Algorithms for NLP

Parsing III

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley
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!
EM for HMMs: Quantities

- Total path values (correspond to probabilities here):

\[
\alpha_i(s) = P(w_0 \ldots w_i, s_i) = \sum_{s_{i-1}} P(s_i|s_{i-1})P(w_i|s_i)\alpha_{i-1}(s_{i-1})
\]

\[
\beta_i(s) = P(w_i + 1 \ldots w_n|s_i) = \sum_{s_{i+1}} P(s_{i+1}|s_i)P(w_{i+1}|s_{i+1})\beta_{i+1}(s_{i+1})
\]
The State Lattice / Trellis

START       Fed           raises       interest         rates         END
EM for HMMs: Process

- From these quantities, can compute expected transitions:

$$\text{count}(s \rightarrow s') = \frac{\sum_i \alpha_i(s) P(s' | s) P(w_i | s) \beta_{i+1}(s')}{P(w)}$$

- And emissions:

$$\text{count}(w, s) = \frac{\sum_{i: w_i = w} \alpha_i(s) \beta_{i+1}(s)}{P(w)}$$
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>0</th>
<th>100</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
<th>20000</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>77.0</td>
<td>90.0</td>
<td>95.4</td>
<td>96.2</td>
<td>96.6</td>
<td>96.9</td>
<td>97.0</td>
</tr>
<tr>
<td>1</td>
<td>80.5</td>
<td>92.6</td>
<td>95.8</td>
<td>96.3</td>
<td>96.6</td>
<td>96.7</td>
<td>96.8</td>
</tr>
<tr>
<td>2</td>
<td>81.8</td>
<td>93.0</td>
<td>95.7</td>
<td>96.1</td>
<td>96.3</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>3</td>
<td>83.0</td>
<td>93.1</td>
<td>95.4</td>
<td>95.8</td>
<td>96.1</td>
<td>96.2</td>
<td>96.2</td>
</tr>
<tr>
<td>4</td>
<td>84.0</td>
<td>93.0</td>
<td>95.2</td>
<td>95.5</td>
<td>95.8</td>
<td>96.0</td>
<td>96.0</td>
</tr>
<tr>
<td>5</td>
<td>84.8</td>
<td>92.9</td>
<td>95.1</td>
<td>95.4</td>
<td>95.6</td>
<td>95.8</td>
<td>95.8</td>
</tr>
<tr>
<td>6</td>
<td>85.3</td>
<td>92.8</td>
<td>94.9</td>
<td>95.2</td>
<td>95.5</td>
<td>95.6</td>
<td>95.7</td>
</tr>
<tr>
<td>7</td>
<td>85.8</td>
<td>92.8</td>
<td>94.7</td>
<td>95.1</td>
<td>95.3</td>
<td>95.5</td>
<td>95.5</td>
</tr>
<tr>
<td>8</td>
<td>86.1</td>
<td>92.7</td>
<td>94.6</td>
<td>95.0</td>
<td>95.2</td>
<td>95.4</td>
<td>95.4</td>
</tr>
<tr>
<td>9</td>
<td>86.3</td>
<td>92.6</td>
<td>94.5</td>
<td>94.9</td>
<td>95.1</td>
<td>95.3</td>
<td>95.3</td>
</tr>
<tr>
<td>10</td>
<td>86.6</td>
<td>92.6</td>
<td>94.4</td>
<td>94.8</td>
<td>95.0</td>
<td>95.2</td>
<td>95.2</td>
</tr>
</tbody>
</table>
the president said that the downturn was over

| president | the __ of |
| president | the __ said |
| governor  | the __ of |
| governor  | the __ appointed |
| said      | sources __ |
| said      | president __ that |
| reported  | sources __ |

