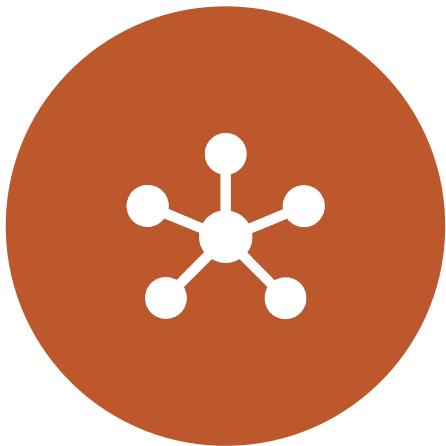


Grammars and Parsing

JURAFSKY AND MARTIN CHAPTERS 12, 13, AND 14

Reminders



HW6 IS DUE TONIGHT BY
11:59PM



HW7 HAS BEEN RELEASED. IT
IS DUE BEFORE SPRING BREAK

Prescriptive v. Descriptive Grammars

Concerned with establishing norms or rules for good writing.

For example:

1. It's important to never split infinitives
2. Prepositions are bad to end sentences with
3. Always have ten items or less

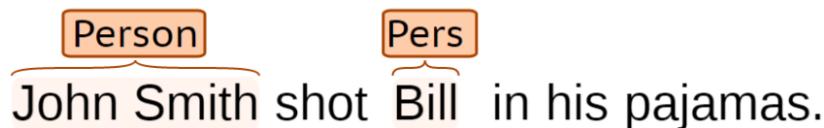
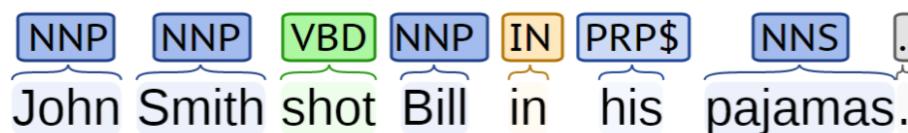
Describe how native speakers actually use a language.

Language is shaped by the speakers of a language and not dictated by central body.

Therefore rules should describe how language is used.

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

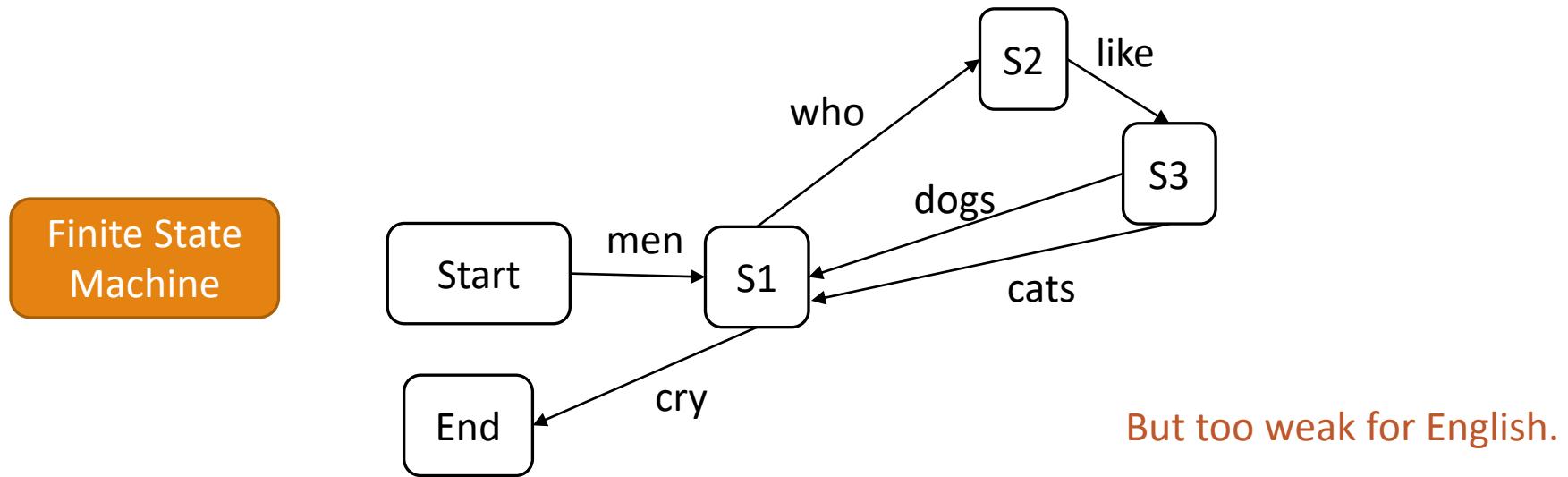
[send [the text message from James] [to Sharon]]

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

1. Grammar checkers
2. Dialog systems
3. High precision question answering
4. Named entity recognition
5. Sentence compression
6. Extracting opinions about products
7. Improved interaction in computer games
8. Helping linguists find data
9. Machine translation
10. Relation extraction systems

Basic Grammar: Regular Expr.

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



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

$A \rightarrow \beta \gamma \delta$

$A \rightarrow B A B$

$A \rightarrow \omega$

$A \rightarrow a \quad B \rightarrow b$

"a"

"bab" "baba" → x

→ b* a b* ←

Grammar applies rules recursively..

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

Context Free Grammars

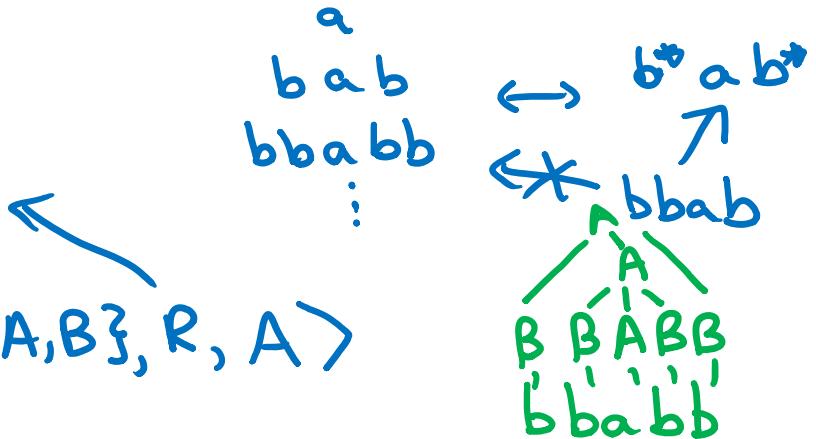
Context Free Grammars

$A \rightarrow ABA$

A > a

B → b

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



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
$S \rightarrow NP VP$	I + want a morning flight
$NP \rightarrow Pronoun$	I
$Proper-Noun$	Los Angeles
$Det Nominal$	a + flight
$Nominal \rightarrow Nominal Noun$	morning + flight
$Noun$	flights
$VP \rightarrow Verb$	do
$Verb NP$	want + a flight
$Verb NP PP$	leave + Boston + in the morning
$Verb PP$	leaving + on Thursday
$PP \rightarrow Preposition NP$	from + Los Angeles

"Lexicon"

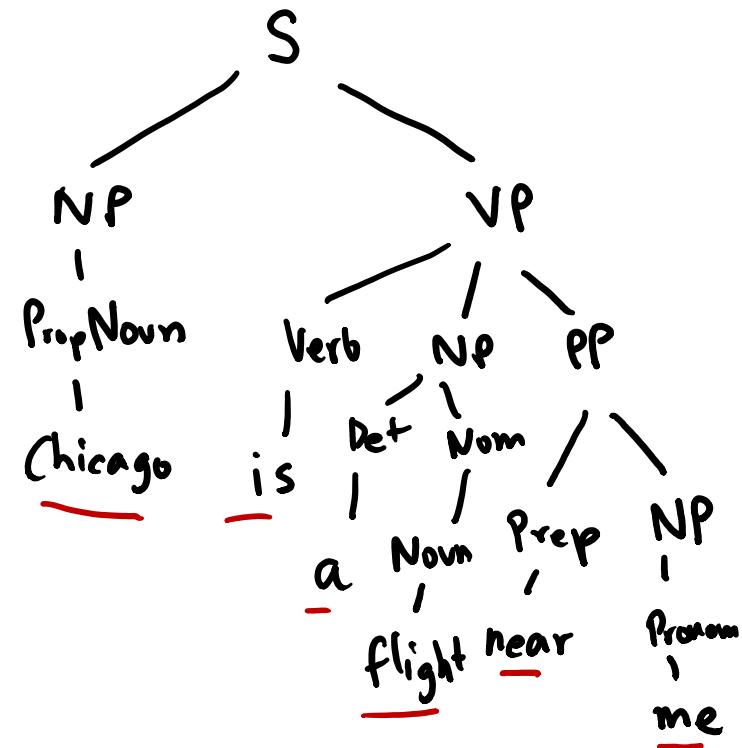
Example CFG

$$A \rightarrow w_1 \mid w_2$$

$$A \rightarrow w_1 \quad A \rightarrow w_2$$

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

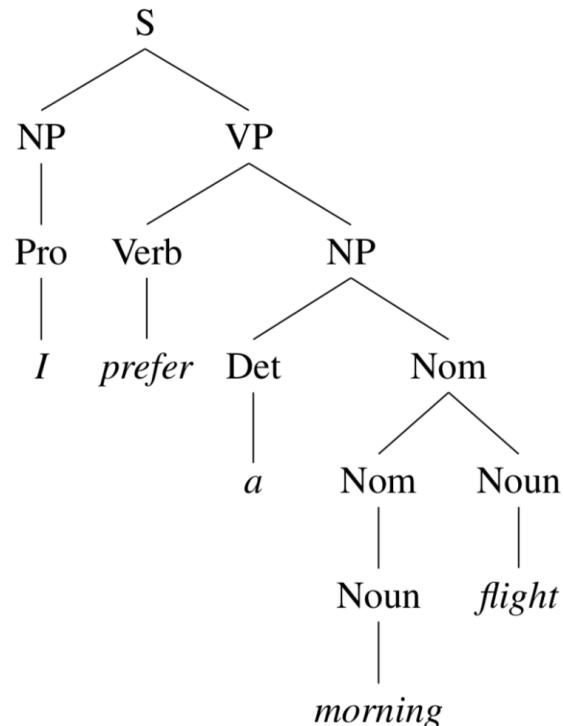
Grammar Rules	Examples
$S \rightarrow NP VP$	I + want a morning flight
$NP \rightarrow Pronoun$	I
$Proper-Noun$	Los Angeles
$Det Nominal$	a + flight
$Nominal \rightarrow Nominal\ Noun$	morning + flight
$Noun$	flights
$VP \rightarrow Verb$	do
$Verb\ NP$	want + a flight
$Verb\ NP\ PP$	leave + Boston +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



Example Parse Tree

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

I prefer a morning flight.

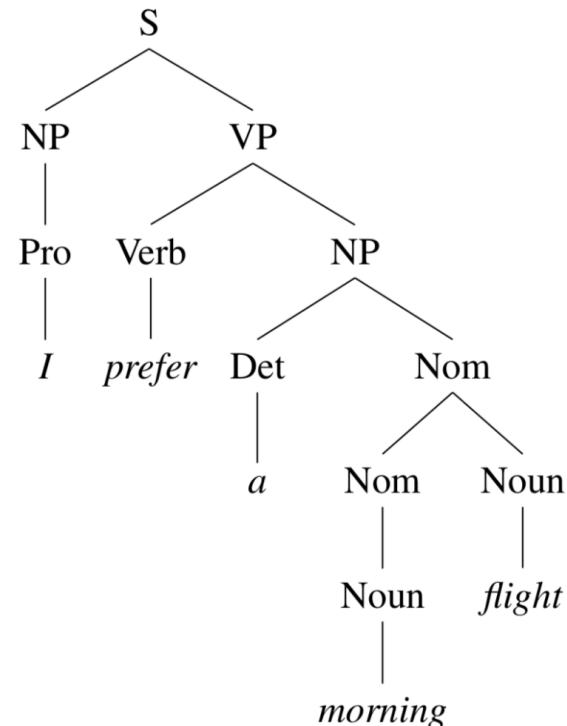


Example Parse Tree

$\text{Nom} \rightarrow N_{\text{time}} F$
 $F \rightarrow \text{flight}$ $N_{\text{time}} \rightarrow$ morning
 high
 evening

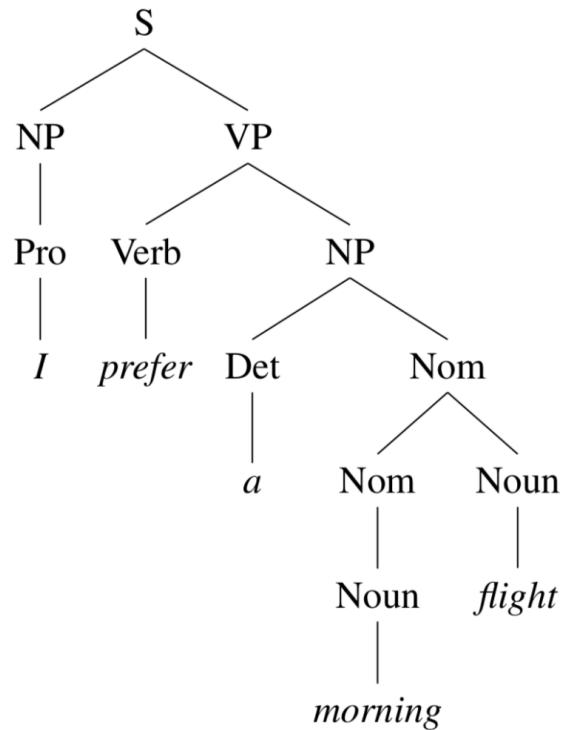
Grammar Rules	Examples
$S \rightarrow NP VP$	I + want a morning flight
$NP \rightarrow \text{Pronoun}$	I
 Proper-Noun	Los Angeles
Det Nominal	a + flight
$Nominal \rightarrow \text{Nominal Noun}$	morning + flight
Noun	flights
$VP \rightarrow \text{Verb}$	do
Verb NP	want + a flight
Verb NP PP	leave + Boston + in the morning
Verb PP	leaving + on Thursday
$PP \rightarrow \text{Preposition NP}$	from + Los Angeles

I prefer a morning flight.



