They solved the problem with statistics.
They solved the problem with statistics.
They solved the problem with statistics
They solved the problem with statistics
Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital, creating another potential obstacle to the government's sale of sick thrifts.
Phrase Structure Parsing

- Phrase structure parsing organizes syntax into *constituents* or *brackets*
- In general, this involves nested trees
- Linguists can, and do, argue about details
- Lots of ambiguity
- Not the only kind of syntax...
- First part of today’s lecture
Dependency Parsing

- Directed edges between pairs of word (head, dependent)
- Can handle free word-order languages
- Very efficient decoding algorithms exist
- Second part of today’s lecture
Classical NLP: Parsing

- Write symbolic or logical rules:
  - VBD  VB
  - VBN  VBZ
  - NNP  NNS  NN  NNS  CD  NN

  Fed raises interest rates 0.5 percent

- Use deduction systems to prove parses from words
  - Minimal grammar on “Fed raises” sentence: 36 parses
  - Real-size grammar: many millions of parses

- This scaled very badly, didn’t yield broad-coverage tools

Fed raises interest rates 0.5 percent

Grammar (CFG)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT → S</td>
<td>NP → NP PP</td>
</tr>
<tr>
<td>S → NP VP</td>
<td>VP → VBP NP</td>
</tr>
<tr>
<td>NP → DT NN</td>
<td>VP → VBP NP PP</td>
</tr>
<tr>
<td>NP → NN NNS</td>
<td>PP → IN NP</td>
</tr>
</tbody>
</table>

Lexicon:

<table>
<thead>
<tr>
<th>Lexicon</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN → interest</td>
<td>NN → interest</td>
</tr>
<tr>
<td>NNS → raises</td>
<td>NNS → raises</td>
</tr>
<tr>
<td>VBP → interest</td>
<td>VBP → interest</td>
</tr>
<tr>
<td>VBZ → raises</td>
<td>VBZ → raises</td>
</tr>
</tbody>
</table>
Attachments

- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink
Probabilistic Context-Free Grammars

- A context-free grammar is a tuple \( <N, T, S, R> \)
  - \( N \) : the set of non-terminals
    - Phrasal categories: \( S, NP, VP, ADJP, \) etc.
    - Parts-of-speech (pre-terminals): \( NN, JJ, DT, VB \)
  - \( T \) : the set of terminals (the words)
  - \( S \) : the start symbol
    - Often written as ROOT or TOP
    - Not usually the sentence non-terminal \( S \)
  - \( R \) : the set of rules
    - Of the form \( X \rightarrow Y_1 Y_2 ... Y_k \), with \( X, Y_i \in N \)
    - Examples: \( S \rightarrow NP \ VP, \ VP \rightarrow VP CC VP \)
    - Also called rewrites, productions, or local trees

- A PCFG adds:
  - A top-down production probability per rule \( P(Y_1 Y_2 ... Y_k \mid X) \)
Treebank Grammars

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn’t work well):

```
S → NP VP .
NP → PRP 0.5
NP → DT NN 0.5
VP → VBD NP 1.0
PRP → She 1.0
```

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers without lexicalization.
Treebank Grammar Scale

- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller
Chomsky Normal Form

- **Chomsky normal form:**
  - All rules of the form $X \rightarrow YZ$ or $X \rightarrow w$
  - In principle, this is no limitation on the space of (P)CFGs
    - N-ary rules introduce new non-terminals

  - Unaries / empties are “promoted”

- In practice it’s kind of a pain:
  - Reconstructing n-aries is easy
  - Reconstructing unaries is trickier
  - The straightforward transformations don’t preserve tree scores

- Makes parsing algorithms simpler!
A Recursive Parser

- Will this parser work?
- Why or why not?
- Memory requirements?

```
bestScore(X,i,j,s)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return max score(X->YZ) * 
               bestScore(Y,i,k) * 
               bestScore(Z,k,j)
```
A Memoized Parser

• One small change:

```java
bestScore(X,i,j,s)
    if (scores[X][i][j] == null)
        if (j = i+1)
            score = tagScore(X,s[i])
        else
            score = max  score(X->YZ) * 
                           bestScore(Y,i,k) * 
                           bestScore(Z,k,j)
    scores[X][i][j] = score
return scores[X][i][j]
```
A Bottom-Up Parser (CKY)

