Constituency Parsing

Instructor: Yoav Artzi

Slides adapted from Dan Klein, Dan Jurafsky, Chris Manning, Michael Collins, Luke Zettlemoyer, Yejin Choi, and Slav Petrov
Overview

• The constituency parsing problem
• CKY parsing
  – Chomsky Normal Form
• The Penn Treebank
Constituency (Phrase Structure) Trees

- Phrase structure organizes words into nested constituents
Constituency (Phrase Structure) Trees

- Phrase structure organizes words into nested constituents
- Linguists can, and do, argue about details
Constituency Tests

• **Distribution:** a constituent behaves as a unit that can appear in different places:
  – John talked to the children about drugs.
  – John talked [to the children] [about drugs].
  – John talked [about drugs] [to the children].
  – *John talked drugs to the children about
Constituency Tests

• Distribution / movement / dislocation
• Substitution by pro-form
  – he, she, it, they, ...
• Question / answer
• Deletion
• Conjunction / coordination
Constituency (Phrase Structure) Trees

• Phrase structure organizes words into nested constituents
• Linguists can, and do, argue about details
• Lots of ambiguity

new art critics write reviews with computers
Context-Free Grammars (CFG)

- Writing parsing rules:
  - N → Fed
  - V → raises
  - NP → N
  - S → NP VP
  - VP → V NP
  - NP → N N
  - NP → NP PP
  - N → interest
  - N → raises

![CFG Diagram]

```plaintext
S
  NP
    N
    Fed
  VP
    V
    raises
    NP
      N
      interest
    N
      rates
```
Context-Free Grammars

• A context-free grammar is a tuple \(<N, \Sigma, S, R>\)
  – \(N\) : the set of non-terminals
    • Phrasal categories: S, NP, VP, ADJP, etc.
    • Parts-of-speech (pre-terminals): 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 \(S\) – why not?
  – \(R\) : 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

\[ N = \{S, \text{NP}, \text{VP}, \text{PP}, \text{DT}, \text{Vi}, \text{Vt}, \text{NN}, \text{IN}\} \]

\[ S = S \]

\[ \Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\} \]

\[ R = \]

\[
\begin{array}{|c|c|c|}
\hline
S & \rightarrow & \text{NP} \quad \text{VP} \\
\hline
\text{VP} & \rightarrow & \text{Vi} \\
\hline
\text{VP} & \rightarrow & \text{Vt} \quad \text{NP} \\
\hline
\text{VP} & \rightarrow & \text{VP} \quad \text{PP} \\
\hline
\text{NP} & \rightarrow & \text{DT} \quad \text{NN} \\
\hline
\text{NP} & \rightarrow & \text{NP} \quad \text{PP} \\
\hline
\text{PP} & \rightarrow & \text{IN} \quad \text{NP} \\
\hline
\end{array}
\]

<table>
<thead>
<tr>
<th></th>
<th>\rightarrow</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Vi</td>
<td>\rightarrow</td>
<td>\text{sleeps}</td>
</tr>
<tr>
<td>Vt</td>
<td>\rightarrow</td>
<td>\text{saw}</td>
</tr>
<tr>
<td>NN</td>
<td>\rightarrow</td>
<td>\text{man}</td>
</tr>
<tr>
<td>NN</td>
<td>\rightarrow</td>
<td>\text{woman}</td>
</tr>
<tr>
<td>NN</td>
<td>\rightarrow</td>
<td>\text{telescope}</td>
</tr>
<tr>
<td>DT</td>
<td>\rightarrow</td>
<td>\text{the}</td>
</tr>
<tr>
<td>IN</td>
<td>\rightarrow</td>
<td>\text{with}</td>
</tr>
<tr>
<td>IN</td>
<td>\rightarrow</td>
<td>\text{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

N = {S, NP, VP, PP, DT, Vi, Vt, NN, IN}

Σ = {sleeps, saw, man, woman, telescope, the, with, in}

R =

\[
\begin{array}{c|cc}
  S & \Rightarrow & NP \quad VP \\
  VP & \Rightarrow & Vi \\
  VP & \Rightarrow & Vt \quad NP \\
  VP & \Rightarrow & VP \quad PP \\
  NP & \Rightarrow & DT \quad NN \\
  NP & \Rightarrow & NP \quad PP \\
  PP & \Rightarrow & IN \quad NP \\
\end{array}
\]

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 Parse

The man sleeps

S

NP

VP

DT

NN

Vt

Vi

The man sleeps

Vi \Rightarrow sleeps
Vt \Rightarrow saw
NN \Rightarrow man
NN \Rightarrow woman
NN \Rightarrow telescope
DT \Rightarrow the
IN \Rightarrow with
IN \Rightarrow in
Example Parse

\[ R = \begin{array}{|l|l|l|}
\hline
S & \Rightarrow & NP \quad VP \\
VP & \Rightarrow & Vi \\
VP & \Rightarrow & Vt \quad NP \\
VP & \Rightarrow & VP \quad PP \\
NP & \Rightarrow & DT \quad NN \\
NP & \Rightarrow & NP \quad PP \\
PP & \Rightarrow & IN \quad NP \\
\hline
\end{array} \]

