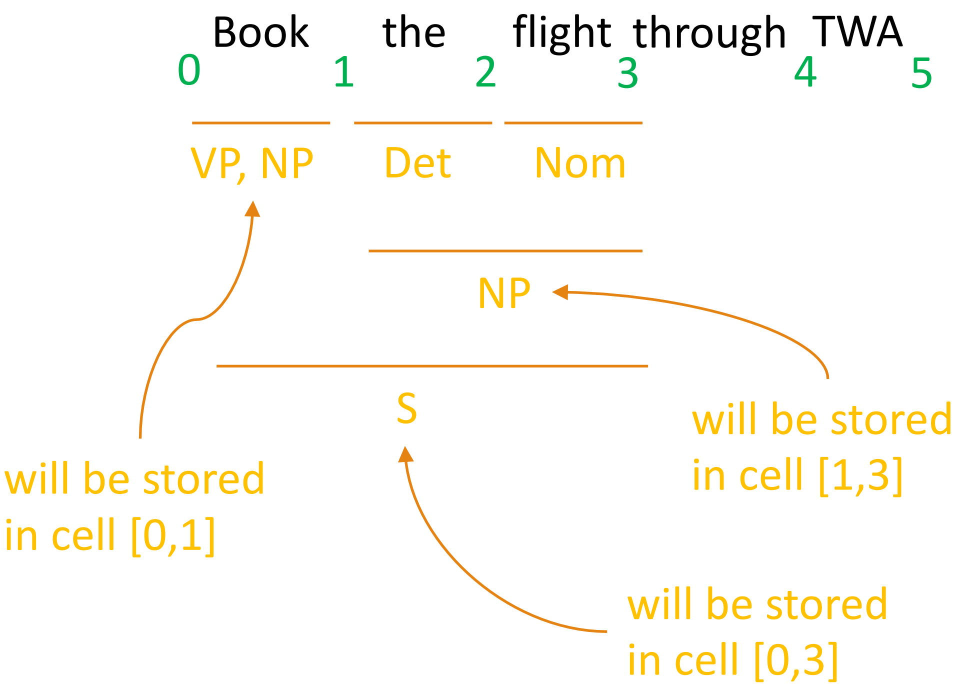
Statistical Constituency Parsing

JURAFSKY AND MARTIN CHAPTER 14

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

Recap: CKY Algorithm



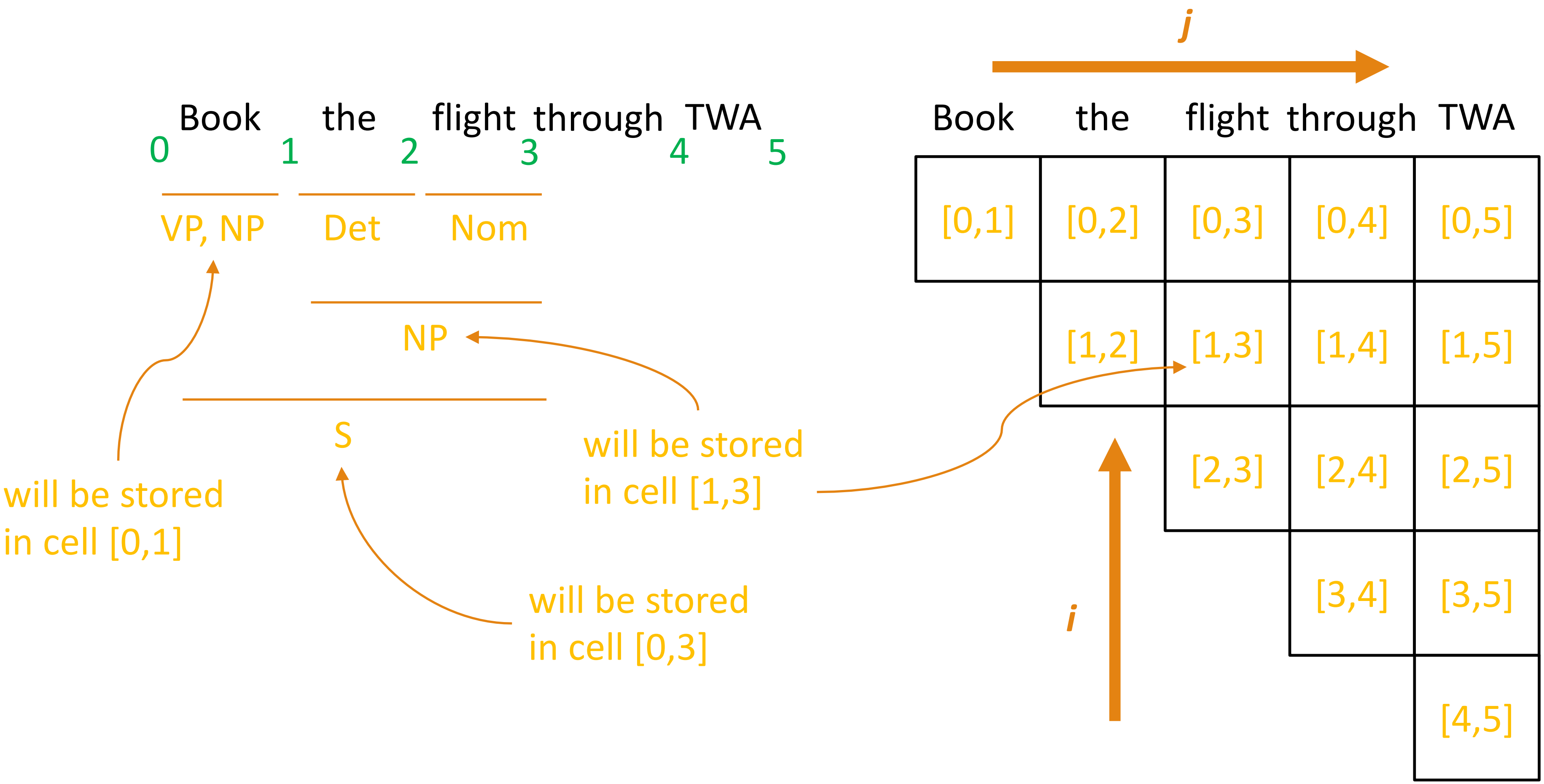
CKY uses a two-dimensional matrix to encode the structure of a tree.

For a sentence of length *n,* we will use an (n + 1) × (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

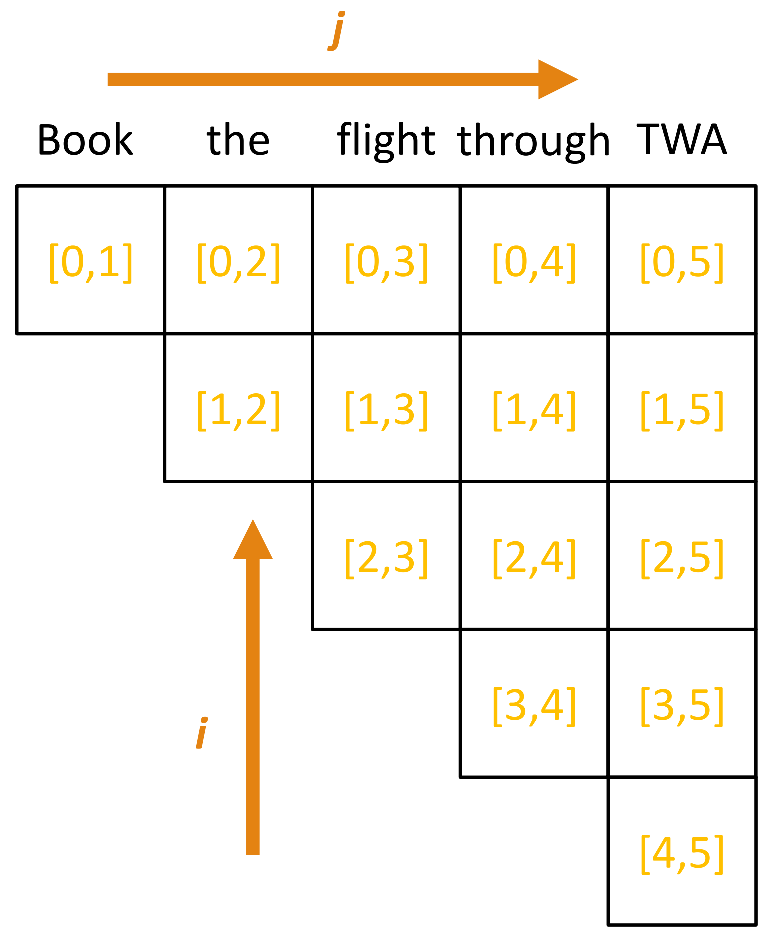
```
function CKY-PARSE(words, grammar) returns table

for j←from 1 to LENGTH(words) do
  for all {A | A → words[ j] ∈ grammar}
    table[j–1, j]←table[j–1, j] ∪ A
  for i←from j–2 downto 0 do
    for k←i+1 to j–1 do
      for all {A|A → BC ∈ grammar and
                B ∈ table[i,k] and
                C ∈ table[k,j]}:
          table[i,j]←table[i,j] ∪ A
```

