Grammars and Constituency Parsing

JURAFSKY AND MARTIN CHAPTERS 12 AND 13

Reminders

QUIZ IS DUE TONIGHT BY 11:59PM

HOMEWORK 7 DUE DATE IS POSTPONED TO 3/18

Formal Definition of a CFG

A context-free grammar *G* is defined by four parameters: N, Σ, R, S

N is a set of **non-terminal symbols** (or variables)

- In NLP, we often use the Penn Treebank tag set

Σ is set of **terminal symbols**

- These are the words (also sometimes called the leaf nodes of the parse tree)

R is a set of production rules, each of the form $A \rightarrow \beta$

- *S → Aux NP VP*
- *Nominal → Nominal Gerund VP* (recursive)

S is the start symbol (a non-terminal)

Where do grammars come from?

Linguists

Write 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*

Noam Chomsky

Ivan Sag

Joan Bresnan

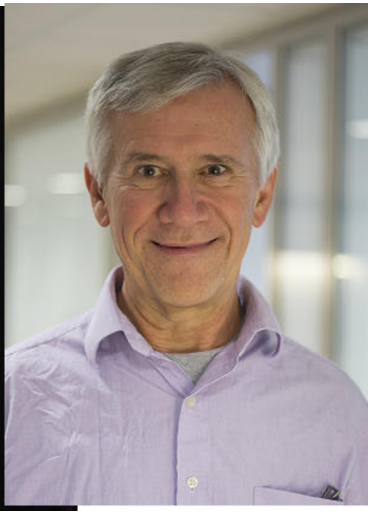
Coverage of grammars

Manually written grammars have two problems.

1. **Coverage** – the tend to only cover a subset of a language, since actual language use is widely varied and hugely complex. Therefore writing a broad coverage grammar by hand takes an entire career.

2. **Overgeneration** – writing rules that will only generate grammatical sentences is hard. Broad coverage tends to come at the expense of overgeneration.


Ann Copestake


Dan Flickinger


Mark Steedman


Arvind Joshi

Treebanks as grammar

Treebanks == data

Initially, building a treebank might seem like it would be a lot slower and less useful than building a grammar.

However, 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

Mitch Marcus

[Marcus et al. 1993, *Computational Linguistics*] penn treebank

Scholar About 24,900 results (**0.07** sec)

Building a large annotated corpus of English: The **Penn Treebank**

M Marcus, B Santorini, MA Marcinkiewicz - 1993 - repository.upenn.edu

In this paper, we review our experience with constructing one such large annotated corpus-- the **Penn Treebank**, a corpus consisting of over 4.5 million words of American English. During the first three-year phase of the **Penn Treebank** Project (1989-1992), this corpus has …

Cited by 7990 Related articles All 30 versions

Semantic **Scholar**

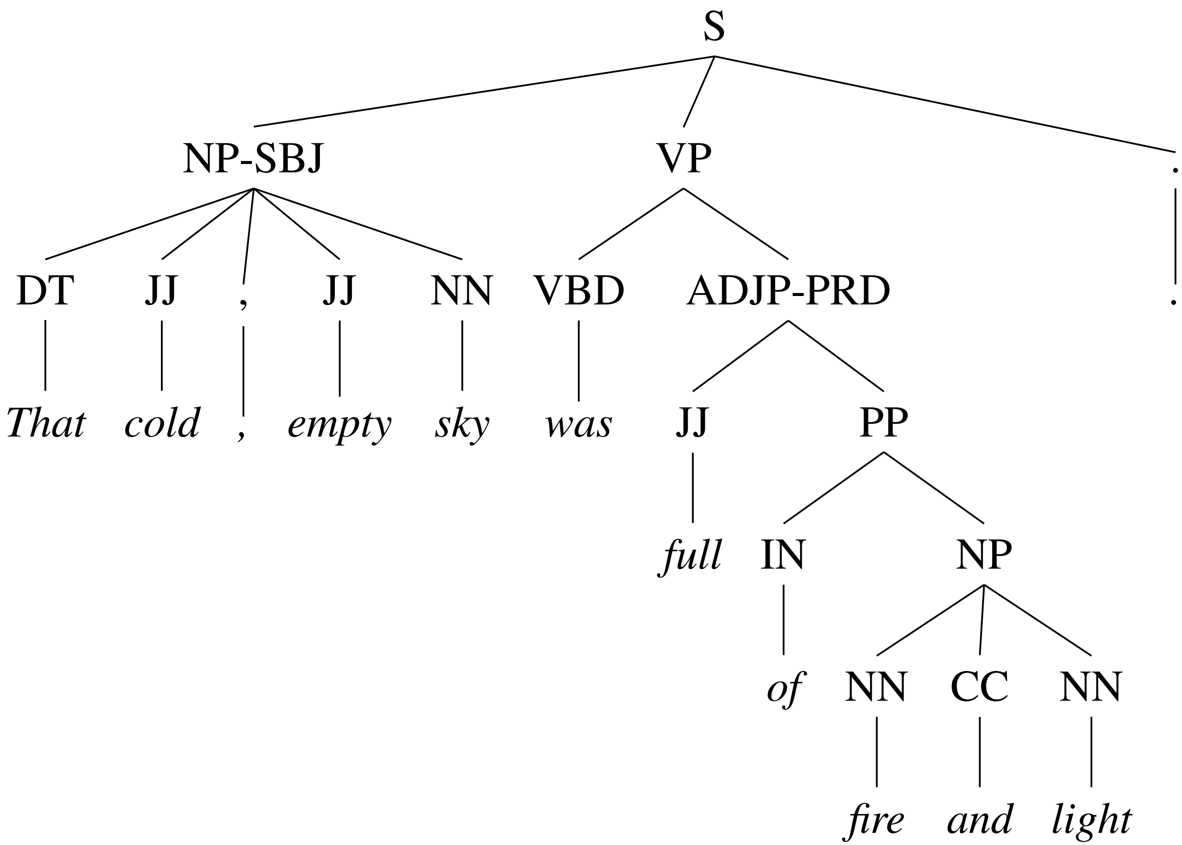
5,604 Citations

1,253 Highly Influenced Papers

3,033 Cite Methods

1,658 Cite Background

50 Cite Results



Extracted rules		
S → NP VP .	DT → That	JJ → full
NP → DT JJ , JJ NN	JJ → cold	IN → of
VP → VBD ADJP	, → ,	NN → fire
ADJP → JJ PP	JJ → empty	CC → and
PP → IN NP	NN → sky	NN → light
NP → NN CC NN	VBD → was	

Some of the rules, with counts

40717 PP → IN NP
33803 S → NP-SBJ VP
22513 NP-SBJ → -NONE-
21877 NP → NP PP
20740 NP → DT NN
14153 S → NP-SBJ VP .
12922 VP → TO VP
11881 PP-LOC → IN NP
11467 NP-SBJ → PRP
11378 NP → -NONE-
11291 NP → NN
...
989 VP → VBG S
985 NP-SBJ → NN
983 PP-MNR → IN NP
983 NP-SBJ → DT
969 VP → VBN VP

100 VP → VBD PP-PRD
100 PRN → : NP :
100 NP → DT JJS
100 NP-CLR → NN
99 NP-SBJ-1 → DT NNP
98 VP → VBN NP PP-DIR
98 VP → VBD PP-TMP
98 PP-TMP → VBG NP
97 VP → VBD ADVP-TMP VP
...
10 WHNP-1 → WRB JJ
10 VP → VP CC VP PP-TMP
10 VP → VP CC VP ADVP-MNR
10 VP → VBZ S , SBAR-ADV
10 VP → VBZ S ADVP-TMP

4500 rules for VP!

Redundant rules?

The Penn Treebank by this series of rules:

$$VP \rightarrow VBD\ NP\ PP$$
$$VP \rightarrow VBD\ NP\ PP\ PP$$
$$VP \rightarrow VBD\ NP\ PP\ PP\ PP$$
$$VP \rightarrow VBD\ NP\ PP\ PP\ PP\ PP$$

We can also represent that with a two-rule grammar as

$$VP \rightarrow VBD\ NP\ PP$$
$$VP \rightarrow VP\ PP$$

NP rules

NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB DT JJ JJ NNS
NP → DT JJ JJ NNP NNS
NP → DT NNP NNP NNP NNP JJ NN
NP → DT JJ NNP CC JJ JJ NN NNS
NP → RB DT JJS NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN
NP → DT JJ JJ VBG NN NNP NNP FW NNP
NP → NP JJ , JJ '' SBAR '' NNS

[DT The] [JJ state-owned] [JJ industrial] [VBG holding] [NN company] [NNP Instituto] [NNP Nacional] [FW de] [NNP Industria]

[NP Shearson's] [JJ easy-to-film], [JJ black-and-white] "[SBAR Where We Stand]" [NNS commercials]

Chomsky normal form

Real rules extracted from the Penn Treebank can get crazy. We can derive an equivalent grammar by converting the rules into binary branching rules.