Vi $\Rightarrow$ sleeps
Vt $\Rightarrow$ saw
NN $\Rightarrow$ man
NN $\Rightarrow$ woman
NN $\Rightarrow$ telescope
DT $\Rightarrow$ the
IN $\Rightarrow$ with
IN $\Rightarrow$ in

The man saw the woman with the telescope

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 Parse

\[ R = \begin{array}{c|cc}
  S & NP & VP \\
  VP & Vi & \\
  VP & Vt & NP \\
  VP & VP & PP \\
  NP & DT & NN \\
  NP & NP & PP \\
  PP & IN & NP \\
\end{array} \]

Vi \Rightarrow \text{sleeps}
Vt \Rightarrow \text{saw}
NN \Rightarrow \text{man}
NN \Rightarrow \text{woman}
NN \Rightarrow \text{telescope}
DT \Rightarrow \text{the}
IN \Rightarrow \text{with}
IN \Rightarrow \text{in}

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

• In NLP, CFG non-terminals often have internal structure
• Phrases are headed by particular word types with some modifiers:
  – VP $\rightarrow \ldots \text{VB}^* \ldots$
  – NP $\rightarrow \ldots \text{NN}^* \ldots$
  – ADJP $\rightarrow \ldots \text{JJ}^* \ldots$
  – ADVP $\rightarrow \ldots \text{RB}^* \ldots$
• This X-bar theory grammar (in a nutshell)
• This captures a dependency
Pre 1990 (“Classical”) NLP Parsing

- Wrote symbolic grammar (CFG or often richer) and lexicon
  
  $S \rightarrow NP \ VP$  
  $NN \rightarrow interest$  
  $NP \rightarrow (DT) \ NN$  
  $NNS \rightarrow rates$  
  $NP \rightarrow NN \ NNS$  
  $NNS \rightarrow raises$  
  $NP \rightarrow NNP$  
  $VBP \rightarrow interest$  
  $VP \rightarrow V \ NP$  
  $VBZ \rightarrow rates$

- Used grammar-proof systems to prove parses from words
- This scaled very badly and didn’t give coverage. For sentence:

  *Fed raises interest rates 0.5% in effort to control inflation*

- Minimal grammar: 36 parses
- Simple 10 rule grammar: 592 parses
- Real-size broad-coverage grammar: millions of parses
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 Ambiguity 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 Ambiguity 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.
Classical NLP Parsing: The problem and its solution

- Categorical constraints can be added to grammars to limit unlikely/weird parses for sentences
  - But the attempt makes the grammars not robust
    - In traditional systems, commonly 30% of sentences in even an edited text would have *no* parse.
- A less constrained grammar can parse more sentences
  - But simple sentences end up with ever more parses with no way to choose between them
- We need mechanisms that allow us to find the most likely parse(s) for a sentence
  - Statistical parsing lets us work with very loose grammars that admit millions of parses for sentences but still quickly find the best parse(s)
The rise of annotated data: The Penn Treebank (PTB)

(S
  (NP-SBJ (DT The) (NN move))
  (VP (VBD followed))
  (NP
    (NP (DT a) (NN round))
    (PP (IN of)
      (NP
        (NP (JJ similar) (NNS increases))
        (PP (IN by)
          (NP (JJ other) (NNS lenders)))
        (PP (IN against)
          (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
(S-ADV
  (NP-SBJ (-NONE- *))
  (VP (VBG reflecting)
    (NP
      (NP (DT a) (VBG continuing) (NN decline))
      (PP-LOC (IN in)
        (NP (DT that) (NN market)))))
  (. .)))
The rise of annotated data

• Starting off, building a treebank seems a lot slower and less useful than building a grammar

• But a treebank gives us many things
  – Reusability of the labor
    • Many parsers, POS taggers, etc.
    • Valuable resource for linguistics
  – Broad coverage
  – Frequencies and distributional information
  – A way to evaluate systems
### 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>

+ all POS tags
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
PTB Size

- Penn WSJ Treebank:
  - 50,000 annotated sentences
- Usual set-up:
  - 40,000 training
  - 2,400 test
Probabilistic Context-Free Grammars (PCFG)

• A context-free grammar is a tuple \(<N, \Sigma, S, R>\)
  – \(N\) : the set of non-terminals
    • Phrasal categories: \(S, NP, VP, ADJP\), etc.
    • Parts-of-speech (pre-terminals): \(NN, JJ, DT, VB\)
  – \(\Sigma\) : the set of terminals (the words)
  – \(S\) : the start symbol
    • Often written as \(ROOT\) or \(TOP\)
    • \textbf{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: \(S \rightarrow NP VP\), \(VP \rightarrow VP \text{ CC VP}\)
    • Also called rewrites, productions, or local trees