j iterates over columns

i iterates over rows

k iterates split points between i and j

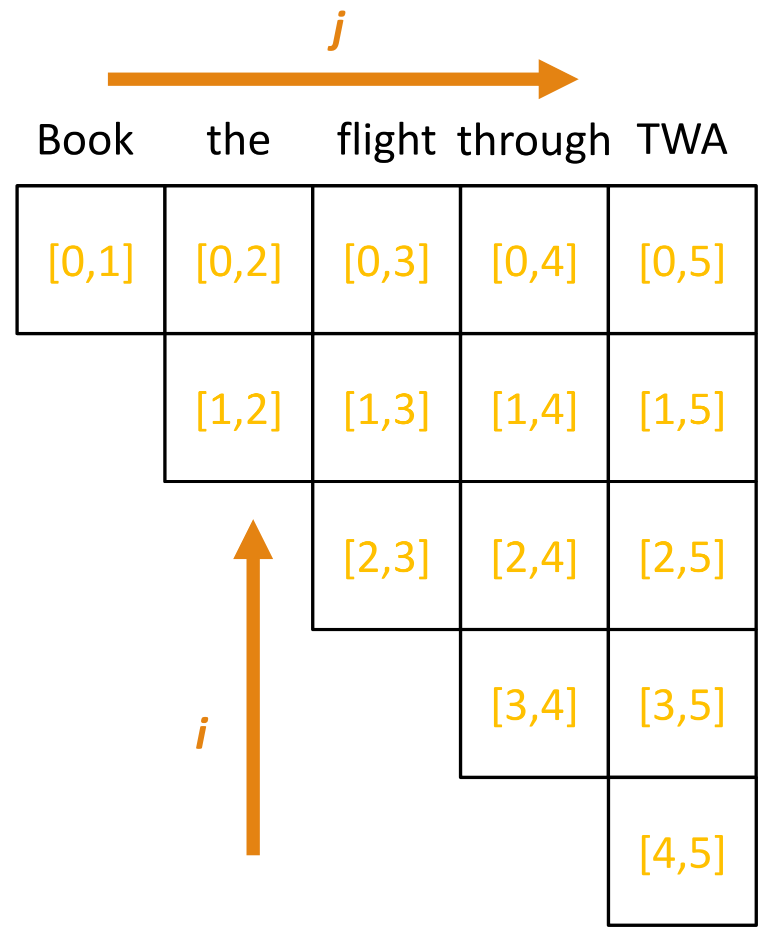


Recap: CKY Algorithm

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

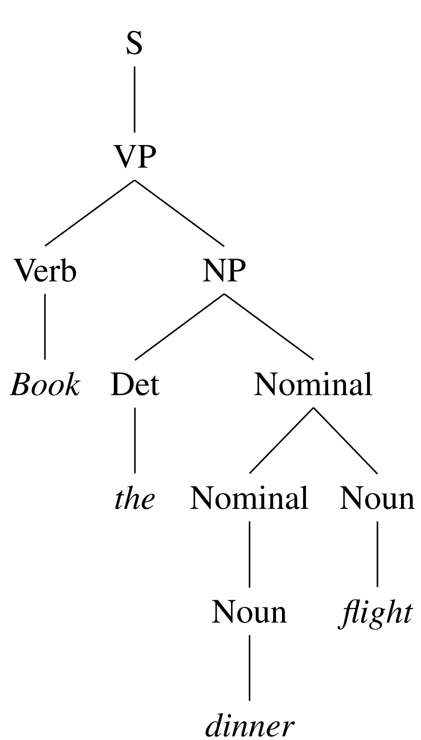
for *j*←**from** 1 **to** LENGTH(*words*) do
  record POS tags for word j in cell [j,j-1]

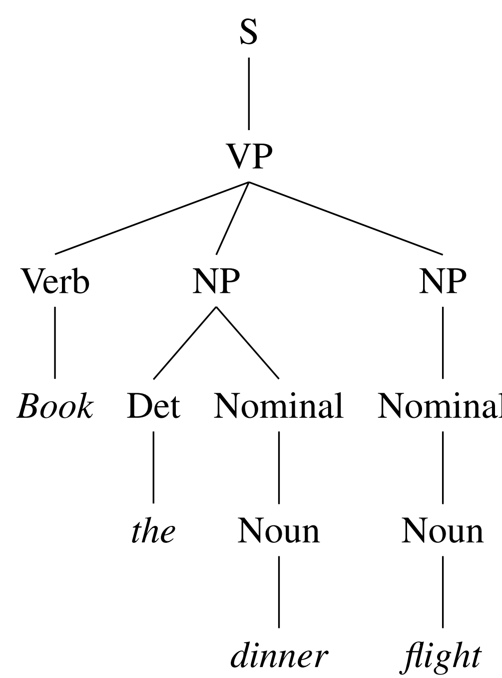
  for *i*←**from** *j*−2 **downto** 0 **do**
    for *k*←*i*+1 **to** *j*−1 **do**
      find all rules *A → BC, such that*
        B spans i-k,
        C spans k-j
      and then record A in cell [i,j]



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?



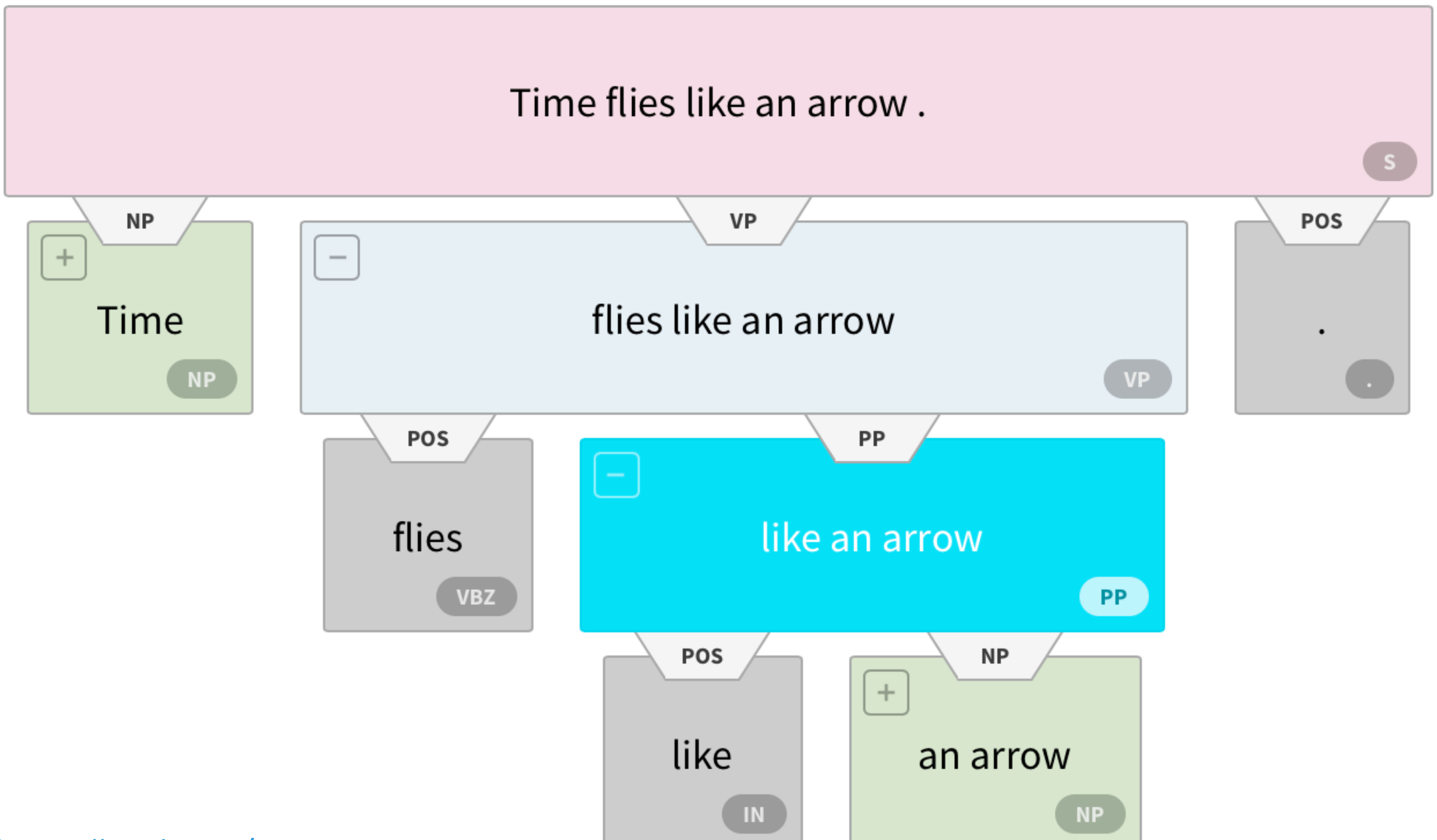


Ambiguity: Which parse?