- Can also organize things bottom-up

```python
bestScore(s)
    for (i : [0,n-1])
        for (X : tags[s[i]])
            score[X][i][i+1] =
                tagScore(X, s[i])
    for (diff : [2,n])
        for (i : [0,n-diff])
            j = i + diff
            for (X->YZ : rule)
                for (k : [i+1, j-1])
                    score[X][i][j] = max score[X][i][j],
                                    score(X->YZ) * 
                                    score[Y][i][k] * 
                                    score[Z][k][j]
```
Time: Theory

• How much time will it take to parse?

  • For each diff (<= n)
    • For each i (<= n)
      • For each rule $X \rightarrow Y Z$
        • For each split point k
          Do constant work

• Total time: $|\text{rules}| \times n^3$

• Something like 5 sec for an unoptimized parse of a 20-word sentences, or 0.2sec for an optimized parser
Unary Rules

- Unary rules?

```python
bestScore(X, i, j, s)
    if (j = i+1)
        return tagScore(X, s[i])
    else
        return max
            max score(X->YZ) * bestScore(Y, i, k) * bestScore(Z, k, j)
        max score(X->Y) * bestScore(Y, i, j)
```
We need unaries to be non-cyclic

- Can address by pre-calculating the unary closure
- Rather than having zero or more unaries, always have exactly one

Alternate unary and binary layers
Reconstruct unary chains afterwards
Alternating Layers

\[
\text{bestScoreB}(X,i,j,s) = \begin{cases} 
\text{tagScore}(X,s_i) & \text{if } (j = i+1) \\
\max \max \text{ score}(X\rightarrow YZ) \times \text{bestScoreU}(Y,i,k) \times \text{bestScoreU}(Z,k,j) & \text{else}
\end{cases}
\]

\[
\text{bestScoreU}(X,i,j,s) = \begin{cases} 
\text{tagScore}(X,s_i) & \text{if } (j = i+1) \\
\max \max \text{ score}(X\rightarrow Y) \times \text{bestScoreB}(Y,i,j) & \text{else}
\end{cases}
\]
Treebank Grammars

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn’t work well):

```
S → NP VP .
NP → PRP 0.5
NP → DT NN 0.5
VP → VBD NP 1.0
PRP → She 1.0
...
```

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak ’96</td>
<td>72.0</td>
</tr>
</tbody>
</table>
Conditional Independence?

- Not every NP expansion can fill every NP slot
  - A grammar with symbols like “NP” won’t be context-free
  - Statistically, conditional independence too strong
• Independence assumptions are often too strong.

Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).

Also: the subject and object expansions are correlated!
The Game of Designing a Grammar

- Structure Annotation [Johnson ‘98, Klein & Manning ’03]
- Lexicalization [Collins ‘99, Charniak ‘00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. ’06]
- (Neural) CRF Parsing [Hall et al. ’14, Durrett & Klein ’15]
A Fully Annotated (Unlexicalized) Tree

[Model] Charniak ’96 72.0
Klein&Manning ’03 86.3
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Head lexicalization [Collins ’99, Charniak ’00]
Problems with PCFGs

- If we do no annotation, these trees differ only in one rule:
  - VP → VP PP
  - NP → NP PP
- Parse will go one way or the other, regardless of words
- We addressed this in one way with unlexicalized grammars (how?)
- Lexicalization allows us to be sensitive to specific words
Problems with PCFGs

- What’s different between basic PCFG scores here?
- What (lexical) correlations need to be scored?
Lexicalized Trees

- Add “headwords” to each phrasal node
  - Syntactic vs. semantic heads
  - Headship not in (most) treebanks
  - Usually use head rules, e.g.:
    - **NP:**
      - Take leftmost NP
      - Take rightmost N*
      - Take rightmost JJ
      - Take right child
    - **VP:**
      - Take leftmost VB*
      - Take leftmost VP
      - Take left child
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

  \[
  \text{VP(saw)} \rightarrow \text{VBD(saw)} \quad \text{NP-C(her)} \quad \text{NP(today)}
  \]

- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps
Lexical Derivation Steps