• A PCFG adds a distribution \(q\):
  – Probability \(q(r)\) for each \(r \in R\), \textbf{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>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S ( \Rightarrow ) NP VP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP ( \Rightarrow ) Vi</td>
<td>0.4</td>
</tr>
<tr>
<td>VP ( \Rightarrow ) Vt NP</td>
<td>0.4</td>
</tr>
<tr>
<td>VP ( \Rightarrow ) VP PP</td>
<td>0.2</td>
</tr>
<tr>
<td>NP ( \Rightarrow ) DT NN</td>
<td>0.3</td>
</tr>
<tr>
<td>NP ( \Rightarrow ) NP PP</td>
<td>0.7</td>
</tr>
<tr>
<td>PP ( \Rightarrow ) P NP</td>
<td>1.0</td>
</tr>
<tr>
<td>Vi ( \Rightarrow )</td>
<td>sleeps</td>
</tr>
<tr>
<td>Vt ( \Rightarrow ) saw</td>
<td>1.0</td>
</tr>
<tr>
<td>NN ( \Rightarrow ) man</td>
<td>0.7</td>
</tr>
<tr>
<td>NN ( \Rightarrow ) woman</td>
<td>0.2</td>
</tr>
<tr>
<td>NN ( \Rightarrow ) telescope</td>
<td>0.1</td>
</tr>
<tr>
<td>DT ( \Rightarrow ) the</td>
<td>1.0</td>
</tr>
<tr>
<td>IN ( \Rightarrow ) with</td>
<td>0.5</td>
</tr>
<tr>
<td>IN ( \Rightarrow ) 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>Rule</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>

The man sleeps

The man saw the woman with the telescope
### PCFG Example

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

**Tree 1 ($t_1$):**
- $S \Rightarrow NP\ VP$
- $NP \Rightarrow DT\ NN$
- $VP \Rightarrow Vi$

**Tree 2 ($t_2$):**
- $S \Rightarrow VP\ PP$
- $VP \Rightarrow Vt\ NP$

**Sentences:**
- The man sleeps
- The man saw the woman with the telescope

**Probabilities:**
- $p(t_1) = 1.0 \times 0.3 \times 1.0 \times 0.7 \times 0.4 \times 1.0$
- $p(t_2) = 1.0 \times 0.3 \times 1.0 \times 0.7 \times 0.2 \times 0.4 \times 0.5 \times 0.3 \times 1.0 \times 0.1$
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 whose \textit{yield} is \( s \) (whose leaves, read left to right, match the words in \( s \))
  \[
  t^*(s) = \arg \max_{t \in T(s)} p(t)
  \]
The Constituency Parsing Problem

PCFG

Rule Prob $\theta_i$

$S \rightarrow NP \ VP \ \theta_0$

$NP \rightarrow NP \ NP \ \theta_1$

$\ldots$

$N \rightarrow \text{fish} \ \theta_{42}$

$N \rightarrow \text{people} \ \theta_{43}$

$V \rightarrow \text{fish} \ \theta_{44}$

$\ldots$
A Recursive Parser

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

- Will this parser work?
- Why or why not?
- Q: Remind you of anything? Can we adapt this to other models / inference tasks?
Cocke-Kasami-Younger (CKY) Constituency Parsing

fish  people  fish    tanks
Cocke-Kasami-Younger (CKY) Constituency Parsing

S → NP VP  0.9
VP → V NP  0.5
VP → V @VP_V  0.3
VP → V PP  0.1
@VP_V → NP PP  1.0
NP → NP NP  0.1
NP → NP PP  0.2
PP → P NP  1.0
Cocke-Kasami-Younger (CKY) Constituency Parsing

S → NP VP 0.9
VP → V NP 0.5
VP → V @VP_V 0.3
VP → V PP 0.1
@VP_V → NP PP 1.0
NP → NP NP 0.1
NP → NP PP 0.2
PP → P NP 1.0

people

fish
Cocke-Kasami-Younger (CKY) Constituency Parsing

- **S** → NP VP 0.9
- **VP** → V NP 0.5
- **VP** → V @VP_V 0.3
- **VP** → V PP 0.1
- @VP_V → NP PP 1.0
- **NP** → NP NP 0.1
- **NP** → NP PP 0.2
- PP → P NP 1.0

NP 0.35
V 0.1
N 0.5

VP 0.06
NP 0.14
V 0.6
N 0.2

S 0.0189
NP 0.0098

people

fish
Cocke-Kasami-Younger (CKY)
Constituency Parsing

S → NP VP 0.9
VP → V NP 0.5
VP → V @VP_V 0.3
@VP_V → NP PP 1.0
NP → NP NP 0.1
NP → NP PP 0.2
PP → P NP 1.0
CKY Parsing

- 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:
  \[ \pi(1, n, S) = \arg \max_t \in T_G(x) \]

- Via the recursion:
  \[ \pi(i, j, X) = \]

- With base case:
  \[ \pi(i, i, X) = \]
The CKY Algorithm

- **Input:** a sentence \( s = x_1 \ldots x_n \) and a PCFG = \(<N, \Sigma, S, R, q>\)
- **Initialization:** For \( i = 1 \ldots n \) and all \( X \) in \( N \)
  \[
  \pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}
  \]
- For \( l = 1 \ldots (n-1) \) [iterate all phrase lengths]
  - For \( i = 1 \ldots (n-l) \) and \( j = i+l \) [iterate all phrases of length \( l \)]
    - For all \( X \) in \( N \) [iterate all non-terminals]
  
  \[
  \pi(i, j, X) = \max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
  \]

