td>
<td>ARG2</td>
<td>subject/modifiee of adjective</td>
<td>6.4 (21)</td>
</tr>
<tr>
<td>Right Node Raising (rnr)</td>
<td>verb/prep2</td>
<td>ARG2</td>
<td>shared noun</td>
<td>2.8 (9)</td>
</tr>
<tr>
<td></td>
<td>verb/prep1</td>
<td>ARG2</td>
<td>shared noun</td>
<td>6.1 (12)</td>
</tr>
<tr>
<td>Expletive It (itexpl)</td>
<td><em>it</em>-subject taking verb</td>
<td>!ARG1</td>
<td><em>it</em></td>
<td>1.2 (3)</td>
</tr>
<tr>
<td></td>
<td>raising-to-object verb</td>
<td>!ARG2</td>
<td><em>it</em></td>
<td>–</td>
</tr>
<tr>
<td>Verb+particle constructions (vpart)</td>
<td>particle</td>
<td>!ARG2</td>
<td>complement</td>
<td>2.7 (9)</td>
</tr>
<tr>
<td></td>
<td>verb+particle</td>
<td>ARG2</td>
<td>complement</td>
<td>3.7 (10)</td>
</tr>
<tr>
<td>Adj/Noun2 + Noun1-<em>ed</em> (ned)</td>
<td>head noun</td>
<td>MOD</td>
<td>Noun1-<em>ed</em></td>
<td>2.4 (17)</td>
</tr>
<tr>
<td></td>
<td>Noun1-<em>ed</em></td>
<td>ARG1/MOD</td>
<td>Adj/Noun2</td>
<td>1.0 (1.5)</td>
</tr>
<tr>
<td>Absolutives (absol)</td>
<td>absolute predicate</td>
<td>ARG1</td>
<td>subject of absolute</td>
<td>1.7 (12)</td>
</tr>
<tr>
<td></td>
<td>main clause predicate</td>
<td>MOD</td>
<td>absolute predicate</td>
<td>9.8 (26)</td>
</tr>
<tr>
<td>Verbal gerunds (vger)</td>
<td>selecting head gerund</td>
<td>ARG[1,2]</td>
<td>gerund</td>
<td>1.9 (13)</td>
</tr>
<tr>
<td></td>
<td>selecting verb</td>
<td>ARG2/MOD</td>
<td>first complement/modifier of gerund</td>
<td>2.3 (8)</td>
</tr>
<tr>
<td>Interleaved arg/adj (argadj)</td>
<td>selecting verb</td>
<td>MOD</td>
<td>interleaved adjunct</td>
<td>1.2 (7)</td>
</tr>
<tr>
<td></td>
<td>selecting verb</td>
<td>ARG[2,3]</td>
<td>displaced complement</td>
<td>5.9 (26)</td>
</tr>
<tr>
<td>Control (control)</td>
<td>“upstairs” verb</td>
<td>ARG[2,3]</td>
<td>“downstairs” verb</td>
<td>2.4 (23)</td>
</tr>
<tr>
<td></td>
<td>“downstairs” verb</td>
<td>ARG1</td>
<td>shared argument</td>
<td>4.8 (17)</td>
</tr>
</tbody>
</table>

Table 3: Dependencies labeled for each phenomenon type, including average and maximum surface distances.

To evaluate, extraction patterns using about 400 regex patterns
The players

<table>
<thead>
<tr>
<th>Trained ‘Directly’ on the (WSJ Portion of the) PTB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Stanford</strong> (Klein &amp; Manning, 2003) factored model; GR output;</td>
</tr>
<tr>
<td>• <strong>C&amp;J</strong> (Charniak &amp; Johnson, 2005) Stanford GR post-processor;</td>
</tr>
<tr>
<td>• <strong>MST</strong> (McDonald et al., 2005) second-order projective model.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trained Indirectly on the (WSJ Portion of the) PTB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Enju</strong> (Miyao et al., 2004) HPSG; predicate–argument outputs;</td>
</tr>
<tr>
<td>• <strong>C&amp;C</strong> (Clark &amp; Curran, 2007) CCG; grammatical relation outputs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(Partly) Analytically Engineered</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>RASP</strong> (Briscoe et al., 2006) PoS ‘tag sequence grammar’; GRs;</td>
</tr>
<tr>
<td>• <strong>XLE</strong> (Kaplan et al., 2004) hand-built LFG and lexicon; f-structures.</td>
</tr>
</tbody>
</table>
Individual dependency recall

Good Recovery of Some Phenomena: VGER, VPART, CONTROL.

6.863J/9.611J Fall 2012  Lecture 12
**Predictable:** ITEXPL requires lexical knowledge (not in ‘PTB’).
Some Dependencies Lost on Most Parsers: RNR, NED, ABSOL.

Only TWO parsers exceed 50% for 5 phenomena
ALL SYSTEMS BELOW 50% for 3 phenomena
Perhaps more work needs to be done?

C&J vs. Stanford: Average 56 % vs. 52 %.
Propagated Errors: PP attachment

(a) Parser output

(b) Gold tree
Error Types (1+6 more)

1. (PP attachment: high vs. low)
2. NP attachment: high vs. low)
3. Modifier attachment
4. Clause attachment
5. Unary error: S, NP, etc.
6. Conjunction
7. NP internal structure
What happens when we move to a new domain?
The usual setup for language acquisition

Initial State → Learning Procedure → Final State

Input Data

What is a treebank analog?
= $2500/baby

So are we done???

Question: How well does this do, from a cognitive perspective?
The lessons

• Cognitive ‘Turing tests’ for treebank parsers - How does their acquired knowledge hold up as a model for cognitively accurate language acquisition?

• Three lessons

1. Jackendoff’s lesson: can computers learn to read? – the Linguistic Society of America’s pamphlet on Why can’t computers use English?

2. Levin’s Lesson: Robustness, sensitivity, and lexical semantics: can computers learn to drink milk with cookies?

3. The Las Vegas lesson: what happens in Las Vegas stays in Las Vegas – computers can count, and people cannot