A derivation of a local tree [Collins ‘99]

Choose a head tag and word

Choose a complement bag

Generate children (incl. adjuncts)

Recursively derive children
Lexicalized Grammars

- **Challenges:**
  - Many parameters to estimate: requires sophisticated smoothing techniques
  - Exact inference is too slow: requires pruning heuristics
  - Difficult to adapt to new languages: At least head rules need to be specified, typically more changes needed

<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>Charniak ’00</td>
<td>90.1</td>
</tr>
</tbody>
</table>
bestScore(X,i,j,h)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return max
      max
        score(X[h]->Y[h] Z[h']) *
          bestScore(Y,i,k,h)
          bestScore(Z,k,j,h')
      score(X[h]->Y[h'] Z[h]) *
          bestScore(Y,i,k,h')
          bestScore(Z,k,j,h)
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Automatic clustering
Latent Variable Grammars

[Matsuzaki et al. ’05, Petrov et al. ’06]
Learning Latent Annotations

EM algorithm:

- Brackets are known
- Base categories are known
- Only induce subcategories

Just like Forward-Backward for HMMs.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak ’00</td>
<td>90.1</td>
</tr>
<tr>
<td>Petrov et al. ’06</td>
<td>90.6</td>
</tr>
</tbody>
</table>
Refinement of the DT tag
Hierarchical Refinement

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

 the (0.54)
 a (0.25)
 The (0.09)

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

 the (0.80)
 a (0.01)

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

 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>Flat Training</td>
<td>87.3</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>88.4</td>
</tr>
</tbody>
</table>

Parsing accuracy (F1)

- 74.0000
- 78.2500
- 82.5000
- 86.7500
- 91.0000

Total Number of grammar symbols

- 100
- 525
- 950
- 1375
- 1800
Refinement of the `, tag

- Splitting all categories equally is wasteful:
Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful
Adaptive Splitting Results

<table>
<thead>
<tr>
<th>Total Number of grammar symbols</th>
<th>100</th>
<th>500</th>
<th>900</th>
<th>1300</th>
<th>1700</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parsing accuracy (F1)</td>
<td>74.00</td>
<td>78.25</td>
<td>82.50</td>
<td>86.75</td>
<td>91.00</td>
</tr>
</tbody>
</table>

50% Merging
Hierarchical Training
Flat Training

<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%</td>
<td>89.5</td>
</tr>
</tbody>
</table>
Number of Phrasal Subcategories
# Learned Splits

- **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>

- **Personal pronouns (PRP):**

<table>
<thead>
<tr>
<th>PRP-0</th>
<th>It</th>
<th>He</th>
<th>I</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>
Learned Splits

- **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>
Bayesian Symbol Refined TSG

[Shindo et al. ’12]

• Latent Variable Tree-Substitution Grammar

Hierarchical Generation Process:
• Joint model of everything
• Complex sampling scheme needed

Sparse!

1. SR-TSG
2. SR-CFG
3. RU-CFG

Pitman-Yor process
Spectral Learning for PCFGs

• EM is a local method
  • Can never be sure to have the global optimum
  • Significant variance between different runs

• Spectral methods
  • Provably find the global optimum
  • Compute SVD of training data
  • Efficient to run
  • But currently not competitive in practice

[Luque et al. ’12, Cohen et al. ’12]
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - CRF Parsing (+Neural Network Representations)
Generative vs. Discriminative

**Generative**

Maximize joint likelihood of gold tree **and** sentence

EM-algorithm

EASY: expectations over observed trees

[Matsuzaki et al. ‘05, Petrov et al. ‘06]

**Discriminative**

Maximize conditional likelihood of gold tree **given** sentence

Gradient-based algorithm

HARD: expectations over all trees

[Petrov & Klein ‘07, ‘08]
Objective Functions

Generative Objective Function:

$$\max_{\theta} \mathcal{L}_\theta(\gamma, w_1...w_n)$$  
[Petrov, Barrett, Thibaux & Klein ’06]

Discriminative Objective Function:

$$\max_{\theta} \mathcal{L}_\theta(\gamma|w_1...w_n)$$  
[Petrov & Klein ’08, Finkel et. al ’08]