  - also, store back pointers
  
  \[
  bp(i, j, X) = \arg\max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
  \]
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 .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

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 .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

Pick most probable parse

Book            the              flight       through       Houston
The CKY Algorithm

- **Input:** a sentence $s = x_1 .. x_n$ and a PCFG $= \langle N, \Sigma , S, R, q \rangle$
- **Initialization:** For $i = 1 \ldots n$ and all $X$ in $N$
  \[
  \pi(i, i, X) = \begin{cases}
  q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
  0 & \text{otherwise}
  \end{cases}
  \]
- For $l = 1 \ldots (n-1)$ [iterate all phrase lengths]
  - For $i = 1 \ldots (n-l)$ and $j = i+l$ [iterate all phrases of length $l$]
    - For all $X$ in $N$ [iterate all non-terminals]
      \[
      \pi(i, j, X) = \max_{X \rightarrow YZ \in R, \atop s \in \{i .. (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
      \]
  - also, store back pointers
    \[
    bp(i, j, X) = \arg \max_{X \rightarrow YZ \in R, \atop s \in \{i .. (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
    \]
Time: Theory

• For each length (\(\leq n\))
  – For each \(i\) (\(\leq n\))
    • For each split point \(k\)
      – For each rule \(X \rightarrow Y Z\)
        » Do constant work

• Total time: \(|\text{rules}| \times n^3\)
Time: Practice

- Parsing with the vanilla treebank grammar:
  - Longer sentences “unlock” more of the grammar
  - All kinds of systems issues don’t scale

Observed exponent: 3.6
The CKY Algorithm

- **Input:** a sentence $s = x_1 \ldots x_n$ and a PCFG $= <N, \Sigma, S, R, q>$
- **Initialization:** For $i = 1 \ldots n$ and all $X$ in $N$
  \[
  \pi(i, i, X) = \begin{cases} 
  q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
  0 & \text{otherwise}
  \end{cases}
  \]
- For $l = 1 \ldots (n-1)$ [iterate all phrase lengths]
  - For $i = 1 \ldots (n-l)$ and $j = i+l$ [iterate all phrases of length $l$]
    - For all $X$ in $N$ [iterate all non-terminals]
      \[
      \pi(i, j, X) = \max_{X \rightarrowYZ \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
      \]
      - also, store back pointers
      \[
      bp(i, j, X) = \arg \max_{X \rightarrowYZ \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
      \]
Memory

• How much memory does this require?
  – Have to store the score cache
  – Cache size:
    • \(|\text{symbols}|n^2\) doubles

• 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 \([i,j]\) – Exact?

• Pruning: Coarse-to-Fine
  – Use a smaller grammar to rule out most \(X[i,j]\)
Let’s parse with CKY!

• Any problem?

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.
Chomsky Normal Form

- All rules are of the form $X \rightarrow Y Z$ or $X \rightarrow w$
  - $X, Y, Z \in N$ and $w \in T$
- A transformation to this form doesn’t change the weak generative capacity of a CFG
  - That is, it recognizes the same language
    - But maybe with different trees
- Empties and unaries are removed recursively
- $n$-ary rules are divided by introducing new nonterminals ($n > 2$)
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 $X \rightarrow w$
  – Conversion to/from the normal form is easier
  – Q: How does this change CKY?
  – WARNING: Watch for unary cycles…
CKY with Unary Rules

- **Input:** a sentence $s = x_1 .. x_n$ and a PCFG $= <N, \Sigma, S, R, q>$
- **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 $N$:
    $$\pi_U(i, i, X) = \max_{X \rightarrow Y \in R} (q(X \rightarrow Y) \times \pi(i, i, Y))$$

- For $l = 1 \ldots (n-1)$ [iterate all phrase lengths]
  - For $i = 1 \ldots (n-l)$ and $j = i+l$ [iterate all phrases of length $l$]
    - **Step 1:** (Binary)
      - For all $X$ in $N$ [iterate all non-terminals]
        $$\pi_B(i, j, X) = \max_{X \rightarrow Y Z \in R, s \in \{i \ldots (j-1)\}} (q(X \rightarrow Y Z) \times \pi_B(i, j-1, Y)) \times \pi_B(i+1, j, Z))$$
    - **Step 2:** (Unary)
      - For all $X$ in $N$ [iterate all non-terminals]
        $$\pi_U(i, j, X) = \max_{X \rightarrow Y \in R} (q(X \rightarrow Y) \times \pi_B(i, j, Y))$$

Must always have one and exactly one unary rule!
Unary Closure

• Rather than zero or more unaries, always exactly one
• Calculate closure Close(R) for unary rules in R
  – Add $X \rightarrow Y$ if there exists a rule chain $X \rightarrow Z_1, Z_1 \rightarrow Z_2, \ldots, Z_k \rightarrow Y$ with $q(X \rightarrow Y) = q(X \rightarrow Z_1) 	imes q(Z_1 \rightarrow Z_2) \times \ldots \times q(Z_k \rightarrow Y)$
  – Add $X \rightarrow X$ with $q(X \rightarrow X) = 1$ for all $X$ in $N$

