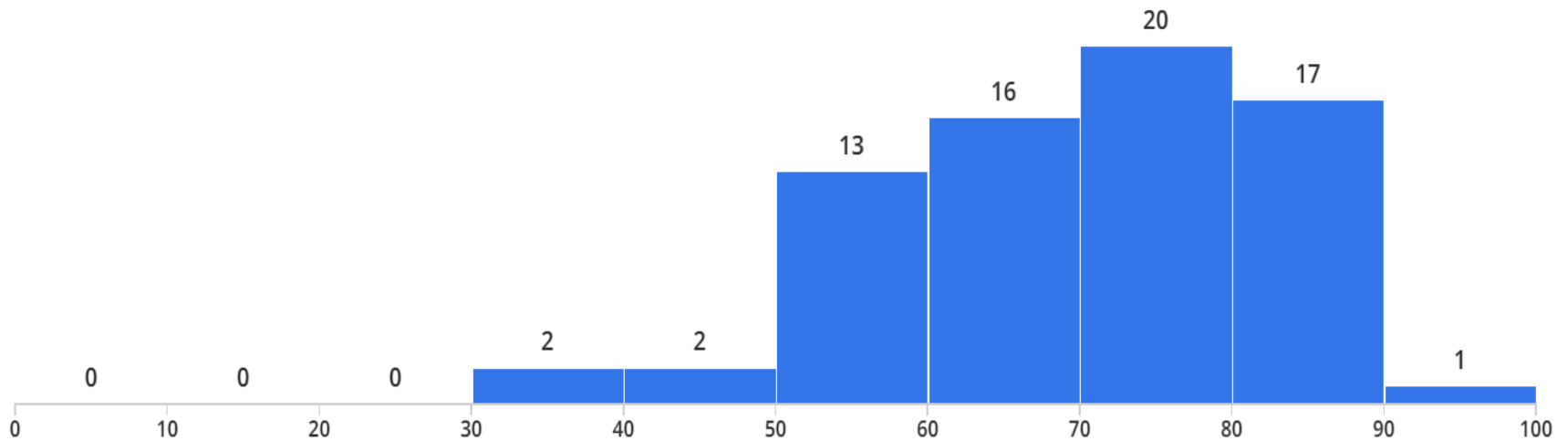


Midterm



Minimum

36.0

Median

71.5

Maximum

91.0

Mean

69.82

Std Dev [?](#)

12.27



Announcement

- ▶ Homework 4
 - ▶ Available in Blackboard -> Homework
 - ▶ Due: Apr. 11, 11:59pm





Overview of Syntactic Parsing



Syntax

- ▶ Syntax studies rules and processes that govern the **structure** of sentences
- ▶ Syntax is only about structure, not about meaning
 - ▶ A sentence can be syntactically well-formed but semantically ill-formed
 - ▶ Colorless green ideas sleep furiously.
 - ▶ Two semantically identical sentences can have different syntactic structures
 - ▶ A dog is chasing a cat.
 - ▶ A cat is being chased by a dog.



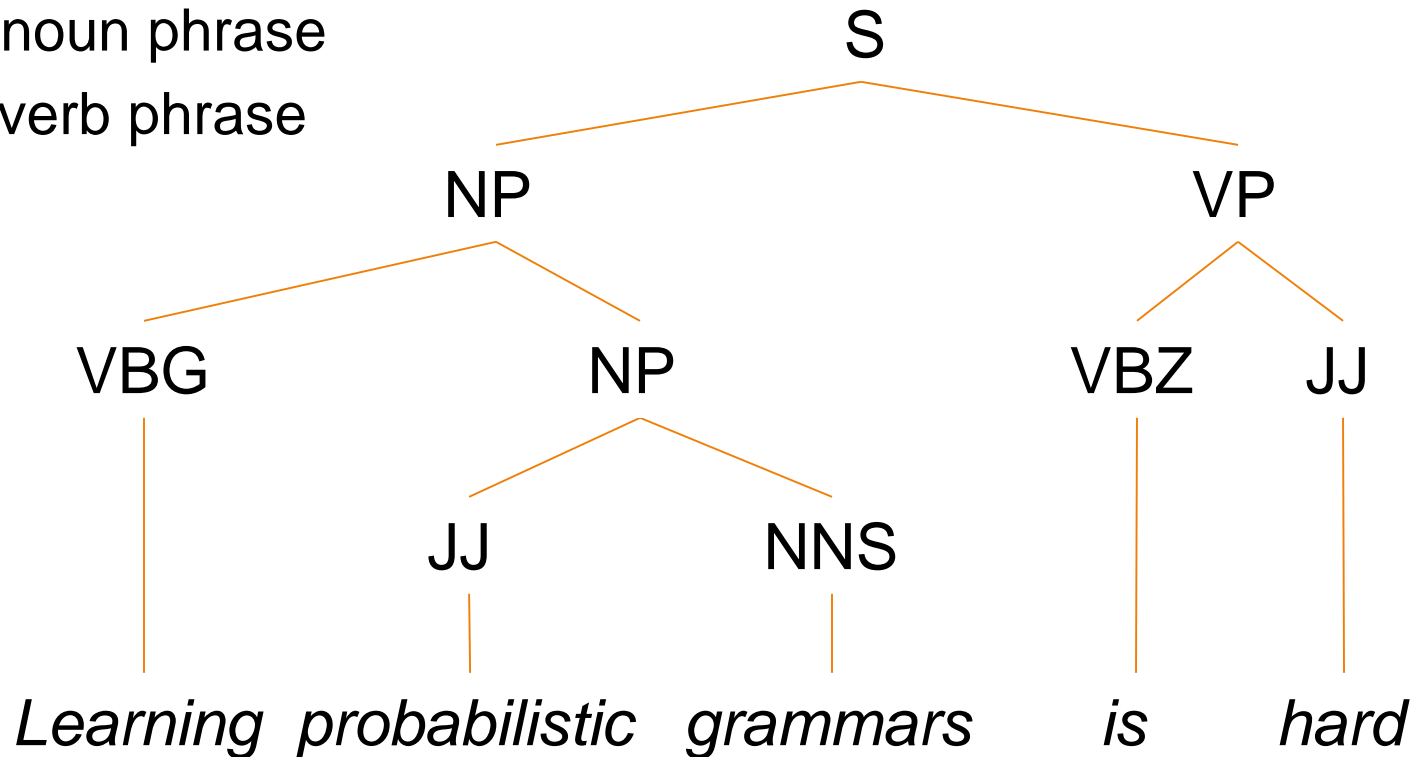
Syntax Research

- ▶ Before DL revolution
 - ▶ A key component of NLP research!
- ▶ After DL revolution
 - ▶ Diminishing importance
 - ▶ Modern neural methods do not explicitly model syntax
- ▶ Why we still study syntax
 - ▶ Perhaps the most elegant part of NLP
 - ▶ Still useful in some scenarios
 - ▶ Its techniques may be extended to solve other problems
 - ▶ The future:
 - ▶ Syntactic structures are an intrinsic property of languages
 - ▶ More explicitly utilizing them should be beneficial



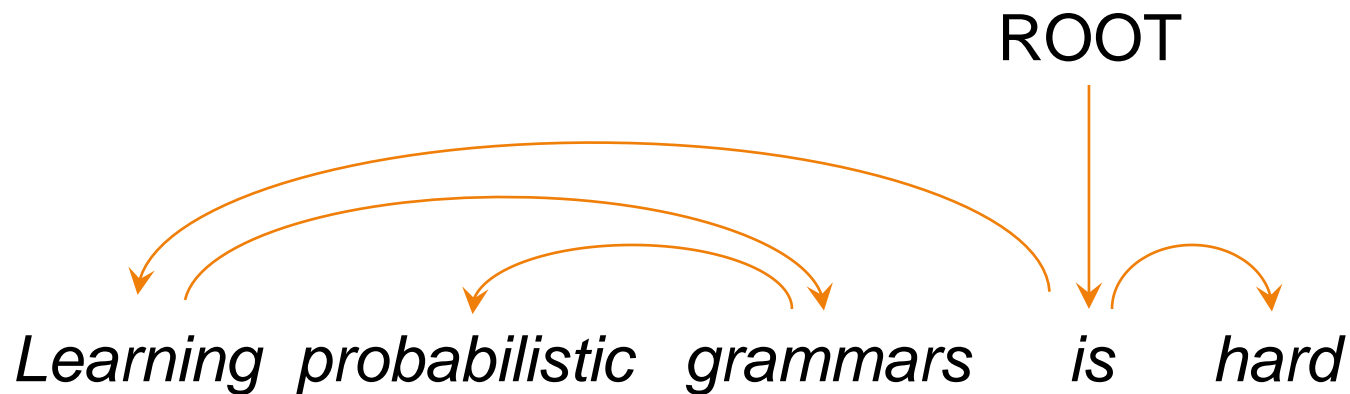
Constituent parse tree

- ▶ Also called a phrase structure parse
- ▶ Each non-leaf node represents a phrase
 - ▶ S: sentence
 - ▶ NP: noun phrase
 - ▶ VP: verb phrase
 - ▶ ...



Dependency parse tree

- ▶ Each arc represents a binary dependency relation between two words
 - ▶ Relations may be typed (labeled)
 - ▶ Ex. Subject relation between a verb and a noun



Parse tree scoring

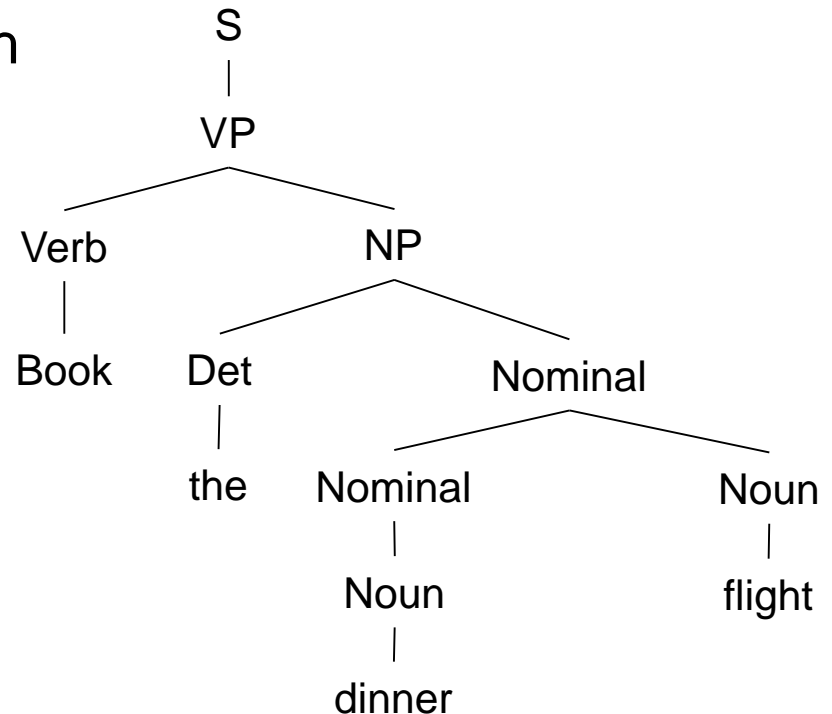
- ▶ Goal: assign a probability or score to each parse tree of a sentence
- ▶ Why?
 - ▶ Disambiguation!
 - ▶ A natural language sentence may have many possible parses
- ▶ Common approach:
 - ▶ Decompose a parse tree into many parts
 - ▶ Score each part
 - ▶ Take the sum or product



Parsing

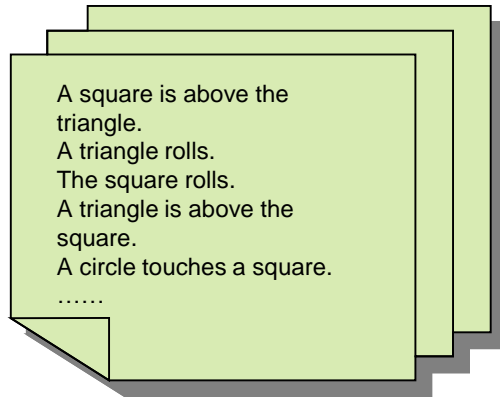
- ▶ Goal: given a sentence, find the parse tree with the highest score or probability
- ▶ Common approaches:
 - ▶ Dynamic programming
 - ▶ Greedy search / beam search

Book the dinner flight

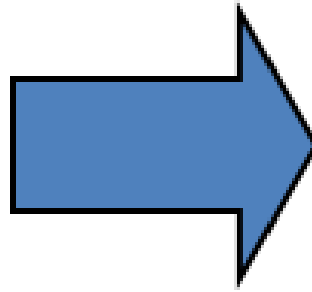


Learning

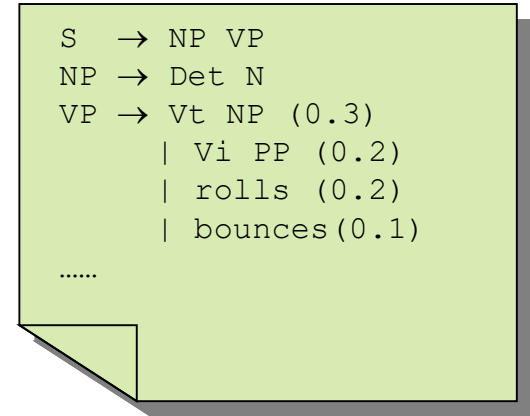
Training Corpus



Learning



Grammar / Parser



- ▶ Supervised Methods
 - ▶ Rely on a training corpus of sentences annotated with parses (treebank)
- ▶ Unsupervised Methods (Grammar Induction)
 - ▶ Do not require annotated data



Constituency Parsing

SLP3 Ch 12, 13; INLP Ch 9, 10

Constituency

▶ Constituents

- ▶ Groups of words within sentences can be shown to act as single units.
- ▶ Ex: (The fox)(jumps(over(the dog)))

▶ These units form coherent classes

- ▶ Units in the same class behave in similar ways
 - ▶ ...with respect to their *internal* structure
 - ▶ ...and with respect to other (*external*) units in the language
- ▶ E.g., noun phrases



Constituency

- ▶ For example, it makes sense to say that the following are all *noun phrases* in English...

Harry the Horse
the Broadway coppers
they

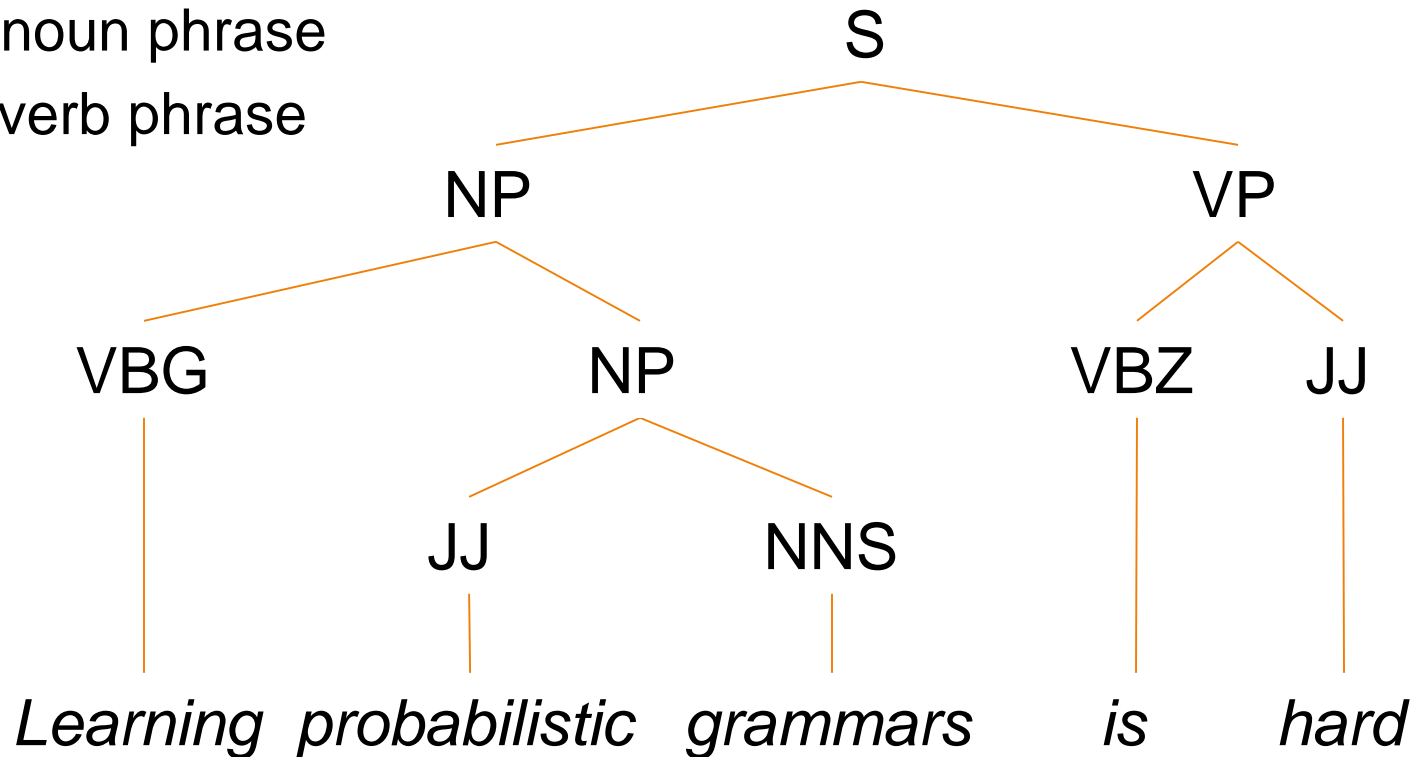
a high-class spot such as Mindy's
the reason he comes into the Hot Box
three parties from Brooklyn

- ▶ Why?
 - ▶ Similar internal structures
 - ▶ e.g., determiner + modifier + noun + modifier
 - ▶ They can all precede verbs (external evidence)



Constituent parse tree

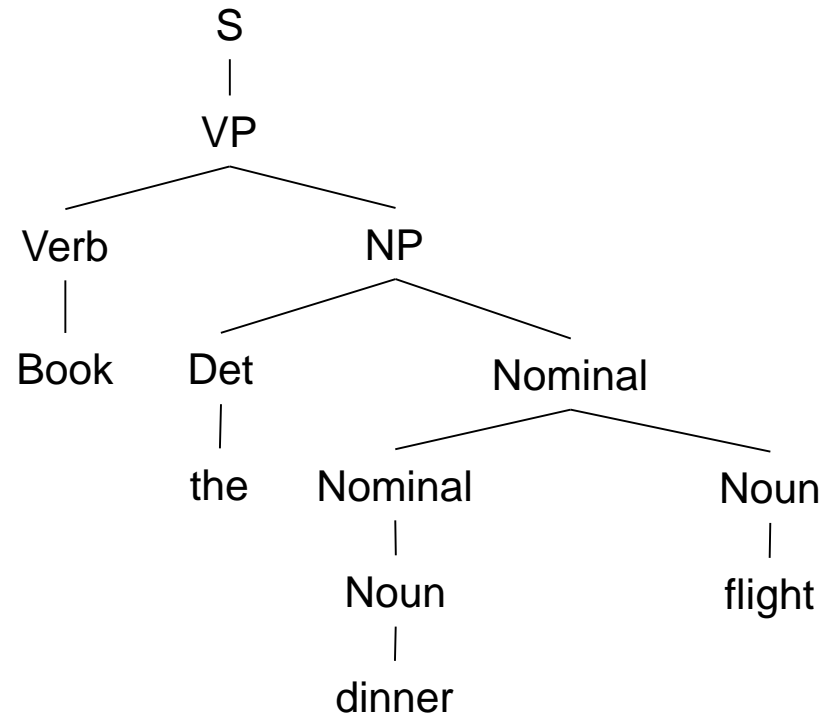
- ▶ Also called a phrase structure parse
- ▶ Each non-leaf node represents a phrase
 - ▶ S: sentence
 - ▶ NP: noun phrase
 - ▶ VP: verb phrase
 - ▶ ...



Constituency Parsing

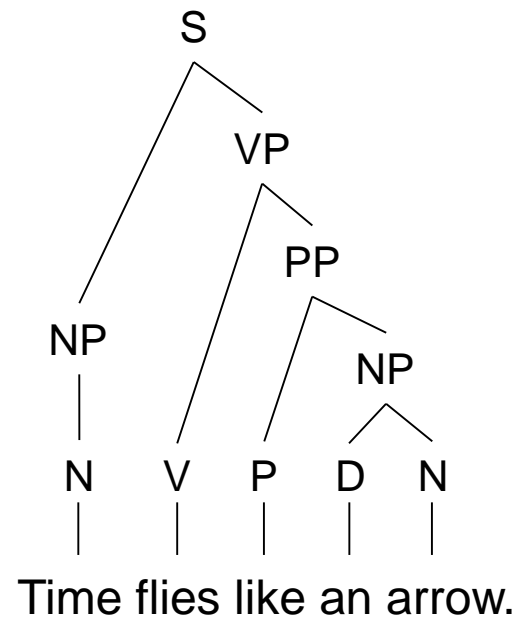
- ▶ The task of assigning proper constituent parse trees to input strings

Book the dinner flight



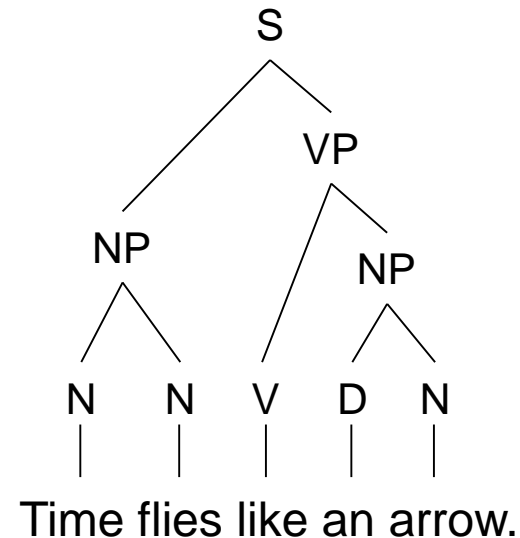
Ambiguity in Constituency Parsing

- ▶ A sentence is ambiguous if it has more than one possible parse tree
 - ▶ ...and hence more than one interpretation
- ▶ Examples
 - ▶ Time flies like an arrow.



