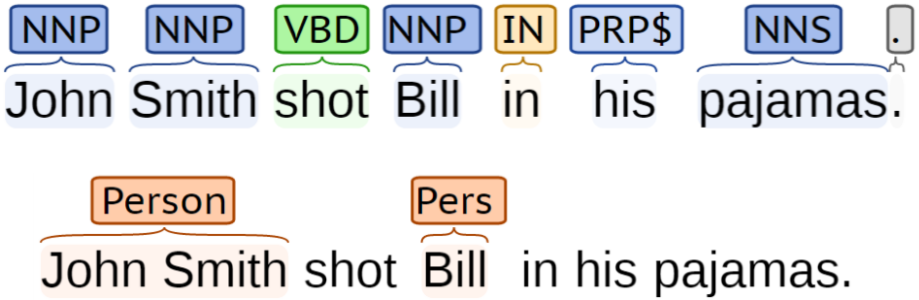
Constituency Grammars

JURAFSKY AND MARTIN CHAPTER 12

Based on slides from Sameer Singh, Dan Jurafsky, Noah Smith, Slav Petrov, and everyone else they copied from.

Limitations of Sequence Tags

John Smith shot Bill in his pajamas.



What happened?
Who shot who?
Who was wearing the pajamas?

How do words group together in English?

A **noun phrase** is a sequence of words surrounding at least one noun.

What evidence do we have that these words group together?

Noun Phrases
Harry the Horse
the Broadway coppers
they
a high-class spot such as Mindy's
the reason he comes into the Hot Box
three parties from Brooklyn

Constituents

- Constituent behave as a unit that can be rearranged:

  John talked **[**to the children**]** **[**about drugs**]**.

  John talked **[**about drugs**]** **[**to the children**]**.

  John talked drugs to the children about

- Or substituted/expanded:

  John talked **[**to the children taking the drugs**]** **[**about alcohol**]**.

Harry the Horse
a high-class spot such as Mindy's
the Broadway coppers
the reason he comes into the Hot Box
they
three parties from Brooklyn

X

arrive(s)
attract(s)
love(s)
sit(s)

"Noun phrases appear before verbs in English."

Constituents and Grammars

Grammar

- Tells you how the constituents can be arranged
- Implicit knowledge for us (we often can't tell *why* something is wrong)
- Generate all, and only, the possible sentences of the language
- Different from meaning:

Colorless green ideas sleep furiously.

- The words are in the right order,
- And that ideas are green and colorless,
- And that ideas sleep,
- And that sleeping is done furiously,
- As opposed to: "sleep green furiously ideas colorless"

Uses of Parsing

[send [the text message from James] [to Sharon]]

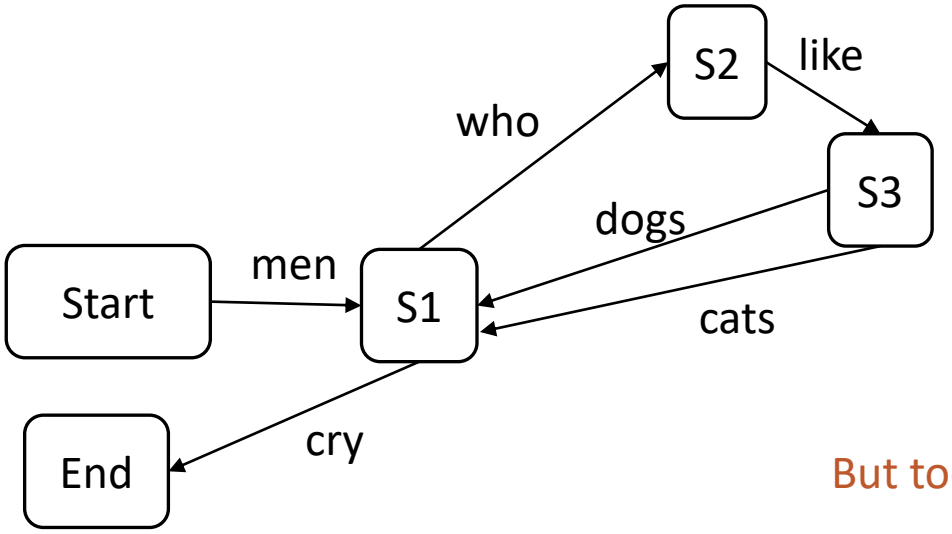
[translate [the message] [from Hindi] [to English]]

- Grammar checkers
- Dialog systems
- High precision question answering
- Named entity recognition
- Sentence compression
- Extracting opinions about products
- Improved interaction in computer games
- Helping linguists find data
- Machine translation
- Relation extraction systems

Basic Grammar: Regular Expr.

- You can capture individual words:
  - (man|dog|cat)
- Simple sentences:
  - (man|dog|cat)(ate|loves|consumed)(.|food|lunch)
- Infinite length? Yes!
  - men (who like (cats|dogs))* cry.

Finite State Machine



But too weak for English.

Context-Free Grammars

Grammar, G

Terminal Symbols

Non-terminal Symbols

Rules

Grammar applies rules recursively..
If we can construct the input sentence, it is in the grammar, otherwise not.

Context-Free Grammars

Grammar, G

T

Terminal Symbols

words

{man, dog, food... }

N "Constituents"

Non-terminal Symbols

S, NP, VP, ...

Rules

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

$A \rightarrow w$

$A \rightarrow B\,A\,B$

$A \rightarrow a \qquad B \rightarrow b$

"a"

"bab" "baba" —x

$\rightarrow b^* a\, b^+ \leftarrow$

Grammar applies rules recursively..

If we can construct the input sentence, it is in the grammar, otherwise not.

Context Free Grammars

Context Free Grammars

$G \equiv \langle T, N, R, S \rangle$

- T: words
- N: constituents
- R: how to compose them
- S: sentence

$A \to ABA$

$A \to a$

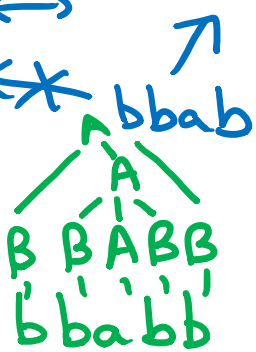
$B \to b$

$G \equiv \langle \{a,b\}, \{A,B\}, R, A \rangle$

a
bab
bbabb
⋮

$\leftrightarrow$ $b^* a b^*$

$\Leftrightarrow$ bbab



Example CFG

Noun	→	*flights* | *breeze* | *trip* | *morning*
Verb	→	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective	→	*cheapest* | *non-stop* | *first* | *latest* | *other* | *direct*
Pronoun	→	*me* | *I* | *you* | *it*
Proper-Noun	→	*Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American*
Determiner	→	*the* | *a* | *an* | *this* | *these* | *that*
Preposition	→	*from* | *to* | *on* | *near*
Conjunction	→	*and* | *or* | *but*

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

"Lexicon"

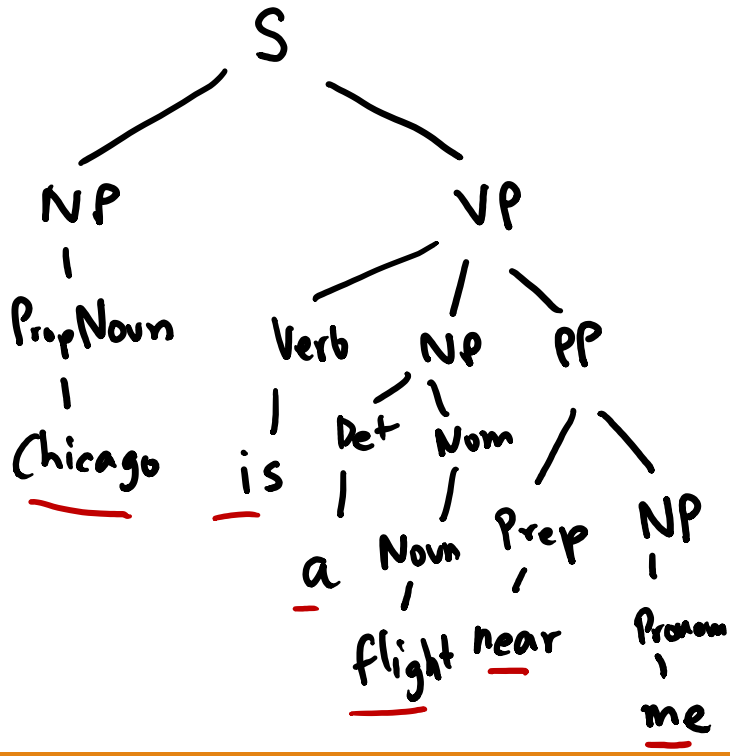
Example CFG

A→w1 | w2

A→w1 A→w2

Noun	→	*flights* | *breeze* | *trip* | *morning*
Verb	→	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective	→	*cheapest* | *non-stop* | *first* | *latest* | *other* | *direct*
Pronoun	→	*me* | *I* | *you* | *it*
Proper-Noun	→	*Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American*
Determiner	→	*the* | *a* | *an* | *this* | *these* | *that*
Preposition	→	*from* | *to* | *on* | *near*
Conjunction	→	*and* | *or* | *but*

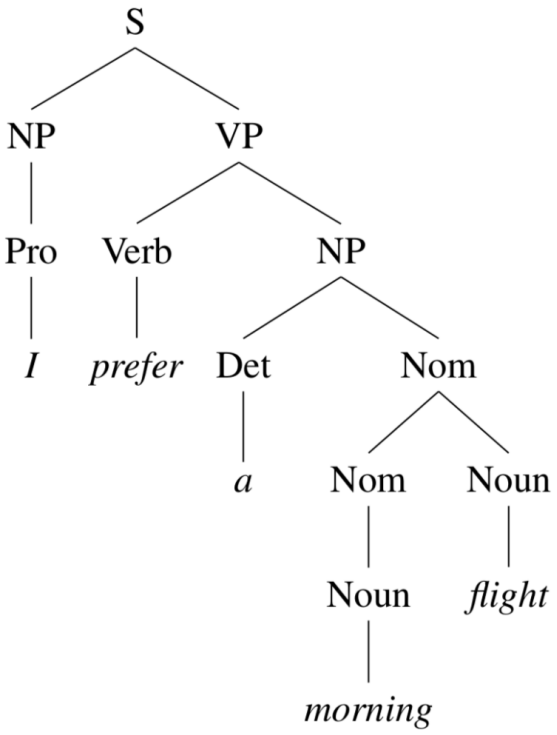
Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles



Example Parse Tree

I prefer a morning flight.