• In CKY and chart: Alternate unary and binary layers
• Reconstruct unary chains afterwards (with extra marking)
Other Chart Computations

- Max inside score
  - Score of the max parse of $x_i$ to $x_j$ with root $X$
    \[ \pi(i, j, X) \]
  - Marginalize over internal structure
- Max outside score
- Sum inside/outside
Other Chart Computations

- Max inside score
- Max outside score
  - Score of max parse of the complete span with a gap between i and j
  - Details in notes
- Sum inside/outside
Other Chart Computations

- Max inside score
- Max outside score
- Sum inside/outside
  - Do sums instead of maxes
Just Like Sequences

• Locally normalized:
  – Generative
  – MaxEnt

• Globally normalized:
  – CRFs

• Additive, un-normalized:
  – Perceptron
Treebank Parsing

(S
  (NP-SBJ (DT The) (NN move))
  (VP (VBD followed))
  (NP
    (NP (DT a) (NN round))
    (PP (IN of)
      (NP
        (NP (JJ similar) (NNS increases))
        (PP (IN by)
          (NP (JJ other) (NNS lenders))))
        (PP (IN against)
          (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
  (, ,)
(S-ADV
  (NP-SBJ (-NONE- *))
  (VP (VBG reflecting))
  (NP
    (NP (DT a) (VBG continuing) (NN decline))
    (PP-LOC (IN in)
      (NP (DT that) (NN market))))))
  (, .)))

[Marcus et al. 1993]
Treebank Grammars

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees:

```
ROOT → S 1
S → NP VP . 1
NP → PRP 1
VP → VBD ADJP 1
    |   |   |
    |   |   |
PRP  VBD  ADJP
    |   |   |
    |   |   |
  He  was  JJ
    |   |   |
    |   |   |
    |   |   |
  right
```
Typical Experimental Setup

• The Penn Treebank is divided into sections:
  – Training: sections 2-18
  – Development: section 22 (also 0-1 and 24)
  – Testing: section 23

• Evaluation?
Evaluating Constituency Parsing

Gold standard brackets: S-(0:11), NP-(0:2), VP-(2:9), VP-(3:9), NP-(4:6), PP-(6:9), NP-(7,9), NP-(9:10)

Candidate brackets: S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:6), PP-(6-10), NP-(7,10)
Evaluating Constituency Parsing

• Recall:
  – Recall = (# correct constituents in candidate) / (# constituents in gold)

• Precision:
  – Precision = (# correct constituents in candidate) / (# constituents in candidate)

• 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)
Evaluating Constituency Parsing

Gold standard brackets:
S-(0:11), NP-(0:2), VP-(2:9), VP-(3:9), NP-(4:6), PP-(6-9), NP-(7,9), NP-(9:10)

Candidate brackets:
S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:6), PP-(6-10), NP-(7,10)

• Precision: \[
\frac{3}{7} = 42.9\%
\]
• Recall: \[
\frac{3}{8} = 37.5\%
\]
• F1: \[
40\%
\]
• Also, tagging accuracy: \[
\frac{11}{11} = 100\%
\]
How Good are PCFGs?

• Robust
  – Usually admit everything, but with low probability
• Partial solution for grammar ambiguity
  – A PCFG gives some idea of the plausibility of a parse
  – But not so good because the independence assumptions are too strong
• Give a probabilistic language model
  – But in the simple case it performs worse than a trigram model
• The problem seems to be that PCFGs lack the lexicalization of a trigram model

Penn WSJ parsing performance:
~ 73% F1
The man saw the woman with the hat
Extra Slides
Chomsky Normal Form

- All rules are of the form $X \rightarrow Y Z$ or $X \rightarrow w$
  - $X, Y, Z \in N$ and $w \in T$
- A transformation to this form doesn’t change the weak generative capacity of a CFG
  - That is, it recognizes the same language
    - But maybe with different trees
- Empties and unaries are removed recursively
- n-ary rules are divided by introducing new nonterminals ($n > 2$)
Example: Before Binarization

```
ROOT
  S
    NP
      N
        people
    VP
      NP
        V
          fish
        PP
          P
            N
              tanks
          NP
            N
              rods
    with
```
Example: After Binarization

```
ROOT
  S
    NP  VP
      N  @VP_V
         V
           NP  PP
               N  P  NP
                  N
people  fish  tanks  with  rods
```
A Phrase Structure Grammar