Time flies like an arrow.

Ambiguity: Which parse?

Time flies like an arrow.



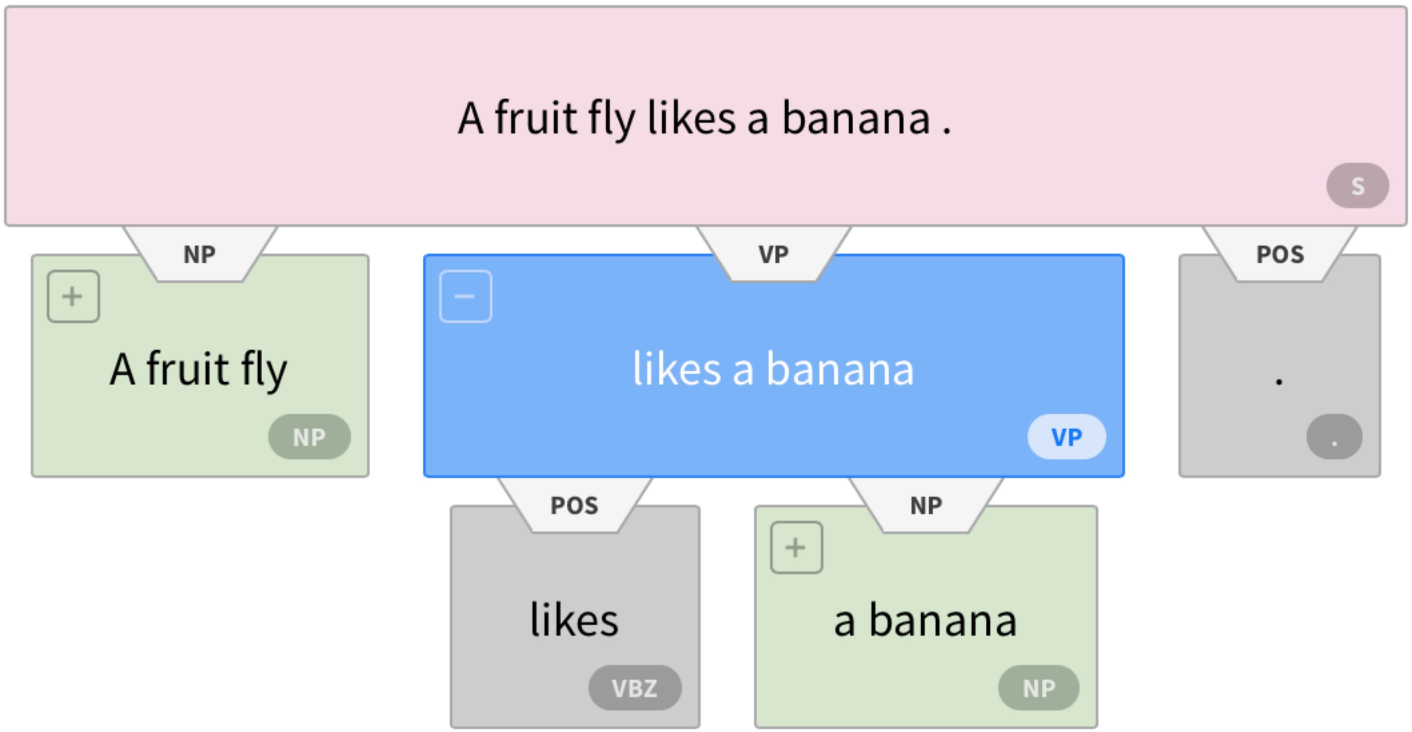
https://demo.allennlp.org/constituency-parsing

Ambiguity: Which parse?

Fruit flies like a banana.

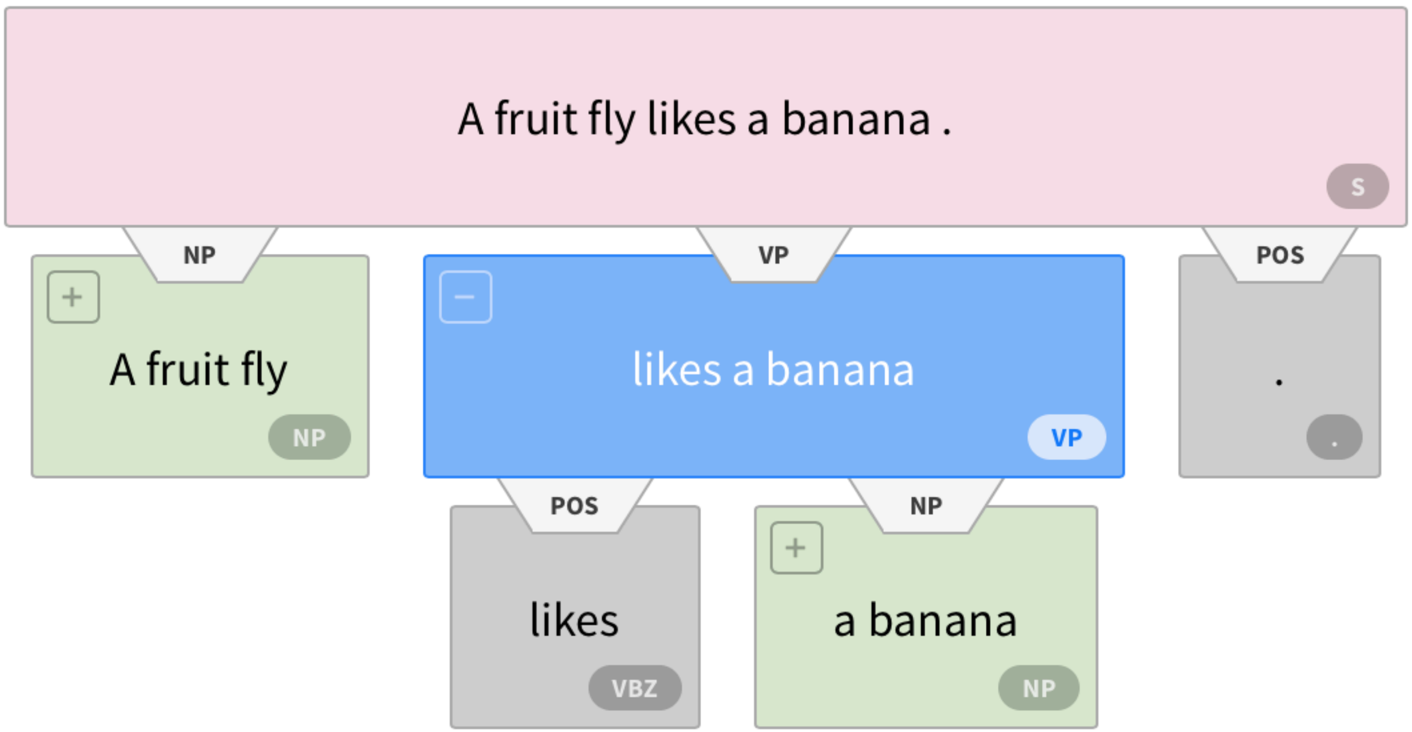
Ambiguity: Which parse?

Fruit flies like a banana.



