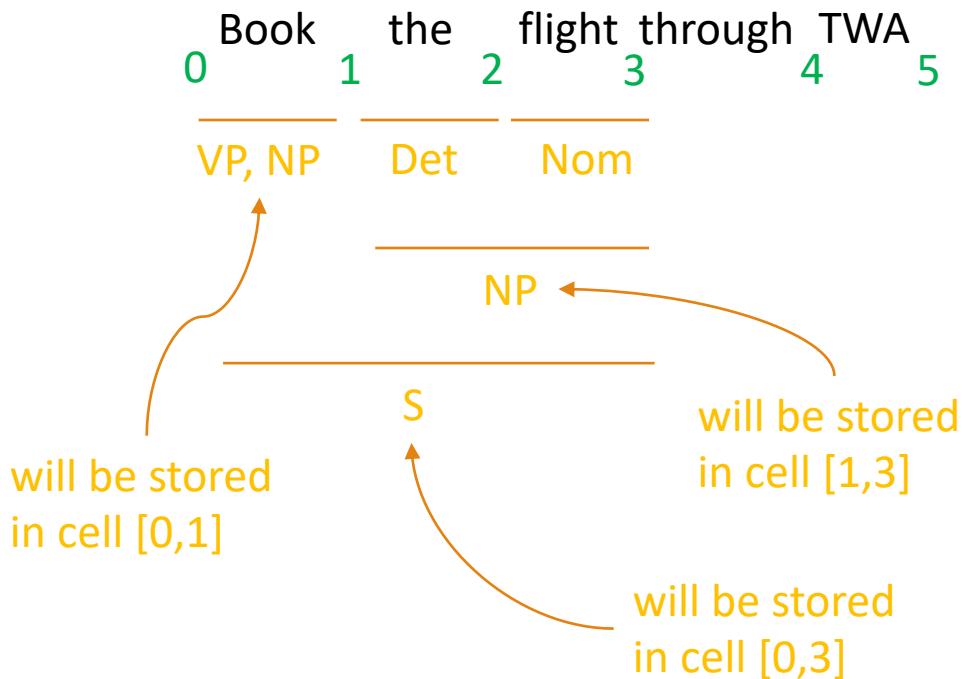


Statistical Constituency Parsing

JURAFSKY AND MARTIN CHAPTER 14

Recap: CKY Algorithm



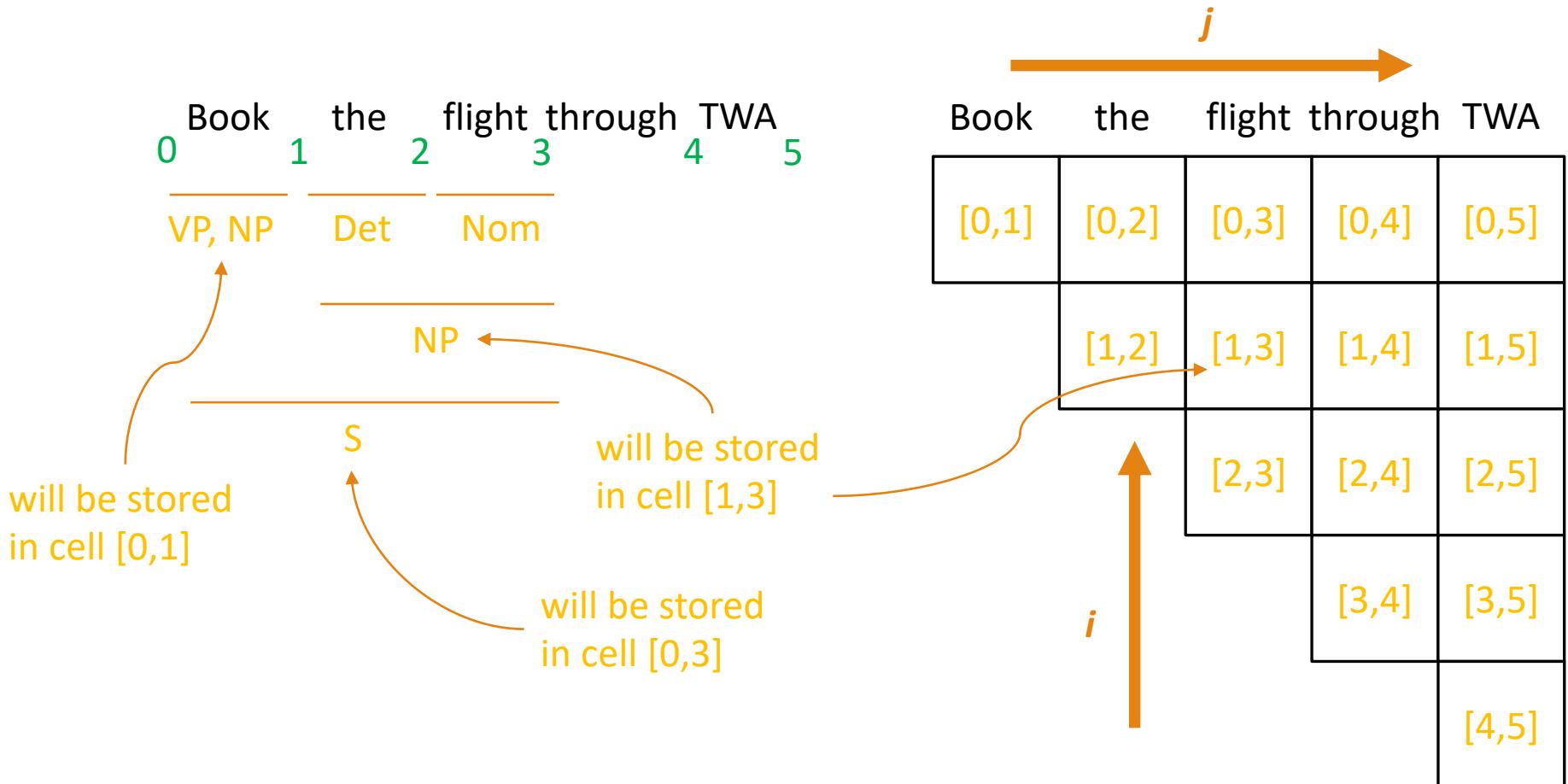
CKY uses a two-dimensional matrix to encode the structure of a tree.

For a sentence of length n , we will use an $(n + 1) \times (n + 1)$ matrix.

You can think of the indexes as pointing at the gaps between the input words.

All constituents that can span positions i through j of the input will be stored in the cell $[i,j]$

Recap: CKY Algorithm



We only work with the upper-triangular portion of the $(n + 1) \times (n + 1)$ matrix.
 Each cell $[i, j]$ records all non-terminals that can span positions i through j of the input.

Recap: CKY Algorithm

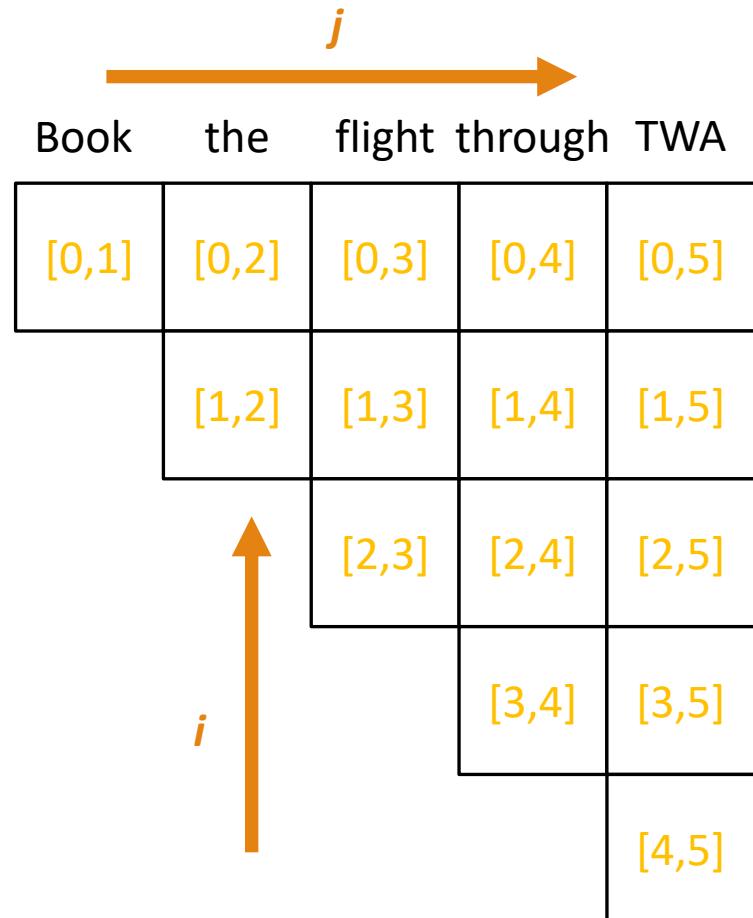
function CKY-PARSE(*words, grammar*)
returns *table*

for $j \leftarrow$ from 1 to LENGTH(words) **do**
for all { $A \mid A \rightarrow words[j] \in grammar$ }

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

for $i \leftarrow$ from $j-2$ downto 0 do
 for $k \leftarrow i+1$ to $j-1$ do
 i iterates over rows for all $\{A \mid A \rightarrow BC \in \text{grammar}$ and
 k iterates $B \in \text{table}[i,k]$ and
 over rows $C \in \text{table}[k,j]\}:$
 $\text{table}[i,j] \leftarrow \text{table}[i,j] \cup A$

k iterates
split points
between i
and j



Recap: CKY Algorithm

function CKY-PARSE(*words, grammar*)

returns *table*

for $j \leftarrow$ from 1 to LENGTH(*words*) **do**
 record POS tags for word j in cell $[j, j-1]$