<table>
<thead>
<tr>
<th>Production</th>
<th>Left-Hand Side</th>
<th>Right-Hand Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>NP VP</td>
<td>N \to people</td>
</tr>
<tr>
<td>VP</td>
<td>V NP</td>
<td>N \to fish</td>
</tr>
<tr>
<td>VP</td>
<td>V NP PP</td>
<td>N \to tanks</td>
</tr>
<tr>
<td>NP</td>
<td>NP NP</td>
<td>N \to rods</td>
</tr>
<tr>
<td>NP</td>
<td>NP PP</td>
<td>V \to people</td>
</tr>
<tr>
<td>NP</td>
<td>N</td>
<td>V \to fish</td>
</tr>
<tr>
<td>NP</td>
<td>e</td>
<td>V \to tanks</td>
</tr>
<tr>
<td>PP</td>
<td>P NP</td>
<td>P \to with</td>
</tr>
</tbody>
</table>
Chomsky Normal Form

Step 1: Remove epsilon rules

\[
S \rightarrow NP \ VP \\
VP \rightarrow V \ NP \\
VP \rightarrow V \ NP \ PP \\
NP \rightarrow NP \ NP \\
NP \rightarrow NP \ PP \\
NP \rightarrow N \\
NP \rightarrow e \\
PP \rightarrow P \ NP
\]

\[
N \rightarrow people \\
N \rightarrow fish \\
N \rightarrow tanks \\
N \rightarrow rods \\
V \rightarrow people \\
V \rightarrow fish \\
V \rightarrow tanks \\
P \rightarrow with
\]
Chomsky Normal Form

Step 1: Remove epsilon rules

\[
\begin{align*}
S & \rightarrow NP \ VP \\
VP & \rightarrow V \ NP \\
VP & \rightarrow V \ NP \ PP \\
NP & \rightarrow NP \ NP \\
NP & \rightarrow NP \ PP \\
NP & \rightarrow N \\
NP & \rightarrow e \\
PP & \rightarrow P \ NP \\
N & \rightarrow people \\
N & \rightarrow fish \\
N & \rightarrow tanks \\
N & \rightarrow rods \\
V & \rightarrow people \\
V & \rightarrow fish \\
V & \rightarrow tanks \\
P & \rightarrow with
\end{align*}
\]
Chomsky Normal Form

Step 1: Remove epsilon rules

\[
\begin{align*}
S & \rightarrow NP \ VP \\
VP & \rightarrow V \ NP \\
VP & \rightarrow V \ NP \ PP \\
NP & \rightarrow NP \ NP \\
NP & \rightarrow NP \ PP \\
NP & \rightarrow N \\
NP & \rightarrow e \\
PP & \rightarrow P \ NP
\end{align*}
\]

\[
\begin{align*}
N & \rightarrow people \\
N & \rightarrow fish \\
N & \rightarrow tanks \\
N & \rightarrow rods \\
V & \rightarrow people \\
V & \rightarrow fish \\
V & \rightarrow tanks \\
P & \rightarrow with
\end{align*}
\]

Recognizing the same language? For every rule with NP, create a unary rule
Chomsky Normal Form

Step 1: Remove epsilon rules

S → NP VP
S → VP
VP → V NP
VP → V
VP → V NP PP
VP → V PP
NP → NP NP
NP → NP
NP → NP PP
NP → PP
NP → N
PP → P NP
PP → P

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with
Chomsky Normal Form

Step 2: Remove unary rules

\[
\begin{align*}
S & \rightarrow \text{NP VP} & N & \rightarrow \text{people} \\
S & \rightarrow \text{VP} & N & \rightarrow \text{fish} \\
\text{VP} & \rightarrow \text{V NP} & N & \rightarrow \text{tanks} \\
\text{VP} & \rightarrow \text{V} & N & \rightarrow \text{rods} \\
\text{VP} & \rightarrow \text{V NP PP} & \text{V} & \rightarrow \text{people} \\
\text{VP} & \rightarrow \text{V PP} & \text{V} & \rightarrow \text{fish} \\
\text{NP} & \rightarrow \text{NP NP} & \text{V} & \rightarrow \text{tanks} \\
\text{NP} & \rightarrow \text{NP} \\
\text{NP} & \rightarrow \text{NP PP} \\
\text{NP} & \rightarrow \text{PP} \\
\text{NP} & \rightarrow \text{N} \\
\text{PP} & \rightarrow \text{P NP} \\
\text{PP} & \rightarrow \text{P} \\
\end{align*}
\]
Chomsky Normal Form

Step 2: Remove unary rules

\[
\begin{align*}
    S & \rightarrow NP \ VP \\
    S & \rightarrow VP \\
    VP & \rightarrow V \ NP \\
    VP & \rightarrow V \ PP \\
    VP & \rightarrow V \ NP \ PP \\
    NP & \rightarrow NP \ NP \\
    NP & \rightarrow NP \\
    NP & \rightarrow NP \ PP \\
    NP & \rightarrow PP \\
    NP & \rightarrow N \\
    PP & \rightarrow P \ NP \\
    PP & \rightarrow P \\
    N & \rightarrow people \\
    N & \rightarrow fish \\
    N & \rightarrow tanks \\
    N & \rightarrow rods \\
    V & \rightarrow people \\
    V & \rightarrow fish \\
    V & \rightarrow tanks \\
    P & \rightarrow with
\end{align*}
\]
Chomsky Normal Form