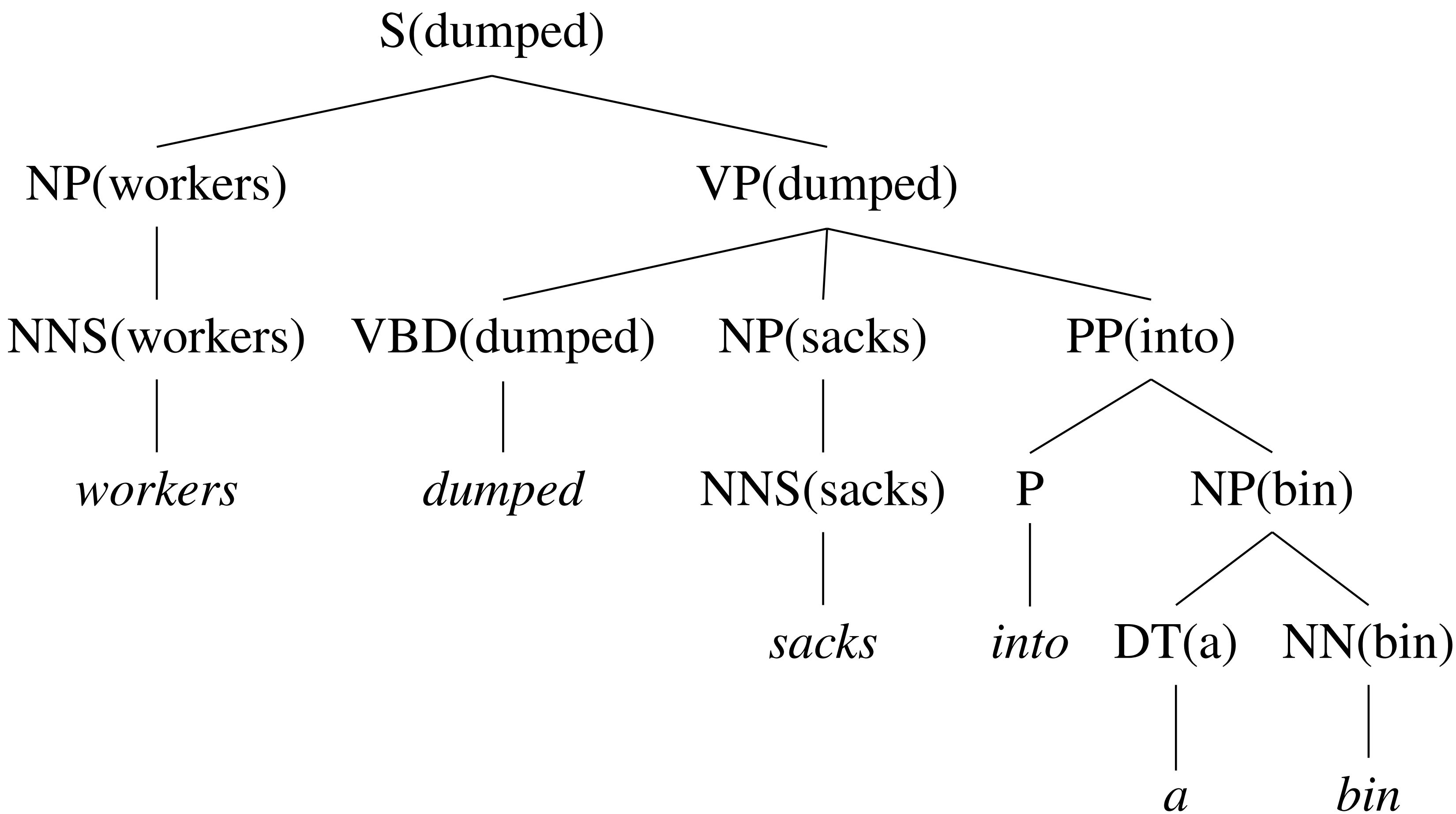
$$A \rightarrow B\ C\ D$$

can be converted to two rules:

$$A \rightarrow B\ X$$
$$X \rightarrow C\ D$$

These are called Chomsky Normal Form rules. The resulting binary branching grammar is **weakly equivalent** because it generates the same set of strings but assigns different phrase structures to sentences.

Lexical Heads



Lexicalized Grammars

Unlike rules derived from the Penn Treebank, which tends to emphases **phrase structure rules**, many modern syntactic theories emphasize the role of the **lexicon.** The lexicon can encode information like subcategorization frames.

1. HPSG – Head-driven phrase structure grammars
2. LFG – Lexical functional grammars
3. TAG – Tree adjoining Grammar
4. CCG – Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG)

A set of categories, a **lexicon that associates words with categories**, and a set of rules that govern how categories combine in context.

word	category	
flight	N	atomic
Miami	NP	atomic
cancel	(S\NP)/NP	complex category / function

Parsing in CCG

United *serves* *Miami*

Rules are simple:

$X/Y \quad Y \;\Rightarrow\; X$ ***forward function application***

$Y \quad X\backslash Y \;\Rightarrow\; X$ ***backward function application***

Parsing in CCG

We	*flew*	*to*	*Geneva*	*and*	*drove*	*to*	*Chamonix*
NP	$(S\backslash NP)/PP$	PP/NP	NP	$CONJ$	$(S\backslash NP)/PP$	PP/NP	NP

Rules are simple:

$X/Y \;\; Y \Rightarrow X$ ***forward function application***

$Y \;\; X\backslash Y \Rightarrow X$ ***backward function application***

$X \; CONJ \; X \Rightarrow X$ **Conjunction**

CCG Bank

Julia Hockenmaier created a treebank for CCG.

It was created by translating phrase-structure trees from the Penn Treebank.

It resulted in 48,934 sentences paired with CCG derivations, and a lexicon with 44,000 words and 1200 categories.