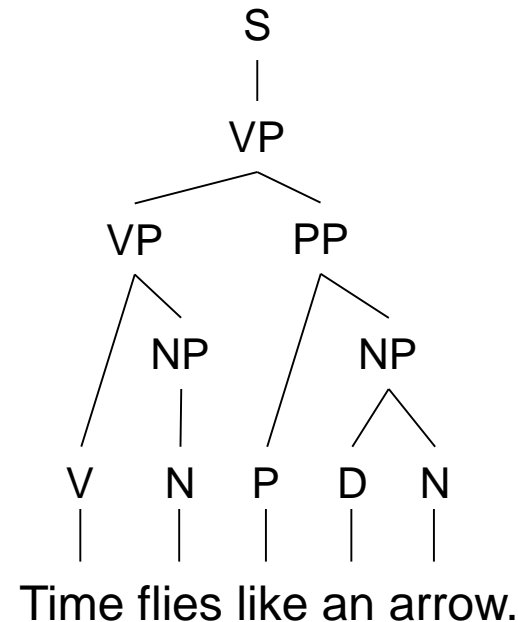
Ambiguity in Constituency Parsing

- ▶ A sentence is ambiguous if it has more than one possible parse tree
 - ▶ ...and hence more than one interpretation
- ▶ Examples
 - ▶ Time flies like an arrow.



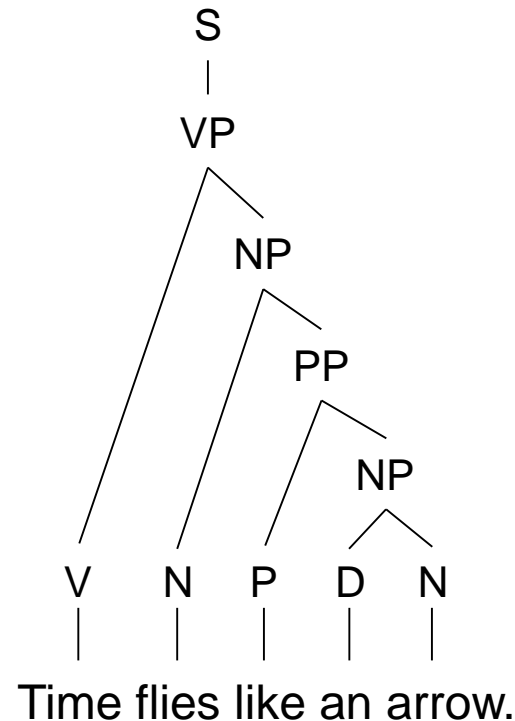
Ambiguity in Constituency Parsing

- ▶ A sentence is ambiguous if it has more than one possible parse tree
 - ▶ ...and hence more than one interpretation
- ▶ Examples
 - ▶ Time flies like an arrow.



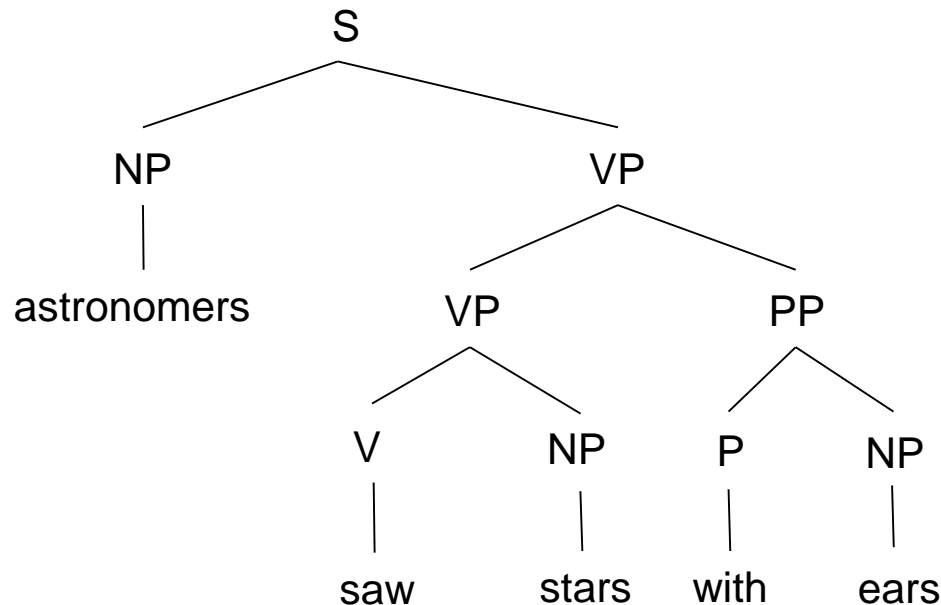
Ambiguity in Constituency Parsing

- ▶ A sentence is ambiguous if it has more than one possible parse tree
 - ▶ ...and hence more than one interpretation
- ▶ Examples
 - ▶ Time flies like an arrow.



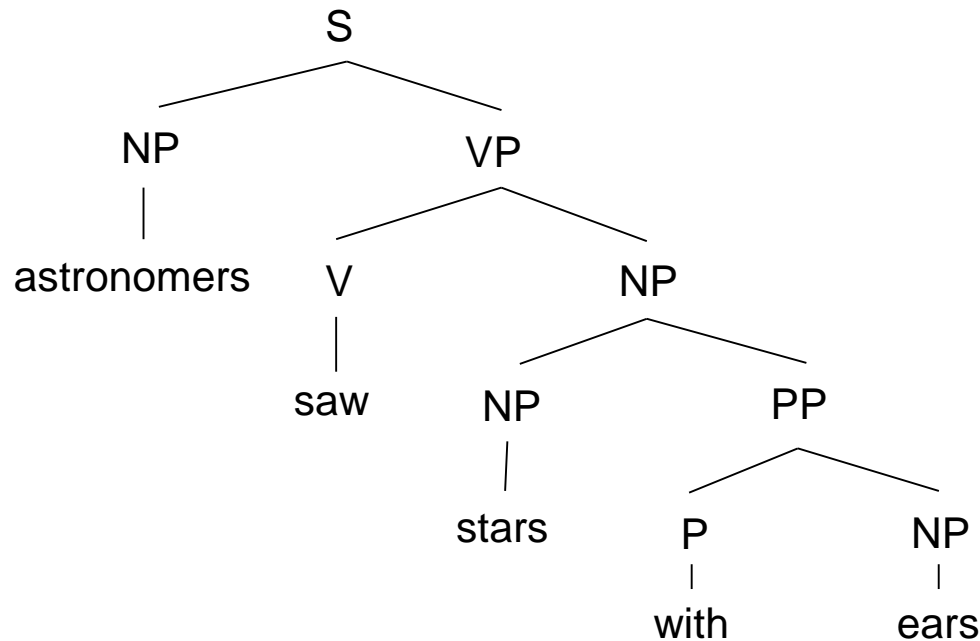
Ambiguity in Constituency Parsing

- ▶ 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.



Ambiguity in Constituency Parsing

- ▶ 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.



Ambiguity in Constituency Parsing

- ▶ A sentence is ambiguous if it has more than one possible parse tree
 - ▶ ...and hence more than one interpretation
- ▶ Disambiguation
 - ▶ Parse tree scoring: assign a probability or score to each parse tree of a sentence
 - ▶ Find the parse tree t of a sentence x with the highest score or probability



Constituency Parsing

- ▶ Find the parse tree t of a sentence x with the highest score or probability
- ▶ Generative parsing

$$\hat{t} = \operatorname{argmax}_{t \in T_x} S(t)$$

- ▶ Discriminative parsing

$$\hat{t} = \operatorname{argmax}_{t \in T_x} S(t|x)$$

- ▶ We take into account the sentence x when scoring the parse tree t



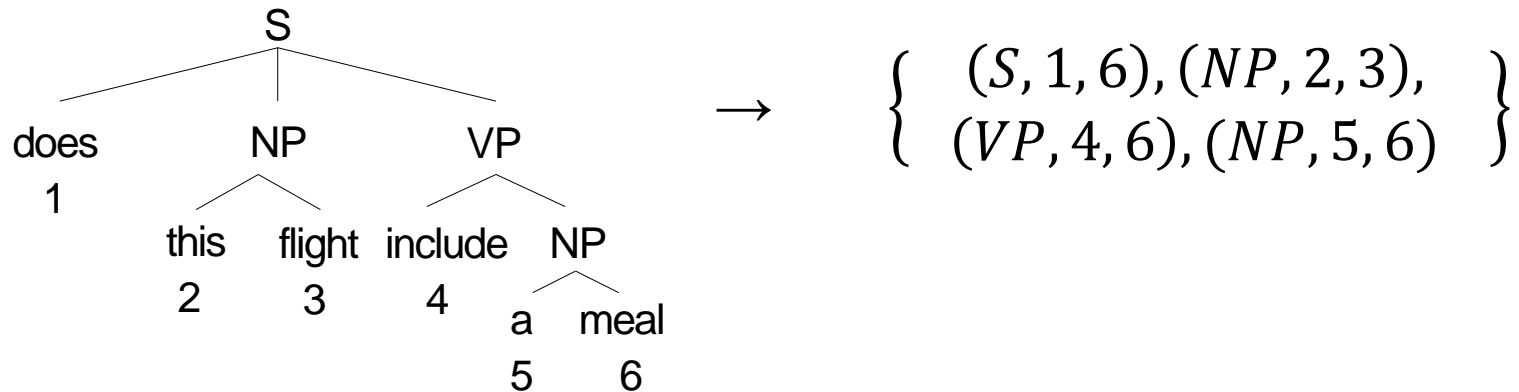
Parsing methods

- ▶ 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)
- ▶ Methods
 - ▶ Dynamic programming
 - ▶ Span-based, grammar-based
 - ▶ Greedy search / beam search
 - ▶ Transition-based



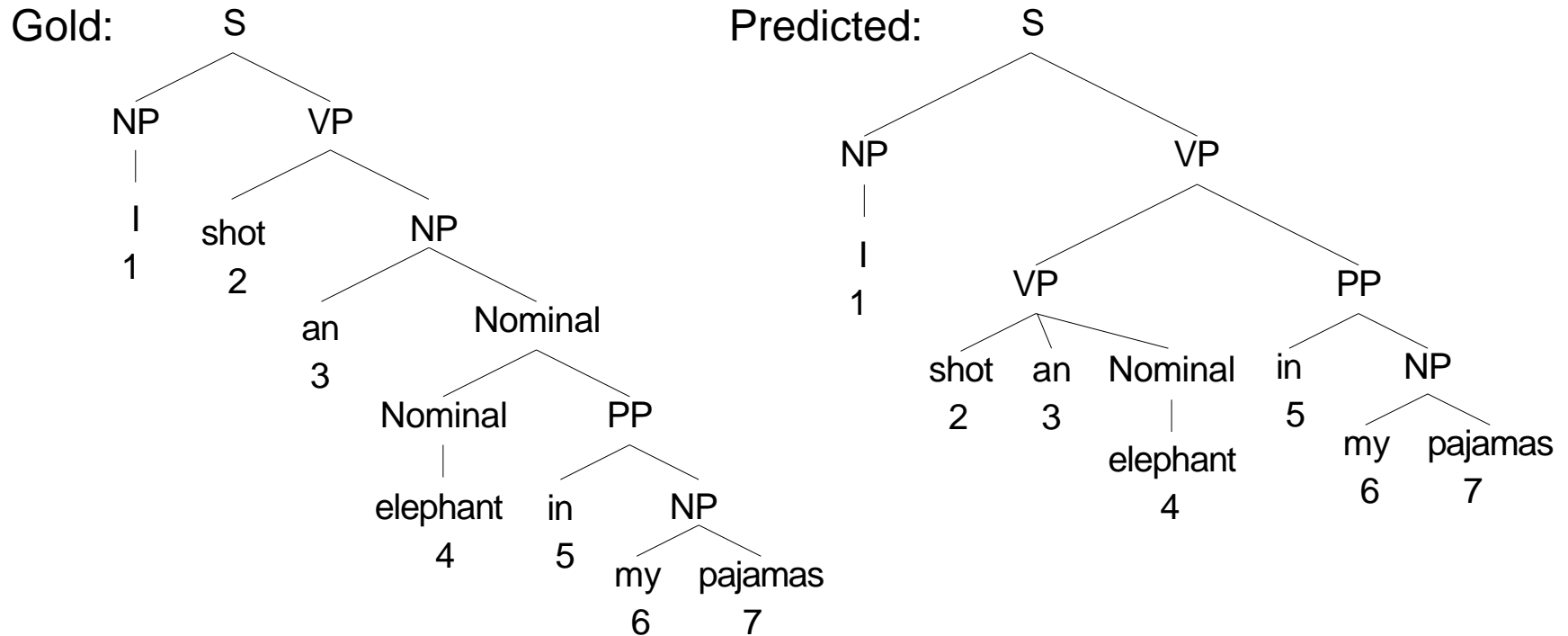
Parser evaluation

- ▶ Represent a parse tree as a collection of tuples:
 $\langle (l_1, i_1, j_1), (l_2, i_2, j_2), \dots, (l_n, i_n, j_n) \rangle$
 - ▶ l_k is the nonterminal labeling the k-th phrase
 - ▶ i_k is the index of the first word in the k-th phrase
 - ▶ j_k is the index of the last word in the k-th phrase



- ▶ Convert gold-standard and predicted trees into this representation, then estimate precision, recall, and F1

Tree Comparison Example



$$\left\{ \begin{array}{l} (NP, 3, 7), \\ (Nominal, 4, 7) \end{array} \right\} \left\{ \begin{array}{l} (NP, 1, 1), (S, 1, 7) \\ (VP, 2, 7), (PP, 5, 7) \\ (NP, 6, 7), (Nominal, 4, 4) \end{array} \right\} \left\{ (VP, 2, 4) \right\}$$

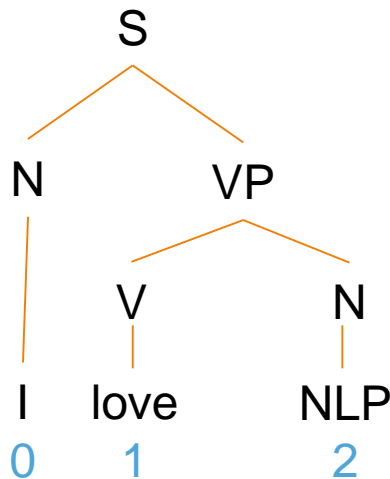
only in gold tree in both tree only in predicted tree

$$\begin{aligned}
 P &= 6/7 \\
 R &= 6/8 \\
 F &= 4/5
 \end{aligned}$$

Span-based Constituency Parsing

Span-based Parsing

- ▶ Tree score = sum of constituent scores
- ▶ Parsing
 - ▶ Score every span of the sentence: $s(i, j, l)$
 - ▶ Find the best parse tree: $\arg \max_t \sum_{(i,j,l) \in t} s(i, j, l)$



$$s(t) = s(0, 0, N) + s(1, 1, V) + s(2, 2, N) \\ + s(1, 2, VP) + s(0, 2, S)$$



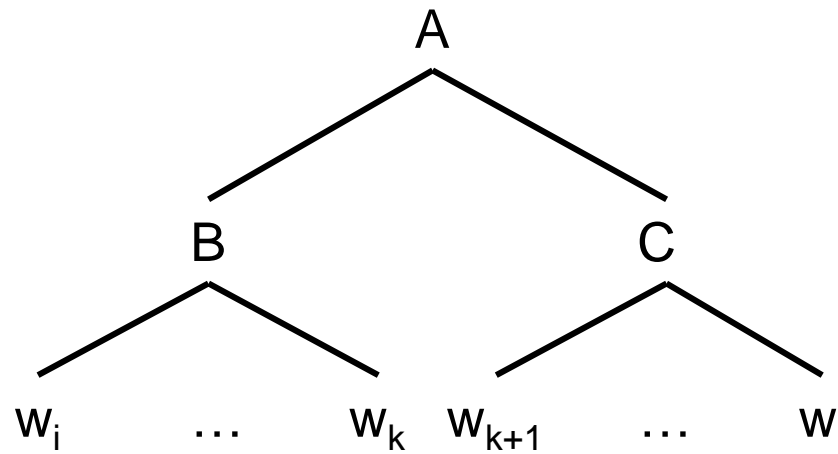
Span Scoring

- ▶ We score every span of the sentence
- ▶ Discriminative parsing
 - ▶ Compute scores conditioned on the input sentence
 - ▶ ...using features of the span
 - ▶ ...or using a neural network with word embeddings of the span as inputs
 - ▶ Ex: $s(i, j, l) = \text{Biaffine}_l(\mathbf{r}_i, \mathbf{r}_j) = [\mathbf{r}_i; 1]^T \mathbf{W}_l [\mathbf{r}_j; 1]$
 - ▶ \mathbf{r}_i is embedding of the i -th word
 - ▶ \mathbf{W}_l is a learnable matrix



Parsing

- ▶ Find the best parse tree: $\arg \max_t \sum_{(i,j,l) \in t} s(i,j,l)$
- ▶ 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



Parsing algorithm

- ▶ Bottom-up dynamic programming: CYK
 - ▶ Also known as CKY
 - ▶ Applies to **binary** parsing
 - ▶ Recursively compute $s_{best}(i, j)$, score of the best parse tree over span (i,j)
- ▶ Proprocessing
 - ▶ $s(i, j) = \max_l s(i, j, l)$
 - ▶ Why?



CYK

- ▶ Base case:

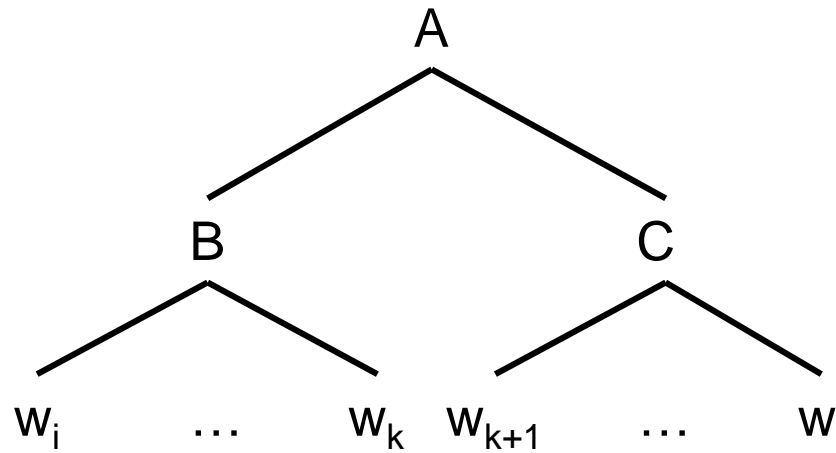
- ▶ Spans of a single token

$$s_{best}(i, i) = s(i, i)$$

- ▶ Recursion:

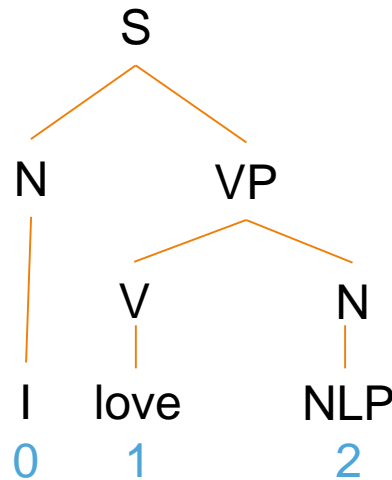
$$s_{best}(i, j) = s(i, j) + \max_k [s_{best}(i, k) + s_{best}(k + 1, j)]$$

- ▶ k: split point



CYK: Example

- ▶ Consider a simple example
 - ▶ “I love NLP”
- ▶ The gold constituency tree looks like:



CYK: Example

- ▶ Suppose we have all the span scores
- ▶ I love NLP

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



CYK: Example

► I love NLP

$$s_{best}(0,1) = s(0,1) + s_{best}(0,0) + s_{best}(1,1) = 1 + 3 + 3 = 7$$

	0	1	2
0	N 3	NP 1+3+3 =7	S 2
1		V 3	VP 2
2			N 5

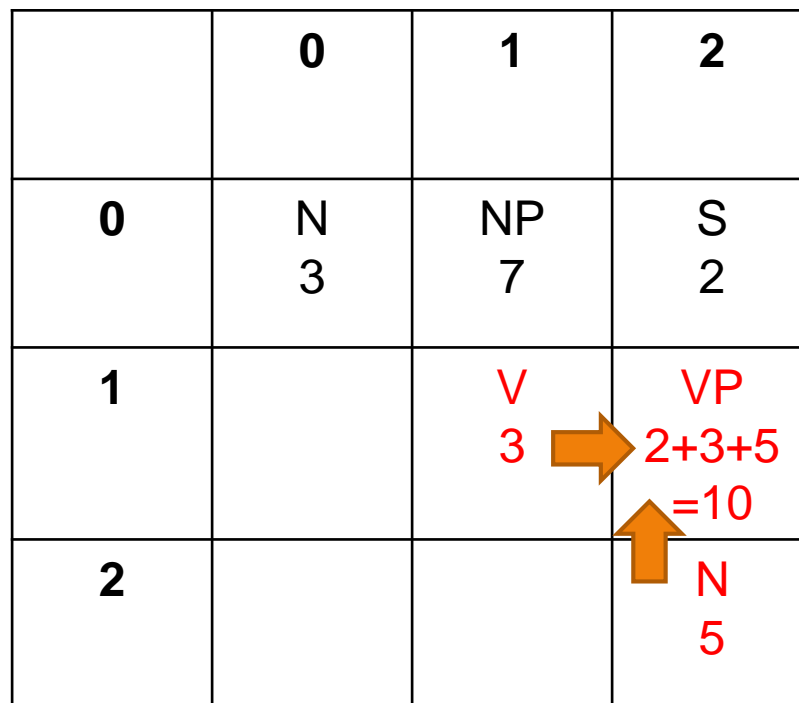


CYK: Example

► I love NLP

$$s_{best}(1,2) = s(1,2) + s_{best}(1,1) + s_{best}(2,2) = 2 + 3 + 5 = 10$$

	0	1	2
0	N 3	NP 7	S 2
1		V 3	VP 2+3+5 =10
2			N 5



CYK: Example

► I love NLP

$$s_{best}(0,2) = \dots$$

	0	1	2
0	N 3	NP 7	S 2
1		V 3	VP 10
2			N 5

The diagram illustrates the CYK algorithm's dynamic programming table for the sentence "I love NLP". The table is a 4x4 grid where the first column contains indices 0, 1, 2 and the first row contains non-terminal symbols N, NP, V, VP. The cells contain the best non-terminal and its score for a given span. Red arrows indicate the transitions used to compute the best score for span (0,2): a curved arrow from (0,0) to (0,2), a straight arrow from (0,1) to (0,2), and a straight arrow from (1,1) to (1,2). A large blue curved arrow on the right side of the table indicates the overall flow of the algorithm.