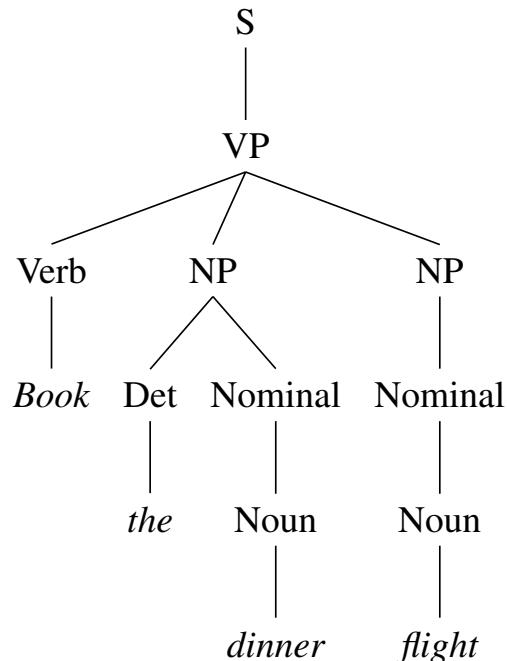
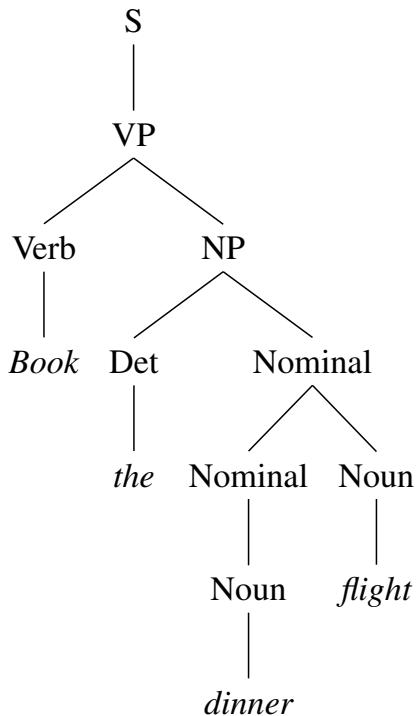
for $i \leftarrow$ from $j-2$ down to 0 **do**
 for $k \leftarrow i+1$ to $j-1$ **do**
 find all rules $A \rightarrow BC$, such that
 B spans $i-k$,
 C spans $k-j$
 and then record A in cell $[i, j]$

The diagram shows a CKY table for the sentence "Book the flight through TWA". The words are listed along the top: Book, the, flight, through, TWA. An orange arrow labeled j points to the word "TWA". The table has 5 columns and 5 rows. The first column is labeled "Book", the second "the", the third "flight", the fourth "through", and the fifth "TWA". The first row is labeled "[0,1]" and corresponds to the word "Book". The second row is labeled "[1,2]" and corresponds to the word "the". The third row is labeled "[2,3]" and corresponds to the word "flight". The fourth row is labeled "[3,4]" and corresponds to the word "through". The fifth row is labeled "[4,5]" and corresponds to the word "TWA". Orange arrows point from the labels "j" and "i" to the top and left edges of the table respectively.

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

Find *best* parse

CKY parsing record **ALL** possible parses into the table. How do we figure out what the best parse is for a sentence?



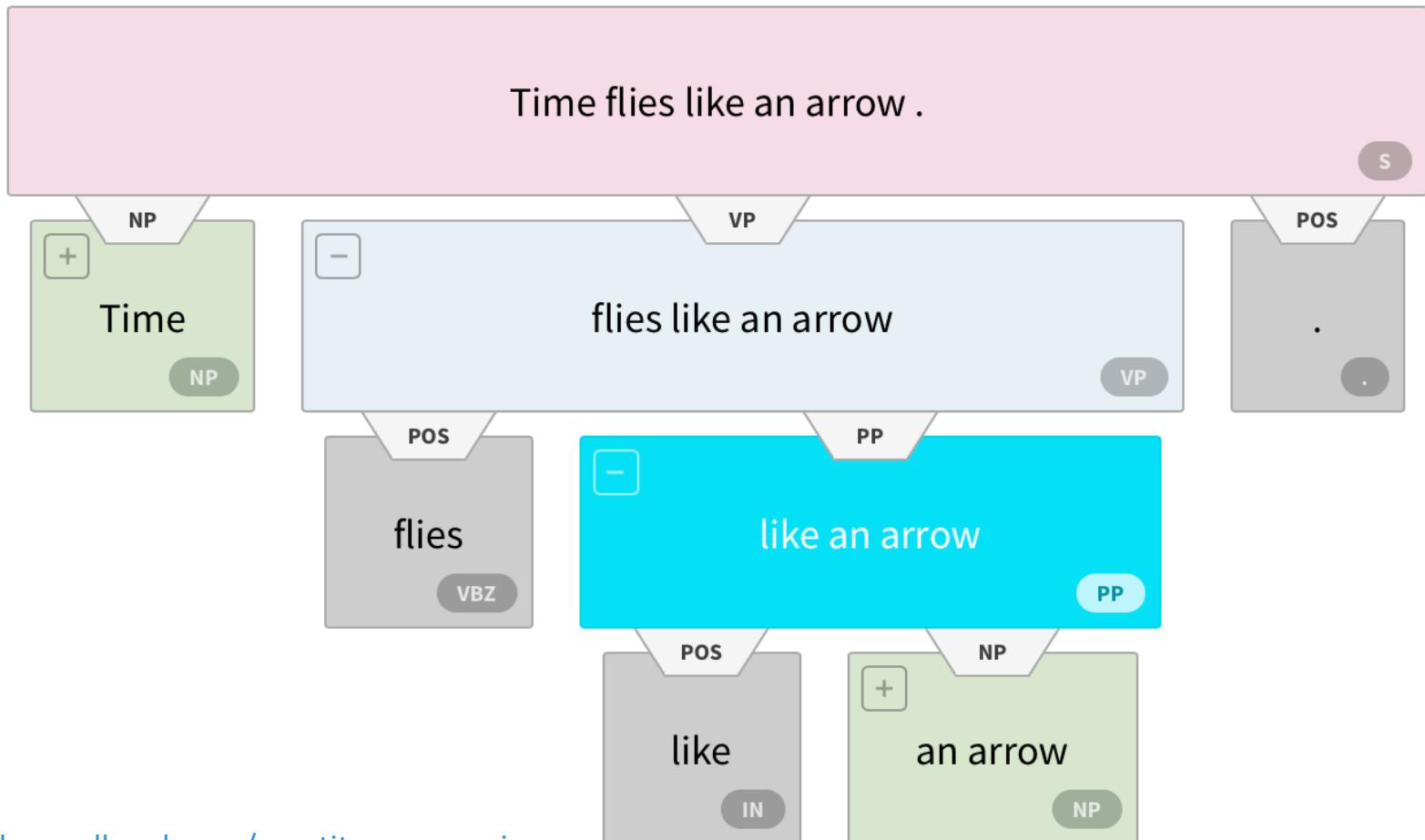
Ambiguity: Which parse?

Time flies like an arrow.



Ambiguity: Which parse?

Time flies like an arrow.



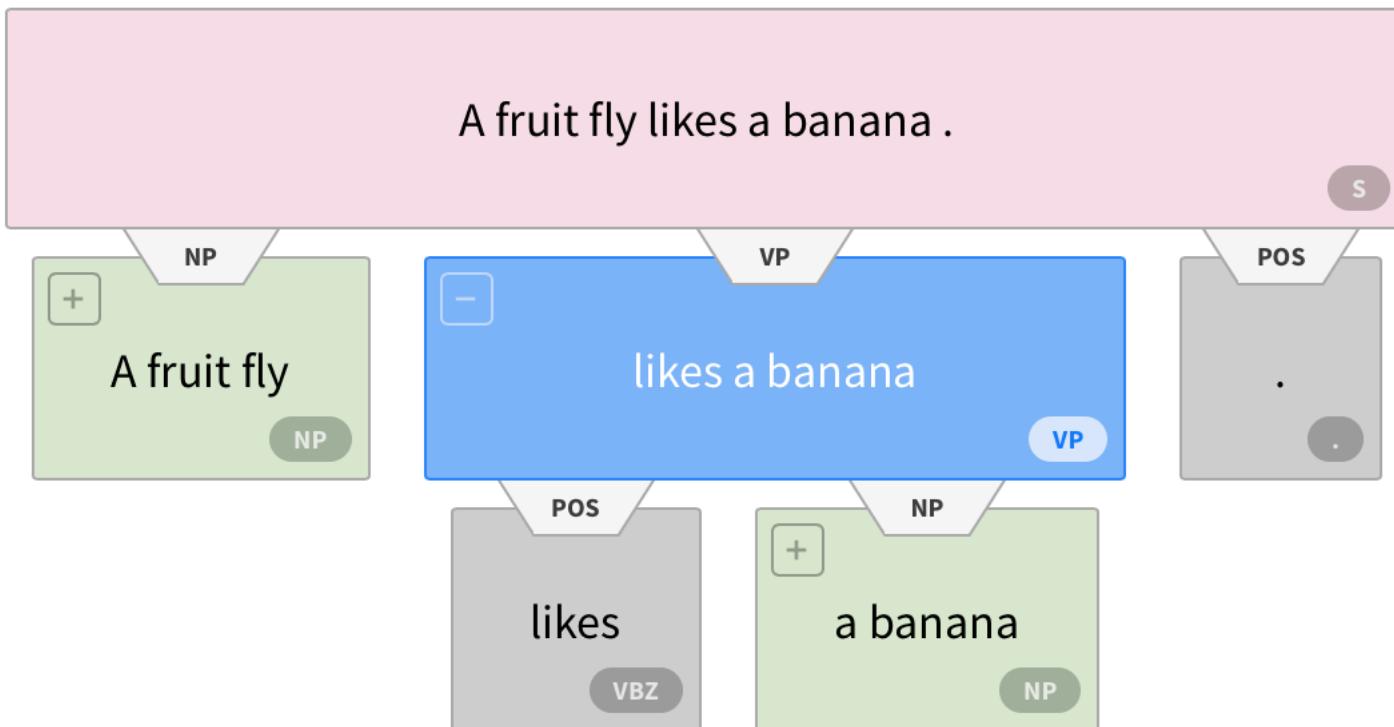
Ambiguity: Which parse?