[Finch and Chater 92, Shuetze 93, many others]
Distributional Clustering

- Three main variants on the same idea:
  - Pairwise similarities and heuristic clustering
    - E.g. [Finch and Chater 92]
    - Produces dendrograms
  - Vector space methods
    - E.g. [Shuetze 93]
    - Models of ambiguity
  - Probabilistic methods
    - Various formulations, e.g. [Lee and Pereira 99]
<table>
<thead>
<tr>
<th>word</th>
<th>nearest neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>accompanied</td>
<td>submitted  banned  financed  developed  authorized  headed  canceled  awarded  barred</td>
</tr>
<tr>
<td>almost</td>
<td>virtually  merely  formally  fully  quite  officially  just  nearly  only  less</td>
</tr>
<tr>
<td>causing</td>
<td>reflecting  forcing  providing  creating  producing  becoming  carrying  particularly</td>
</tr>
<tr>
<td>classes</td>
<td>elections  courses  payments  losses  computers  performances  violations  levels  pictures</td>
</tr>
<tr>
<td>directors</td>
<td>professionals  investigations  materials  competitors  agreements  papers  transactions</td>
</tr>
<tr>
<td>goal</td>
<td>mood  roof  eye  image  tool  song  pool  scene  gap  voice</td>
</tr>
<tr>
<td>japanese</td>
<td>chinese  iraqi  american  western  arab  foreign  european  federal  soviet  indian</td>
</tr>
<tr>
<td>represent</td>
<td>reveal  attend  deliver  reflect  choose  contain  impose  manage  establish  retain</td>
</tr>
<tr>
<td>think</td>
<td>believe  wish  know  realize  wonder  assume  feel  say  mean  bet</td>
</tr>
<tr>
<td>york</td>
<td>angeles  francisco  sox  rouge  kong  diego  zone  vegas  inning  layer</td>
</tr>
<tr>
<td>on</td>
<td>through  in  at  over  into  with  from  for  by  across</td>
</tr>
<tr>
<td>must</td>
<td>might  would  could  cannot  will  should  can  may  does  helps</td>
</tr>
<tr>
<td>they</td>
<td>we  you  i  he  she  nobody  who  it  everybody  there</td>
</tr>
</tbody>
</table>
Dendrograms
Dendrograms
Vector Space Version

- [Shuetze 93] clusters words as points in $\mathbb{R}^n$

- Vectors too sparse, use SVD to reduce

Cluster these 50-200 dim vectors instead.
A Probabilistic Version?

\[ P(S, C) = \prod_i P(c_i) P(w_i \mid c_i) P(w_{i-1}, w_{i+1} \mid c_i) . \]

\[ \text{the president said that the downturn was over} \]

\[ \text{the president said that the downturn was over} \]
Various newer ideas:

- Context distributional clustering [Clark 00]
- Morphology-driven models [Clark 03]
- Contrastive estimation [Smith and Eisner 05]
- Feature-rich induction [Haghighi and Klein 06]

Also:

- What about ambiguous words?
- Using wider context signatures has been used for learning synonyms (what’s wrong with this approach?)
- Can extend these ideas for grammar induction (later)
Computing Marginals

\[ P(s_t | x) = \frac{P(s_t, x)}{P(x)} \]

\[ = \text{sum of all paths through } s \text{ at } t \]

\[ \text{sum of all paths} \]
Forward Scores

\[ v_t(s_t) = \max_{s_{t-1}} v_{t-1}(s_{t-1}) \phi_t(s_{t-1}, s_t) \]

\[ \alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \phi_t(s_{t-1}, s_t) \]
Backward Scores

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \phi_t(s_t, s_{t+1})$$
Total Scores

\[ P(s_t, x) = \alpha_t(s_t)\beta_t(s_t) \]

\[ P(x) = \sum_{s_t} \alpha_t(s_t)\beta_t(s_t) \]

\[ = \alpha_T(\text{stop}) \]

\[ = \beta_0(\text{start}) \]
Syntax
The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market.
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...

new art critics write reviews with computers
Constituency Tests

- How do we know what nodes go in the tree?