Julia Hockenmaier

Constituency Parsing

JURAFSKY AND MARTIN CHAPTER 13

Headlines

Iraqi Head Seeks Arms

Juvenile Court to Try Shooting Defendant

Teacher Strikes Idle Kids

Stolen Painting Found by Tree

Kids Make Nutritious Snacks

Local High School Dropouts Cut in Half

British Left Waffles on Falkland Islands

Red Tape Holds Up New Bridges

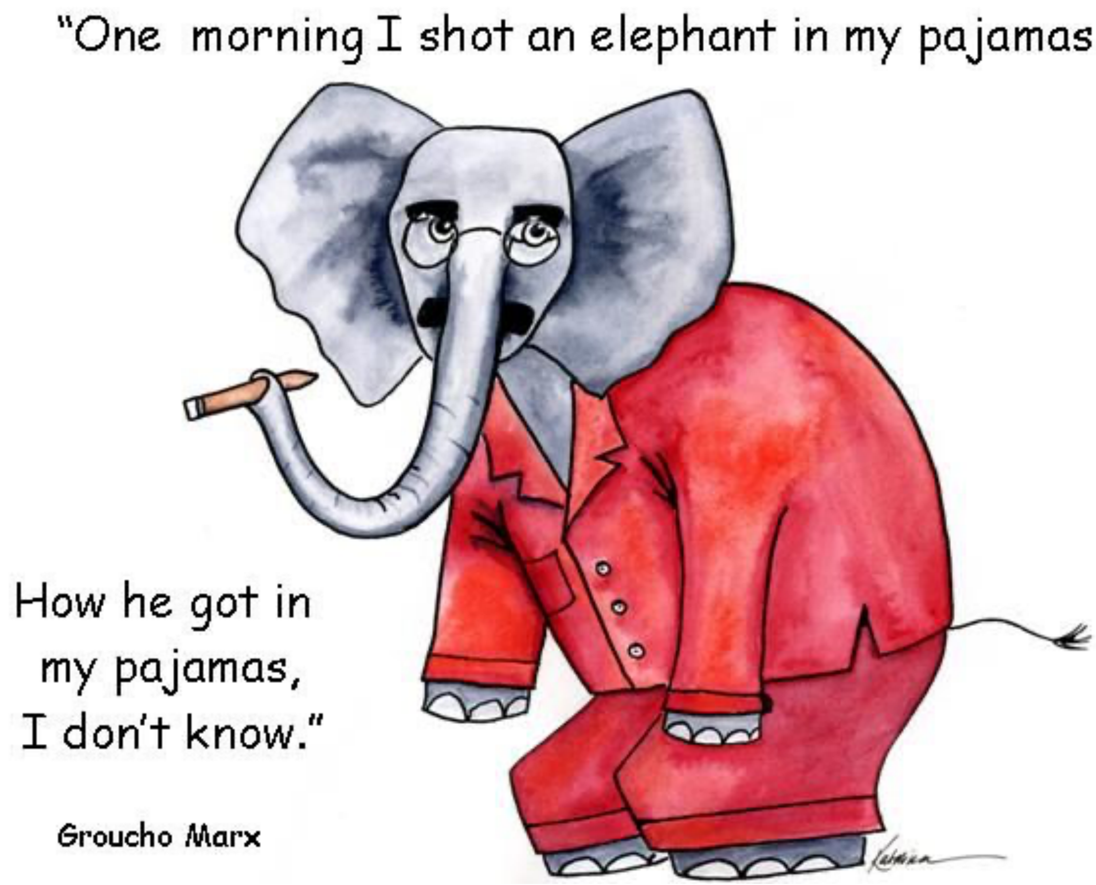
Ban on Nude Dancing on Governor's Desk

Trump Wins on Economy, but More Lies Ahead

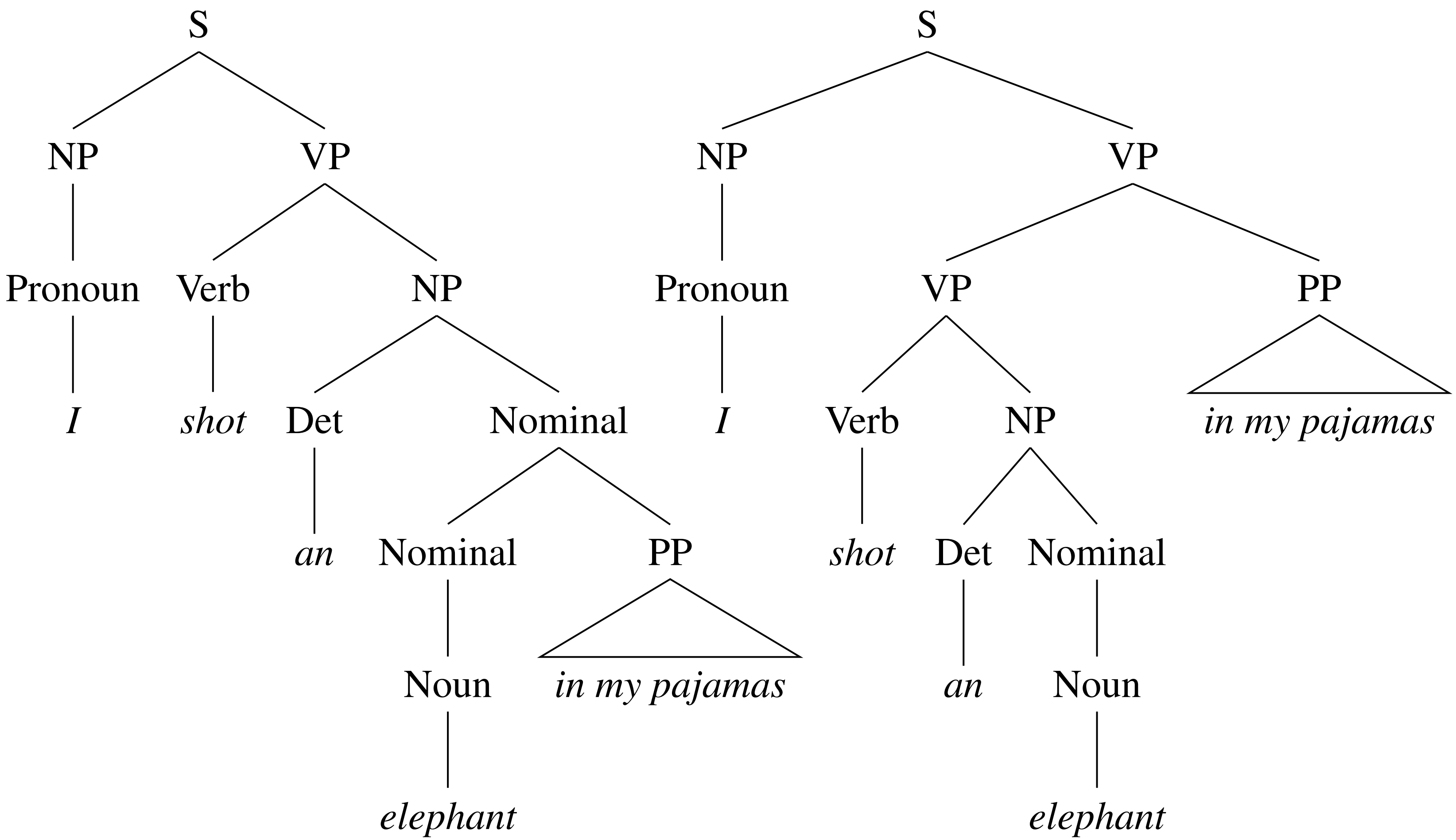
Examples from Chris Manning, Jason Eisner, and Andrew McCallum

Ambiguity

Ambiguity can arise because of words with **multiple senses or POS tags**. Many kinds of ambiguity are also structural.



Attachment Ambiguity



Parsing Algorithms

NLP systems need to choose a single correct parse from many, many possible parses. They need to perform **syntactic disambiguation.**

Effective disambiguation algorithms require a variety of information to be integrated into parsing algorithms. Such information includes:

1. statistical information
2. semantic understanding
3. contextual knowledge

We'll start by looking at an efficient dynamic programming algorithm, and then see how to add statistics to it.

The Parsing Problem

Given sentence **x** and grammar **G**,

Recognition

Is sentence **x** in the grammar? If so, prove it.

"Proof" is a deduction, valid parse tree.

Parsing

Show one or more derivations for **x** in **G**.

Even with small grammars, brute force grows exponentially!

"Book that flight"

Left to Right?

The old man the boat.

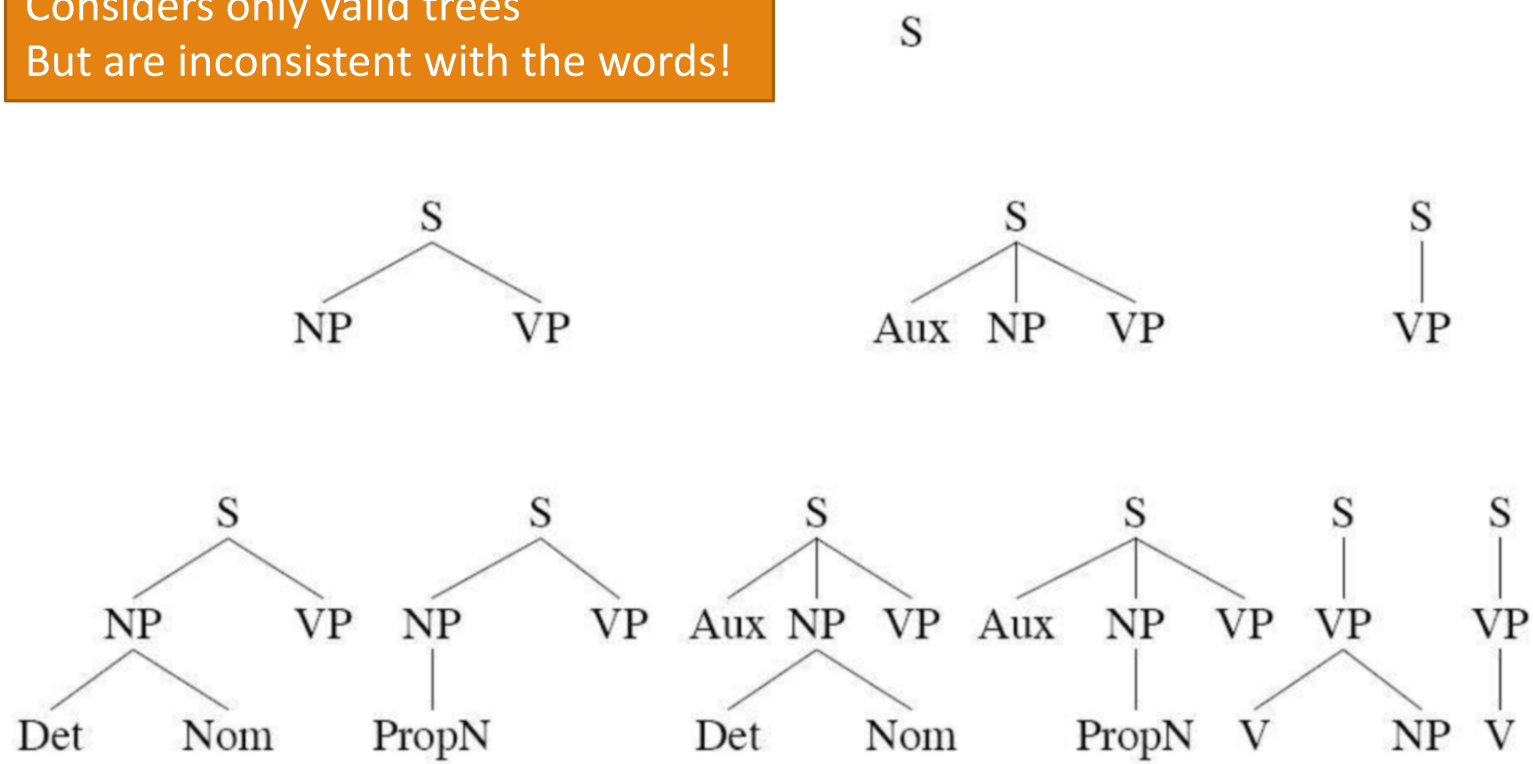
The complex houses married and single soldiers and their families.