Step 2: Remove unary rules

\[
\begin{align*}
S & \rightarrow \text{NP VP} \\
S & \rightarrow \text{VP} \\
\text{VP} & \rightarrow V \text{ NP} \\
\text{VP} & \rightarrow V \\
\text{VP} & \rightarrow V \text{ NP PP} \\
\text{VP} & \rightarrow V \text{ PP} \\
\text{NP} & \rightarrow \text{NP NP} \\
\text{NP} & \rightarrow \text{NP} \\
\text{NP} & \rightarrow \text{NP PP} \\
\text{NP} & \rightarrow \text{PP} \\
\text{NP} & \rightarrow N \\
\text{PP} & \rightarrow P \text{ NP} \\
\text{PP} & \rightarrow P \\
\text{N} & \rightarrow \text{people} \\
\text{N} & \rightarrow \text{fish} \\
\text{N} & \rightarrow \text{tanks} \\
\text{N} & \rightarrow \text{rods} \\
\text{V} & \rightarrow \text{people} \\
\text{V} & \rightarrow \text{fish} \\
\text{V} & \rightarrow \text{tanks} \\
\text{P} & \rightarrow \text{with}
\end{align*}
\]
Chomsky Normal Form

Step 2: Remove unary rules

S → NP VP
S → VP
VP → V NP
VP → V
VP → V NP PP
VP → V PP
NP → NP NP
NP → NP
NP → NP PP
NP → PP
NP → N
PP → P NP
PP → P

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with

Recognizing the same language?
Work your way down to propagate
Step 2: Remove unary rules

S → NP VP
VP → V NP
S → V NP
VP → V
S → V
VP → V NP PP
S → V NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP
NP → NP PP
NP → PP
NP → N
PP → P NP
PP → P

Just added a unary rule!
Need to apply until they are all gone

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with
Chomsky Normal Form

Step 2: Remove unary rules

S → NP VP
VP → V NP
S → V NP
VP → V
S → V
VP → V NP PP
S → V NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP
NP → NP PP
NP → PP
NP → N
PP → P NP
PP → P

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with

Just added a unary rule! Need to apply until they are all gone
Chomsky Normal Form

Step 2: Remove unary rules

S → NP VP
VP → V NP
S → V NP
VP → V
VP → V NP PP
S → V NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP
NP → NP PP
NP → PP
NP → N
PP → P NP
PP → P

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with
Chomsky Normal Form

Step 2: Remove unary rules

S → NP VP
VP → V NP
S → V NP
VP → V
VP → V NP PP
S → V NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP
NP → NP PP
NP → PP
NP → N
PP → P NP
PP → P

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with
Chomsky Normal Form

Step 2: Remove unary rules

S \rightarrow \text{NP VP}
VP \rightarrow \text{V NP}
S \rightarrow \text{V NP}
VP \rightarrow \text{V NP PP}
S \rightarrow \text{V NP PP}
VP \rightarrow \text{V PP}
S \rightarrow \text{V PP}
\text{NP} \rightarrow \text{NP NP}
\text{NP} \rightarrow \text{NP}
\text{NP} \rightarrow \text{NP PP}
\text{NP} \rightarrow \text{PP}
\text{NP} \rightarrow \text{N}
\text{PP} \rightarrow \text{P NP}
\text{PP} \rightarrow \text{P}

N \rightarrow \text{people}
N \rightarrow \text{fish}
N \rightarrow \text{tanks}
N \rightarrow \text{rods}
V \rightarrow \text{people}
V \rightarrow \text{fish}
V \rightarrow \text{tanks}
P \rightarrow \text{with}
Chomsky Normal Form

Step 2: Remove unary rules

S → NP VP
VP → V NP
S → V NP
VP → V NP PP
S → V NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP PP
NP → PP
NP → N
PP → P NP
PP → P

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with

Recognizing the same language? Yes!
Chomsky Normal Form

Step 2: Remove unary rules

S \rightarrow NP \ VP
VP \rightarrow V \ NP
S \rightarrow V \ NP
VP \rightarrow V \ NP \ PP
S \rightarrow V \ NP \ PP
VP \rightarrow V \ PP
S \rightarrow V \ PP
NP \rightarrow NP \ NP
NP \rightarrow NP
NP \rightarrow NP \ PP
NP \rightarrow PP
NP \rightarrow N
PP \rightarrow P \ NP
PP \rightarrow P

N \rightarrow people
N \rightarrow fish
N \rightarrow tanks
N \rightarrow rods
V \rightarrow people
V \rightarrow fish
V \rightarrow tanks
P \rightarrow with

