CS5740: Natural Language Processing
Spring 2017

Constituency Parsing

Instructor: Yoav Artzi

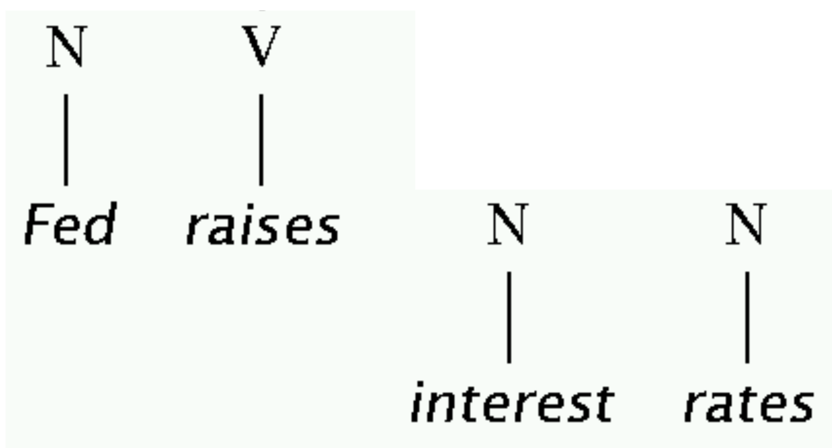
Slides adapted from Dan Klein, Dan Jurafsky, Chris Manning, Michael Collins, Luke Zettlemoyer, Yejin Choi, and Slav Petrov

Overview

- The constituency parsing problem
- CKY parsing
  - Chomsky Normal Form
- The Penn Treebank

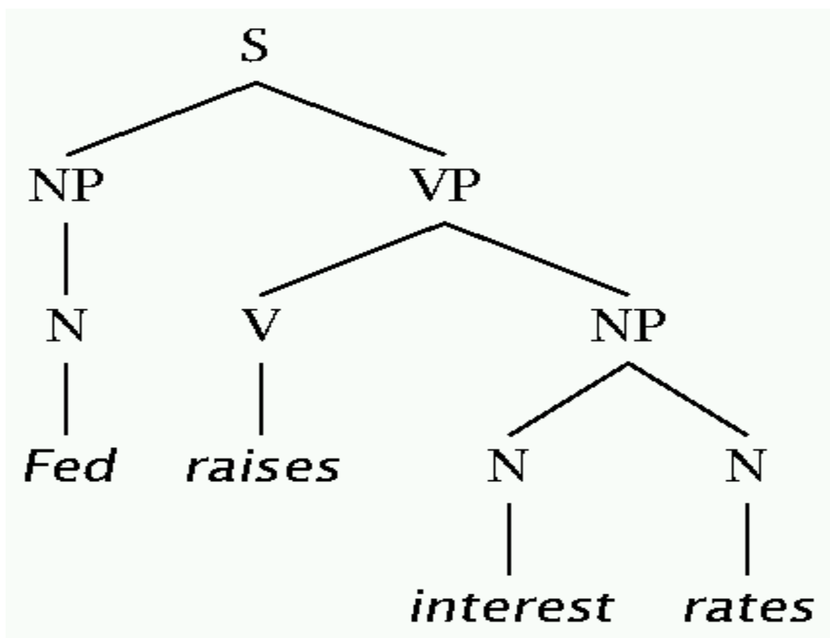
Constituency (Phrase Structure) Trees

- 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

- <u>Distribution:</u> 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

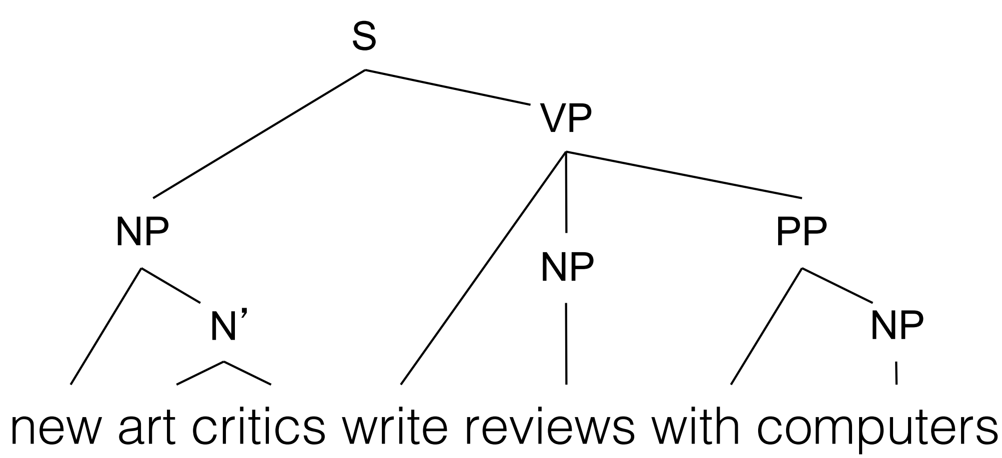
- Substitution/expansion/pro-forms:
  - I sat near the table
  - I sat [on the box/right on top of the box/there].

Constituency Tests

- Distribution / movement / dislocation
- Substitution by pro-form
  - he, she, it, they, ...
- Question / answer
- Deletion
- Conjunction / coordination

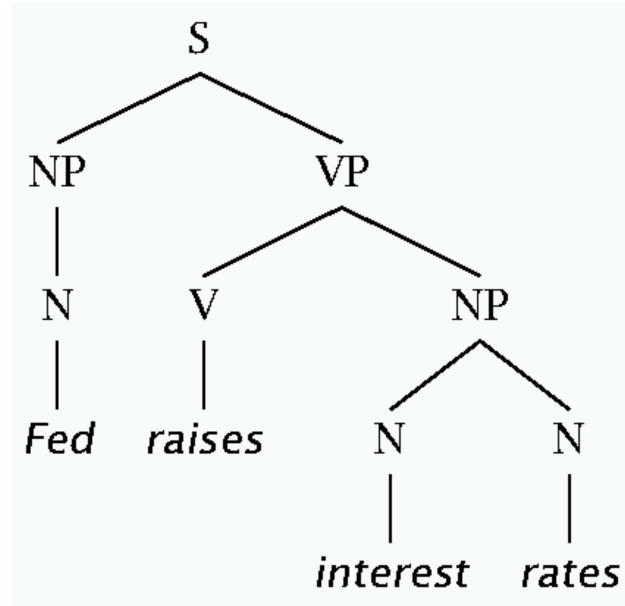
Constituency (Phrase Structure) Trees

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



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



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
  - Σ : 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 $X \rightarrow Y_1\ Y_2 \dots Y_n$, with $X \in N$, $n \geq 0$, $Y_i \in (N \cup \Sigma)$
    - Examples: S → NP VP,   VP → VP CC VP
    - Also called rewrites, productions, or local trees

Example Grammar

N = $\{$S, NP, VP, PP, DT, Vi, Vt, NN, IN$\}$
S = S
Σ = $\{$sleeps, saw, man, woman, telescope, the, with, in$\}$

$R =$

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

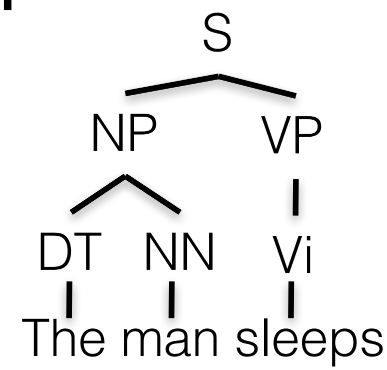
S=sentence, VP-verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

Example Parse

$R =$

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



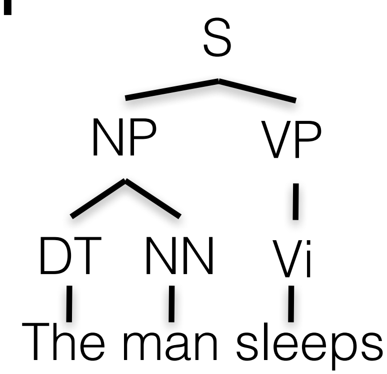
S=sentence, VP-verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

Example Parse

$R =$

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



The man saw the woman with the telescope

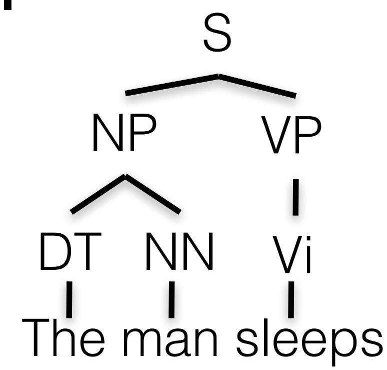
S=sentence, VP-verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

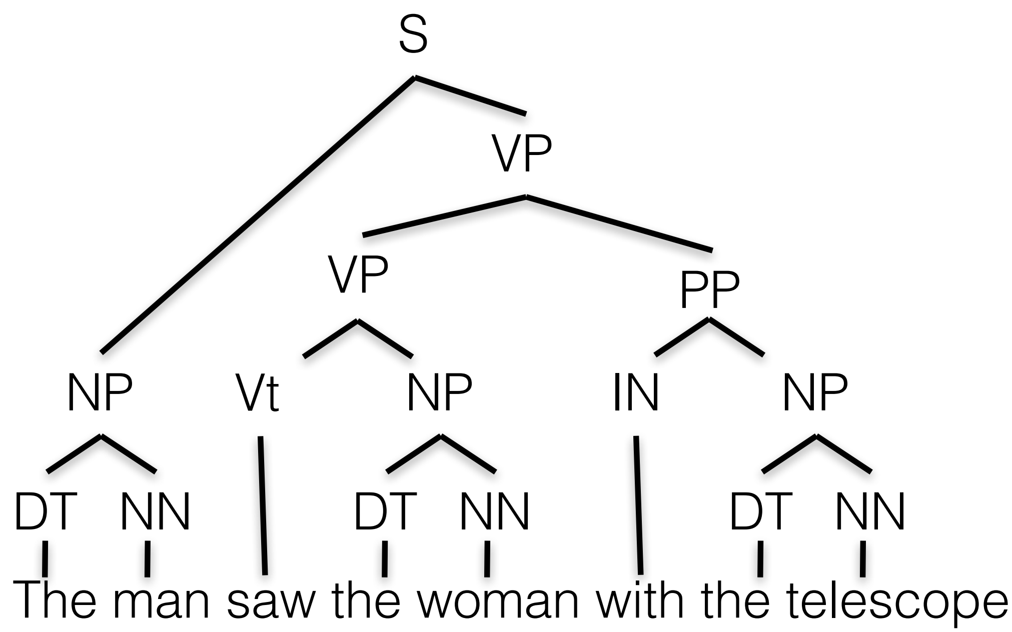
Example Parse

$R =$

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





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 → … VB* …
  - NP → … NN* …
  - ADJP → … JJ* …
  - ADVP → … RB* …
- 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 → NP VP	NN → *interest*
NP → (DT) NN	NNS → *rates*
NP → NN NNS	NNS → *raises*
NP → NNP	VBP → *interest*
VP → V NP	VBZ → *rates*