CYK: Example

► I love NLP

$s_{best}(0,2)$

$= s(0,2) + \max\{s_{best}(0,1) + s_{best}(2,2), s_{best}(0,0) + s_{best}(1,2)\}$

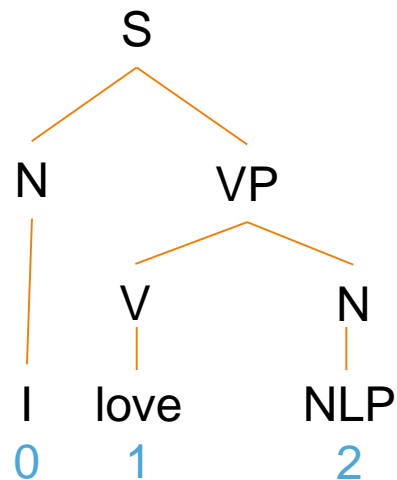
$= 2 + \max\{7 + 5, 3 + 10\} = 15$

	0	1	2
0	N 3	NP 7	S 2+3+10 =15
1		V 3	VP 10
2			N 5



CYK: Example

- ▶ To get the parse tree, follow the pointers recursively from the top down



	0	1	2
0	N 3	NP 7	S 2+3+10 =15
1		V 3	VP 10
2			N 5



Context-Free Grammars



Grammars and Constituency

- ▶ Generative grammars
 - ▶ The classic way of modeling syntax
 - ▶ A grammar specifies a set of constituent classes and 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 (constituent classes, types of 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



Example Grammar

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

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
- ▶ The generative process specifies the **grammatical structure (parse tree)** of the string



Example

$S \rightarrow NP VP$

$S \rightarrow Aux NP VP$

$S \rightarrow VP$

$NP \rightarrow Pronoun$

$NP \rightarrow Proper-Noun$

$NP \rightarrow Det Nominal$

$NP \rightarrow Nominal$

$Nominal \rightarrow Noun$

$Nominal \rightarrow Nominal Noun$

$Nominal \rightarrow Nominal PP$

$VP \rightarrow Verb$

$VP \rightarrow Verb NP$

$VP \rightarrow Verb NP PP$

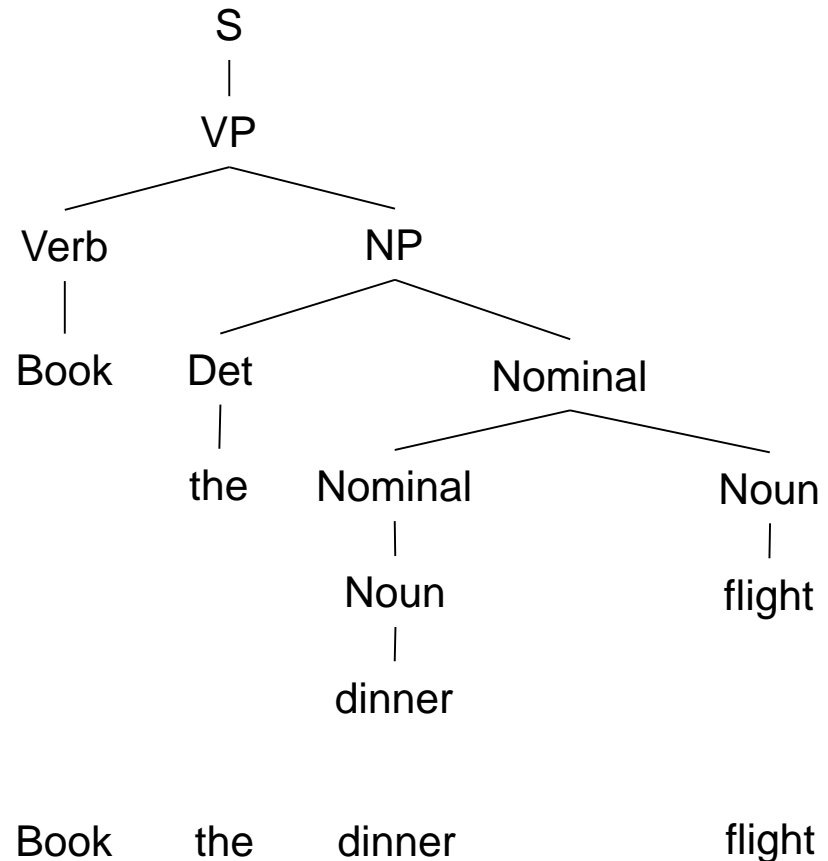
$VP \rightarrow Verb PP$

$VP \rightarrow Verb NP NP$

$VP \rightarrow VP PP$

$PP \rightarrow Preposition NP$

.....



Probabilistic Grammars

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

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

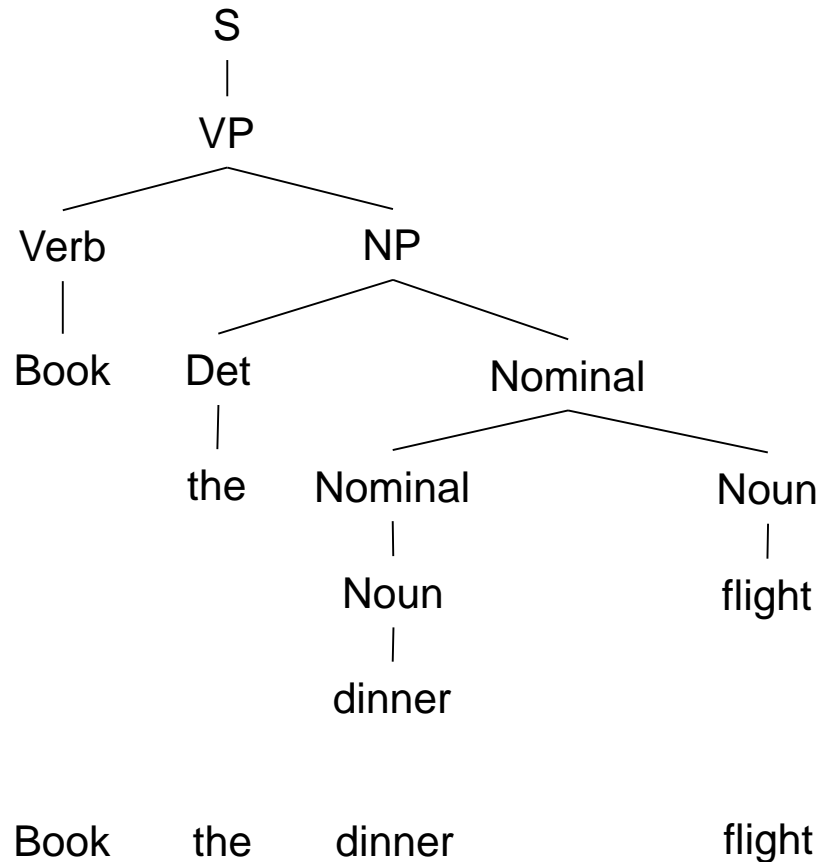
- ▶ Probability of a parse tree = product of the probabilities of all the rules used in generating the parse tree
- ▶ Weighted grammars
 - ▶ Replace probabilities with positive weights
 - ▶ Same expressiveness



Example

$S \rightarrow NP VP$	[.80]
$S \rightarrow Aux NP VP$	[.15]
$S \rightarrow VP$	[.05]
$NP \rightarrow Pronoun$	[.35]
$NP \rightarrow Proper-Noun$	[.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]

.....



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



Example

$S \rightarrow NP VP$ 1.0

$PP \rightarrow P NP$ 1.0

$VP \rightarrow V NP$ 0.7

$VP \rightarrow VP PP$ 0.3

$P \rightarrow \textit{with}$ 1.0

$V \rightarrow \textit{saw}$ 1.0

$NP \rightarrow NP PP$ 0.4

$NP \rightarrow \textit{astronomers}$ 0.1

$NP \rightarrow \textit{ears}$ 0.18

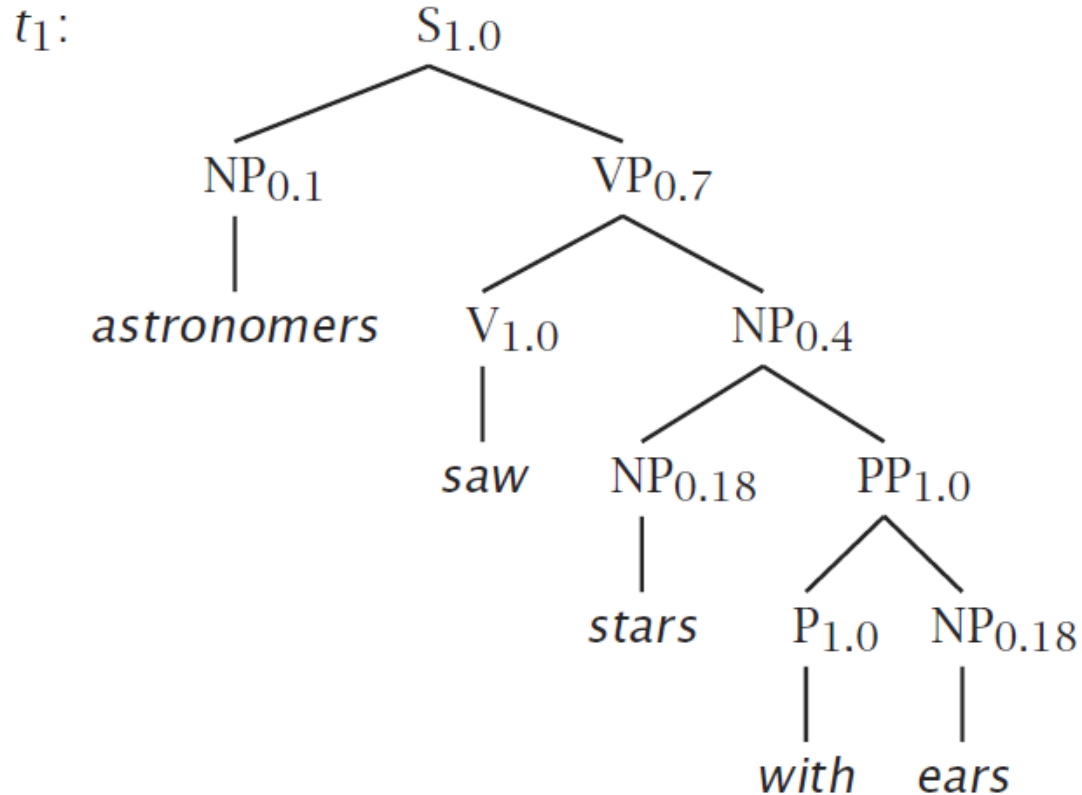
$NP \rightarrow \textit{saw}$ 0.04

$NP \rightarrow \textit{stars}$ 0.18

$NP \rightarrow \textit{telescopes}$ 0.1



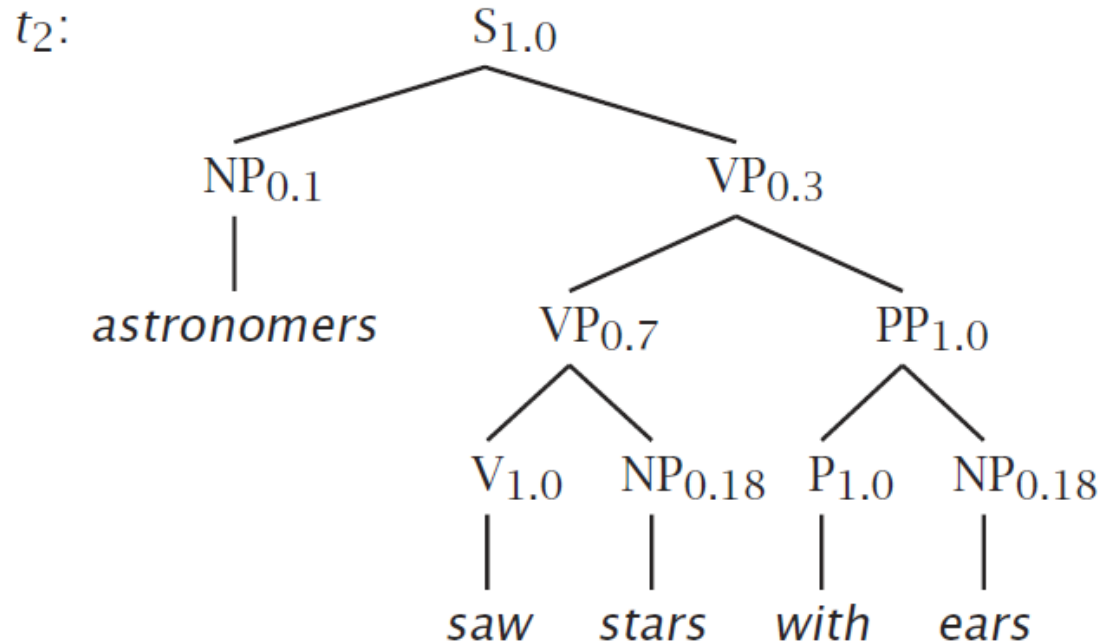
Example



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



Example



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





CFG Parsing



Parsing algorithms

- ▶ Bottom-up DP: CYK
 - ▶ Applies to CFG in **Chomsky Normal Form** (CNF)
- ▶ Top-down DP: Earley parser
 - ▶ Applies to any CFG



Chomsky Normal Form

- ▶ Only two types of production rules

$$A \rightarrow B C$$

$$A \rightarrow w$$

- ▶ What if your grammar isn't in CNF?
- ▶ Any arbitrary CFG can be rewritten into CNF
 - ▶ The resulting grammar accepts (and rejects) the same set of strings as the original grammar
 - ▶ But the resulting parse trees are different (i.e., binarized)



Conversion to CNF

- ▶ Eliminate chains of unary productions.
 - ▶ So... $A \rightarrow B, B \rightarrow C$ turns into $A \rightarrow C$
- ▶ Introduce new intermediate non-terminals into the grammar that distribute rules with length > 2 over several rules.
 - ▶ So... $S \rightarrow A B C$ turns into $S \rightarrow X C$ and $X \rightarrow A B$
 - ▶ X is a symbol that doesn't occur anywhere else in the grammar



CNF conversion

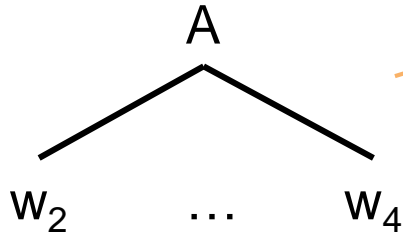
\mathcal{L}_1 Grammar	\mathcal{L}_1 in CNF
$S \rightarrow NP VP$	$S \rightarrow NP VP$
$S \rightarrow Aux NP VP$	$S \rightarrow X1 VP$
	$X1 \rightarrow Aux NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb NP$
	$S \rightarrow X2 PP$
	$S \rightarrow Verb PP$
	$S \rightarrow VP PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper-Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det Nominal$	$NP \rightarrow Det Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal Noun$	$Nominal \rightarrow Nominal Noun$
$Nominal \rightarrow Nominal PP$	$Nominal \rightarrow Nominal PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb NP$	$VP \rightarrow Verb NP$
$VP \rightarrow Verb NP PP$	$VP \rightarrow X2 PP$
	$X2 \rightarrow Verb NP$
$VP \rightarrow Verb PP$	$VP \rightarrow Verb PP$
$VP \rightarrow VP PP$	$VP \rightarrow VP PP$
$PP \rightarrow Preposition NP$	$PP \rightarrow Preposition NP$



Here we use [exclusive, inclusive] span index

CYK

- ▶ Build a table so that a nonterminal **A** spanning from i to j in the input is placed in cell $[i-1, j]$ in the table.
 - ▶ So a cell may contain zero, one or multiple nonterminals



	1	2	3	4	5
0					
1				A	
2					
3					
4					

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

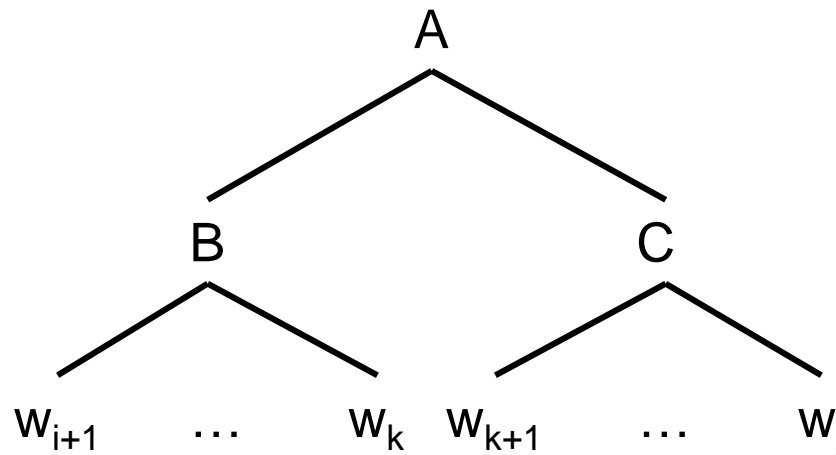
CYK

- ▶ 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**

$table[i, j] \leftarrow table[i, j] \cup$

$\{A \mid A \rightarrow BC \in grammar,$

$B \in table[i, k],$

$C \in table[k, j]\}$

► Time complexity: $O(n^3|G|)$

► n is Sentence length and $|G|$ is number of grammar rules



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

	1	2	3	4	5
0					
1					
2					
3					
4					



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

	1	2	3	4	5
0	Det				
1					
2					
3					
4					



CYK

- *The **flight** includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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



CYK Parsing

- ▶ Is that really a parser?
 - ▶ We want a parse tree, not a yes/no answer
- ▶ Simple changes
 - ▶ 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.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

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

Ambiguity

- NP \rightarrow Det N
- NP \rightarrow N N

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



Ambiguity

- $NP \rightarrow Det\ N$
- $NP \rightarrow N\ N$


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



Ambiguity

- NP \rightarrow Det N
- NP \rightarrow N N

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



Ambiguity

- NP → Det NP
- NP → NP PP

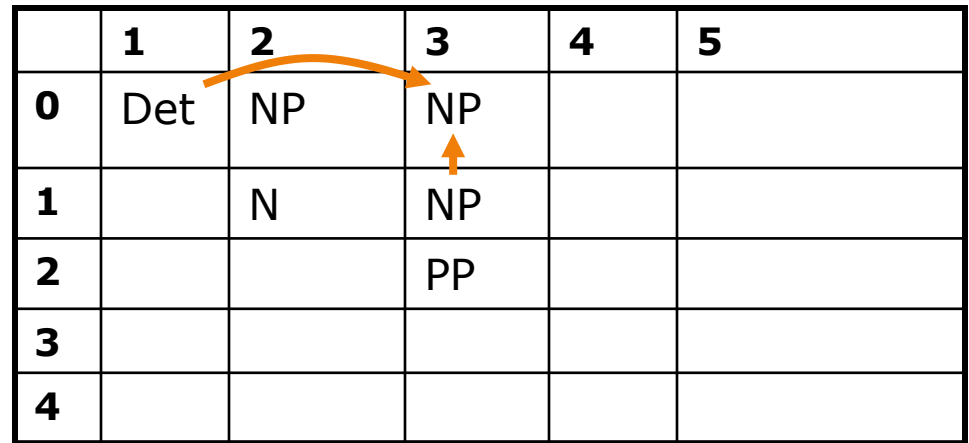
	1	2	3	4	5
0	Det	NP	??		
1		N	NP		
2			PP		
3					
4					



