Natural Language Processing

AIMA Ch 23

[Some slides were adapted from slides by James H. Martin at U Colorado Boulder, by Albert Gatt at University of Malta, and by Eric Xing of CMU]

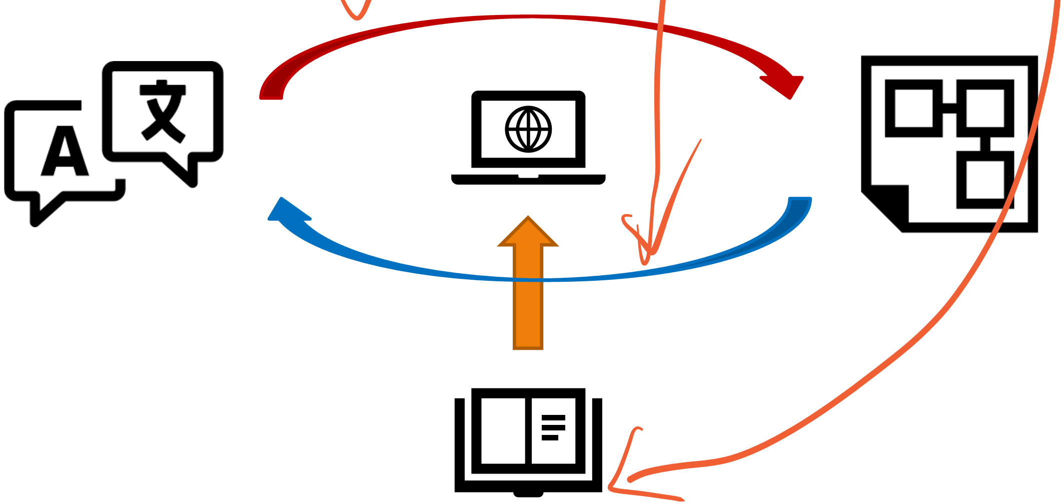
Additional Reference

- [SLP] Speech and Language Processing, Daniel Jurafsky and James H. Martin
  - 2nd edition, 2008
  - 3rd edition, Sept. 2021

- Parsing
  - [AIMA] Ch 23
  - [SLP] Ch 12, 13, 14

What is NLP?

- Automating the analysis, generation, and acquisition of human ("natural") language



NLP Applications

- ChatBot
  - Question answering, conversation
- Machine translation
- Information extraction
  - From financial and law documents, e-commerce websites, etc.
- Chinese IME 输入法
- Grammatical checker
- Essay scoring
- News generation

Levels of NLP Research

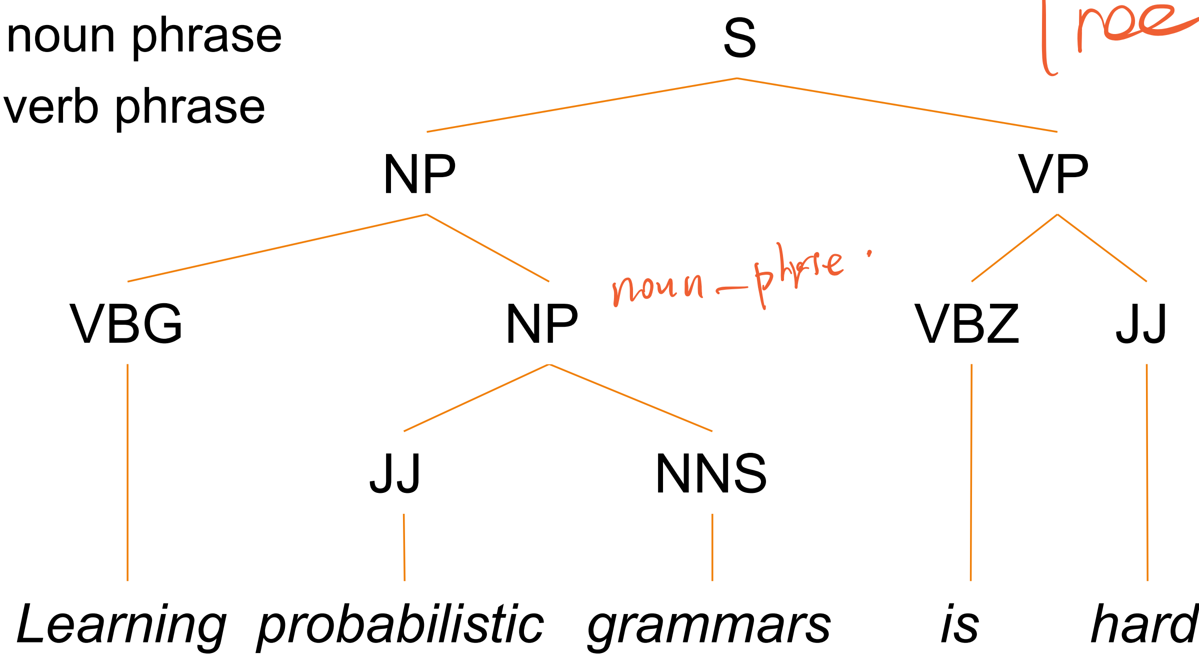
Phonetics and phonology	knowledge about linguistic sounds
Morphology	knowledge of the meaningful components of words
Syntax	knowledge of the structural relationships between words 句法
Lexical semantics 词	knowledge of word meaning
Compositional semantics 句	knowledge of the meaning of sentences
Pragmatics	knowledge of the relationship of meaning to the goals and intentions of the speaker
Discourse	knowledge about linguistic units larger than a single sentence

Constituency Parsing

Constituent parse tree

- Also called a phrase structure parse
- Each non-leaf node represents a phrase (i.e., constituent)
  - S: sentence
  - NP: noun phrase
  - VP: verb phrase
  - …

Tree



Grammars

- Grammar
  1. 成分. 2.
  - the set of constituents and the rules that govern how they combine
- Lots of different theories of grammar
- Context-free grammars (CFGs)
  - Also known as: Phrase structure grammars
  - One of the simplest and most basic grammar formalisms