Fruit flies like a banana.



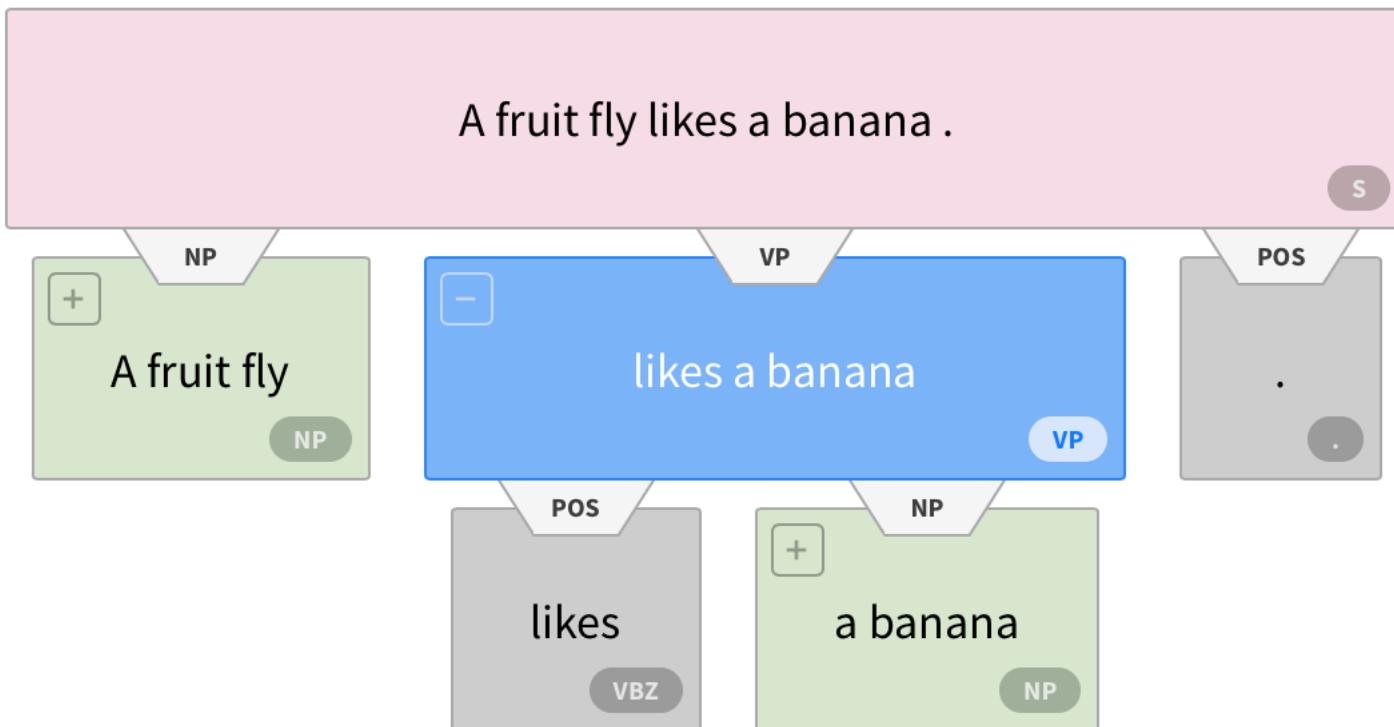
Ambiguity: Which parse?

Fruit flies like a banana.



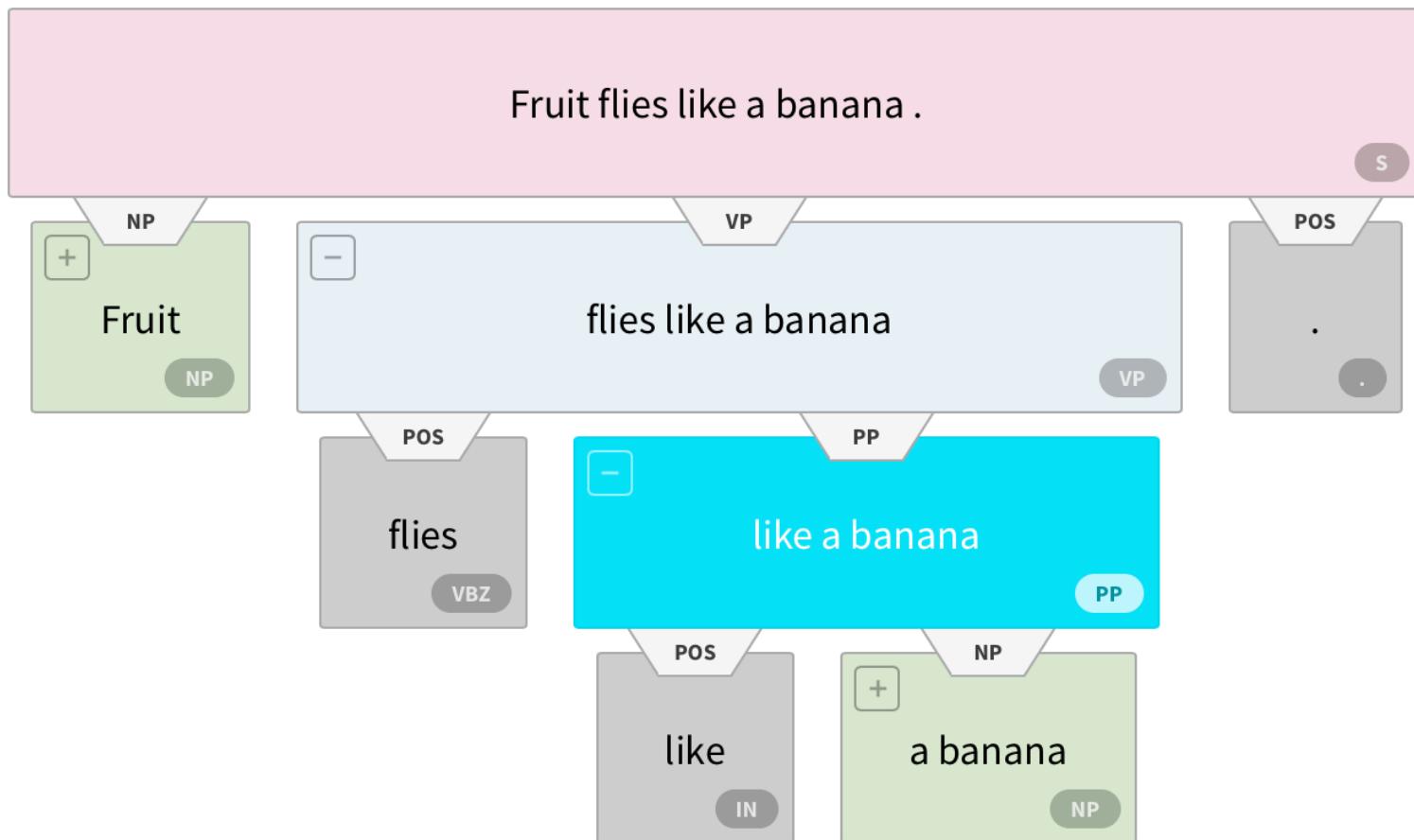
Ambiguity: Which parse?

