Algorithms for NLP

Parsing V

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley
Agenda-Based Parsing
Agenda-Based Parsing

- Agenda-based parsing is like graph search (but over a hypergraph)

- Concepts:
  - Numbering: we number fenceposts between words
  - “Edges” or items: spans with labels, e.g. PP[3,5], represent the sets of trees over those words rooted at that label (cf. search states)
  - A chart: records edges we’ve expanded (cf. closed set)
  - An agenda: a queue which holds edges (cf. a fringe or open set)
Building an item for the first time is called discovery. Items go into the agenda on discovery.

To initialize, we discover all word items (with score 1.0).

AGENDA

| critics[0,1], write[1,2], reviews[2,3], with[3,4], computers[4,5] |

CHART [EMPTY]

<table>
<thead>
<tr>
<th>critics</th>
<th>write</th>
<th>reviews</th>
<th>with</th>
<th>computers</th>
</tr>
</thead>
</table>
Unary Projection

- When we pop a word item, the lexicon tells us the tag item successors (and scores) which go on the agenda
Item Successors

- When we pop items off of the agenda:
  - Graph successors: unary projections (NNS → critics, NP → NNS)
    
    \[ Y[i,j] \text{ with } X \rightarrow Y \text{ forms } X[i,j] \]
  - Hypergraph successors: combine with items already in our chart
    
    \[ Y[i,j] \text{ and } Z[j,k] \text{ with } X \rightarrow Y \ Z \text{ form } X[i,k] \]

  - Enqueue / promote resulting items (if not in chart already)
  - Record backtraces as appropriate
  - Stick the popped edge in the chart (closed set)

- Queries a chart must support:
  - Is edge X[i,j] in the chart? (What score?)
  - What edges with label Y end at position j?
  - What edges with label Z start at position i?
An Example

Critics write reviews with computers.
Empty Elements

- Sometimes we want to posit nodes in a parse tree that don’t contain any pronounced words:

  I want you to parse this sentence
  I want [   ] to parse this sentence

- These are easy to add to a agenda-based parser!
  - For each position $i$, add the “word” edge $\varepsilon[i,i]$
  - Add rules like $\text{NP} \rightarrow \varepsilon$ to the grammar
  - That’s it!
With weighted edges, order matters
- Must expand optimal parse from bottom up (subparses first)
- CKY does this by processing smaller spans before larger ones
- UCS pops items off the agenda in order of decreasing Viterbi score
- A* search also well defined

You can also speed up the search without sacrificing optimality
- Can select which items to process first
- Can do with any “figure of merit” [Charniak 98]
- If your figure-of-merit is a valid A* heuristic, no loss of optimality [Klein and Manning 03]
There was nothing magical about words spanning exactly one position.

When working with speech, we generally don’t know how many words there are, or where they break.

We can represent the possibilities as a lattice and parse these just as easily.
Learning PCFGs
Treebank PCFGs

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

```
ROOT
  |__ S
    |__ NP
    |____ PRP
    |____ VBD
  |___ VP
    |____ ADJP
      |____ JJ
        |____ right

ROOT -> S 1
S -> NP VP . 1
NP -> PRP 1
VP -> VBD ADJP 1
.....
```

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>72.0</td>
</tr>
</tbody>
</table>
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
Non-Independence

- 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!
Grammar Refinement

- Example: PP attachment

![Grammar Tree Diagram]

They raised a point of order
Grammar Refinement

- Structure Annotation [Johnson ’98, Klein&Manning ’03]
- Lexicalization [Collins ’99, Charniak ’00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. ’06]
Structural Annotation
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Structural annotation
Typical Experimental Setup

- Corpus: Penn Treebank, WSJ

- Training: sections 02-21
- Development: section 22 (here, first 20 files)
- Test: section 23

- Accuracy – F1: harmonic mean of per-node labeled precision and recall.
- Here: also size – number of symbols in grammar.
- Vertical Markov order: rewrites depend on past $k$ ancestor nodes. (cf. parent annotation)

![Diagram of Vertical Markov Order 1 and 2]

![Bar charts for Vertical Markov Order 1 and 2]