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles



Example Parse Tree

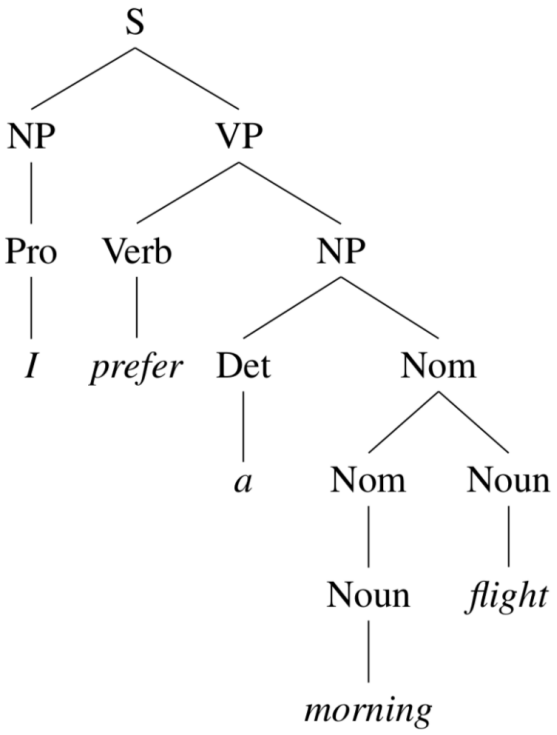
Nom → N_{time} F

F → flight

N_{time} → morning, night, evening

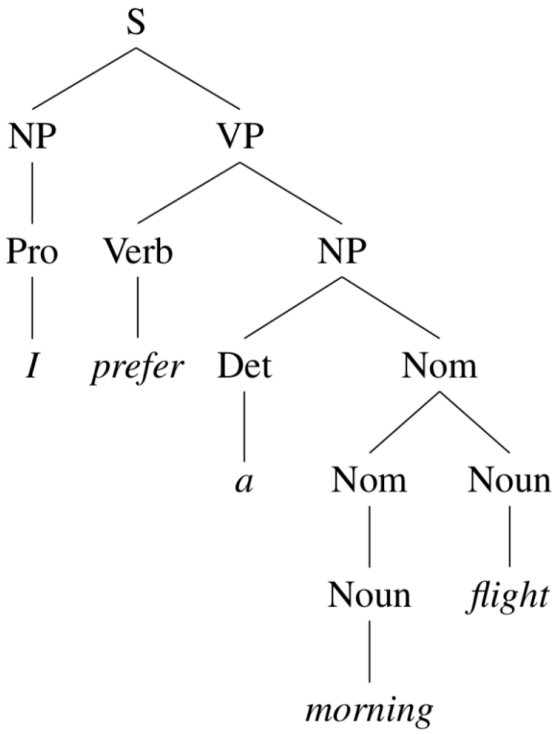
Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

I prefer a morning flight.



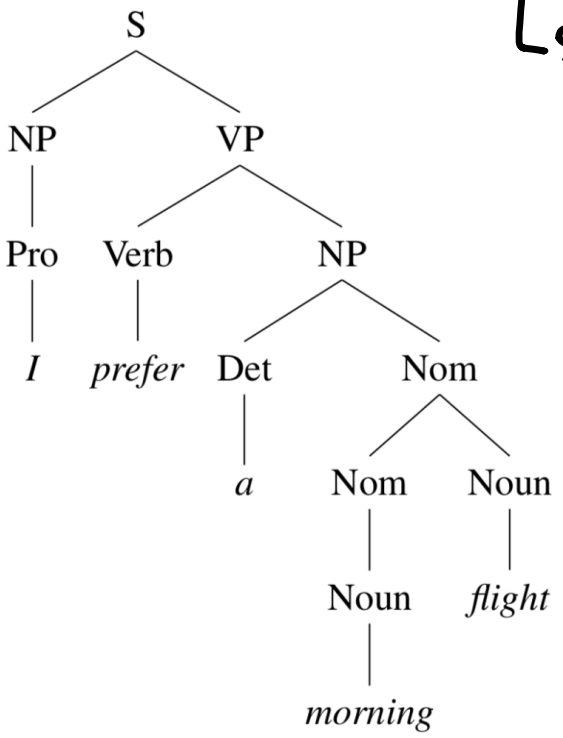
Example Parse Tree: Brackets

I prefer a morning flight.



Example Parse Tree: Brackets

I prefer a morning flight.



[S [NP [Pro I]][VP [Verb prefer][NP [Det ...] ..]]]

More details: Noun Phrases

Simple Noun Phrases

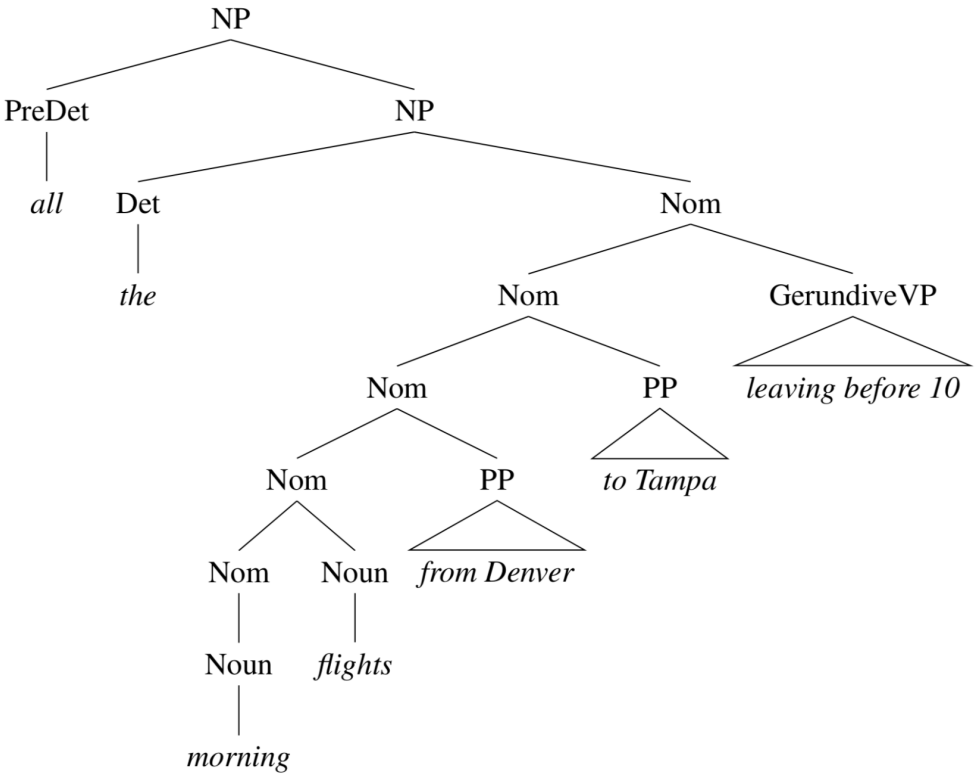
NP → ProperNoun

NP → Det Nominal

Nominal → Noun | Noun Nominal

Complex Noun Phrases

"all the morning flights from Denver to Tampa leaving before 10"



Recursive Noun Phrases

this is the house

this is the house that Jack built

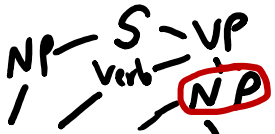
this is the cat that lives in the house that Jack built

this is the dog that chased the cat that lives in the house that Jack built

this is the flea that bit the dog that chased the cat that lives in the house the Jack built

this is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built

Recursive Noun Phrases



this is the house

this is the house that Jack built

this is the cat that lives in the house that Jack built

this is the dog that chased the cat that lives in the house that Jack built

this is the flea that bit the dog that chased the cat that lives in the house the Jack built

this is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built

More details: Verb Phrases

Simple Verb Phrases

VP → Verb disappear
VP → Verb NP prefer a morning flight
VP → Verb NP PP leave Boston in the morning
VP → Verb PP leave in the morning

But all verbs are not the same!
(this grammar overgenerates)

Solution: subcategorize!

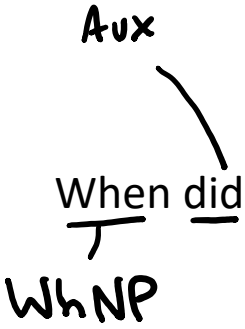
Sneezed: John sneezed.
Find: Please find a flight to NY.
Give: Give me a cheaper fare.
Help: Can you help me with a flight?
Prefer: I prefer to leave earlier.
Told: I was told United has a flight.

Types of Sentences

Declarative	$S \rightarrow NP\ VP$	A plane left.
Imperative	$S \rightarrow VP$	Show the plane.
Yes/no Questions	$S \rightarrow Aux\ NP\ VP$	Did the plane leave?
Wh-Questions	$S \rightarrow WhNP\ Aux\ NP\ VP$	When did the plane leave?

Types of Sentences

Declarative	$S \rightarrow NP\ VP$	A plane left.
Imperative	$S \rightarrow VP$	Show the plane.
Yes/no Questions	$S \rightarrow Aux\ NP\ VP$	Did the plane leave?
Wh-Questions	$S \rightarrow WhNP\ Aux\ NP\ VP$	When did the plane leave?



Source of Grammar?

Manual

Noam Chomsky

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*

Used grammar/proof systems to prove parses from words

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

Source of Grammar?

From data!

The Penn Treebank

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

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

```
( (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)))))))
    (. .)))
```

Some of the rules, with counts

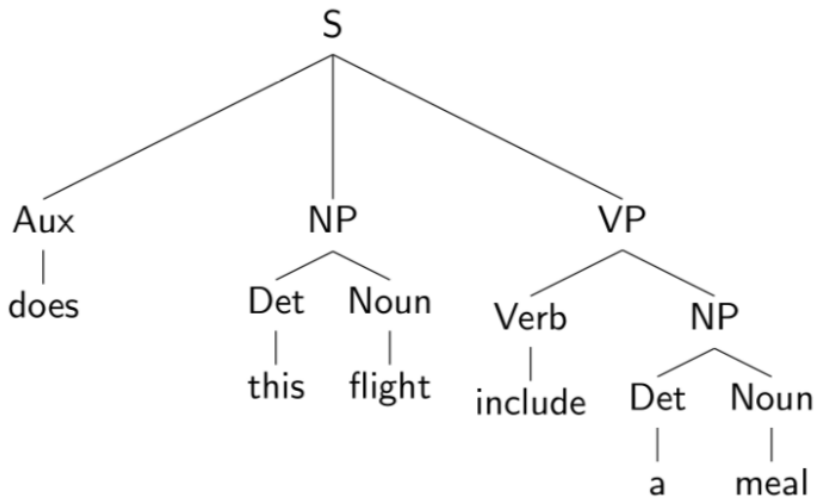
40717 PP → IN NP
33803 S → NP-SBJ VP
22513 NP-SBJ → -NONE-
21877 NP → NP PP
20740 NP → DT NN
14153 S → NP-SBJ VP .
12922 VP → TO VP
11881 PP-LOC → IN NP
11467 NP-SBJ → PRP
11378 NP → -NONE-
11291 NP → NN
...
989 VP → VBG S
985 NP-SBJ → NN
983 PP-MNR → IN NP
983 NP-SBJ → DT
969 VP → VBN VP

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!

Evaluating Parses

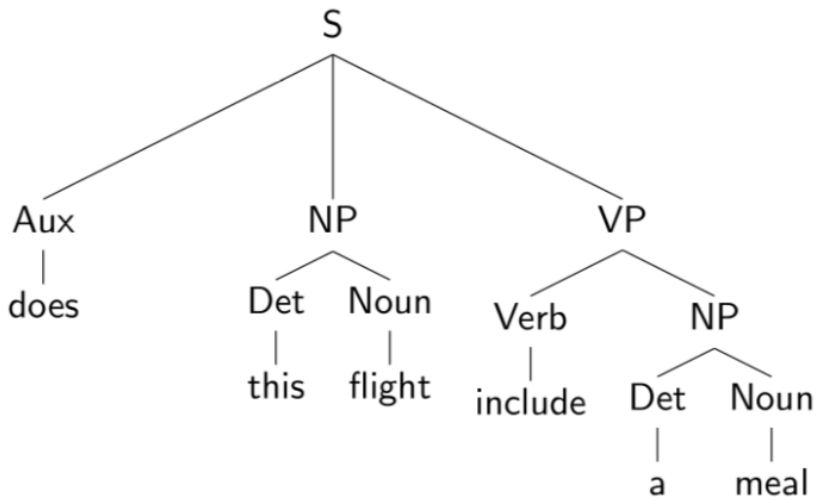
Each parse tree is represented by a list of tuples:



Use this to estimate precision/recall!

