CSEP 517
Natural Language Processing
Autumn 2013

Parsing: PCFGs and Treebank Parsing

Luke Zettlemoyer - University of Washington

[Slides from Dan Klein, Michael Collins, and Ray Mooney]
Topics

- Parse Trees
- (Probabilistic) Context Free Grammars
  - Supervised learning
  - Parsing: most likely tree, marginal distributions
- Treebank Parsing (English, edited text)
The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market.
Table 1.2. The Penn Treebank syntactic tagset

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJP</td>
<td>Adjective phrase</td>
</tr>
<tr>
<td>ADVP</td>
<td>Adverb phrase</td>
</tr>
<tr>
<td>NP</td>
<td>Noun phrase</td>
</tr>
<tr>
<td>PP</td>
<td>Prepositional phrase</td>
</tr>
<tr>
<td>S</td>
<td>Simple declarative clause</td>
</tr>
<tr>
<td>SBAR</td>
<td>Subordinate clause</td>
</tr>
<tr>
<td>SBARQ</td>
<td>Direct question introduced by <em>wh</em>-element</td>
</tr>
<tr>
<td>SINV</td>
<td>Declarative sentence with subject-aux inversion</td>
</tr>
<tr>
<td>SQ</td>
<td>Yes/no questions and subconstituent of SBARQ excluding <em>wh</em>-element</td>
</tr>
<tr>
<td>VP</td>
<td>Verb phrase</td>
</tr>
<tr>
<td>WHADVP</td>
<td>Wh-adverb phrase</td>
</tr>
<tr>
<td>WHNP</td>
<td>Wh-noun phrase</td>
</tr>
<tr>
<td>WHPP</td>
<td>Wh-prepositional phrase</td>
</tr>
<tr>
<td>X</td>
<td>Constituent of unknown or uncertain category</td>
</tr>
<tr>
<td>*</td>
<td>“Understood” subject of infinitive or imperative</td>
</tr>
<tr>
<td>0</td>
<td>Zero variant of <em>that</em> in subordinate clauses</td>
</tr>
<tr>
<td>T</td>
<td>Trace of <em>wh</em>-Constituent</td>
</tr>
</tbody>
</table>
The Penn Treebank: Size

- Penn WSJ Treebank = 50,000 sentences with associated trees
- Usual set-up: 40,000 training sentences, 2400 test sentences

An example tree:

Canadian Utilities had 1988 revenue of C$ 1.16 billion, mainly from its natural gas and electric utility businesses in Alberta, where the company serves about 800,000 customers.
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
    - he, she, it, they, ...
  - Question / answer
  - Deletion
  - Movement / dislocation
  - Conjunction / coordination
- 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.

La vélocité des ondes sismiques
Non-Local Phenomena

- **Dislocation / gapping**
  - Which book should Peter buy?
  - A debate arose which continued until the election.

- **Binding**
  - **Reference**
    - The IRS audits itself
  - **Control**
    - I want to go
    - I want you to go
Classical NLP: Parsing

- **Write symbolic or logical rules:**

<table>
<thead>
<tr>
<th>Grammar (CFG)</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT $\rightarrow$ S</td>
<td>NP $\rightarrow$ NP PP</td>
</tr>
<tr>
<td>S $\rightarrow$ NP VP</td>
<td>VP $\rightarrow$ VBP NP</td>
</tr>
<tr>
<td>NP $\rightarrow$ DT NN</td>
<td>VP $\rightarrow$ VBP NP PP</td>
</tr>
<tr>
<td>NP $\rightarrow$ NN NNS</td>
<td>PP $\rightarrow$ IN NP</td>
</tr>
</tbody>
</table>

- **Use deduction systems to prove parses from words**
  - Minimal grammar on “Fed raises” 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: PP Attachment

The children ate the cake with a spoon.