<table>
<thead>
<tr>
<th>Vertical Markov Order</th>
<th>1</th>
<th>2v</th>
<th>2</th>
<th>3v</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbols</td>
<td>72%</td>
<td>73%</td>
<td>74%</td>
<td>75%</td>
<td>76%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vertical Markov Order</th>
<th>1</th>
<th>2v</th>
<th>2</th>
<th>3v</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbols</td>
<td>0</td>
<td>5000</td>
<td>10000</td>
<td>20000</td>
<td>25000</td>
</tr>
</tbody>
</table>
Horizontal Markovization

Order 1

Order ∞

Symbols

Horizonal Markov Order

<table>
<thead>
<tr>
<th>Horizontal Markov Order</th>
<th>0</th>
<th>1</th>
<th>2v</th>
<th>2</th>
<th>inf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>70%</td>
<td>71%</td>
<td>72%</td>
<td>73%</td>
<td>74%</td>
</tr>
</tbody>
</table>

Symbols

Horizontal Markov Order

<table>
<thead>
<tr>
<th>Horizontal Markov Order</th>
<th>0</th>
<th>1</th>
<th>2v</th>
<th>2</th>
<th>inf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2v</td>
<td>2</td>
<td>inf</td>
</tr>
</tbody>
</table>
 Unary Splits

- Problem: unary rewrites used to transmute categories so a high-probability rule can be used.

- Solution: Mark unary rewrite sites with -U

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>77.8</td>
<td>7.5K</td>
</tr>
<tr>
<td>UNARY</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
</tbody>
</table>
Tag Splits

- Problem: Treebank tags are too coarse.

- Example: Sentential, PP, and other prepositions are all marked IN.

- Partial Solution:
  - Subdivide the IN tag.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
<tr>
<td>SPLIT-IN</td>
<td>80.3</td>
<td>8.1K</td>
</tr>
</tbody>
</table>
A Fully Annotated (Unlex) Tree

```
ROOT
  /\      \
S^ROOT-v
  |     /\  \
""S  NP^S-B  VP^S-VBF-v
  |     |    /\ /\  |
" This VBZ^BE^VP NP^VP-B
     |        |        |
     "panic  "buying
```
### Some Test Set Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>LP</th>
<th>LR</th>
<th>F1</th>
<th>CB</th>
<th>0 CB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magerman 95</td>
<td>84.9</td>
<td>84.6</td>
<td>84.7</td>
<td>1.26</td>
<td>56.6</td>
</tr>
<tr>
<td>Collins 96</td>
<td>86.3</td>
<td>85.8</td>
<td>86.0</td>
<td>1.14</td>
<td>59.9</td>
</tr>
<tr>
<td>Unlexicalized</td>
<td>86.9</td>
<td>85.7</td>
<td>86.3</td>
<td>1.10</td>
<td>60.3</td>
</tr>
<tr>
<td>Charniak 97</td>
<td>87.4</td>
<td>87.5</td>
<td>87.4</td>
<td>1.00</td>
<td>62.1</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.7</td>
<td>88.6</td>
<td>88.6</td>
<td>0.90</td>
<td>67.1</td>
</tr>
</tbody>
</table>

- Beats “first generation” lexicalized parsers.
- Lots of room to improve – more complex models next.
Binarization / Markovization

NP
  /    \
DT  JJ  NN  NN

v=1, h=∞

NP
  /    \
DT  @NP[DT]
  |
JJ  @NP[DT,JJ]
  |
NN  @NP[DT,JJ,NN]

v=1, h=1

NP
  /    \
DT  @NP[DT]
  |
JJ  @NP[...JJ]
  |
NN  @NP[...NN]

v=1, h=0

NP
  /    \
DT  @NP
  |
JJ  @NP
  |
NN  @NP
  |
NN
Binarization / Markovization

NP

v=2, h=∞

NP^VP
DT^NP @NP^VP[DT]
JJ^NP @NP^VP[DT,JJ]
NN^NP @NP^VP[DT,JJ,NN]
NN^NP

v=2, h=1

NP^VP
DT^NP @NP^VP[DT]
JJ^NP @NP^VP[...J]
NN^NP @NP^VP[...J,NN]
NN^NP

v=2, h=0

NP^VP
DT^NP @NP
JJ^NP @NP
NN^NP @NP
NN^NP
Grammar Projections

Coarse Grammar

NP
  ---
  DT  @NP
    ---
    JJ  @NP
      ---
      NN  @NP
        ---
        NN

Fine Grammar

NP^VP
  ---
  DT^NP  @NP^VP[DT]
    ---
    JJ^NP  @NP^VP[... JJ]
      ---
      NN^NP  @NP^VP[... NN]
        ---
        NN^NP

