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

\[
\begin{align*}
\text{S} & \quad \text{S} & \quad \text{S} \\
/ & \quad/ & \quad/ \\
\text{A} & \quad \text{A} & \quad \text{C} \\
/ & \quad/ & \quad/ \\
\text{B} & \quad \text{C} & \quad \text{B} \\
/ & \quad/ & \quad/ \\
\text{a} & \quad \text{a} & \quad \text{a} \\
\end{align*}
\]

\[
\begin{array}{c|c|c}
\text{10x} & \text{20x} & \text{50x} \\
\hline
10 &= 0.125 & S \rightarrow AB \\
10 &= 0.25 & S \rightarrow AC \\
10 &= 0.625 & S \rightarrow C \\
10 &= 0.334 & A \rightarrow aa \\
10 &= 0.667 & A \rightarrow a \\
10 &= 0.285 & B \rightarrow aa \\
10 &= 0.714 & C \rightarrow aaa \\
\end{array}
\]

Parse of \text{a a a a} 

\[
\begin{align*}
\text{S} & \quad \text{S} \\
/ & \quad/ \\
\text{A} & \quad \text{A} \\
/ & \quad/ \\
\text{B} & \quad \text{C} \\
/ & \quad/ \\
\text{a} & \quad \text{a} \\
\end{align*}
\]

\[
p_1 = 0.125 \cdot 0.334 \cdot 0.285 = 0.01189, \quad p_2 = 0.25 \cdot 0.667 \cdot 0.714 = 0.119
\]
What’s the problem?

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’
  
  \[
  \begin{align*}
  S &\rightarrow \text{NP VP} \quad (\text{VP is the head — nonstandard!}) \\
  \text{VP} &\rightarrow \text{Vt NP} \quad (\text{Vt is the head}) \\
  \text{NP} &\rightarrow \text{DT NN NN} \quad (\text{rightmost NN is head})
  \end{align*}
  \]

- 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 → NNP PP
NP → DT JJ
NP → DT

Lexicalized Chomsky normal form
grammar definition

\( G \) is a 4-tuple: \( (N, T, R, \text{Start}) \)

- \( N \) 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 \( w \in T \)
- \( \text{Start} \in N \) 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 $O(n^5|N|^3)$.

---

Propagating both heads and part of speech tags

```
S(questioned, Vt)
  / \          / \    
NP(lawyer, NN) VP(questioned, Vt)
 /     \     / \     
DT the NN Vt questioned NP(witness)
     /     / \\
    lawyer questioned the \\
```

---
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: $S \rightarrow NP(\text{NN}) \ VP(\text{Vt})$

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

$\Downarrow$

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

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

$$\Downarrow$$

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

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

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

\[ p(\text{NP}(\text{NN}) \ \text{VP}(\text{Vt}) | \text{S}(\text{questioned}, \ \text{Vt})) = \]
\[ \lambda_1 \times \frac{\text{Count}(\text{S}(\text{questioned}, \ \text{Vt}) \rightarrow \text{NP}(\text{NN})\text{VP}(\text{Vt}))}{\text{Count}(\text{S}(\text{questioned}, \ \text{Vt}))} + \lambda_2 \times \frac{\text{Count}(\text{S}(\text{ _, Vt}) \rightarrow \text{NP}(\text{NN})\text{VP}(\text{Vt}))}{\text{Count}(\text{S}(\text{ _, Vt}))} \]
\[ 0 \leq \lambda_1, \lambda_2 \leq 1; \lambda_1 + \lambda_2 = 1 \]

Interpolation between word and tag

\[ p(\text{lawyer} | \text{S}(\text{questioned}, \ \text{Vt}, \ \text{VP}, \ \text{NP}(\text{NN})) = \]
\[ \lambda_3 \times \frac{\text{Count}(\text{lawyer}(\text{S}(\text{questioned}, \ \text{Vt}), \ \text{VP}, \ \text{NP}(\text{NN}))}{\text{Count}(\text{S}(\text{questioned}, \ \text{Vt}, \ \text{VP}, \ \text{NP}(\text{NN}))} + \lambda_4 \times \frac{\text{Count}(\text{lawyer}(\text{S}(\text{ _, Vt}), \ \text{VP}, \ \text{NP}(\text{NN}))}{\text{Count}(\text{S}(\text{ _, Vt}, \ \text{VP}, \ \text{NP}(\text{NN}))} + \lambda_5 \times \frac{\text{Count}(\text{lawyer}(\text{NN})}{\text{Count}(\text{NN})} \]
\[ 0 \leq \lambda_3, \lambda_4, \lambda_5 \leq 1; \lambda_3 + \lambda_4 + \lambda_5 = 1 \]