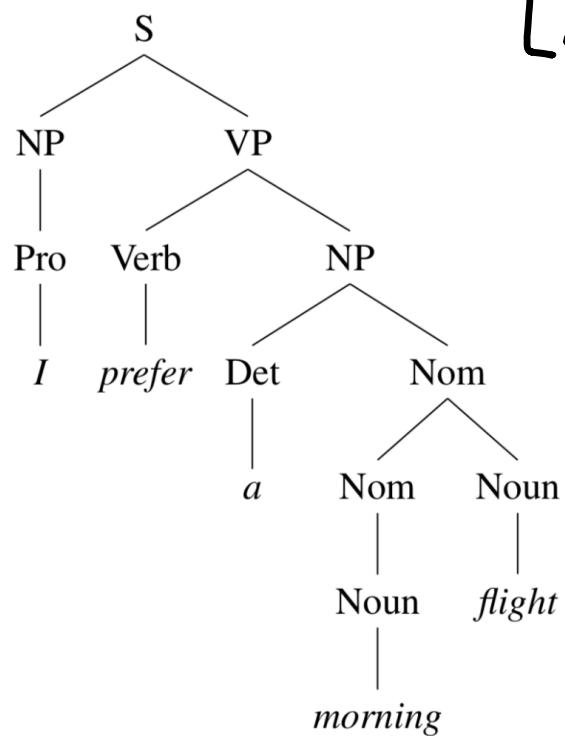
Example Parse Tree: Brackets

I prefer a morning flight.



Example Parse Tree: Brackets

I prefer a morning flight.



$[S [NP [Pro I]] [VP [Verb prefer] [NP [Det a] [Nom [Noun morning]]]]]$

More details: Noun Phrases

Simple Noun Phrases

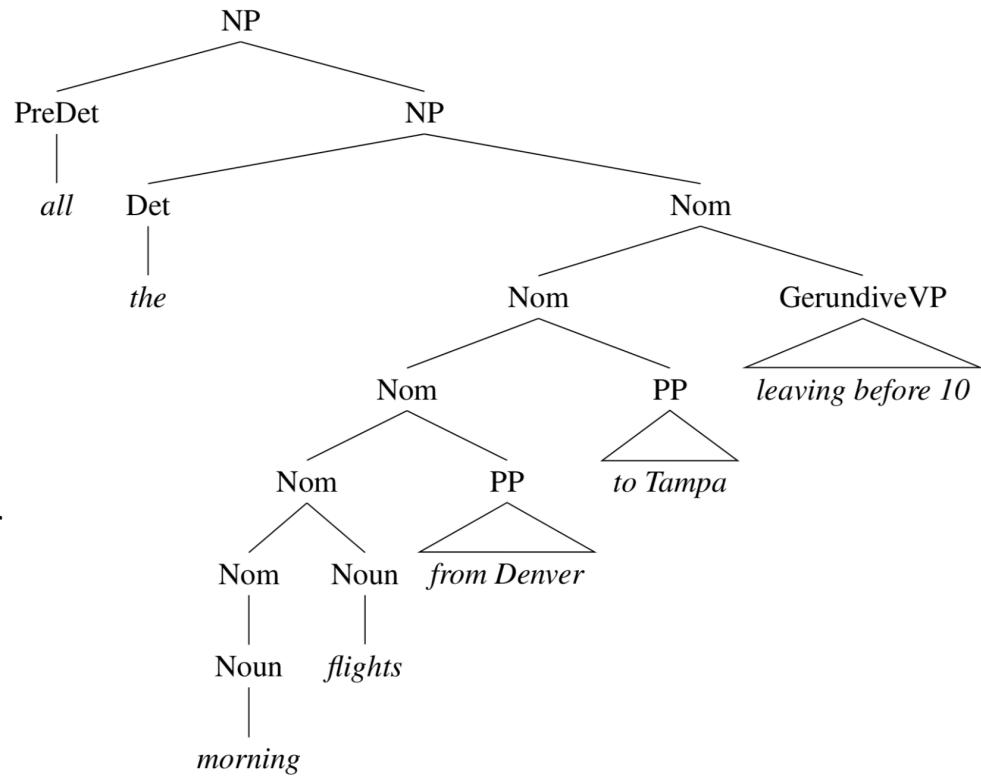
NP → ProperNoun

NP → Det Nominal

Nominal → Noun | Noun Nominal

Complex Noun Phrases

“all the morning flights from Denver
to Tampa leaving before 10”



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?

A Computer Model of a Grammar for English Questions

Chris Callison-Burch

June 2000

Contents

1	Introduction	3
1.1	“Grammar Engineering” Explained	3
1.2	The LKB	4
1.3	Downloading the Software	4
2	Summary of Ginzburg and Sag (2000)	6
2.1	Overview of HPSG	6
2.2	Semantic Argument Selection	8
2.3	The Content of Questions	10
2.4	Polar Interrogative Constructions	12
2.5	<i>Wh</i> -Questions	14
2.5.1	Unbounded Dependencies	14
2.5.2	Nonsubject <i>Wh</i> -Interrogatives	17
2.5.3	Subject <i>Wh</i> -Interrogatives	19
2.6	Multiple <i>Wh</i> -Questions	21
3	Theory vs. Implementation	28
3.1	Determinacy in Grammar Rule Length	28
3.2	Difference Lists	29
3.3	Pumping Rules and Intermediate Types	31
3.4	Sets Treated as Lists	34
4	Key Examples	35
4.1	Simple Declaratives	35
4.1.1	Semantic Subcategorization	38
4.2	Focused and Unfocused “Do”	40
4.3	Non-subject <i>Wh</i> -Interrogatives	42
4.3.1	Distinguished From Topicalization	44
4.3.2	As Embedded Questions	44
4.4	Subject <i>Wh</i> -Interrogatives	45
4.5	Sensitivity to the Presence of <i>Wh</i> -Words	46
4.5.1	Pied Piping	46
4.5.2	“The Hell” Examples	47
4.6	Multiple <i>Wh</i> -Questions	49

Clauses versus phrases

A clause expresses a “complete thought”. One example is a complete sentence, which has a main **verb** and all of its **arguments** filled.

I prefer a morning flight

Verb: Prefer

Arguments:

1. The subject (the holder of the preference)
2. The direct object (the thing preferred)

The Noun Phrase

Three of the most frequent types of noun phases in English are:

1. Pronouns
2. Proper nouns
3. The construction $NP \rightarrow Det\ Nominal$

For the third type, often, we have simple determiner like:

a flight	the flights	this flight
those flights	any flights	some flight

Remember that mass nouns don't require a determiner.

Complex determiners

Sometimes instead of single words like **a, the, those, this, some, any** determiners can be complex phrases themselves.

In English this can have with the possessive marker '**s**'.

CCB's laptop

CCB's laptop's broken keyboard

CCB's laptop's keyboard's butterfly keys

This is our first example of recursive rules.

$NP \rightarrow Det\ Nominal$

$Det \rightarrow NP\ 's$

Nominals

$NP \rightarrow Det\ Nominal$

In English, a *nominal* follows a determiner. This can be as simple as a single noun

Nominal \rightarrow *Noun*

But can also be more complex with **pre-** or **post-head noun modifiers**.

Before the noun	After the head noun
Cardinal and ordinal numbers	prepositional phrases
Quantifiers (<i>many, few, several</i>)	non-finite clauses
Adjectives (<i>first-class, earliest</i>)	relative clauses

Recursive Noun Phrases

this is the house

this is the house that Jack built

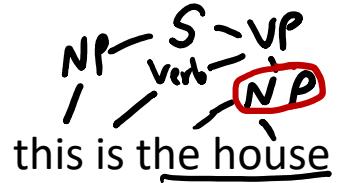
this is the cat that lives in the house that Jack built

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 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

Verb Phrases

Simple Verb Phrases

$VP \rightarrow Verb$

$VP \rightarrow Verb\ NP$

$VP \rightarrow Verb\ NP\ PP$

$VP \rightarrow Verb\ PP$

disappear

prefer a morning flight

leave Boston in the morning

leave in the morning

But all verbs are not the same!
(this grammar over-generates)

Solution: subcategorize!

Sneezed: Skyler sneezed.

Read: Gaurav read the book

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.

Subcategorization frames

Frame	Verb	Example
–	eat, sleep	I ate
NP	prefer, find, leave	Find [NP the one ring]
NP NP	show, give	Give [NP Frodo] [NP the ring]
PP _{from} PP _{to}	fly, travel	Fly [PP from Philadelphia] [PP to Honolulu]
NP PP _{with}	help	Help [NP me] [PP with my homework]
VP _{infinitival}	prefer, want, need	I would prefer [VP to finish] early
VP _{bare}	can, would, might	I can [VP finish] my homework
S	mean	Does this mean [S we are done]?

Expanding our rule set

We could create subtypes of the class Verb capture the association between verbs and their complements:

Verb-with-NP-complement → *find| leave| repeat| ...*

Verb-with-S-complement → *think| believe| say| ...*

Verb-with-Inf-VP-complement → *want| try| need| ...*

Then modify each VP rule to give the right argument types:

VP → *Verb-with-no-complement*

VP → *Verb-with-NP-comp NP*

VP → *Verb-with-S-comp S*

Predicate-argument relations

Another way of talking about the relation between the verb and these other constituents is to think of the verb as a logical predicate and the constituents as logical arguments of the predicate.

FIND(BILBO, THE RING)

WANTS(SAURON, THE RING)

THROW(FRODO, THE RING, MOUNT DOOM)

Coordination

Conjunctions like **and**, **or**, and **but** can form larger constructions of the same type. For example, a coordinate noun phrase can consist of two other noun phrases separated by a conjunction:

Please repeat [NP [NP the flights] and [NP the costs]]

I need to know [NP [NP the aircraft] and [NP the flight number]]

Here's a rule that allows these structures:

$NP \rightarrow NP \text{ and } NP$

This also works with other types, like VP, S and Nominal. So some people create “Metarules” like

$X \rightarrow X \text{ and } X$

Source of Grammar?