Context-Free Grammars

- A context-free grammar has four components
  - A set Σ of terminals (words)
  - A set N of nonterminals (phrases)
  - A start symbol $S \in N$
  - A set R of production rules
    - Specifies how a nonterminal can produce a string of terminals and/or nonterminals

Non teremind

Example Grammar

Grammar Rules			Examples
			NP VP
S	$\rightarrow$	*NP VP*	I + want a morning flight
" a sentence is conposed of Noun phrase + verb phrase "			
NP	$\rightarrow$	*Pronoun* 代词	I
NP	$\rightarrow$ |	*Proper-Noun* 专有名词	Los Angeles
NP	$\rightarrow$ |	*Det Nominal* 冠词短语	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：除了

terminal

Example Grammar

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*

Sentence Generation

- A grammar can be used to generate a string
  - starting from a string containing only the start symbol S
  - recursively applying the rules to rewrite the string
  - until the string contains only terminals → only words.
- The generative process specifies the grammatical structure (parse tree) of the string

Example

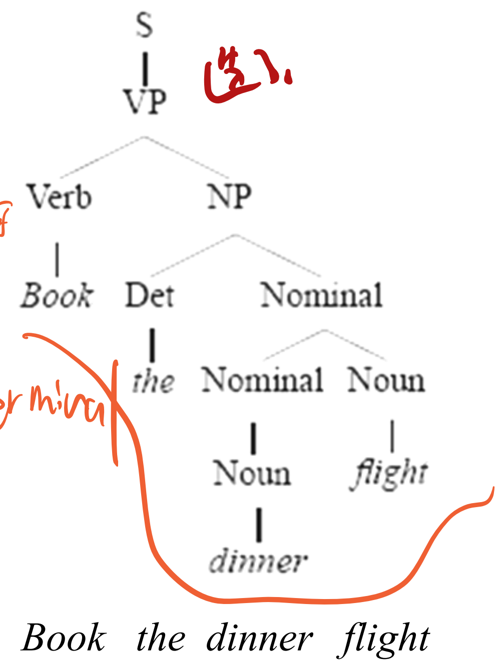
$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$NP \rightarrow Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow Verb\ NP\ NP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

......

iters'

verb → 动词

terminal



句法解析

Sentence Parsing

- Parsing is the process of taking a string and a grammar and returning one or more parse tree(s) for that string
  - If no parse tree can be found, then the string does not belong to the language
  - Parsing algorithms: CYK, Earley, etc.
    - To be introduced later

概率
Probabilistic Grammars

- Also called stochastic grammars
- Each rule is associated with a probability

$$\alpha \to \beta : P(\alpha \to \beta | \alpha)$$

every rule.

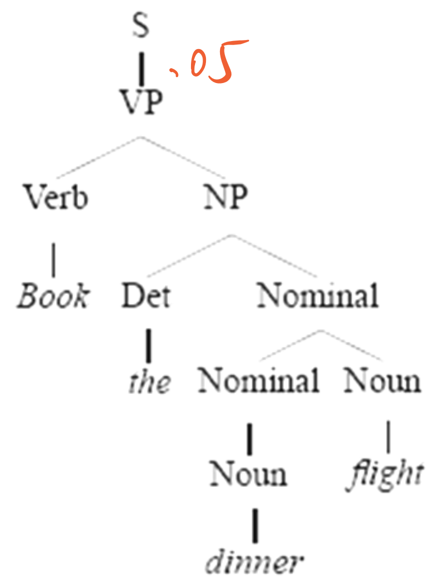
- The probability of a parse tree is the product of the probabilities of all the rules used in generating the parse tree

Example

Sum=1

S	$\rightarrow$	$NP\ VP$	[.80]
S	$\rightarrow$	$Aux\ NP\ VP$	[.15]
S	$\rightarrow$	VP	[.05]
NP	$\rightarrow$	$Pronoun$	[.35]
NP	$\rightarrow$	$Proper\text{-}Noun$	[.30]
NP	$\rightarrow$	$Det\ Nominal$	[.20]
NP	$\rightarrow$	$Nominal$	[.15]
$Nominal$	$\rightarrow$	$Noun$	[.75]
$Nominal$	$\rightarrow$	$Nominal\ Noun$	[.20]
$Nominal$	$\rightarrow$	$Nominal\ PP$	[.05]
VP	$\rightarrow$	$Verb$	[.35]
VP	$\rightarrow$	$Verb\ NP$	[.20]
VP	$\rightarrow$	$Verb\ NP\ PP$	[.10]
VP	$\rightarrow$	$Verb\ PP$	[.15]
VP	$\rightarrow$	$Verb\ NP\ NP$	[.05]
VP	$\rightarrow$	$VP\ PP$	[.15]
PP	$\rightarrow$	$Preposition\ NP$	[1.0]

……



Book the dinner flight

$P(T) = .05 \times .20 \times .20 \times .20 \times .75 \times .30 \times .60 \times .10 \times .40 = 2.2 \times 10^{-6}$

Ambiguity

歧义

- A sentence is ambiguous if it has more than one possible parse tree
  - …and hence more than one interpretation
- Examples
  - Astronomers saw stars with ears.

stars ?

saw ?