Evaluating Parses

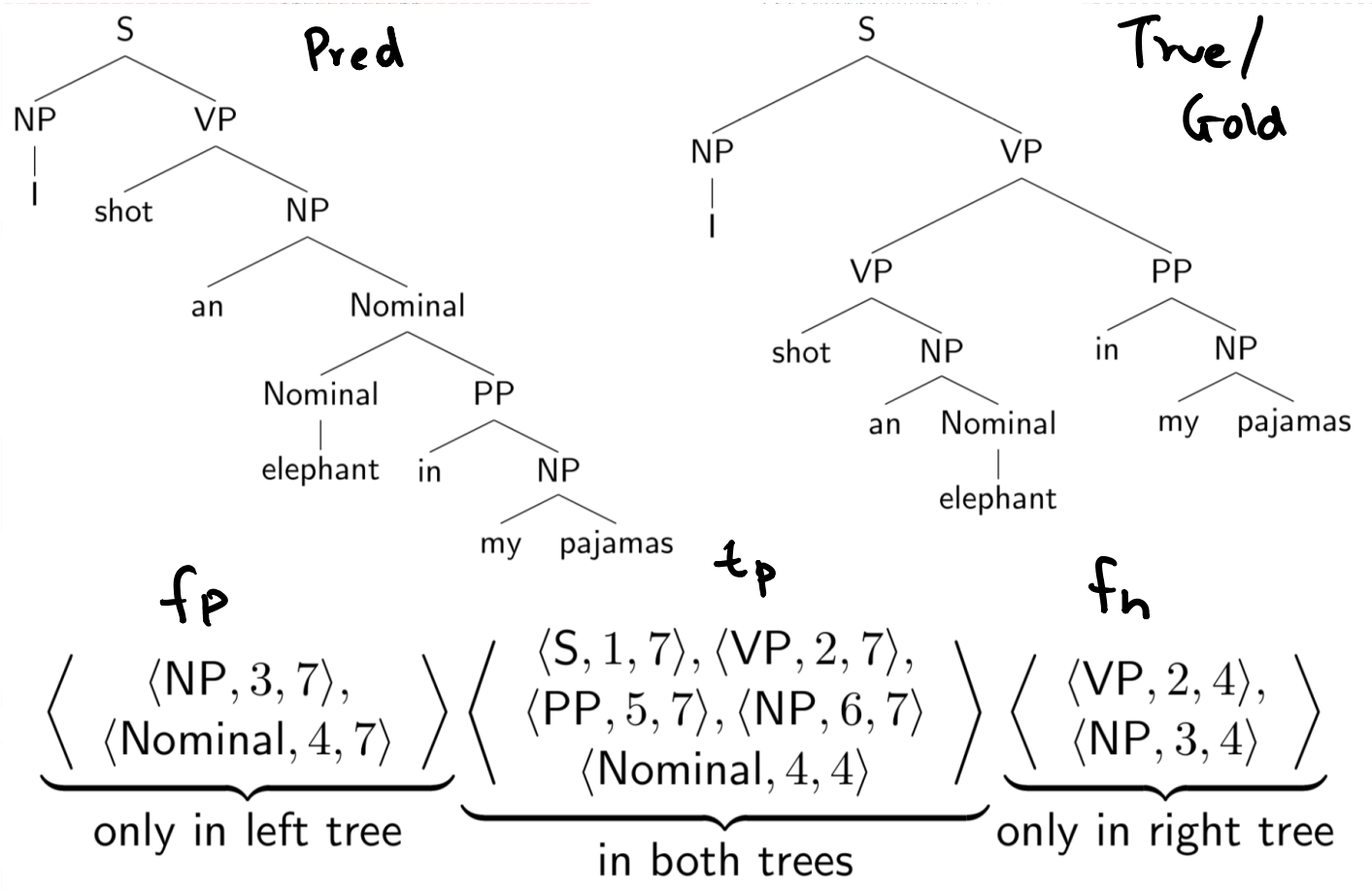
Each parse tree is represented by a list of tuples: $\{<t_i, s_i, e_i>\}$



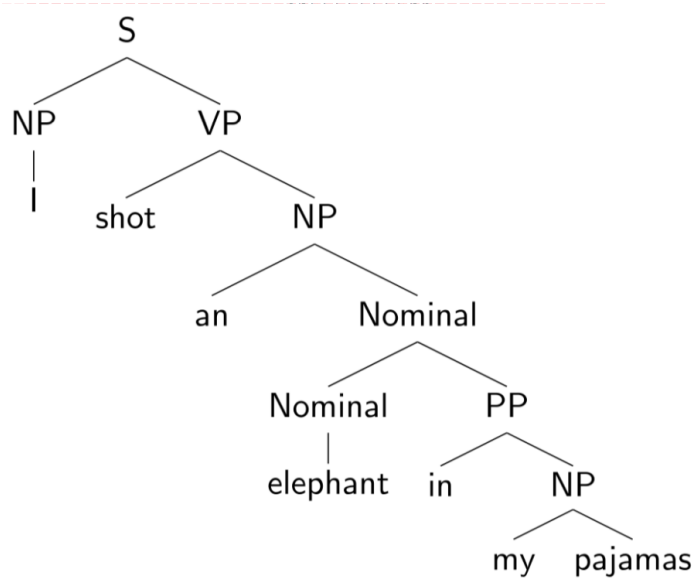
<S, 0, 6> <Aux, 0, 1>
<NP, 1, 3> <DET, 1, 2>
<Noun, 2, 3> <NP, 4, 6>
<VP, 3, 6> ...

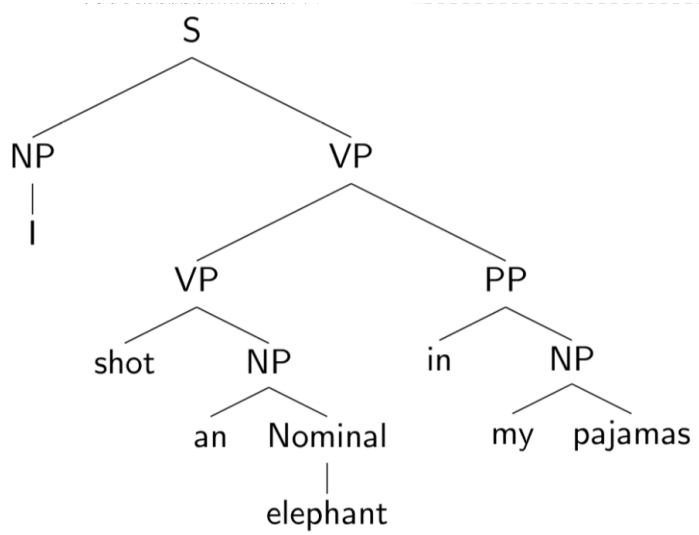
Use this to estimate precision/recall!

Evaluating Parses: Example



Evaluating Parses: Example





$$\underbrace{\left\langle \begin{array}{c} \langle \text{NP}, 3, 7 \rangle, \\ \langle \text{Nominal}, 4, 7 \rangle \end{array} \right\rangle}_{\text{only in left tree}} \quad \underbrace{\left\langle \begin{array}{c} \langle \text{S}, 1, 7 \rangle, \langle \text{VP}, 2, 7 \rangle, \\ \langle \text{PP}, 5, 7 \rangle, \langle \text{NP}, 6, 7 \rangle \\ \langle \text{Nominal}, 4, 4 \rangle \end{array} \right\rangle}_{\text{in both trees}} \quad \underbrace{\left\langle \begin{array}{c} \langle \text{VP}, 2, 4 \rangle, \\ \langle \text{NP}, 3, 4 \rangle \end{array} \right\rangle}_{\text{only in right tree}}$$

Outline

Context Free Grammars

Parsing: CKY Algorithm

Extensions: Probabilistic and Lexicalized

Dependency Parsing

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"

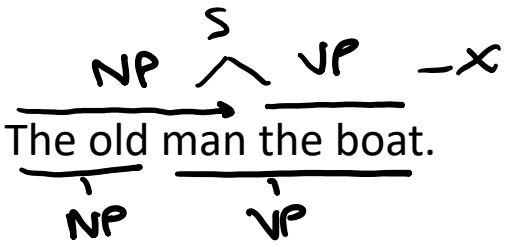
Left to Right?

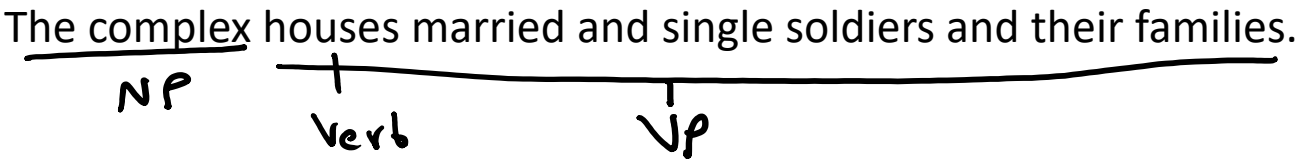
The old man the boat.

The complex houses married and single soldiers and their families.

Garden Path Sentences

Left to Right?



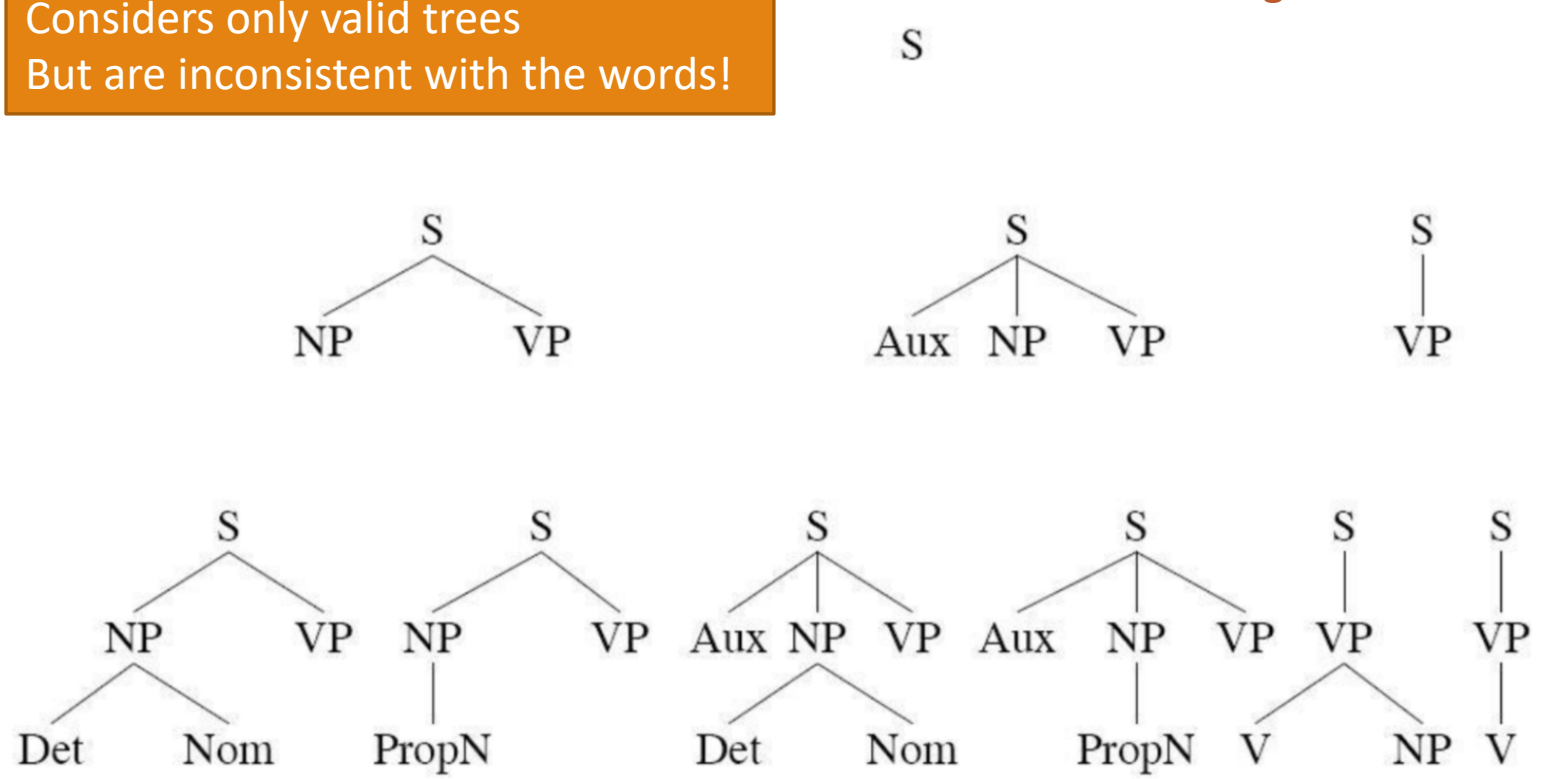


Garden Path Sentences

Top Down Parsing

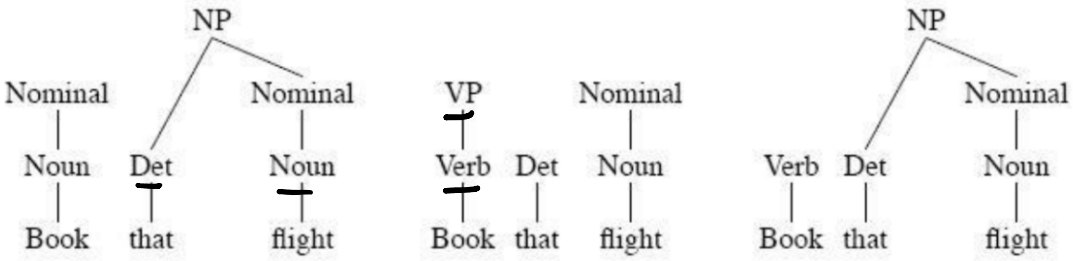
Considers only valid trees
But are inconsistent with the words!