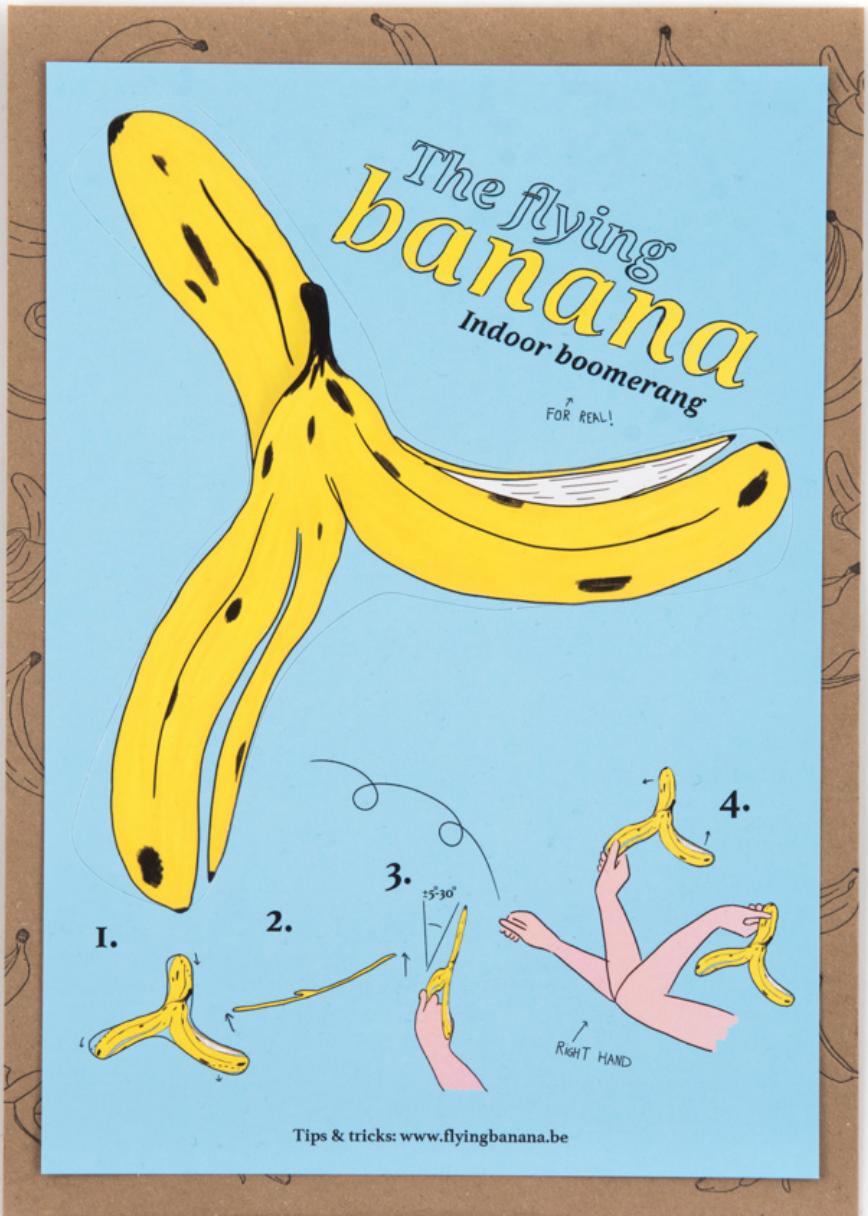
Fruit flies like a banana.



Ambiguity: Which parse?

Fruit flies like a banana.





Formal Definition of a CFG

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

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

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

Σ is set of **terminal symbols**

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

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

- $S \rightarrow Aux\ NP\ VP$
- $Nominal \rightarrow Nominal\ Gerund\ VP$ (recursive)

S is the start symbol (a non-terminal)

Formal Definition of a PCFG

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

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

- $S \rightarrow NP VP$ [0.8]
- $S \rightarrow Aux NP VP$ [0.15]
- $S \rightarrow VP$ [0.05]

For a rule $A \rightarrow B C$, the probability can be represented

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

or

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

or

$$P(RHS | LHS)$$

Grammar		Lexicon
$S \rightarrow NP VP$	[.80]	$Det \rightarrow that [.10] \mid a [.30] \mid the [.60]$
$S \rightarrow Aux NP VP$	[.15]	$Noun \rightarrow book [.10] \mid flight [.30]$
$S \rightarrow VP$	[.05]	$\mid meal [.05] \mid money [.05]$
$NP \rightarrow Pronoun$	[.35]	$\mid flight [.40] \mid dinner [.10]$
$NP \rightarrow Proper-Noun$	[.30]	$Verb \rightarrow book [.30] \mid include [.30]$
$NP \rightarrow Det Nominal$	[.20]	$\mid prefer [.40]$
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I [.40] \mid she [.05]$
$Nominal \rightarrow Noun$	[.75]	$\mid me [.15] \mid you [.40]$
$Nominal \rightarrow Nominal Noun$	[.20]	$Proper-Noun \rightarrow Houston [.60]$
$Nominal \rightarrow Nominal PP$	[.05]	$\mid NWA [.40]$
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does [.60] \mid can [.40]$
$VP \rightarrow Verb NP$	[.20]	$Preposition \rightarrow from [.30] \mid to [.30]$
$VP \rightarrow Verb NP PP$	[.10]	$\mid on [.20] \mid near [.15]$
$VP \rightarrow Verb PP$	[.15]	$\mid through [.05]$
$VP \rightarrow Verb NP NP$	[.05]	
$VP \rightarrow VP PP$	[.15]	
$PP \rightarrow Preposition NP$	[1.0]	

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

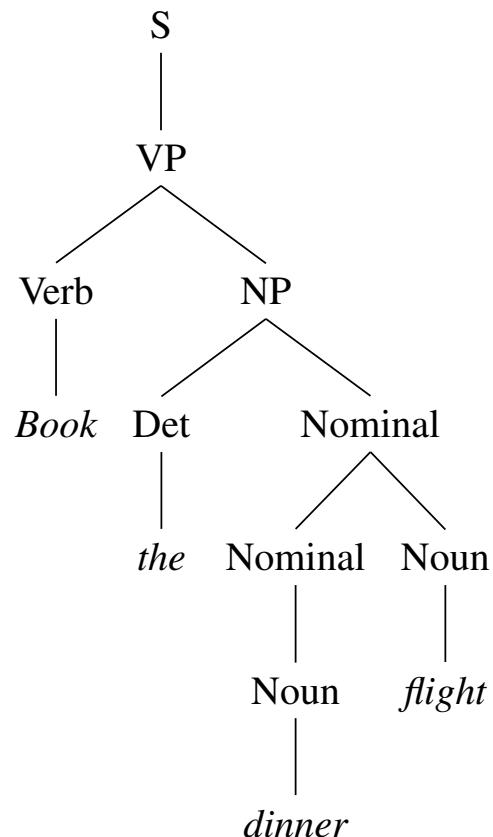
Probabilistic CFGs

A PCFG assigns a probability to each parse tree T of a sentence S . This is useful in disambiguation, since we can pick the most likely parse tree.

The probability of a parse T is defined as the product of the probabilities of all the rules used to expand each of the non-terminal nodes in the parse tree.

$$P(T, S) = \prod_{i=1}^n P(RHS_i | LHS_i)$$

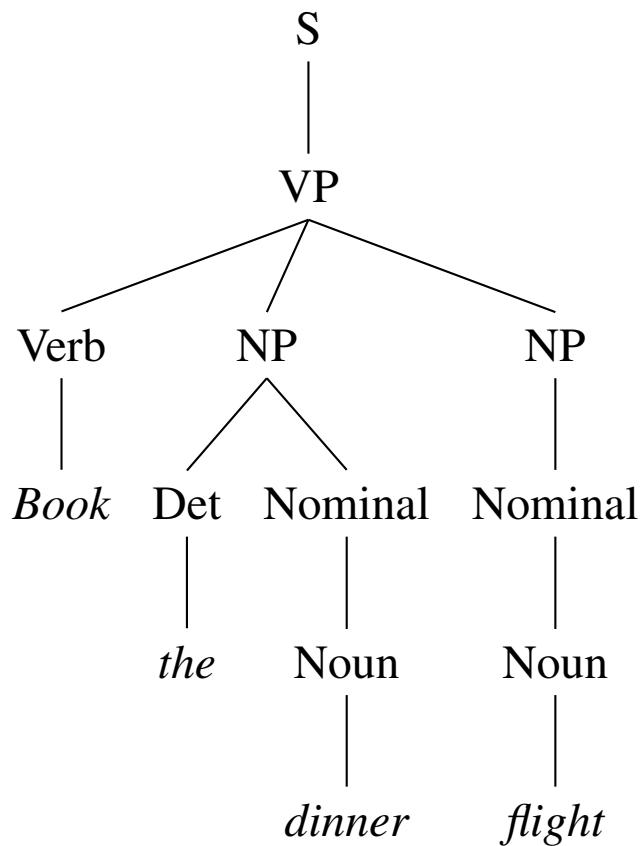
Probability of a parse



	Rules	P
S	\rightarrow VP	.05
VP	\rightarrow Verb NP	.20
NP	\rightarrow Det Nominal	.20
Nominal	\rightarrow Nominal Noun	.20
Nominal	\rightarrow Noun	.75
Verb	\rightarrow book	.30
Det	\rightarrow the	.60
Noun	\rightarrow dinner	.10
Noun	\rightarrow flight	.40

$$P(T, S) = \prod_{i=1}^n P(RHS_i | LHS_i) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = 2.2 \times 10^{-6}$$

Probability of a parse



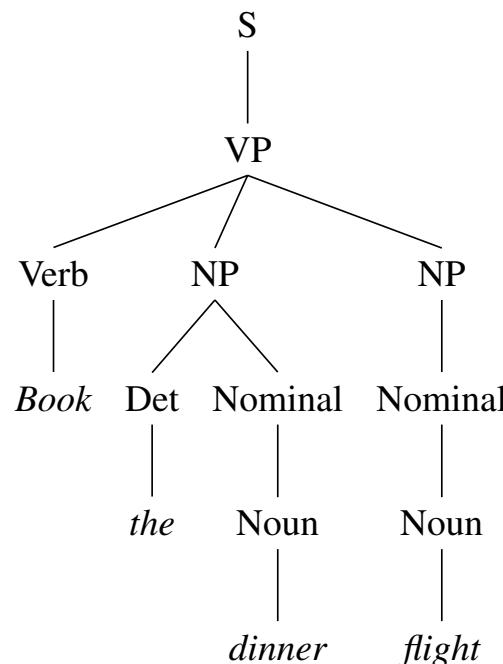
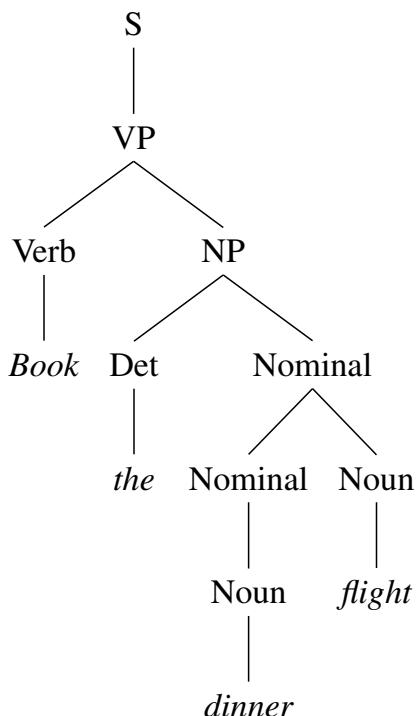
	Rules	P
S	\rightarrow VP	.05
VP	\rightarrow Verb NP NP	.10
NP	\rightarrow Det Nominal	.20
NP	\rightarrow Nominal	.15
Nominal	\rightarrow Noun	.75
Nominal	\rightarrow Noun	.75
Verb	\rightarrow book	.30
Det	\rightarrow the	.60
Noun	\rightarrow dinner	.10
Noun	\rightarrow flight	.40

$$P(T, S) = \prod_{i=1}^n P(RHS_i | LHS_i) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