Garden Path Sentences

Top Down Parsing

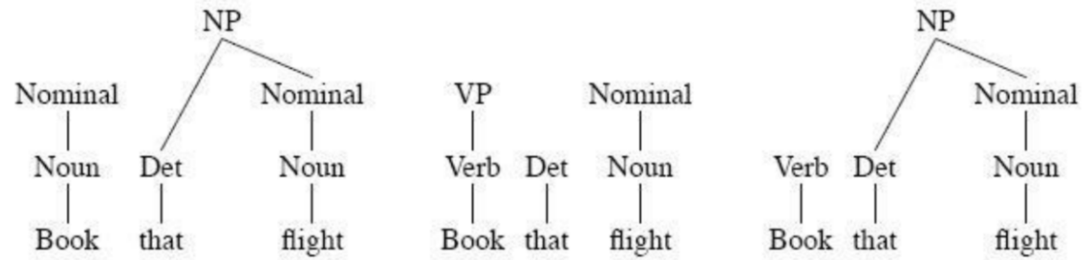
Considers only valid trees
But are inconsistent with the words!

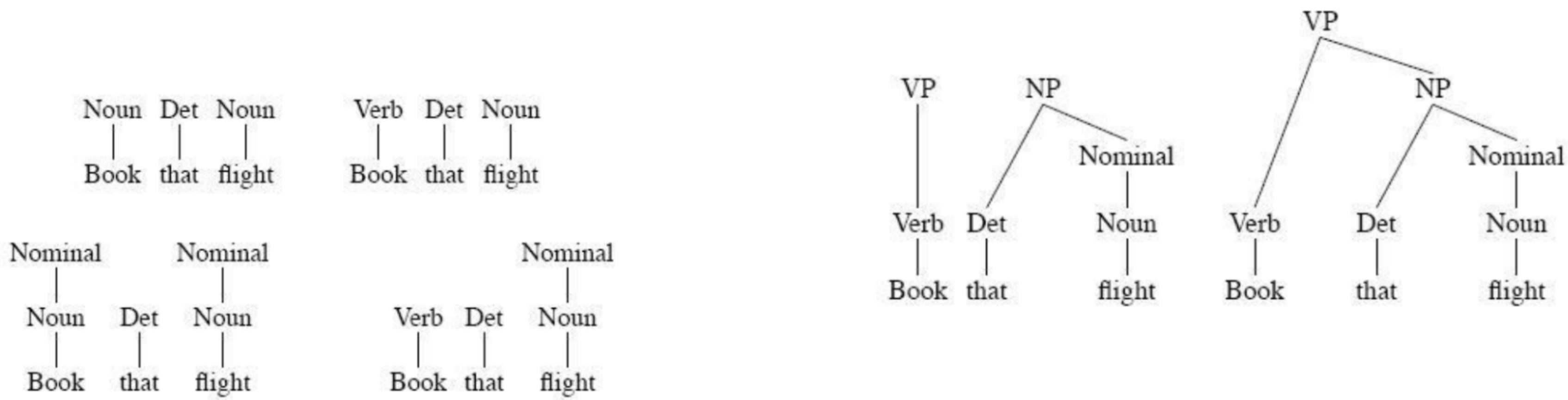
"Book that flight"



Bottom-up Parsing

"Book that flight"





Builds only consistent trees
But most of them are invalid (don't go anywhere)!

Chomsky Normal Form

Context free grammar where all non-terminals to go:

- 2 non-terminals, or
- A single terminal

$A \rightarrow B\ C$ $\qquad$ $D \rightarrow w$

Converting to CNF

Case 1

$A \rightarrow B$
$B \rightarrow C\ D$
$B \rightarrow w$

$\Rightarrow$

$A \rightarrow C\ D$
$A \rightarrow w$

Case 2

$A \rightarrow B\ C\ D\ E$

$\Rightarrow$

$A \rightarrow X\ E$
$X \rightarrow Y\ D$
$Y \rightarrow B\ C$

Original Grammar	Chomsky Normal Form
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

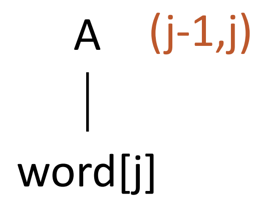
Dynamic Programming

table[i,j] = Set of all valid non-terminals for the constituent span (i,j)

Base case

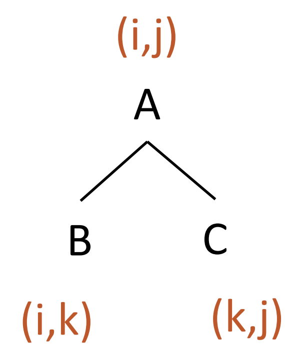
Rule: $A \rightarrow word[j]$

A should be in table[j-1,j]



Recursion

Rule: $A \rightarrow B\ C$



If you find a k such that

B is in table[i,k], and

C is in table[k,j], then A should be in table[i,j]

Dynamic Programming

$S \in table[0,n]$

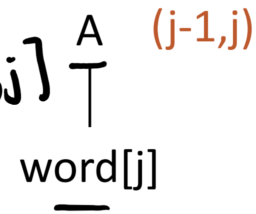
table[i,j] = Set of all valid non-terminals for the constituent span (i,j)

Base case

Rule: $A \rightarrow word[j]$ ←

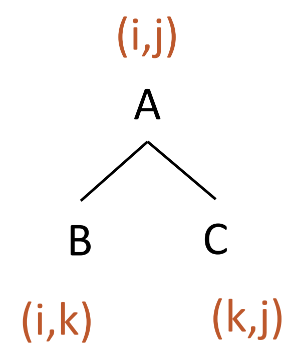
A should be in table[j-1,j]

$A \in table[j-1,j]$



Recursion

Rule: $A \rightarrow B\ C$



If you find a k such that

B is in table[i,k], and

C is in table[k,j], then A should be in table[i,j]

CKY Algorithm

function CKY-PARSE(*words, grammar*) **returns** *table*

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
          $table[i,j] \leftarrow table[i,j] \cup A$

DEMO AT HTTP://LXMLS.IT.PT/2015/CKY.HTML

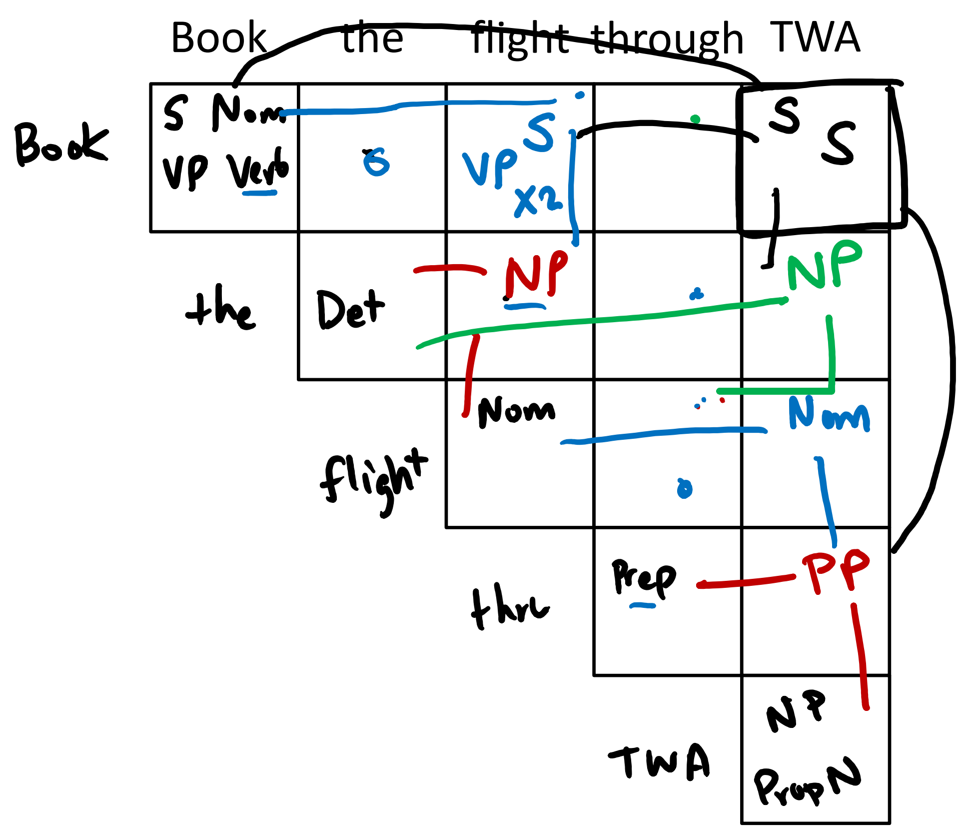
CKY Algorithm

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

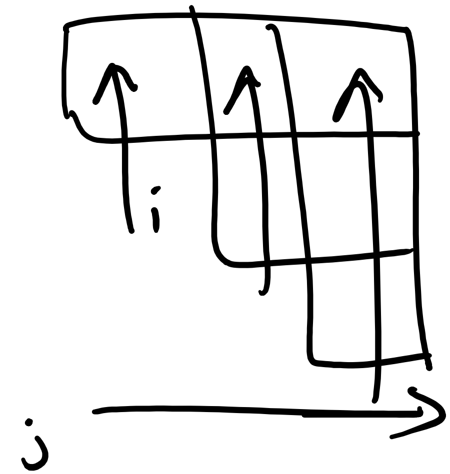
Book	the	flight	through	TWA
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	[1,2]	[1,3]	[1,4]	[1,5]
		[2,3]	[2,4]	[2,5]
			[3,4]	[3,5]
				[4,5]