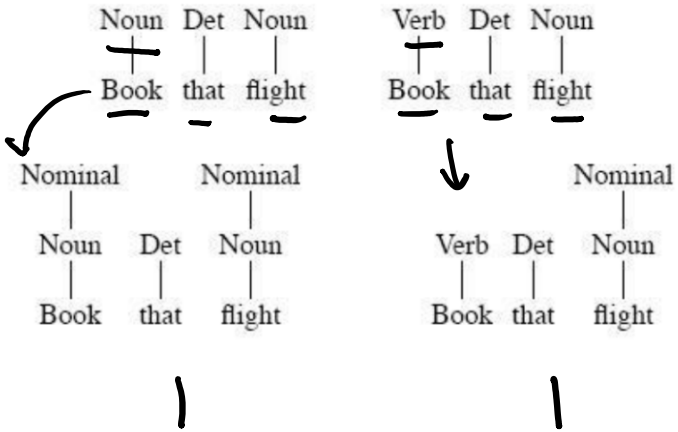
"Book that flight"

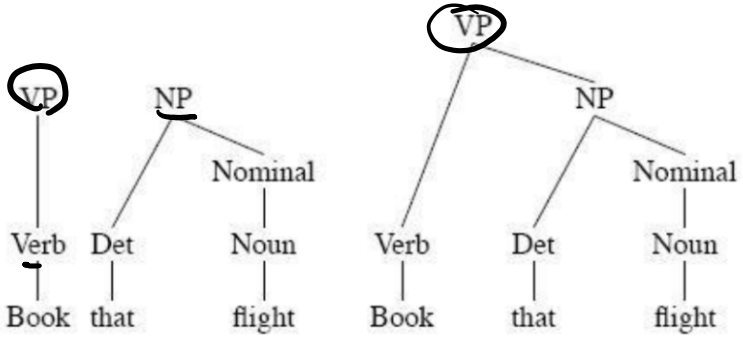


Bottom-up Parsing

"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:

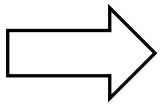
- 2 non-terminals, or
- A single terminal

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

Converting to CNF

Case 1

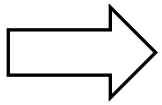
$A \rightarrow B$
$B \rightarrow C\ D$
$B \rightarrow w$



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

Case 2

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



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

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

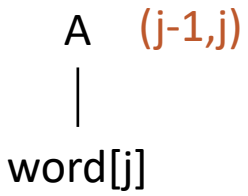
Dynamic Programming

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

Base case

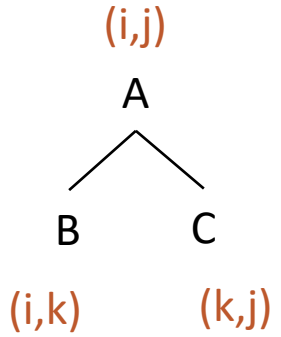
Rule: $A \rightarrow word[j]$

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]

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

A

(j-1,j)

word[j]

Recursion

Rule: $A \rightarrow B\ C$

(i,j)

A

B C

(i,k) (k,j)

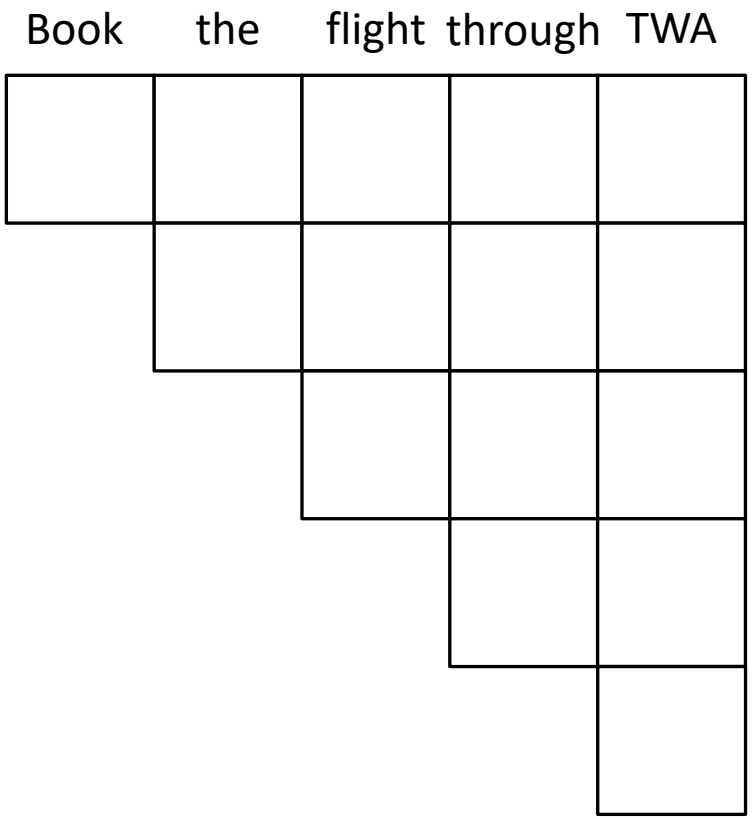
If you find a k such that

B is in table[i,k], and

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

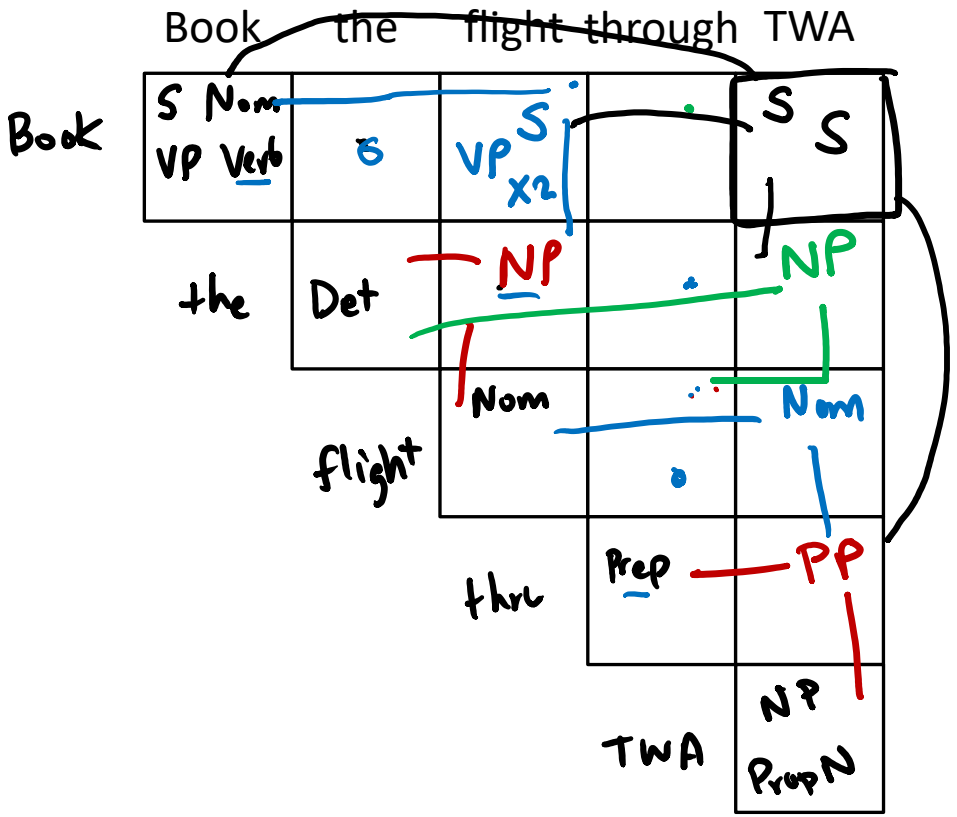
CKY Algorithm

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



CKY Algorithm

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

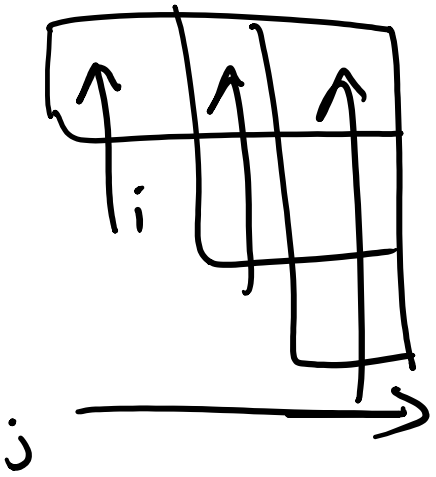


CKY Algorithm

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** $C \in table[k,j]\}$
        $table[i,j] \leftarrow table[i,j] \cup A$