- Classic constituency tests:
  - Substitution by *proform*
  - Question answers
  - Semantic grounds
    - Coherence
    - Reference
    - Idioms
  - Dislocation
  - Conjunction

- Cross-linguistic arguments, too
Conflicting Tests

- Constituency isn’t always clear
  - Units of transfer:
    - think about ~ penser à
    - talk about ~ hablar de
  - Phonological reduction:
    - I will go → I’ll go
    - I want to go → I wanna go
    - a le centre → au centre
  - Coordination
    - He went to and came from the store.
Classical NLP: Parsing

- Write symbolic or logical rules:

  Grammar (CFG)  
  
<table>
<thead>
<tr>
<th>NP</th>
<th>NP PP</th>
<th>NN</th>
<th>interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP</td>
<td>VBP NP</td>
<td>NNS</td>
<td>raises</td>
</tr>
<tr>
<td>VP</td>
<td>VBP NP PP</td>
<td>VBP</td>
<td>interest</td>
</tr>
<tr>
<td>PP</td>
<td>IN NP</td>
<td>VBZ</td>
<td>raises</td>
</tr>
</tbody>
</table>

- Use deduction systems to prove parses from words
  - Minimal grammar on sentence: 36 parses
  - Simple 10-rule grammar: 592 parses
  - Real-size grammar: many millions of parses

- This scaled very badly, didn’t yield broad-coverage tools
Ambiguities
The board approved [its acquisition] [by Royal Trustco Ltd.] [of Toronto] [for $27 a share] [at its monthly meeting].
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
Prepositional phrases:  
They cooked the beans in the pot on the stove with handles.

Particle vs. preposition:  
The puppy tore up the staircase.

Complement structures  
The tourists objected to the guide that they couldn’t hear.  
She knows you like the back of her hand.

Gerund vs. participial adjective  
Visiting relatives can be boring.  
Changing schedules frequently confused passengers.
Modifier scope within NPs
*impractical design requirements*
*plastic cup holder*

Multiple gap constructions
*The chicken is ready to eat.*
*The contractors are rich enough to sue.*

Coordination scope:
*Small rats and mice can squeeze into holes or cracks in the wall.*
Dark Ambiguities

- **Dark ambiguities**: most analyses are shockingly bad (meaning, they don’t have an interpretation you can get your mind around)

  This analysis corresponds to the correct parse of
  
  “This will panic buyers ! ”

- Unknown words and new usages
- **Solution**: We need mechanisms to focus attention on the best ones, probabilistic techniques do this
Ambiguities as Trees

(a) VP
   |   
   VBG  NP
   ... raising
   NP   PP
   $ 30 billion  from debt ...

(b) NP
   |   
   QP
   PDT  DT  PDT  PDT
   ... half  a  dozen newspapers

(a) S
   |   
   NP  VP
   NP  PP
   Lehman Hutton Inc.  by yesterday afternoon

(c) VP
   |   
   VBZ  ADVP  ADJP  ADJP
   ADJP
   ... 's  RB  just  JJ
   fine
PCFGs
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 \( \text{ROOT} \) or \( \text{TOP} \)
    - \textit{Not} usually the sentence non-terminal \( S \)
  - \( R \): the set of rules
    - Of the form \( X \rightarrow Y_1 Y_2 \ldots Y_k \), 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 \( P(Y_1 Y_2 \ldots Y_k | X) \)
( (S (NP-SBJ The move)
   (VP followed
    (NP (NP a round)
     (PP of
      (NP (NP similar increases)
       (PP by
        (NP other lenders))
       (PP against
        (NP Arizona real estate loans))))))
,
(S-ADV (NP-SBJ *))
(VP reflecting
  (NP (NP a continuing decline)
   (PP-LOC in
    (NP that market))))))
Need a PCFG for broad coverage parsing.

Can take a grammar right off the trees (doesn’t work well):

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get state-of-the-art 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 $X \rightarrow Y Z$ 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!
CKY Parsing
A Recursive Parser

```clojure
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)
```

- Will this parser work?
- Why or why not?
- Memory requirements?
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])
```
Unary Rules

Unary rules?