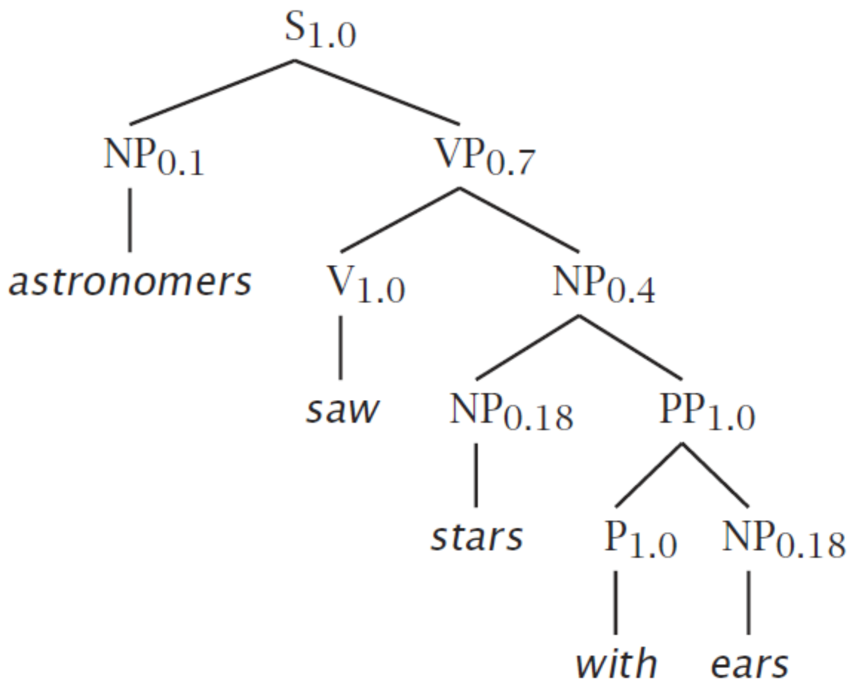
Example

all sum to 1

S → NP VP	1.0		NP → NP PP	0.4
PP → P NP	1.0		NP → *astronomers*	0.1
VP → V NP	0.7		NP → *ears*	0.18
VP → VP PP	0.3		NP → *saw*	0.04
P → *with*	1.0		NP → *stars*	0.18
V → *saw*	1.0		NP → *telescopes*	0.1

Example

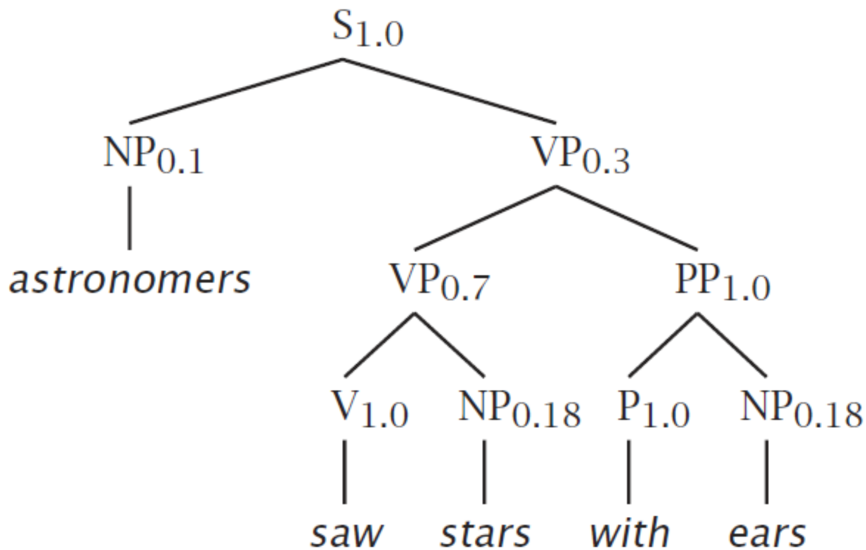
t_1:



$$
\begin{aligned}
P(t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0009072
\end{aligned}
$$

Example

t_2:



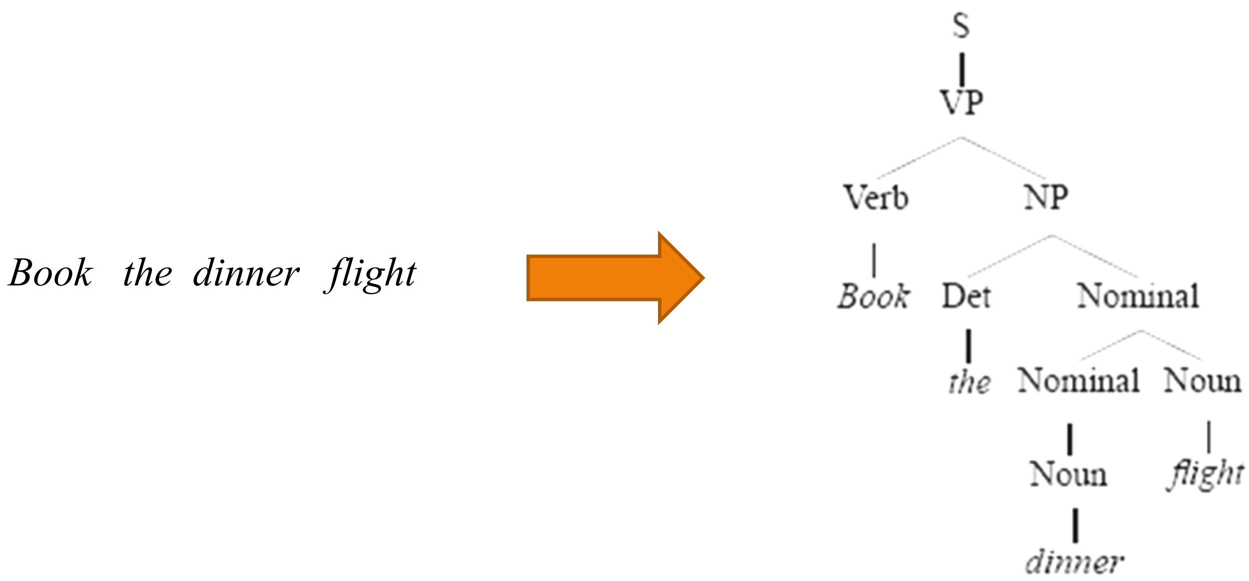
$$
\begin{aligned}
P(t_2) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0006804
\end{aligned}
$$

Just
it

Parsing Algorithm

Parsing

- Parsing with CFGs is the task of assigning proper parse trees to input strings



Parsing

- A brute-force approach
  - Enumerate all parse trees consistent with the input string
- Problem
  - Number of binary trees with *n* leaves is the Catalan number C_{n-1}
  - (Exponential growth)