Finding best parse

Pick the parse with the highest probability. Consider all the possible parse trees for a given sentence S . The string of words S is called the yield of any parse tree over S .

$$\hat{T}(S) = \underset{T \text{ s.t. } S = \text{yield}(T)}{\operatorname{argmax}} P(T|S)$$



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 .

$$\underset{t \in T_x}{\operatorname{argmax}} p(t \mid x)$$

Even with small grammars, grows exponentially!

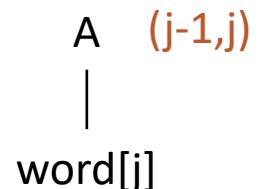
Probabilistic CKY Algorithm

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

Base case

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

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

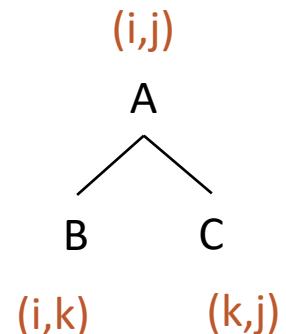


Recursion

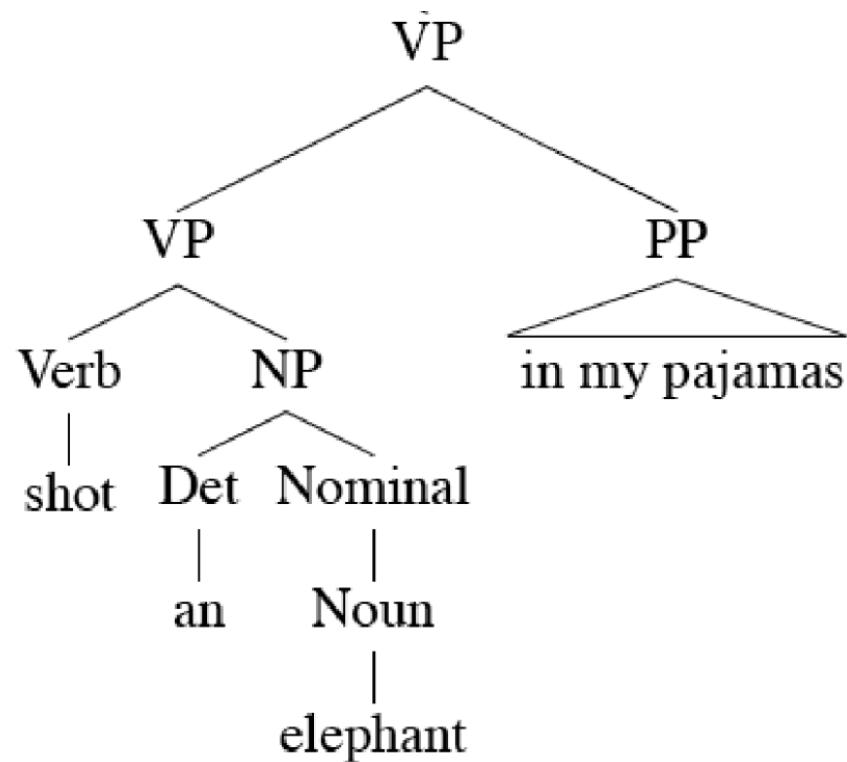
Rule: $P(A \rightarrow B C)$

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

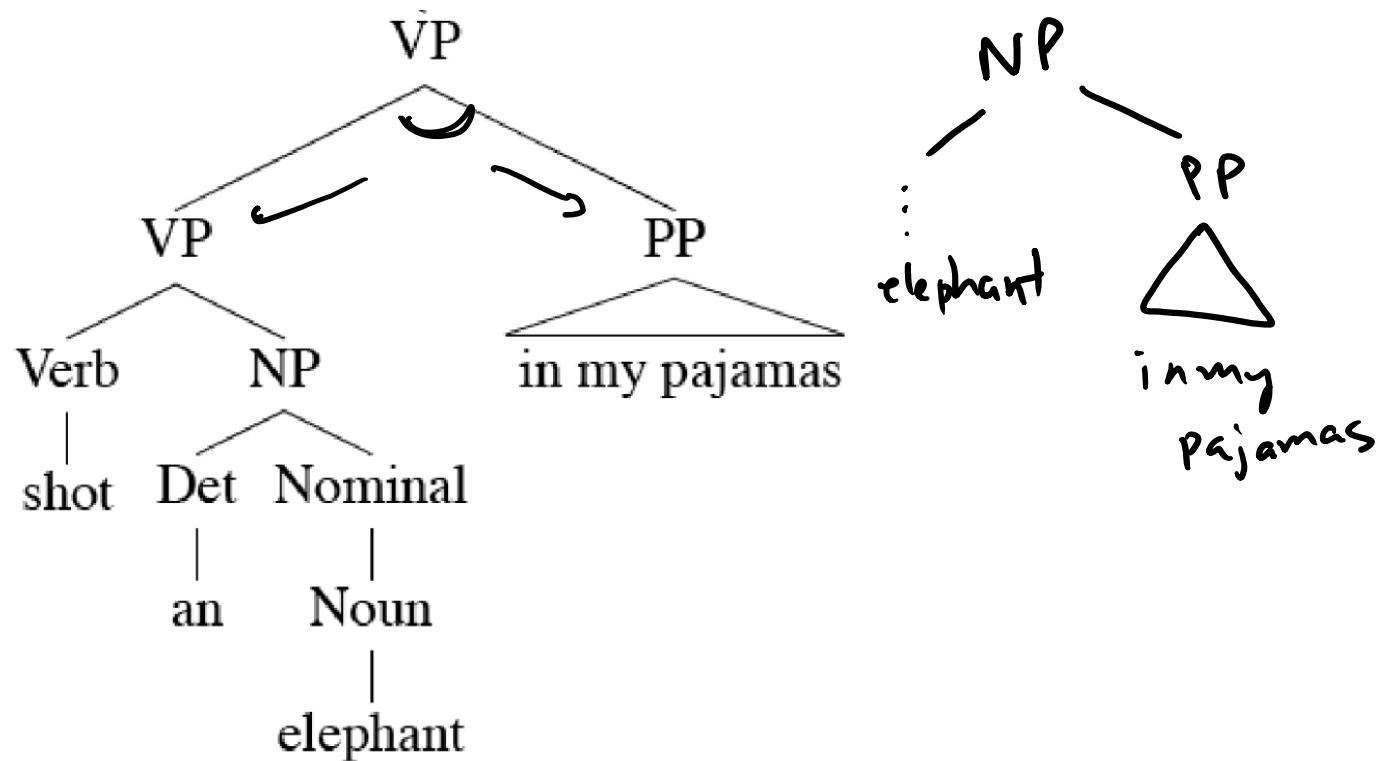
$$T[i,j,A] = \max_k P(B C \mid A) T[i,k,B] T[k,j,C]$$