NP → DT @NP

NP^VP → DT^NP @NP^VP[DT]

Note: X-Bar Grammars are projections with rules like XP → Y @X or XP → @X Y or @X → X
## Grammar Projections

<table>
<thead>
<tr>
<th>Coarse Symbols</th>
<th>Fine Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>NP^VP</td>
</tr>
<tr>
<td>@NP</td>
<td>NP^S</td>
</tr>
<tr>
<td>DT</td>
<td>@NP^VP[DT]</td>
</tr>
<tr>
<td></td>
<td>@NP^S[DT]</td>
</tr>
<tr>
<td></td>
<td>@NP^VP[...,JJ]</td>
</tr>
<tr>
<td></td>
<td>@NP^S[...,JJ]</td>
</tr>
<tr>
<td></td>
<td>DT^NP</td>
</tr>
</tbody>
</table>
Efficient Parsing for Structural Annotation
Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability:

$$P(X| i, j, S) < \text{threshold}$$

E.g. consider the span 5 to 12:
Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability:

$$\frac{\alpha(X, i, j) \cdot \beta(X, i, j)}{\alpha(\text{root}, 0, n)} < \text{threshold}$$

E.g. consider the span 5 to 12:
Computing Marginals

\[ \alpha(X, i, j) = \sum_{X \rightarrow YZ} \sum_{k \in (i,j)} P(X \rightarrow YZ) \alpha(Y, i, k) \alpha(Z, k, j) \]
Computing Marginals

\[
\beta(X, i, j) = \sum_{Y \rightarrow ZX} \sum_{k \in [0, i]} P(Y \rightarrow ZX) \beta(Y, k, j) \alpha(B, k, i) \\
+ \sum_{Y \rightarrow XZ} \sum_{k \in (j, n]} P(Y \rightarrow XZ) \beta(Y, i, k) \alpha(Z, j, k)
\]
Computing (Max-)Marginals

\[ \beta(x, i, j) = \sum_{y \in \mathbb{Y}} \beta(y, i, j, k) \cdot p(y | x) \cdot \beta(x, k) \]
Computing (Max-)Marginals

\[ \beta(x_i, y_j) = \max_{x_k} \sum_{y_k} \beta(x_i, y_j, k) \]

\[ \beta(x_i, y_j, k) \]

Diagram:

- Node 0
- Node i
- Node j
- Node n
- Node \( \text{NP} \)
- Node \( \text{Root} \)
- Edges connecting nodes

X, Y, Z, Z, j, i, k
Inside and Outside Scores

\[ \alpha(x, i, j) = \sum_{\lambda \rightarrow A, \delta \rightarrow B \lambda} \rho(A \rightarrow Bx). \]

\[ \beta(B, k, i) \]

\[ \alpha(A, k, j) \]
Pruning with A*

- You can also speed up the search without sacrificing optimality

- For agenda-based parsers:
  - Can select which items to process first
  - Can do with any “figure of merit” [Charniak 98]
  - If your figure-of-merit is a valid A* heuristic, no loss of optimiality [Klein and Manning 03]
Efficient Parsing for Lexical Grammars
Lexicalized Trees

- Add "head words" 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

```
NP:  
  |  
VP:  
  |  
```

```tree
S
  |  
NP  VP
  |  
DT NN  Vt
  |  |  
the lawyer questioned
  |  
NP NP
  |  |  
PT NN  Vt
  |  |  
NP NP
  |  |  
DT NN  Vt
  |  |  
DX the lawyer questioned
```
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