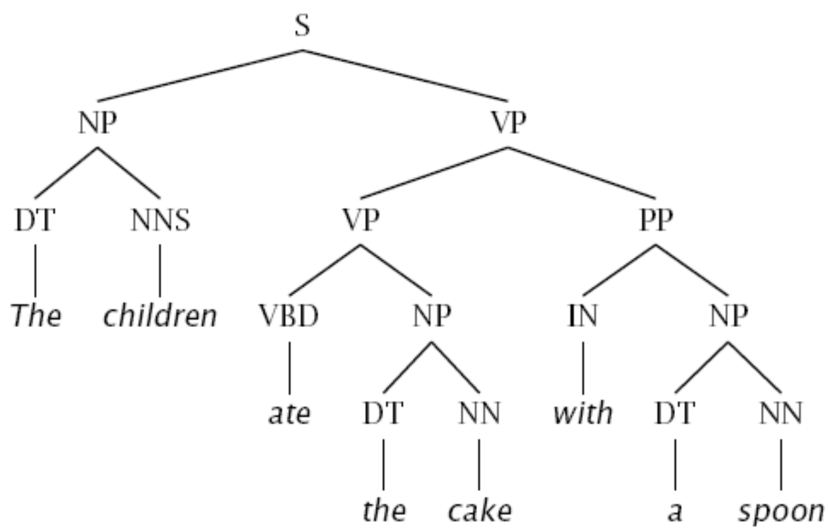
- Used grammar/proof systems to prove parses from words
- This scaled very badly and didn't give coverage. For sentence:

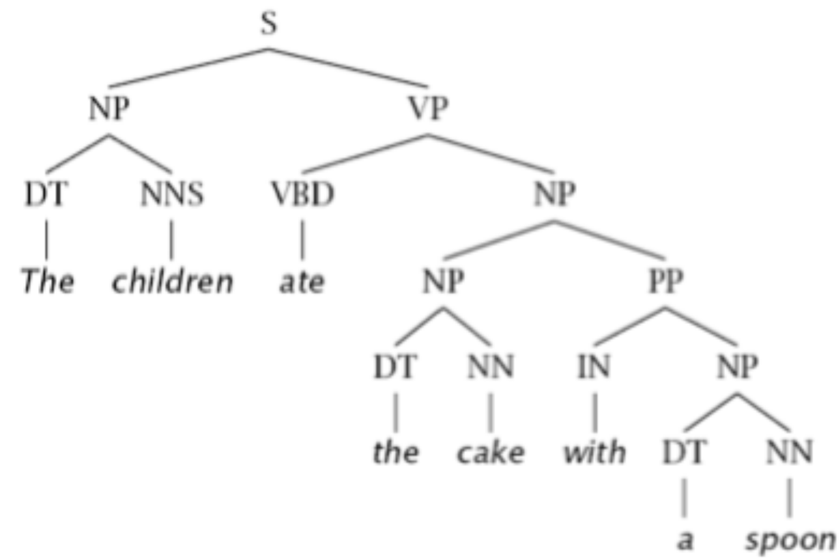
  Fed raises interest rates 0.5% in effort to control inflation