Manual



Noam Chomsky

Write symbolic grammar (CFG or often richer) and lexicon

$S \rightarrow NP\ VP$	$NN \rightarrow interest$
$NP \rightarrow (DT)\ NN$	$NNS \rightarrow rates$
$NP \rightarrow NN\ NNS$	$NNS \rightarrow raises$
$NP \rightarrow NNP$	$VBP \rightarrow interest$
$VP \rightarrow V\ NP$	$VBZ \rightarrow rates$

Used grammar/proof systems to prove parses from words

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

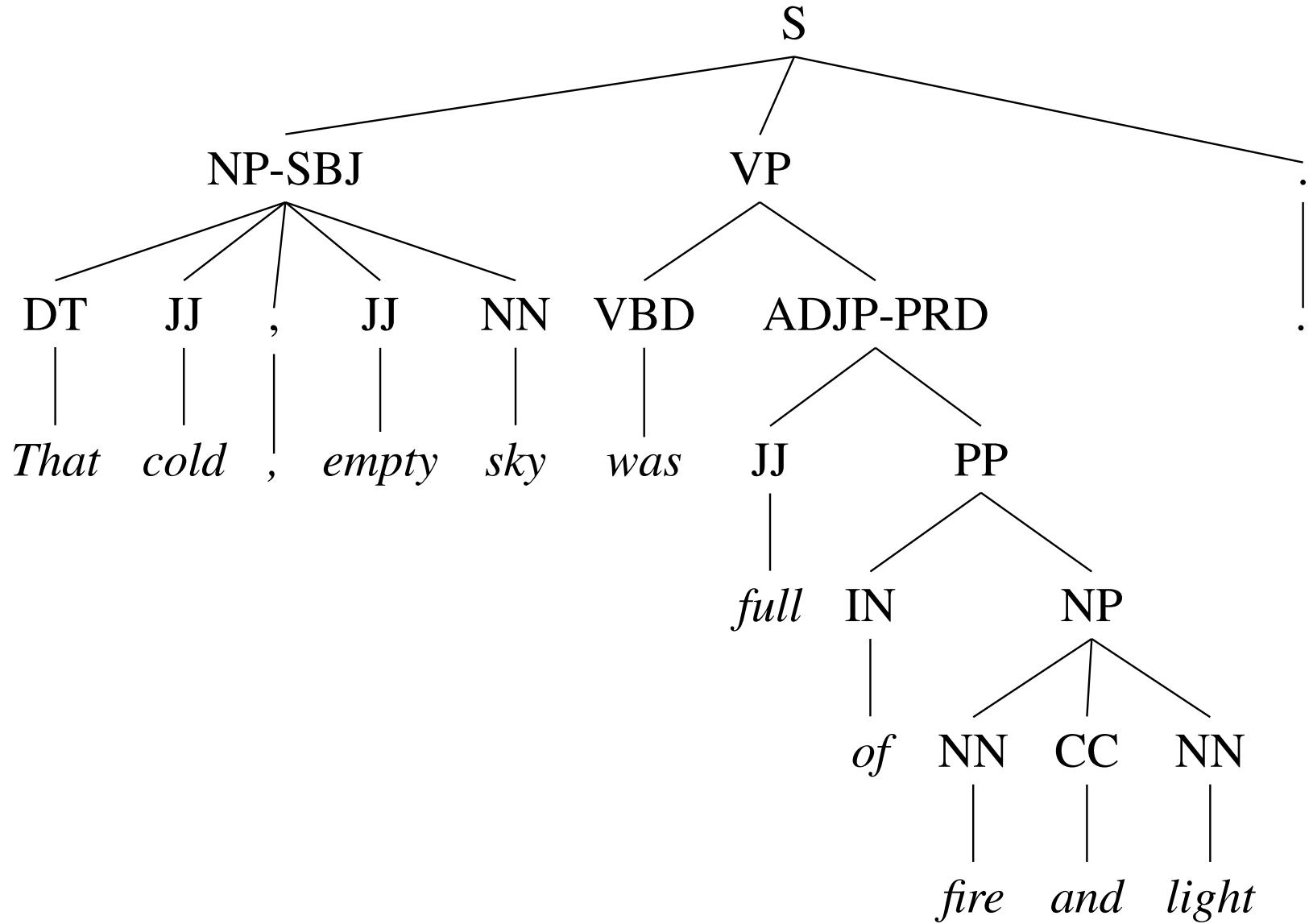
Building a treebank seems a lot slower and less useful than building a grammar