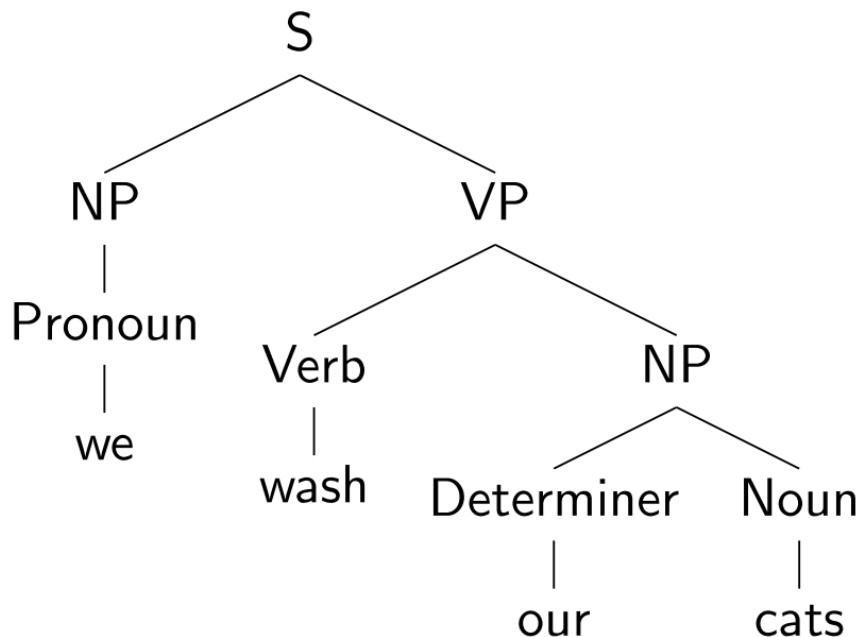
Lexicalized PCFGs



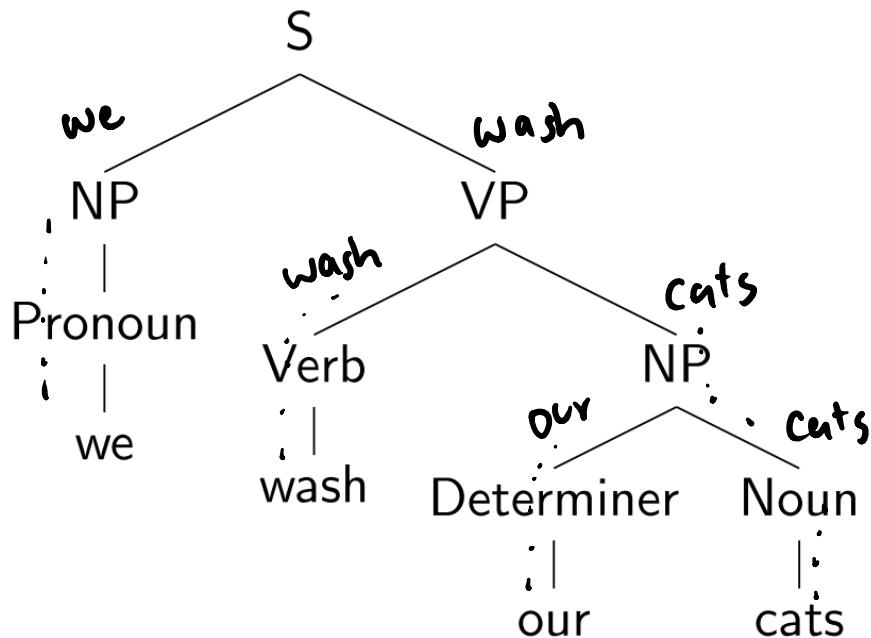
Lexicalized PCFGs



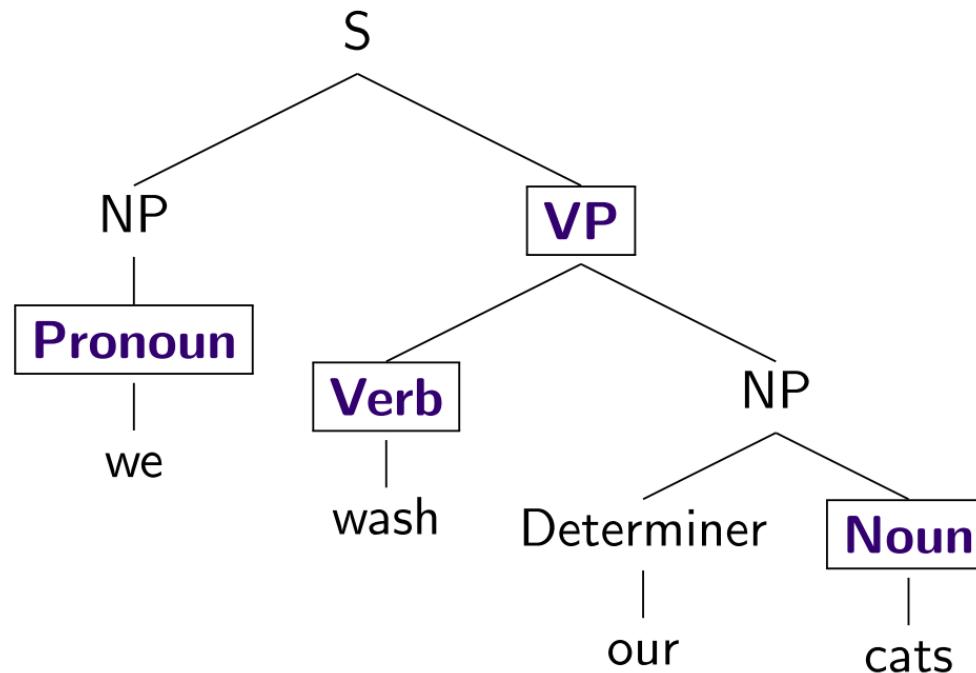
Lexicalizing a CFG



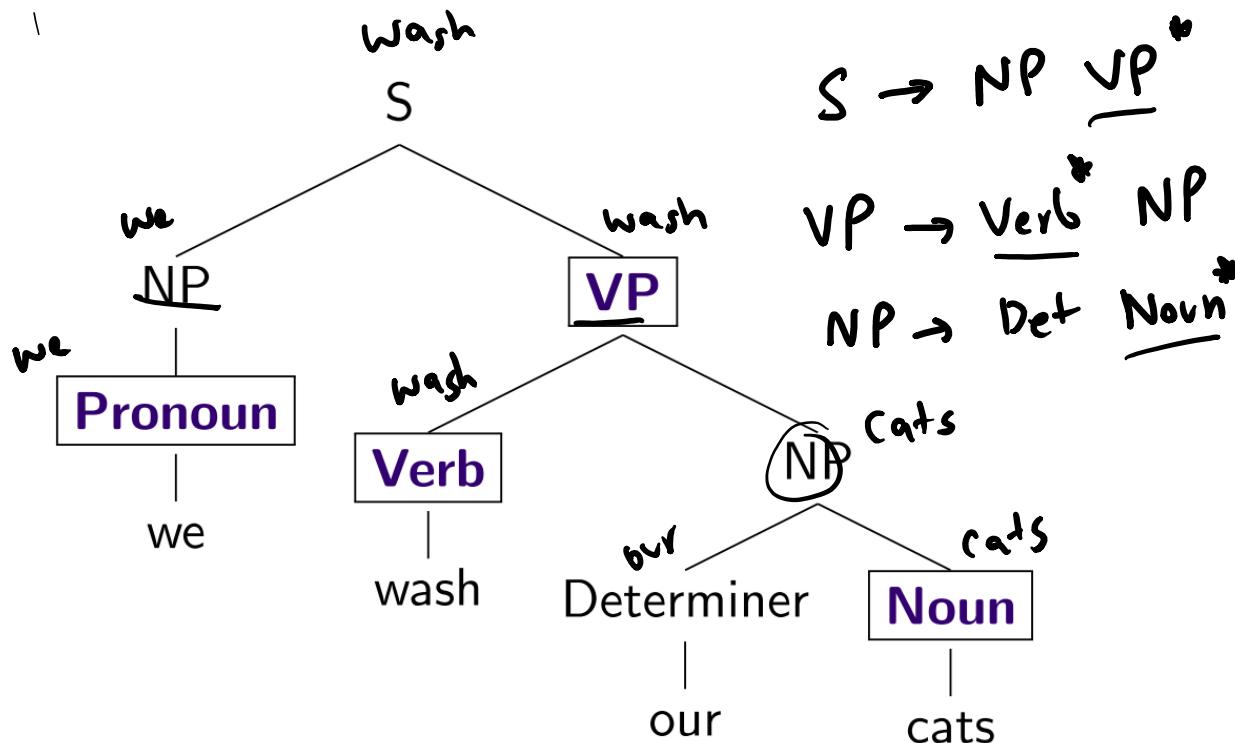
Lexicalizing a CFG



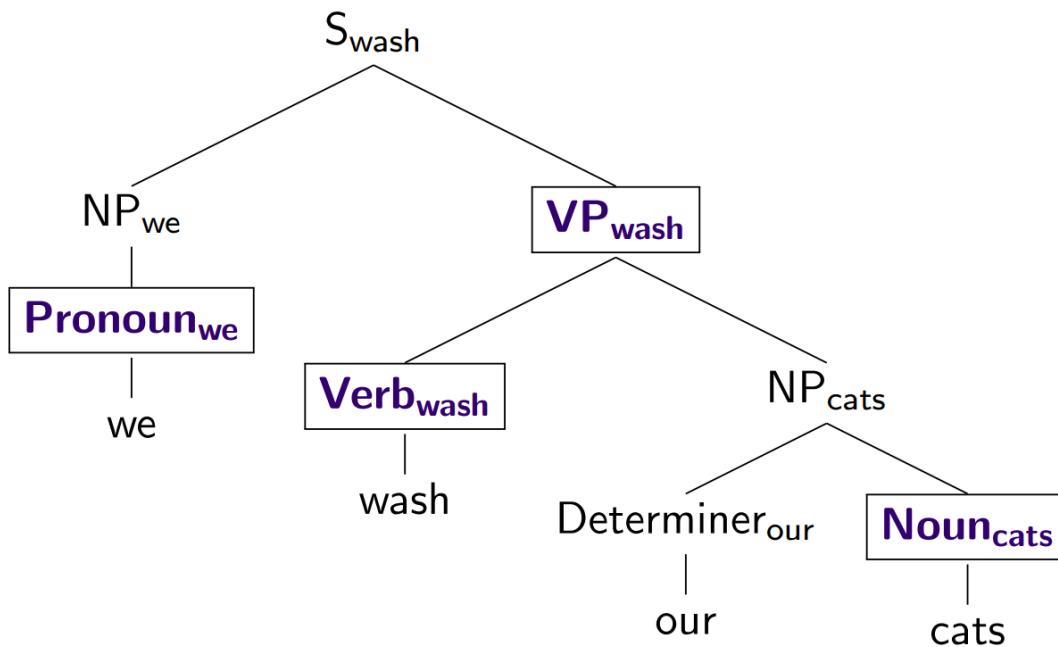
Lexicalizing a CFG



Lexicalizing a CFG