  – 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 make 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)))))))
   (. .)))
```

[Marcus et al. 1993]

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

PTB Non-terminals

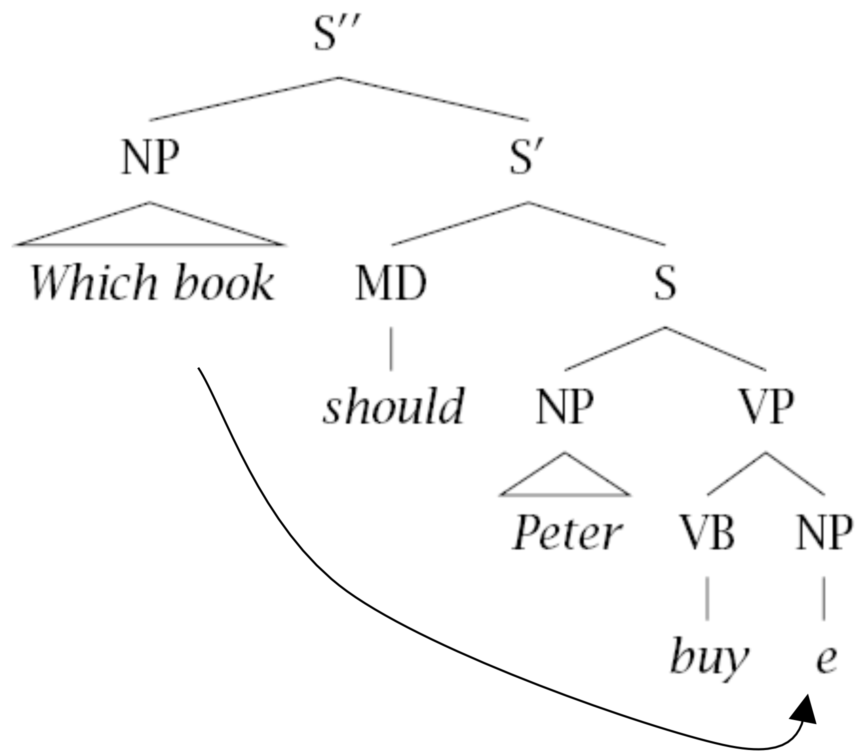
Table 1.2. The Penn Treebank syntactic tagset

ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	Simple declarative clause
SBAR	Subordinate clause
SBARQ	Direct question introduced by *wh*-element
SINV	Declarative sentence with subject-aux inversion
SQ	Yes/no questions and subconstituent of SBARQ excluding *wh*-element
VP	Verb phrase
WHADVP	Wh-adverb phrase
WHNP	Wh-noun phrase
WHPP	Wh-prepositional phrase
X	Constituent of unknown or uncertain category
*	"Understood" subject of infinitive or imperative
0	Zero variant of *that* in subordinate clauses
T	Trace of wh-Constituent

+ all POS tags

Non Local Phenomena

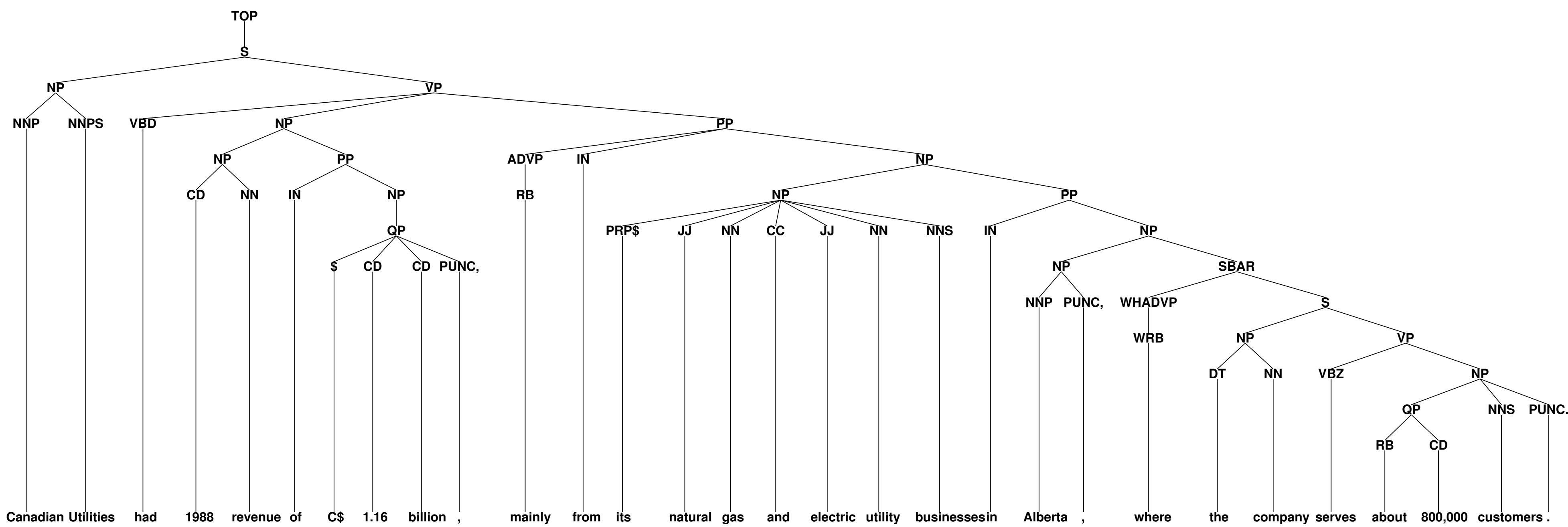
- Dislocation / gapping
  - Which book should Peter buy?
  - A debate arose which continued until the election.



- 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***, 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
  - 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 → NP VP, VP → VP CC VP
    - Also called rewrites, productions, or local trees
- 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

S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	woman	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

- 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

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5

The man sleeps

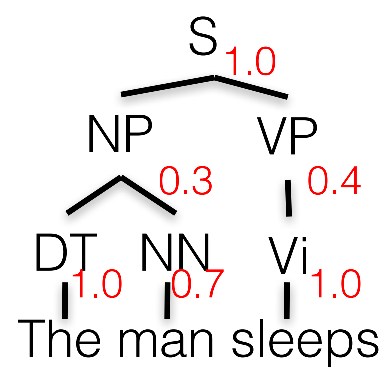
The man saw the woman with the telescope

PCFG Example

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

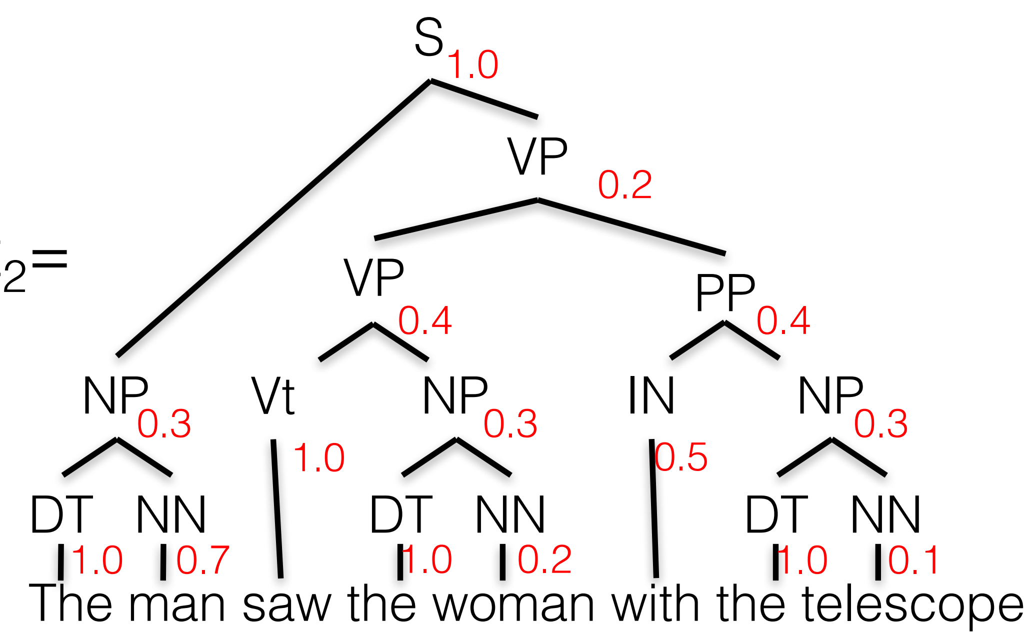
Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5

$t_1 =$



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

$t_2 =$



$p(t_s)=1.0*0.3*1.0*0.7*0.2*0.4*1.0*0.3*1.0*0.2*0.4*0.5*0.3*1.0*0.1$

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

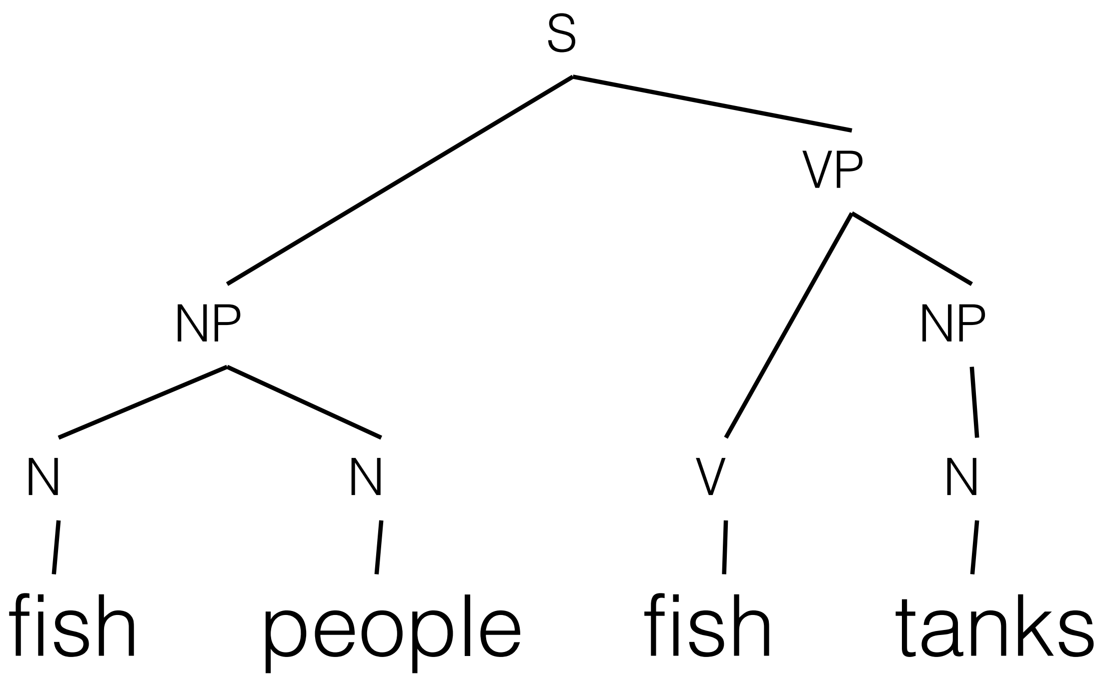
$$q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

  - and use all of our standard smoothing tricks!

- Inference
  - For input sentence s, define T(s) to be the set of trees whose *yield* is s (whose leaves, read left to right, match the words in s)

$$t^*(s) = \arg \max_{t \in \mathcal{T}(s)} p(t)$$

The Constituency Parsing Problem



PCFG

Rule Prob $\boldsymbol{\theta_i}$

$S \rightarrow NP\ VP \quad \theta_0$

$NP \rightarrow NP\ NP \quad \theta_1$

…

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

$N \rightarrow people\theta_{43}$

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

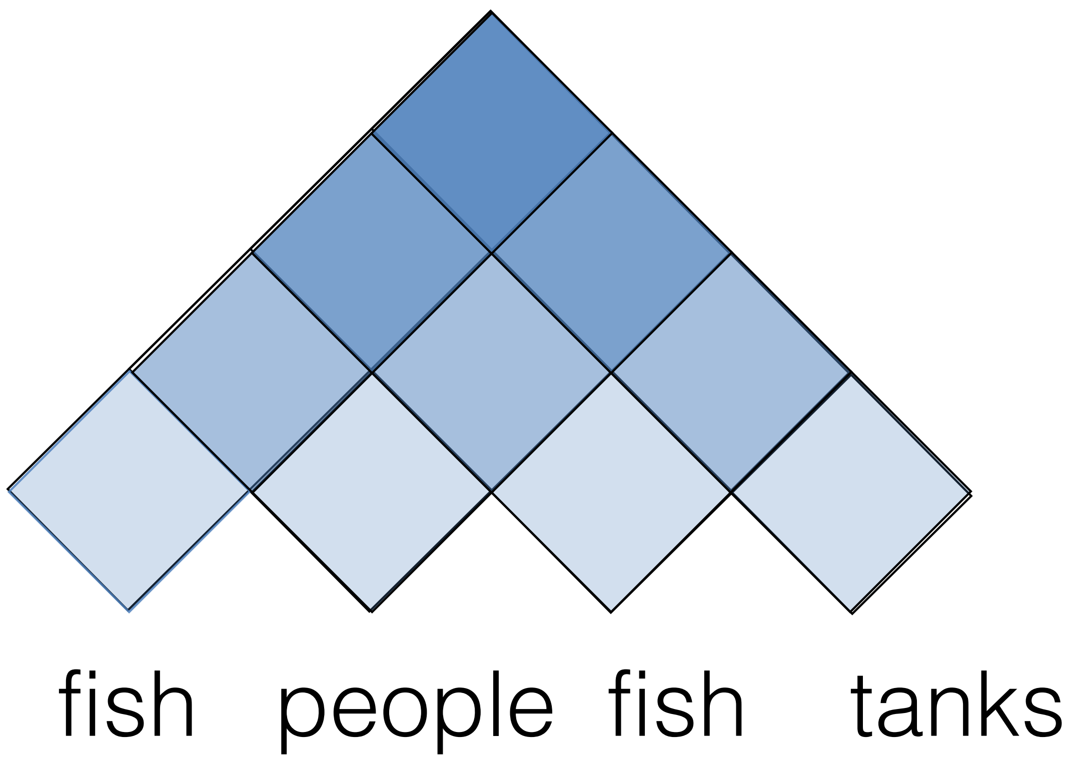
…

A Recursive Parser

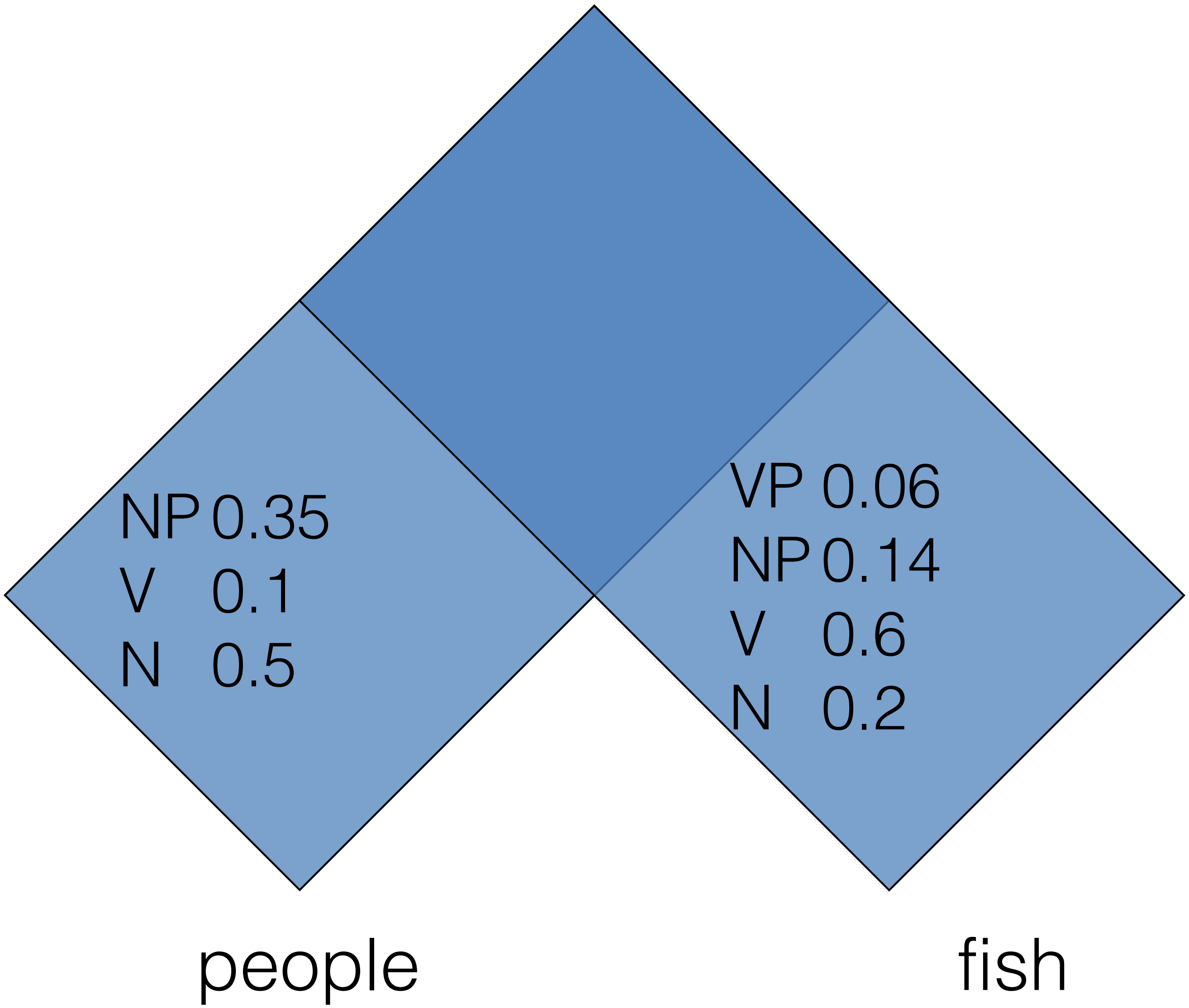