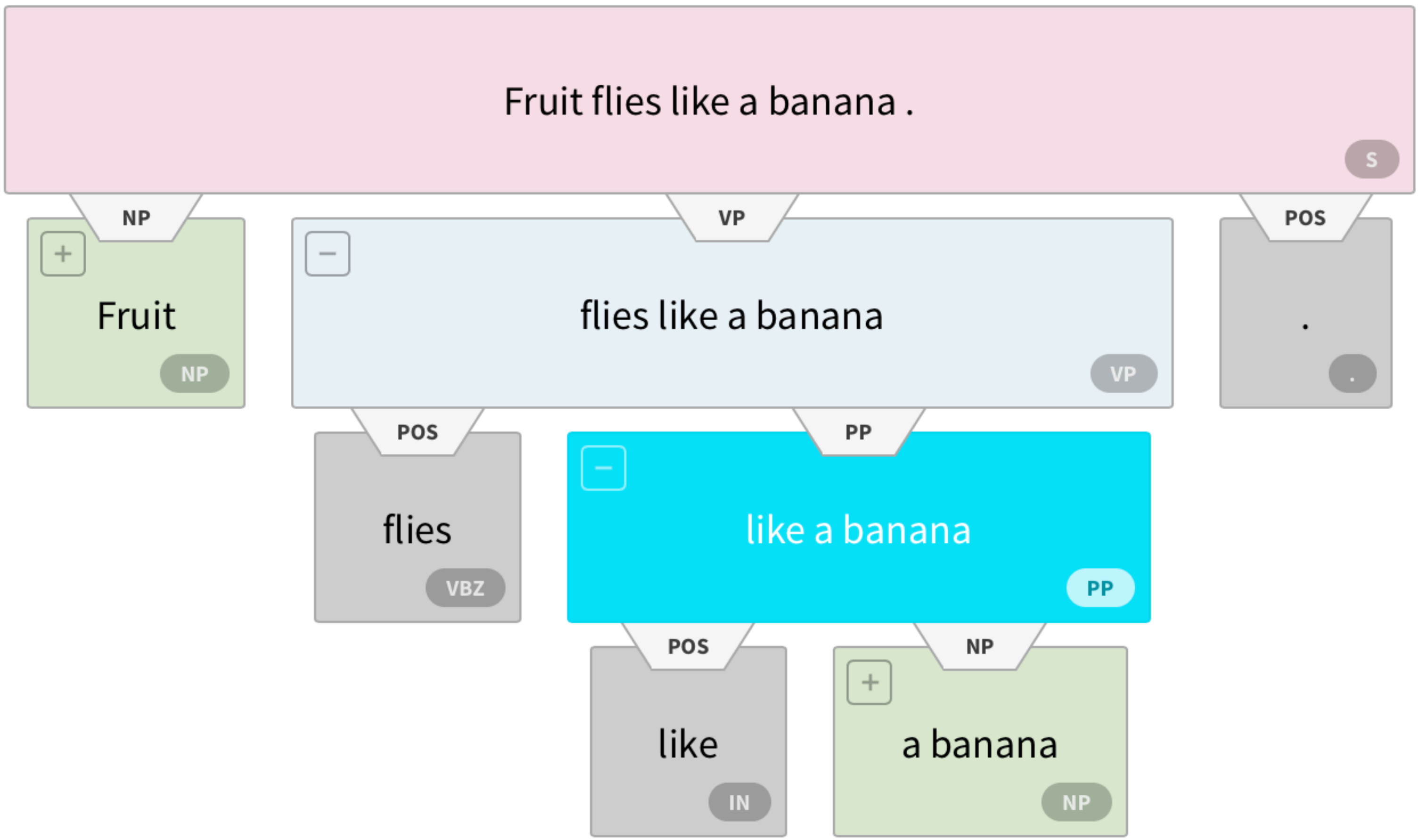
Ambiguity: Which parse?

Fruit flies like a banana.



Ambiguity: Which parse?

Fruit flies like a banana.




The flying
banana
Indoor boomerang
FOR REAL!
1.
2.
3.
±5-30°
4.
RIGHT HAND
Tips & tricks: www.flyingbanana.be

Formal Definition of a CFG

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

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

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

Σ is set of **terminal symbols**

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

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

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

S is the start symbol (a non-terminal)

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

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

or

$$P(RHS \mid LHS)$$

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

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

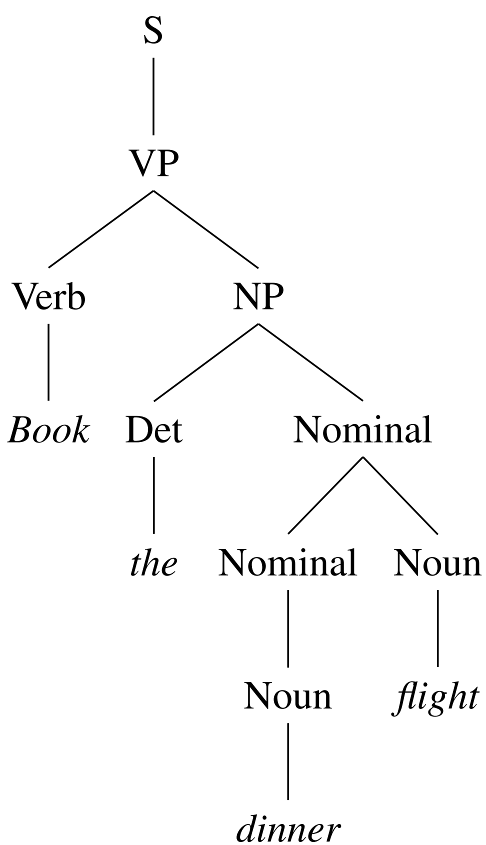
Probabilistic CFGs

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.

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

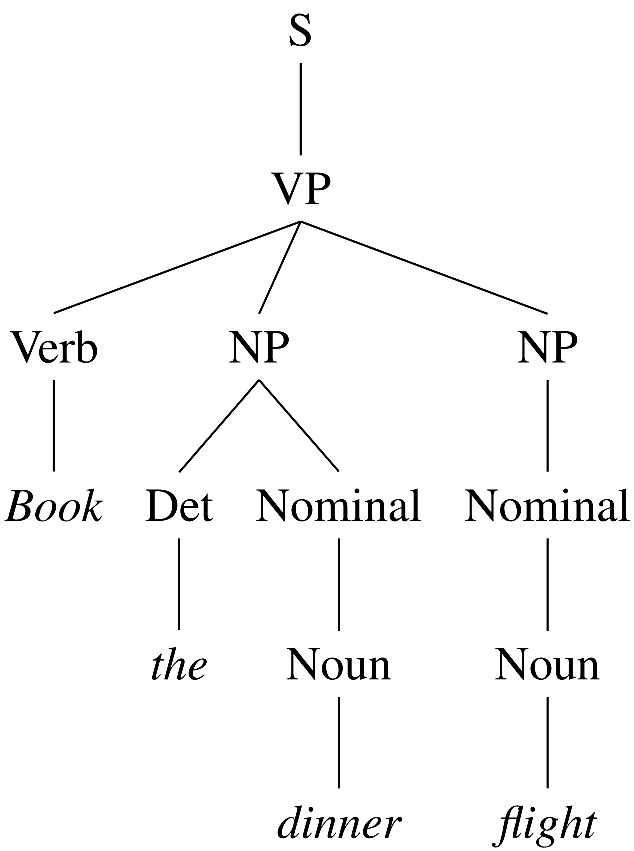
Probability of a parse



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

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

Probability of a parse



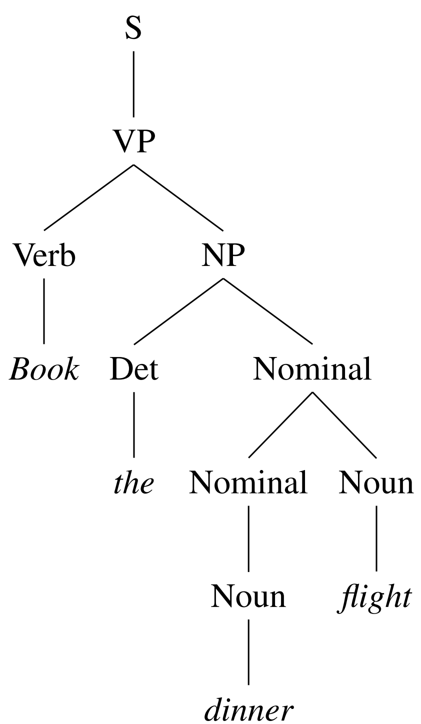
Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flight	.40

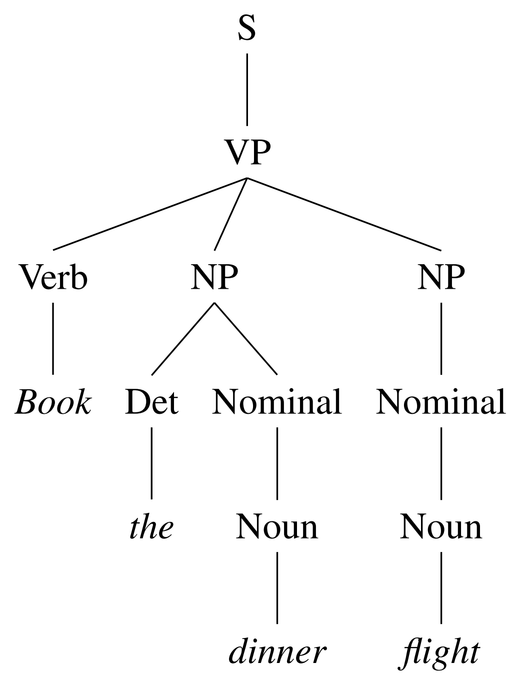
$$P(T,S) = \prod_{i=1}^{n} P(RHS_i|LHS_i) \quad = \quad .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

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*.