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** $C \in table[k,j]\}$
        $table[i,j] \leftarrow table[i,j] \cup A$

n n 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

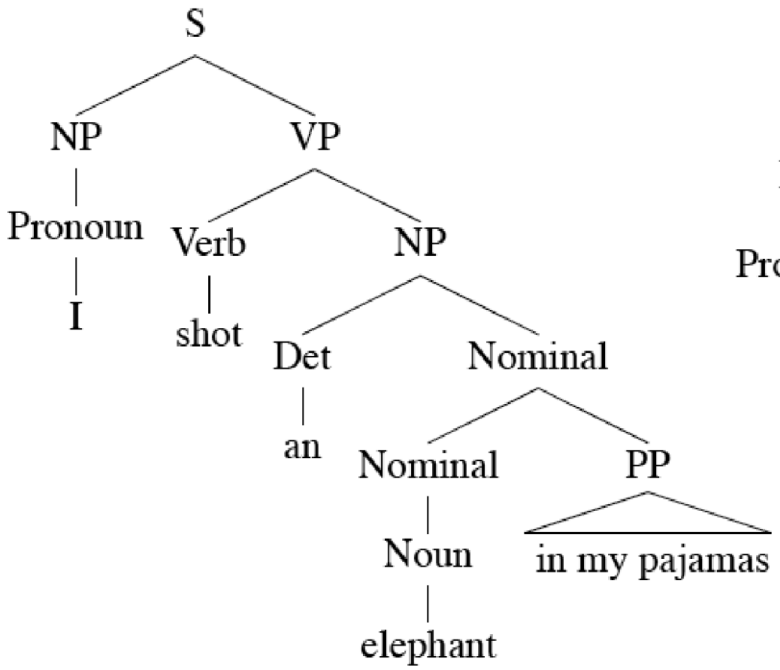
Parsing: CKY Algorithm

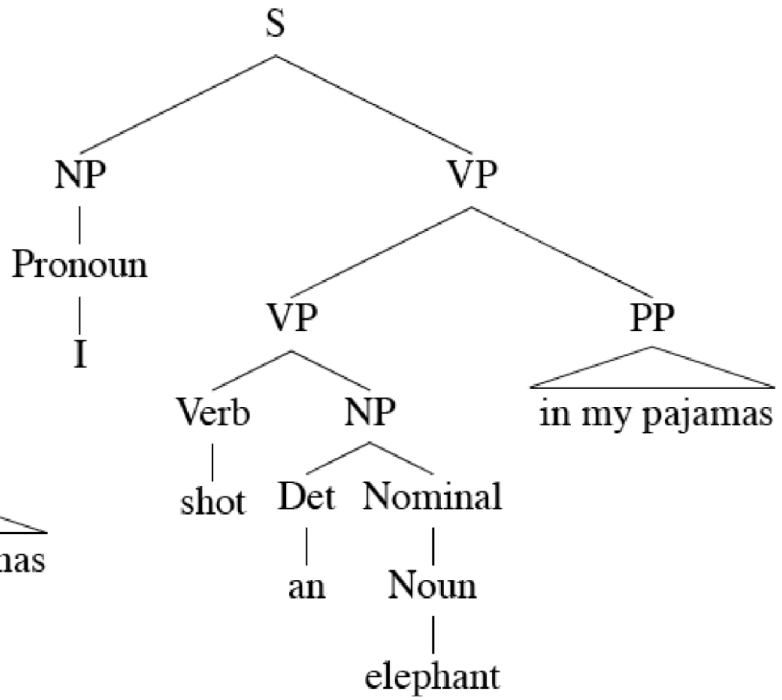
Extensions: Probabilistic and Lexicalized

Dependency Parsing

Ambiguity: Which parse?

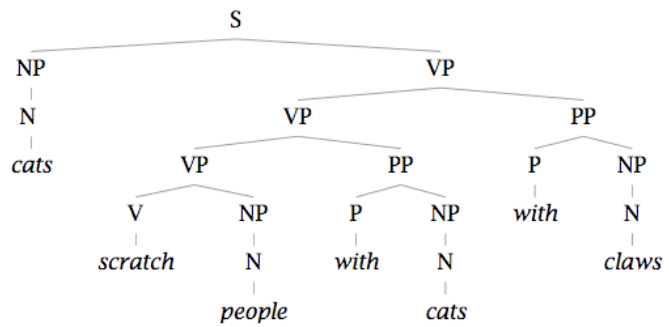
I shot an elephant in my pajamas.

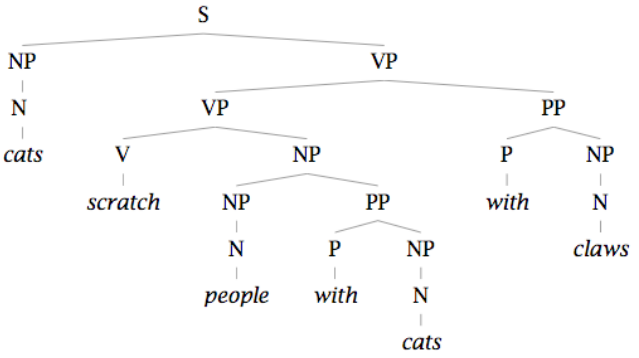


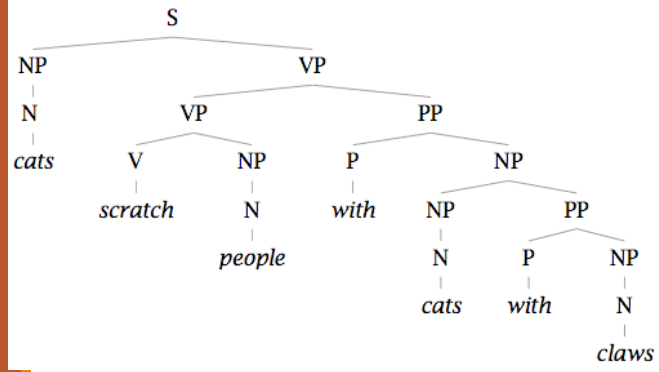


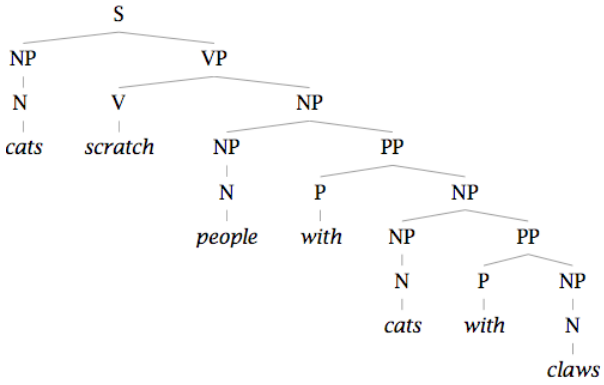
Finding the *Best* Parse Tree

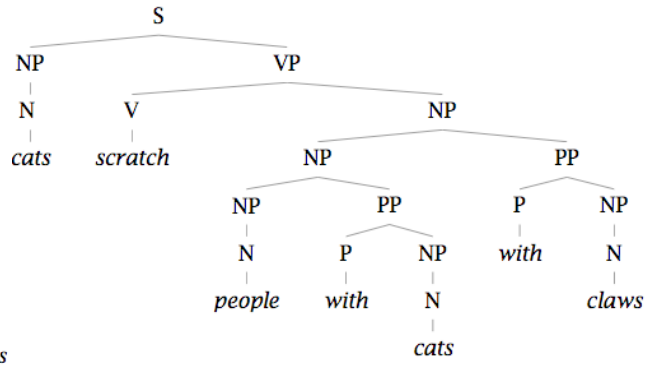
Cats scratch people with cats with claws.





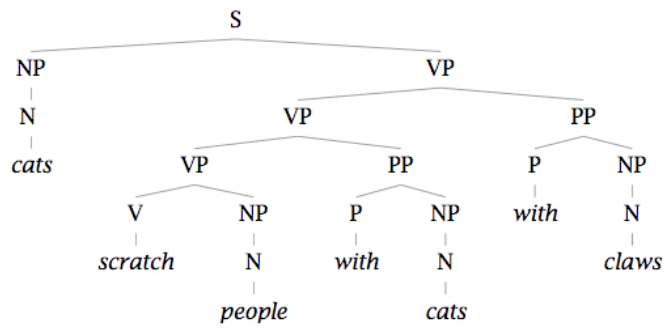


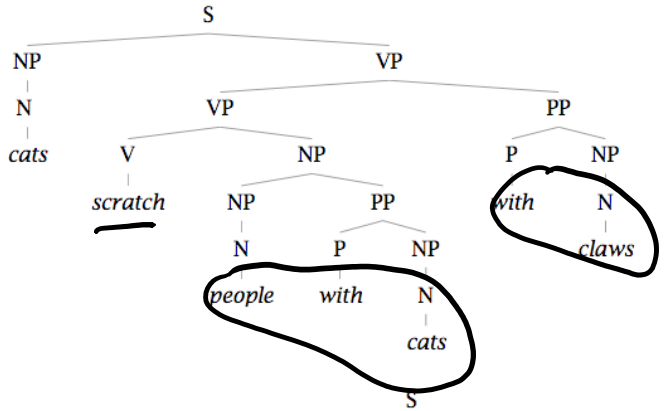


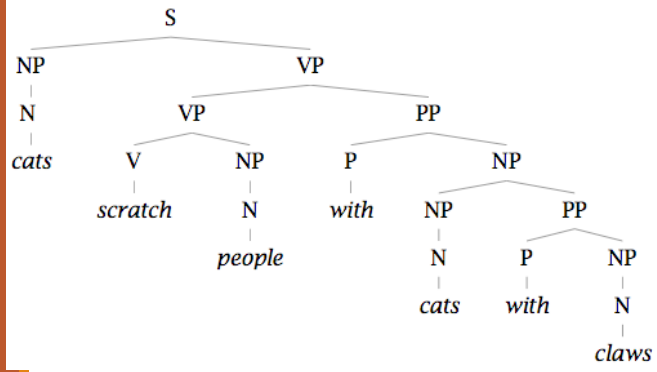


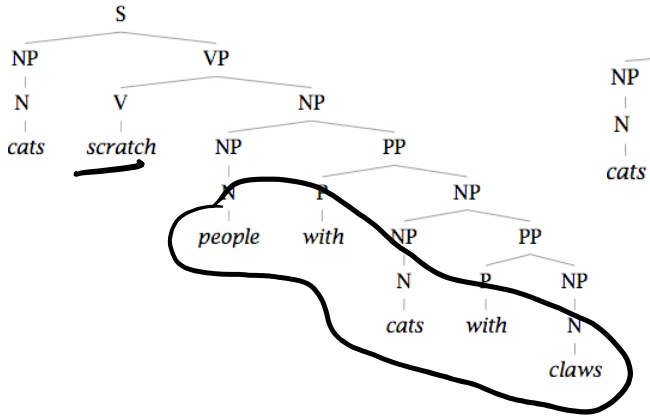
Finding the *Best* Parse Tree

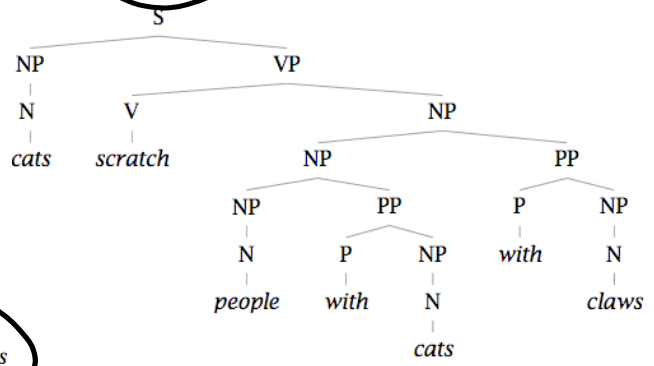
Cats scratch people with cats with claws.











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:

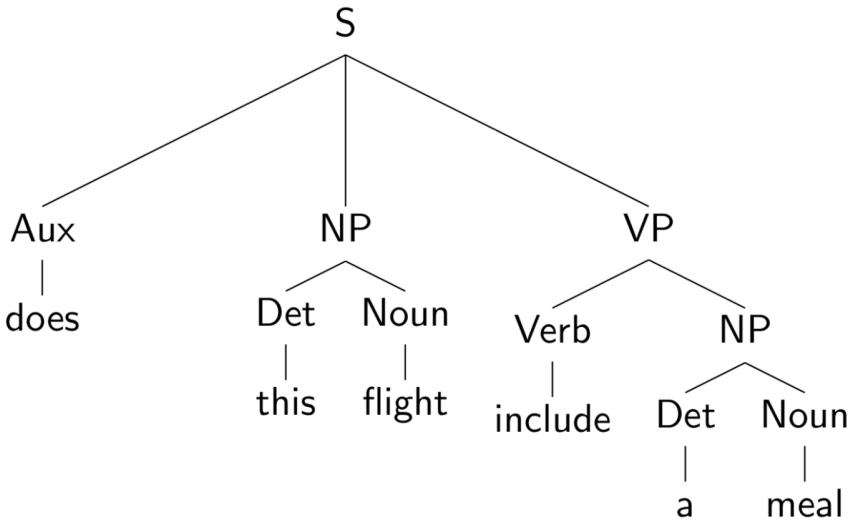
- Terminal, non-terminals, and rules
- Additionally, attach a probability to each rule!