CKY Algorithm

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 PP
S → Verb PP
S → VP PP
NP → I | she | me
NP → TWA | Houston
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → X2 PP
X2 → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm



function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**

n

  for all $\{A \mid A \rightarrow words[j] \in grammar\}$

    $table[j-1, j] \leftarrow table[j-1, j] \cup A$

  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**

    for $k \leftarrow i+1$ **to** $j-1$ **do**

n

      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$

        $table[i,j] \leftarrow table[i,j] \cup A$

n

|R|

CKY Algorithm: Complexity

$|N|$: Number of non-terminals

$|R|$: Number of rules

n: Number of tokens in the sentence

Memory

Time

CKY Algorithm: Complexity

$|N|$: Number of non-terminals

$|R|$: Number of rules

n: Number of tokens in the sentence

Memory $O(n^2 |N|)$

Time $O(n^3 |R|)$

Outline

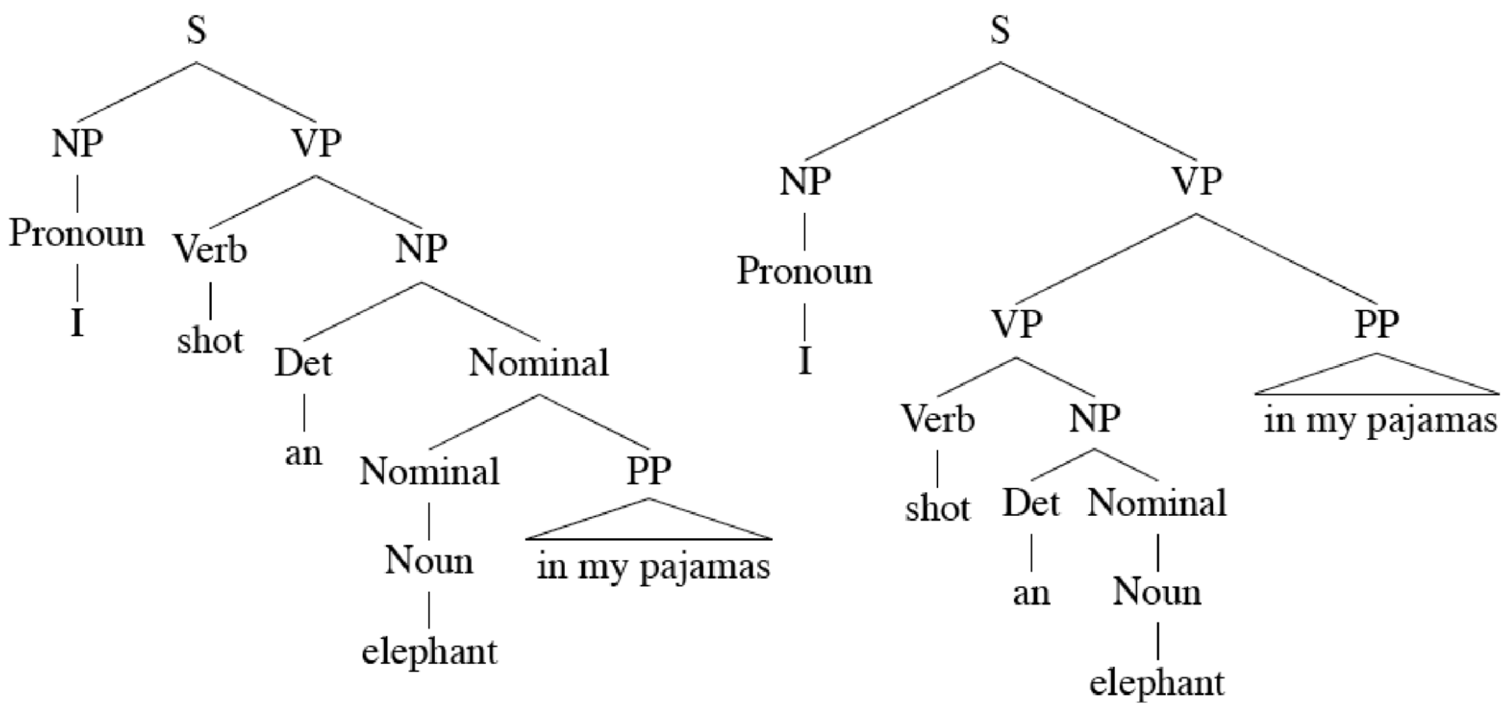
Parsing: CKY Algorithm

Extensions: Probabilistic and Lexicalized

Dependency Parsing

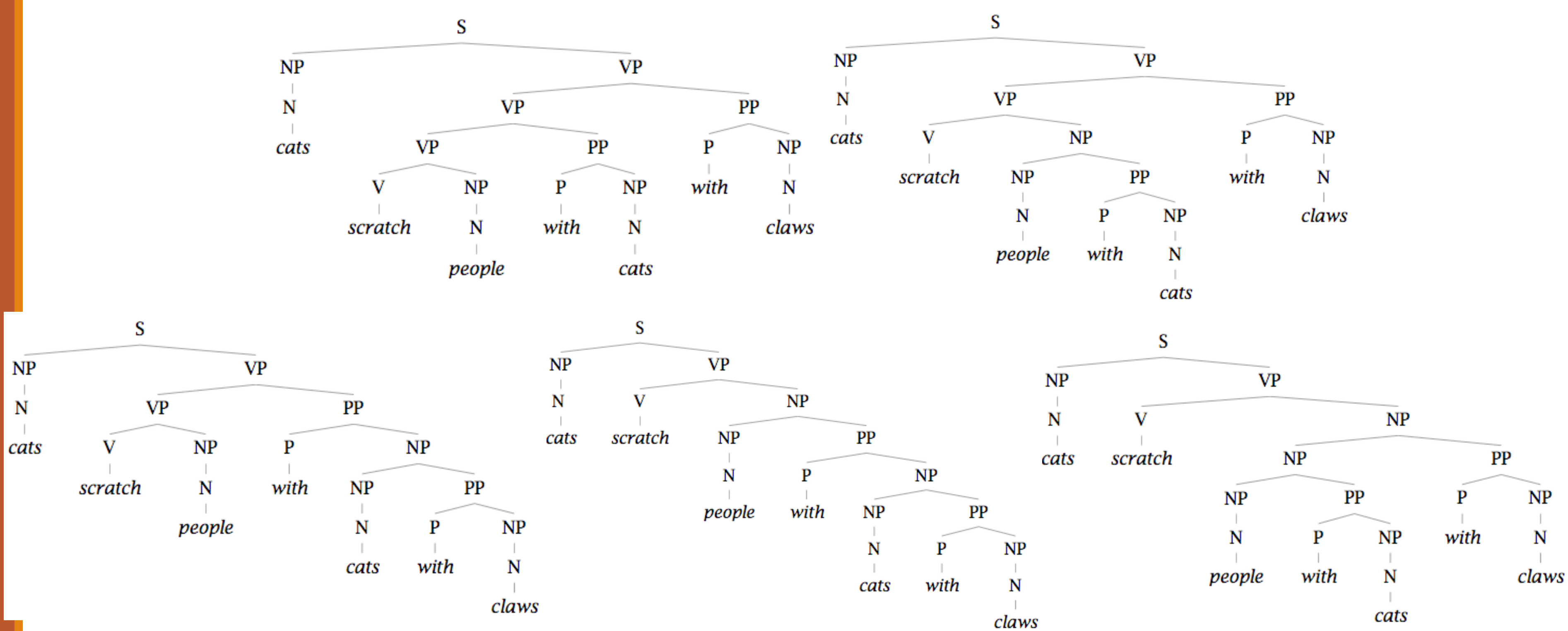
Ambiguity: Which parse?

I shot an elephant in my pajamas.