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 That)
 (JJ cold) (, ,)
 (JJ empty) (NN sky))
 (VP (VBD was)
 (ADJP-PRD (JJ full)
 (PP (IN of)
 (NP (NN fire)
 (CC and)
 (NN light)))))
 (. .))))



```
( (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	100 VP → VBD PP-PRD
33803 S → NP-SBJ VP	100 PRN → : NP :
22513 NP-SBJ → -NONE-	100 NP → DT JJS
21877 NP → NP PP	100 NP-CLR → NN
20740 NP → DT NN	99 NP-SBJ-1 → DT NNP
14153 S → NP-SBJ VP .	98 VP → VBN NP PP-DIR
12922 VP → TO VP	98 VP → VBD PP-TMP
11881 PP-LOC → IN NP	98 PP-TMP → VBG NP
11467 NP-SBJ → PRP	97 VP → VBD ADVP-TMP VP
11378 NP → -NONE-	...
11291 NP → NN	10 WHNP-1 → WRB JJ
...	10 VP → VP CC VP PP-TMP
989 VP → VBG S	10 VP → VP CC VP ADVP-MNR
985 NP-SBJ → NN	10 VP → VBZ S , SBAR-ADV
983 PP-MNR → IN NP	10 VP → VBZ S ADVP-TMP
983 NP-SBJ → DT	
969 VP → VBN VP	

4500 rules
for VP!

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

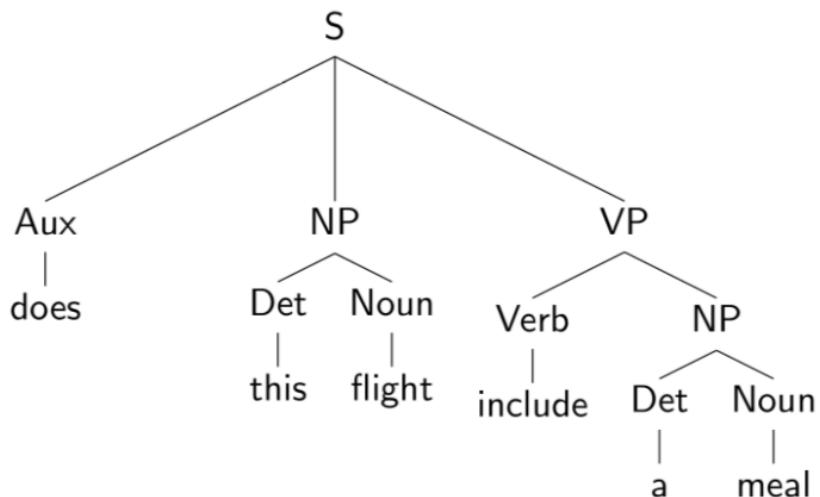
NP → DT JJ NNS , NNS CC NN NNS NN NP → DT JJ JJ VBG NN NNP NNP
FW NNP NP → NP JJ , JJ " SBAR " NNS

[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]

Evaluating Parses

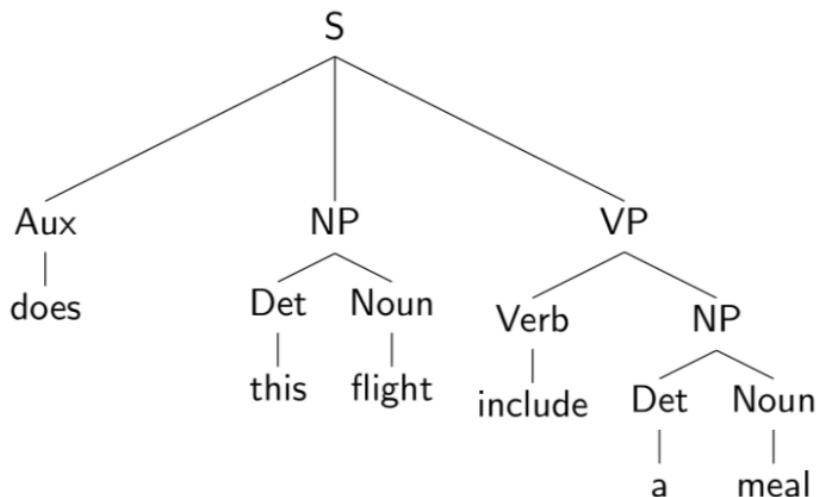
Each parse tree is represented by a list of tuples:



Use this to estimate precision/recall!

Evaluating Parses

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



$\langle S, 0, 6 \rangle \quad \langle \text{Aux}, 0, 1 \rangle$

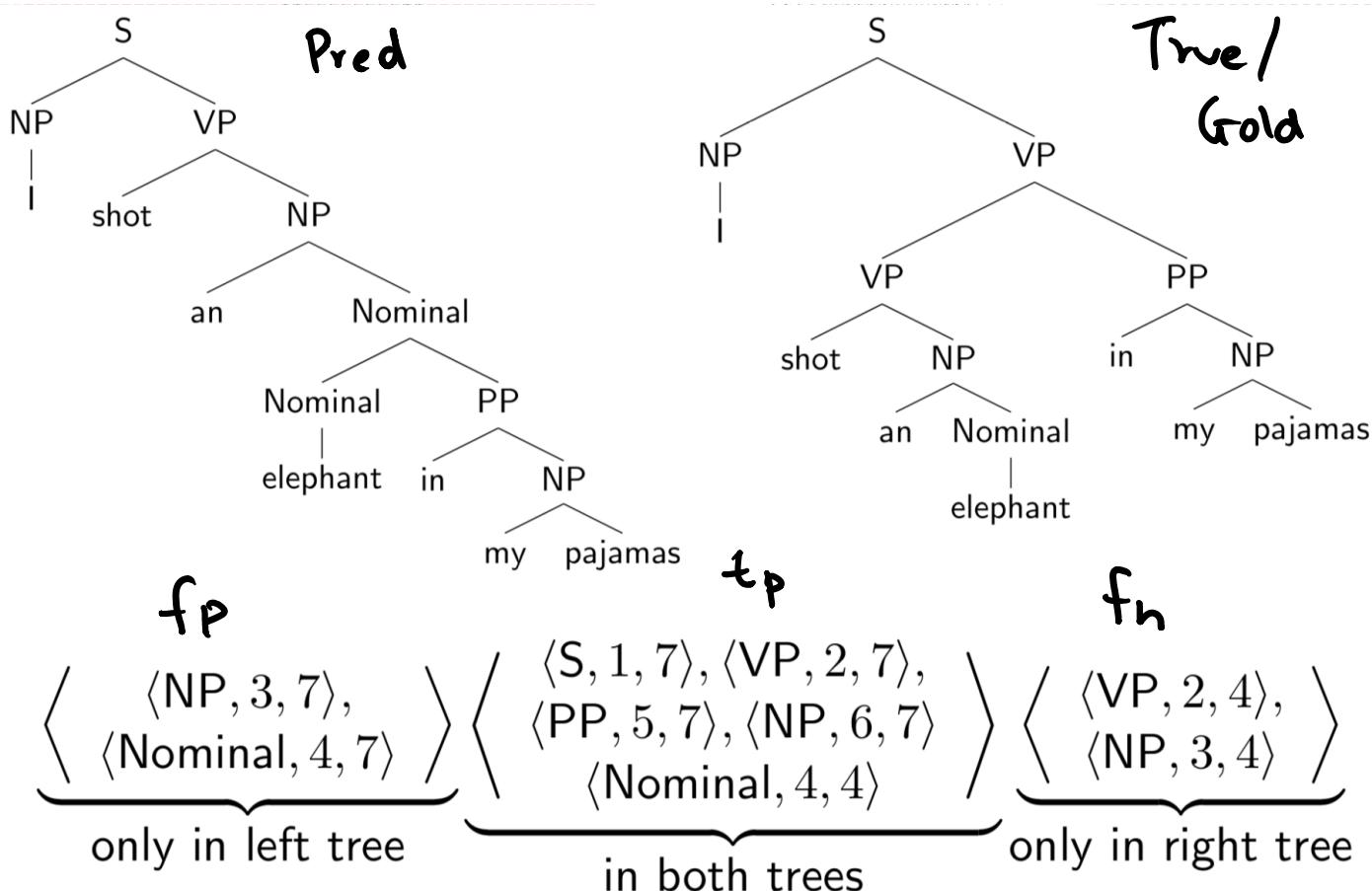
$\langle NP, 1, 3 \rangle \quad \langle \text{DET}, 1, 2 \rangle$

$\langle \text{Noun}, 2, 3 \rangle \quad \langle NP, 4, 6 \rangle$

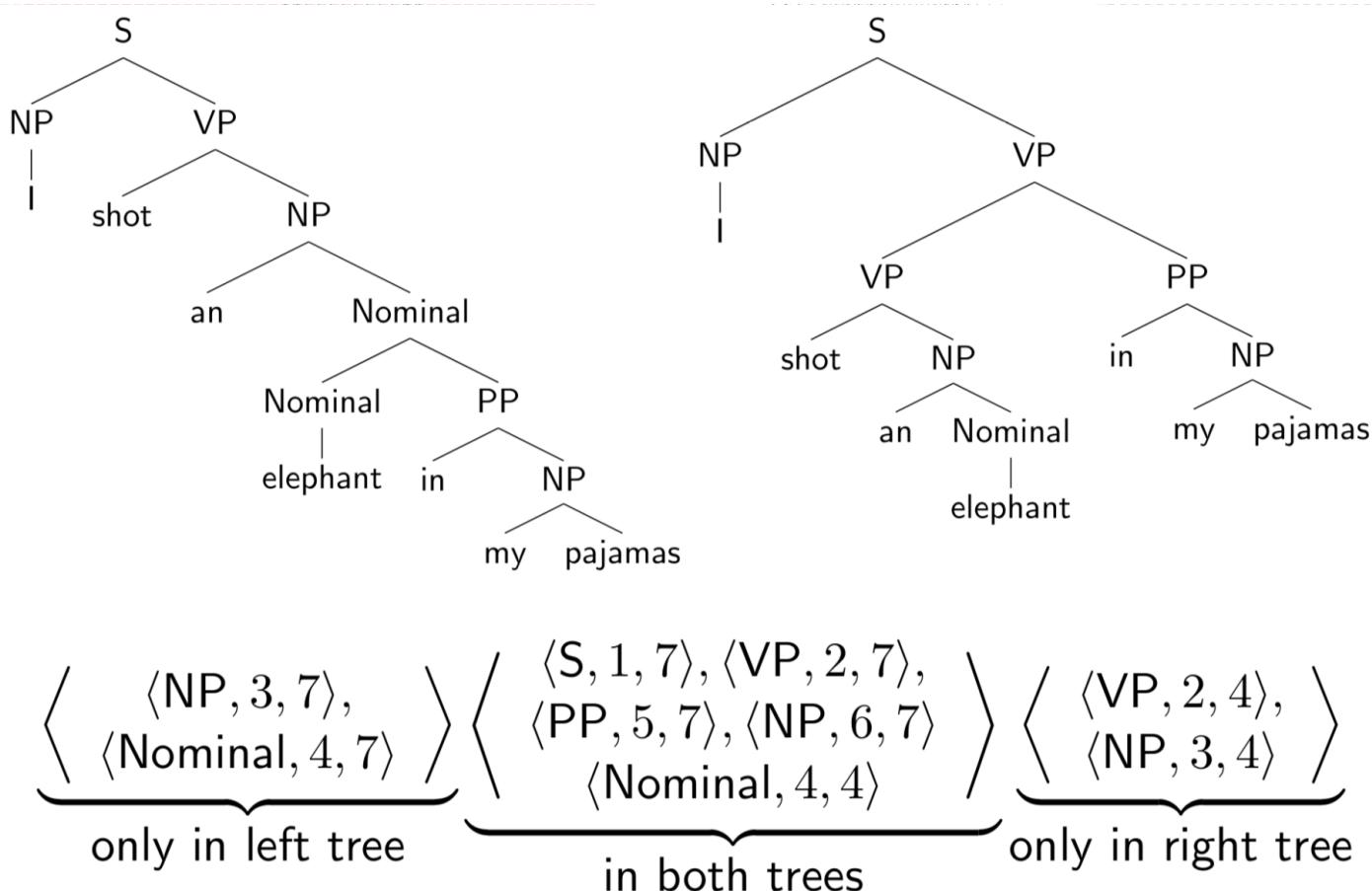
$\langle VP, 3, 6 \rangle \quad \dots$

Use this to estimate precision/recall!

Evaluating Parses: Example



Evaluating Parses: Example



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”

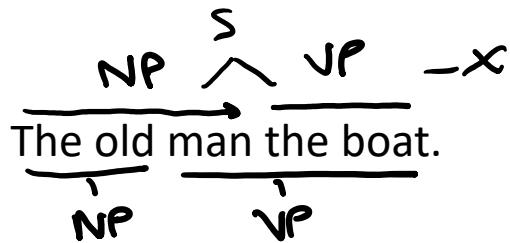
Left to Right?

The old man the boat.

The complex houses married and single soldiers and their families.

Garden Path Sentences

Left to Right?



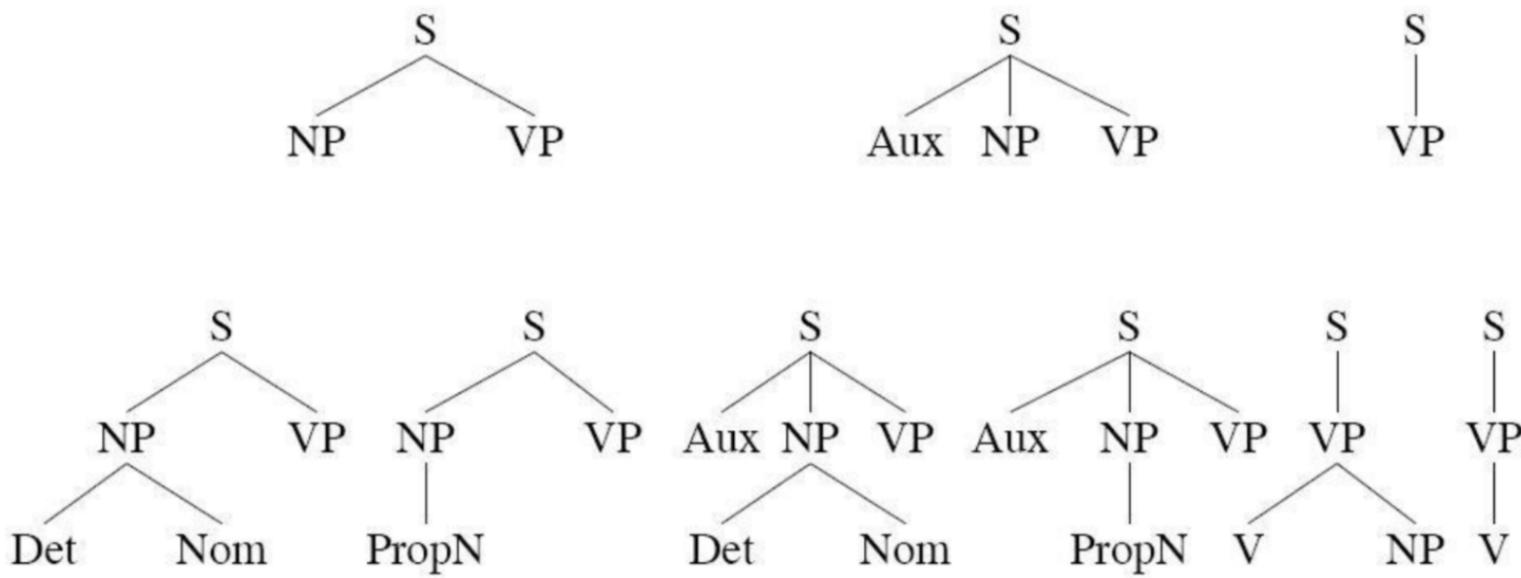
Garden Path Sentences

Top Down Parsing

Considers only valid trees
But are inconsistent with the words!

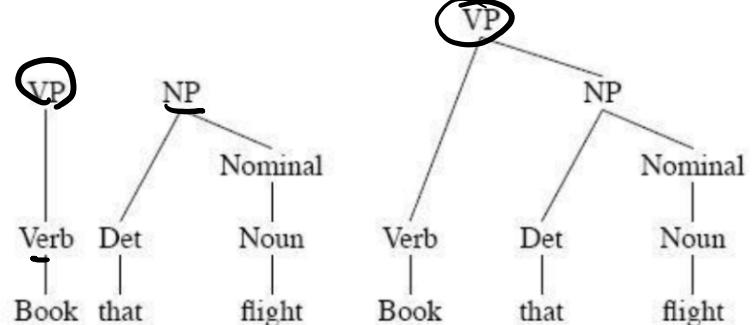
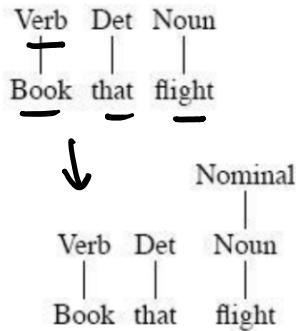
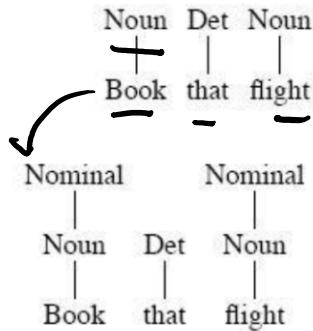
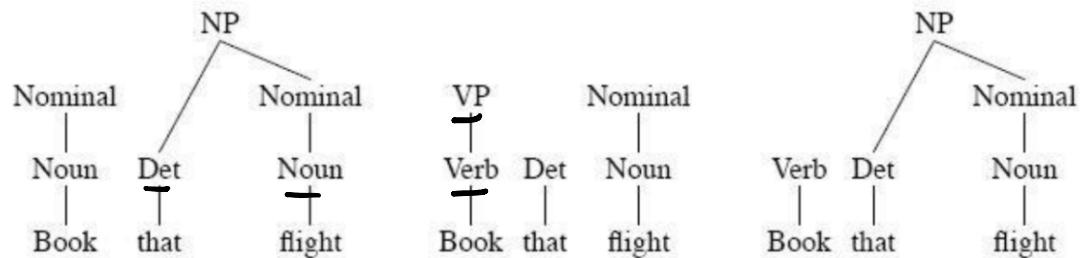
“Book that flight”

S



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

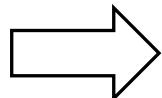
$$A \rightarrow B C$$

$$D \rightarrow w$$

Converting to CNF

Case 1

$$\begin{array}{l} A \rightarrow B \\ B \rightarrow C D \\ B \rightarrow w \end{array}$$



$$\begin{array}{l} A \rightarrow C D \\ A \rightarrow w \end{array}$$

Case 2

$$A \rightarrow B C D E$$



$$\begin{array}{l} A \rightarrow X E \\ X \rightarrow Y D \\ Y \rightarrow B C \end{array}$$

Original Grammar

$S \rightarrow NP VP$

$S \rightarrow Aux NP VP$

$S \rightarrow VP$

$NP \rightarrow Pronoun$

$NP \rightarrow Proper-Noun$

$NP \rightarrow Det Nominal$

$Nominal \rightarrow Noun$

$Nominal \rightarrow Nominal Noun$

$Nominal \rightarrow Nominal PP$

$VP \rightarrow Verb$

$VP \rightarrow Verb NP$

$VP \rightarrow Verb NP PP$

$VP \rightarrow Verb PP$

$VP \rightarrow VP PP$

$PP \rightarrow Preposition NP$

Chomsky Normal Form

$S \rightarrow NP VP$

$S \rightarrow X1 VP$

$X1 \rightarrow Aux NP$

$S \rightarrow book | include | prefer$

$S \rightarrow Verb NP$

$S \rightarrow X2 PP$

$S \rightarrow Verb PP$

$S \rightarrow VP PP$

$NP \rightarrow I | she | me$

$NP \rightarrow TWA | Houston$

$NP \rightarrow Det Nominal$

$Nominal \rightarrow book | flight | meal | money$

$Nominal \rightarrow Nominal Noun$

$Nominal \rightarrow Nominal PP$

$VP \rightarrow book | include | prefer$

$VP \rightarrow Verb NP$

$VP \rightarrow X2 PP$

$X2 \rightarrow Verb NP$

$VP \rightarrow Verb PP$

$VP \rightarrow VP PP$

$PP \rightarrow Preposition NP$

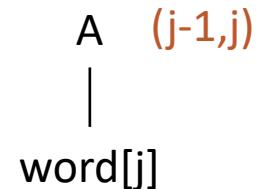
Dynamic Programming

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

Base case

Rule: $A \rightarrow \text{word}[j]$

A should be in $\text{table}[j-1,j]$



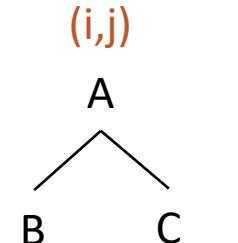
Recursion

Rule: $A \rightarrow B C$

If you find a k such that

B is in $\text{table}[i,k]$, and

C is in $\text{table}[k,j]$, then A should be in $\text{table}[i,j]$



$(i,k) \quad (k,j)$

Dynamic Programming

$$S \in \text{table}[0, n]$$

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

Base case

Rule: $A \rightarrow \text{word}[j] \leftarrow A \in \text{table}[\underline{j-1}, \underline{j}]$

A should be in $\text{table}[\underline{j-1}, \underline{j}]$

$\begin{array}{c} A \\ \top \\ \text{word}[j] \end{array}$

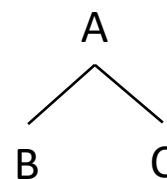
Recursion

Rule: $A \rightarrow B C$

If you find a k such that

B is in $\text{table}[i,k]$, and

C is in $\text{table}[k,j]$, then A should be in $\text{table}[i,j]$



(i,j)
 $(i,k) \quad (k,j)$

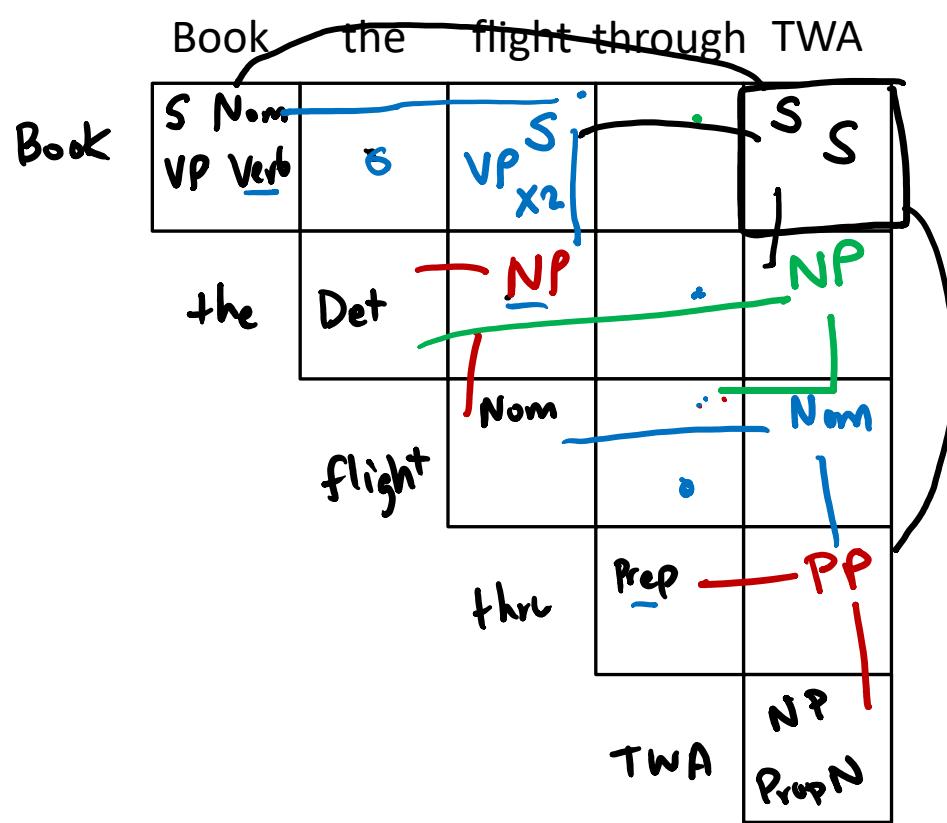
CKY Algorithm

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

Book the flight through TWA

CKY Algorithm

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



CKY Algorithm

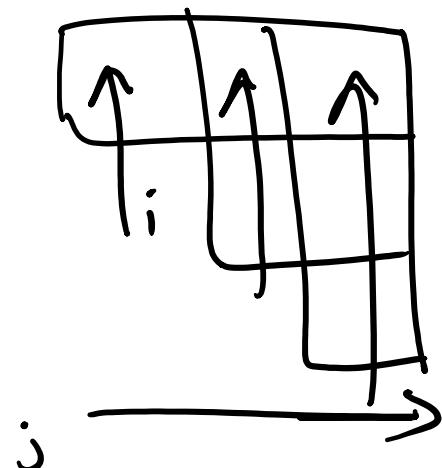
```
function CKY-PARSE(words, grammar) returns table
    for j  $\leftarrow$  from 1 to LENGTH(words) do
        for all  $\{A \mid A \rightarrow words[j] \in grammar\}$ 
            table[j - 1, j]  $\leftarrow$  table[j - 1, j]  $\cup$  A
    for i  $\leftarrow$  from j - 2 downto 0 do
        for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