Rule: $A \rightarrow B\ C$

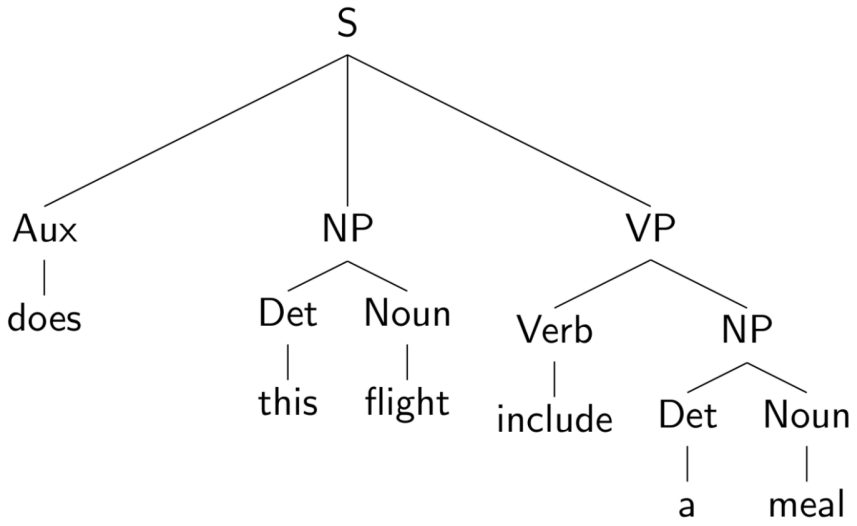
Probability: P($A \rightarrow B\ C \mid A$)

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



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

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

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

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

$\vdots$

Estimating the probabilities

Estimating the probabilities

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

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**.

$$\underset{t \in \mathcal{T}_x}{\operatorname{argmax}}\, p(t \mid x)$$

Even with small grammars, grows exponentially!

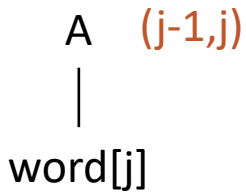
Probabilistic CKY Algorithm

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)

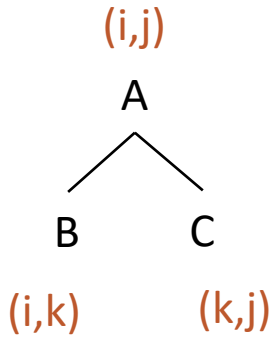


Recursion

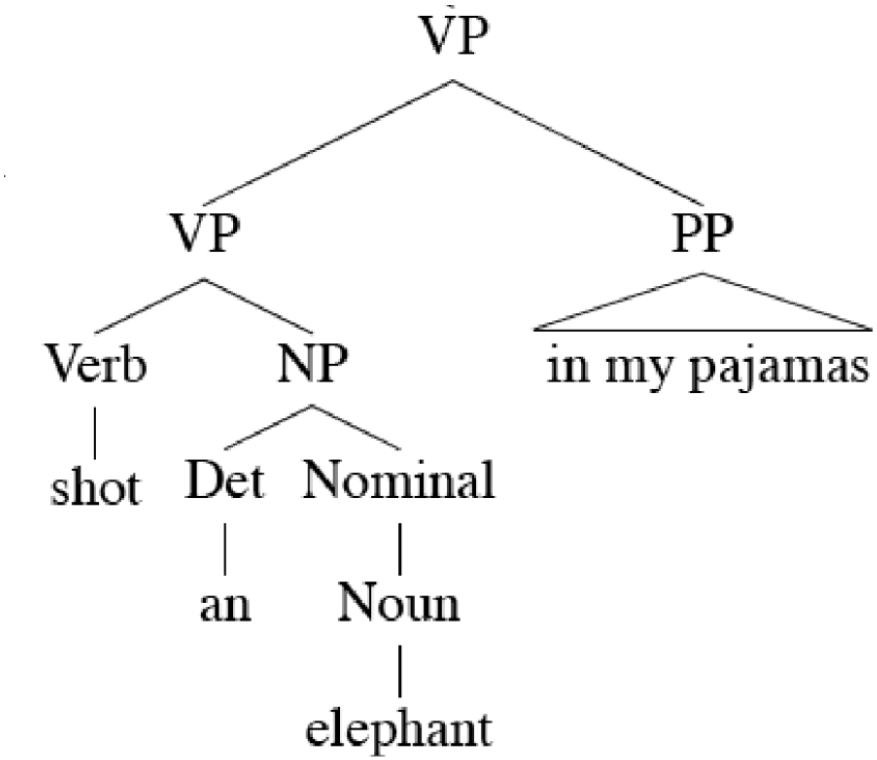
Rule: P($A \rightarrow B\ C$)

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

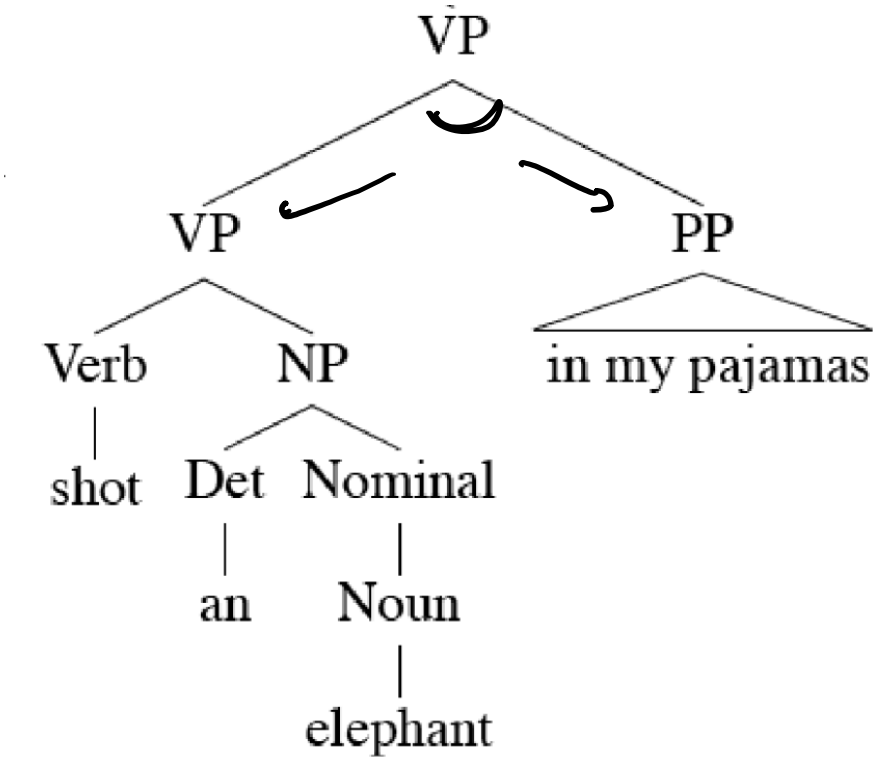
$$T[i,j,A] = \max_{k} P(B\ C \mid A)\ T[i,k,B]\ T[k,j,C]$$