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





Joint Probability of T and S

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

By definition of joint probability:

$$P(T,S) = P(T)P(S|T)$$

But since a parse tree includes all the words of the sentence, $P(S|T)$ is 1

$$P(T,S) = P(T)P(S|T) = P(T)$$

Estimating the probabilities

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

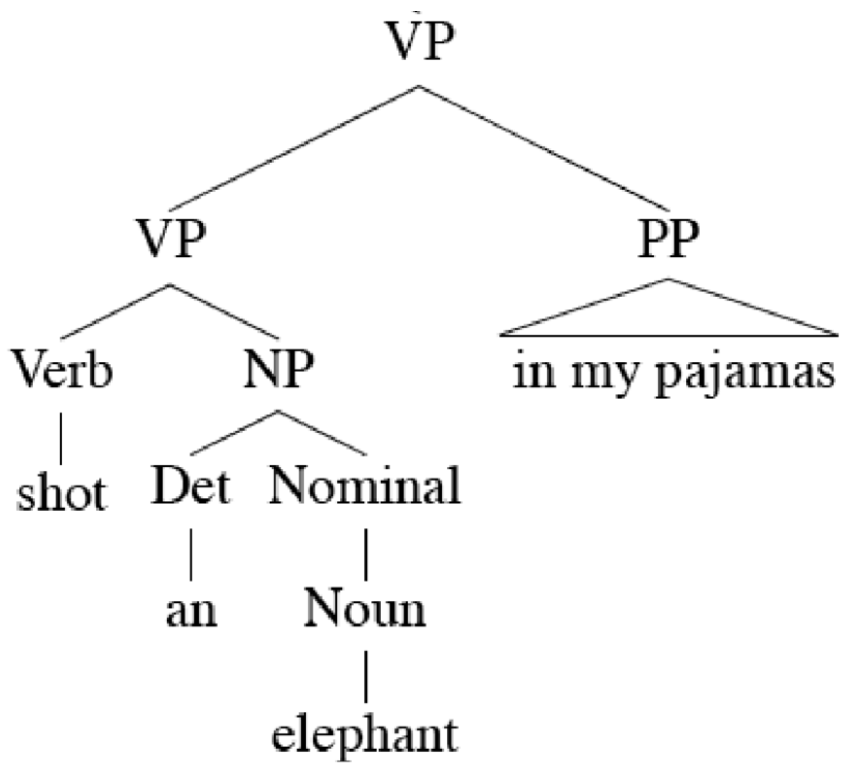
Problem 1

Poor independence assumptions: CFG rules impose an independence assumption on probabilities that leads to poor modeling of structural dependencies across the parse tree.

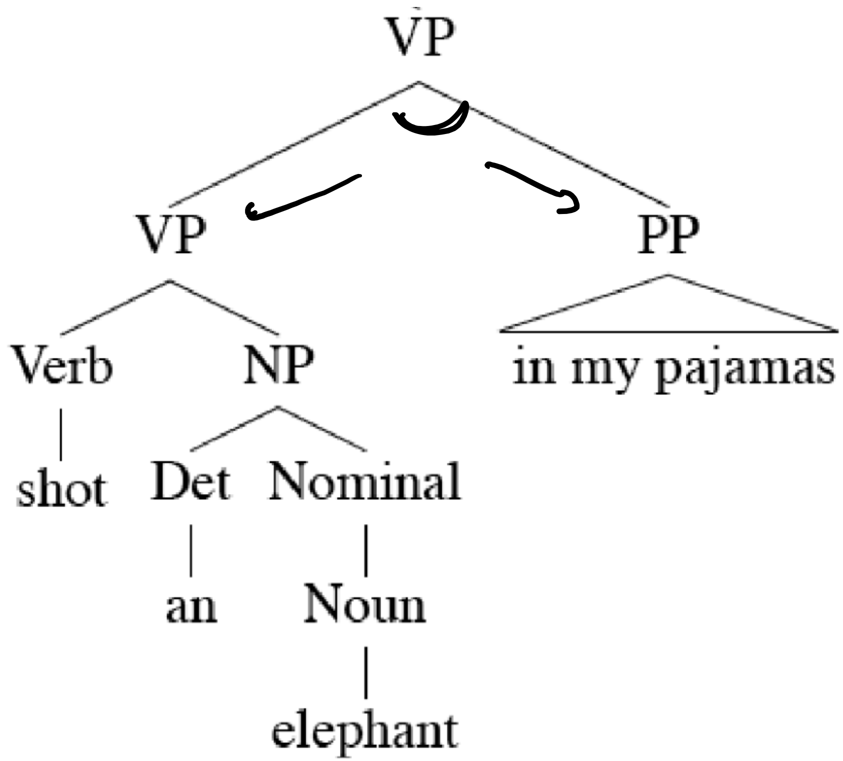
Problem 2

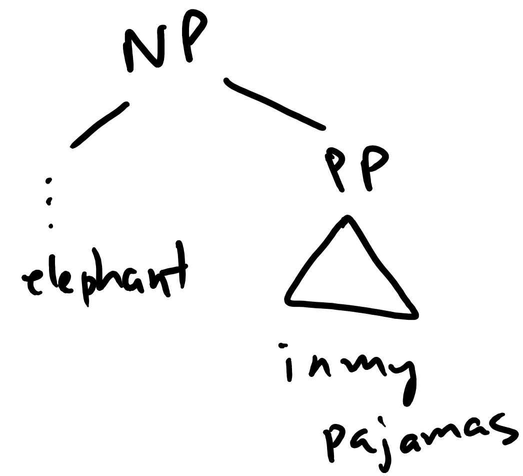
Lack of lexical conditioning: CFG rules don't model syntactic facts about specific words, leading to problems with subcategorization ambiguities, preposition attachment, and coordinate structure ambiguities.