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

- 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 CKY**

\[
\text{bestScore}(X,i,j,h) \\
\text{if (j = i+1)} \\
\quad \text{return tagScore}(X,s[i]) \\
\text{else} \\
\quad \text{return max max} \\
\quad \quad \text{score}(X[h]->Y[h] \ Z[h']) \ast \\
\quad \quad \text{bestScore}(Y,i,k,h) \ast \\
\quad \quad \text{bestScore}(Z,k,j,h') \\
\quad \text{max max} \\
\quad \quad \text{score}(X[h]->Y[h'] \ Z[h]) \ast \\
\quad \quad \text{bestScore}(Y,i,k,h') \ast \\
\quad \quad \text{bestScore}(Z,k,j,h)
\]
Quartic Parsing

- Turns out, you can do (a little) better [Eisner 99]

- Gives an $O(n^4)$ algorithm
- Still prohibitive in practice if not pruned
The Collins parser prunes with per-cell beams [Collins 99]

- Essentially, run the $O(n^5)$ CKY
- Remember only a few hypotheses for each span $<i,j>$.
- If we keep $K$ hypotheses at each span, then we do at most $O(nK^2)$ work per span (why?)
- Keeps things more or less cubic (and in practice is more like linear!)

Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)
Pruning with a PCFG

- The Charniak parser prunes using a two-pass, coarse-to-fine approach [Charniak 97+]
  - First, parse with the base grammar
  - For each \( X: [i,j] \) calculate \( P(X | i, j, s) \)
    - This isn’t trivial, and there are clever speed ups
  - Second, do the full \( O(n^5) \) CKY
    - Skip any \( X : [i,j] \) which had low (say, < 0.0001) posterior
    - Avoids almost all work in the second phase!

- Charniak et al 06: can use more passes
- Petrov et al 07: can use many more passes
Results

- Some results
  - Collins 99 – 88.6 F1 (generative lexical)
  - Charniak and Johnson 05 – 89.7 / 91.3 F1 (generative lexical / reranked)
  - Petrov et al 06 – 90.7 F1 (generative unlexical)
  - McClosky et al 06 – 92.1 F1 (gen + rerank + self-train)
Latent Variable PCFGs
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Parent annotation [Johnson ’98]
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Parent annotation [Johnson ‘98]
- Head lexicalization [Collins ‘99, Charniak ‘00]
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Parent annotation [Johnson ‘98]
- Head lexicalization [Collins ‘99, Charniak ‘00]
- Automatic clustering?
Latent Variable Grammars

Parse Tree

Sentence $T$

Derivations $t : T$

Parameters $\theta$

Grammar $G$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP_0 \ VP_0$</td>
<td>?</td>
</tr>
<tr>
<td>$S \rightarrow NP_1 \ VP_0$</td>
<td>?</td>
</tr>
<tr>
<td>$S \rightarrow NP_0 \ VP_1$</td>
<td>?</td>
</tr>
<tr>
<td>$S \rightarrow NP_1 \ VP_1$</td>
<td>?</td>
</tr>
<tr>
<td>$S_1 \rightarrow NP_0 \ VP_0$</td>
<td>?</td>
</tr>
</tbody>
</table>

Lexicon

<table>
<thead>
<tr>
<th>Rule</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRP_0 → She</td>
<td>?</td>
</tr>
<tr>
<td>PRP_1 → She</td>
<td>?</td>
</tr>
<tr>
<td>VBD_0 → was</td>
<td>?</td>
</tr>
<tr>
<td>VBD_1 → was</td>
<td>?</td>
</tr>
<tr>
<td>VBD_2 → was</td>
<td>?</td>
</tr>
</tbody>
</table>

He was right

He was right

He was right
Learning Latent Annotations

EM algorithm:
- Brackets are known
- Base categories are known
- Only induce subcategories

He was right.