And one more smoothed estimate for last step, step 3

\[ p(\text{NP}(\text{lawyer}, \ \text{NN}), \ \text{VP}) | \text{S}(\text{questioned}, \ \text{Vt})) = \]
\[ \left( \lambda_1 \times \frac{\text{Count}(\text{S}(\text{questioned}, \ \text{Vt}) \rightarrow \text{NP}(\text{NN})\text{VP}(\text{Vt}))}{\text{Count}(\text{S}(\text{questioned}, \ \text{Vt}))} \right) + \lambda_2 \times \frac{\text{Count}(\text{S}(\text{ _, Vt}) \rightarrow \text{NP}(\text{NN})\text{VP}(\text{Vt}))}{\text{Count}(\text{S}(\text{ _, Vt}))} \]
\[ + \left( \lambda_3 \times \frac{\text{Count}(\text{lawyer}(\text{S}(\text{questioned}, \ \text{Vt}), \ \text{VP}, \ \text{NP}(\text{NN}))}{\text{Count}(\text{S}(\text{questioned}, \ \text{Vt}, \ \text{VP}, \ \text{NP}(\text{NN}))} \right) + \lambda_4 \times \frac{\text{Count}(\text{lawyer}(\text{S}(\text{ _, Vt}), \ \text{VP}, \ \text{NP}(\text{NN}))}{\text{Count}(\text{S}(\text{ _, Vt}, \ \text{VP}, \ \text{NP}(\text{NN}))} \]
\[ + \lambda_5 \times \frac{\text{Count}(\text{lawyer}(\text{NN})}{\text{Count}(\text{NN})} \right) \]
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 subcategorization information)

Using Markov processes in tree

• Another method: model rule productions as Markov processes
• Step 1: generate category of head child
  \[ S(\text{smiled, V}) \]
  \[ \Downarrow \]
  \[ S(\text{smiled, V}) \]

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

\[ p_h(VP|S, \text{smiled}, V) \times p_d(NP(\text{Obama}, NNP)|S, VP, \text{smiled}, V, \text{LEFT}) \]

Step 2: generate left modifiers

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

\[
\begin{align*}
S & \rightarrow \text{smiled, V} \\
\downarrow & \\
S & \rightarrow \text{smiled, V} \\
\downarrow & \\
\text{STOP} & \rightarrow \text{yesterday, NN} \frac{\text{Obama, NNP}}{\text{smiled, V}} \\
\end{align*}
\]

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

Now generate right side the same way, from head to the right...

\[
\begin{align*}
S & \rightarrow \text{smiled, V} \\
\downarrow & \\
S & \rightarrow \text{smiled, V} \\
\downarrow & \\
\text{STOP} & \rightarrow \text{yesterday, NN} \frac{\text{Obama, NNP}}{\text{smiled, V}} \\
\end{align*}
\]

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

Q: what distinguishes yesterday from Obama?

Arguments vs. adjuncts
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

S

NP-C

argument

VP

verb

NP

adjunct

V

verb

S

(smiled, V)

NP(yesterday, NN)

VP

S

Arg(Obama, NNP)

VP

(smiled, V)

NP

Romney

NP

yesterday

SBAR

(that, COMP)

...
Add probabilistic selection of a particular subcategorization frame (≧/type args to verb)

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

Step 2: generate head

\[ \downarrow \]

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

\[ VP(\text{told}, V) \]

\[ \text{NP-C} \]

Was: \( p_h(\text{VP}|S, \text{told}, V) \)

Now: \( p_h(\text{VP}|S, \text{told}, V) \times p_{lc}(\text{NP-C} | S, \text{VP}, \text{told}, V) \)

---

Generate left modifiers as Markov chain

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

\[ \uparrow \]

\[ ?\ ? \]

\[ VP(\text{told}, V) \]

\[ \{ \} \]

\[ \downarrow \]

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

\[ \text{NP}(\text{Obama}, \text{NNP}) \]

\[ \text{VP}(\text{told}, V) \]

\[ p_h(\text{VP}|S, \text{told}, V) \times p_{lc}(\text{NP-C} | S, \text{VP}, \text{told}, V) \times \]

\[ p_d(\text{NP}(\text{Obama}, \text{NNP})|S, \text{VP}, \text{told}, V, \text{LEFT}, \{\text{NP-C}\}) \]
\[ p_h(VP|S, \text{told, V}) \times p_{le}(\{\text{NP-C}\} | S, VP, \text{told, V}) \times p_d(\text{NP}(\text{Obama, NNP}) | S, VP, \text{told, V, LEFT, } \{\text{NP-C}\}) \times p_d(\text{NP}(\text{yesterday, NN}) | S, VP, \text{told, LEFT } \{\}) \]

6.863J/9.611J Fall 2012 Lecture 12
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

### Gold standard

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

### Precision & Recall

```
P = # in parse output = 6
C = # correct = 6
Recall = C/G x 100 = 6/7 x 100 = 85.7%
Precision = C/P = 6/6 x 100 = 100%
```

### Table

<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 = # constituents in gold standard = 7

P = # in parse output = 6

Recall = C/G x 100 = 6/7 x 100 = 85.7%

Precision = C/P = 6/6 x 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?

Automatic Annotation Induction?

[Matsuzaki 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.
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

- Hierarchical Training
- Adaptive Splitting
- Parameter Smoothing

Limit of computational resources
Refinement of the DT tag
(like 6.034 decision trees)

Too many categories? How do we know?