```
bestScore(X,i,j,s)
  if (j == i)
      return q(X->s[i])
  else
      return max q(X->YZ) *
                      bestScore(Y,i,k,s) *
                      bestScore(Z,k+1,j,s)
```

- 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

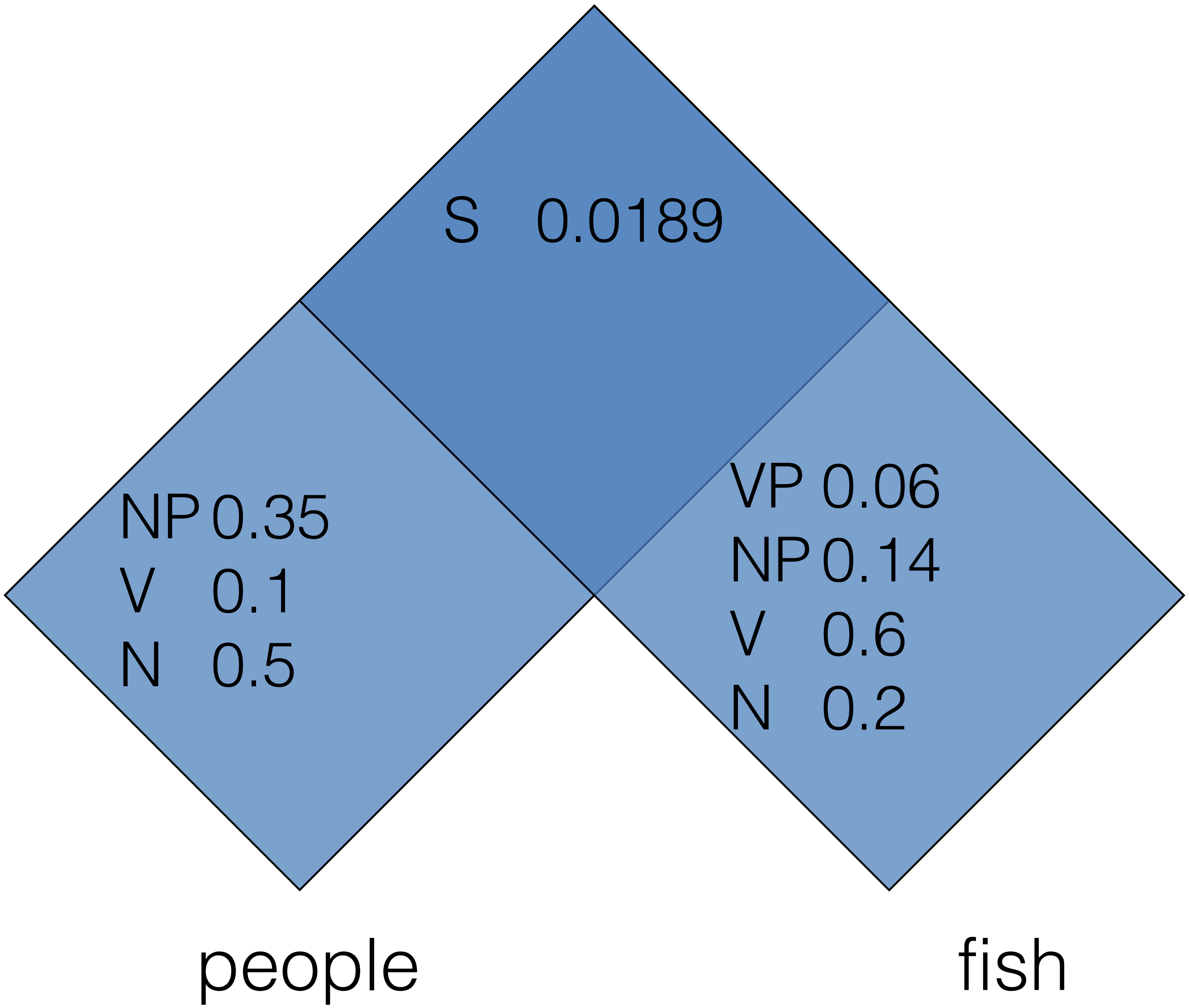


Cocke-Kasami-Younger (CKY) Constituency Parsing



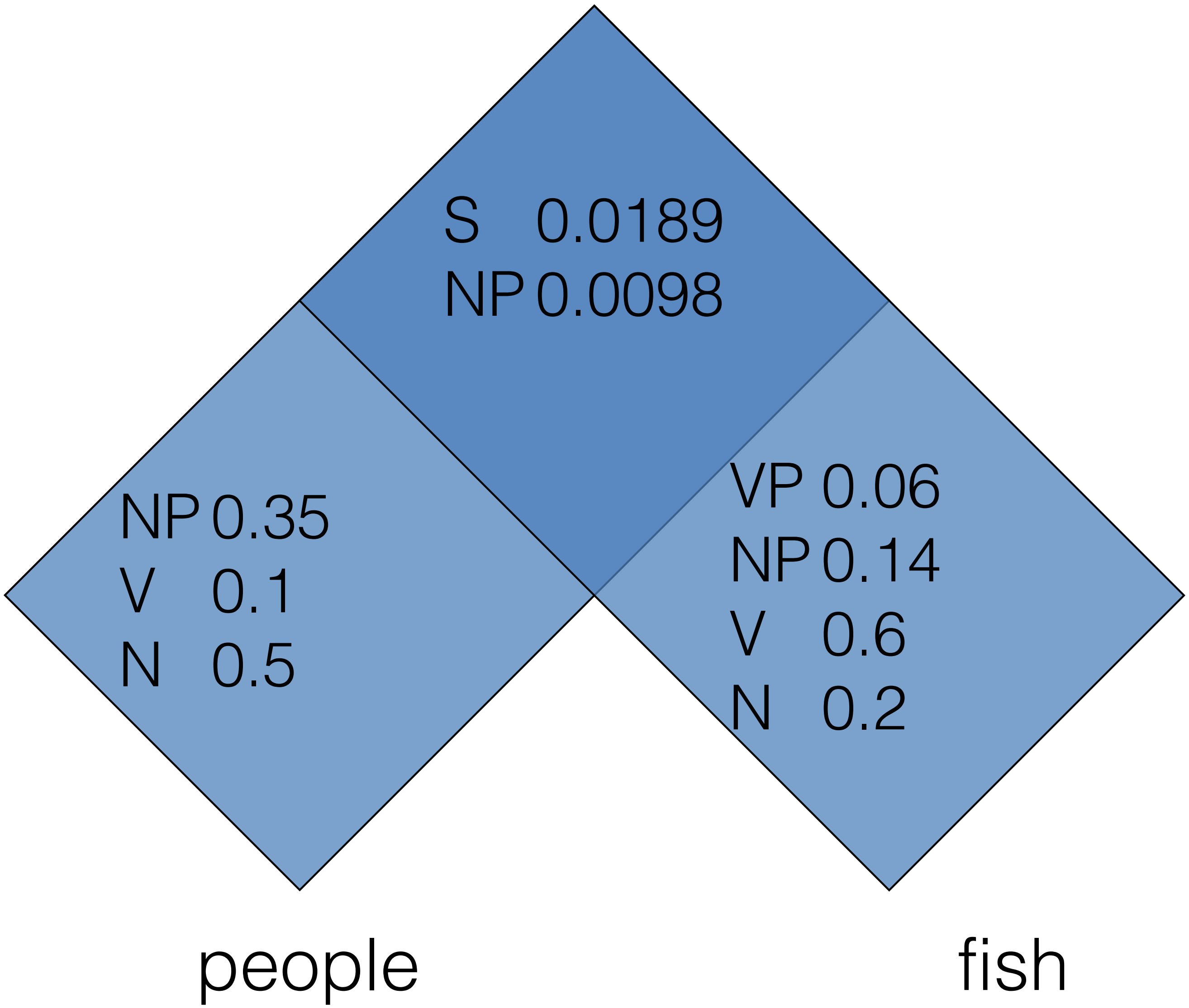
S → NP VP	0.9
VP → V NP	0.5
VP → V @VP_V	0.3
VP → V PP	0.1
@VP_V → NP PP	1.0
NP → NP NP	0.1
NP → NP PP	0.2
PP → P NP	1.0

Cocke-Kasami-Younger (CKY) Constituency Parsing



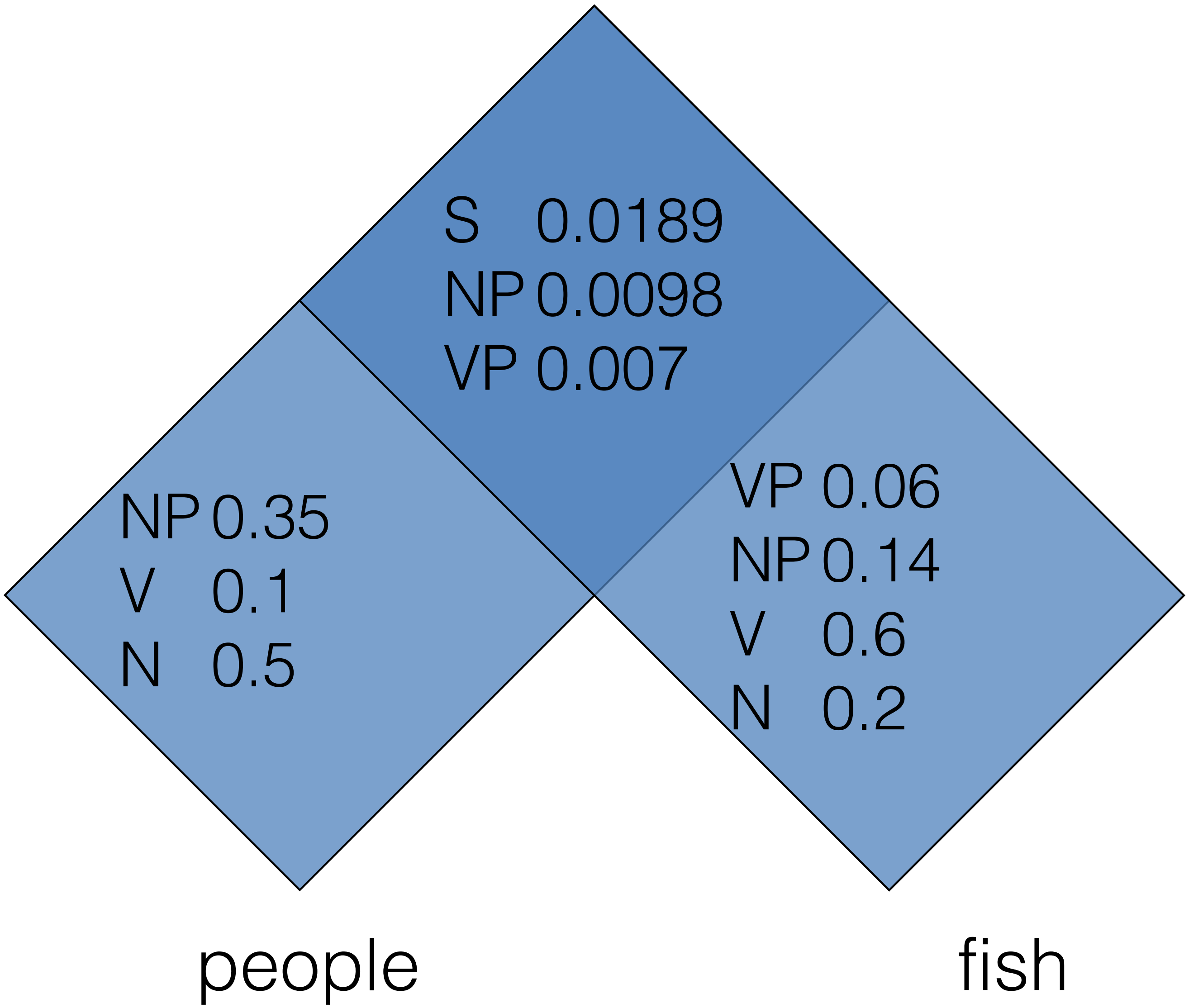
S $\rightarrow$ NP VP	**0.9**
VP $\rightarrow$ V NP	0.5
VP $\rightarrow$ V @VP_V	0.3
VP $\rightarrow$ V PP	0.1
@VP_V $\rightarrow$ NP PP	1.0
NP $\rightarrow$ NP NP	0.1
NP $\rightarrow$ NP PP	0.2
PP $\rightarrow$ P NP	1.0

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

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

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:

$$\pi(1, n, S) = \arg\max_t \in \mathcal{T}_G(x)$$

- Via the recursion:

$$\pi(i, j, X) =$$

- With base case:

$$\pi(i, i, X) \quad =$$

The CKY Algorithm

- Input: a sentence s = x_1 .. x_n and a PCFG = <*N, Σ ,S, R, q*>
- Initialization: For i = 1 … 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 … (n-1) [iterate all phrase lengths]
  - For i = 1 … (n-l) and j = i+l [iterate all phrases of length l]
    - For all X in N [iterate all non-terminals]

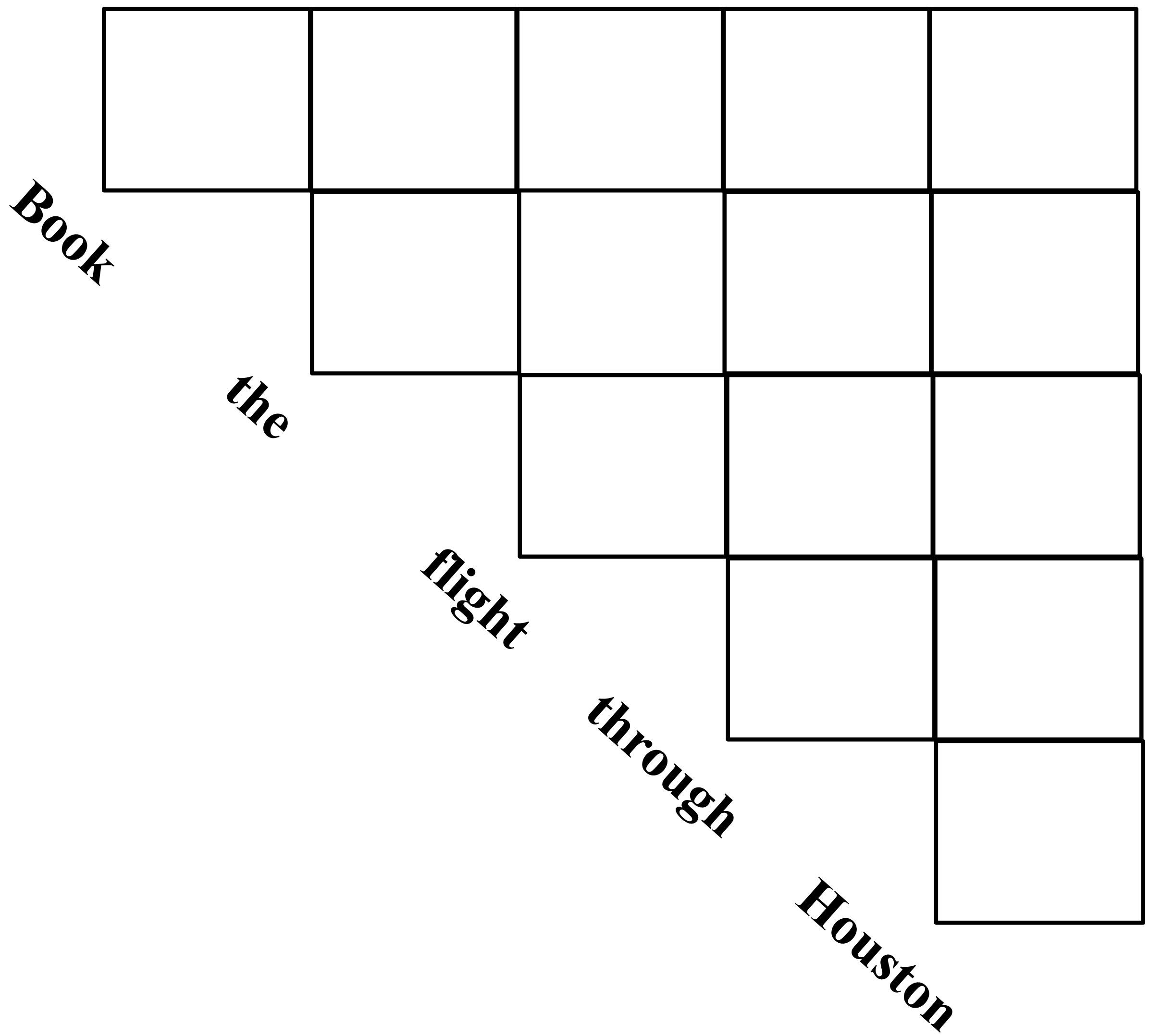
$$\pi(i,j,X) = \max_{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \dots (j-1)\}}} \left(q(X \rightarrow Y Z) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

    - also, store back pointers

$$bp(i,j,X) = \arg \max_{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \dots (j-1)\}}} \left(q(X \rightarrow Y Z) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

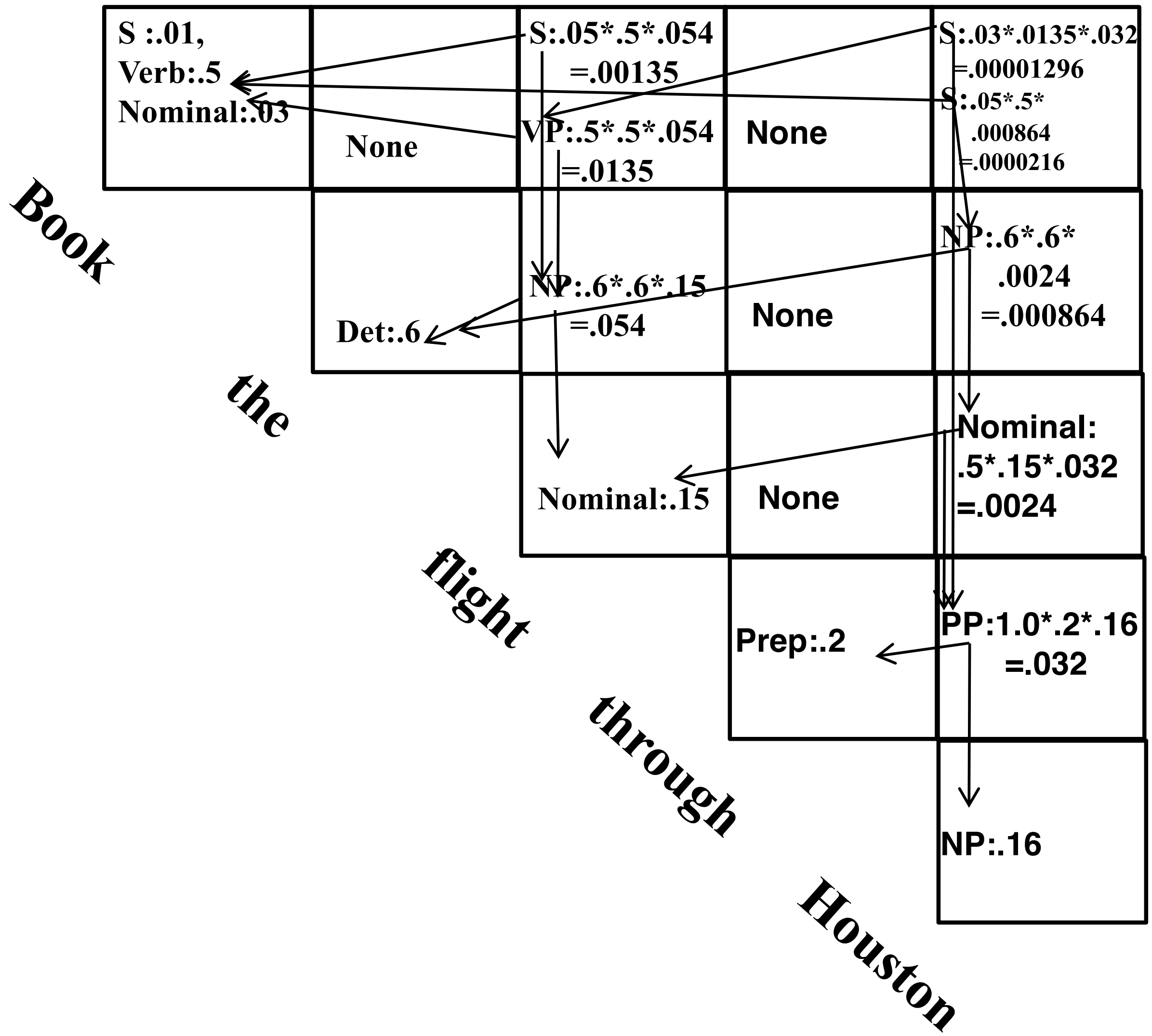
Probabilistic CKY Parser

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

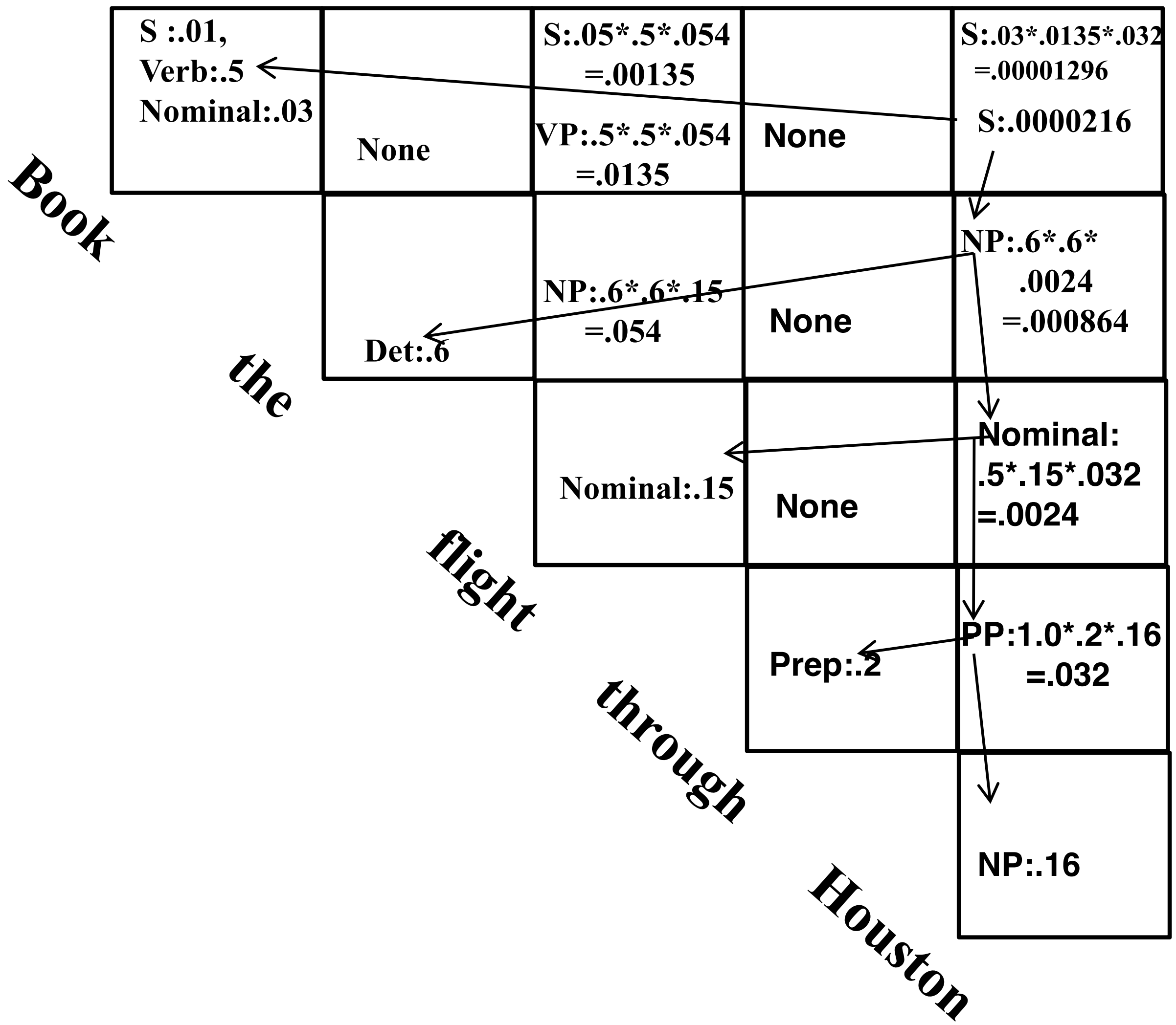


Probabilistic CKY Parser

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

The CKY Algorithm

- Input: a sentence s = x_1 .. x_n and a PCFG = <*N, Σ ,S, R, q*>
- Initialization: For i = 1 … 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 … (n-1) [iterate all phrase lengths]
  - For i = 1 … (n-l) and j = i+l [iterate all phrases of length l]
    - For all X in N [iterate all non-terminals]

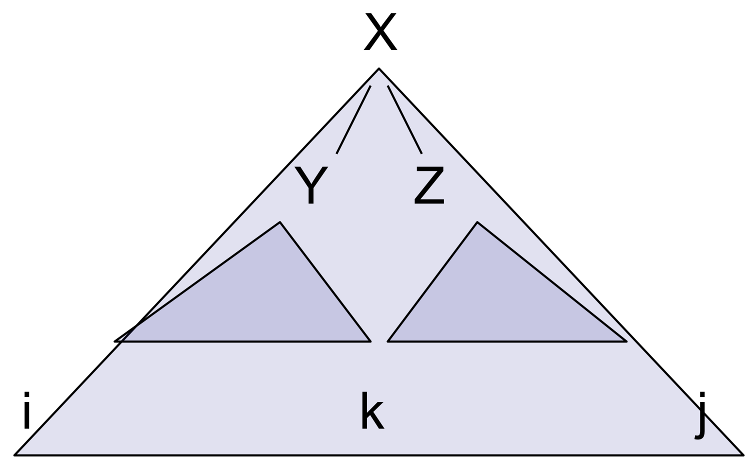
$$\pi(i,j,X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

    - also, store back pointers

$$bp(i,j,X) = \arg \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

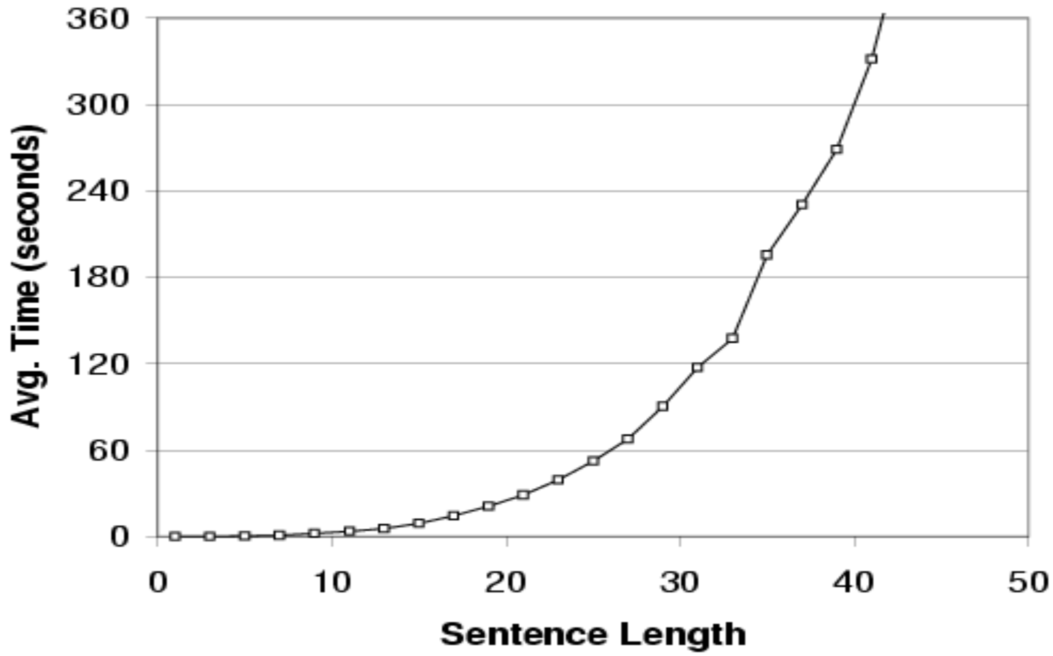
Time: Theory

- For each length (<= n)
  - For each i (<= n)
    - For each split point k
      - For each rule X → Y Z