CKY Algorithm

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

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

|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

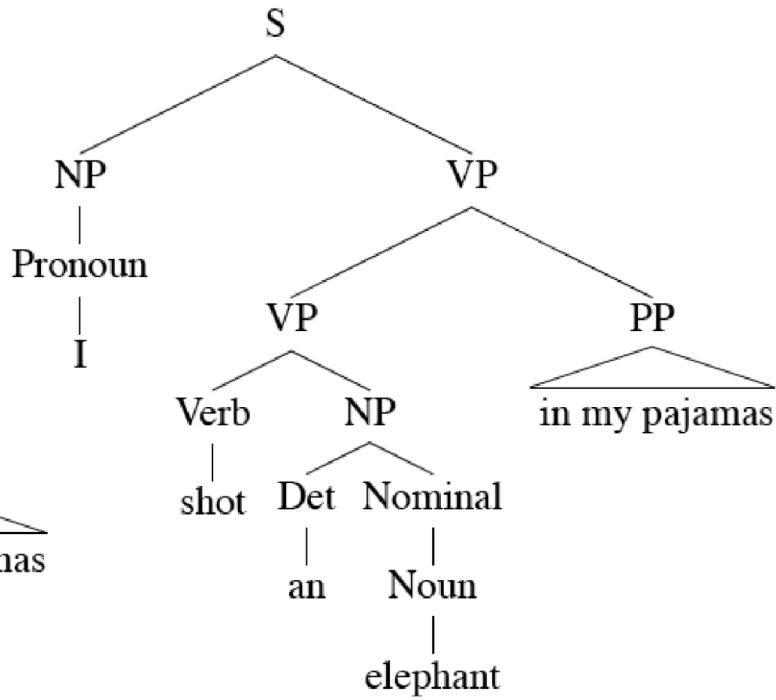
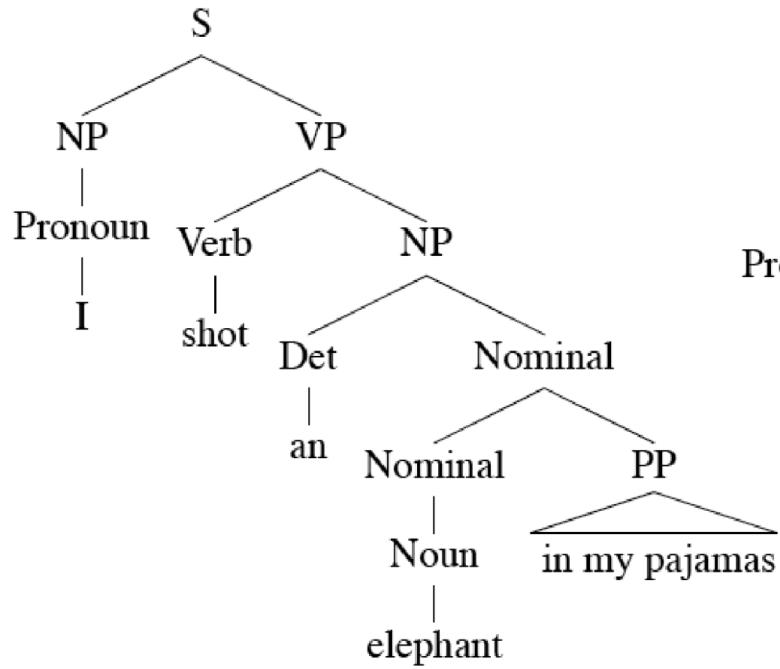
Parsing: CKY Algorithm

Extensions: Probabilistic and Lexicalized

Dependency Parsing

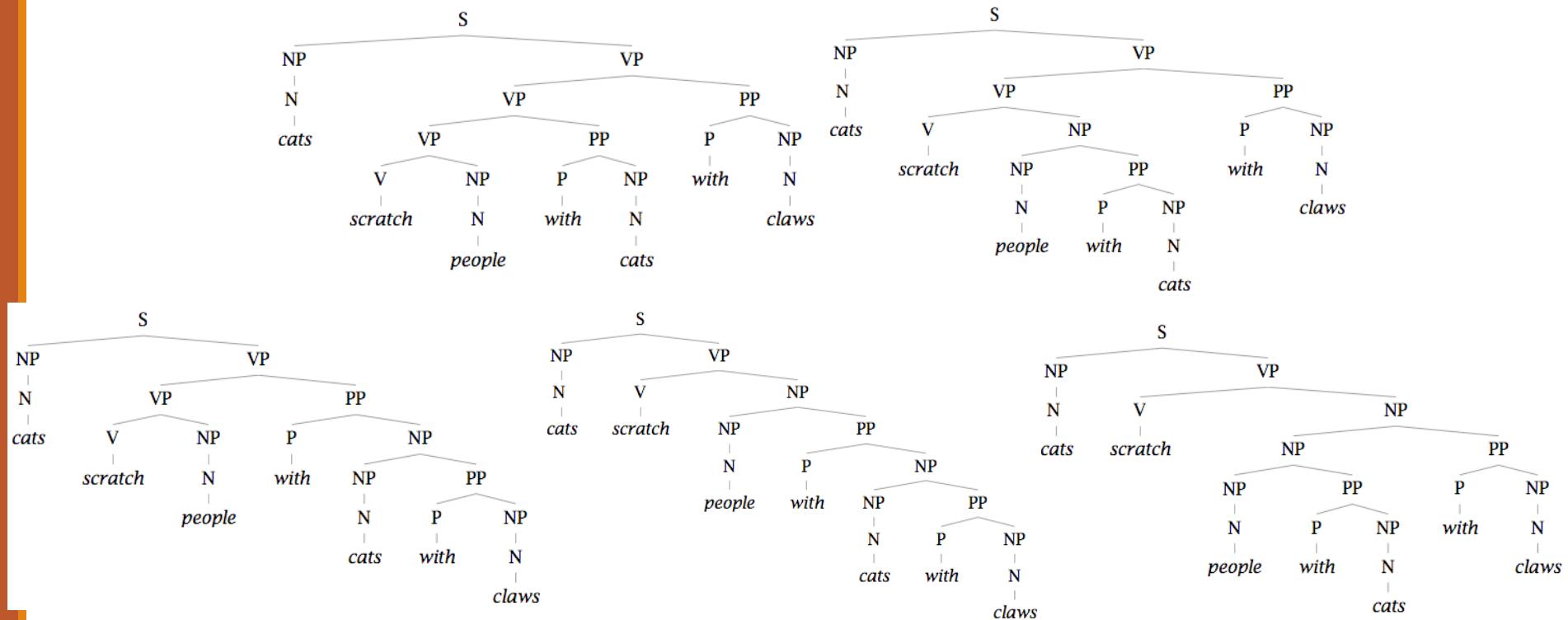
Ambiguity: Which parse?

I shot an elephant in my pajamas.



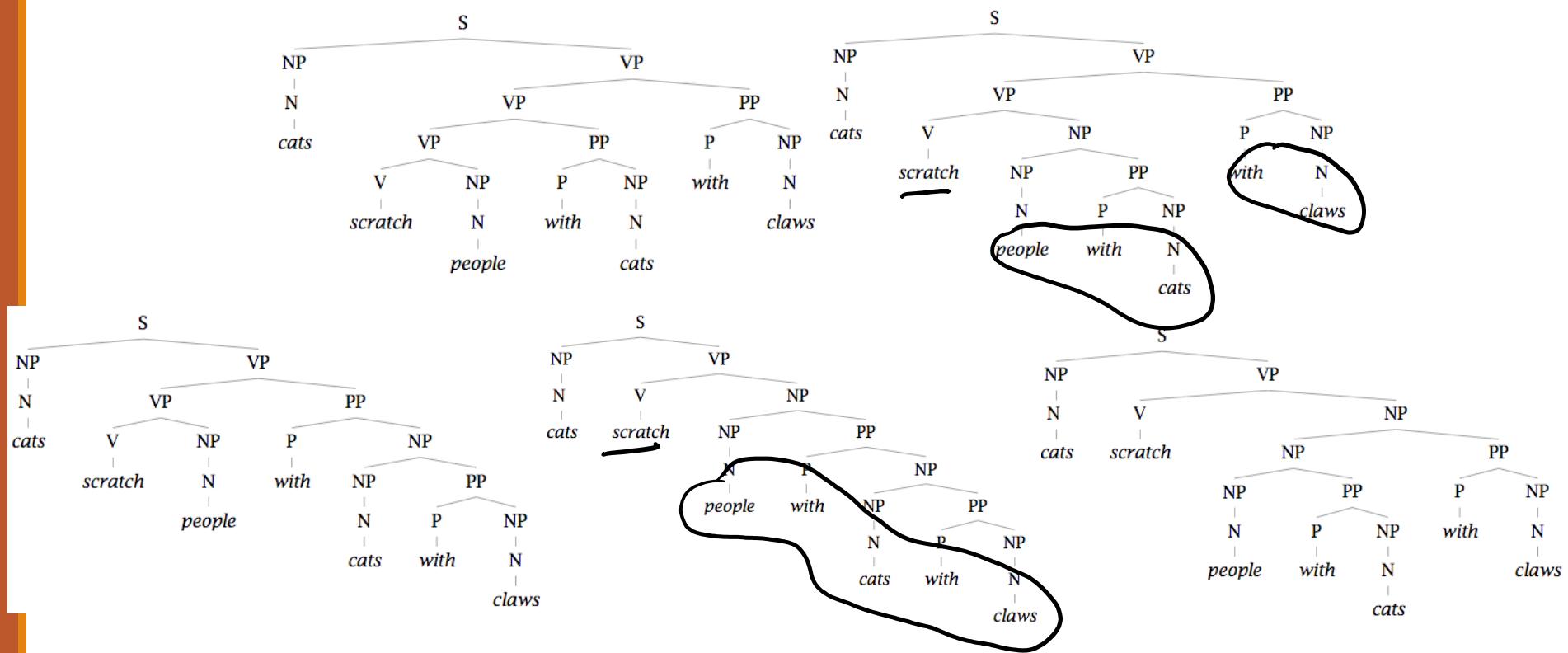
Finding the Best Parse Tree

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

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!