Cocke–Younger–Kasami Algorithm (CYK)

- A bottom-up dynamic programming algorithm
- Applies to CFG in Chomsky Normal Form (CNF)
  - Only two types of production rules

A → B C

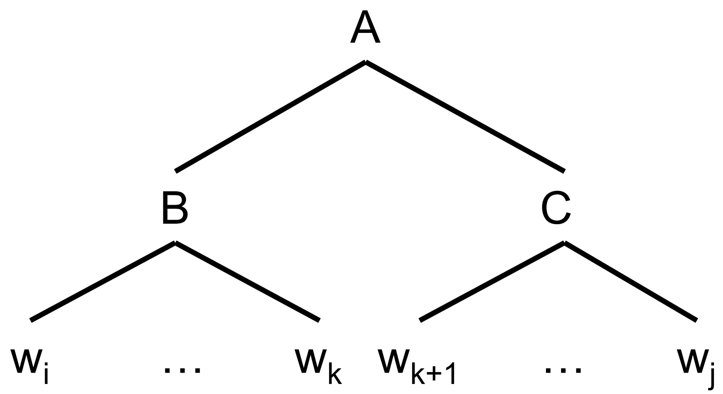
A → *w*

Non → Non → Non

Non → terminal

Parsing

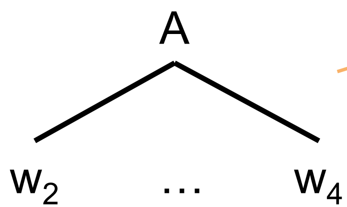
- Dynamic programming
  - Divide the problem into many sub-problems
    - Sub-problem: parsing the substring between positions *i* and *j*
  - Solutions to smaller sub-problems are reused in solving larger sub-problems



CYK

- Build a table so that a non-terminal A spanning from i to j in the input is placed in cell [i-1, j] in the table.

5 words



“2到4 word 存在 parse tree, 根为A”

	1	2	3	4	5
0					
1				A	
2					
3					
4					

target place

- So a non-terminal spanning an entire string will sit in cell [0, n]
  - Hopefully an *S*

want to find S

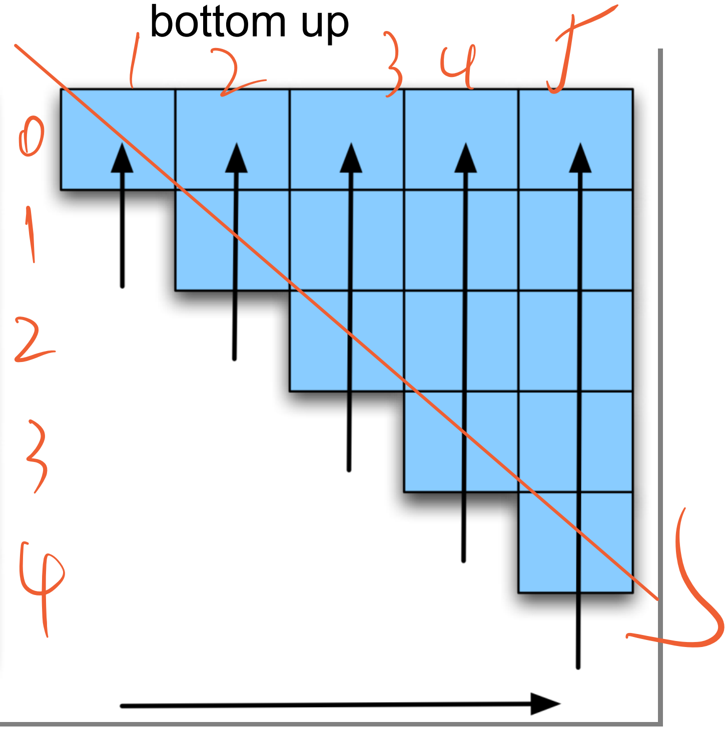
here

Example

A completed table for input "Book the flight through Houston"

Book	the	flight	through	Houston
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

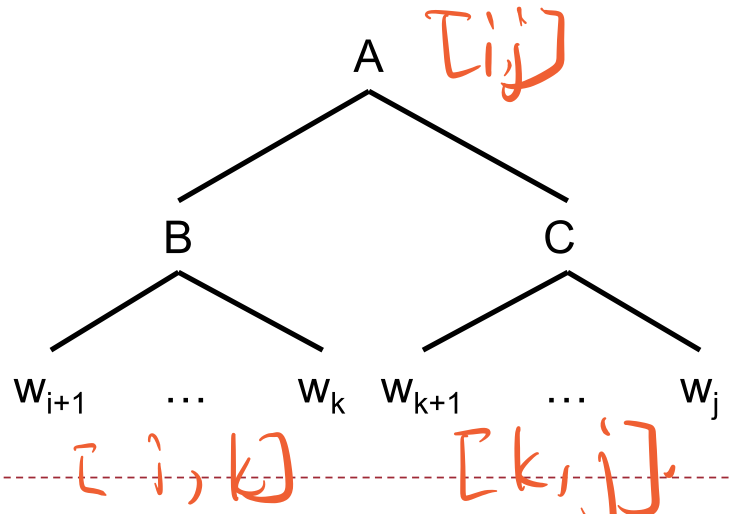
We fill the table from bottom up



CYK (dp)

对角线

- Base case:
  - A is in cell [i-1,i] iff. there exists a rule $A \rightarrow w_i$
- Recursion:
  - A is in cell [i,j] iff. for some rule $A \rightarrow B\ C$ there is a B in cell [i,k] and a C in cell [k,j] for some k.



CYK Algorithm

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

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

$table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$

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

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

$$table[i,j] \leftarrow table[i,j] \cup \{A \mid A \rightarrow BC \in grammar, B \in table[i,k], C \in table[k,j]\}$$

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0					
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det				
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det 1	3			
1		N 2			
2					
3					
4					

CYK

"the | fight"

divide k : known

- *The flight* includes a meal.