```
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
bestScoreB\( (X,i,j,s) \)

return max \( \text{max score}(X\rightarrow YZ) \) * 
  \( \text{bestScoreU}(Y,i,k) \) * 
  \( \text{bestScoreU}(Z,k,j) \)

bestScoreU\( (X,i,j,s) \)

if (\( j = i+1 \))
  return tagScore\( (X,s[i]) \)
else
  return max \( \text{max score}(X\rightarrow Y) \) * 
    \( \text{bestScoreB}(Y,i,j) \)
Memory

- **How much memory does this require?**
  - Have to store the score cache
  - Cache size: $|\text{symbols}| \times n^2$ doubles
  - For the plain treebank grammar:
    - $X \sim 20K$, $n = 40$, double $\sim 8$ bytes $\sim 256$MB
    - Big, but workable.

- **Pruning: Beams**
  - $\text{score}[X][i][j]$ can get too large (when?)
  - Can keep beams (truncated maps $\text{score}[i][j]$) which only store the best few scores for the span $[i,j]

- **Pruning: Coarse-to-Fine**
  - Use a smaller grammar to rule out most $X[i,j]$
  - Much more on this later...
How much time will it take to parse?

- For each diff \((\leq n)\)
  - For each \(i\) \((\leq n)\)
    - For each rule \(X \rightarrow YZ\)
      - 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 sentence
Parsing with the vanilla treebank grammar:

Why’s it worse in practice?

- Longer sentences “unlock” more of the grammar
- All kinds of systems issues don’t scale

~ 20K Rules (not an optimized parser!)

Observed exponent: 3.6
Same-Span Reachability

- TOP
- ADJP ADVP
- FRAG INTJ NP
- PP PRN QP S
- SBAR UCP VP
- WHNP
- NX
- SQ
- X
- RRC
- SINV
- WHADJP
- SBARQ
- WHPP
- WHADVP
- LST
- CONJP
- NAC
Many states are more likely to match larger spans!
Efficient CKY

- Lots of tricks to make CKY efficient
  - Some of them are little engineering details:
    - E.g., first choose k, then enumerate through the Y:[i,k] which are non-zero, then loop through rules by left child.
    - Optimal layout of the dynamic program depends on grammar, input, even system details.
  - Another kind is more important (and interesting):
    - Many X[i,j] can be suppressed on the basis of the input string
    - We’ll see this next class as figures-of-merit, A* heuristics, coarse-to-fine, etc
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).

<table>
<thead>
<tr>
<th>AGENDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>critics[0,1], write[1,2], reviews[2,3], with[3,4], computers[4,5]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHART [EMPTY]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  1  2  3  4  5</td>
</tr>
<tr>
<td>critics     write    reviews    with    computers</td>
</tr>
</tbody>
</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

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 $\epsilon[i,i]$
  - Add rules like $NP \rightarrow \epsilon$ to the grammar
  - That’s it!

\[\begin{array}{cccccc}
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}\]
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]
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
  │    │        ├── He
  │    │        └── was
  │    └── 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>
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

<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
       │   └── DT-U^NP
       │       └── This
       │   ├── NP^S-B
       │   │   └── ""S
       │   │       └── VBZ^BE^VP
       │   │           └── is
       │   │       └── ""S
       │   │           └── NP^VP-B
       │   │               └── panic
       │   │                   └── buying
       └── ""S
```
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

Fine Grammar

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

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

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

E.g. consider the span 5 to 12:

coarse: $\ldots$ CP NP VP $\ldots$

refined: 

\[\text{Diagram showing the relationship between coarse and refined chart items.}\]
Computing (Max-)Marginals

\[ \beta(x_i, j) = \sum_{y} \gamma(x_i, y) \beta(y, j). \]

\[ \beta(y, j, k). \]

\[ \beta(z, k, j). \]
Inside and Outside Scores

\[ \alpha(x, i, j) = \sum_{a \rightarrow b \times} \rho(a \rightarrow b^x) \]

\[ \alpha(A, k, j) = \beta(B, k, i) \]
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 optimiality [Klein and Manning 03].
## A* Parsing

<table>
<thead>
<tr>
<th>Estimate</th>
<th>SX</th>
<th>SXL</th>
<th>SXLR</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, &quot;,&quot;)</td>
<td>(entire context)</td>
</tr>
<tr>
<td>Best Tree</td>
<td><img src="tree1.png" alt="Tree Diagram" /></td>
<td><img src="tree2.png" alt="Tree Diagram" /></td>
<td><img src="tree3.png" alt="Tree Diagram" /></td>
<td><img src="tree4.png" alt="Tree Diagram" /></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
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
Problems with PCFGs

- 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

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

- 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
bestScore(X,i,j,h)
  if \( j = i+1 \)
    return \( \text{tagScore}(X, s[i]) \)
  else
    return
      \[
      \max_{k,h',X\rightarrow YZ} \max_{h} \text{score}(X[h] \rightarrow Y[h] \ Z[h']) \times
      \text{bestScore}(Y, i, k, h) \times
      \text{bestScore}(Z, k, j, h')
      \]
      \[
      \max_{k,h',X\rightarrow YZ} \max_{h} \text{score}(X[h] \rightarrow Y[h'] \ Z[h]) \times
      \text{bestScore}(Y, i, k, h') \times
      \text{bestScore}(Z, k, j, h)
      \]
A Recursive Parser

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