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
Syntactic Ambiguities I

- **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.
Syntactic Ambiguities II

- 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
A context-free grammar is a tuple \(<N, \Sigma, S, R>\)

- **\(N\)**: the set of non-termsinals
  - Phrasal categories: S, NP, VP, ADJP, etc.
  - Parts-of-speech (pre-termsinals): NN, JJ, DT, VB
- **\(\Sigma\)**: the set of terminals (the words)
- **\(S\)**: the start symbol
  - Often written as ROOT or TOP
  - Not usually the sentence non-terminal S
- **\(R\)**: the set of rules
  - Of the form \(X \rightarrow Y_1 Y_2 \ldots Y_n\), with \(X \in N, n \geq 0, Y_i \in (N \cup \Sigma)\)
  - Examples: \(S \rightarrow NP\ VP\), \(VP \rightarrow VP\ CC\ VP\)
  - Also called rewrites, productions, or local trees
Example Grammar

\[ N = \{S, \text{NP}, \text{VP}, \text{PP}, \text{DT}, \text{Vi}, \text{Vt}, \text{NN}, \text{IN}\} \]
\[ S = S \]
\[ \Sigma = \{\text{sleeps}, \text{saw}, \text{man}, \text{woman}, \text{telescope}, \text{the}, \text{with}, \text{in}\} \]

\[ R = \]

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>NP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP</td>
<td>=&gt;</td>
<td>Vi</td>
<td></td>
</tr>
<tr>
<td>VP</td>
<td>=&gt;</td>
<td>Vt</td>
<td>NP</td>
</tr>
<tr>
<td>VP</td>
<td>=&gt;</td>
<td>VP</td>
<td>PP</td>
</tr>
<tr>
<td>NP</td>
<td>=&gt;</td>
<td>DT</td>
<td>NN</td>
</tr>
<tr>
<td>NP</td>
<td>=&gt;</td>
<td>NP</td>
<td>PP</td>
</tr>
<tr>
<td>PP</td>
<td>=&gt;</td>
<td>IN</td>
<td>NP</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th></th>
<th>Vi</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Vt</td>
<td>=&gt;</td>
<td>saw</td>
</tr>
<tr>
<td>NN</td>
<td>=&gt;</td>
<td>man</td>
</tr>
<tr>
<td>NN</td>
<td>=&gt;</td>
<td>woman</td>
</tr>
<tr>
<td>NN</td>
<td>=&gt;</td>
<td>telescope</td>
</tr>
<tr>
<td>DT</td>
<td>=&gt;</td>
<td>the</td>
</tr>
<tr>
<td>IN</td>
<td>=&gt;</td>
<td>with</td>
</tr>
<tr>
<td>IN</td>
<td>=&gt;</td>
<td>in</td>
</tr>
</tbody>
</table>

S=sentence, VP-verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition
A Context-Free Grammar for English

\[ R = \]

\[
\begin{array}{l}
S \rightarrow NP \ VP \\
VP \rightarrow Vi \\
VP \rightarrow Vt \ NP \\
VP \rightarrow VP \ PP \\
NP \rightarrow DT \ NN \\
NP \rightarrow NP \ PP \\
PP \rightarrow IN \ NP \\
Vi \rightarrow sleeps \\
Vt \rightarrow saw \\
NN \rightarrow man \\
NN \rightarrow woman \\
NN \rightarrow telescope \\
DT \rightarrow the \\
IN \rightarrow with \\
IN \rightarrow in \\
\end{array}
\]

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

Example Parses

The man sleeps

The man saw the woman with the telescope
Probabilistic Context-Free Grammars

- A context-free grammar is a tuple <N, Σ, S, R>
  - N: the set of non-terminals
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB, etc.
  - Σ: the set of terminals (the words)
  - S: the start symbol
    - Often written as ROOT or TOP
    - Not usually the sentence non-terminal S
  - R: the set of rules
    - Of the form \( X \rightarrow Y_1 Y_2 \ldots Y_n \), with \( X \in N, n \geq 0, Y_i \in (N \cup \Sigma) \)
    - Examples: \( S \rightarrow NP \ VP, \ VP \rightarrow VP \ CC \ VP \)
- A PCFG adds a distribution \( q \):
  - Probability \( q(r) \) for each \( r \in R \), such that for all \( X \in N \):
    \[
    \sum_{\alpha \rightarrow \beta \in R: \alpha = X} q(\alpha \rightarrow \beta) = 1
    \]
PCFG Example

