Algorithms for NLP

Parsing IV

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)
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?
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 an 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!

```
I like to parse empties
```
UCS / A*

- 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 optimiality [Klein and Manning 03]
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  VP
    PRP VBD ADJP
    He was 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>
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
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

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

Solution: Mark unary rewrite sites with -U
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
## 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.
Efficient Parsing for Structural Annotation
Grammar Projections

Coarse Grammar

NP
  PRP VBD ADJP
   |   |   
   He was right

NP → DT N'

Fine Grammar

NP^S
  PRP VBD ADVP^VP
   |   |   
   He was right

NP^S → DT^NP N'[...DT]^NP

Note: X-Bar Grammars are projections with rules like XP → Y X' or XP → X' Y or X' → X
Coarse-to-Fine Pruning

For each coarse 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:
Computing (Max-)Marginals

\[ \beta(x, i, j) = \sum_k \beta(x, i, k) \cdot \beta(y_k, i, j) \]
Inside and Outside Scores

\[ \alpha(X, i, j) = \sum_{k \neq k'} \beta(A, k, j) \]

\[ \beta(B, k, i) \]
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 optimality [Klein and Manning 03]
### A* Parsing

<table>
<thead>
<tr>
<th>Estimate</th>
<th>SX</th>
<th>SXL</th>
<th>SXLRL</th>
<th>TRUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>(1,6,NP)</td>
<td>(1,6,NP,VBZ)</td>
<td>(1,6,NP,VBZ,”,”)</td>
<td>(entire context)</td>
</tr>
<tr>
<td>Best Tree</td>
<td><img src="image1.png" alt="Tree" /></td>
<td><img src="image2.png" alt="Tree" /></td>
<td><img src="image3.png" alt="Tree" /></td>
<td><img src="image4.png" alt="Tree" /></td>
</tr>
<tr>
<td>Score</td>
<td>-11.3</td>
<td>-13.9</td>
<td>-15.1</td>
<td>-18.1</td>
</tr>
</tbody>
</table>
Lexicalization
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Structural annotation [Johnson ’98, Klein and Manning 03]
  - 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
What’s different between basic PCFG scores here?
What (lexical) correlations need to be scored?
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
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like
  \[
  \text{VP(saw)} \rightarrow \text{VBD(saw)} \text{ NP-C(her) NP(today)}
  \]

- 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

\[
\text{bestScore}(X,i,j,h) \\
\quad \text{if } (j = i+1) \\
\quad \quad \text{return } \text{tagScore}(X,s[i]) \\
\text{else} \\
\quad \text{return} \\
\quad \quad \max_{k,h',X \rightarrow YZ} \text{score}(X[h] \rightarrow Y[h] Z[h']) \times \\
\quad \quad \text{bestScore}(Y,i,k,h) \times \\
\quad \quad \text{bestScore}(Z,k,j,h') \\
\quad \quad \max_{k,h',X \rightarrow YZ} \text{score}(X[h] \rightarrow Y[h'] Z[h]) \times \\
\quad \quad \text{bestScore}(Y,i,k,h') \times \\
\quad \quad \text{bestScore}(Z,k,j,h)
\]
Efficient Parsing for Lexical Grammars
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
Pruning with Beams

- 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)

- **However**
  - Bilexical counts rarely make a difference (why?)
  - Gildea 01 – Removing bilexical counts costs < 0.5 F1
Latent Variable PCFGs
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 $T$

Sentence $w$

Derivations $t : T$

Parameters $\theta$

Grammar $G$

$S_0 \rightarrow NP_0 VP_0$
$S_0 \rightarrow NP_1 VP_0$
$S_0 \rightarrow NP_0 VP_1$
$S_0 \rightarrow NP_1 VP_1$
$S_1 \rightarrow NP_0 VP_0$
$S_1 \rightarrow NP_1 VP_1$

Lexicon

$PRP_0 \rightarrow She$
$PRP_1 \rightarrow She$
$VBD_0 \rightarrow was$
$VBD_1 \rightarrow was$
$VBD_2 \rightarrow was$
Learning Latent Annotations

EM algorithm:
- 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
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 Training</td>
<td>88.4</td>
</tr>
</tbody>
</table>
Refinement of the , tag

- Splitting all categories equally is wasteful:

```
  , (1.00)
  \-----\------\------
  , (1.00)   , (1.00)   , (1.00)
  \-----\------\------
  , (1.00)   , (1.00)   , (1.00)
```


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
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 Description</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
    PRP
      They
  VP
    ??????????
  V
    raised
  NP
    DT
      a
    NN
      point
  PP
    IN
      of
    NP
      order
```
Hierarchical Pruning

coarse: ...

split in two: ...

split in four: ...

split in eight: ...

QP NP VP...
Bracket Posteriors
1621 min

111 min

35 min

15 min

(no search error)
Unsupervised Tagging
Unsupervised Tagging?

- AKA part-of-speech induction

**Task:**

- Raw sentences in
- Tagged sentences out

**Obvious thing to do:**

- Start with a (mostly) uniform HMM
- Run EM
- Inspect results
EM for HMMs: Process

- Alternate between recomputing distributions over hidden variables (the tags) and reestimating parameters.
- Crucial step: we want to tally up how many (fractional) counts of each kind of transition and emission we have under current params:

\[
\text{count}(w, s) = \sum_{i: w_i = w} P(t_i = s | w)
\]

\[
\text{count}(s \rightarrow s') = \sum_i P(t_{i-1} = s, t_i = s' | w)
\]

- Same quantities we needed to train a CRF!
Some (discouraging) experiments [Merialdo 94]

Setup:
- You know the set of allowable tags for each word
- Fix $k$ training examples to their true labels
  - Learn $P(w|t)$ on these examples
  - Learn $P(t|t_{-1},t_{-2})$ on these examples
- On $n$ examples, re-estimate with EM

Note: we know allowed tags but not frequencies
## Merialdo: Results

<table>
<thead>
<tr>
<th>Iter</th>
<th>Correct tags (% words) after ML on 1M words</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>77.0</td>
</tr>
<tr>
<td>1</td>
<td>80.5</td>
</tr>
<tr>
<td>2</td>
<td>81.8</td>
</tr>
<tr>
<td>3</td>
<td>83.0</td>
</tr>
<tr>
<td>4</td>
<td>84.0</td>
</tr>
<tr>
<td>5</td>
<td>84.8</td>
</tr>
<tr>
<td>6</td>
<td>85.3</td>
</tr>
<tr>
<td>7</td>
<td>85.8</td>
</tr>
<tr>
<td>8</td>
<td>86.1</td>
</tr>
<tr>
<td>9</td>
<td>86.3</td>
</tr>
<tr>
<td>10</td>
<td>86.6</td>
</tr>
</tbody>
</table>