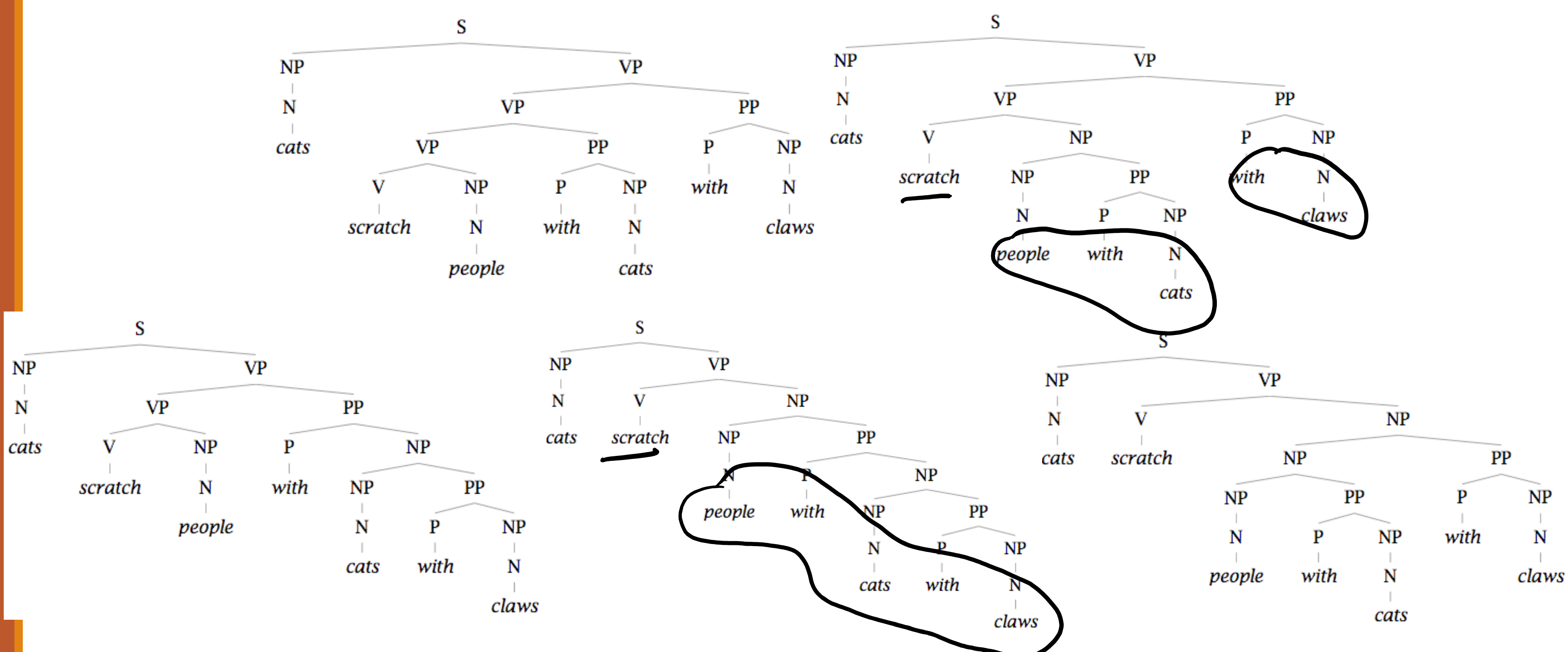
Finding the *Best* Parse Tree

Cats scratch people with cats with claws.



Finding the *Best* Parse Tree

Cats scratch people with cats with claws.



Probabilistic CFGs

Same as a regular context-free grammar:

- Terminal, non-terminals, and rules
- Additionally, attach a probability to each rule!

Rule: $A \rightarrow B\ C$ Probability: $P(A \rightarrow B\ C \mid A)$

Compute the probability of a parse tree:

Probabilistic CFGs

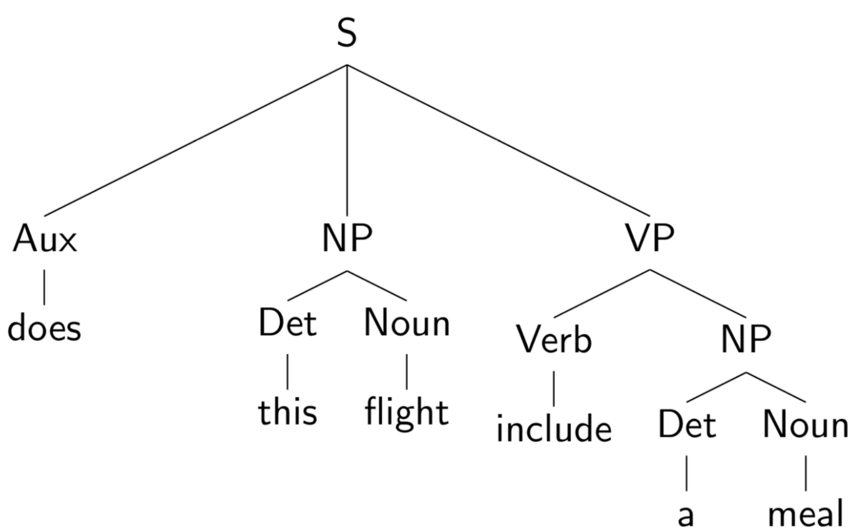
Same as a regular context-free grammar:
- Terminal, non-terminals, and rules
- Additionally, attach a probability to each rule!

Rule: $A \rightarrow B\ C$

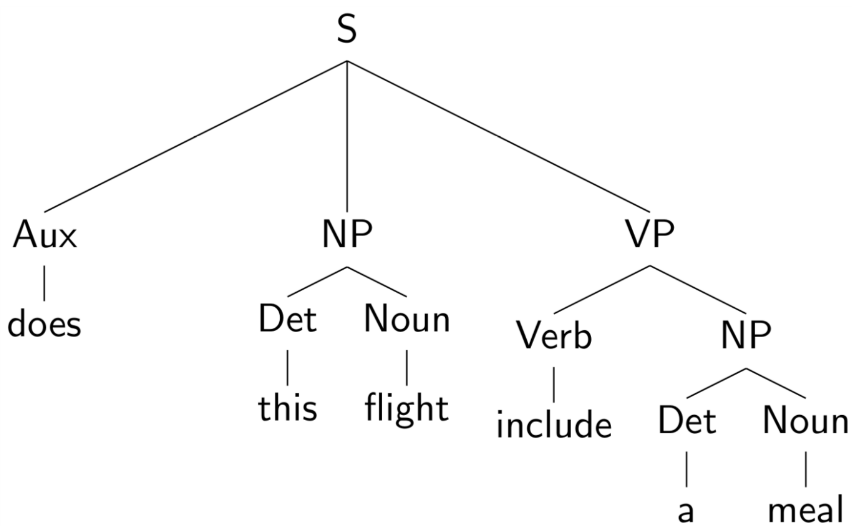
Probability: $P(A \rightarrow B\ C \mid A)$

Compute the probability of a parse tree: $\prod_{A \rightarrow BC \in T} P(A \rightarrow BC \mid A)$

Example of a PCFG



Example of a PCFG



$P(\text{Aux NP VP} \mid S)$

$P(\text{does} \mid \text{AUX})$

$P(\text{Det Noun} \mid NP)$

$P(\text{this} \mid \text{DET})\ P(\text{flight} \mid \text{Noun})$

$\vdots$

Estimating the probabilities

Estimating the probabilities

$$P(\alpha \to \beta \mid \alpha) = \frac{\# \alpha \to \beta}{\# \alpha}$$

The Parsing Problem

Given sentence **x** and grammar **G**,

Recognition

Is sentence **x** in the grammar? If so, prove it.

"Proof" is a deduction, valid parse tree.

Parsing

Show one or more derivations for **x** in **G**.

$$\operatorname*{argmax}_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}}} p(\boldsymbol{t} \mid \boldsymbol{x})$$

Even with small grammars, grows exponentially!

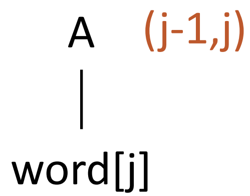
Probabilistic CKY Algorithm

T[i,j,A] = Probability of the best parse with root A for the span (i,j)

Base case

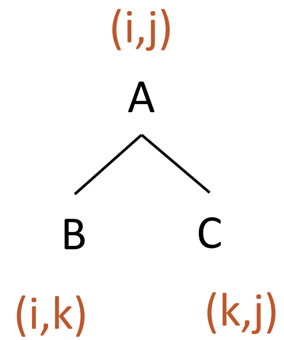
Rule: $P(A \rightarrow word[j])$

T[j-1,j,A] = P(word[j] | A)