Brackets are known
Base categories are known
Only induce subcategories

Just like Forward-Backward for HMMs.
Refinement of the DT tag

DT

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

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

the (0.80)
The (0.15)
a (0.01)

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

some (0.20)
all (0.19)
those (0.12)

DT-1 DT-2 DT-3 DT-4
Hierarchical refinement

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

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

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

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

- the (0.80)
  The (0.15)
  a (0.01)

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

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

- **Total Number of grammar symbols**
- **Parsing accuracy (F1)**

<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 Training</td>
<td>88.4</td>
</tr>
</tbody>
</table>
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>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 Subcategories
Number of Lexical 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>
## Final Results (Accuracy)

<table>
<thead>
<tr>
<th>Language</th>
<th>Method</th>
<th>≤ 40 words F1</th>
<th>all F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG</td>
<td>Charniak &amp; Johnson ‘05 (generative)</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>90.6</td>
<td>90.1</td>
</tr>
<tr>
<td>GER</td>
<td>Dubey ‘05</td>
<td>76.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>80.8</td>
<td>80.1</td>
</tr>
<tr>
<td>CHN</td>
<td>Chiang et al. ‘02</td>
<td>80.0</td>
<td>76.6</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>86.3</td>
<td>83.4</td>
</tr>
</tbody>
</table>

Still higher numbers from reranking / self-training methods
Efficient Parsing for Hierarchical Grammars
Coarse-to-Fine Inference

- **Example:** PP attachment

```
S  
/  
NP   VP
  /   /  
PRP ??       ??
  /   /   
They ??       ??
   /   /   
  V   NP   PP
     /   /    
    raised DT IN
             /   /
             a    NP
                  /   /
                  of  order
```

- `They` raised a point of order
Hierarchical Pruning

coarse: … QP NP VP …

split in two: … QP1 QP2 NP1 NP2 VP1 VP2 …

split in four: … QP1 QP1 QP3 QP4 NP1 NP2 NP3 NP4 VP1 VP2 VP3 VP4 …

split in eight: … … … … … … … … … … … … … … … … … … … …
Bracket Posteriors
1621 min
111 min
35 min
15 min
(no search error)
Other Syntactactic Models
Lexicalized parsers can be seen as producing dependency trees.

Each local binary tree corresponds to an attachment in the dependency graph.
Pure dependency parsing is only cubic [Eisner 99]

Some work on non-projective dependencies
- Common in, e.g. Czech parsing
- Can do with MST algorithms [McDonald and Pereira 05]
Shift-Reduce Parsers

- Another way to derive a tree:

- Parsing
  - No useful dynamic programming search
  - Can still use beam search [Ratnaparkhi 97]
Tree Insertion Grammars

- Rewrite large (possibly lexicalized) subtrees in a single step

- Formally, a tree-insertion grammar
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable parse is NP-complete
TIG: Insertion

φ

ψ

φ'

S

NP↓

VP

V NP↓

saw

NP

D↓ N

man

S

NP

D↓ N V NP↓

man saw
Tree-adjoining grammars

- Start with *local trees*
- Can insert structure with *adjunction* operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- ... as well as other weird stuff that CFGs don’t capture well (e.g. cross-serial dependencies)
TAG: Long Distance

S
  V  S
   |   
  does NP  VP
     |       
    Bill  S*
     |       
    think

S
  NP(wh)\textsubscript{i}
    |    
  who

S
  NP
    |    
  Harry

S
  NP
    |    
  Bill

S
  V
    |    
  does NP
     |       
    Harry

S
  V
    |    
  think NP
     |       
    Harry

S
  V
    |    
  likes NP
     |       
     ε
CCG Parsing

- Combinatory Categorial Grammar
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus (more later)
  - Can have spurious ambiguities (why?)

$John \vdash NP$

$shares \vdash NP$

$buys \vdash (S\backslash NP)/NP$

$sleeps \vdash S\backslash NP$

$well \vdash (S\backslash NP)\backslash(S\backslash NP)$

```
S
   /\ 
 NP  S\backslash NP
   /   /
 John (S\backslash NP)/NP \ NP
      /   /
     buys shares
```
Empty Elements
Empty Elements

S
  /\   /
 NP  VP
   /\   /
 NN  NNS  VBD  NP
 Housing  lobbies  persuaded
   /\   /
 NP  NNP
  /\   /
 NP  Congress
     /
     TO
     /\   /
     to  to
     /
     VP
     /\   /
 VB  NP
    /\   /
 raise  PP
   /
   to $124,875
   /
   /
   DT  NN
    /
    the  ceiling
In the PTB, three kinds of empty elements:

- Null items (usually complementizers)
- Dislocation (WH-traces, topicalization, relative clause and heavy NP extraposition)
- Control (raising, passives, control, shared argumentation)

Need to reconstruct these (and resolve any indexation)
Example: German

```
S
  |   VAFIN
  |    *T2*
  |     PP
  |      VP
  |       S,
  |        VP-1
AP-2
  |   ADV
  |     NP
  |      ADJD
  |       *will*
  |        PP
  |         NP
  |          VZ
  |           .
Erst not until lange Zeit
ADJA NN später long time
  |       PROAV *begonnen*
  |        ART
  |         NE
  |          PTKZU
  |           VVINF
damit with it den RMV zu schaffen
  |           the RMV to form
```
Types of Empties

<table>
<thead>
<tr>
<th>Antecedent</th>
<th>POS</th>
<th>Label</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>NP</td>
<td>*</td>
<td>18,334</td>
<td>NP trace (e.g., Sam was seen *)</td>
</tr>
<tr>
<td>NP</td>
<td>*</td>
<td></td>
<td>9,812</td>
<td>NP PRO (e.g., * to sleep is nice)</td>
</tr>
<tr>
<td>WHNP</td>
<td>NP</td>
<td><em>T</em></td>
<td>8,620</td>
<td>WH trace (e.g., the woman who you saw <em>T</em>)</td>
</tr>
<tr>
<td>WHNP</td>
<td><em>U</em></td>
<td></td>
<td>7,478</td>
<td>Empty units (e.g., $ 25 <em>U</em>)</td>
</tr>
<tr>
<td>WHNP</td>
<td>0</td>
<td></td>
<td>5,635</td>
<td>Empty complementizers (e.g., Sam said 0 Sasha snores)</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td><em>T</em></td>
<td>4,063</td>
<td>Moved clauses (e.g., Sam had to go, Sasha explained <em>T</em>)</td>
</tr>
<tr>
<td>WHADVP</td>
<td>ADVP</td>
<td><em>T</em></td>
<td>2,492</td>
<td>WH trace (e.g., Sam explained how to leave <em>T</em>)</td>
</tr>
<tr>
<td>WHADVP</td>
<td>SBAR</td>
<td></td>
<td>2,033</td>
<td>Empty clauses (e.g., Sam had to go, Sasha explained (SBAR))</td>
</tr>
<tr>
<td>WHADVP</td>
<td>WHNP</td>
<td>0</td>
<td>1,759</td>
<td>Empty relative pronouns (e.g., the woman 0 we saw)</td>
</tr>
<tr>
<td>WHADVP</td>
<td>WHADVP</td>
<td>0</td>
<td>575</td>
<td>Empty relative pronouns (e.g., no reason 0 to leave)</td>
</tr>
</tbody>
</table>
A Pattern-Matching Approach