<table>
<thead>
<tr>
<th></th>
<th>Production</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>NP VP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP</td>
<td>Vi</td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>Vt NP</td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>VP PP</td>
<td>0.2</td>
</tr>
<tr>
<td>NP</td>
<td>DT NN</td>
<td>0.3</td>
</tr>
<tr>
<td>NP</td>
<td>NP PP</td>
<td>0.7</td>
</tr>
<tr>
<td>PP</td>
<td>P NP</td>
<td>1.0</td>
</tr>
<tr>
<td>Vi</td>
<td>sleeps</td>
<td>1.0</td>
</tr>
<tr>
<td>Vt</td>
<td>saw</td>
<td>1.0</td>
</tr>
<tr>
<td>NN</td>
<td>man</td>
<td>0.7</td>
</tr>
<tr>
<td>NN</td>
<td>woman</td>
<td>0.2</td>
</tr>
<tr>
<td>NN</td>
<td>telescope</td>
<td>0.1</td>
</tr>
<tr>
<td>DT</td>
<td>the</td>
<td>1.0</td>
</tr>
<tr>
<td>IN</td>
<td>with</td>
<td>0.5</td>
</tr>
<tr>
<td>IN</td>
<td>in</td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Probability of a tree $t$ with rules

  \[
  \alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n
  \]

  is

  \[
  p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)
  \]

  where $q(\alpha \rightarrow \beta)$ is the probability for rule $\alpha \rightarrow \beta$. 
PCFG Example

<table>
<thead>
<tr>
<th>Grammar Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP → Vi</td>
<td>0.4</td>
</tr>
<tr>
<td>VP → Vt NP</td>
<td>0.4</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → DT NN</td>
<td>0.3</td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>0.7</td>
</tr>
<tr>
<td>PP → P NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

- Probability of a tree $t$ with rules $\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$ is $p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$, where $q(\alpha_i \rightarrow \beta_i)$ is the probability for rule $\alpha_i \rightarrow \beta_i$.

Example:

Sentence: The man sleeps

Tree $t_1$: $S \rightarrow NP \rightarrow DT NN \rightarrow Vi \rightarrow sleeps \rightarrow VP \rightarrow VP PP \rightarrow P NP$

$p(t_1)=1.0*0.3*1.0*0.7*0.4*1.0$

Sentence: The man saw the woman with the telescope

Tree $t_2$: $S \rightarrow VP \rightarrow NP \rightarrow DT NN \rightarrow Vi \rightarrow saw \rightarrow VP \rightarrow NP \rightarrow DT NN \rightarrow IN \rightarrow with \rightarrow PP \rightarrow NP$

$p(t_2)=1.8*0.3*1.0*0.7*0.2*0.4*1.0*0.3*0.1*0.5*0.3*1.0*0.1$
PCFGs: Learning and Inference

- **Model**
  - The probability of a tree $t$ with $n$ rules $\alpha_i \rightarrow \beta_i$, $i = 1..n$
  
  $$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

- **Learning**
  - Read the rules off of labeled sentences, use ML estimates for probabilities
  
  $$q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

  - and use all of our standard smoothing tricks!

- **Inference**
  - For input sentence $s$, define $T(s)$ to be the set of trees whole *yield* is $s$ (whole leaves, read left to right, match the words in $s$)
  
  $$t^*(s) = \arg \max_{t \in T(s)} p(t)$$
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!
The Parsing Problem

Critics write reviews with computers.
A Recursive Parser

\[
\text{bestScore}(X,i,j,s) \\
\quad \text{if (j == i)} \\
\quad \quad \text{return } q(X\rightarrow s[i]) \\
\quad \text{else} \\
\quad \quad \text{return } \max_{k, X\rightarrow YZ} q(X\rightarrow YZ) \times \text{bestScore}(Y,i,k,s) \times \text{bestScore}(Z,k+1,j,s)
\]

- Will this parser work?
- Why or why not?
- Memory/time requirements?
- Q: Remind you of anything? Can we adapt this to other models / inference tasks?
Dynamic Programming

- We will store: score of the max parse of $x_i$ to $x_j$ with root non-terminal $X$
  \[ \pi(i, j, X) \]

- So we can compute the most likely parse:
  \[ \pi(1, n, S) = \arg \max_{t \in T_G(s)} \]

- Via the recursion:
  \[ \pi(i, j, X) = \max_{X \rightarrow YZ \in R, \ s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z)) \]

- With base case:
  \[ \pi(i, i, X) = \begin{cases} 
  q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
  0 & \text{otherwise}
\end{cases} \]
The CKY Algorithm