Lexicalized PCFGs

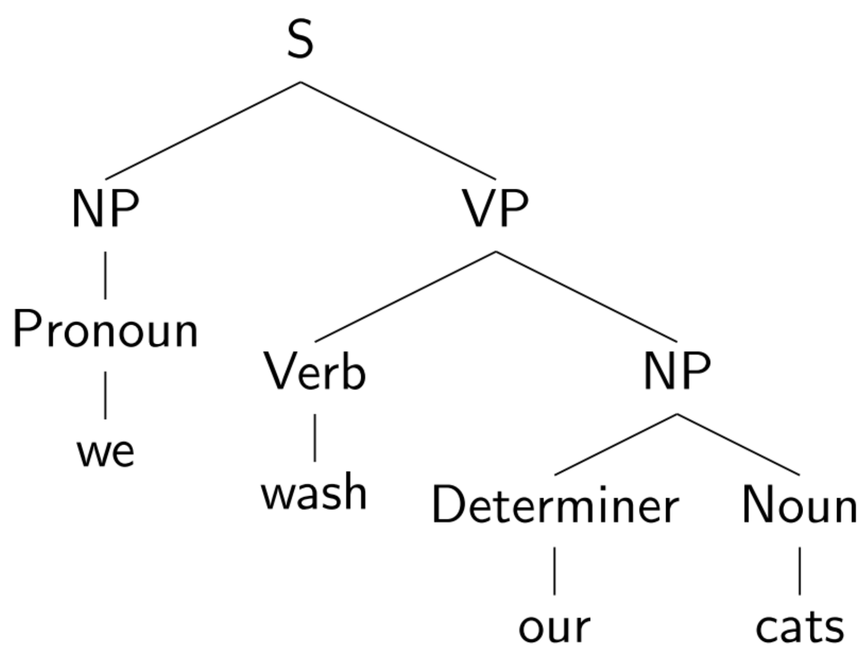


Lexicalized PCFGs

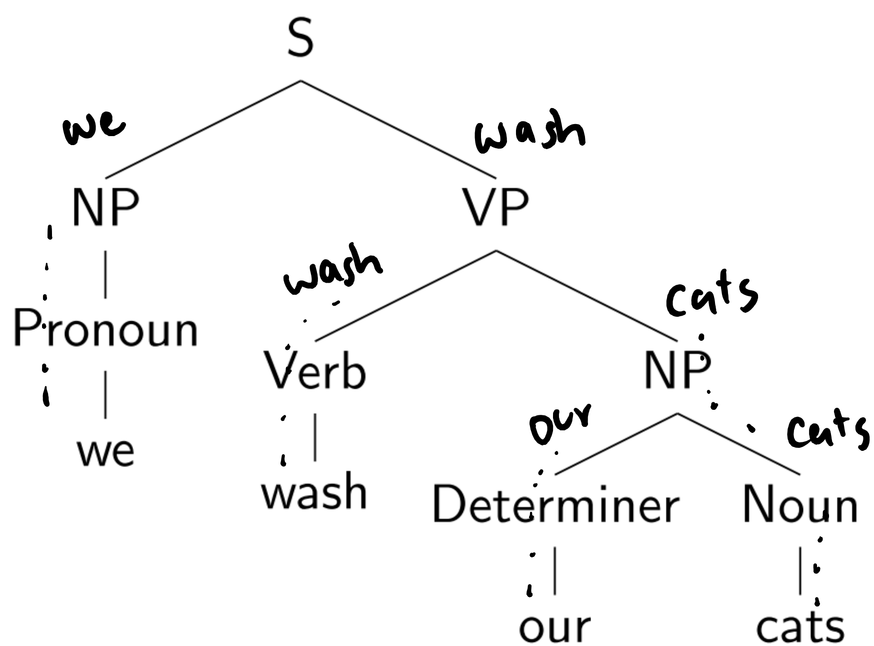




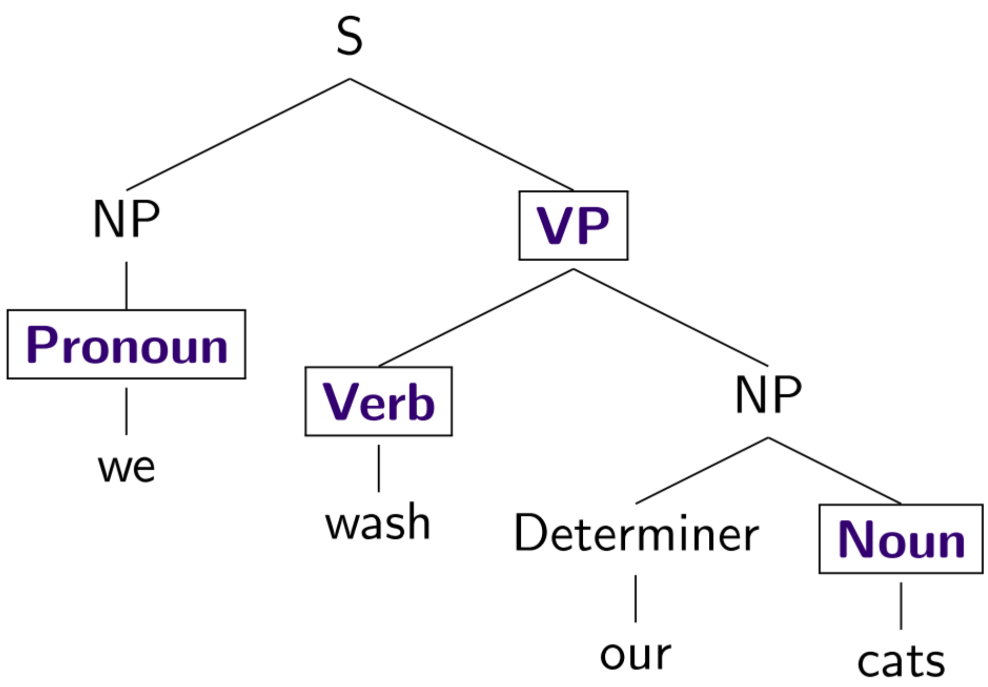
Lexicalizing a CFG



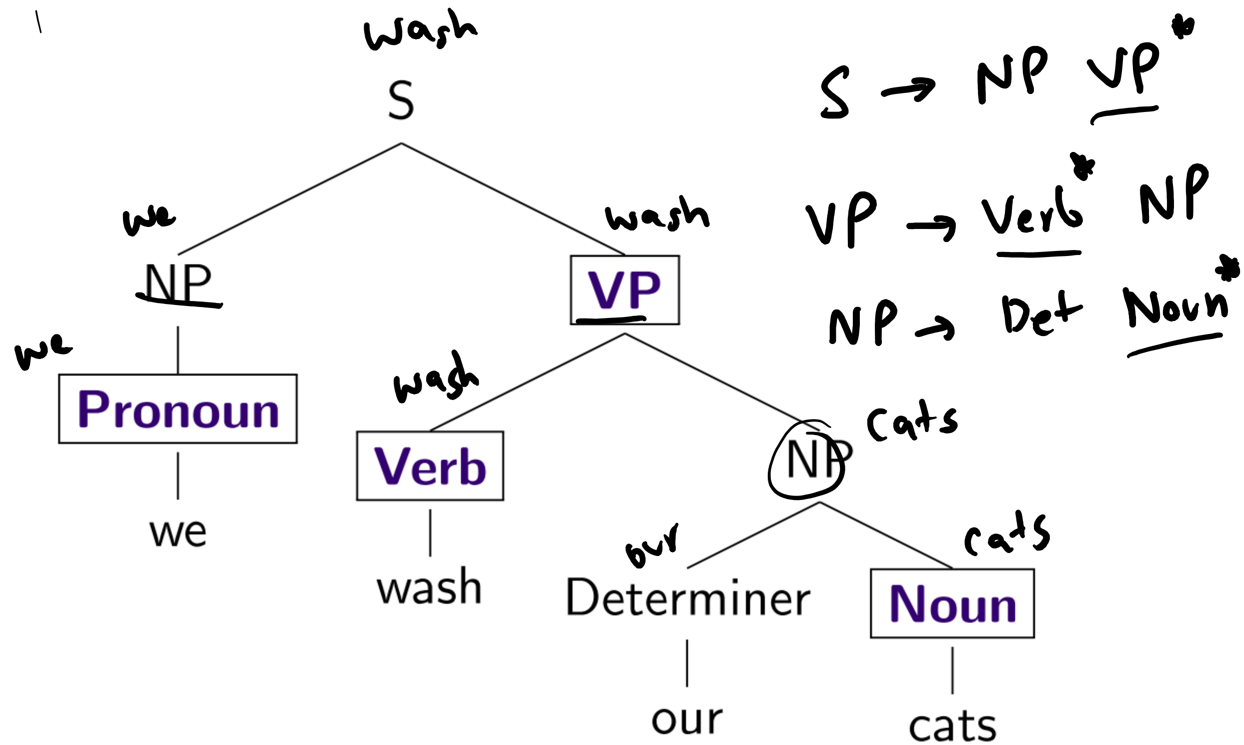
Lexicalizing a CFG



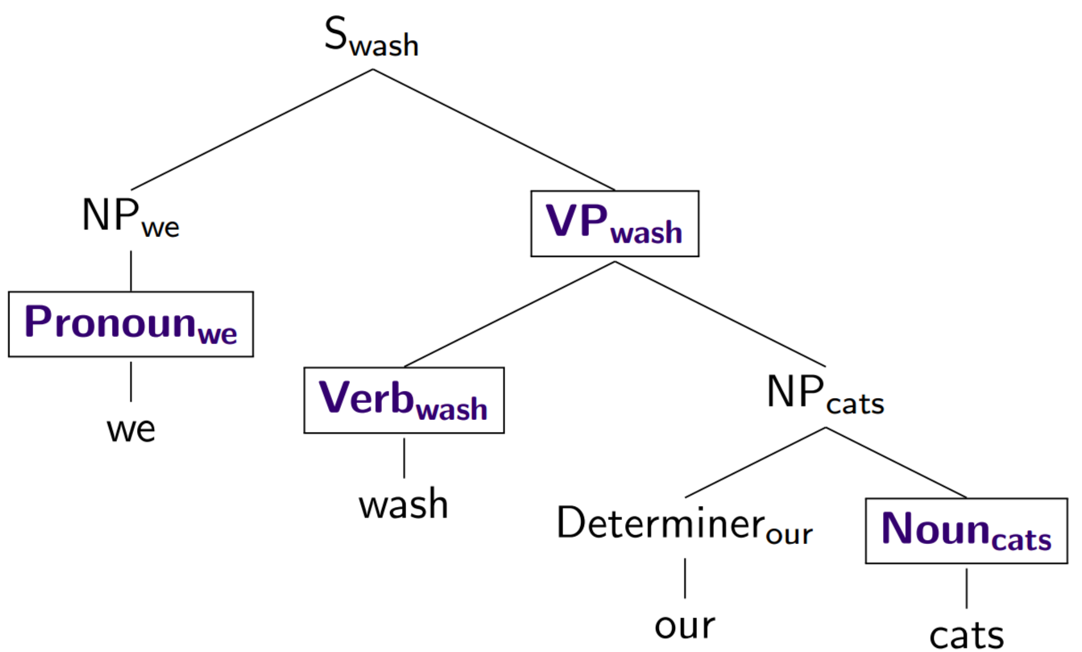
Lexicalizing a CFG



Lexicalizing a CFG



Lexicalizing a CFG



The Parsing Problem

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

Recognition

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

"Proof" is a deduction, valid parse tree.

Parsing

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

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

Even with small grammars, grows exponentially!