- [Johnson 02]
Pattern-Matching Details

- Something like transformation-based learning
- Extract patterns
  - Details: transitive verb marking, auxiliaries
  - Details: legal subtrees
- Rank patterns
  - Pruning ranking: by correct / match rate
  - Application priority: by depth
- Pre-order traversal
- Greedy match
<table>
<thead>
<tr>
<th>Count</th>
<th>Match</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>5816</td>
<td>6223</td>
<td>(S (NP (¬NONE- *) VP)</td>
</tr>
<tr>
<td>5605</td>
<td>7895</td>
<td>(SBAR (¬NONE- 0) S)</td>
</tr>
<tr>
<td>5312</td>
<td>5338</td>
<td>(SBAR WHNP-1 (S (NP (¬NONE- <em>T</em>-1)) VP))</td>
</tr>
<tr>
<td>4434</td>
<td>5217</td>
<td>(NP QP (¬NONE- <em>U</em>))</td>
</tr>
<tr>
<td>1682</td>
<td>1682</td>
<td>(NP $ CD (¬NONE- <em>U</em>))</td>
</tr>
<tr>
<td>1327</td>
<td>1593</td>
<td>(VP VBN_t (NP (¬NONE- *) PP)</td>
</tr>
<tr>
<td>700</td>
<td>700</td>
<td>(ADJP QP (¬NONE- <em>U</em>))</td>
</tr>
<tr>
<td>662</td>
<td>1219</td>
<td>(SBAR (WHNP-1 (¬NONE- 0)) (S (NP (¬NONE- <em>T</em>-1)) VP))</td>
</tr>
<tr>
<td>618</td>
<td>635</td>
<td>(S S-1 , NP (VP VBD (SBAR (¬NONE- 0) (S (¬NONE- <em>T</em>-1))))) .)</td>
</tr>
<tr>
<td>499</td>
<td>512</td>
<td>(SINV <code> </code> S-1 , <code> </code> (VP VBZ (S (¬NONE- <em>T</em>-1))) NP .)</td>
</tr>
<tr>
<td>361</td>
<td>369</td>
<td>(SINV <code> </code> S-1 , <code> </code> (VP VBZ (S (¬NONE- <em>T</em>-1))) NP .)</td>
</tr>
<tr>
<td>352</td>
<td>320</td>
<td>(S NP-1 (VP VBZ (S (NP (¬NONE- *-1)) VP)))</td>
</tr>
<tr>
<td>346</td>
<td>273</td>
<td>(S NP-1 (VP AUX (VP VBN_t (NP (¬NONE- *-1)) PP)))</td>
</tr>
<tr>
<td>322</td>
<td>467</td>
<td>(VP VBD_t (NP (¬NONE- *)) PP)</td>
</tr>
<tr>
<td>269</td>
<td>275</td>
<td>(S <code> </code> S-1 , <code> </code> NP (VP VBD (S (¬NONE- <em>T</em>-1))) .)</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Empty node POS Label</th>
<th>Section 23 $P$</th>
<th>$R$</th>
<th>$f$</th>
<th>Parser output $P$</th>
<th>$R$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Overall)</td>
<td>0.93</td>
<td>0.83</td>
<td>0.88</td>
<td>0.85</td>
<td>0.74</td>
<td>0.79</td>
</tr>
<tr>
<td>NP *</td>
<td>0.95</td>
<td>0.87</td>
<td>0.91</td>
<td>0.86</td>
<td>0.79</td>
<td>0.82</td>
</tr>
<tr>
<td>NP <em>T</em></td>
<td>0.93</td>
<td>0.88</td>
<td>0.91</td>
<td>0.85</td>
<td>0.77</td>
<td>0.81</td>
</tr>
<tr>
<td>NP 0</td>
<td>0.94</td>
<td>0.99</td>
<td>0.96</td>
<td>0.86</td>
<td>0.89</td>
<td>0.88</td>
</tr>
<tr>
<td>NP <em>U</em></td>
<td>0.92</td>
<td>0.98</td>
<td>0.95</td>
<td>0.87</td>
<td>0.96</td>
<td>0.92</td>
</tr>
<tr>
<td>S <em>T</em></td>
<td>0.98</td>
<td>0.83</td>
<td>0.90</td>
<td>0.97</td>
<td>0.81</td>
<td>0.88</td>
</tr>
<tr>
<td>ADVP <em>T</em></td>
<td>0.91</td>
<td>0.52</td>
<td>0.66</td>
<td>0.84</td>
<td>0.42</td>
<td>0.56</td>
</tr>
<tr>
<td>SBAR</td>
<td>0.90</td>
<td>0.63</td>
<td>0.74</td>
<td>0.88</td>
<td>0.58</td>
<td>0.70</td>
</tr>
<tr>
<td>WHNP 0</td>
<td>0.75</td>
<td>0.79</td>
<td>0.77</td>
<td>0.48</td>
<td>0.46</td>
<td>0.47</td>
</tr>
</tbody>
</table>
Semantic Roles
Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:

  \[ \text{Judge} \text{ She } \text{blames} [\text{Eval} \text{uee} \text{ the Government}] [\text{Reason} \text{ for failing to do enough to help}] \].

  Holman would characterise this as \text{blaming} [\text{Eval} \text{uee} \text{ the poor}] .