        - » Do constant work
- Total time: |rules|*n^3



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

The CKY Algorithm

- Input: a sentence s = x_1 .. x_n and a PCFG = <*N*, Σ ,*S*, *R*, *q*>
- Initialization: For i = 1 … 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 … (n-1) [iterate all phrase lengths]
  - For i = 1 … (n-l) and j = i+l [iterate all phrases of length l]
    - For all X in N [iterate all non-terminals]

$$\pi(i,j,X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

    - also, store back pointers

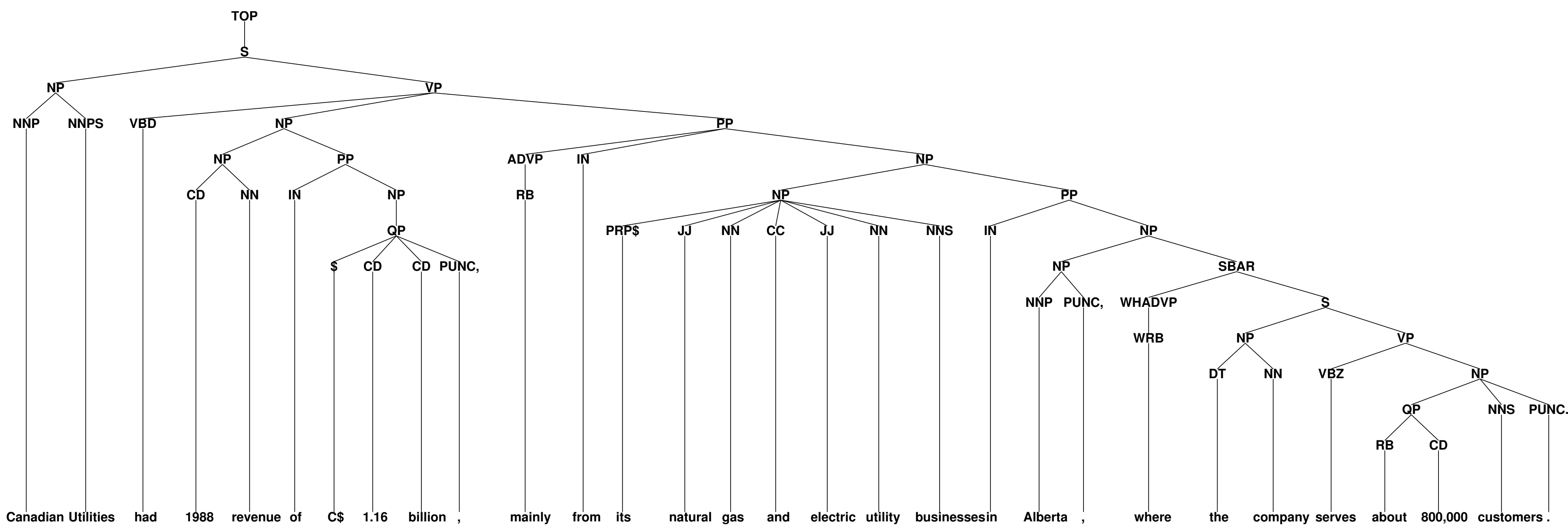
$$bp(i,j,X) = \arg \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

Memory

- How much memory does this require?
  – Have to store the score cache
  – Cache size:
    - |symbols|*n^2 doubles
- 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] – Exact?
- Pruning: Coarse-to-Fine
  – Use a smaller grammar to rule out most X[i,j]

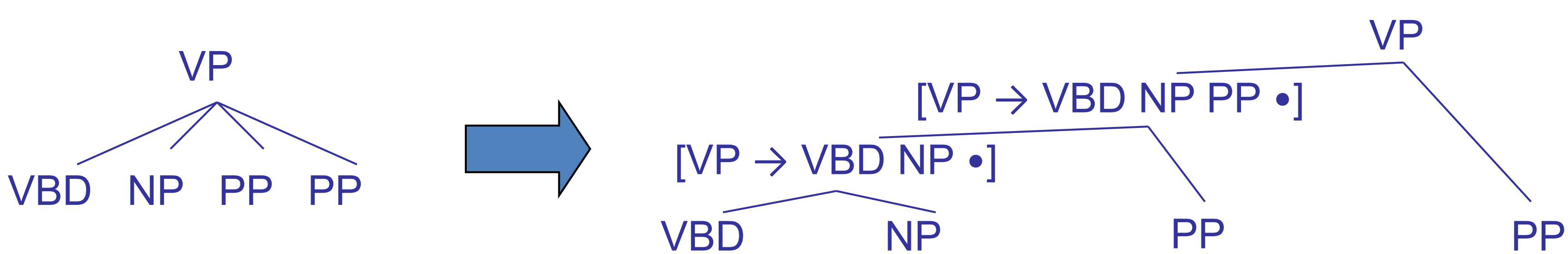
Let's parse with CKY!

- Any problem?



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 → Y Z or X → w
  - Makes parsing easier!
- Can also allow unary rules
  - All rules of the form X → Y Z, X → Y, or X → 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, Σ ,S, R, q>
- Initialization: For i = 1 … 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 … (n-1) [iterate all phrase lengths]
  - For i = 1 … (n-l) and j = i+l [iterate all phrases of length l]
    - Step 1: (Binary)
      - For all X in N [iterate all non-ter

$$\pi_B(i,j,X) = \max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)\}} (q(\quad Z)$$

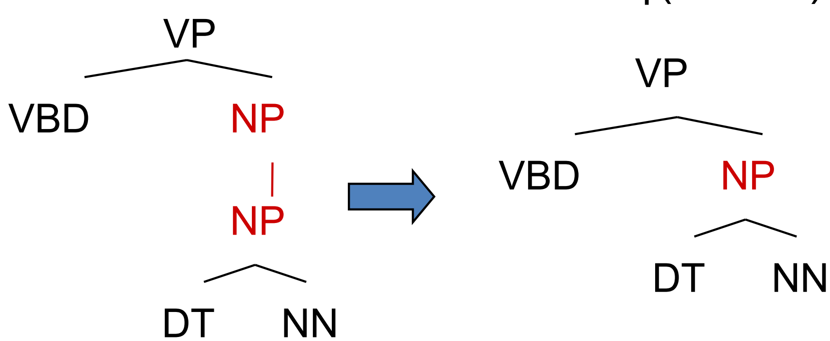
    - Step 2: (Unary)
      - For all X in N [iterate all non-ter

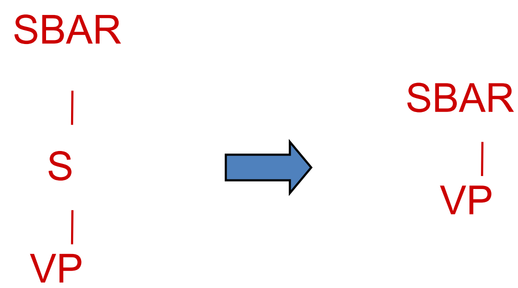
$$\pi_U(i,j,X) = \max_{X \rightarrow Y \in R} (q(X \rightarrow Y) \times \pi_B$$

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 $q(X \rightarrow Y) = q(X \rightarrow Z_1) * q(Z_1 \rightarrow Z_2) * \ldots * q(Z_k \rightarrow Y)$
  - Add $X \rightarrow X$ with $q(X \rightarrow X)=1$ for all X in N





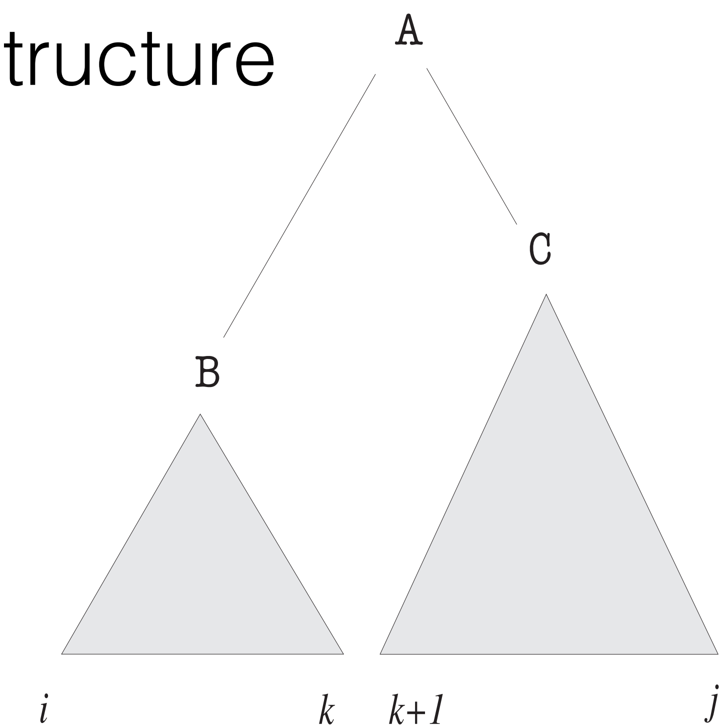
- In CKY and chart: Alternate unary and binary layers
- Reconstruct unary chains afterwards (with extra marking)

Other Chart Computations

- Max inside score
  - Score of the max parse of x_i to x_j with root X

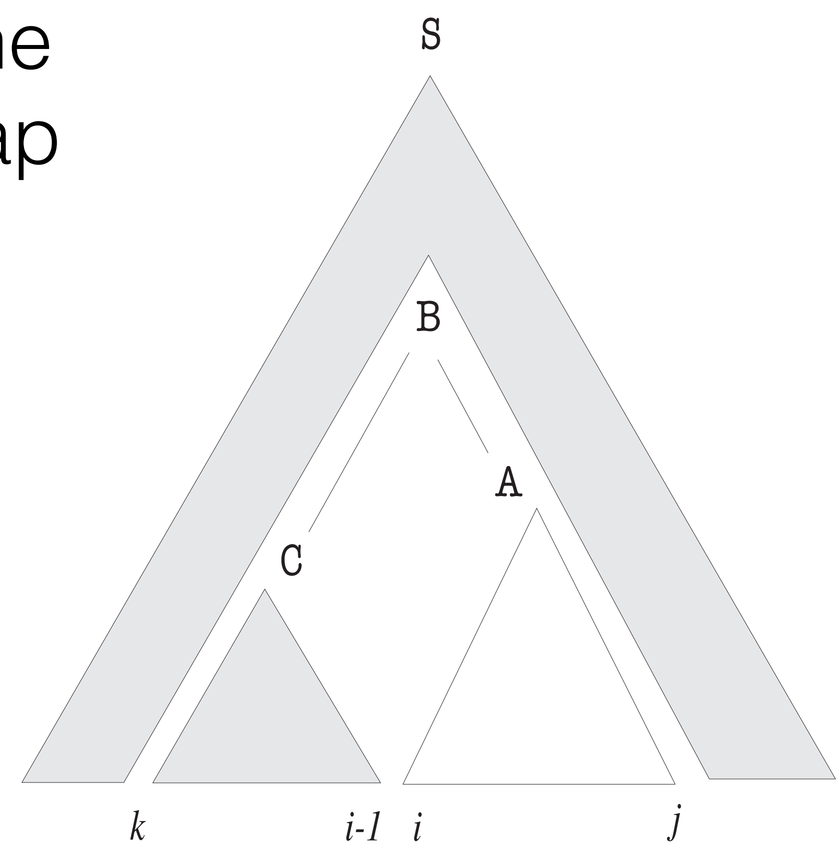
$$\pi(i, j, X)$$

  - Marginalize over internal structure
- Max outside score
- Sum inside/outside



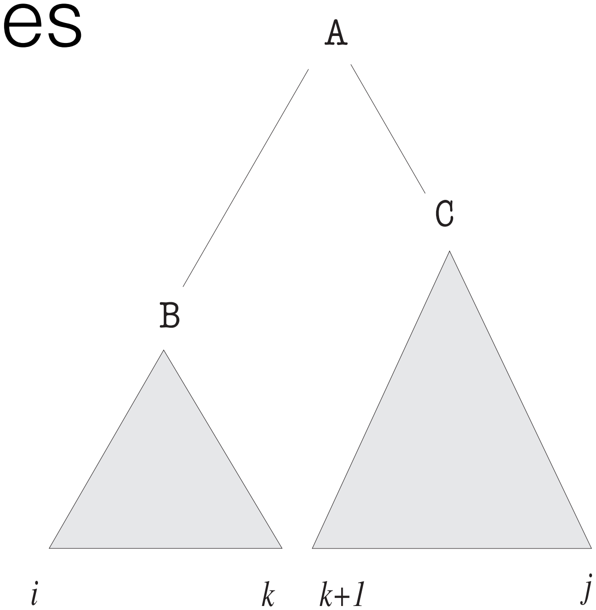
Other Chart Computations

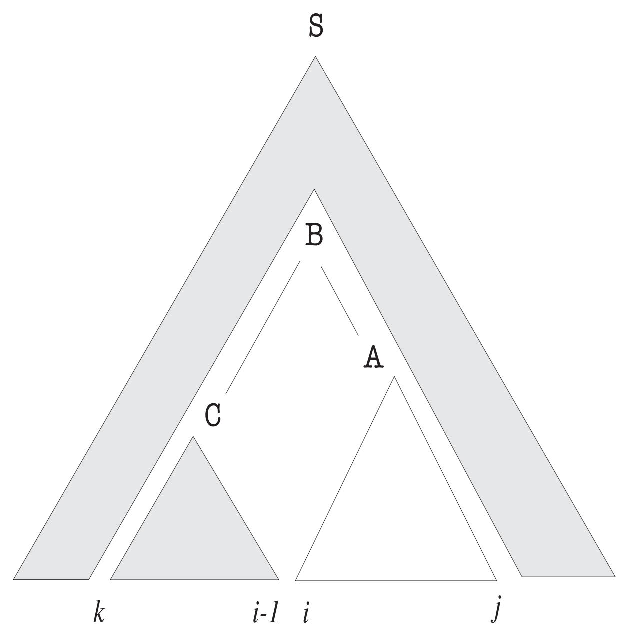
- Max inside score
- Max outside score
  - Score of max parse of the complete span with a gap between i and j
  - Details in notes
- Sum inside/outside



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

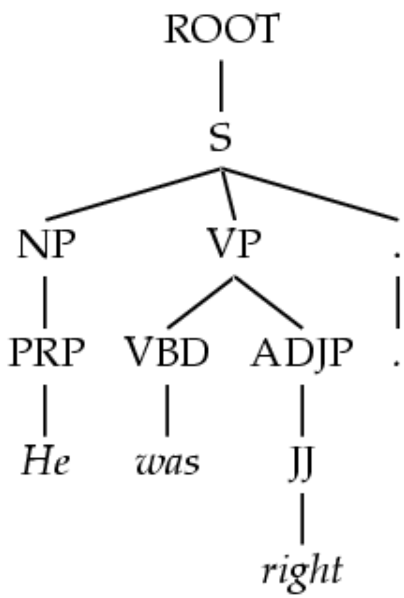
Treebank Parsing