Lexicalizing a CFG



Outline

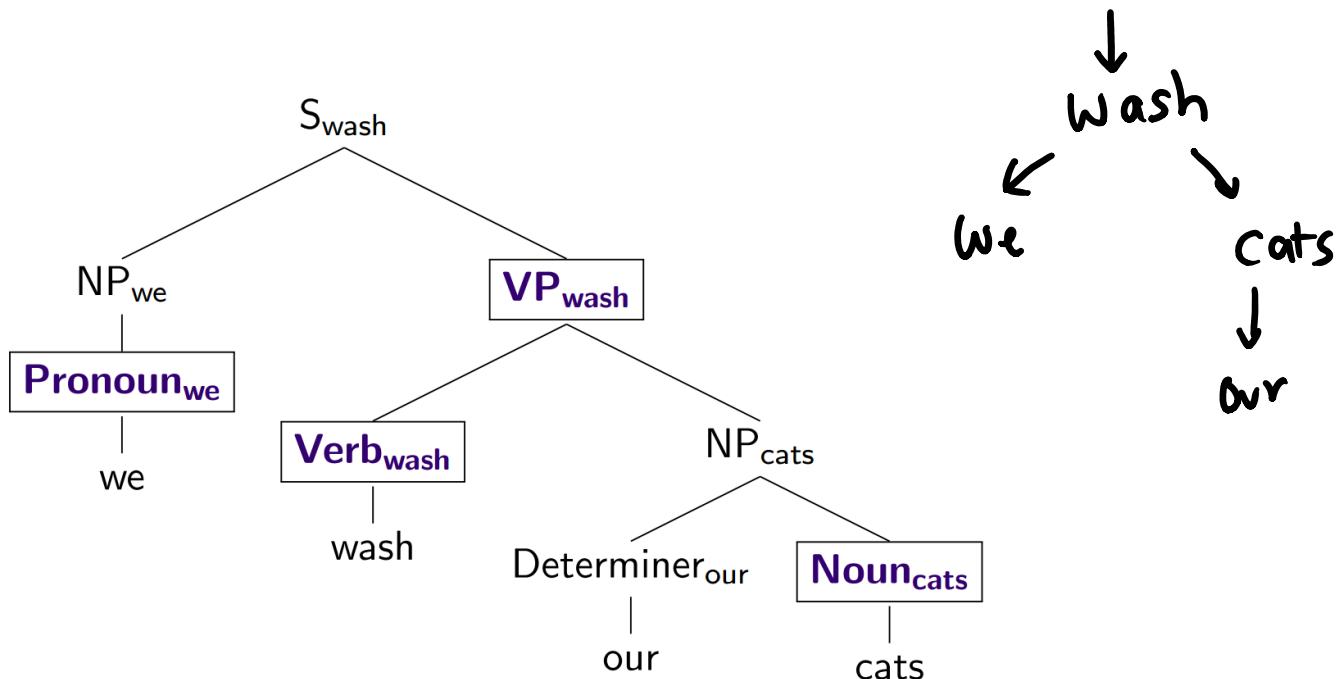
Parsing: CKY Algorithm

Extensions: Probabilistic and Lexicalized

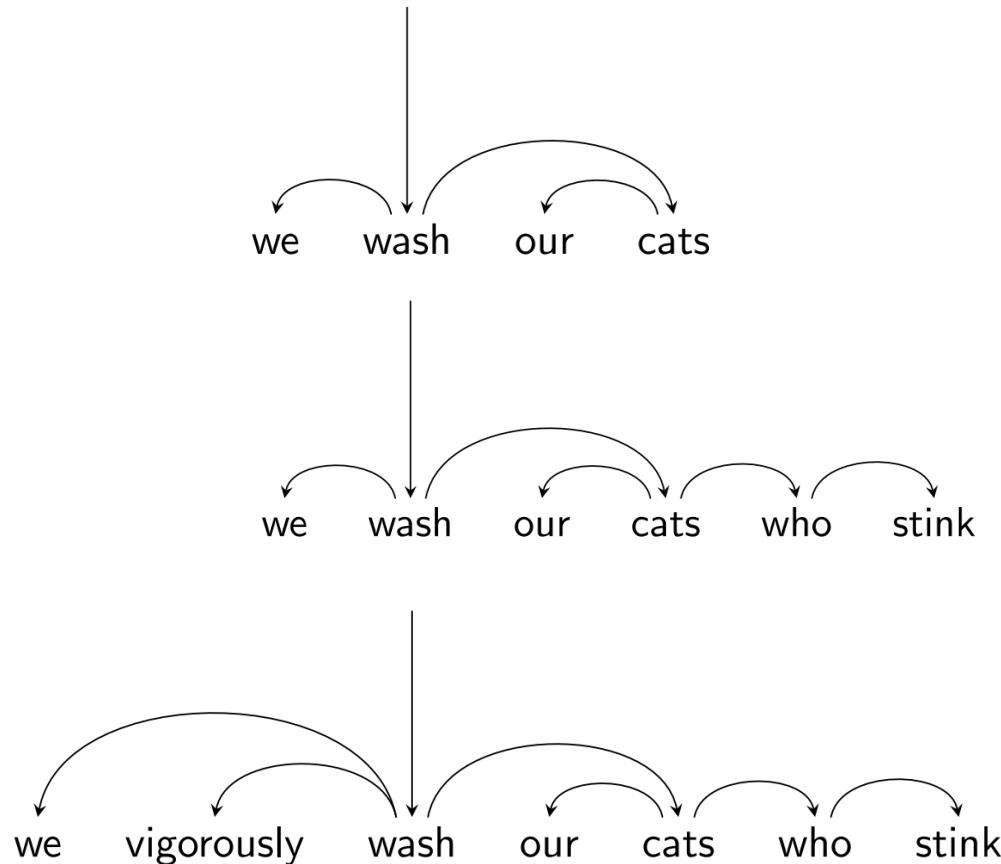
Dependency Parsing

Dependencies

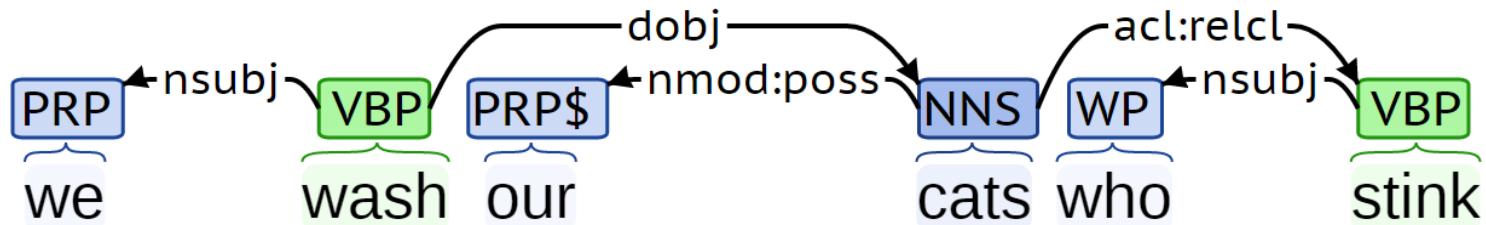
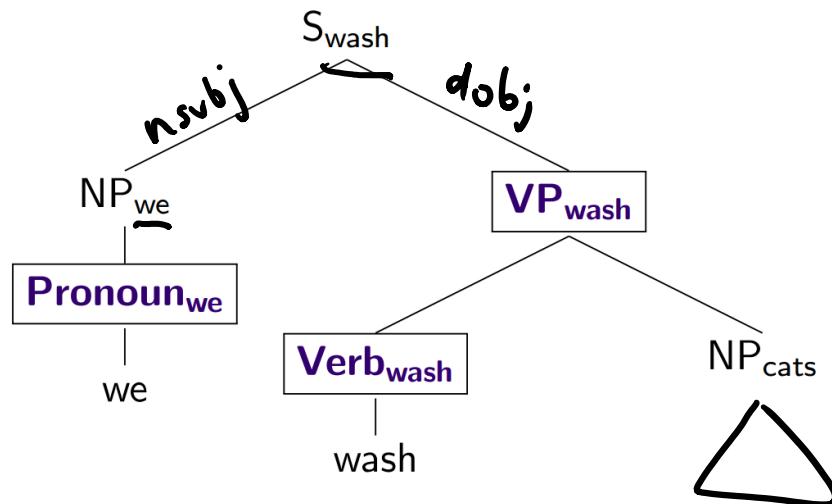
Represent only the syntactic dependencies...