Recursion

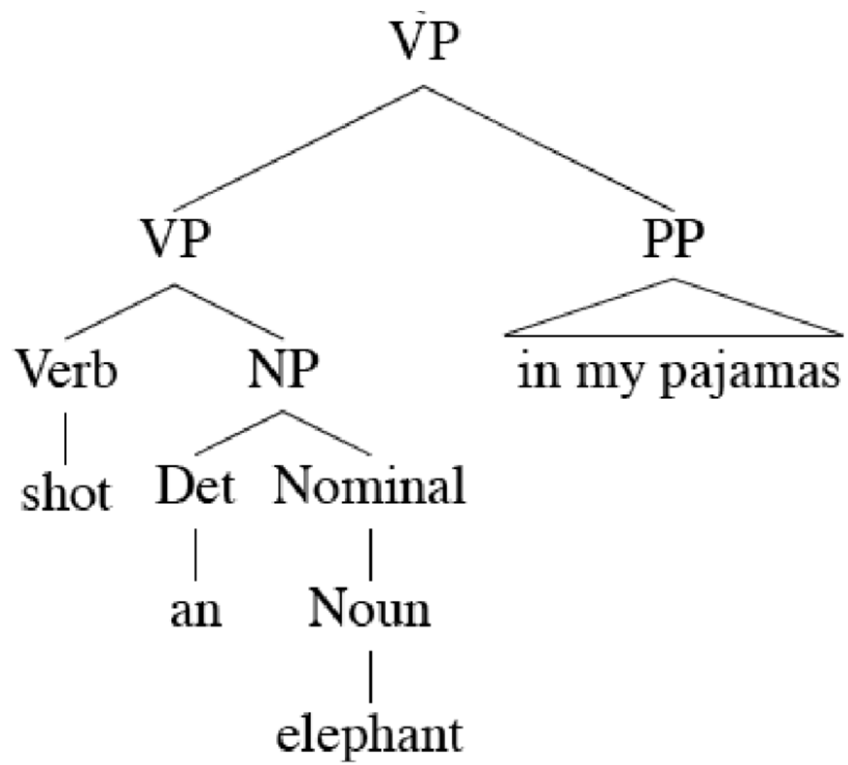
Rule: $P(A \rightarrow B\ C)$



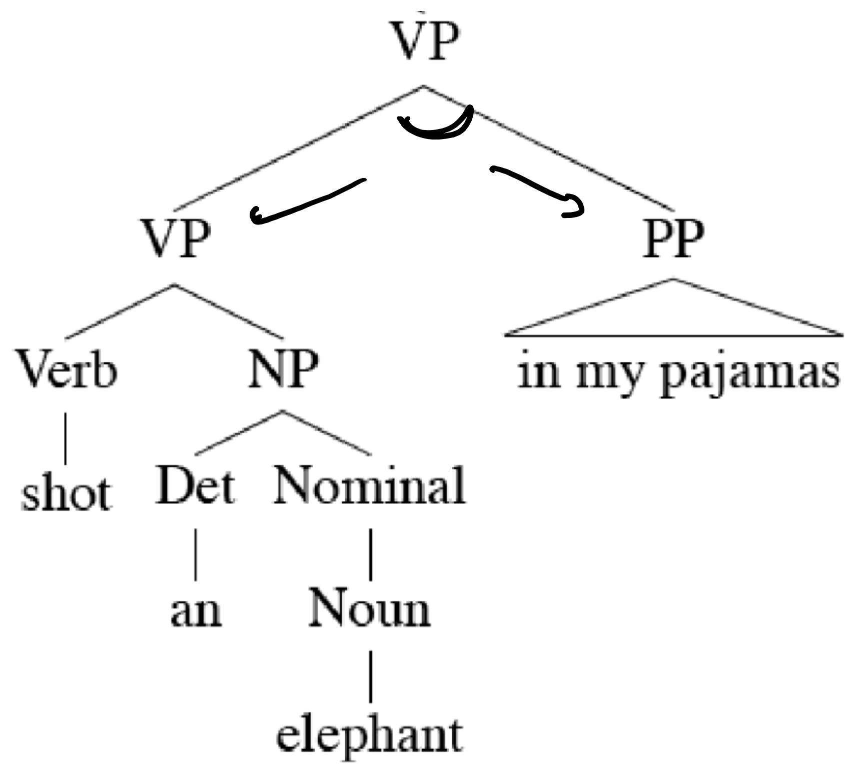
Try every position k, and every non-terminal pair:

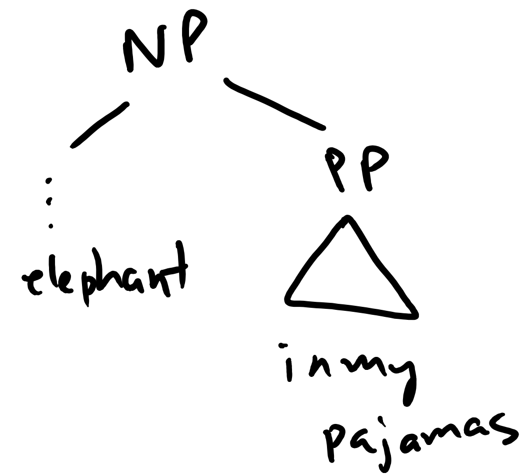
$$T[i,j,A] = \max_{k}\ P(B\ C \mid A)\ T[i,k,B]\ T[k,j,C]$$