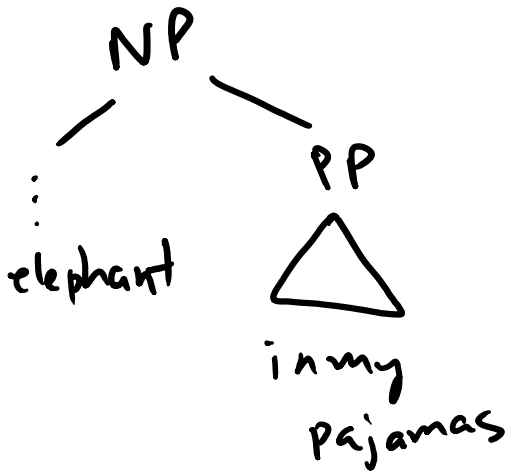


Lexicalized PCFGs

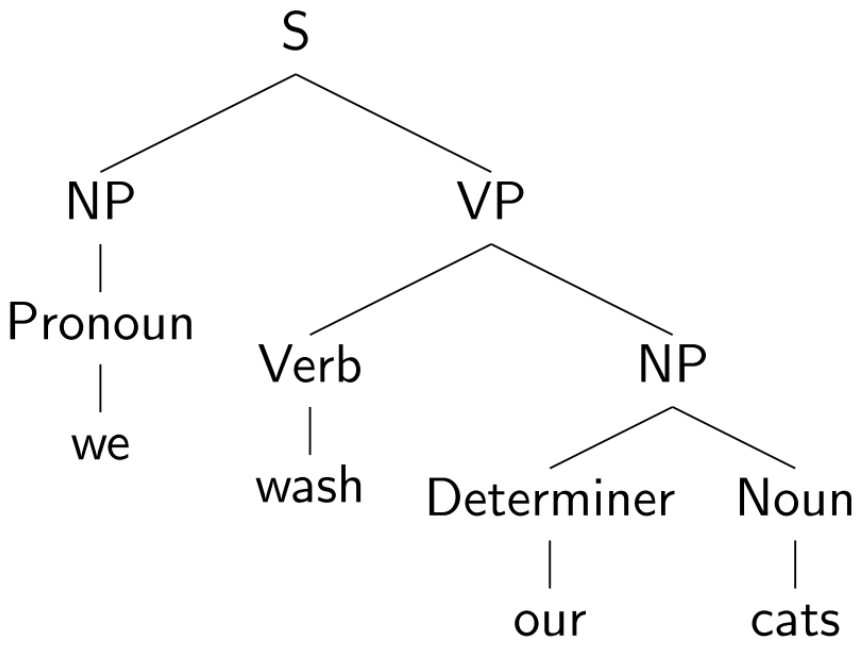


Lexicalized PCFGs

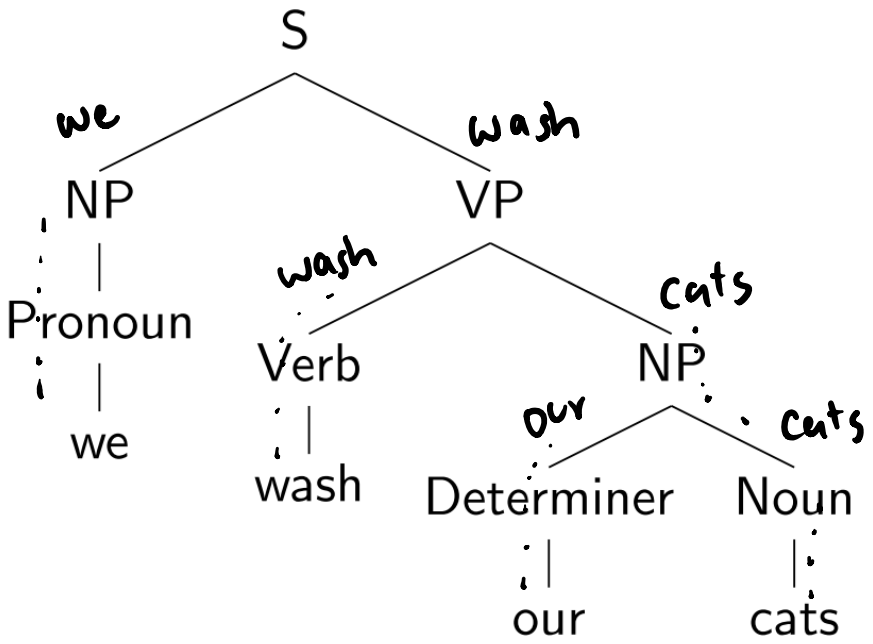




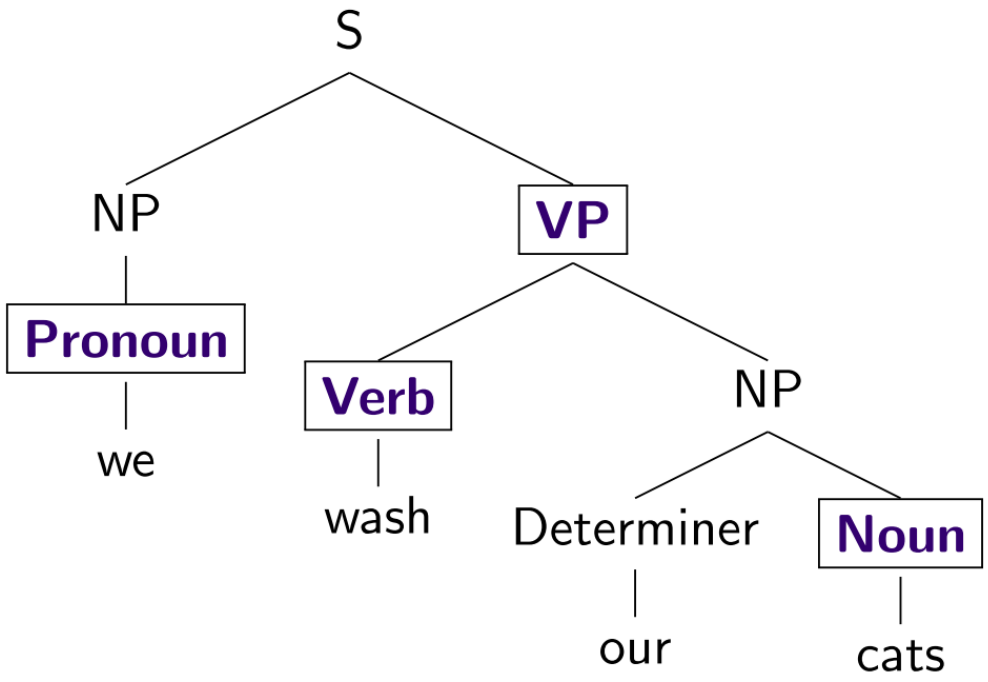
Lexicalizing a CFG



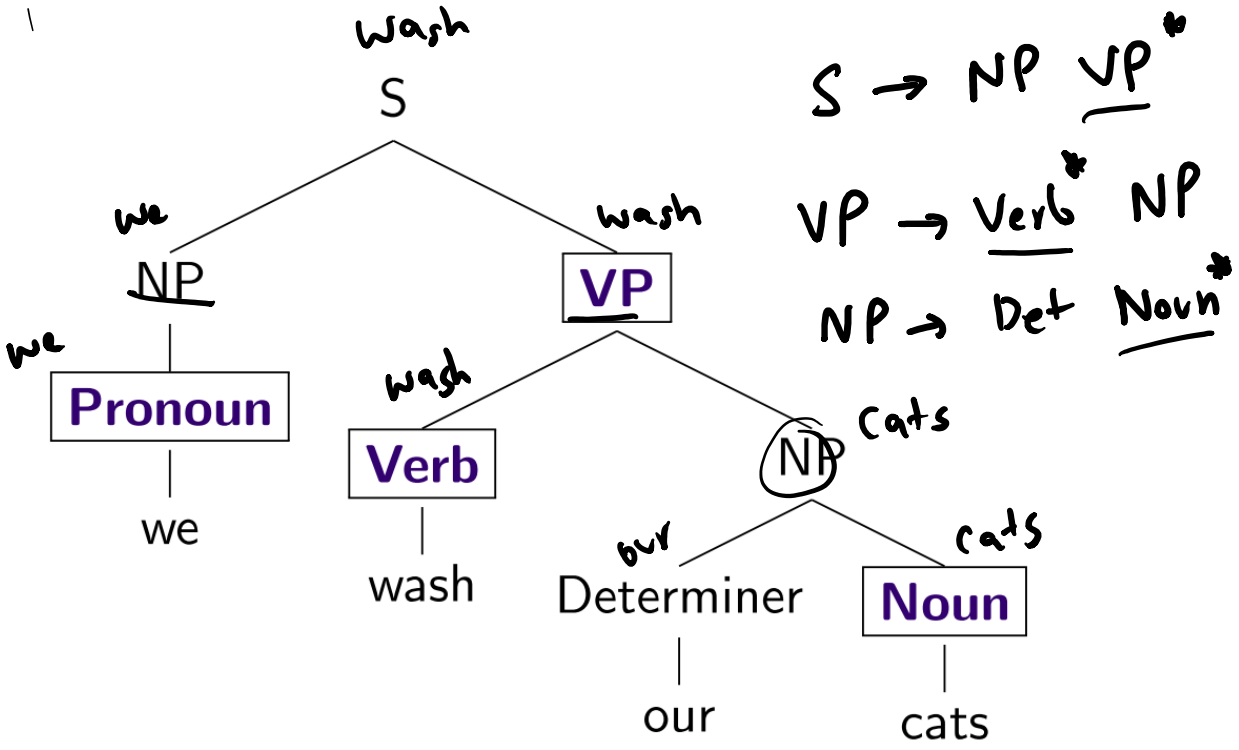
Lexicalizing a CFG



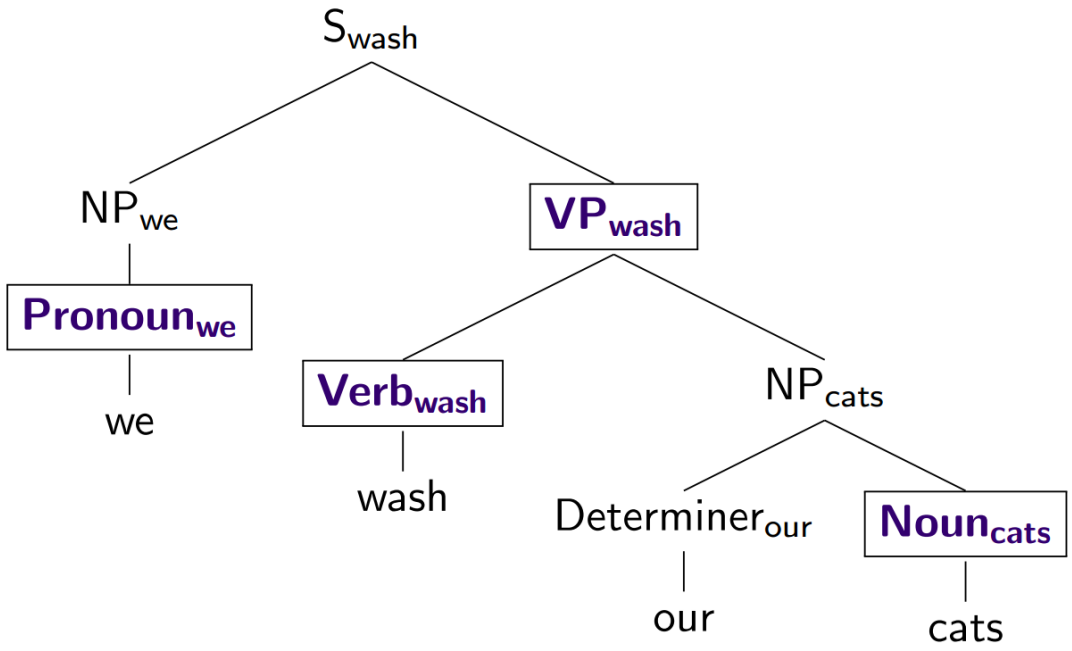
Lexicalizing a CFG



Lexicalizing a CFG



Lexicalizing a CFG



Outline

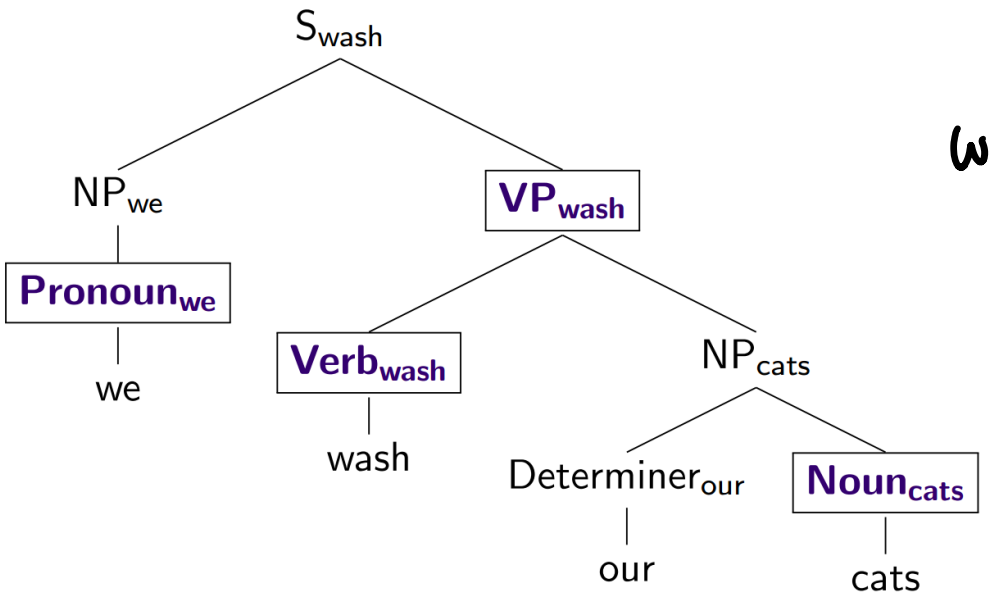
Parsing: CKY Algorithm

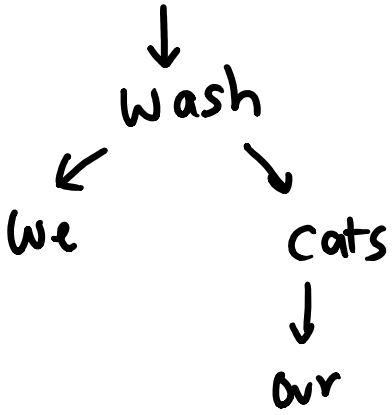
Extensions: Probabilistic and Lexicalized