Nested Structure = Subtrees



Dependency Labels



Dependency Labels

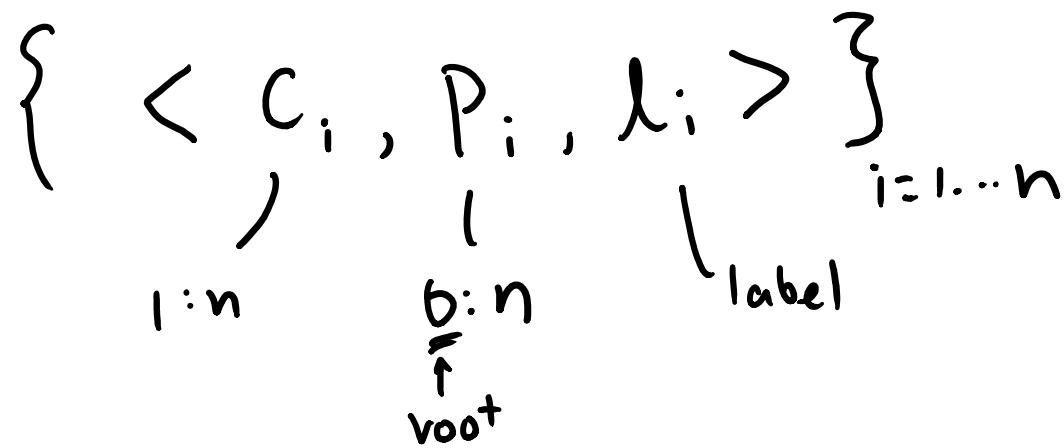
Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement

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

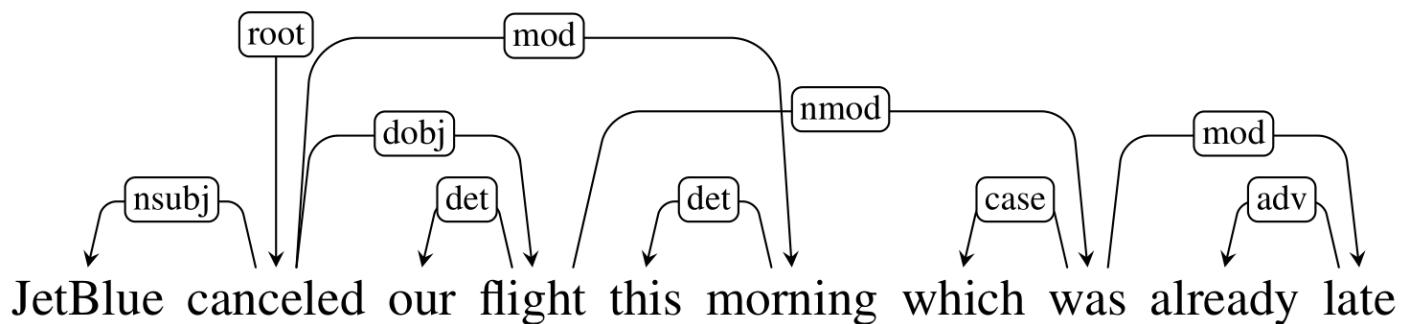
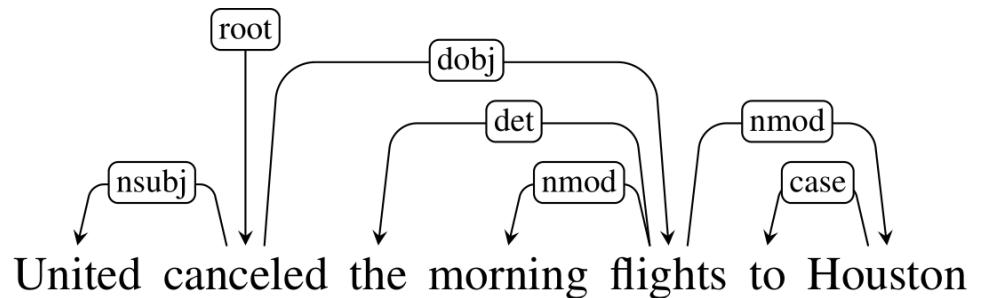
Dependency Trees

Dependency Trees

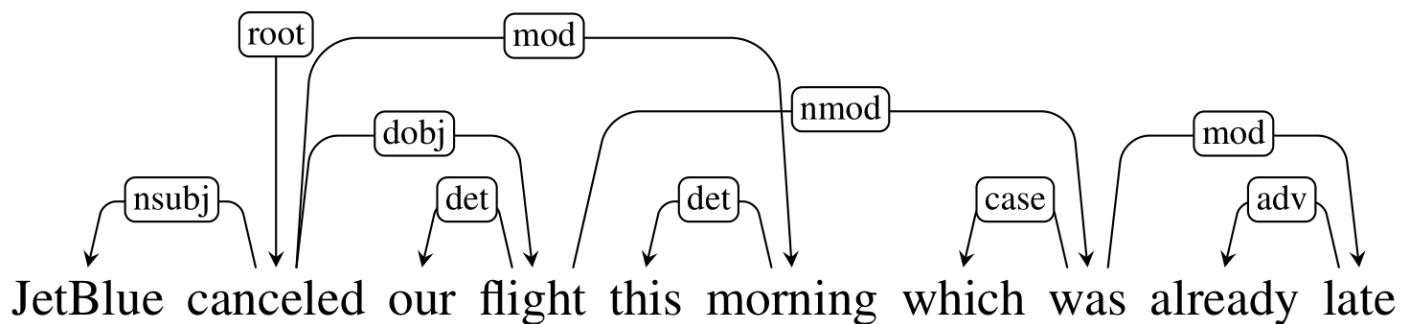
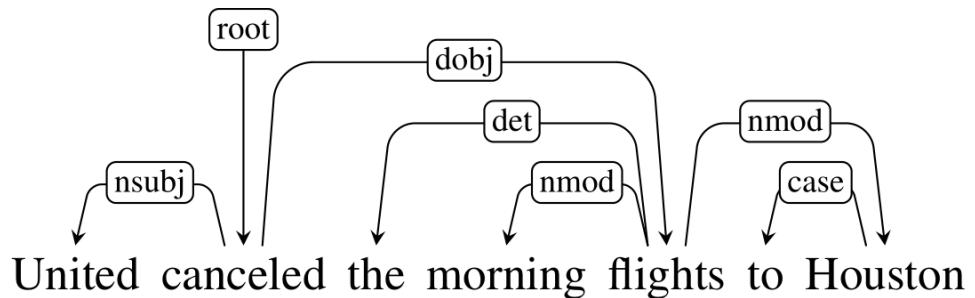
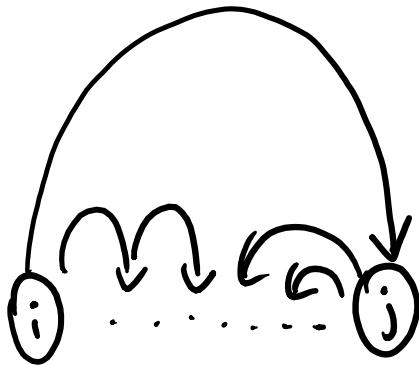


- single headed $\exists i$ only one $p_i = 0$
- connected - acyclic
- Projective vs non-projective

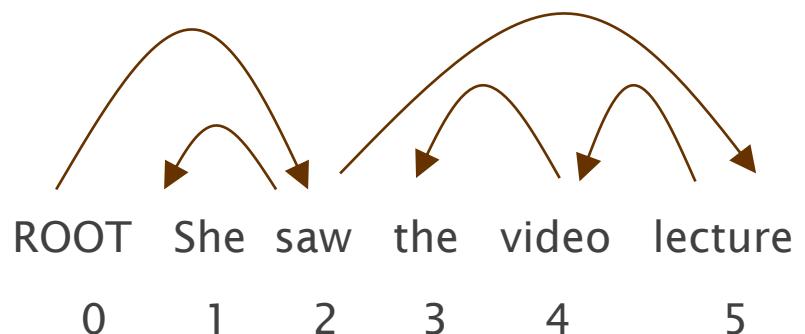
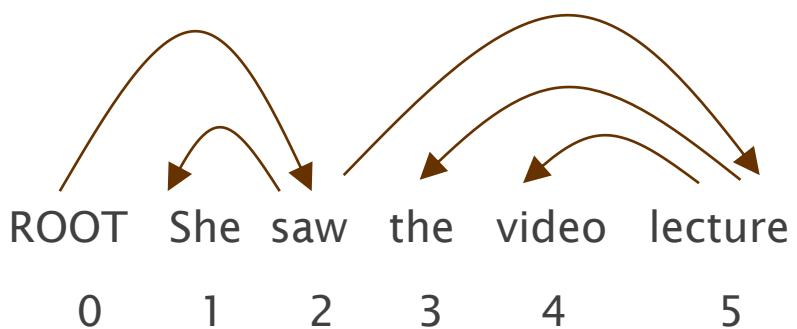
Projective vs Non-projective



Projective vs Non-projective



Evaluating Dependency Parses



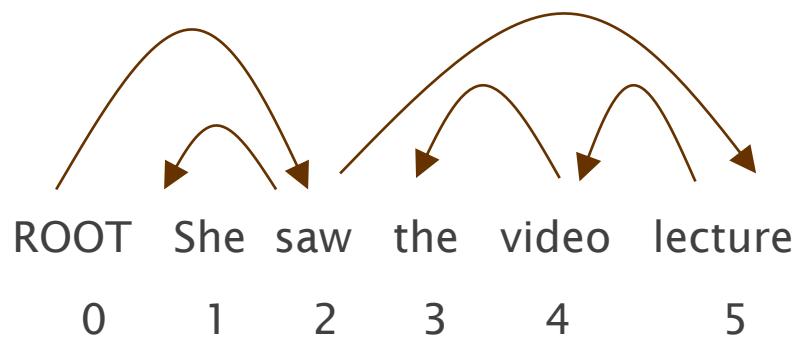
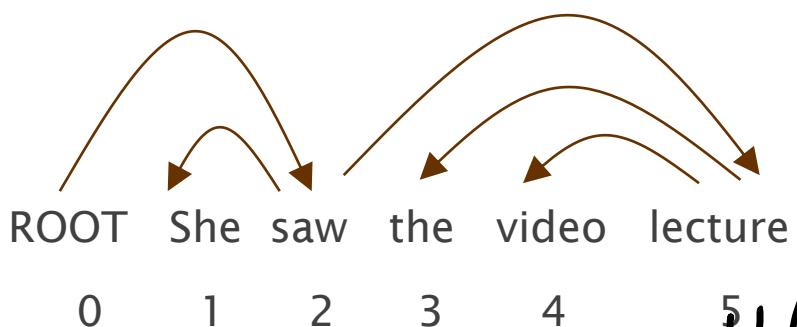
Gold

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

Parsed

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

Evaluating Dependency Parses



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

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

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

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

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