- **Input:** a sentence $s = x_1 .. x_n$ and a PCFG $= \langle N, \Sigma, S, R, q \rangle$
- **Initialization:** For $i = 1 ... n$ and all $X$ in $N$
  \[ \pi(i, i, X) = \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases} \]
  - For $l = 1 ... (n-1)$
    - For $i = 1 ... (n-l)$ and $j = i+l$
      - For all $X$ in $N$
        - [iterate all phrase lengths]
          - [iterate all phrases of length $l$]
            - [iterate all non-terminals]

  \[ \pi(i, j, X) = \max_{X \to YZ \in R, s \in \{i ... (j-1)\}} (q(X \to YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z)) \]

- also, store back pointers

  \[ bp(i, j, X) = \arg \max_{X \to YZ \in R, s \in \{i ... (j-1)\}} (q(X \to YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z)) \]
Probabilistic CKY Parser

S → NP VP 0.8
S → X1 VP 0.1
X1 → Aux NP 1.0
S → book | include | prefer
     0.01  0.004  0.006
S → Verb NP 0.05
S → VP PP 0.03
NP → I | he | she | me
     0.1  0.02  0.02  0.06
NP → Houston | NWA
     0.16  0.04
Det → the | a | an
     0.6  0.1  0.05
NP → Det Nominal 0.6
Nominal → book | flight | meal | money
     0.03  0.15  0.06  0.06
Nominal → Nominal Nominal 0.2
Nominal → Nominal PP 0.5
Verb → book | include | prefer
     0.5  0.04  0.06
VP → Verb NP 0.5
VP → VP PP 0.3
Prep → through | to | from
     0.2  0.3  0.3
PP → Prep NP 1.0
## Probabilistic CKY Parser

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
<th>Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S :.01</td>
<td>S :.05*.5*.054 =.00135</td>
<td>None</td>
<td>None</td>
<td>S :.0000216</td>
<td>.0135</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NP :.6*.6*.0024 =.000864</td>
<td>None</td>
<td>None</td>
<td>NP :.6*.6*.16 =.054</td>
<td>.054</td>
<td></td>
</tr>
<tr>
<td>Det :.6</td>
<td>NP :.6*.6*.15 =.054</td>
<td>None</td>
<td>None</td>
<td>Nominal :.5*.15*.032 =.0024</td>
<td>.0024</td>
<td></td>
</tr>
<tr>
<td>Nominal :.16</td>
<td>Prep :.2<em>1.0</em>.16 =.032</td>
<td>None</td>
<td>PP :1.0*.2*.16 =.032</td>
<td>NP :.16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table above illustrates the Probabilistic CKY Parser process for the sentence "Book the flight through Houston." The parser calculates the probability of each possible constituent and cell combination, denoted by the entries in the table. The most probable parse is determined by selecting the maximum probability from all possible derivations for each constituent and cell.
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 (<= n)
  For each i (<= n)
    For each rule X → Y Z
      For each split point k
        Do constant work

Total time: |rules|*n^3
Something like 5 sec for an unoptimized parse of a 20-word sentences
Time: Practice

- Parsing with the vanilla treebank grammar:
  - ~20K Rules (not an optimized parser!)
  - Observed exponent: 3.6

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

Can also compute other quantities:

- **Best Inside**: score of the max parse of \( w_i \) to \( w_j \) with root non-terminal \( X \)

- **Best Outside**: score of the max parse of \( w_0 \) to \( w_n \) with a gap from \( w_i \) to \( w_j \) rooted with non-terminal \( X \)
  - see notes for derivation, it is a bit more complicated

- **Sum Inside/Outside**: Do sums instead of maxes
Special Case: Unary Rules

- **Chomsky normal form (CNF):**
  - All rules of the form $X \rightarrow Y Z$ or $X \rightarrow w$
  - Makes parsing easier!

- **Can also allow unary rules**
  - All rules of the form $X \rightarrow Y Z$, $X \rightarrow Y$, or $X \rightarrow w$
  - You will need to do this case in your homework!
  - Conversion to/from the normal form is easier
  - Q: How does this change CKY?
  - WARNING: Watch for unary cycles…
We need unaries to be non-cyclic

- Calculate closure $\text{Close}(R)$ for unary rules in $R$
  - Add $X \rightarrow Y$ if there exists a rule chain $X \rightarrow Z_1$, $Z_1 \rightarrow Z_2$, ..., $Z_k \rightarrow Y$
    with $q(X \rightarrow Y) = q(X \rightarrow Z_1) \times q(Z_1 \rightarrow Z_2) \times ... \times q(Z_k \rightarrow Y)$
  - Add $X \rightarrow X$ with $q(X \rightarrow X) = 1$ for all $X$ in $N$