Dependency Parsing

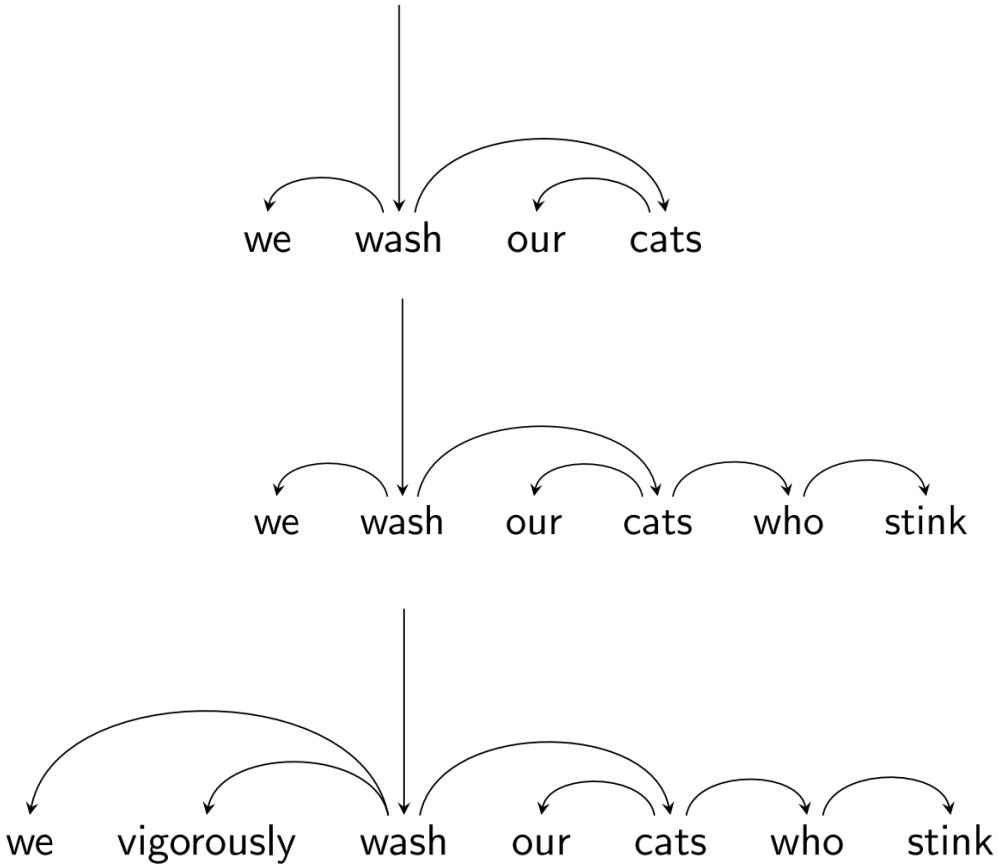
Dependencies

Represent only the syntactic dependencies…

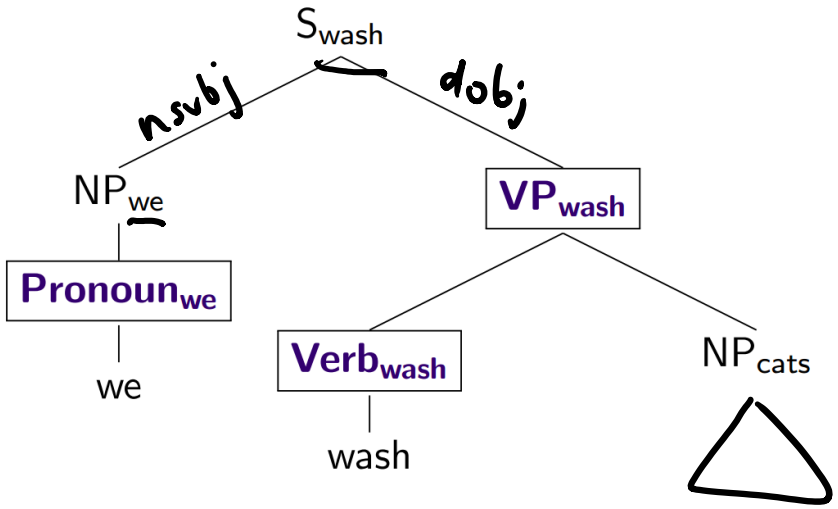


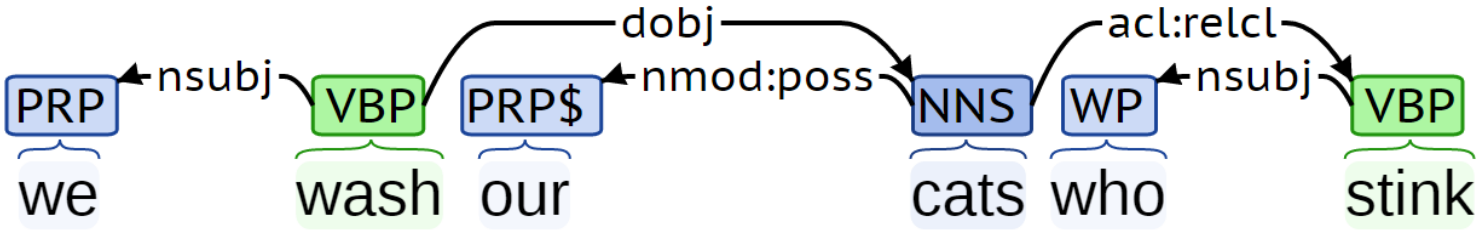


Nested Structure = Subtrees



Dependency Labels





Dependency Labels

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Dependency Trees

$$\{ \langle c_i, p_i, l_i \rangle \}_{i=1 \ldots n}$$

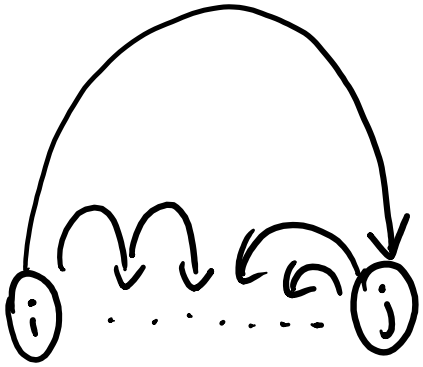
c_i: $1 : n$

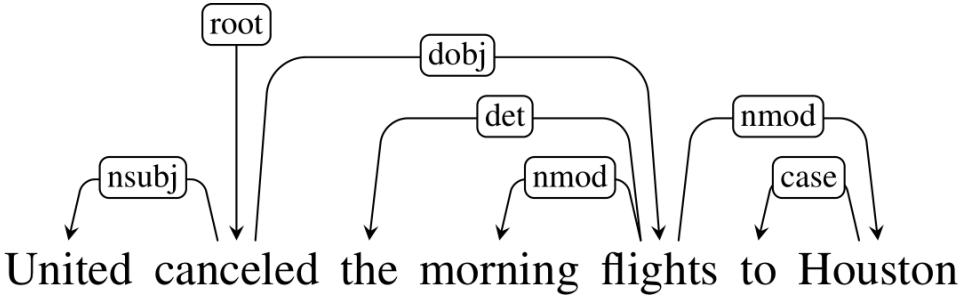
p_i: $0 : n$ (↑ root)

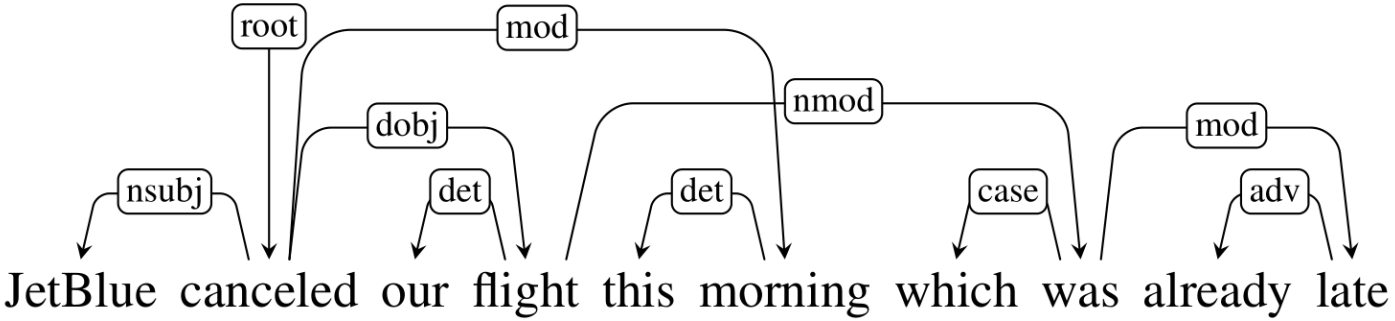
l_i: label

- single headed $\exists i$ only one $p_i = 0$
- connected
- acyclic
- Projective vs non-projective

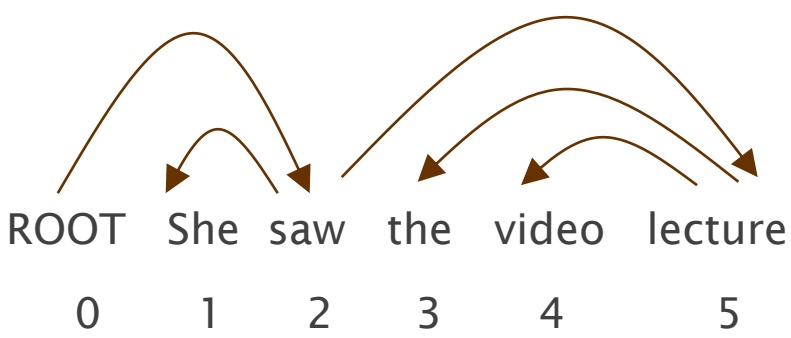
Projective vs Non-projective

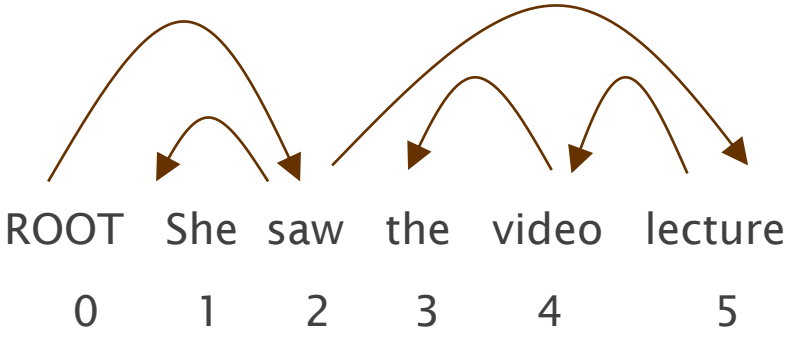






Evaluating Dependency Parses





Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	nsubj

UAS = $\frac{4}{5}$ = 80%

LAS = $\frac{2}{5}$ = 40%

Parsed

1	2 ✓	She	✓ ✓
2	0 ✗	saw	root ✗
3	4 ✓	the	det ✗
4	5 ✓	video	✗
5	2	lecture	nsubj

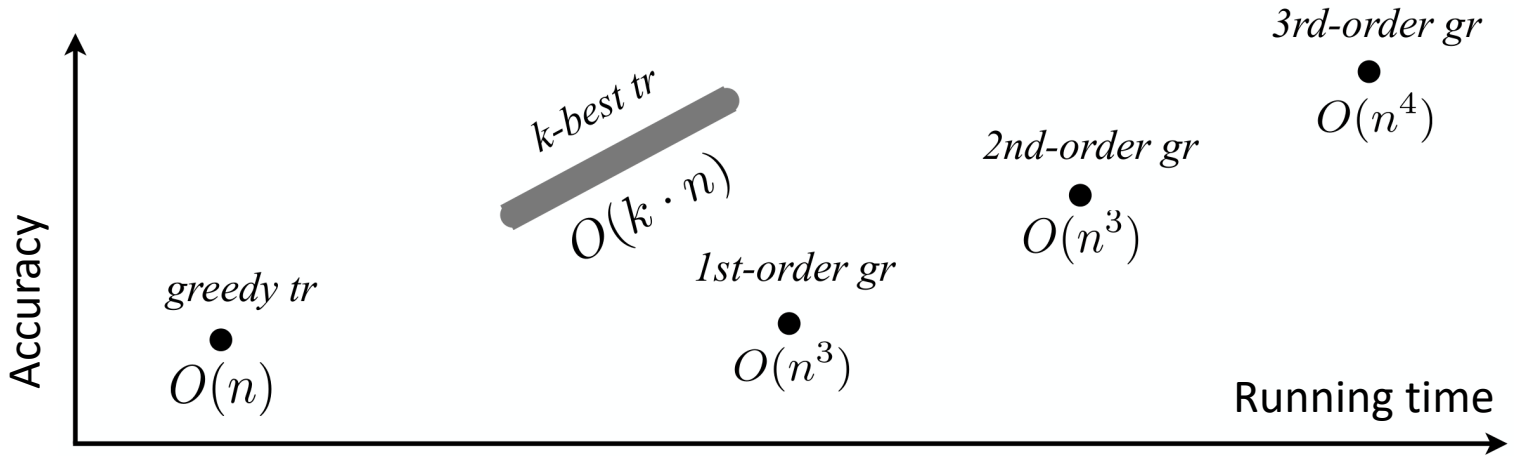
Parsing Algorithms

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

$$\operatorname*{argmax}_{t \in T} \; \text{score}(t, \theta)$$

$$\text{score}(t, \theta) = \begin{cases} \text{1st order / fully factored} \\ \sum_i \theta_i \, \phi(e_i) \quad (c_i, p_i, l_i) \end{cases}$$

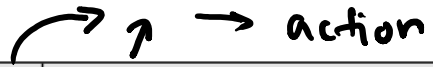
2nd order $\phi(e_i, e_j)$

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

Proj: Dynamic Programming

NonProj: Maximum Spanning Tree

Transition-based Parsing



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

$\theta \cdot \phi$(stack, wlist, action)