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2					
3					
4					

X

(0,1) (1,3)

The flights include

the flight | include

CYK

- *The flight includes a meal.*

(0,2) (2,3)
NP V

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP	X		
1		N	X		
2			V		
3					
4					

NO → NtV.

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	
4					N

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	NP
4					N

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		VP
3				Det	NP
4					N

"includes

a meal"

(2,3)(3,5) ✓

(2,4)(4,5) ✗

CYK

- *The flight includes a meal.*

有s!

	1	2	3	4	5
0	Det	NP			S
1		N			
2			V		VP
3				Det	NP
4					N

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

CYK Parsing

- Is that really a parser?
  - We want a parse tree, not a yes/no answer
- Simple changes know the tree
  - Add back-pointers so that each state knows where it came from.
  - After filling the table, recursively retrieve the constituents from the top (i.e., the start symbol) down

CYK

- *The flight includes a meal.*

back pointers

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			S
1		N			
2			V		VP
3				Det	NP
4					N

直角法!

Prob ed CYK

Probabilistic Parsing

- What if we have a PCFG, and we want to find the parse tree of an input string with the highest probability?
- Still run CYK, but:
  - In cell [i-1,j] of the table, associate each nonterminal A with the probability of the best parse tree rooted at A covering substring from i to j
  - The probabilities can be computed with a bottom-up recursive formula during CYK steps