```plaintext
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:

```python
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

```
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]
```
 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

\[
\text{bestScoreB}(X, i, j, s) \\
\quad \text{return } \max \max \text{ score}(X \rightarrow YZ) \times \\
\quad \quad \quad \text{bestScoreU}(Y, i, k) \times \\
\quad \quad \quad \text{bestScoreU}(Z, k, j)
\]

\[
\text{bestScoreU}(X, i, j, s) \\
\quad \text{if } (j = i+1) \\
\quad \quad \text{return } \text{tagScore}(X, s[i]) \\
\quad \text{else} \\
\quad \quad \text{return } \max \max \text{ score}(X \rightarrow Y) \times \\
\quad \quad \quad \text{bestScoreB}(Y, i, j)
\]
Analysis
Memory

- **How much memory does this require?**
  - Have to store the score cache
  - Cache size: \(|\text{symbols}| \times n^2\) doubles
  - For the plain treebank grammar:
    - \(X \sim 20K, n = 40, \text{double} \sim 8\text{ bytes} = \sim 256\text{MB}\)
    - Big, but workable.

- **Pruning: Beams**
  - score\([X][i][j]\) can get too large (when?)
  - Can keep beams (truncated maps score\([i][j]\)) which only store the best few scores for the span \([i,j]\)

- **Pruning: Coarse-to-Fine**
  - Use a smaller grammar to rule out most \(X[i,j]\)
  - Much more on this later...
How much time will it take to parse?

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

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

Something like 5 sec for an unoptimized parse of a 20-word sentence
Time: Practice

- Parsing with the vanilla treebank grammar:

  - Why’s it worse in practice?
    - Longer sentences “unlock” more of the grammar
    - All kinds of systems issues don’t scale

- Observed exponent: 3.6

- ~ 20K Rules
  (not an optimized parser!)
Lots of tricks to make CKY efficient

Some of them are little engineering details:

- E.g., first choose \( k \), then enumerate through the \( Y: [i, k] \) which are non-zero, then loop through rules by left child.
- Optimal layout of the dynamic program depends on grammar, input, even system details.

Another kind is more important (and interesting):

- Many \( X[i, j] \) can be suppressed on the basis of the input string
- We’ll see this next class as figures-of-merit, A* heuristics, coarse-to-fine, etc
Binarization

- **Chomsky normal form:**
  - All rules of the form $X \rightarrow Y Z$ 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!
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)

Diagram:

- Nodes represent fenceposts:
  - 0: critics
  - 1: write
  - 2: reviews
  - 3: with
  - 4: computers

- Edge: PP
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]
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 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  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!
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

- 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)
Horizontal Markovization

Order 1

Order $\infty$

Symbols

Horizontal Markov Order

Horizontal Markov Order
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>
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

Fine Grammar

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

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

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

E.g. consider the span 5 to 12:
Computing (Max-)Marginals

\[
\beta(x, j, j) = \frac{1}{\beta(x)} \sum_{y_8, \ldots, y_k} p(y_8|x). \\
\beta(y, j, k). \\
\beta(\theta, k, j)
\]
Inside and Outside Scores

\[ \alpha(x, i, j) = \sum_{x_S, y_B} \rho(A \rightarrow Bx) \cdot \alpha(A, k, j) \]

[Diagram with labels and arrows]
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]