Ambiguity

- NP → Det NP
- NP → NP PP

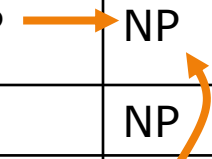
	1	2	3	4	5
0	Det	NP	NP		
1		N	NP		
2			PP		
3					
4					



Ambiguity

- NP → Det NP
- NP → NP PP

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



Probabilistic Parsing

- ▶ We have a probabilistic grammar, e.g., PCFG
- ▶ We want to find the parse tree of an input string with the highest probability
- ▶ Probabilistic CYK
 - ▶ 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



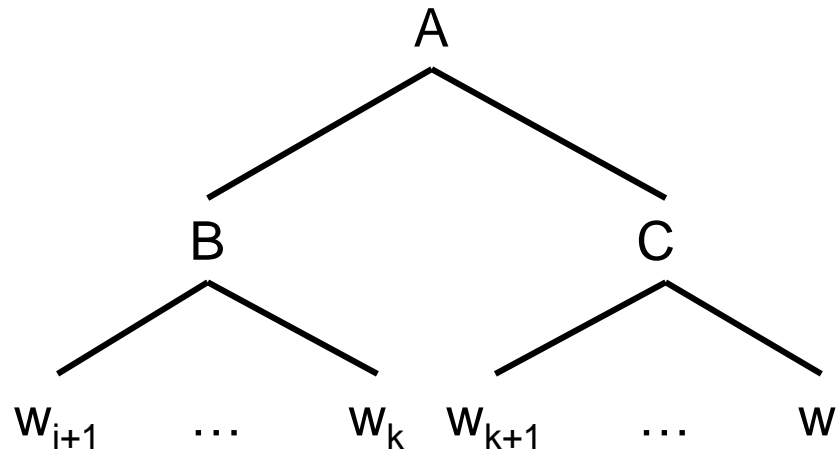
Probabilistic CYK

- ▶ Base case:

- ▶ A is in cell $[i-1, i]$ iff. there exists a rule $A \rightarrow w_i$
- ▶ $P_{A,i-1,i} = P(A \rightarrow w_i)$

- ▶ 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 .
- ▶ $P_{A,i,j} = \max_{B,C,k} P(A \rightarrow BC) \times P_{B,i,k} \times P_{C,k,j}$



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow includes$ [.05]
- $Det \rightarrow the$ [.4]
- $Det \rightarrow a$ [.4]
- $N \rightarrow meal$ [.01]
- $N \rightarrow flight$ [.02]

	1	2	3	4	5
0					
1					
2					
3					
4					



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow includes$ [.05]
- $Det \rightarrow the$ [.4]
- $Det \rightarrow a$ [.4]
- $N \rightarrow meal$ [.01]
- $N \rightarrow flight$ [.02]

	1	2	3	4	5
0	Det 0.4				
1					
2					
3					
4					



CYK

- *The **flight** includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow \text{includes}$ [.05]
- $Det \rightarrow \text{the}$ [.4]
- $Det \rightarrow \text{a}$ [.4]
- $N \rightarrow \text{meal}$ [.01]
- $N \rightarrow \text{flight}$ [.02]

	1	2	3	4	5
0	Det 0.4				
1		N 0.02			
2					
3					
4					



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow \text{includes}$ [.05]
- $Det \rightarrow \text{the}$ [.4]
- $Det \rightarrow \text{a}$ [.4]
- $N \rightarrow \text{meal}$ [.01]
- $N \rightarrow \text{flight}$ [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2					
3					
4					



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow includes$ [.05]
- $Det \rightarrow the$ [.4]
- $Det \rightarrow a$ [.4]
- $N \rightarrow meal$ [.01]
- $N \rightarrow flight$ [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3					
4					



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow includes$ [.05]
- $Det \rightarrow the$ [.4]
- $Det \rightarrow a$ [.4]
- $N \rightarrow meal$ [.01]
- $N \rightarrow flight$ [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	
4					



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow includes$ [.05]
- $Det \rightarrow the$ [.4]
- $Det \rightarrow a$ [.4]
- $N \rightarrow meal$ [.01]
- $N \rightarrow flight$ [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	
4					N 0.01



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow includes$ [.05]
- $Det \rightarrow the$ [.4]
- $Det \rightarrow a$ [.4]
- $N \rightarrow meal$ [.01]
- $N \rightarrow flight$ [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	NP 0.001
4					N 0.01



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow includes$ [.05]
- $Det \rightarrow the$ [.4]
- $Det \rightarrow a$ [.4]
- $N \rightarrow meal$ [.01]
- $N \rightarrow flight$ [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		VP .00001
3				Det 0.4	NP 0.001
4					N 0.01



CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$ [.80]
- $NP \rightarrow Det N$ [.30]
- $VP \rightarrow V NP$ [.20]
- $V \rightarrow \text{includes}$ [.05]
- $Det \rightarrow \text{the}$ [.4]
- $Det \rightarrow \text{a}$ [.4]
- $N \rightarrow \text{meal}$ [.01]
- $N \rightarrow \text{flight}$ [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			S .00000001 92
1		N 0.02			
2			V .05		VP .00001
3				Det 0.4	NP 0.001
4					N 0.01



Ambiguity


- NP \rightarrow Det N [0.7]
- NP \rightarrow N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8				
1		N 0.02			
2					
3					
4					



Ambiguity

- NP \rightarrow Det N [0.7]
- NP \rightarrow N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0056 			
1		N 0.02			
2					
3					
4					



Ambiguity


- NP → Det N [0.7]
- NP → N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0048			
1		N 0.02			
2					
3					
4					



Ambiguity

- NP \rightarrow Det N [0.7]
- NP \rightarrow N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0056 			
1		N 0.02			
2					
3					
4					



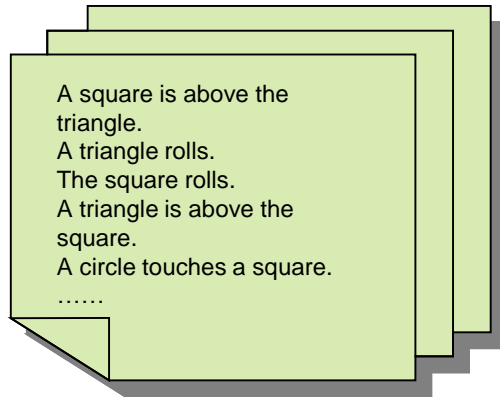


CFG Learning

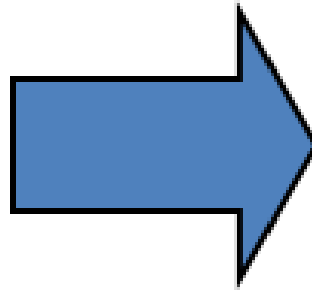


Learning a grammar from a corpus

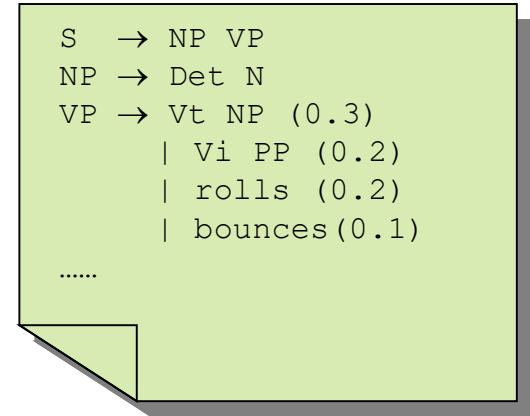
Training Corpus



Induction



Grammar / Parser



- ▶ Supervised Methods
 - ▶ Rely on a training corpus of sentences annotated with parses (treebank)
- ▶ Unsupervised Methods (Grammar Induction)
 - ▶ Do not require annotated data



Supervised Methods

▶ Treebank

- ▶ A corpus in which each sentence has been (manually) paired with a parse tree
- ▶ Manual annotation is labor intensive and generally requires linguistic knowledge and detailed guidelines.
- ▶ Most well known is the Wall Street Journal section of the Penn TreeBank.
 - ▶ 1 M words from the 1987-1989 Wall Street Journal



Generative Methods

- ▶ Learning a PCFG from treebanks
 - ▶ Maximum likelihood estimation (treebank grammar)
 - ▶ We maximize $P(x, t^*) = P(t^*)$
 - ▶ Closed-form solution: count and normalize

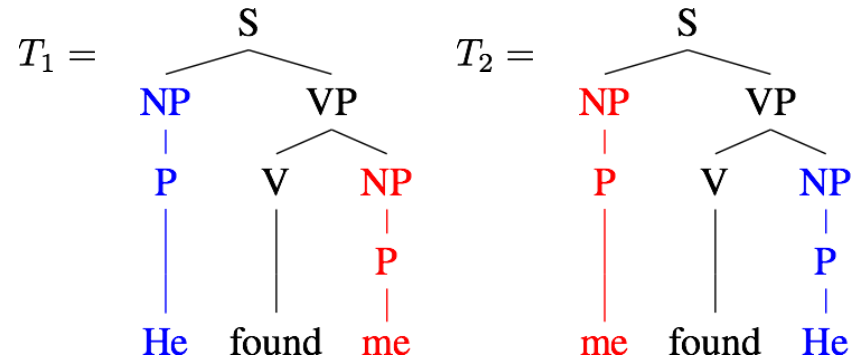
	Counts in treebank	MLE of rule probabilities
VP→Verb	20	0.2
VP→Verb NP	40	0.4
VP→Verb NP NP	25	0.25
VP→Verb PP	15	0.15



Generative Methods

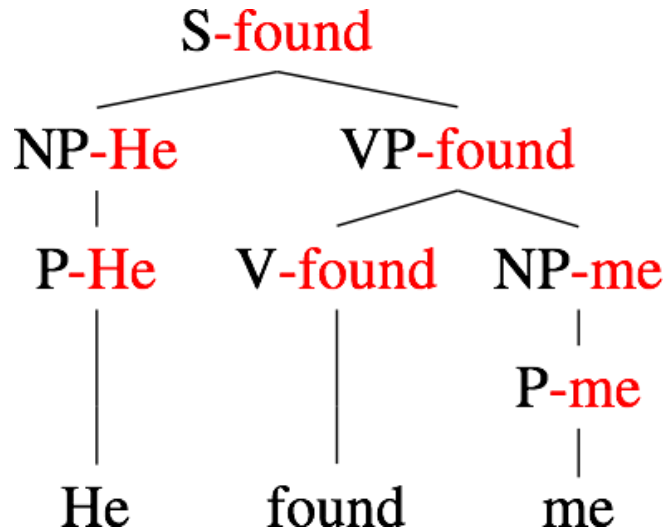
- ▶ Learning a PCFG from treebanks
 - ▶ Maximum likelihood estimation (treebank grammar)
 - ▶ MLE has bad performance (F1 score <80)
 - ▶ Main reason: standard treebank nonterminals are not sufficiently informative

$S \rightarrow NP VP$	1
$VP \rightarrow V NP$	0.2
$NP \rightarrow DT NP$	0.5
$NP \rightarrow NP NP$	0.3
... ..	
$P \rightarrow He$	0.2
$P \rightarrow me$	0.2
$V \rightarrow found$	0.1
... ..	



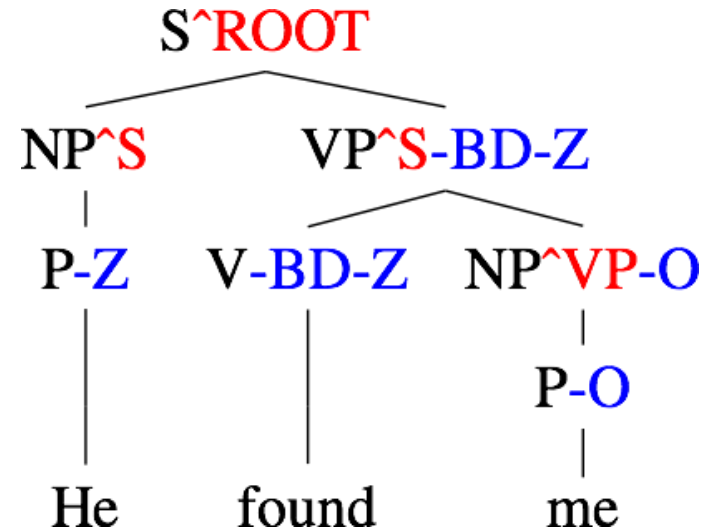
$$P(T_1) = P(T_2)$$

Generative Methods beyond MLE



Lexicalization

[Collins. 1997; Charniak. 2000]



Tree Annotation

[Johnson. 1998; Klein et al. 2003]

► Problems

- Each rule has less training data. Smoothing is very important!
- Need to do additional annotations or design rules

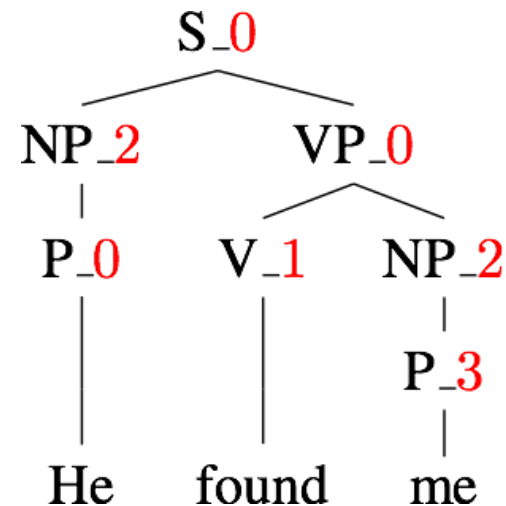


Generative Methods beyond MLE

- ▶ **Latent Variable Grammars** [Matsuzaki et al. 2005; Petrov et al. 2007]
 - ▶ Each nonterminal is split into a finite number of subtypes
 - ▶ Regard nonterminal subtypes in training parse trees as **latent variables**
 - ▶ Learning by EM

<i>Subtype rule</i>	<i>Prob.</i>
$S[0] \rightarrow NP[0] VP[0]$	0.1
$S[1] \rightarrow NP[0] VP[0]$	0.15
$S[0] \rightarrow NP[1] VP[0]$	0.05
$S[1] \rightarrow NP[1] VP[0]$	0.08
...	...

E-step
→
←
M-step



A subtype parse tree

Discriminative Methods

- ▶ We assume a **weighted** context-free grammar and maximize conditional likelihood $P(t^*|x)$

$$P(t|x) = \frac{\prod_{r \in (t,x)} \exp(W(r|x))}{Z(x)}$$

- ▶ We formulate the weight of a rule based on the input sentence
 - ▶ This is a CRF!
 - ▶ Rule weights depend on the sentence
 - ▶ ...and even for positions in the sentence, as we will see
 - ▶ More expressive than earlier generative methods

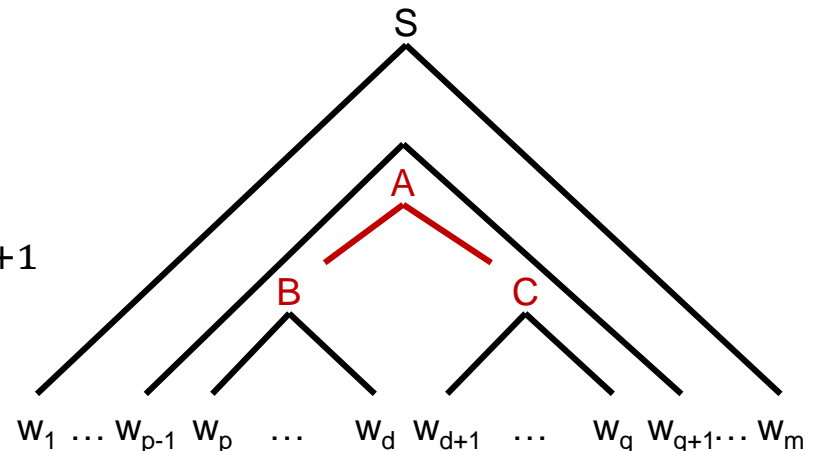


Discriminative Methods

- ▶ We formulate the weight of a rule based on the input sentence
 - ▶ ...from features of the (**anchored**) rule in the sentence

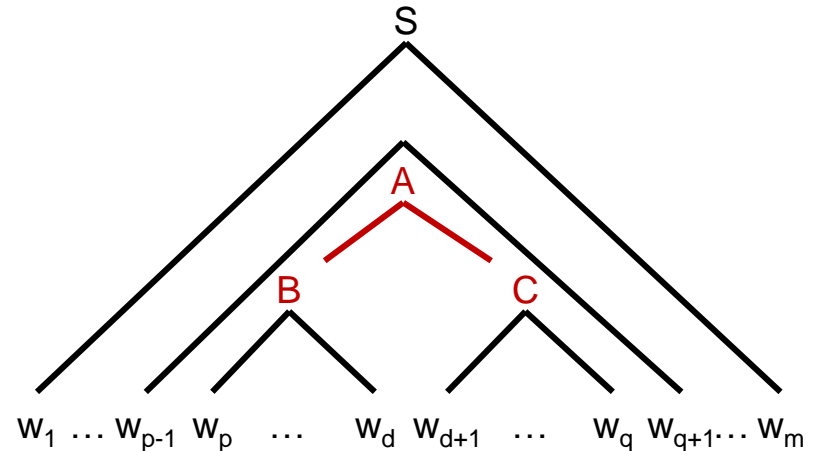
$$W(r: A \rightarrow BC, \langle p, q, d \rangle) = \sum_i \alpha_i f_i(r, \langle p, q, d \rangle)$$

- ▶ Possible features:
 - ▶ Rule identity $A \rightarrow BC$
 - ▶ Words at the span boundary
 $w_{p-1}, w_p, w_q, w_{q+1}$
 - ▶ Words at the split point w_d, w_{d+1}



Discriminative Methods

- ▶ We formulate the weight of a rule based on the input sentence
 - ▶ ...from features of the (anchored) rule in the sentence
 - ▶ ...using a neural network with nonterminal embeddings and word embeddings at the span boundary and split point as input



Discriminative Methods

- ▶ Maximizing conditional likelihood

$$P(t|x) = \frac{\prod_{r \in (t,x)} \exp(W(r|x))}{Z(x)}$$

- ▶ Partition function $Z(x)$ is computed with the inside algorithm (to be introduced next)
 - ▶ Optimization by SGD
-
- ▶ Alternative objective: margin-based loss
-
- ▶ Discriminative methods can also be applied to span-based parsing



Unsupervised Methods

- ▶ Learning from sentences with no parse tree annotation
 - ▶ ...sometimes with POS annotation
- ▶ Most work learns a PCFG
- ▶ Two tasks
 - ▶ Structure search
 - ▶ Try to find an optimal set of grammar rules
 - ▶ Parameter learning
 - ▶ Given a set of grammar rules, try to learn their probabilities



Structure search

- ▶ Two classes of approaches
 - ▶ Heuristic approaches
 - ▶ Create nonterminals and production rules using **heuristic criteria and rules**
 - ▶ Ex: Frequency of a substring, substitutability heuristic, distributional clustering, ...
 - ▶ Optimization-based approaches
 - ▶ Optimizing an **explicit objective function** (e.g., posterior) of the grammar structure by **local search**
 - ▶ Start with a trivial grammar
 - ▶ Search with a set of structure-change operations
- ▶ Empirical results
 - ▶ Poor on real data, often below simple baselines 😞



Parameter learning

- ▶ Maximum marginal likelihood estimation
 - ▶ We maximize $P(\text{sentence})$ with the parse tree marginalized
- ▶ Learning algorithm?
 - ▶ Expectation-maximization!
 - ▶ E-step: Compute the distribution of the parse tree for each training sentence
 - ▶ Infeasible to enumerate. But can compute **expected counts** of rule usage using **the inside-outside algorithm**.
 - ▶ M-step: Update the parameters to maximize expected log likelihood based on the distribution over the parse tree
 - ▶ Closed-form solution: normalize the expected counts



Inside-outside algorithm

- ▶ Assume the grammar is in the Chomsky normal form (CNF)
 - ▶ Only two types of rules
 - ▶ $N_1 \rightarrow N_2 N_3$
 - ▶ $N_1 \rightarrow w$



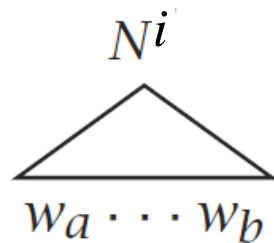
Inside-outside algorithm

► Notations

Sentence: sequence of words $w_1 \cdot \cdot \cdot w_m$

w_{ab} : the subsequence $w_a \cdot \cdot \cdot w_b$

N_{ab}^i : nonterminal N^i dominates $w_a \cdot \cdot \cdot w_b$



Span index:

- Here we use [inclusive, inclusive]
- When introducing CYK, we use [exclusive, inclusive], i.e., $(a - 1, b)$ for the same span