Regular Grammar

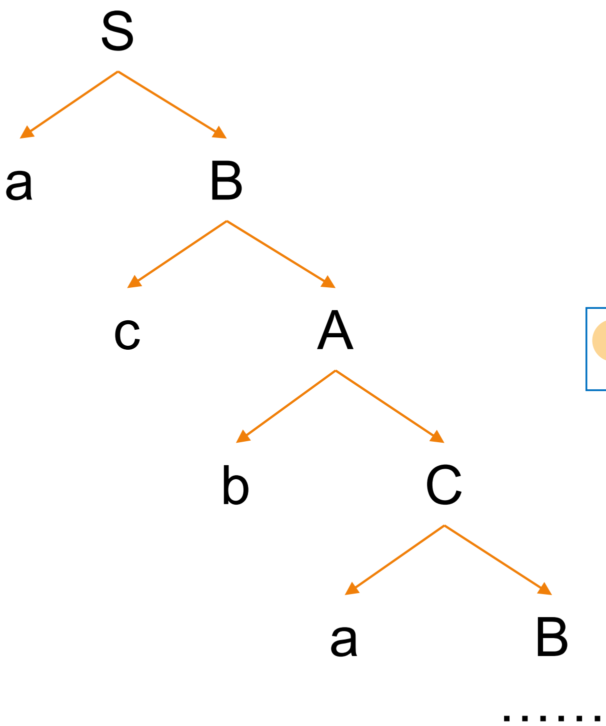
正则

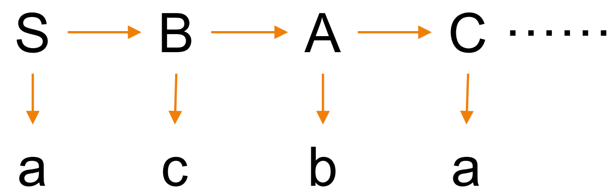
Regular Grammars

non → ter

non → non + ter

- Production rules are of the form $A \rightarrow aB$ or $A \rightarrow a$





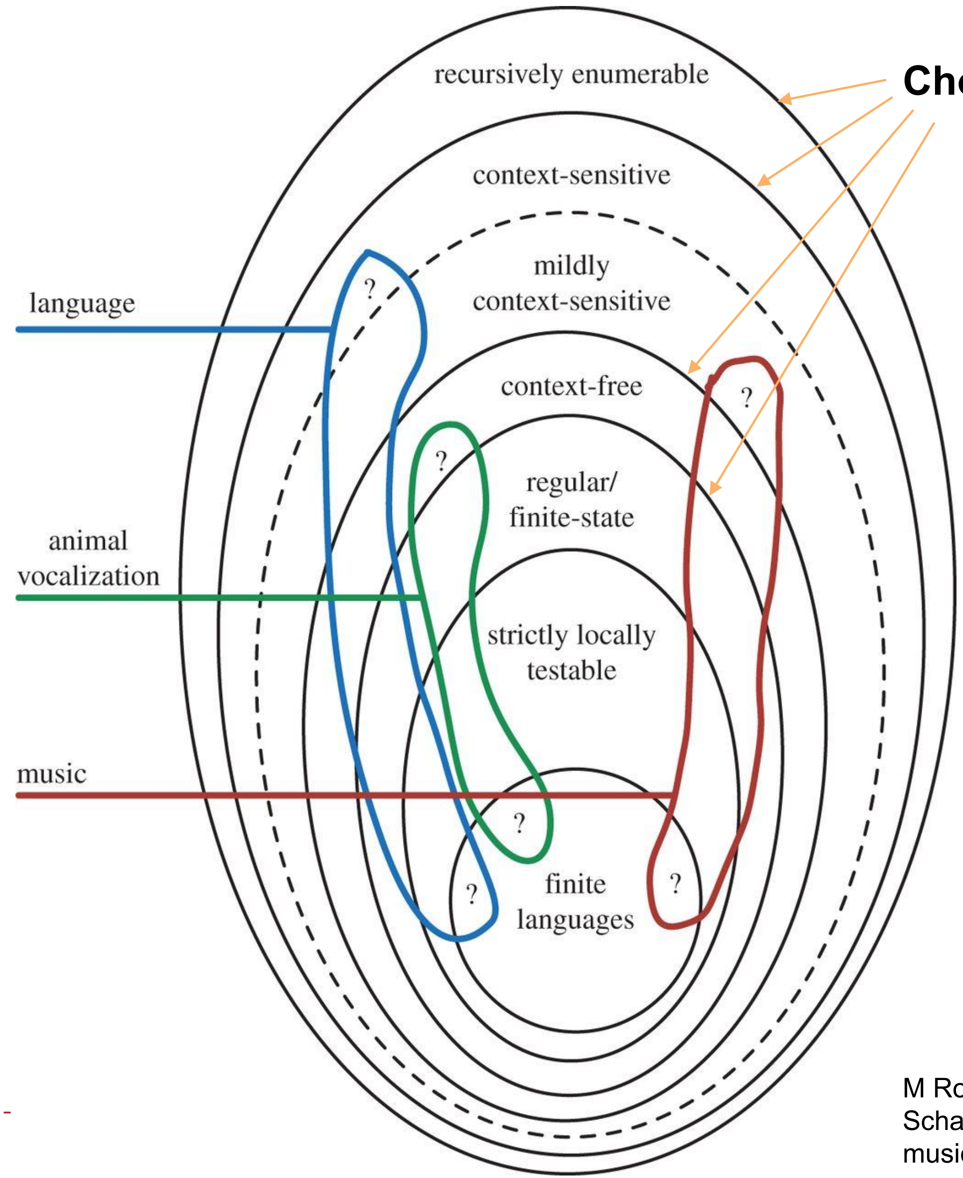
$\subset$

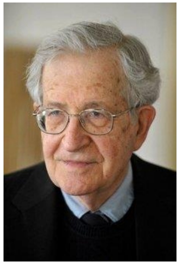
HMM = Probabilistic RG $\subset$ PCFG

Viterbi ~ Probabilistic CYK

(dp)

Chomsky Hierarchy





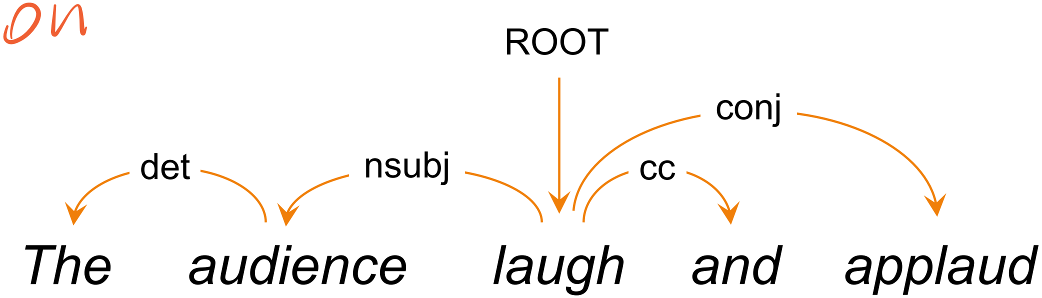
M Rohrmeier, W Zuidema, G Wiggins, C Scharff. Principles of structure building in music, language and animal song.

Dependency Parsing

Dependency Parse

- A dependency parse is a directed tree where
  - the nodes are the words in a sentence
    - ROOT: a special root node
  - The links between the words represent their dependency relations
    - Typically drawn as a directed arc from head to dependent
    - Dependency arcs may be typed (labeled)

"Relation"



Dependency Types

Argument Dependencies	Description
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

Dependency Parsing

- Advantages
  - Deals well with free word order languages where the constituent structure is quite fluid
    - Ex: Czech, Turkish
  - Dependency parses of sentences having the same meaning are more similar across languages than constituency parses
  - Dependency structure often captures the syntactic relations needed by later applications
  - Parsing can be faster than CFG-bases parsers

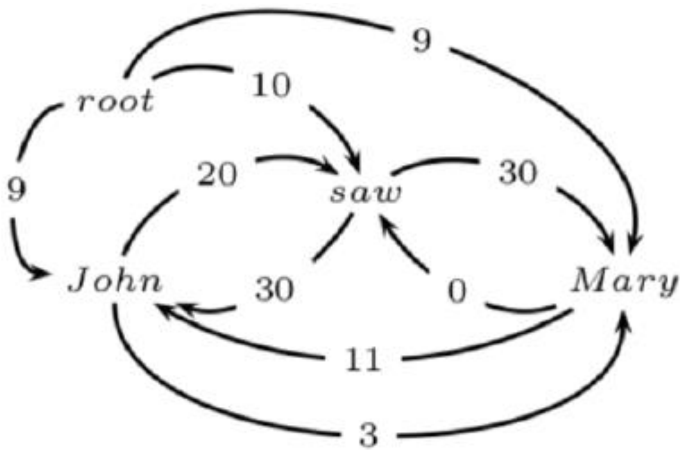
Dependency Parsing

- Parsing
  - Taking a string and a grammar and returning one or more parse tree(s) for that string
- Probabilistic parsing
  - Find the highest-scoring parse tree

- Several approaches to dependency parsing
  - Next: a brief intro of graph-based parsing

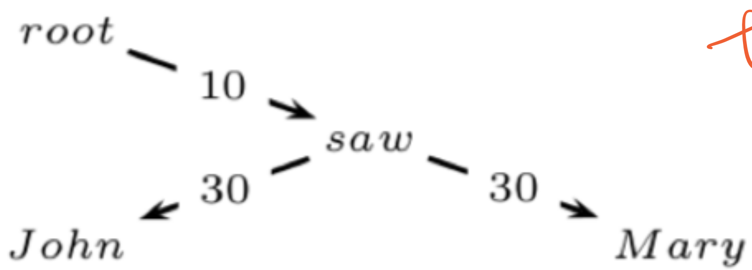
Graph-based parsing

- Each arc has a non-negative score.
  - An arc score is often computed from features of the two words and the context.



Graph-based parsing

- The tree score is the product of arc scores.