Only place N appears
So can get rid of it altogether
Chomsky Normal Form

Step 2: Remove unary rules

<table>
<thead>
<tr>
<th>Grammar Rule</th>
<th>Simplified Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow V \ NP$</td>
<td></td>
</tr>
<tr>
<td>$S \rightarrow V \ NP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow V \ NP \ PP$</td>
<td></td>
</tr>
<tr>
<td>$S \rightarrow V \ NP \ PP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow V \ PP$</td>
<td></td>
</tr>
<tr>
<td>$S \rightarrow V \ PP$</td>
<td></td>
</tr>
<tr>
<td>$NP \rightarrow NP \ NP$</td>
<td>$NP \rightarrow people$</td>
</tr>
<tr>
<td>$NP \rightarrow NP \ PP$</td>
<td>$NP \rightarrow fish$</td>
</tr>
<tr>
<td>$NP \rightarrow PP$</td>
<td>$NP \rightarrow tanks$</td>
</tr>
<tr>
<td>$PP \rightarrow P \ NP$</td>
<td>$NP \rightarrow rods$</td>
</tr>
<tr>
<td>$PP \rightarrow P$</td>
<td></td>
</tr>
</tbody>
</table>

P $\rightarrow with$
Chomsky Normal Form

Step 2: Remove unary rules

S → NP VP
VP → V NP
S → V NP
VP → V NP PP
S → V NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP PP
NP → PP
PP → P NP
PP → P

NP → people
NP → fish
NP → tanks
NP → rods
V → people
V → fish
V → tanks
P → with
Chomsky Normal Form

Step 2: Binarize

S $\rightarrow$ NP VP
VP $\rightarrow$ V NP
S $\rightarrow$ V NP
VP $\rightarrow$ V NP PP
S $\rightarrow$ V NP PP
VP $\rightarrow$ V PP
S $\rightarrow$ V PP
NP $\rightarrow$ NP NP
NP $\rightarrow$ NP PP
NP $\rightarrow$ P NP
PP $\rightarrow$ P NP

NP $\rightarrow$ people
NP $\rightarrow$ fish
NP $\rightarrow$ tanks
NP $\rightarrow$ rods
V $\rightarrow$ people
S $\rightarrow$ people
VP $\rightarrow$ people
V $\rightarrow$ fish
S $\rightarrow$ fish
VP $\rightarrow$ fish
V $\rightarrow$ tanks
S $\rightarrow$ tanks
VP $\rightarrow$ tanks
P $\rightarrow$ with
PP $\rightarrow$ with
Chomsky Normal Form

Step 2: Binarize

S \rightarrow \text{NP } \text{VP}

\text{VP } \rightarrow \text{V } \text{NP}

S \rightarrow \text{V } \text{NP}

\text{VP } \rightarrow \text{V } \text{NP } \text{PP}

S \rightarrow \text{V } \text{NP } \text{PP}

\text{VP } \rightarrow \text{V } \text{PP}

S \rightarrow \text{V } \text{PP}

\text{NP } \rightarrow \text{NP } \text{NP}

\text{NP } \rightarrow \text{NP } \text{PP}

\text{NP } \rightarrow \text{P } \text{NP}

\text{PP } \rightarrow \text{P } \text{NP}

\text{NP } \rightarrow \text{people}

\text{NP } \rightarrow \text{fish}

\text{NP } \rightarrow \text{tanks}

\text{NP } \rightarrow \text{rods}

\text{V } \rightarrow \text{people}

\text{S } \rightarrow \text{people}

\text{VP } \rightarrow \text{people}

\text{V } \rightarrow \text{fish}

\text{S } \rightarrow \text{fish}

\text{VP } \rightarrow \text{fish}

\text{V } \rightarrow \text{tanks}

\text{S } \rightarrow \text{tanks}

\text{VP } \rightarrow \text{tanks}

\text{P } \rightarrow \text{with}

\text{PP } \rightarrow \text{with}
Chomsky Normal Form

Step 2: Binarize

S → NP VP
VP → V NP
S → V NP
VP → V @VP_V
@VP_V → NP PP
S → V @S_V
@S_V → NP PP
VP → V PP
S → V PP
NP → NP NP
NP → V NP
PP → P NP
PP → P NP

NP → people
NP → fish
NP → tanks
NP → rods
V → people
S → people
VP → people
V → fish
S → fish
VP → fish
V → tanks
S → tanks
VP → tanks
P → with
PP → with
Chomsky Normal Form: Source

S → NP VP
VP → V NP
VP → V NP PP
NP → NP NP
NP → NP PP
NP → N
NP → e
PP → P NP

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with
Chomsky Normal Form

S → NP VP
VP → V NP
S → V NP
VP → V @VP_V
@VP_V → NP PP
S → V @S_V
@S_V → NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP PP
NP → P NP
PP → P NP

NP → people
NP → fish
NP → tanks
NP → rods
V → people
S → people
VP → people
V → fish
S → fish
VP → fish
V → tanks
S → tanks
VP → tanks
P → with
PP → with
Chomsky Normal Form

• You should think of this as a transformation for efficient parsing
• With some extra book-keeping in symbol names, you can even reconstruct the same trees with a detransform
• In practice full Chomsky Normal Form is a pain
  – Reconstructing n-aries is easy
  – Reconstructing unaries/empties is trickier

• Binarization is crucial for cubic time CFG parsing

• The rest isn’t necessary; it just makes the algorithms cleaner and a bit quicker
Treebank: empties and unaries

PTB Tree

NoFuncTags

NoEmpties

High

Low

NoUnaries