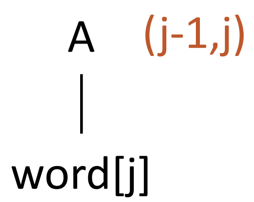
Probabilistic CKY Algorithm

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

Base case

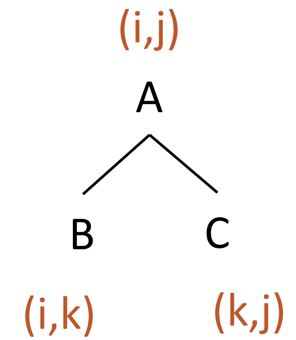
Rule: P($A \rightarrow word[j]$)

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



Recursion

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



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

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

Outline

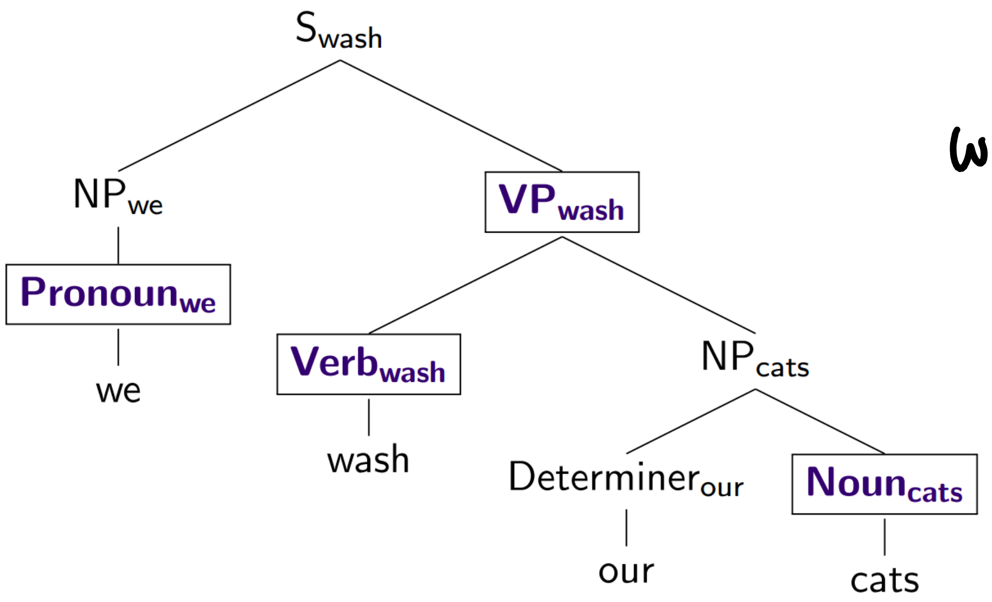
Parsing: CKY Algorithm

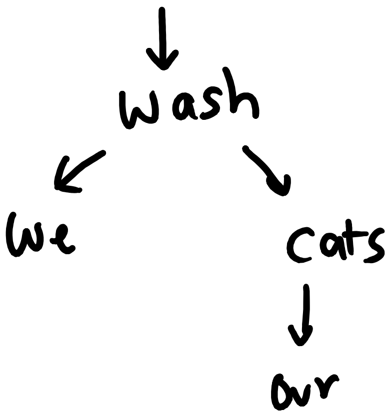
Extensions: Probabilistic and Lexicalized