6.863J/9.611J Fall 2012 Lecture 12
Hierarchical refinement of the DT tag

![Diagram showing hierarchical refinement of DT tag]

Hierarchical Estimation Results

<table>
<thead>
<tr>
<th>Total Number of grammar symbols</th>
<th>Parsing accuracy (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>74</td>
</tr>
<tr>
<td>300</td>
<td>76</td>
</tr>
<tr>
<td>500</td>
<td>78</td>
</tr>
<tr>
<td>700</td>
<td>80</td>
</tr>
<tr>
<td>900</td>
<td>82</td>
</tr>
<tr>
<td>1100</td>
<td>84</td>
</tr>
<tr>
<td>1300</td>
<td>86</td>
</tr>
<tr>
<td>1500</td>
<td>88</td>
</tr>
<tr>
<td>1700</td>
<td>90</td>
</tr>
</tbody>
</table>

<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:

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
  = \[ \frac{\text{Data likelihood with split reversed}}{\text{Data likelihood with split}} \]

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

- Evaluate loss in likelihood from removing each split
  
  \[
  \text{Data likelihood with split reversed} \quad \frac{=}{=} \quad \text{Data likelihood with split}
  \]

- 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

Number of phrasal subcategories
Smoothing

- Heavy splitting can lead to overfitting
- Smoothing can fix this.

Number of Lexical Subcategories

![Graph showing the number of lexical subcategories with different categories such as NNP, JJ, NNS, and NN.](image-url)
Linear smoothing over rules

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

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

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

<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>90.2</td>
<td>89.7</td>
</tr>
</tbody>
</table>

Do the categories make sense?

- **Proper Nouns (NNP):**
  
<table>
<thead>
<tr>
<th>NNP-14</th>
<th>NNP-12</th>
<th>NNP-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Robert</td>
<td>James</td>
</tr>
<tr>
<td>J.</td>
<td>E.</td>
<td>L.</td>
</tr>
<tr>
<td>Bush</td>
<td>Noriega</td>
<td>Peters</td>
</tr>
<tr>
<td>New</td>
<td>San</td>
<td>Wall</td>
</tr>
<tr>
<td>York</td>
<td>Francisco</td>
<td>Street</td>
</tr>
</tbody>
</table>

- **Proper Names (PRP):**

<table>
<thead>
<tr>
<th>PRP-0</th>
<th>PRP-1</th>
<th>PRP-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>It</td>
<td>He</td>
<td>I</td>
</tr>
<tr>
<td>it</td>
<td>he</td>
<td>they</td>
</tr>
<tr>
<td>it</td>
<td>them</td>
<td>him</td>
</tr>
</tbody>
</table>

6.863J/9.611J Fall 2012 Lecture 12
Linguistic Candy

- Relative adverbs (RBR):
  - RBR-0: further, lower, higher
  - RBR-1: more, less, More
  - RBR-2: earlier, Earlier, later

- Cardinal Numbers (CD):
  - CD-7: one, two, Three
  - CD-11: million, billion, trillion
  - CD-0: 1, 50, 100
  - CD-3: 1, 30, 31
  - CD-9: 78, 58, 34

The dangers of eating with chopsticks

Tree 1
VP
  | VP
  | V
  | eat

Tree 2
VP
  | VP
  | PP
  | P
  | with
  | NP
  | rice
  | eat

Tree 3
VP
  | NP
  | PP
  | P
  | with
  | NP
  | rice
  | with
  | chopsticks

108 times 6 times 2 times

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…

Tree 1

VP

V

I

eat

VP

V

NP

P

I

eat

rice

with

chopsticks

Tree 2

VP

V

NP

P

P

I

eat

rice

with

chopsticks

Tree 3

VP

V

NP

P

I

eat

rice

with

chopsticks

108 times

<table>
<thead>
<tr>
<th>Rule</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP → V</td>
<td>108</td>
</tr>
<tr>
<td>VP → V NP</td>
<td>6 + 2 = 8</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>6</td>
</tr>
<tr>
<td>Total VP →</td>
<td>122</td>
</tr>
<tr>
<td>p(VP → VP PP)</td>
<td>6/122 = 0.0492</td>
</tr>
</tbody>
</table>

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>ENHANCED TRAINING / SYSTEMS</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>STANDARD PARSERS</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
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 3: Dependencies labeled for each phenomenon type, including average and maximum surface distances.

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

Trained ‘Directly’ on the (WSJ Portion of the) PTB

- **Stanford** (Klein & Manning, 2003) factored model; GR output;
- **C&J** (Charniak & Johnson, 2005) Stanford GR post-processor;
- **MST** (McDonald et al., 2005) second-order projective model.

Trained Indirectly on the (WSJ Portion of the) PTB

- **Enju** (Miyao et al., 2004) HPSG; predicate–argument outputs;
- **C&C** (Clark & Curran, 2007) CCG; grammatical relation outputs.

(Partly) Analytically Engineered

- **RASP** (Briscoe et al., 2006) PoS ‘tag sequence grammar’; GRs;
- **XLE** (Kaplan et al., 2004) hand-built LFG and lexicon; f-structures.

Individual dependency recall

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

So are we done???

Knowledge of Language

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