```
( (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:



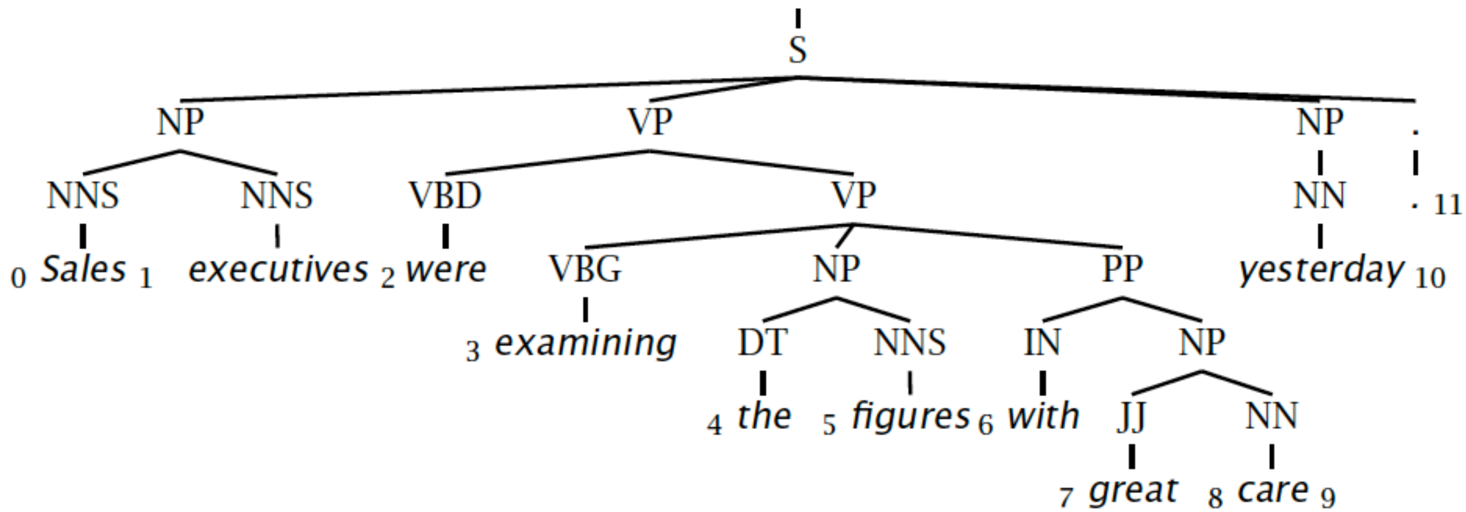
ROOT → S	1
S → NP VP .	1
NP → PRP	1
VP → VBD ADJP	1
…..	