Bayesian Objective Function:

$$\max_{\theta} \mathcal{P}(\theta|\gamma)\mathcal{L}_\theta(\gamma, w_1...w_n)$$  
[Liang, Petrov, Jordan & Klein ’07]
(Neural) CRF Parsing


Be a tree

Score of VP over this span

$w \cdot f_s$

dense neural network

sparse log-linear model

He gave a speech
CRF Parsing Sparse Features

\[ P(T|x) \propto \prod_{r \in T} \exp(\text{score}(r)) \]

\[ \text{score}(2\text{NP}_7 \rightarrow 2\text{NP}_4 \ 4\text{PP}_7) = w \top f(2\text{NP}_7 \rightarrow 2\text{NP}_4 \ 4\text{PP}_7) \]

FirstWord = a & NP \rightarrow NP PP

PrevWord = gave & NP \rightarrow NP PP

AfterSplit = on & NP \rightarrow NP PP

FirstWord = a & NP

...
Neural CRF Model

\[
score(2NP_7 \rightarrow 2NP_4 \ 4PP_7) =
\]

\[
W \odot \left( f_s(2X_7 \rightarrow 2X_4 \ 4X_7) f_o^T(NP \rightarrow NP \ PP) \right)
\]

\[
f_s = g(H_v)
\]

(arbitrary neural network)

He gave a speech on foreign policy.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Petrov et al. ’06</td>
<td>90.6</td>
</tr>
<tr>
<td>Durrett et al. ’16</td>
<td>91.3</td>
</tr>
</tbody>
</table>
LSTM Parsing [Vinyals et al. ’15]

- Treat parsing as a sequence-to-sequence prediction problem
- Completely ignores tree structure, uses LSTMs as black boxes

\[
P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})
\]
They solved the problem with statistics
They solved the problem with statistics
For each chart item $X[i,j]$, compute posterior probability:

$$\frac{P_{IN}(X, i, j) \cdot P_{OUT}(X, i, j)}{P_{IN}(\text{root}, 0, n)} < \text{threshold}$$

E.g. consider the span 5 to 12:
Bracket Posteriors
Hierarchical Pruning

coarse: ...

split in two: ...

split in four: ...

split in eight: ...

QP  NP  VP
QP1  QP2  NP1  NP2  VP1  VP2
QP1  QP3  QP4  NP1  NP2  NP3  NP4  VP1  VP2  VP3  VP4

...
Parsing Times (per sentence)

- Fine: 62 seconds
- X-Bar - Fine: 5 seconds
- Coarse-to-Fine original grammars: 2.1 seconds
- Coarse-to-Fine projected grammars: 0.6 seconds
Multi-Lingual Results

Test set $F_1$ all lengths

- Petrov et al. '06*
- Hall et al. '14
- Durrett et al. '15

<table>
<thead>
<tr>
<th>Language</th>
<th>Petrov et al. '06*</th>
<th>Hall et al. '14</th>
<th>Durrett et al. '15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>78.7</td>
<td>80.2</td>
<td>82.0</td>
</tr>
<tr>
<td>Basque</td>
<td>74.7</td>
<td>83.4</td>
<td>83.5</td>
</tr>
<tr>
<td>French</td>
<td>79.8</td>
<td>81.3</td>
<td>90.7</td>
</tr>
<tr>
<td>German</td>
<td>78.7</td>
<td>81.0</td>
<td>86.8</td>
</tr>
<tr>
<td>Hebrew</td>
<td>85.2</td>
<td>88.3</td>
<td>90.7</td>
</tr>
<tr>
<td>Hungarian</td>
<td>85.4</td>
<td>88.6</td>
<td>90.7</td>
</tr>
<tr>
<td>Korean</td>
<td>78.6</td>
<td>82.2</td>
<td>86.8</td>
</tr>
<tr>
<td>Polish</td>
<td>80.2</td>
<td>80.2</td>
<td>80.6</td>
</tr>
<tr>
<td>Swedish</td>
<td>80.9</td>
<td>81.3</td>
<td>83.2</td>
</tr>
<tr>
<td>Average</td>
<td>85.1</td>
<td>86.3</td>
<td>88.3</td>
</tr>
</tbody>
</table>