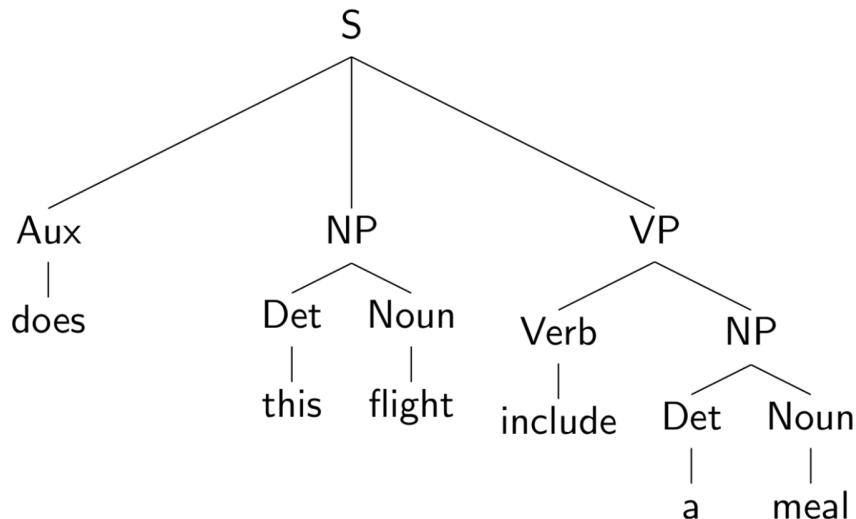
Rule: $A \rightarrow B C$

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

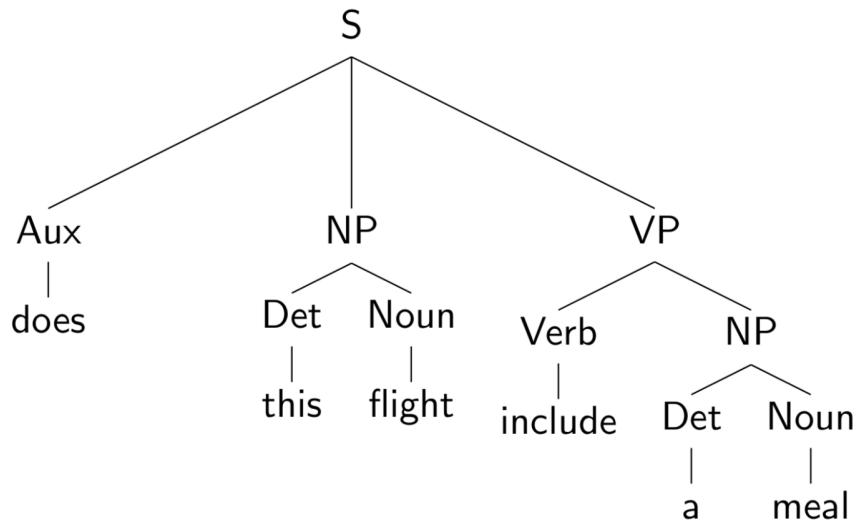
Compute the probability of a parse tree:

$$\prod_{A \rightarrow BC \in T} P(A \rightarrow BC | A)$$

Example of a PCFG



Example of a PCFG



$P(Aux \mid NP \mid VP \mid S)$

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

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

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

⋮

Estimating the probabilities

Estimating the probabilities

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

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!

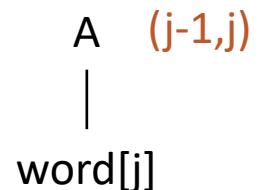
Probabilistic CKY Algorithm

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

Base case

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

$$T[j-1,j,A] = P(word[j] \mid A)$$

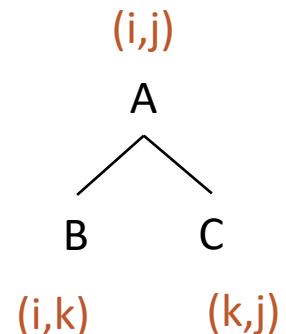


Recursion

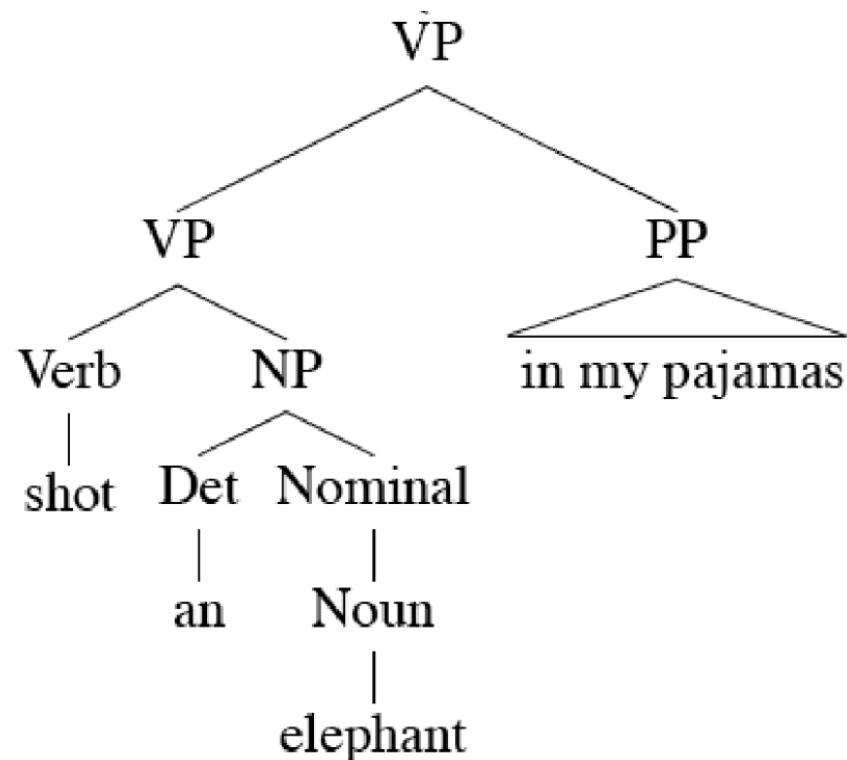
Rule: $P(A \rightarrow B C)$

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

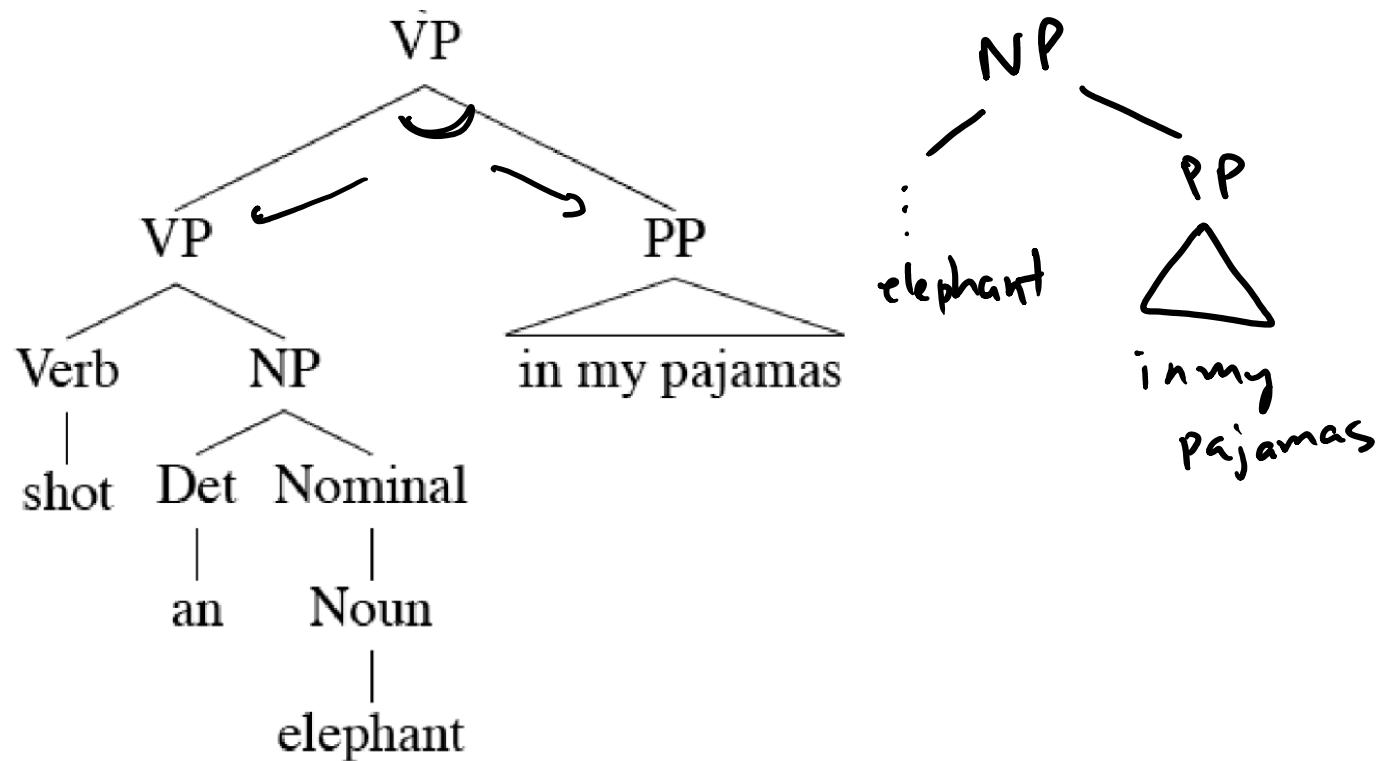
$$T[i,j,A] = \max_k P(B C \mid A) T[i,k,B] T[k,j,C]$$