Dependency Parsing

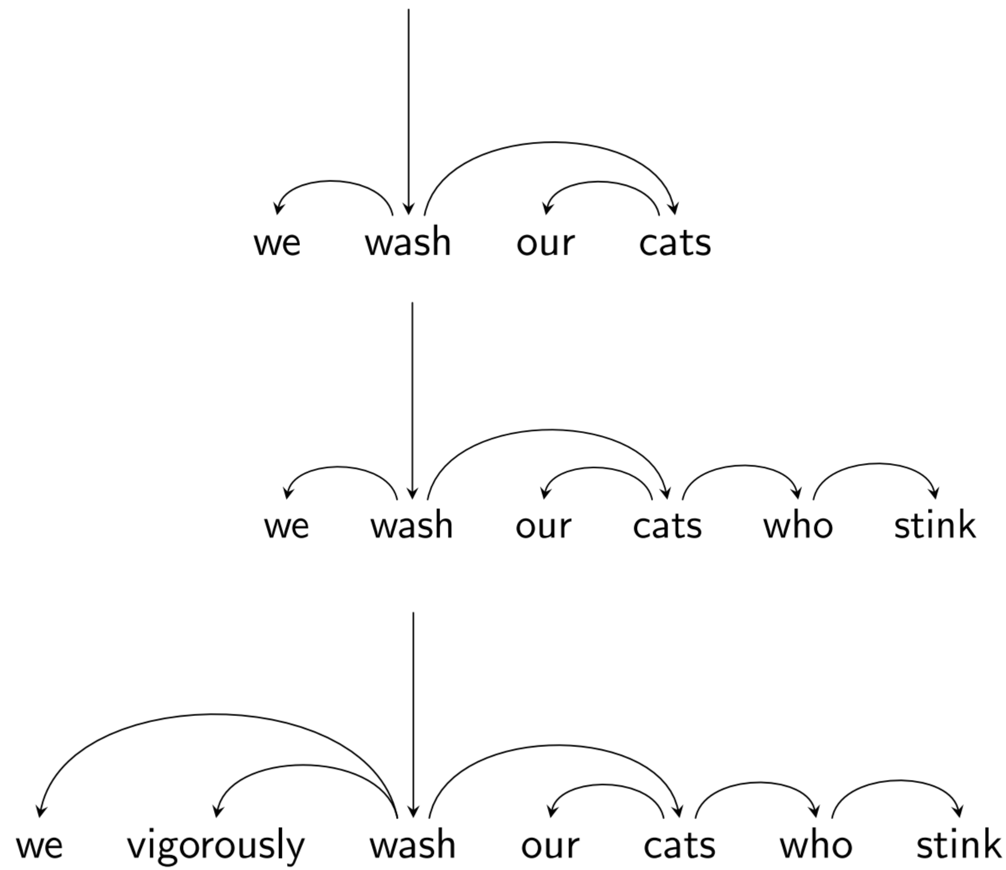
Dependencies

Represent only the syntactic dependencies…

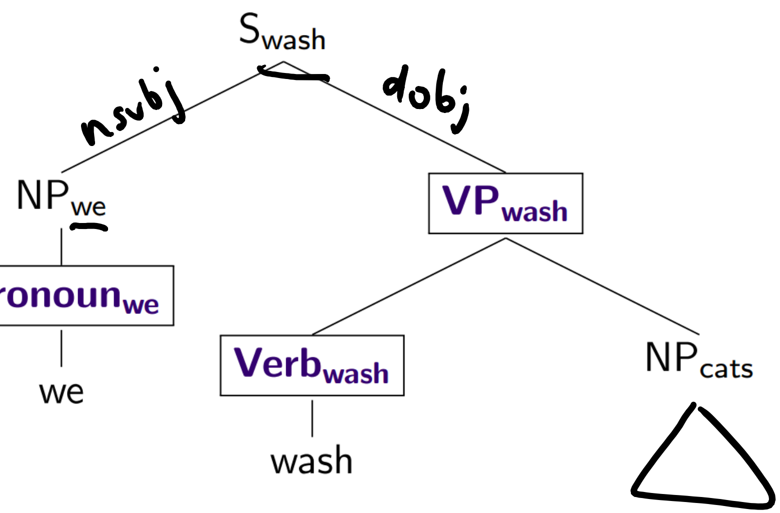


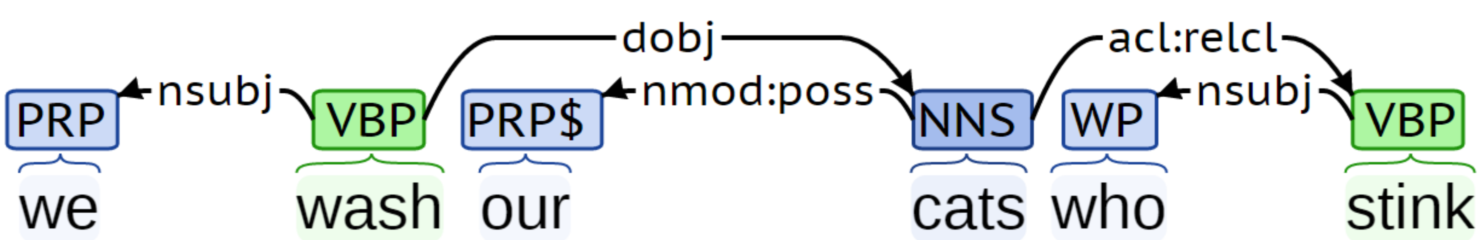


Nested Structure = Subtrees



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

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

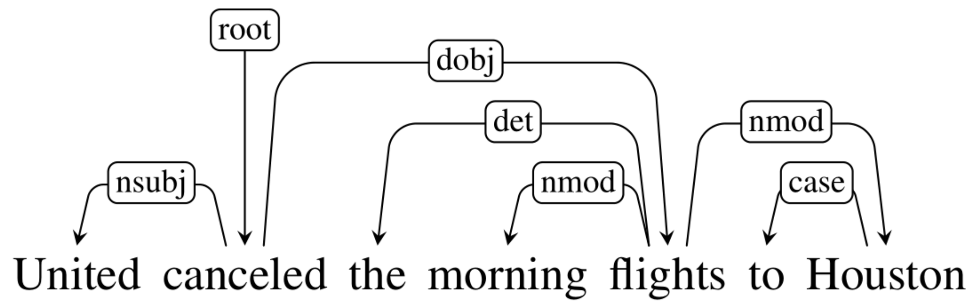
c_i: 1 : n

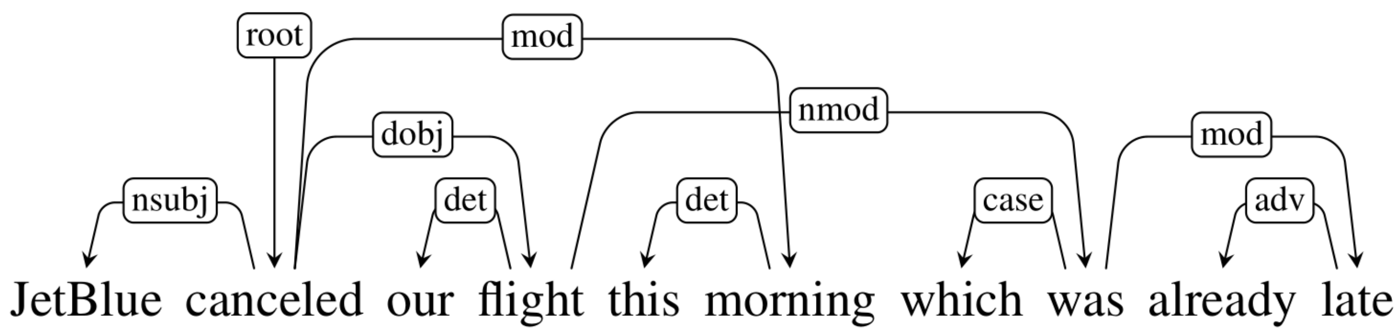
p_i: 0 : n (0 ↑ root)

l_i: label

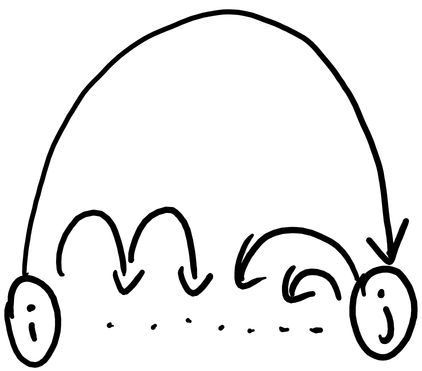
- single headed $\exists\, i$ only one $p_i = 0$
- connected
- acyclic
- projective vs non-projective

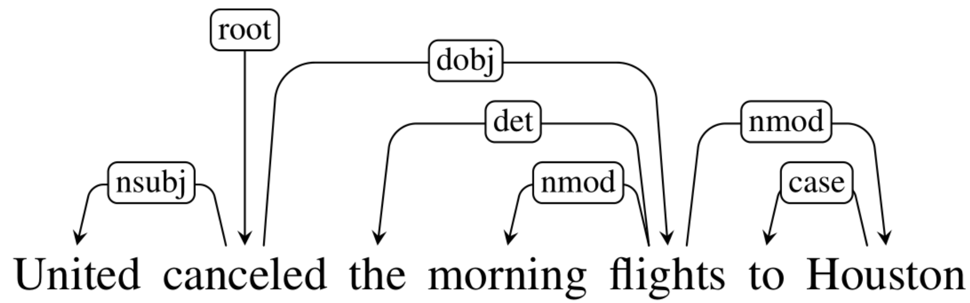
Projective vs Non-projective

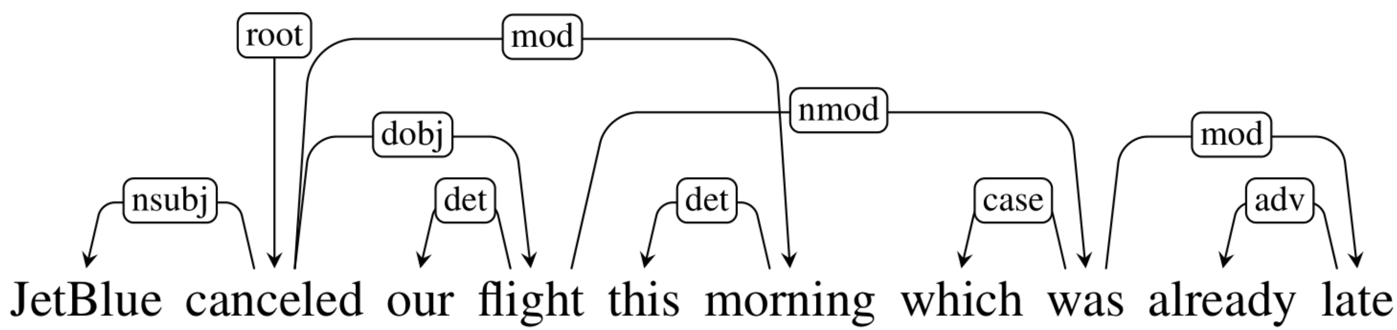




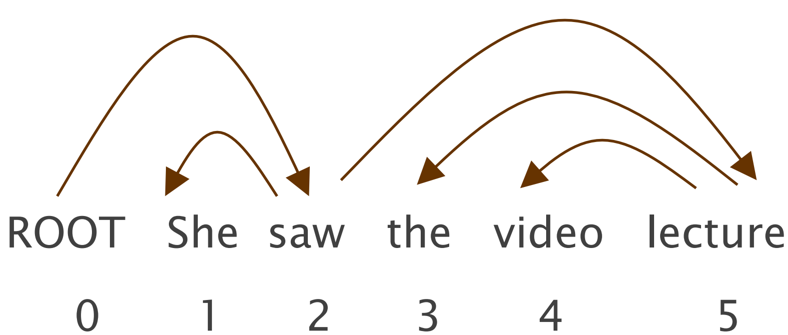
Projective vs Non-projective

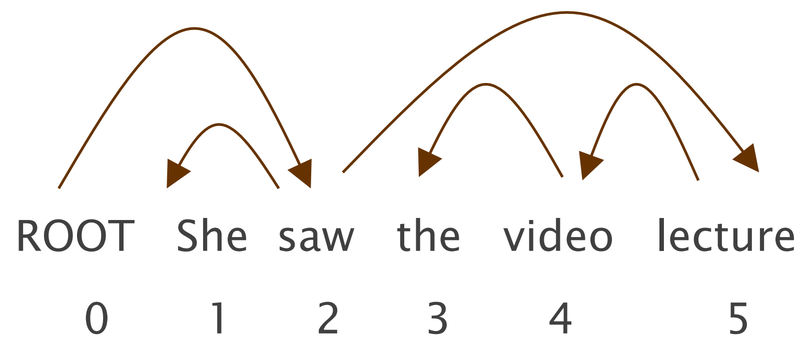






Evaluating Dependency Parses





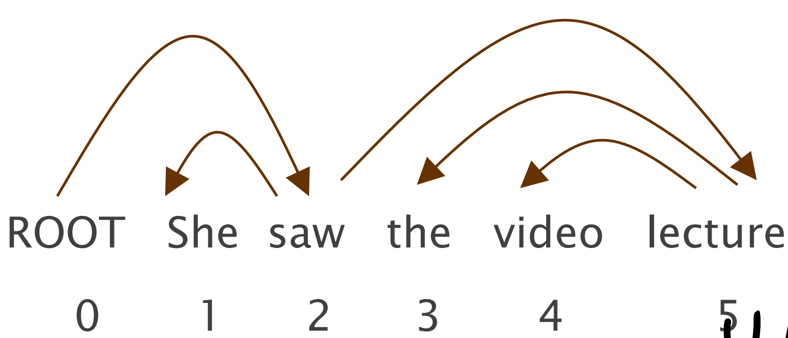
Gold

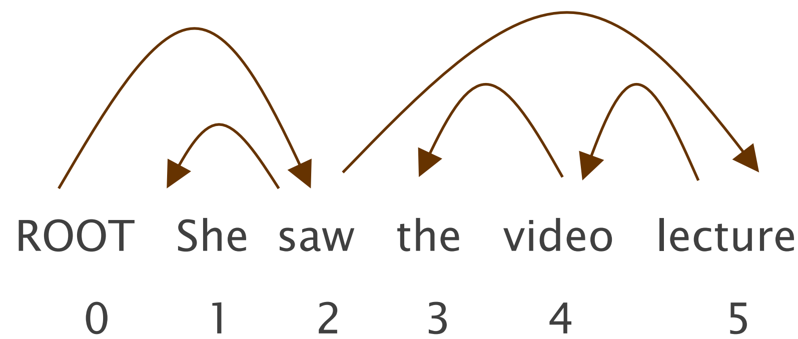
1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	dobj

Parsed

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

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

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	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