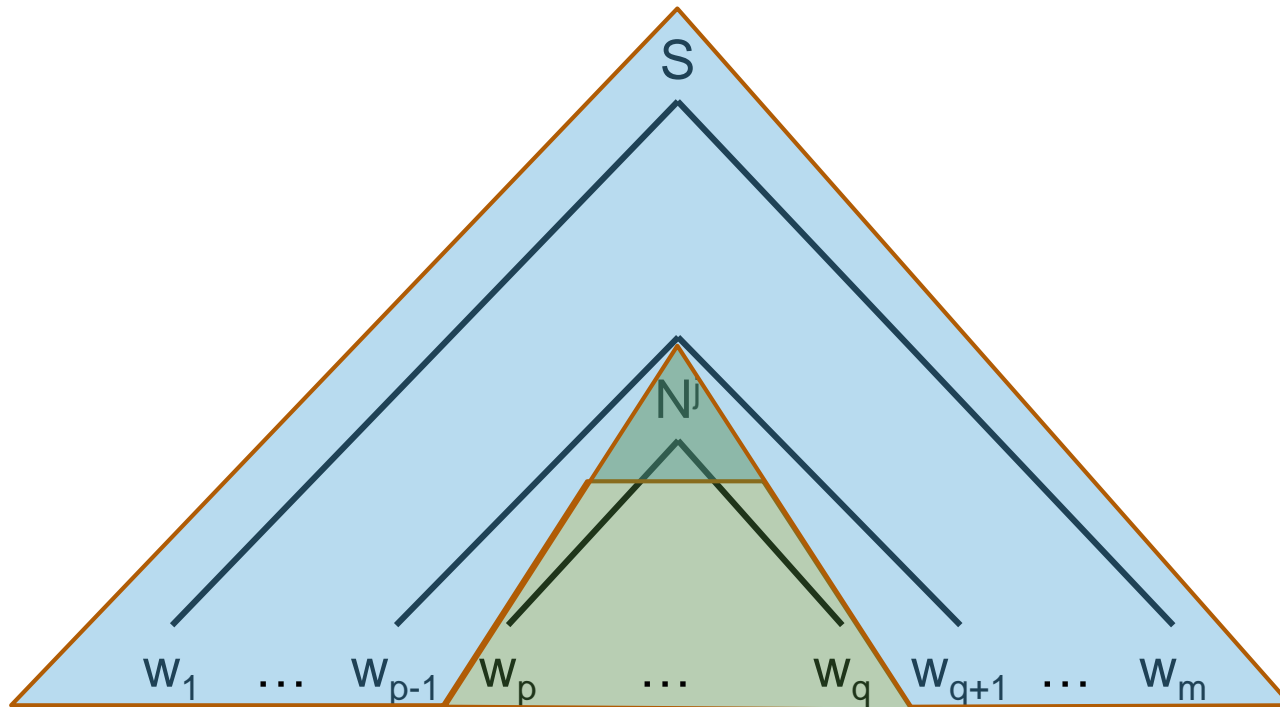


Inside-outside algorithm

- ▶ Given an input string, compute two types of probabilities

$$\text{Outside} = \alpha_j(p, q) = P(w_{1(p-1)}, N_{pq}^j, w_{(q+1)m} | G)$$

$$\text{Inside} = \beta_j(p, q) = P(w_{pq} | N_{pq}^j, G)$$



Computing inside probabilities

- ▶ Base case

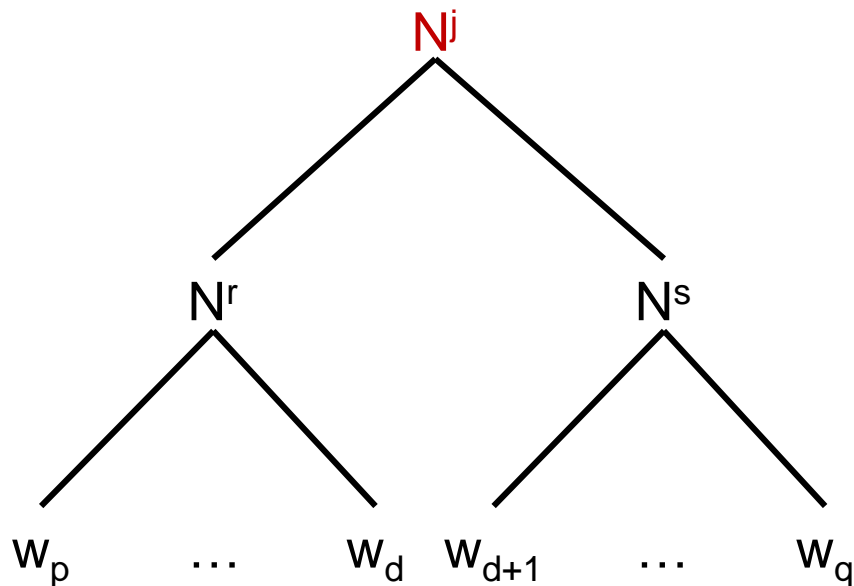
$$\begin{aligned}\beta_j(k, k) &= P(w_k | N_{kk}^j, G) \\ &= P(N^j \rightarrow w_k | G)\end{aligned}$$



Computing inside probabilities

► Bottom-up recursion

$$\begin{aligned}\beta_j(p, q) &= P(w_{pq} | N_{pq}^j, G) \\ &= \sum_{r, s} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p, d) \beta_s(d+1, q)\end{aligned}$$



Computing inside probabilities

- ▶ Almost the same as CYK parsing, but uses **sum** instead of **max**
- ▶ Looks familiar?
 - ▶ HMM
 - ▶ Viterbi algorithm uses **max**
 - ▶ Forward algorithm uses **sum**
 - ▶ HMM is a special case of PCFG
 - ▶ Viterbi algorithm is a special case of CYK algorithm
 - ▶ Forward algorithm is a special case of inside algorithm




Computing outside probabilities

$$\text{Outside} = \alpha_j(p, q) = P(w_{1(p-1)}, N_{pq}^j, w_{(q+1)m} | G)$$

► Base case

(Nonterminal #1 is S)


$$\alpha_1(1, m) = 1$$

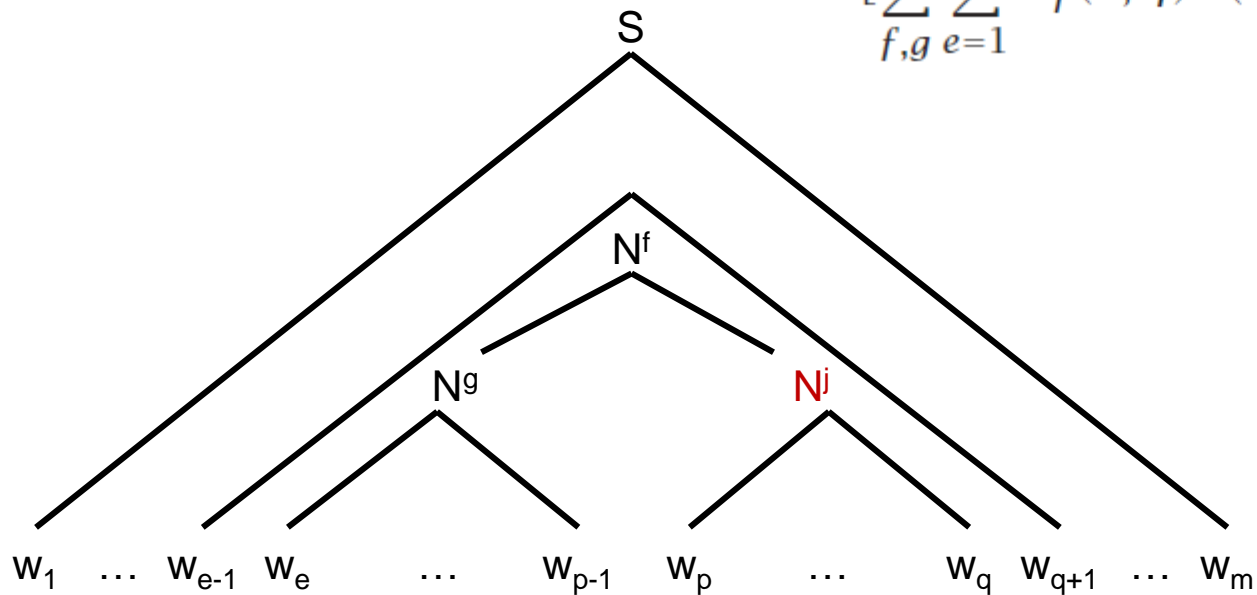
$$\alpha_j(1, m) = 0, \text{ for } j \neq 1$$



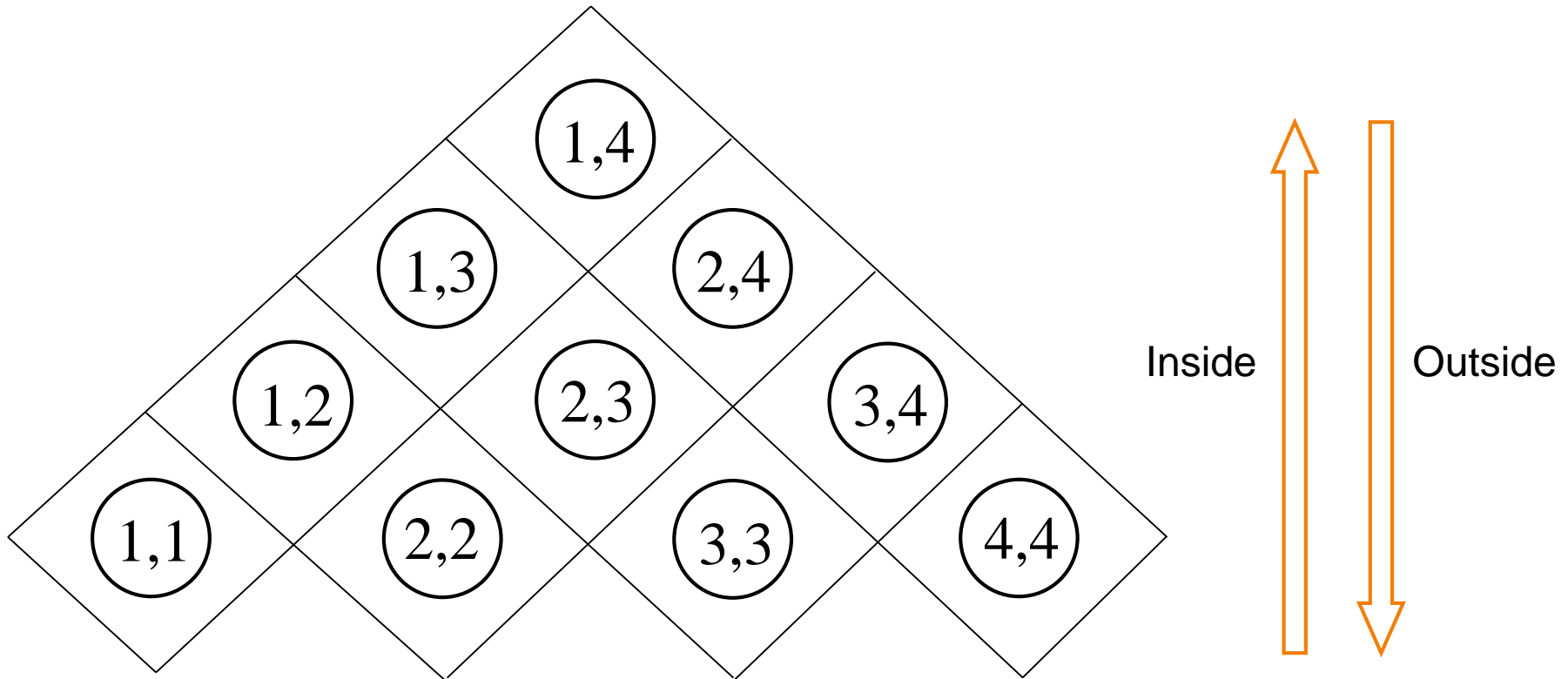
Computing outside probabilities

► Top-down recursion

$$\begin{aligned}\alpha_j(p, q) &= P(w_{1(p-1)}, N_{pq}^j, w_{(q+1)m} | G) \\ &= \left[\sum_{f,g} \sum_{e=q+1}^m \alpha_f(p, e) P(N^f \rightarrow N^j N^g) \beta_g(q+1, e) \right] \\ &\quad + \left[\sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e, q) P(N^f \rightarrow N^g N^j) \beta_g(e, p-1) \right]\end{aligned}$$



Inside-outside algorithm



- ▶ Time complexity: $O(n^3 |G|)$
 - ▶ n is Sentence length and $|G|$ is number of grammar rules



Inside-outside algorithm

- ▶ Expectation-maximization (EM)

- ▶ Initialize the probabilities (e.g., randomly)

- ▶ Repeat until convergence

- ▶ E-step: compute the expected counts

$$C(N^j \rightarrow N^r N^s \text{ used} | X, \Theta^t)$$

$$= \sum_{w_{1:m} \in X} E_{p(t|w_{1:m})} [C(N^j \rightarrow N^r N^s \text{ used in } t)]$$

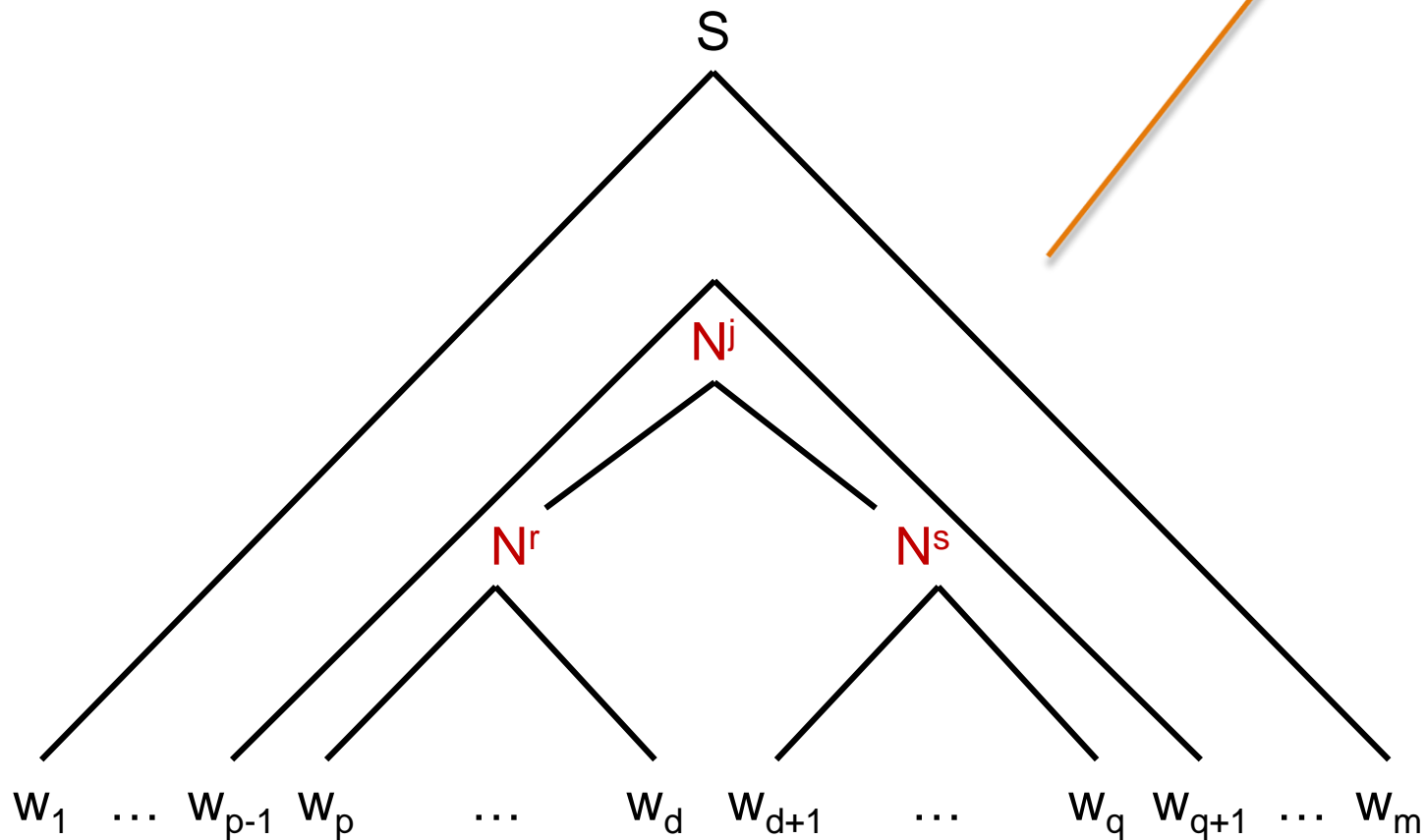
- ▶ M-step: update the probabilities by normalizing expected counts

$$\theta_{jrs}^{t+1} = P(N^j \rightarrow N^r N^s) = \frac{C(N^j \rightarrow N^r N^s \text{ used} | X, \Theta^t)}{C(N^j \text{ used} | X, \Theta^t)}$$



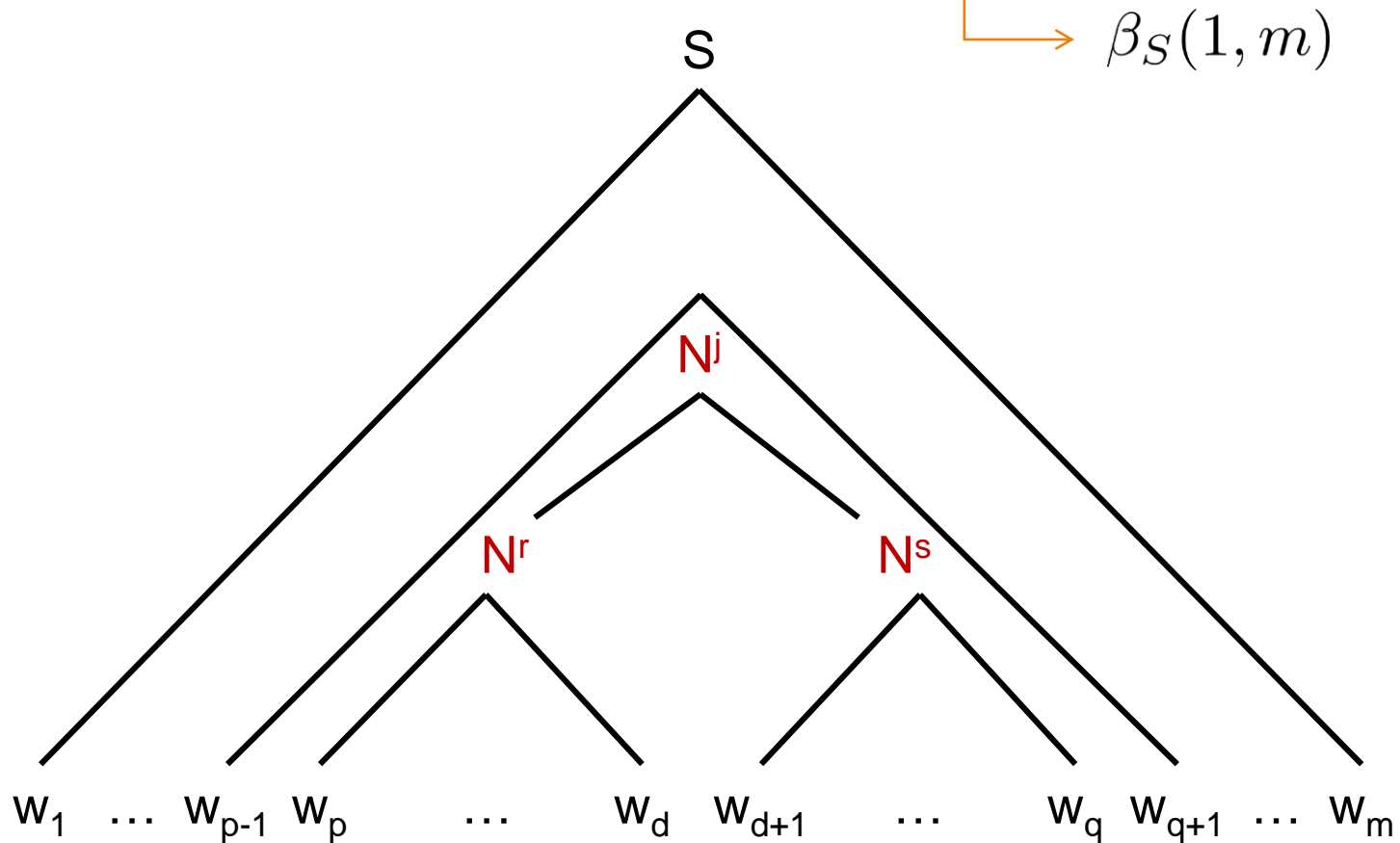
Expected counts

$$C(N^j \rightarrow N^r N^s \text{ used} | X, \Theta^t) = \sum_{w_{1,m} \in X} \sum_{p=1}^{m-1} \sum_{q=p+1}^m \sum_{d=p}^{q-1} P(\dots | w_{1,m}, \Theta^t)$$



Expected counts

$$P(\dots | w_{1,m}, \Theta^t) = \frac{\alpha_j(p,q) P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)}{P(w_{1,m})}$$



Inside-outside algorithm

- ▶ Expectation-maximization (EM)
 - ▶ Initialize the probabilities (e.g., randomly)
 - ▶ Repeat until convergence
 - ▶ E-step
 - ▶ Compute the inside probabilities
 - ▶ Compute the outside probabilities
 - ▶ Compute the expected counts
 - ▶ M-step
 - ▶ Update the probabilities

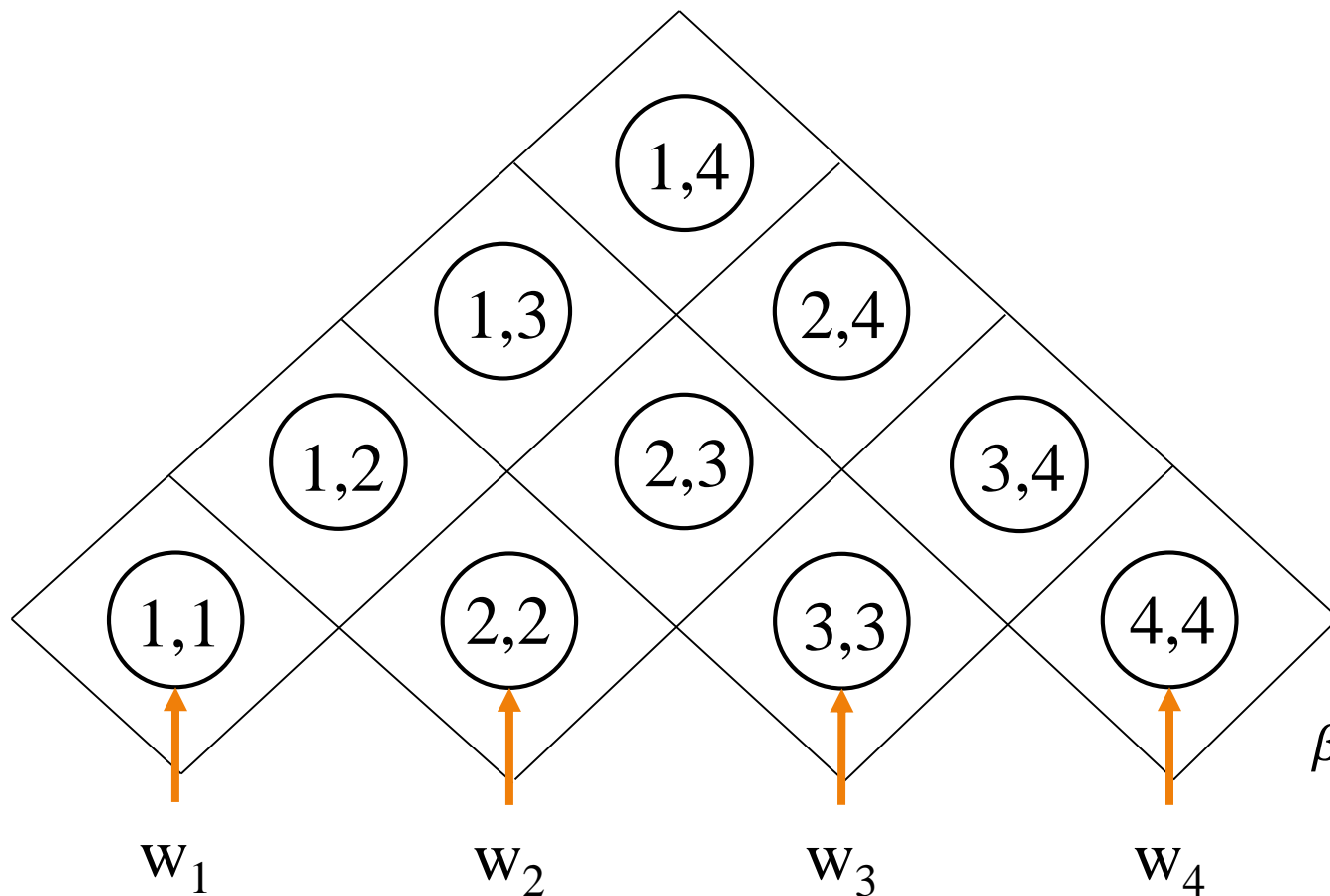


Inside-outside is just backprop!