Rather than zero or more unaries, always exactly one
- Alternate unary and binary layers
- Reconstruct unary chains afterwards
CKY with Unary Closure

- **Input:** a sentence $s = x_1 \ldots x_n$ and a PCFG $= \langle N, \Sigma, S, R, q \rangle$
- **Initialization:** For $i = 1 \ldots n$:
  - Step 1: for all $X$ in $N$:
    $$\pi(i, i, X) = \begin{cases} 
    q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
    0 & \text{otherwise}
    \end{cases}$$
  - Step 2: for all $X$ in $N$:
    $$\pi_U(i, i, X) = \max_{X \rightarrow Y \in \text{Close}(R)} (q(X \rightarrow Y) \times \pi(i, i, Y))$$
- For $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$ and $j = i+l$
    - Step 1: (Binary)
      - For all $X$ in $N$
        $$\pi_B(i, j, X) = \max_{X \rightarrow Y Z \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow Y Z) \times \pi_U(i, s, Y) \times \pi_U(s + 1, j, Z))$$
    - Step 2: (Unary)
      - For all $X$ in $N$
        $$\pi_U(i, j, X) = \max_{X \rightarrow Y \in \text{Close}(R)} (q(X \rightarrow Y) \times \pi_B(i, j, Y))$$
( (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))))))

.)
Treebank Grammars

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

```
ROOT → S 1
S → NP VP . 1
NP → PRP 1
VP → VBD ADJP 1
   PRP VBD ADJP
     He was JJ right
```

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

- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller
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.
  - Passive / complete symbols: NP, NP^S
  - Active / incomplete symbols: NP → NP CC •
PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If P is the system’s parse tree and T is the human parse tree (the “gold standard”):

- Recall = (# correct constituents in P) / (# constituents in T)
- Precision = (# correct constituents in P) / (# constituents in P)

Labeled Precision and labeled recall require getting the non-terminal label on the constituent node correct to count as correct.

- F1 is the harmonic mean of precision and recall.
  - F1 = (2 * Precision * Recall) / (Precision + Recall)
PARSEVAL Example