Typical Experimental Setup

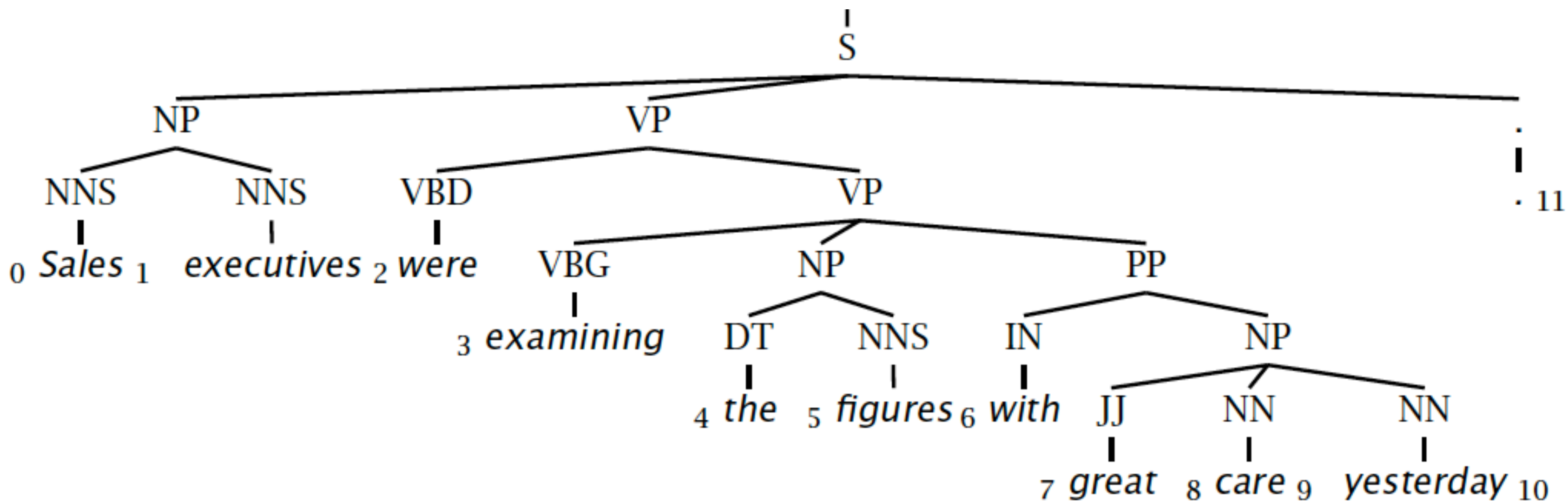
- The Penn Treebank is divided into sections:
  - Training: sections 2-18
  - Development: section 22 (also 0-1 and 24)
  - Testing: section 23
- Evaluation?