- ▶ Expected counts can be computed by backprop
 - ▶ $C(N^j \rightarrow N^r N^s \text{ used} | X, \Theta^t) = \frac{\partial \log P(w_{1:m})}{\partial \Theta_{j,r,s}^t}$
 - ▶ The inside and then backprop procedure is almost the same as Inside-Outside
- ▶ See <https://aclanthology.org/W16-5901.pdf>



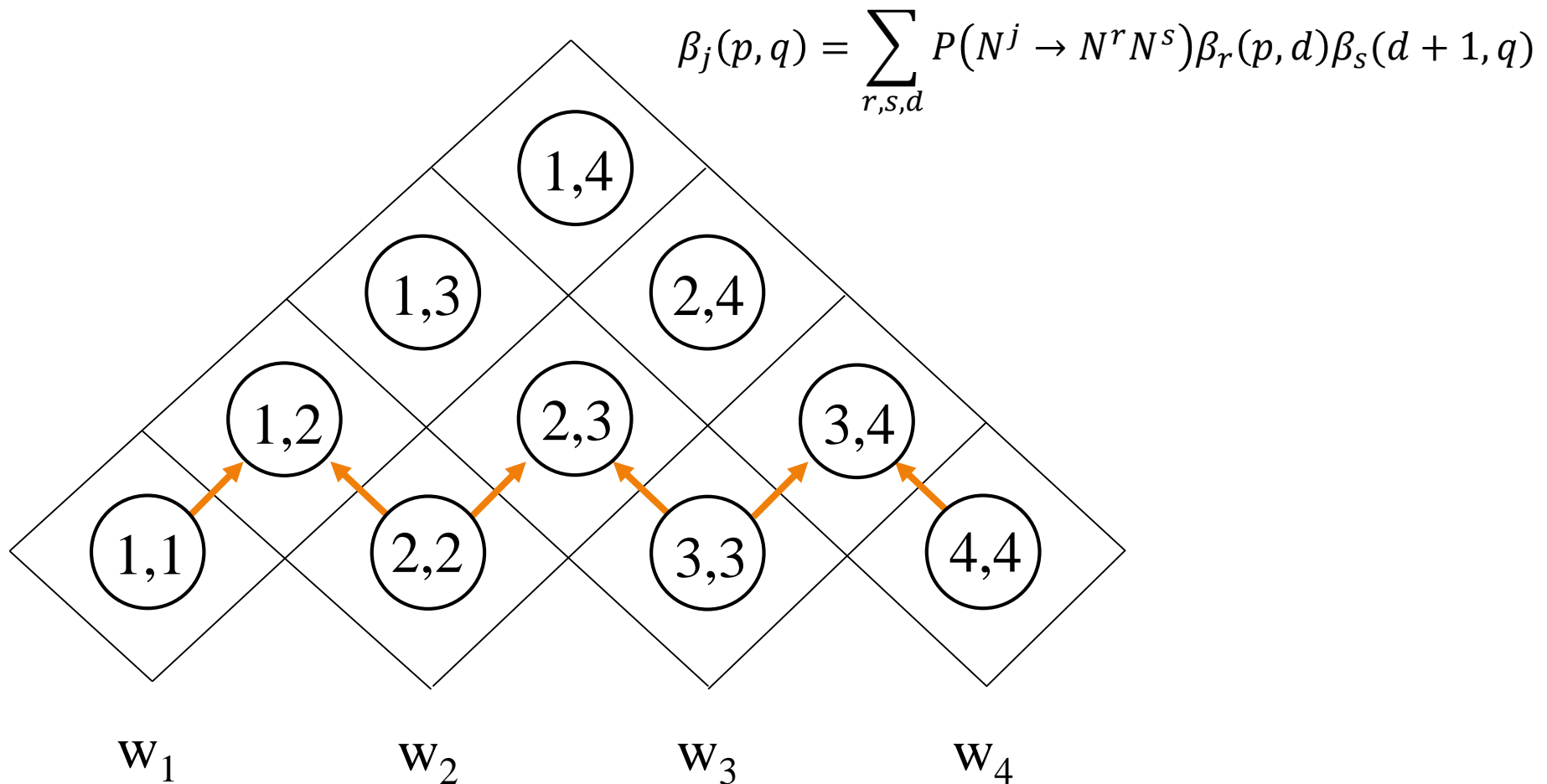
Computation graph of the inside algorithm



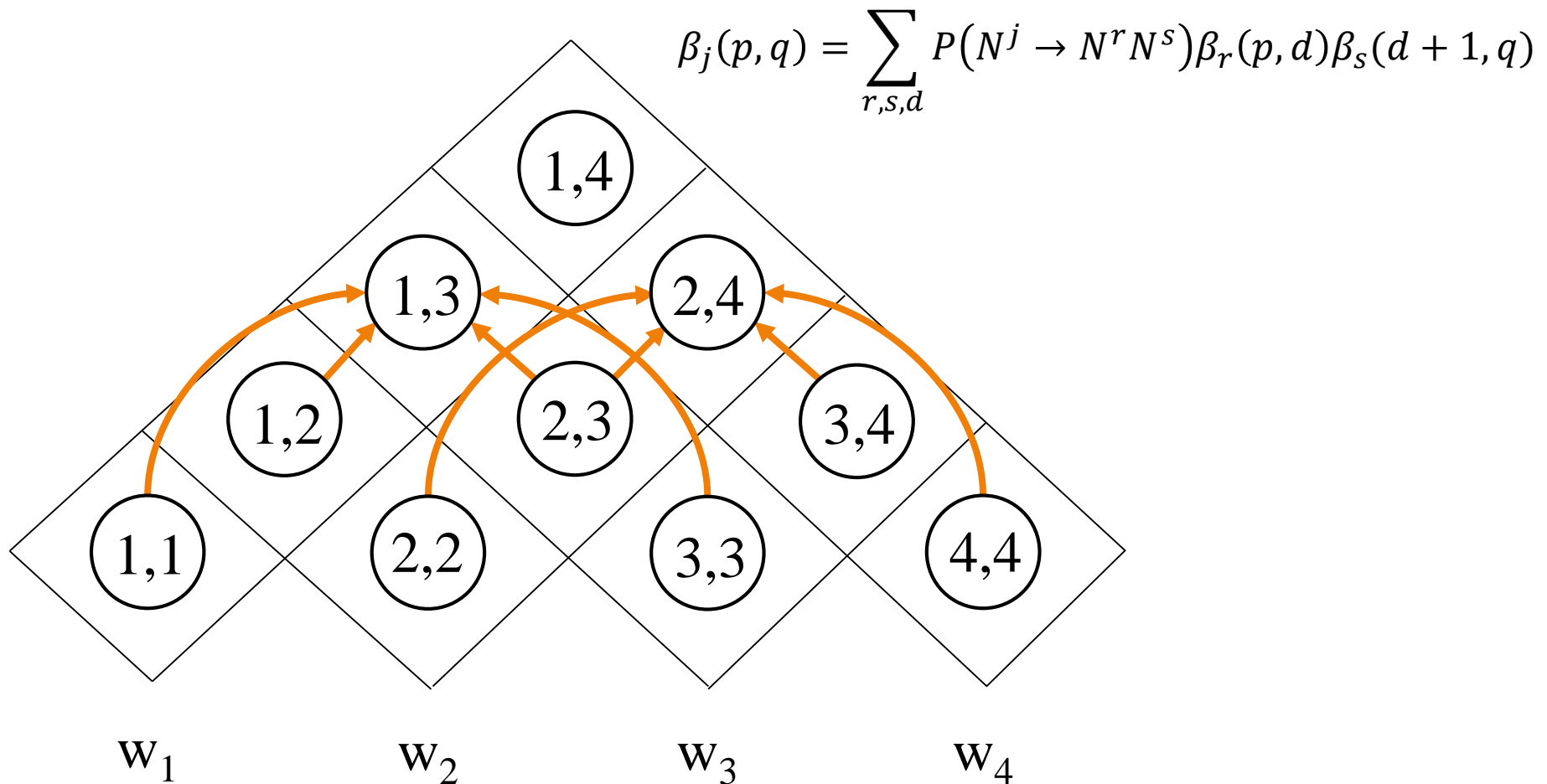
$$\beta_j(i, i) = P(N^j \rightarrow w_i)$$



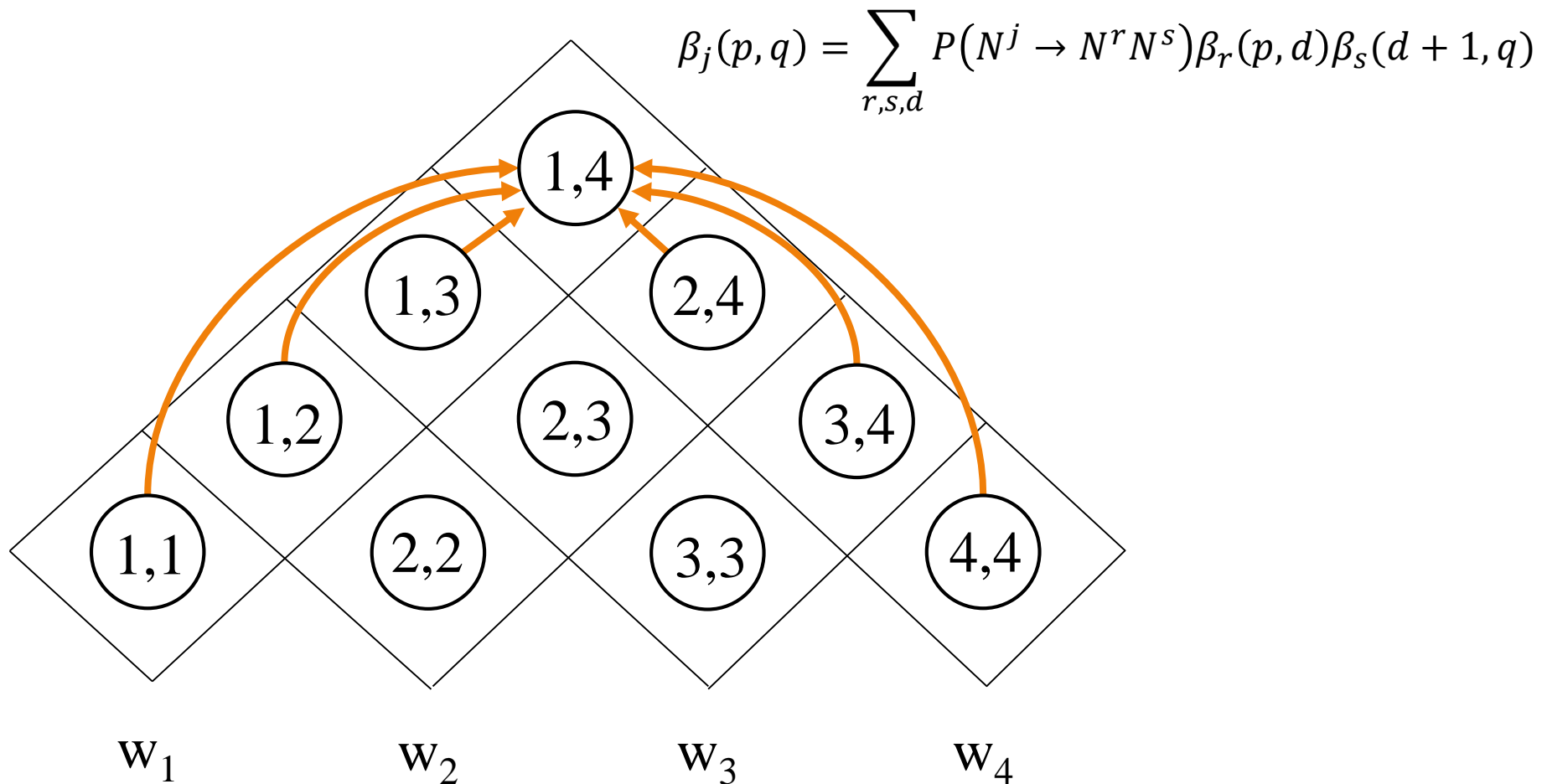
Computation graph of the inside algorithm



Computation graph of the inside algorithm

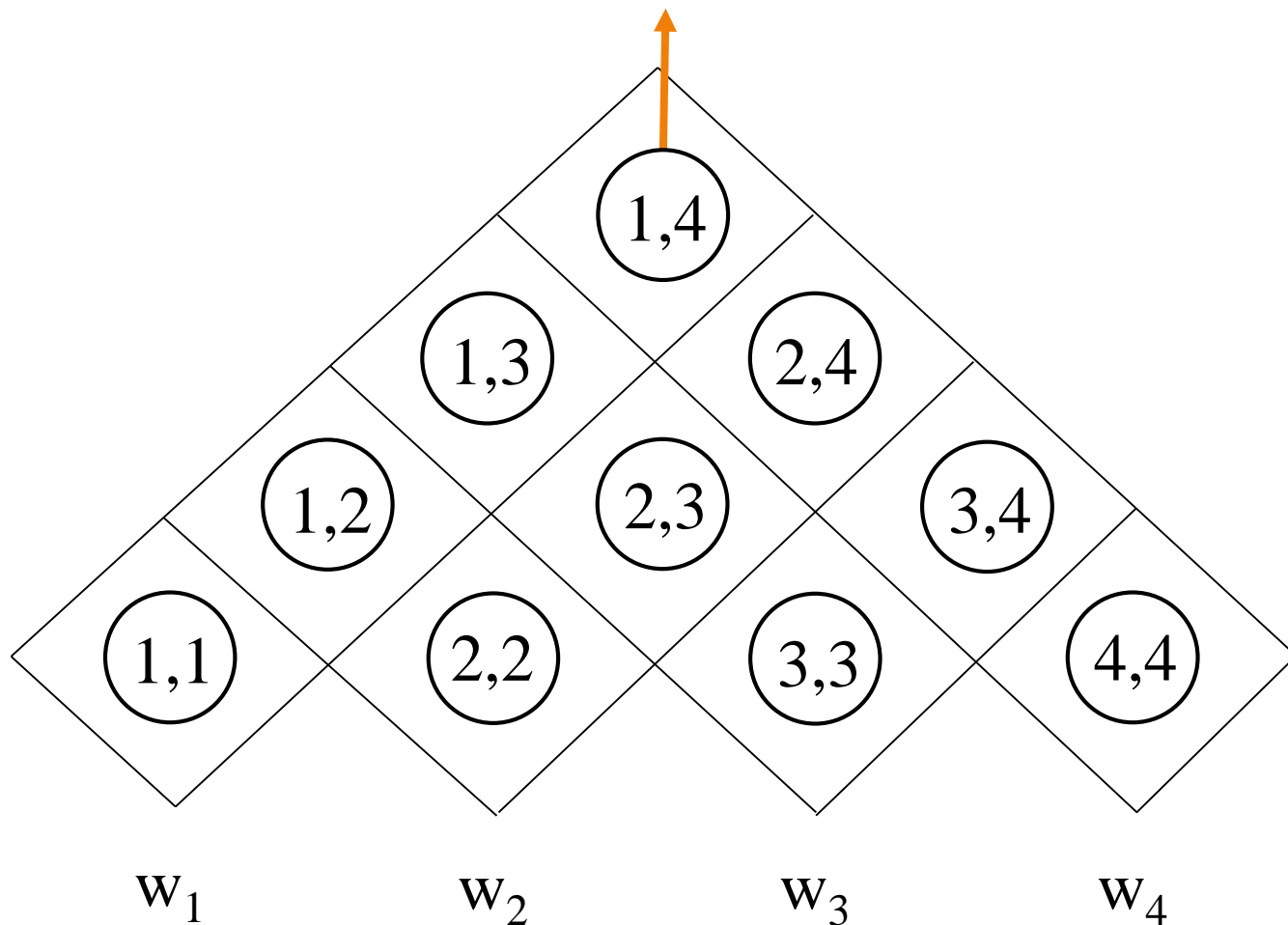


Computation graph of the inside algorithm



Computation graph of the inside algorithm

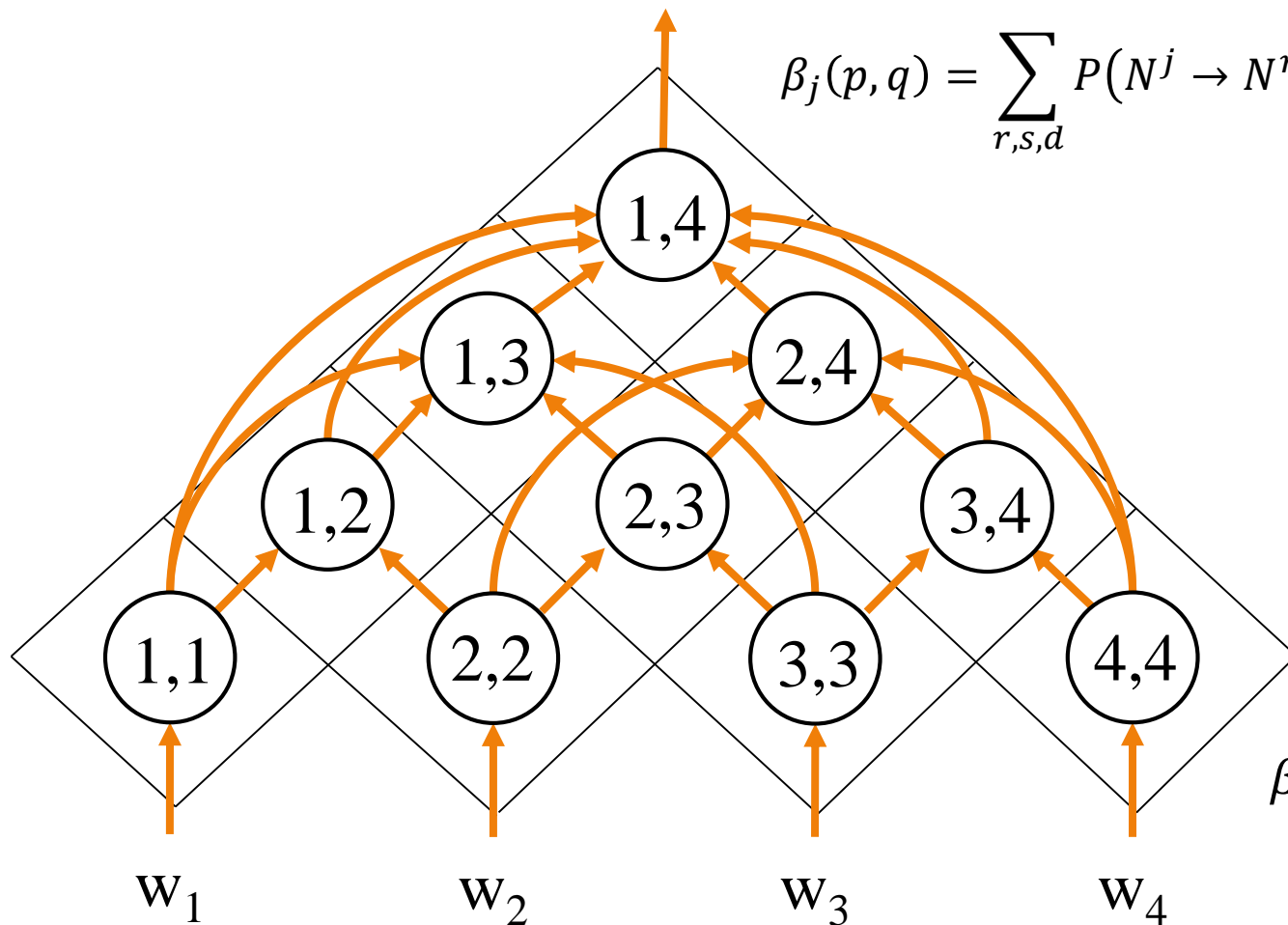
$$P(w_{1:4}) = \beta_S(1,4)$$



Computation graph of the inside algorithm

$$P(w_{1:4}) = \beta_S(1,4)$$

$$\beta_j(p, q) = \sum_{r,s,d} P(N^j \rightarrow N^r N^s) \beta_r(p, d) \beta_s(d+1, q)$$