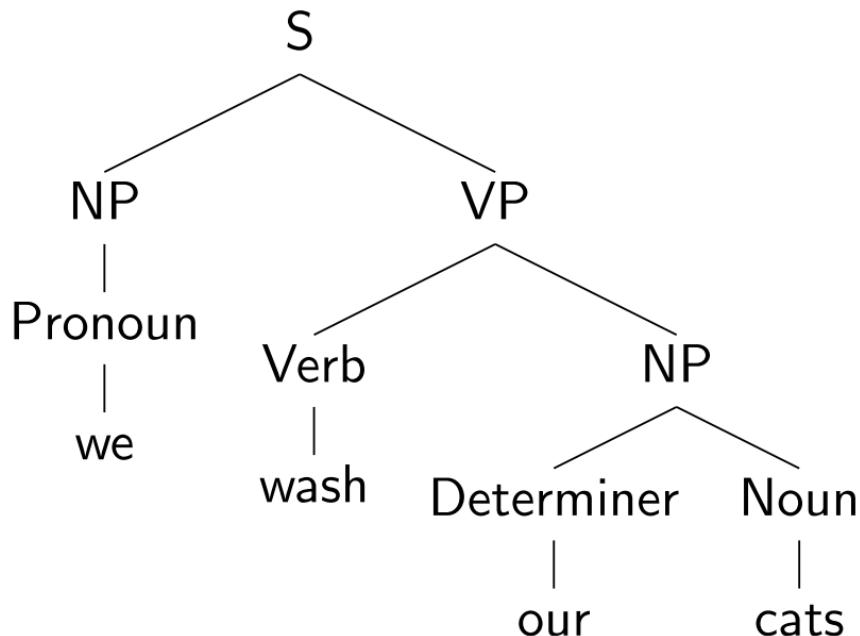
Lexicalized PCFGs



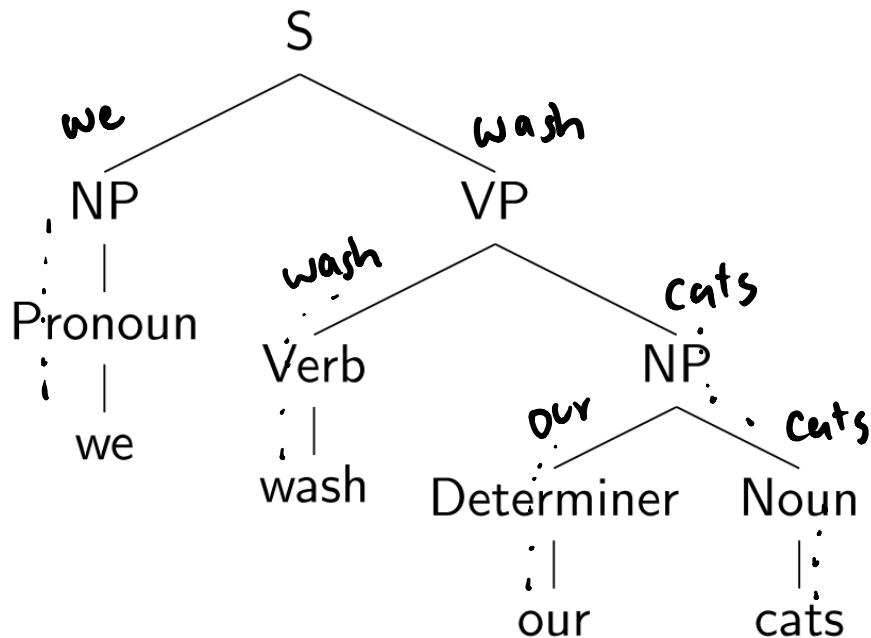
Lexicalized PCFGs



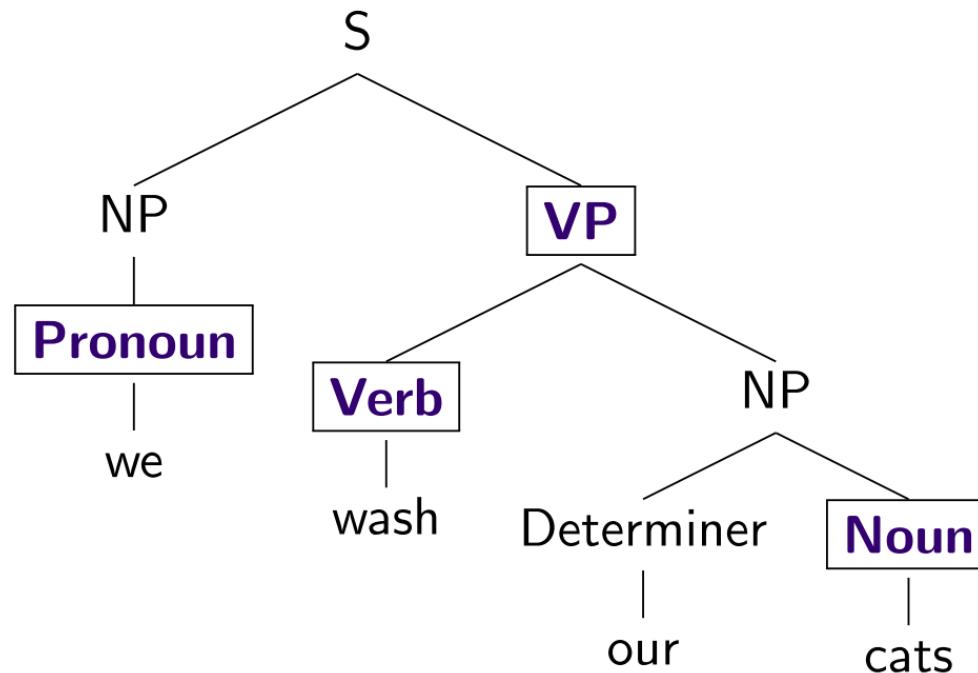
Lexicalizing a CFG



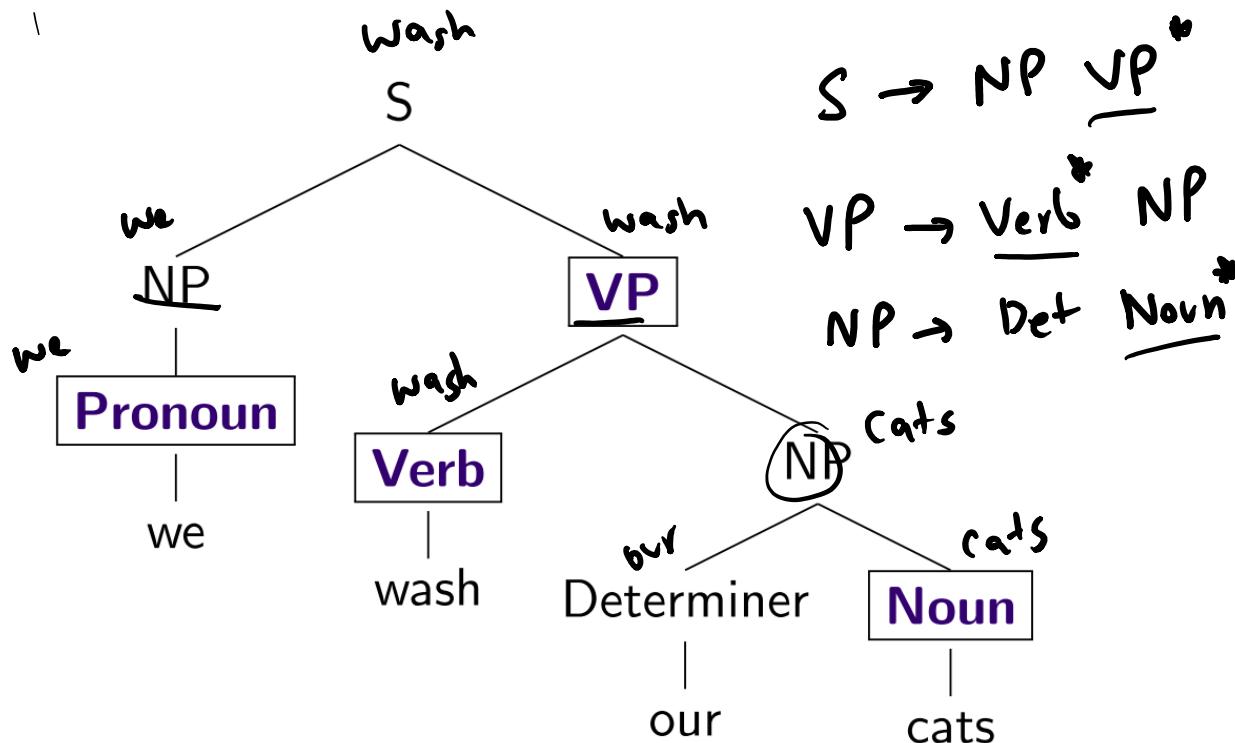
Lexicalizing a CFG



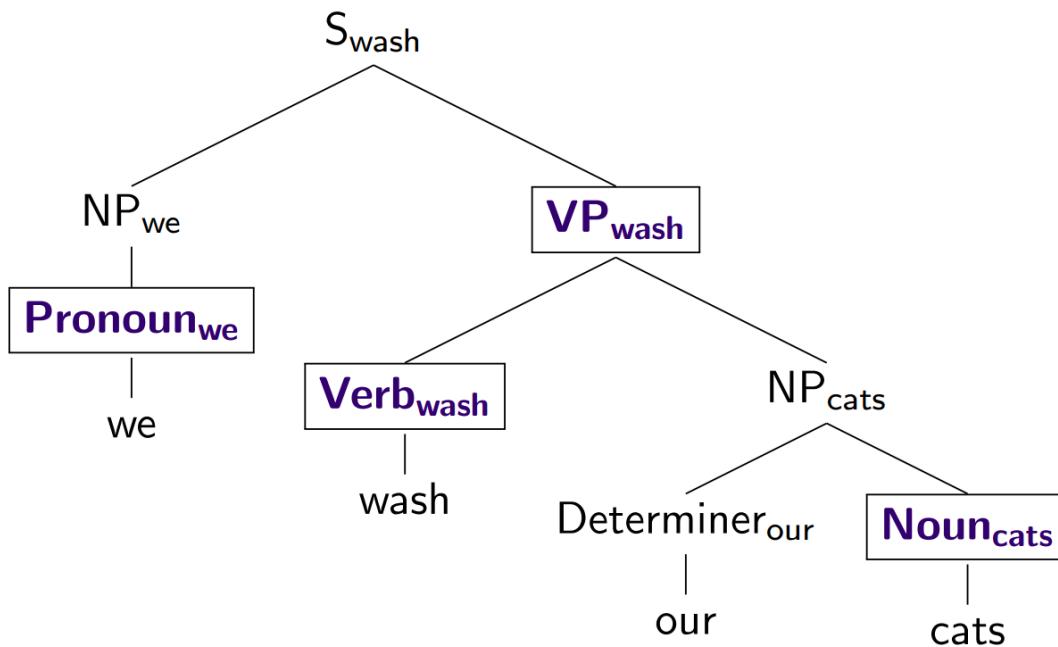
Lexicalizing a CFG



Lexicalizing a CFG



Lexicalizing a CFG



Outline

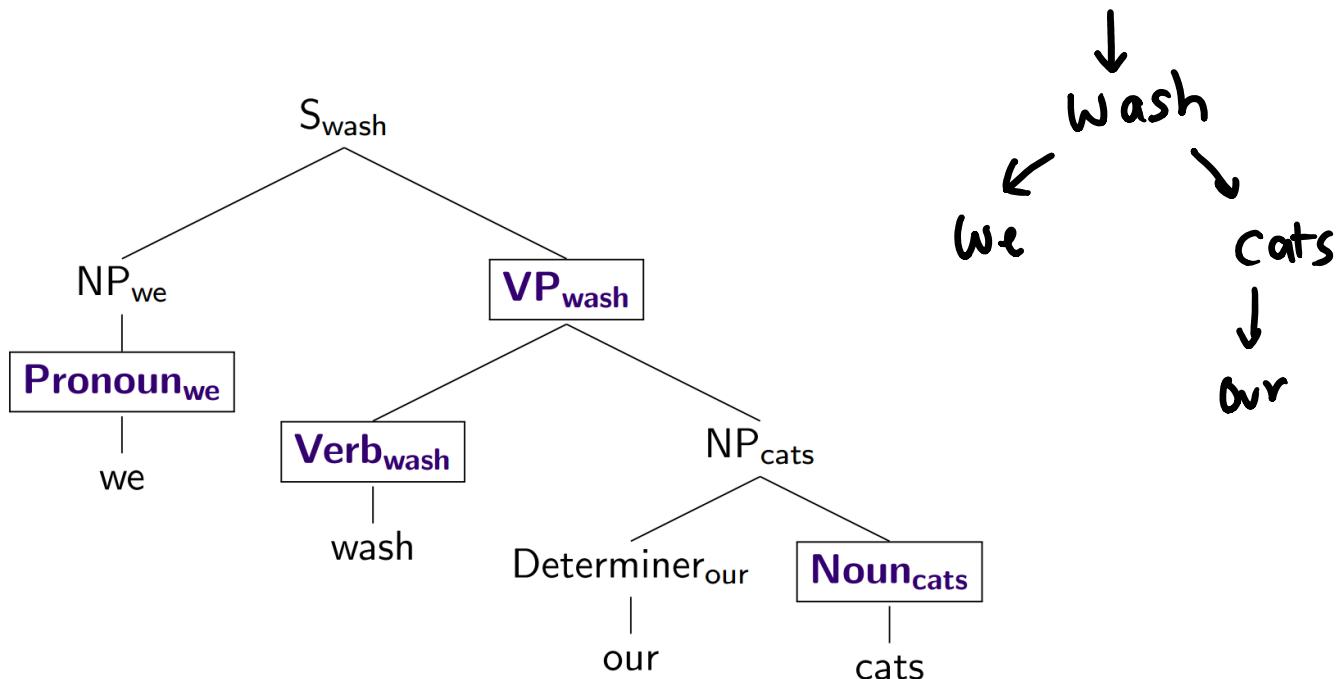
Parsing: CKY Algorithm

Extensions: Probabilistic and Lexicalized

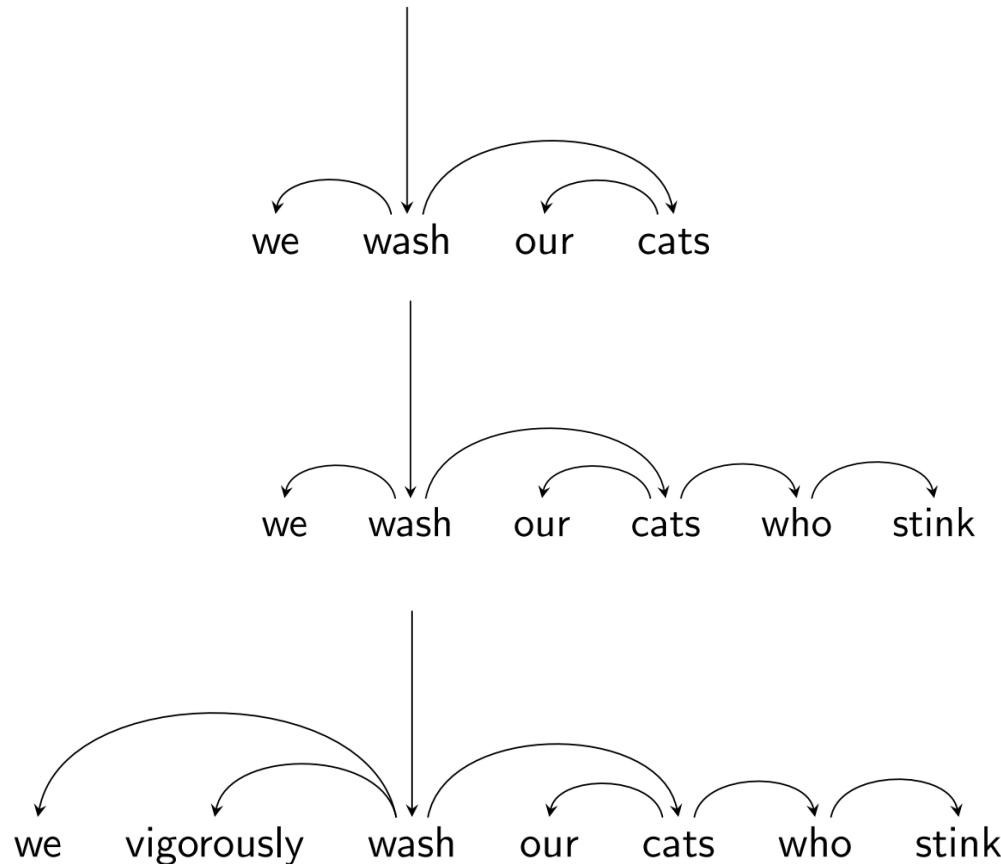
Dependency Parsing

Dependencies

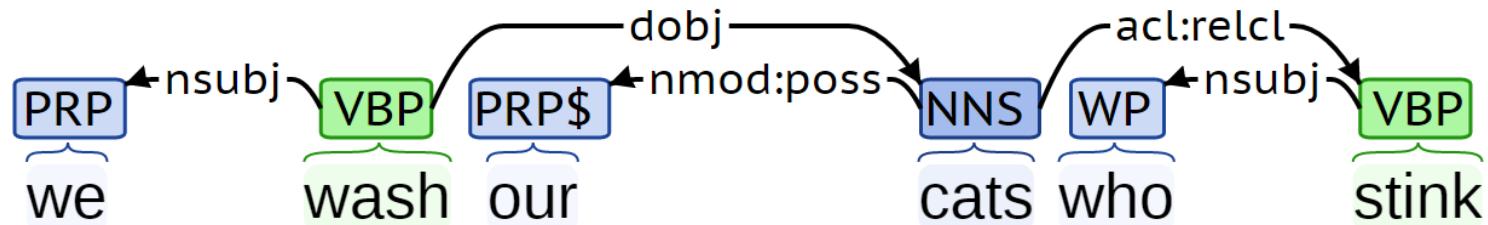
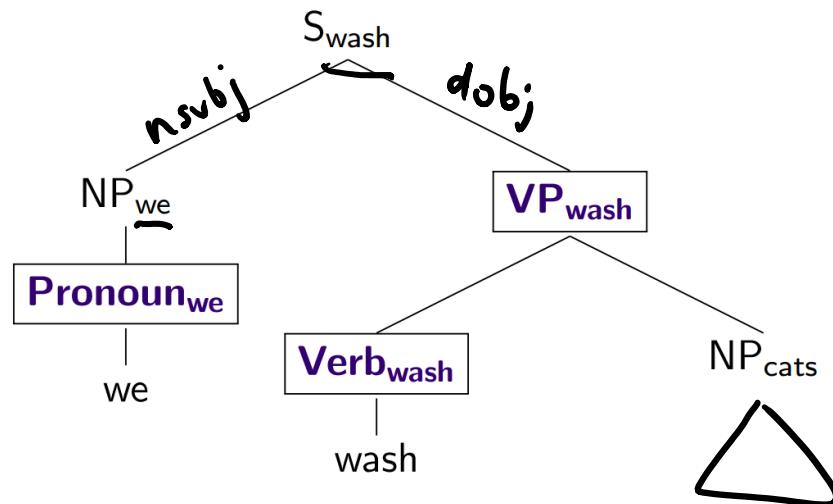
Represent only the syntactic dependencies...



Nested Structure = Subtrees



Dependency Labels



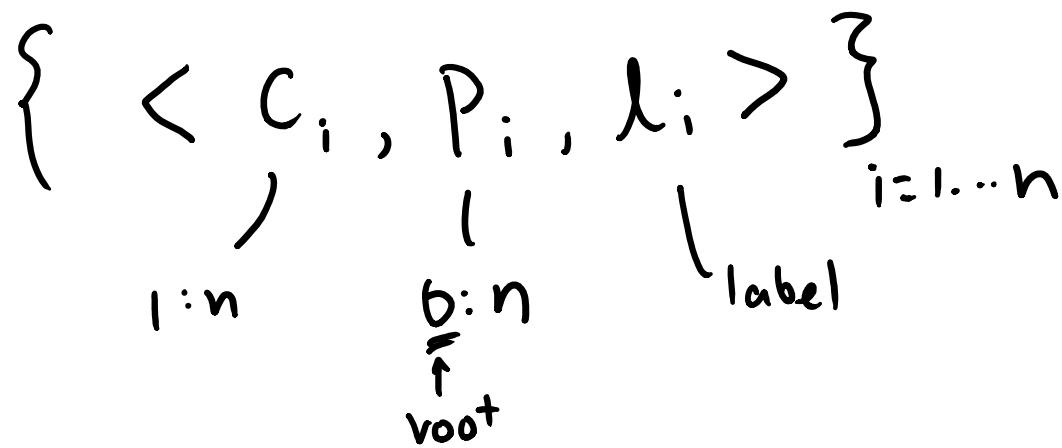
Dependency Labels

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

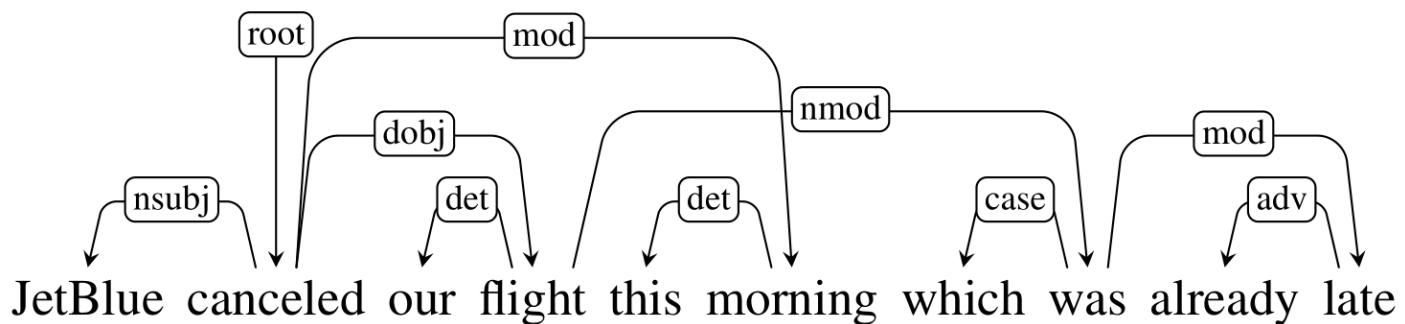
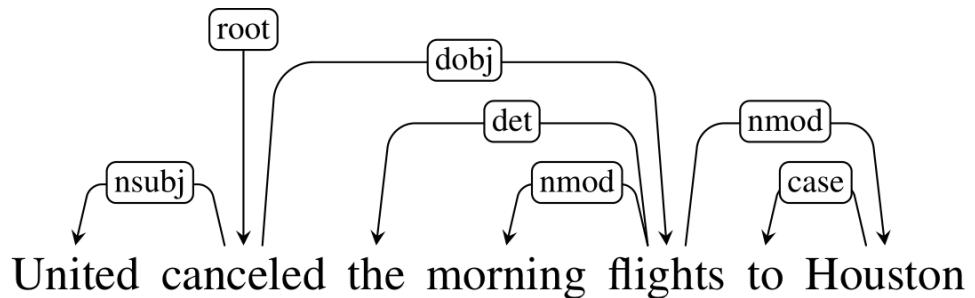
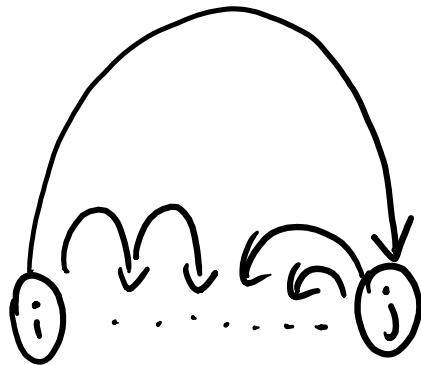
Other Notable Relations	Description