<table>
<thead>
<tr>
<th>Estimate</th>
<th>SX</th>
<th>SXL</th>
<th>SXLR</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>![Diagram SX]</td>
<td>![Diagram SXL]</td>
<td>![Diagram SXLR]</td>
<td>![Diagram TRUE]</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
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.
Problems with PCFGs

- 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
bestScore(X,i,j,h)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return
      \[
      \max_{k,h',X\rightarrow YZ} \max \text{score}(X[h] \rightarrow Y[h] \ Z[h']) \ast \\
      \text{bestScore}(Y,i,k,h) \ast \\
      \text{bestScore}(Z,k,j,h')
      \]
      \[
      \max_{k,h',X\rightarrow YZ} \text{score}(X[h] \rightarrow Y[h'] \ Z[h]) \ast \\
      \text{bestScore}(Y,i,k,h') \ast \\
      \text{bestScore}(Z,k,j,h)
      \]
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
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]
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 $\theta$
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

- 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)
      - some (0.20)
        - all (0.19)
        - those (0.12)
      - that (0.28)
        - That (0.11)
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:
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</th>
<th>further</th>
<th>lower</th>
<th>higher</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBR-0</td>
<td></td>
<td></td>
<td></td>
</tr>
<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</th>
<th>one</th>
<th>two</th>
<th>Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-7</td>
<td>1989</td>
<td>1990</td>
<td>1988</td>
</tr>
<tr>
<td>CD-4</td>
<td></td>
<td></td>
<td></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
    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:
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!
Merialdo: Setup

- 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
EM for HMMs: Quantities

- Total path values (correspond to probabilities here):

\[ \alpha_i(s) = P(w_0 \ldots w_i, s_i) = \sum_{s_{i-1}} P(s_i|s_{i-1})P(w_i|s_i)\alpha_{i-1}(s_{i-1}) \]

\[ \beta_i(s) = P(w_i + 1 \ldots w_n|s_i) = \sum_{s_{i+1}} P(s_{i+1}|s_i)P(w_{i+1}|s_{i+1})\beta_{i+1}(s_{i+1}) \]
The State Lattice / Trellis

START       Fed           raises       interest         rates         END
EM for HMMs: Process

- From these quantities, can compute expected transitions:

\[
\text{count}(s \rightarrow s') = \frac{\sum_i \alpha_i(s) P(s'|s) P(w_i|s) \beta_{i+1}(s')}{P(w)}
\]

- And emissions:

\[
\text{count}(w, s) = \frac{\sum_{i: w_i=w} \alpha_i(s) \beta_{i+1}(s)}{P(w)}
\]
### 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>
Factory payrolls fell in Sept.
A* Speedup

- Total time dominated by calculation of A* tables in each projection... $O(n^3)$
We can relax independence assumptions by encoding dependencies into the PCFG symbols:

- What are the most useful “features” to encode?
Other Tag Splits

- **UNARY-DT**: mark demonstratives as DT^U ("the X" vs. "those")
- **UNARY-RB**: mark phrasal adverbs as RB^U ("quickly" vs. "very")
- **TAG-PA**: mark tags with non-canonical parents ("not" is an RB^VP)
- **SPLIT-AUX**: mark auxiliary verbs with –AUX [cf. Charniak 97]
- **SPLIT-CC**: separate "but" and "&" from other conjunctions
- **SPLIT-%**: "%" gets its own tag.

<table>
<thead>
<tr>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>80.4</td>
<td>8.1K</td>
</tr>
<tr>
<td>80.5</td>
<td>8.1K</td>
</tr>
<tr>
<td>81.2</td>
<td>8.5K</td>
</tr>
<tr>
<td>81.6</td>
<td>9.0K</td>
</tr>
<tr>
<td>81.7</td>
<td>9.1K</td>
</tr>
<tr>
<td>81.8</td>
<td>9.3K</td>
</tr>
</tbody>
</table>