$$\beta_j(i, i) = P(N^j \rightarrow w_i)$$

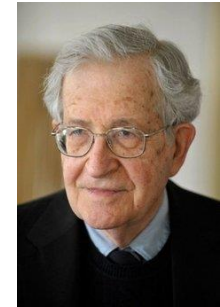
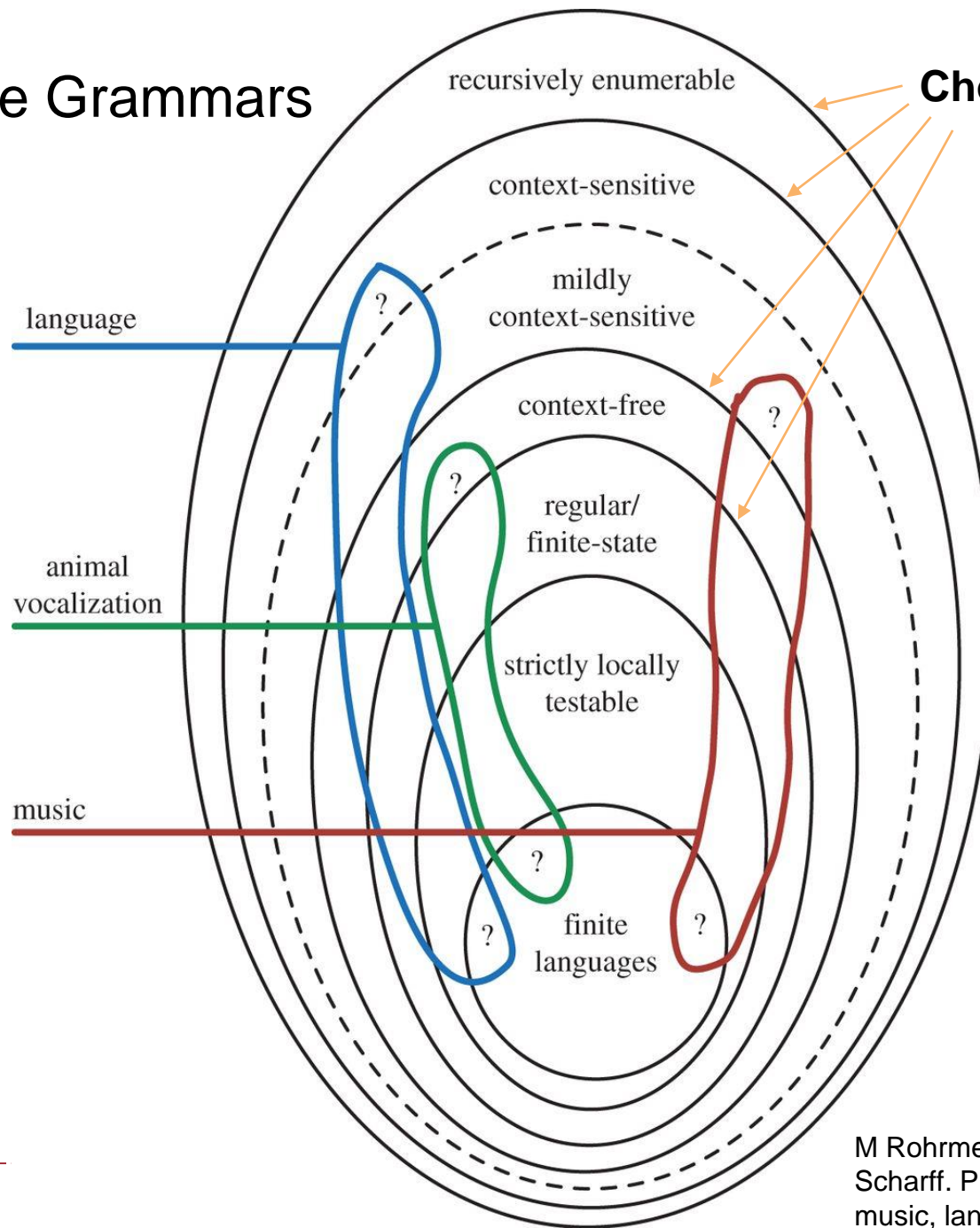


Generative Grammars Beyond CFG



Generative Grammars

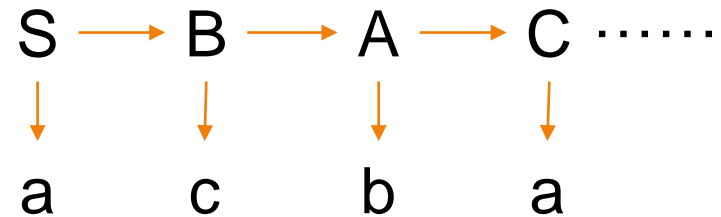
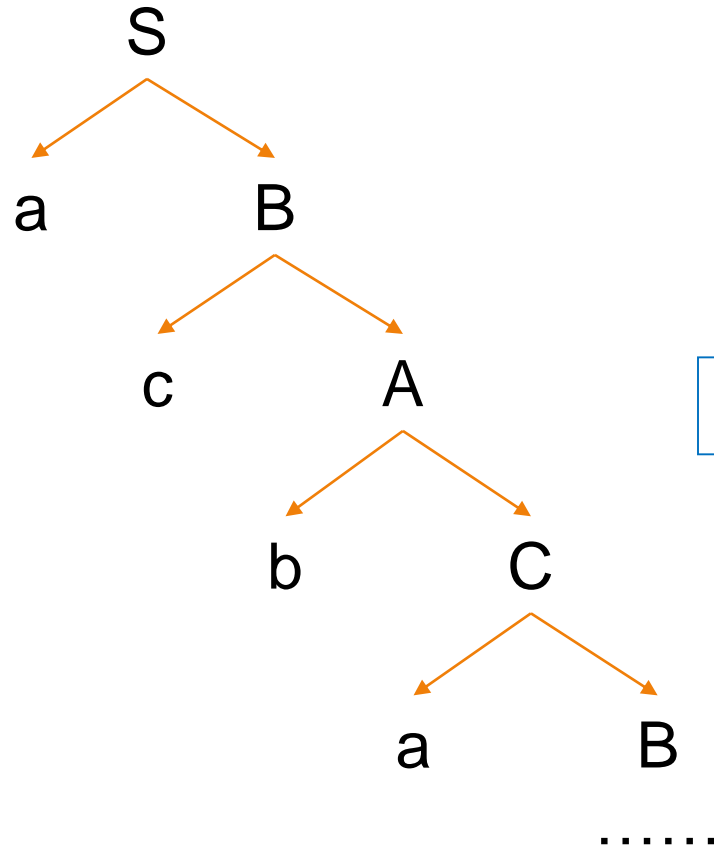
Chomsky Hierarchy



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

Regular Grammars

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



HMM = Probabilistic RG \subset PCFG

Viterbi : Forward-Backward

~

CYK : Inside-Outside

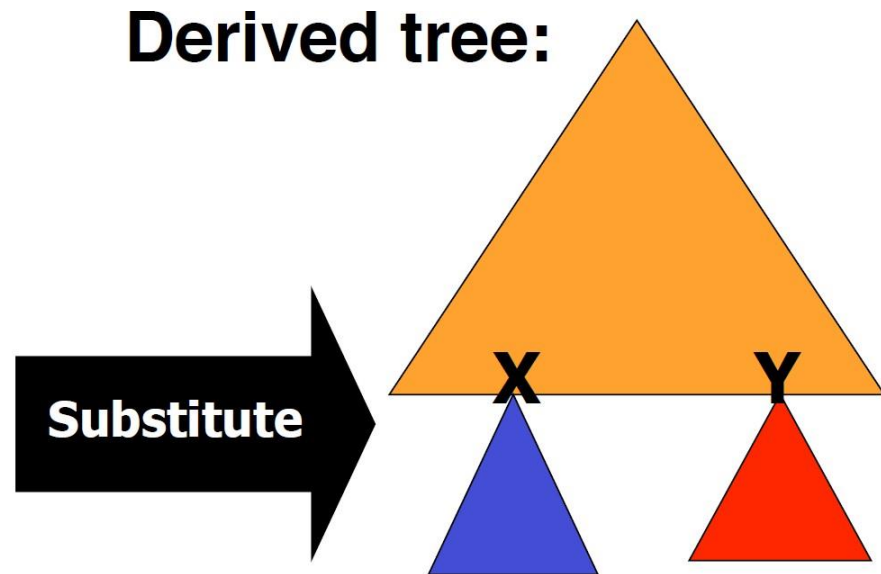
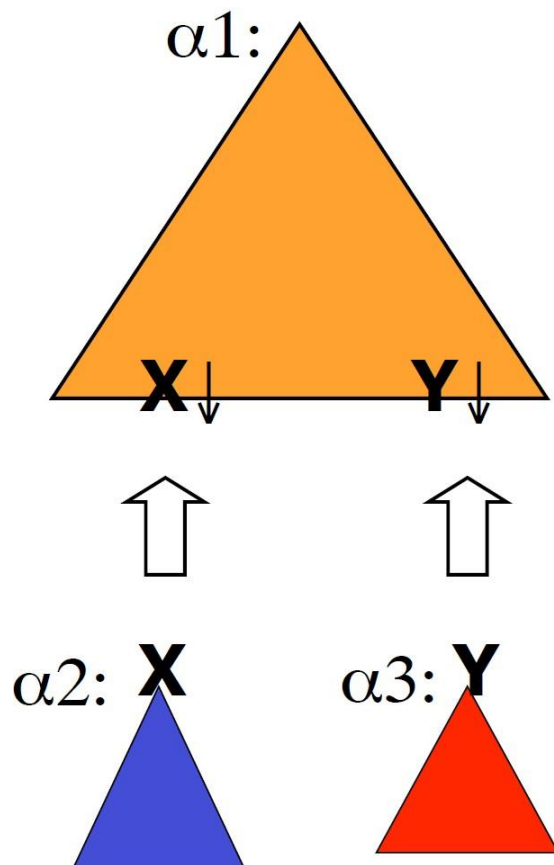


Mildly Context-Sensitive Grammars

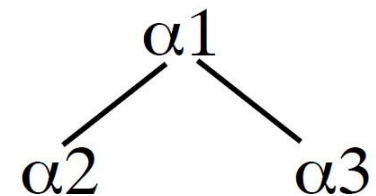
- ▶ More expressive than CFG
 - ▶ Production is no longer independent of context
- ▶ Polynomial-time parsing
- ▶ Several formalisms
 - ▶ Tree-adjoining grammar (TAG)
 - ▶ Combinatory categorial grammar (CCG)
 - ▶ Linear context-free rewriting systems (LCFRS)
 - ▶ Multiple context-free grammars (MCFG)
 - ▶ ...



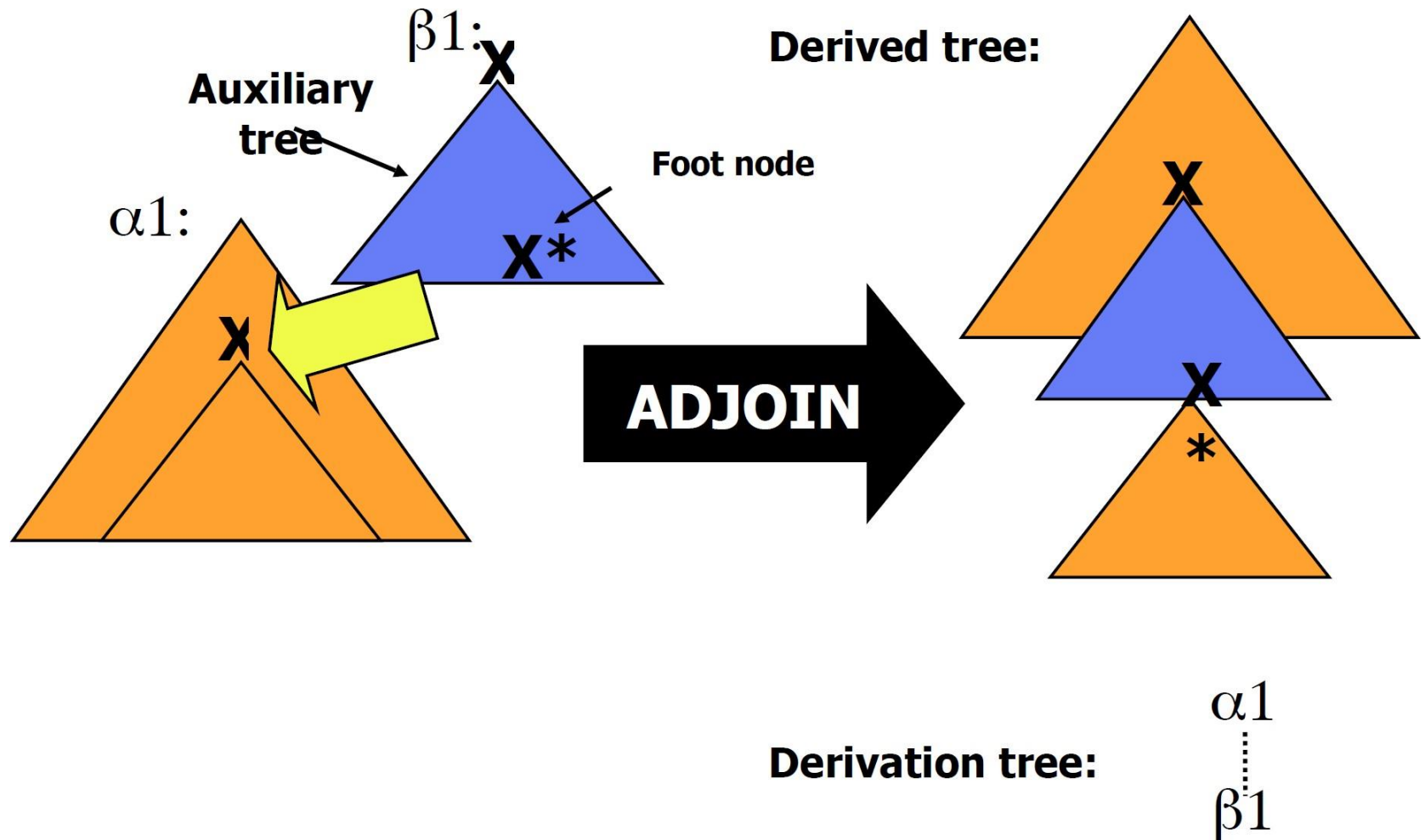
TAG Rule 1: Substitution



Derivation tree:



TAG Rule 2: Adjoin



Example: TAG

α_2 :

NP
|
John

α_3

:
NP
|
tapas

α_1 :

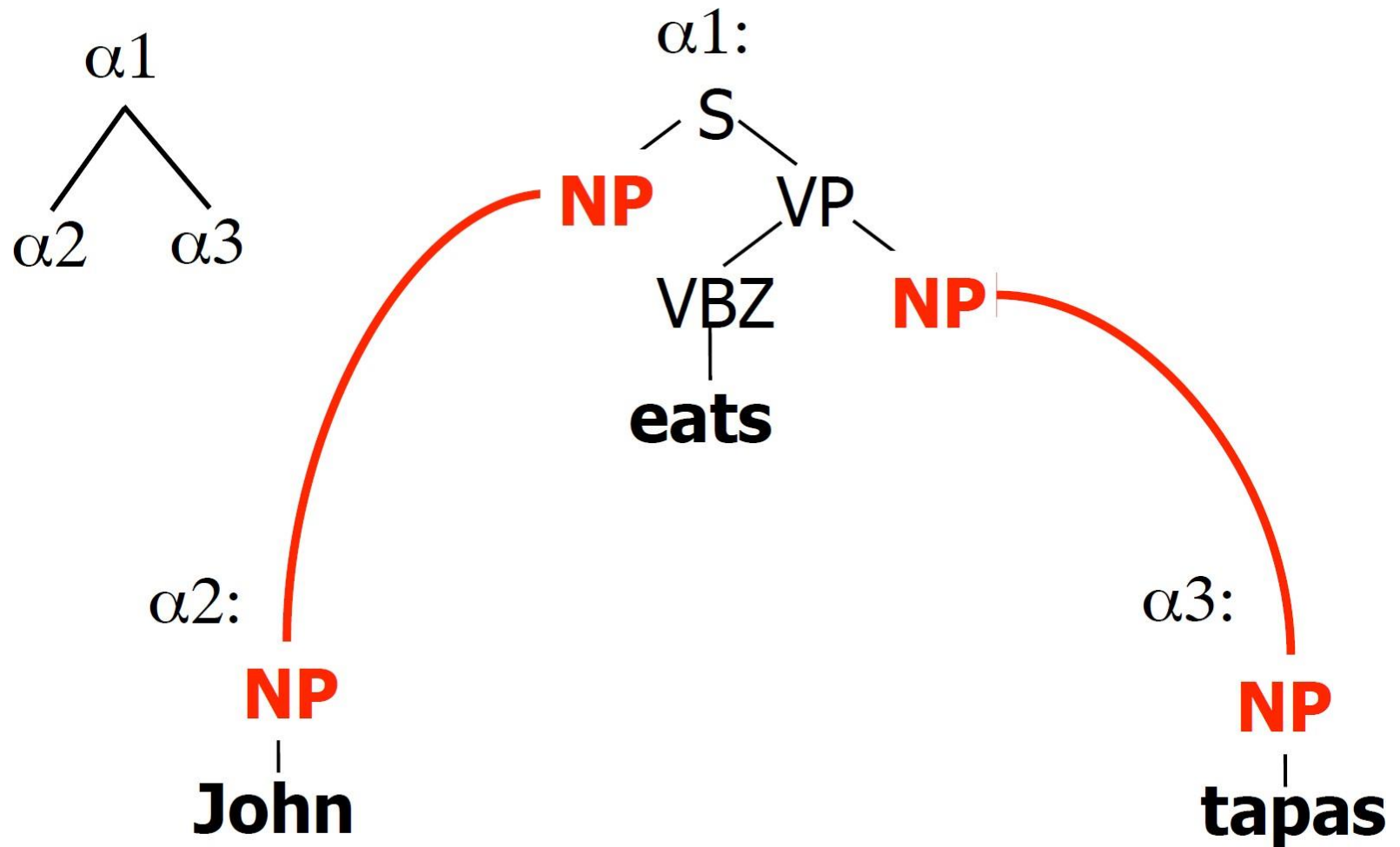
S
/ \
NP VP
/ \
VBZ NP
|
eats

β_1 :

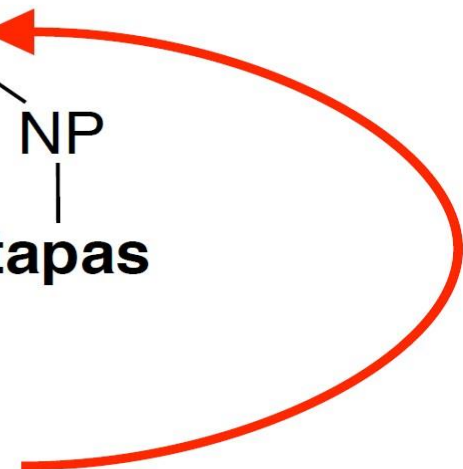
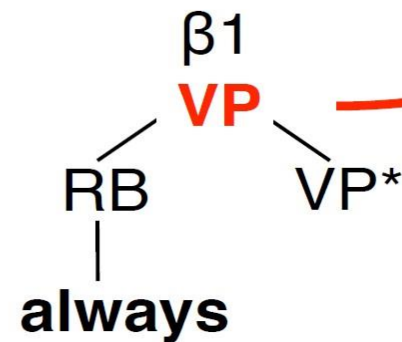
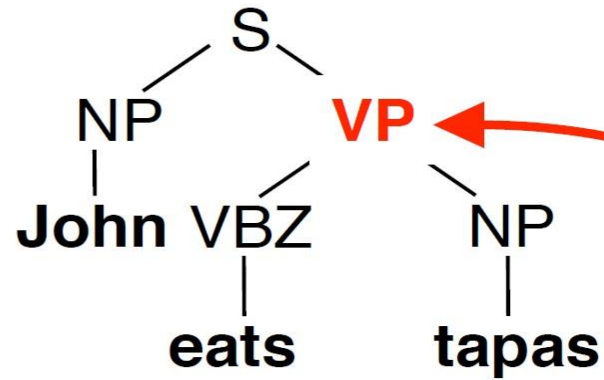
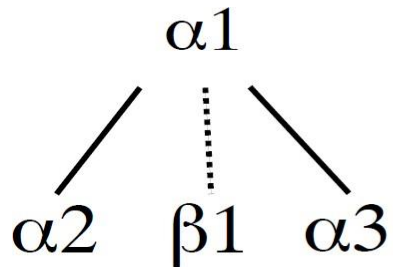
VP
/ \
RB VP*
|
always