CONJ	Conjunct
CC	Coordinating conjunction

Dependency Trees

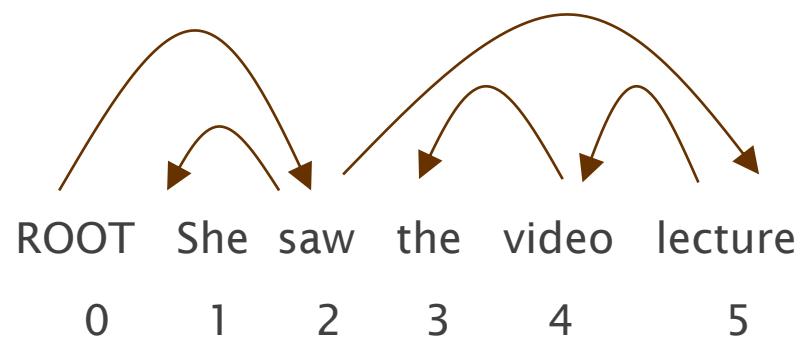
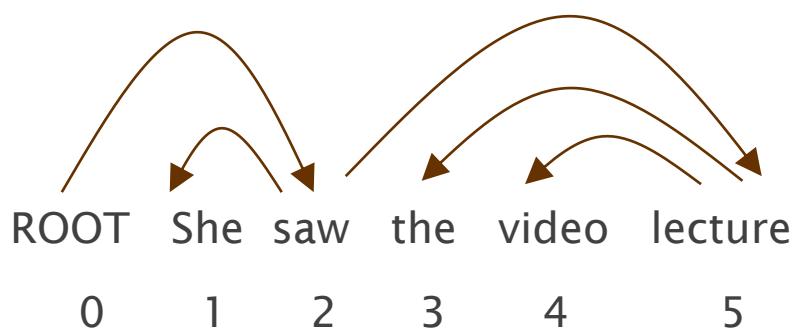


- single headed $\exists i$ only one $p_i = 0$
- connected - acyclic
- Projective vs non-projective

Projective vs Non-projective



Evaluating Dependency Parses



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

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

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

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

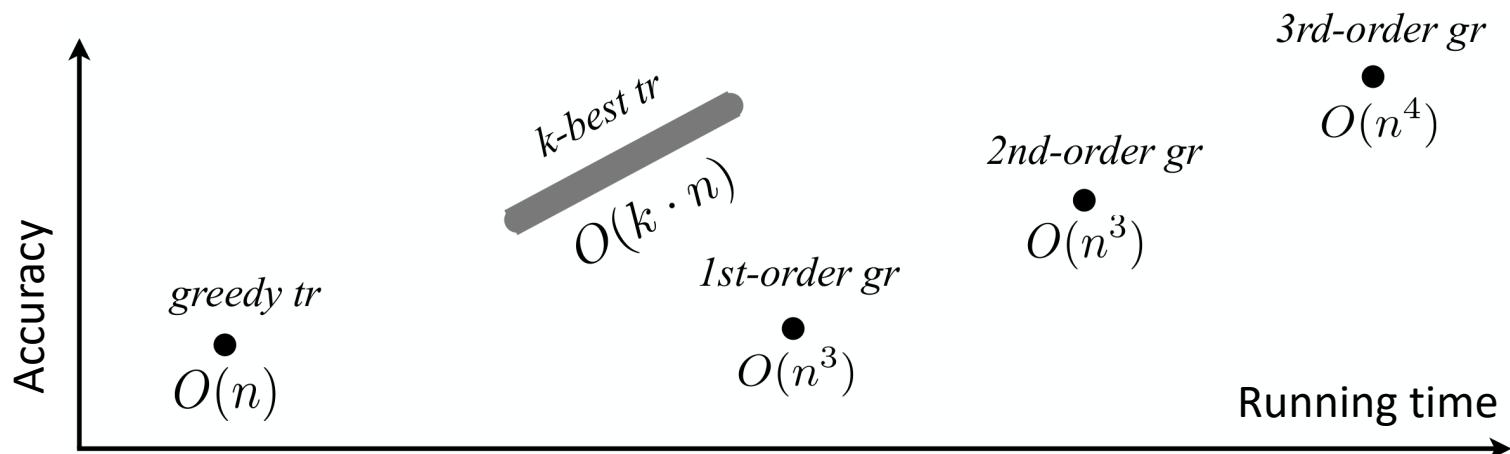
Parsing Algorithms

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

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

(
1st order / fully factored
 $\sum_i \theta_i \phi(e_i | c_i, p_i, l_i)$
;

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	

↷ ↑ → action

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 ↙	

$$\Theta \cdot \phi(\text{stack}, \text{wlist}, \text{action})$$