Lexicalized PCFGs

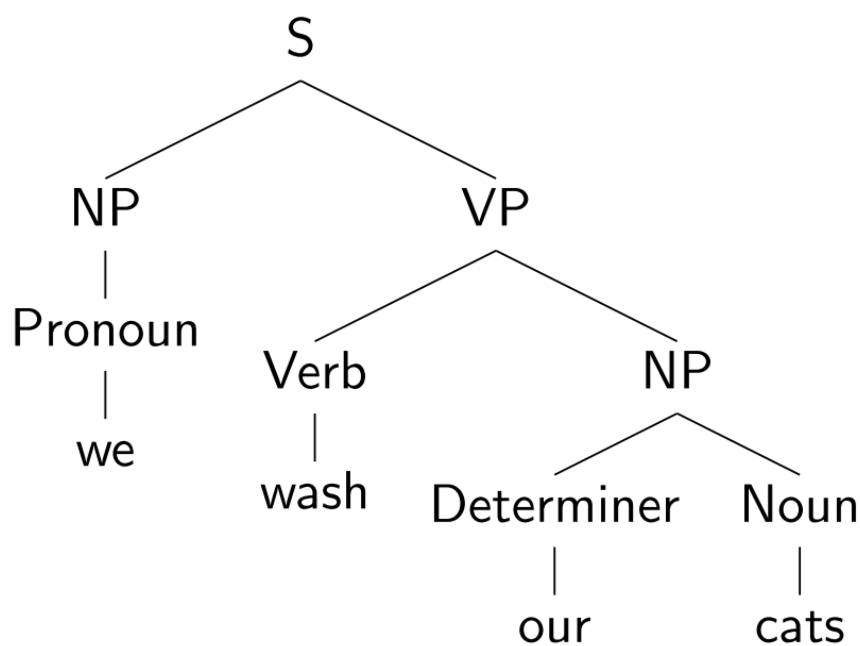


Lexicalized PCFGs

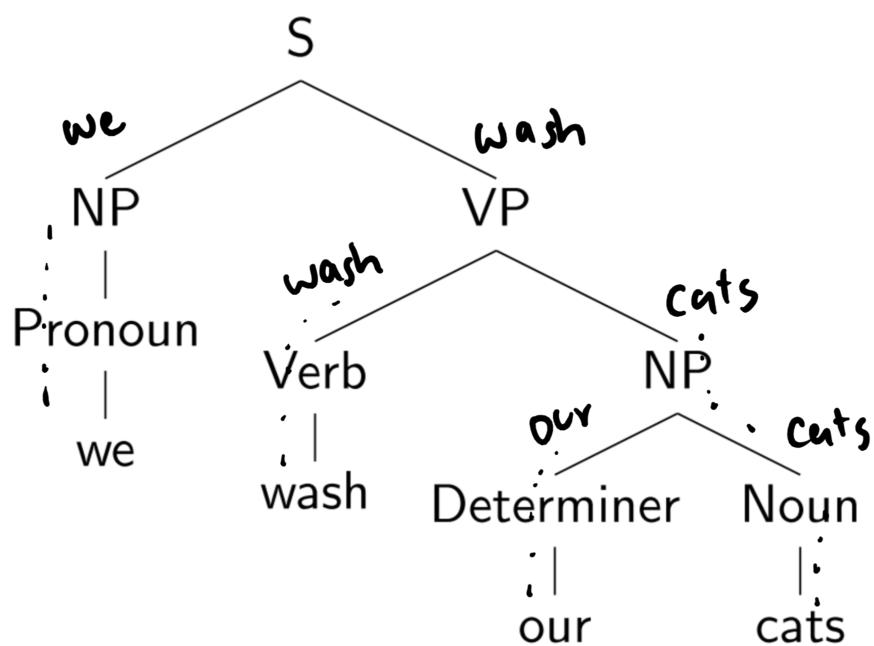




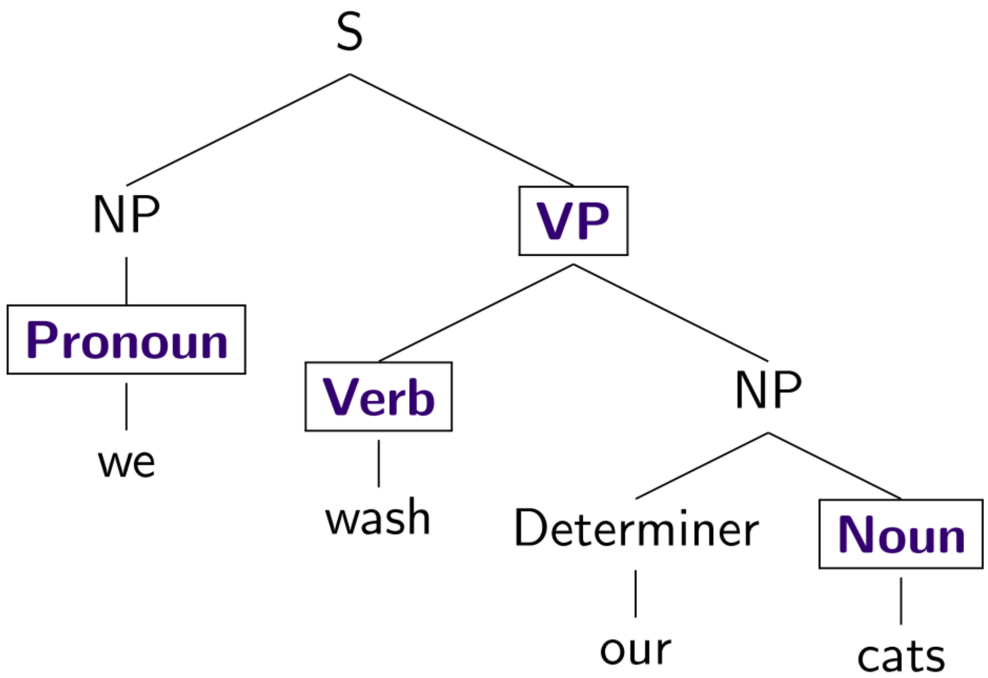
Lexicalizing a CFG



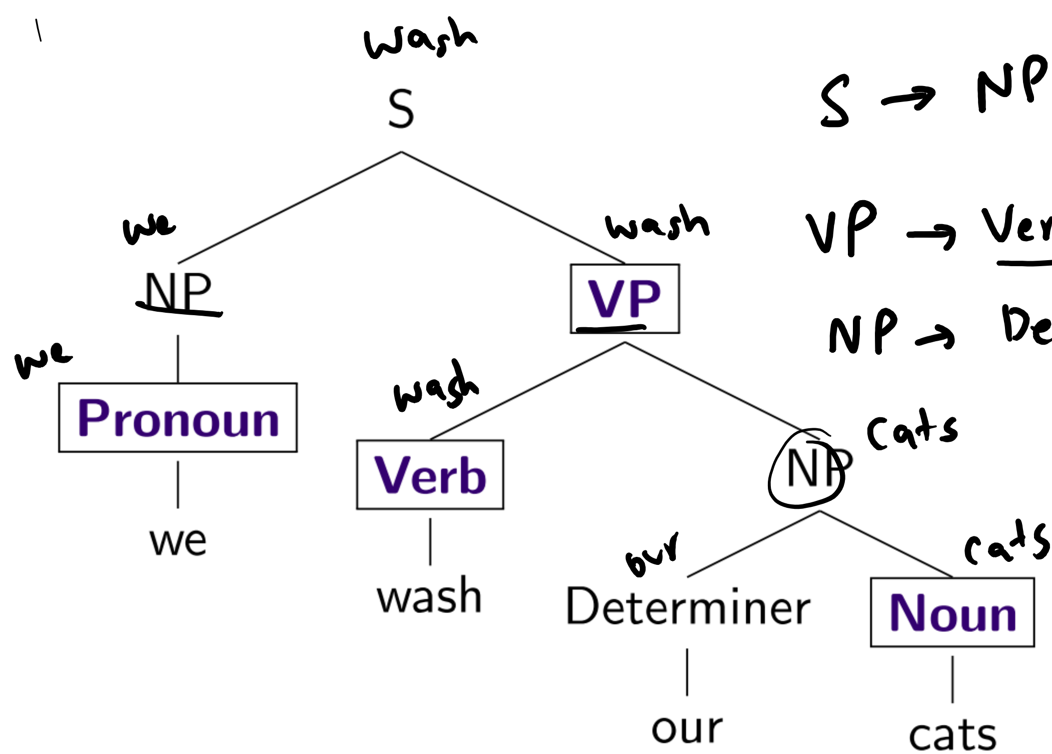
Lexicalizing a CFG



Lexicalizing a CFG



Lexicalizing a CFG

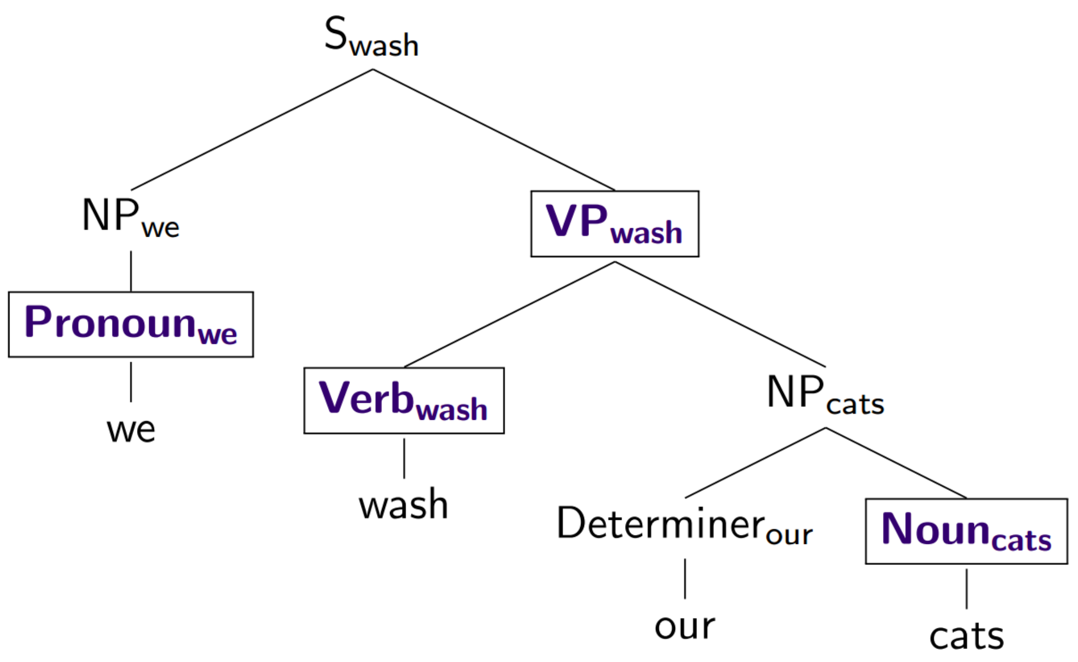


$S \rightarrow NP\ \underline{VP}^{*}$

$VP \rightarrow \underline{Verb}^{*}\ NP$

$NP \rightarrow Det\ \underline{Noun}^{*}$

Lexicalizing a CFG



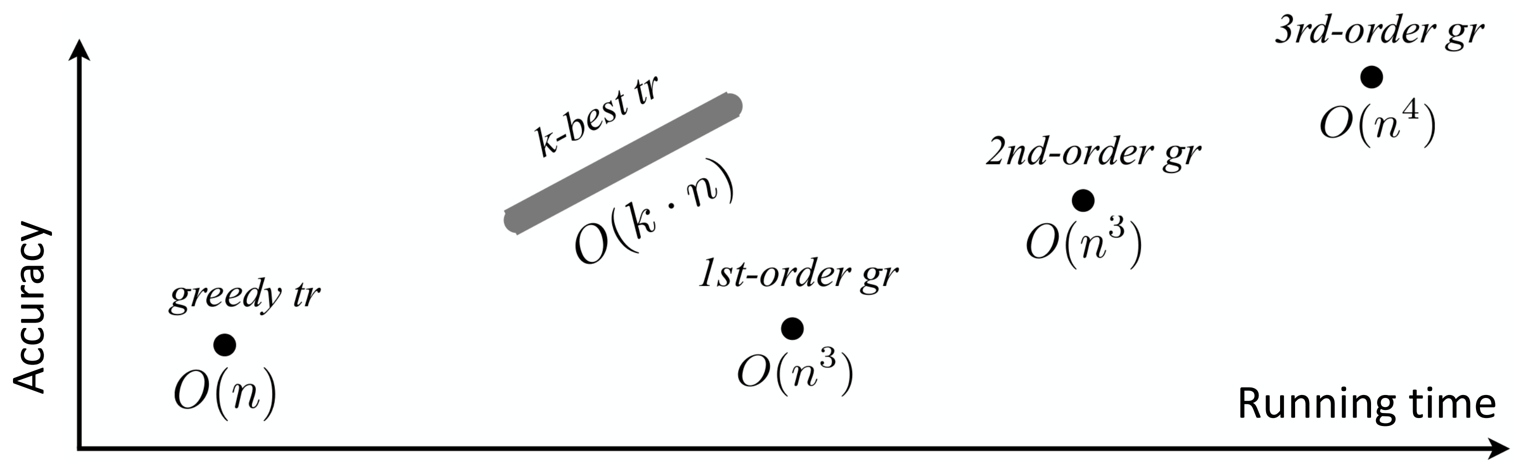
Parsing Algorithms

Transition-based

- Fast, greedy, linear-time
- Trained for greedy search
- Features decide what to do next
- Beam search, i.e. *k*-best

Graph-based

- Slower, exhaustive algorithms
- Dynamic programming, inference
- Features used to score whole trees



Graph-based Parsing

$$\underset{t \in T}{\operatorname{argmax}} \ \text{score}(t, \theta)$$

1st order / fully factored

$$\sum_i \theta_i \, \phi(\underbrace{e_i}_{(c_i,\, p_i,\, l_i)})$$

Proj: Dynamic Programming

NonProj: Maximum Spanning Tree

2nd order $\phi(e_i, e_j)$

3rd order $\phi(e_i, e_j, e_k)$