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)

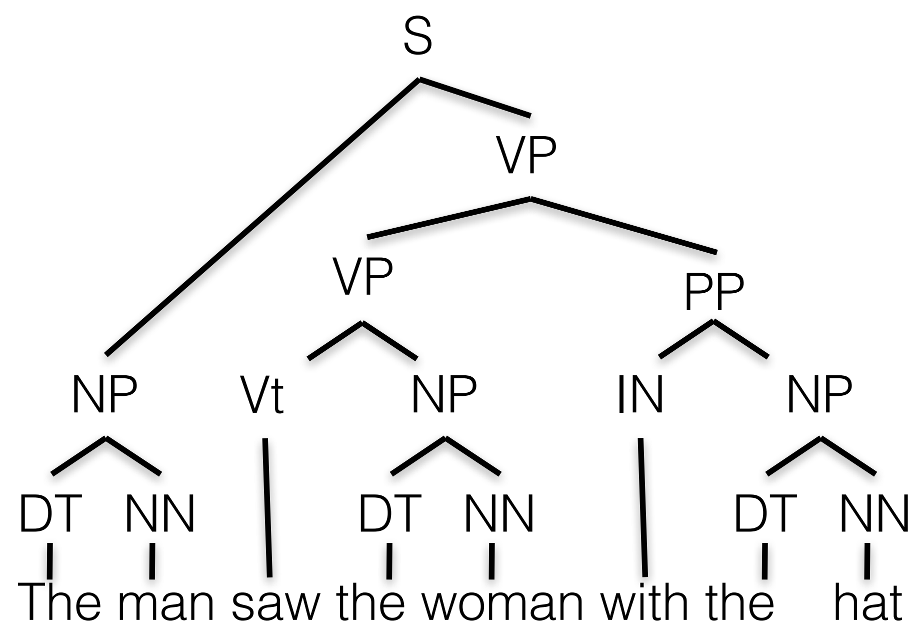
• Precision:	3/7	=	42.9%
• Recall:	3/8	=	37.5%
• F1:			40%
• Also, tagging accuracy:	11/11	=	100%

How Good are PCFGs?

Penn WSJ parsing performance:
~ 73% F1

- 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

The Missing Information?

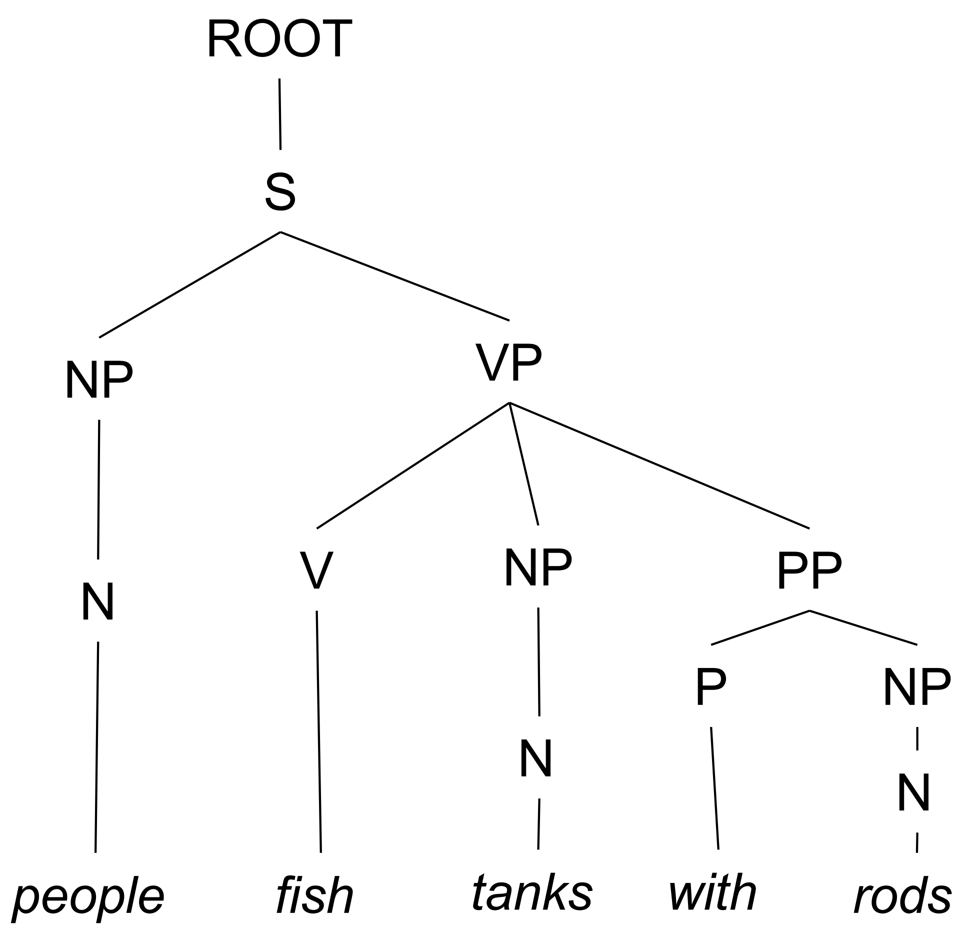


Extra Slides

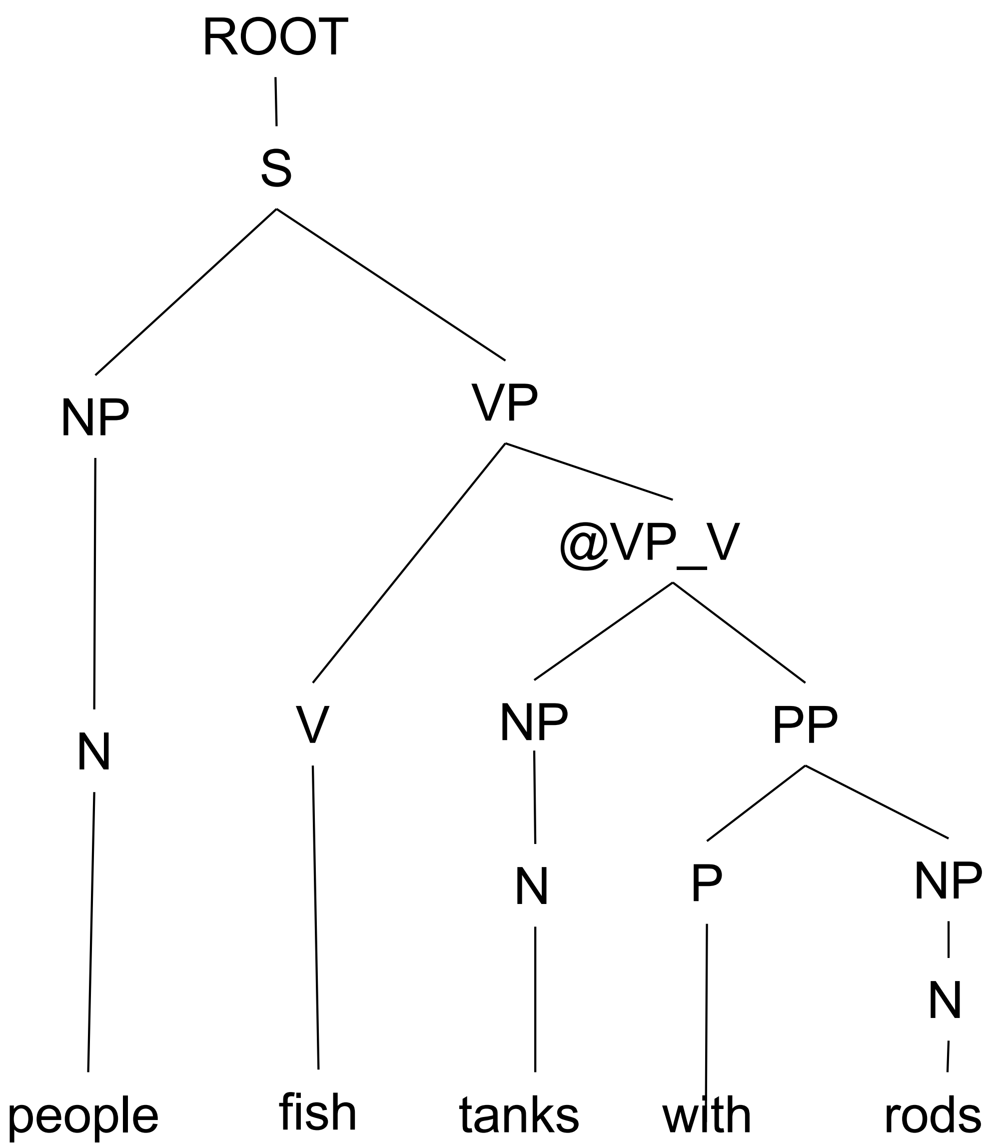
Chomsky Normal Form

- 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



Example: After Binarization



A Phrase Structure Grammar

$S \to NP\ VP$

$VP \to V\ NP$

$VP \to V\ NP\ PP$

$NP \to NP\ NP$

$NP \to NP\ PP$

$NP \to N$

$NP \to e$

$PP \to P\ NP$

$N \to$ *people*

$N \to$ *fish*

$N \to$ *tanks*

$N \to$ *rods*

$V \to$ *people*

$V \to$ *fish*

$V \to$ *tanks*

$P \to$ *with*

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

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

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

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

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

Chomsky Normal Form

Step 1: Remove epsilon rules

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

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

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*

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*

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

Recognizing the same language? Work your way down to propagate

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
~~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

Recognizing the same language? Work your way down to propagate

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

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

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
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 → 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
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 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

Recognizing the same language? Yes!

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 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

Only place N appears
So can get rid of it altogether

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 $\rightarrow$ 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 $\rightarrow$ PP
PP $\rightarrow$ P NP
PP $\rightarrow$ P

NP $\rightarrow$ *people*
NP $\rightarrow$ *fish*
NP $\rightarrow$ *tanks*
NP $\rightarrow$ *rods*
V $\rightarrow$ *people*
V $\rightarrow$ *fish*
V $\rightarrow$ *tanks*
P $\rightarrow$ *with*

Chomsky Normal Form

Step 2: Remove unary rules

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

$NP \rightarrow$ *people*
$NP \rightarrow$ *fish*
$NP \rightarrow$ *tanks*
$NP \rightarrow$ *rods*
$V \rightarrow$ *people*
$V \rightarrow$ *fish*
$V \rightarrow$ *tanks*
$P \rightarrow$ *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 → 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 → 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

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 → 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: Source

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

$N \rightarrow$ *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

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