  The letter quotes Black as saying that \[ \text{Judge} \text{ white and Navajo ranchers} \text{ misrepresent their livestock losses and blame} [\text{Reason} \text{ everything}] [\text{Eval} \text{uee} \text{ on coyotes}] .\]

- Says more than which NP is the subject (but not much more):
- Relations like \text{subject} are syntactic, relations like \text{agent} or \text{message} are semantic
- Typical pipeline:
  - Parse, then label roles
  - Almost all errors locked in by parser
  - Really, SRL is quite a lot easier than parsing
He heard the sound of liquid slurping in a metal container as Farrell approached him from behind.
- **FrameNet**: roles shared between verbs
- **PropBank**: each verb has its own roles
- **PropBank** more used, because it’s layered over the treebank (and so has greater coverage, plus parses)
- **Note**: some linguistic theories postulate fewer roles than FrameNet (e.g. 5-20 total: agent, patient, instrument, etc.)
PropBank Example

fall.01 sense: move downward
roles: Arg1: thing falling
Arg2: extent, distance fallen
Arg3: start point
Arg4: end point

Sales fell to $251.2 million from $278.7 million.
arg1: Sales
rel: fell
arg4: to $251.2 million
arg3: from $278.7 million
rotate.02  sense: shift from one thing to another
roles:  Arg0: causer of shift
Arg1: thing being changed
Arg2: old thing
Arg3: new thing

Many of Wednesday’s winners were losers yesterday as investors quickly took profits and rotated their buying to other issues, traders said.  
(ysj_1723)
arg0: investors
rel: rotated
arg1: their buying
arg3: to other issues
PropBank Example

aim.01 sense: intend, plan
roles: Arg0: aimer, planner
      Arg1: plan, intent

The Central Council of Church Bell Ringers aims *trace* to improve relations with vicars. (wsj_0089)
arg0: The Central Council of Church Bell Ringers
rel: aims
arg1: *trace* to improve relations with vicars

aim.02 sense: point (weapon) at
roles: Arg0: aimer
      Arg1: weapon, etc.
      Arg2: target

Banks have been aiming packages at the elderly.
arg0: Banks
rel: aiming
arg1: packages
arg2: at the elderly
Shared Arguments

(NP-SBJ (JJ massive) (JJ internal) (NN debt) )
  (VP (VBZ has)
    (VP (VBN forced)
      (S
        (NP-SBJ-1 (DT the) (NN government) )
      (VP
        (VP (TO to)
          (VP (VB borrow)
            (ADVP-MNR (RB massively) )...)
Path Features

Path | Description
---|---
VB↑VP↓PP | PP argument/adjunct
VB↑VP↑S↓NP | subject
VB↑VP↓NP | object
VB↑VP↑VP↑S↓NP | subject (embedded VP)
VB↑VP↓ADVP | adverbal adjunct
NN↑NP↑NP↓PP | prepositional complement of noun
Results

- **Features:**
  - Path from target to filler
  - Filler’s syntactic type, headword, case
  - Target’s identity
  - Sentence voice, etc.
  - Lots of other second-order features

- **Gold vs parsed source trees**
  - SRL is fairly easy on gold trees
  - Harder on automatic parses

<table>
<thead>
<tr>
<th></th>
<th>Core</th>
<th></th>
<th>Argm</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>Acc.</td>
<td>F1</td>
<td>Acc.</td>
</tr>
<tr>
<td></td>
<td>92.2</td>
<td>80.7</td>
<td>89.9</td>
<td>71.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Core</th>
<th></th>
<th>Argm</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>Acc.</td>
<td>F1</td>
<td>Acc.</td>
</tr>
<tr>
<td></td>
<td>84.1</td>
<td>66.5</td>
<td>81.4</td>
<td>55.6</td>
</tr>
</tbody>
</table>
**Parse Reranking**

- Assume the number of parses is very small
- We can represent each parse $T$ as a feature vector $\varphi(T)$
  - Typically, all local rules are features
  - Also non-local features, like how right-branching the overall tree is
  - [Charniak and Johnson 05] gives a rich set of features
K-Best Parsing

\[ \gamma = \gamma_0 + r \]