Correct Tree T

Computed Tree P

# Constituents: 11
# Correct Constituents: 10

# Constituents: 12

Recall = 10/11 = 90.9%
Precision = 10/12 = 83.3%
F₁ = 87.4%
Treebank PCFGs

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

```
ROOT
  | S
  | 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

- Structure Annotation [Johnson ’98, Klein&Manning ’03]
- Lexicalization [Collins ’99, Charniak ’00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. ’06]
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Structural annotation
Vertical Markovization

- Vertical Markov order: rewrites depend on past k ancestor nodes. (cf. parent annotation)
Horizontal Markovization

Order 1

Order $\infty$

Symbols

Horizontal Markov Order
Vertical and Horizontal

- **Examples:**
  - Raw treebank: \( v=1, h=\infty \)
  - Johnson 98: \( v=2, h=\infty \)
  - Collins 99: \( v=2, h=2 \)
  - Best F1: \( v=3, h=2v \)

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base: ( v=h=2v )</td>
<td>77.8</td>
<td>7.5K</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>
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>
A Fully Annotated (Unlex) Tree
Some Test Set Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>LP</th>
<th>LR</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magerman 95</td>
<td>84.9</td>
<td>84.6</td>
<td>84.7</td>
</tr>
<tr>
<td>Collins 96</td>
<td>86.3</td>
<td>85.8</td>
<td>86.0</td>
</tr>
<tr>
<td>Unlexicalized</td>
<td>86.9</td>
<td>85.7</td>
<td>86.3</td>
</tr>
<tr>
<td>Charniak 97</td>
<td>87.4</td>
<td>87.5</td>
<td>87.4</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.7</td>
<td>88.6</td>
<td>88.6</td>
</tr>
</tbody>
</table>

- Beats “first generation” lexicalized parsers.
- Lots of room to improve – more complex models next.
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
Problems with PCFGs

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

- Add “headwords” to each phrasal node
  - 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

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

- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps
Complement / Adjunct Distinction

- *warning* - can be tricky, and most parsers don’t model the distinction

Complement: defines a property/argument (often obligatory), ex: [capitol [of Rome]]

Adjunct: modifies / describes something (always optional), ex: [quickly ran]

A Test for Adjuncts: [X Y] --> can claim X and Y
  - [they ran and it happened quickly] vs. [capitol and it was of Rome]
Lexical Derivation Steps

- **Main idea:** define a linguistically-motivated Markov process for generating children given the parent

  - **Step 1:** Choose a head tag and word
  - **Step 2:** Choose a complement bag
  - **Step 3:** Generate children (incl. adjuncts)
  - **Step 4:** Recursively derive children

[Collins 99]
bestScore(X,i,j,h)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return
      \[
      \max_{k,h',X\rightarrow YZ} \max_{k,h',X\rightarrow YZ} \text{score}(X[h]\rightarrow Y[h'] Z[h']) * \\
      \text{bestScore}(Y,i,k,h) * \\
      \text{bestScore}(Z,k,j,h')
      \]
      \[
      \max_{k,h',X\rightarrow YZ} \text{score}(X[h]\rightarrow Y[h'] Z[h]) * \\
      \text{bestScore}(Y,i,k,h') * \\
      \text{bestScore}(Z,k,j,h)
      \]
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

- Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Treebank Grammar</td>
<td>72.6</td>
</tr>
<tr>
<td>Klein &amp; Manning '03</td>
<td>86.3</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.6</td>
</tr>
</tbody>
</table>
The Game of Designing a Grammar

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

- Manually split categories
  - NP: subject vs object
  - DT: determiners vs demonstratives
  - IN: sentential vs prepositional

- Advantages:
  - Fairly compact grammar
  - Linguistic motivations

- Disadvantages:
  - Performance leveled out
  - Manually annotated
Learning Latent Annotations

Latent Annotations:
- Brackets are known
- Base categories are known
- Hidden variables for subcategories

He was right.

Can learn with EM: like Forward-Backward for HMMs.
Automatic Annotation Induction

- **Advantages:**
  - **Automatically learned:**
    - Label all nodes with latent variables.
    - Same number $k$ of subcategories for all categories.

- **Disadvantages:**
  - Grammar gets too large
  - Most categories are oversplit while others are undersplit.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
</tr>
</tbody>
</table>
Refinement of the DT tag

DT

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

DT-1
- a (0.61)
- the (0.19)
- an (0.11)

DT-2
- the (0.80)
- The (0.15)
- a (0.01)

DT-3
- this (0.39)
- that (0.28)
- That (0.11)

DT-4
- some (0.20)
- all (0.19)
- those (0.12)
Hierarchical refinement

- Repeatedly learn more fine-grained subcategories
- Start with two (per non-terminal), then keep splitting
- Initialize each EM run with the output of the last
Adaptive Splitting

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

[Petrov et al. 06]
Adaptive Splitting

- Evaluate loss in likelihood from removing each split =
  \[
  \frac{\text{Data likelihood with split reversed}}{\text{Data likelihood with split}}
  \]
  
- No loss in accuracy when 50% of the splits are reversed.
Adaptive Splitting 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>

![Graph showing parsing accuracy (F1) vs. total number of grammar rules for different training methods: 50% Merging, Hierarchical Training, Flat Training. The graph shows an improvement in F1 score from 88.4 to 89.5 with 50% merging.]
Number of Lexical Subcategories
## Final Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 ≤ 40 words</th>
<th>F1 all words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
<td>85.7</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
<td>86.1</td>
</tr>
<tr>
<td>Collins ’99</td>
<td>88.6</td>
<td>88.2</td>
</tr>
<tr>
<td>Charniak &amp; Johnson ’05</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td>Petrov et. al. 06</td>
<td>90.2</td>
<td>89.7</td>
</tr>
</tbody>
</table>
# 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>
Hierarchical Pruning

Parse multiple times, with grammars at different levels of granularity!

- Coarse:
  - Split in two:
    - Split in four:
      - Split in eight:
1621 min
1111 min
35 min
15 min
(no search error)
## 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
Lexicalized parsers can be seen as producing *dependency trees*

- Each local binary tree corresponds to an attachment in the dependency graph
Dependency Parsing*

- 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]
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*

Tree diagrams representing sentence structures.
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 ⊨ NP
shares ⊨ NP
buys ⊨ (S\NP)/NP
sleeps ⊨ S\NP
well ⊨ (S\NP)\(S\NP)
```

```
S
  NP
    John
  S\NP
    buys
    shares
```