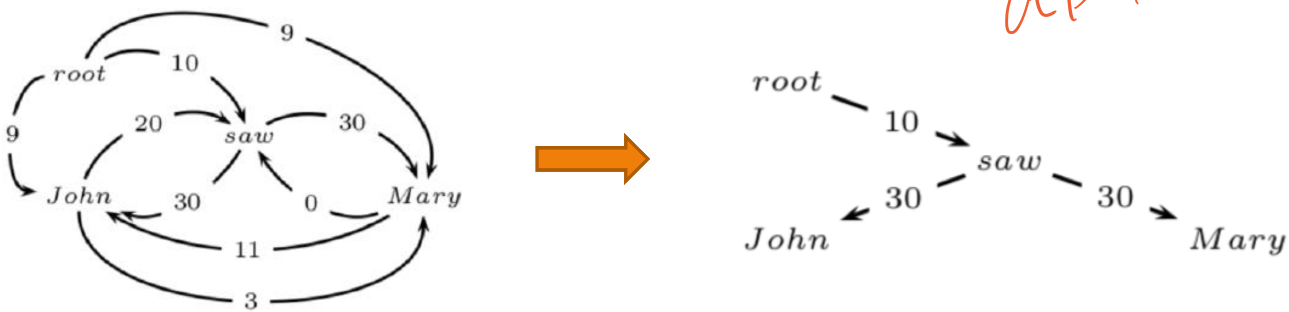


Max spanning tree

Graph-based parsing

- Parsing: find the highest-scoring parse tree
  - = max spanning directed tree (arborescence)
  - Solvable by greedy algorithm (Chu-Liu-Edmonds) and dynamic programming (Eisner)

dp / greedy



Dependency Grammar vs. CFG

DG vs. CFG

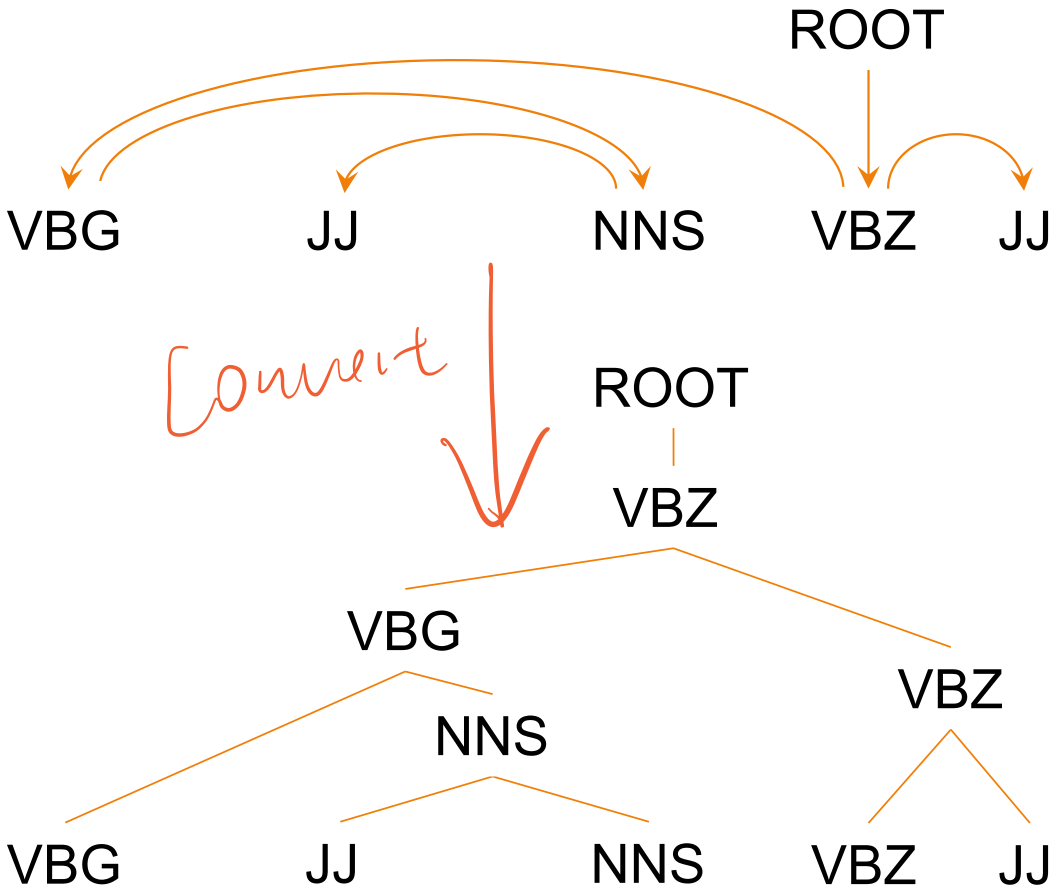
- Dependency grammars are a subclass of CFGs

a → b		A → AB
a → c		A → AC
d ← a	⇒	A → DA
e ← a		A → EA
		A → a
		B → b
		C → c
		D → d
		E → e

left, right

not changed

DG vs. CFG

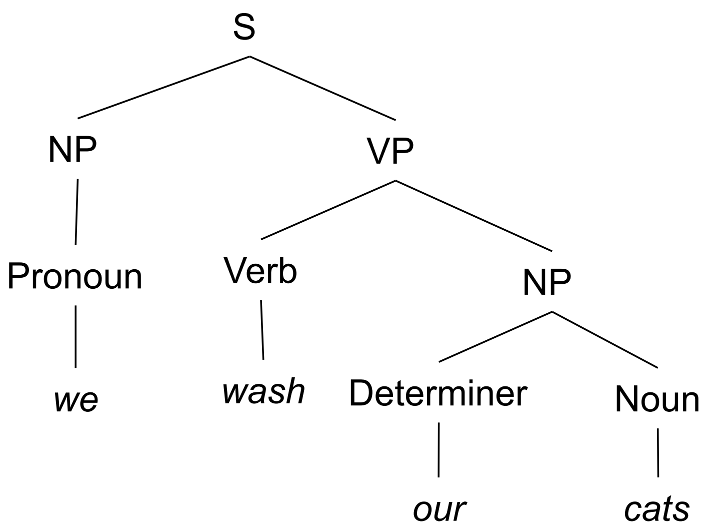

ROOT
VBG
JJ
NNS
VBZ
JJ
Convert
ROOT
VBZ
VBG
VBZ
NNS
VBG
JJ
NNS
VBZ
JJ

Conversion

CFG → DG

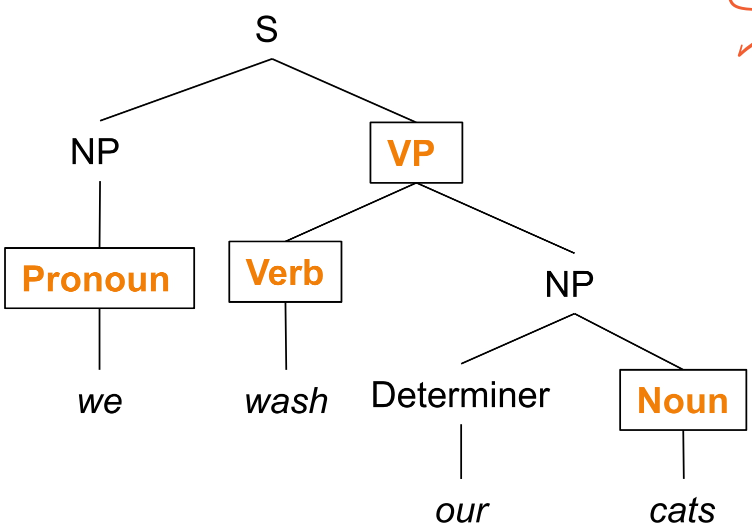
- From a constituent tree to a dependency tree
  - Constituent tree

find the important one in 2 children



Conversion

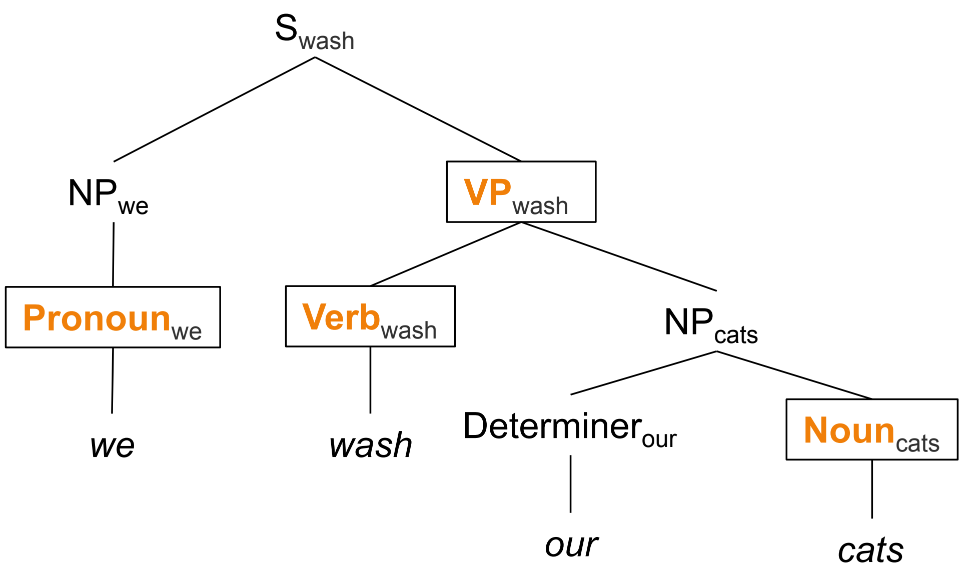
- From a constituent tree to a dependency tree
  - Constituent tree with heads



head → dependent

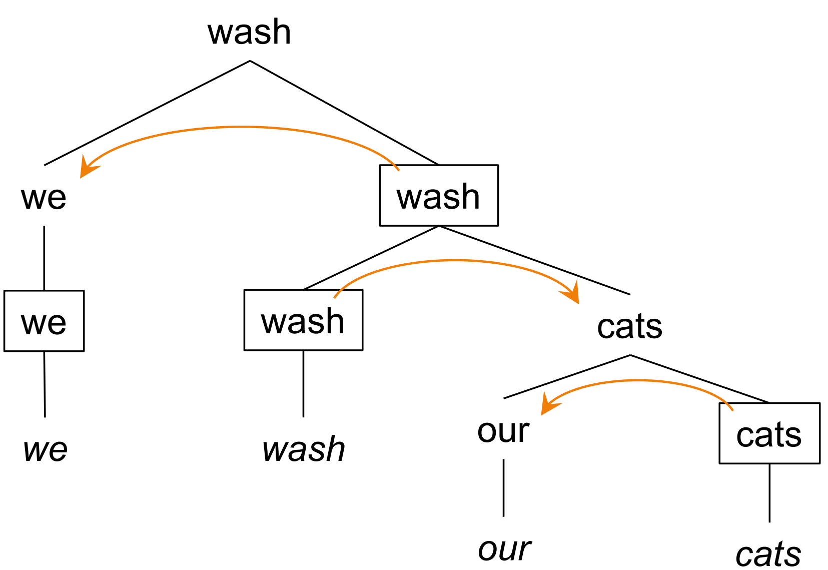
Conversion

- From a constituent tree to a dependency tree
  - Constituent tree with heads, lexicalized



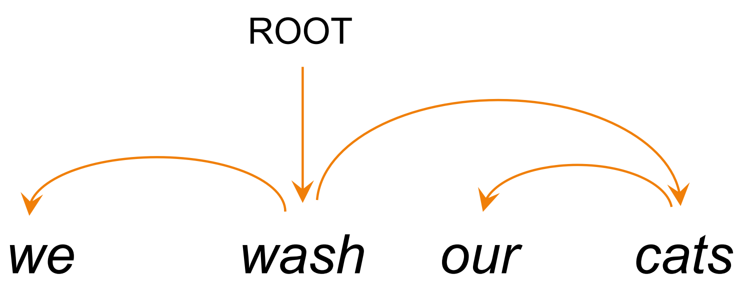
Constituency to dependency

- From a constituent tree to a dependency tree
  - Constituent tree with heads, lexicalized



Conversion

- From a constituent tree to a dependency tree
  - Dependency tree



Summary

- Constituency parsing
  - (Probabilistic) context-free grammars
  - CYK algorithm
- Regular grammars
- Dependency parsing