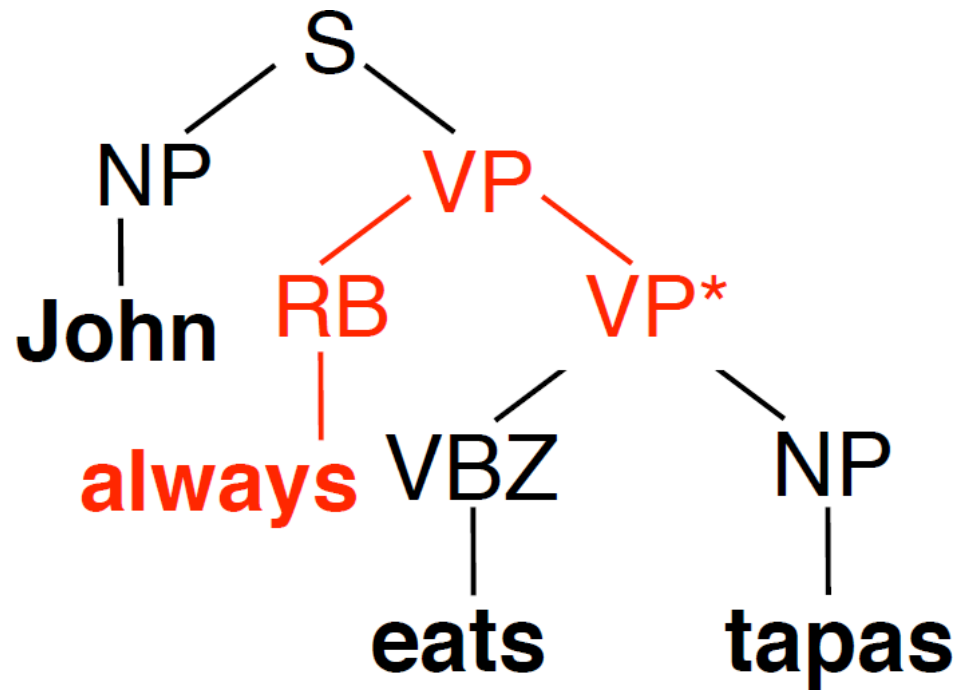
Example: TAG derivation



Example: TAG derivation



Example: TAG derivation



Transition-based Constituency Parsing

Transition-Based Parsing

- ▶ A parse tree represented as a linear sequence of **transitions**.
- ▶ Transitions: simple actions to be executed on a **parser configuration**.
- ▶ Parser configuration
 - ▶ Buffer B : unprocessed words of the input sentence
 - ▶ Stack S : parse tree under construction



Transition-Based Parsing

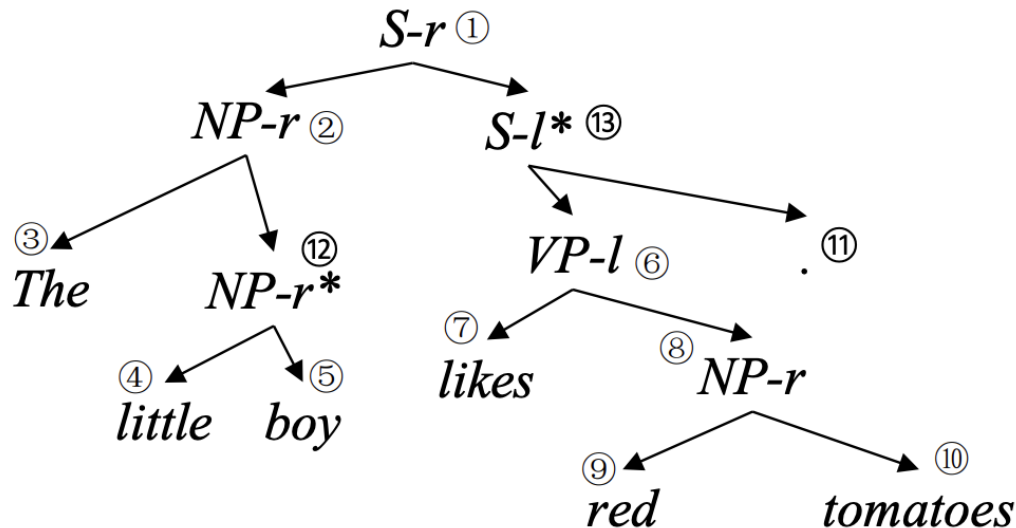
- ▶ Initial Configuration
 - ▶ Buffer B contains the complete input sentence and stack S is empty.
- ▶ During parsing
 - ▶ Apply a classifier to decide which transition to take next.
 - ▶ No backtracking.
- ▶ Final Configuration
 - ▶ Buffer B is empty and stack S contains the entire parse tree.



Transition-Based Parsing

Transition-based parsing systems

- ▶ Bottom up system
- ▶ Top-down system
- ▶ In-order system
- ▶ ...



stack	buffer	action	node
[]	[The little ...]	SHIFT	③
[The]	[little boy ...]	SHIFT	④
[The little]	[boy likes ...]	SHIFT	⑤
[... little boy]	[likes red ...]	REDUCE-R-NP	②
...

(a) bottom-up system

stack	buffer	action	node
[]	[The little ...]	NT-S	①
[(S]	[The little ...]	NT-NP	②
[(S (NP]	[The little ...]	SHIFT	③
[... (NP The]	[little boy ...]	SHIFT	④
[... The little]	[boy likes ...]	SHIFT	⑤
[... little boy]	[likes red ...]	REDUCE	/
...

(b) top-down system

stack	buffer	action	node
[]	[The little ...]	SHIFT	③
[The]	[little boy ...]	PJ-NP	②
[The NP]	[little boy ...]	SHIFT	④
[... NP little]	[boy likes...]	SHIFT	⑤
[... little boy]	[likes red ...]	REDUCE	/
...

(c) in-order system

Bottom-up Transition-Based Parsing

- ▶ Transitions: bottom up transition set (Sagae, 2005)
 - ▶ Assume a CNF grammar
 - ▶ SHIFT: move the word at the front of buffer B onto stack S .
 - ▶ REDUCE- X : $A = \text{pop}(S); B = \text{pop}(S); \text{push}(S, X \rightarrow BA)$.
 - ▶ A, B, X : nonterminal
 - ▶ UNARY- Y : $w = \text{pop}(S); \text{push}(S, Y \rightarrow w)$.
 - ▶ w : terminal (word)
 - ▶ Y : nonterminal



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :

--

the
boy
likes
NLP

Actions:



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :

the

boy

likes

NLP

Actions: **SHIFT**



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :

Det ↓ the

boy
likes
NLP

Actions: **UNARY** – Det



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :

boy
Det ↓ the

likes
NLP

Actions: **SHIFT**



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :

NP ↓ boy
Det ↓ the

likes
NLP

Actions: **UNARY** – NP



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :



Actions: **REDUCE – NP**



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :



Actions: **SHIFT**



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :



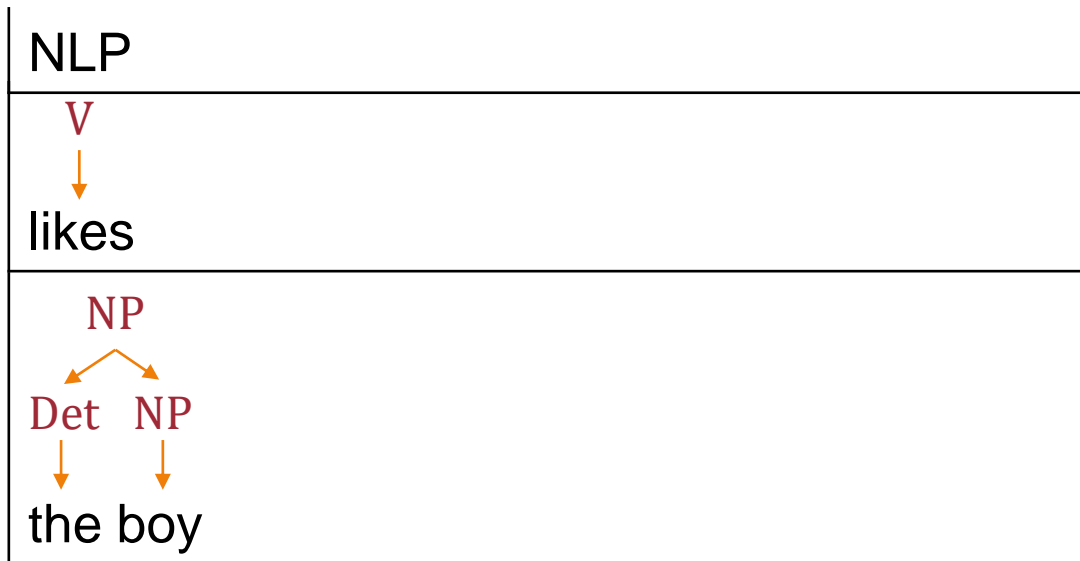
Actions: **UNARY** – V



Bottom-up Transition-Based Parsing: Example

Stack S :

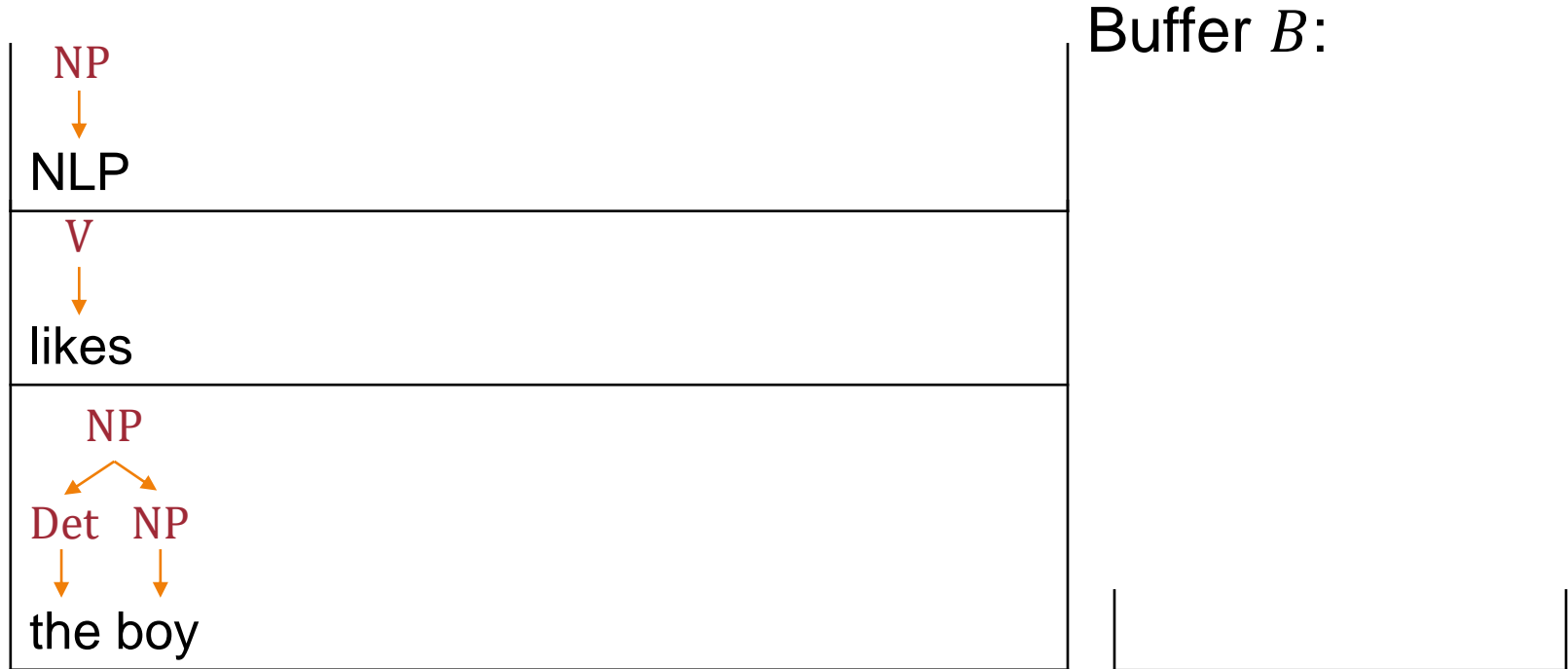
Buffer B :



Actions: **SHIFT**



Bottom-up Transition-Based Parsing: Example

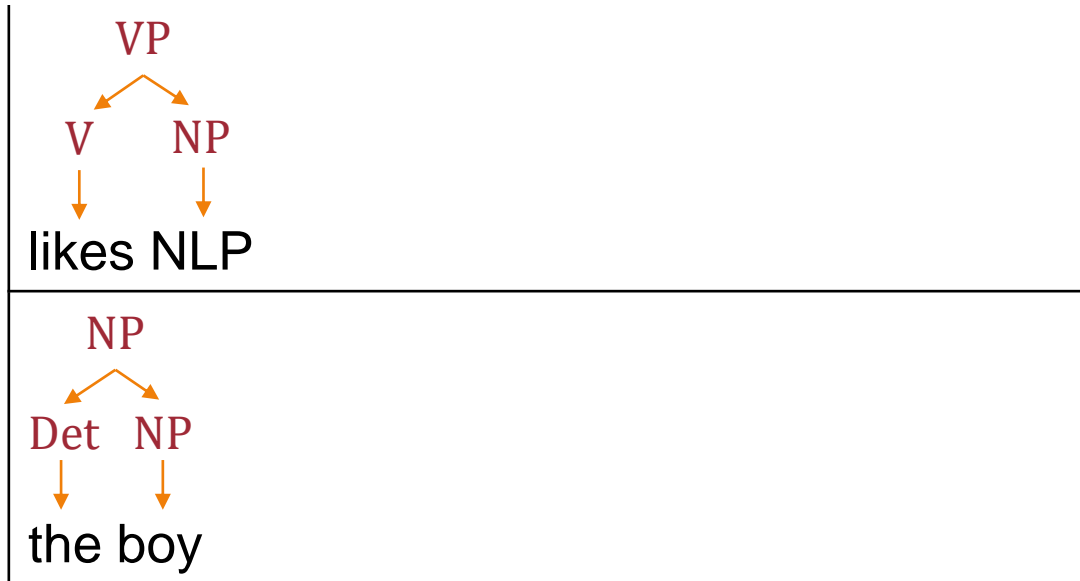


Actions: UNARY – NP



Bottom-up Transition-Based Parsing: Example

Stack S :



Buffer B :



Actions: **REDUCE – VP**



Bottom-up Transition-Based Parsing: Example

Stack S :

Buffer B :

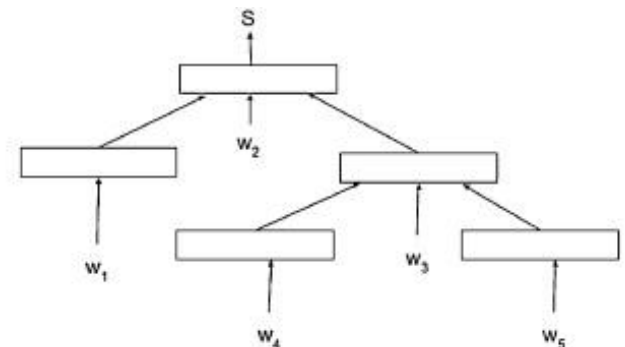


Actions: **REDUCE** – S



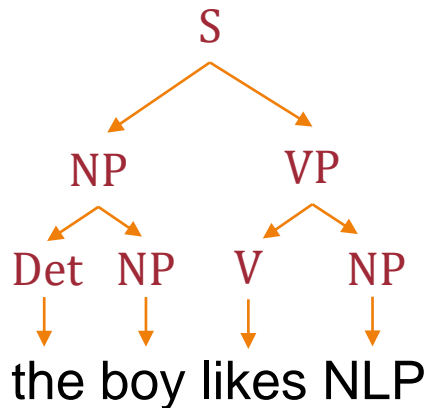
The Core of Transition-Based Parsing: Classification

- ▶ At each iteration, choose among {SHIFT, REDUCE–X, UNARY–Y}
 - ▶ $|N|$ variants of REDUCE–X and UNARY–Y
- ▶ Train a classifier to score actions
 - ▶ Input features can be extracted from stack S , buffer B , and the history H of past actions.
 - ▶ Can use a neural network to summarize info from S, B, H
 - ▶ LSTM over B and H
 - ▶ LSTM + recursive neural net over S



The Core of Transition-Based Parsing: Classification

- ▶ Train a classifier to score actions
 - ▶ Training data
 - ▶ Convert a gold parse tree to an “oracle” sequence of **<configuration, correct transition>** pairs



<B₁, S₁, SHIFT>
→ <B₂, S₂, UNARY-Det>
→ <B₃, S₃, SHIFT>
→ <B₄, S₄, UNARY-NP>
...



The Core of Transition-Based Parsing: Classification

- ▶ Train a classifier to score actions
 - ▶ Training data
 - ▶ Convert a gold parse tree to an “oracle” sequence of **⟨configuration, correct transition⟩** pairs
 - ▶ Potential flaw
 - ▶ The classifier is only trained with correct configurations (i.e., assuming all the preceding classification decisions were all correct).
 - ▶ In testing, however, the classifier can make mistakes and produce incorrect configurations not seen during training.
 - ▶ **Dynamic oracle**
 - ▶ Make random errors to produce incorrect configurations
 - ▶ Find transitions (e.g., using rules) that can change these configurations to a gold configuration in the fastest possible way
 - ▶ Train the classifier to predict these transitions



Transition-Based Parsing: Remarks

- ▶ Action selection

- ▶ Greedy: pick the highest scoring predicted action
- ▶ Beam search: pick multiple actions, maintain multiple hypotheses

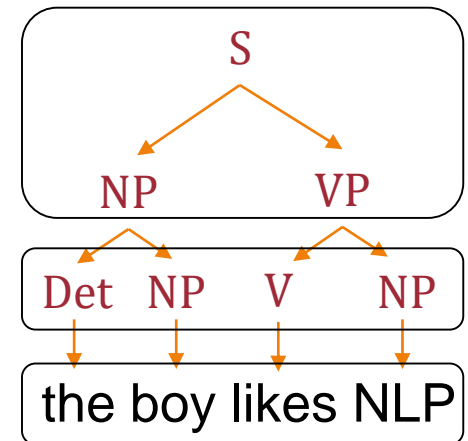
- ▶ Time complexity

- ▶ L SHIFTs
- ▶ L UNARY-Xs
- ▶ $(L - 1)$ REDUCE-Xs
- ▶ Total: $3L - 1$
 - ▶ **Linear time!**

3 REDUCE-Xs

4 UNARY-Xs

4 SHIFTs



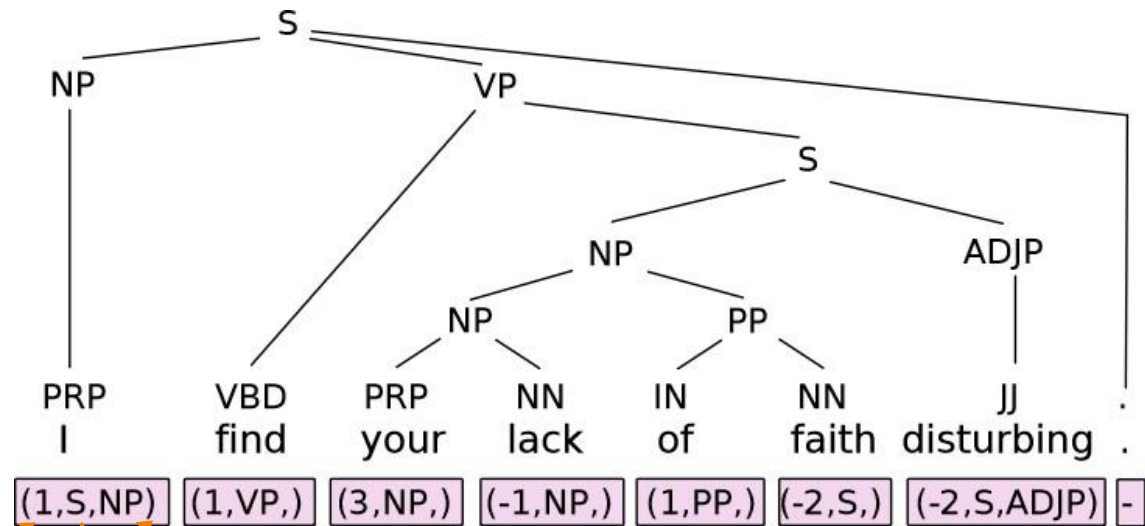
- ▶ Can also be applied to dependency parsing



Other Constituency Parsing Methods

Parsing as sequence labeling

- ▶ Cast constituency parsing as a sequence labeling task
 - ▶ Advantage: faster parsing speed



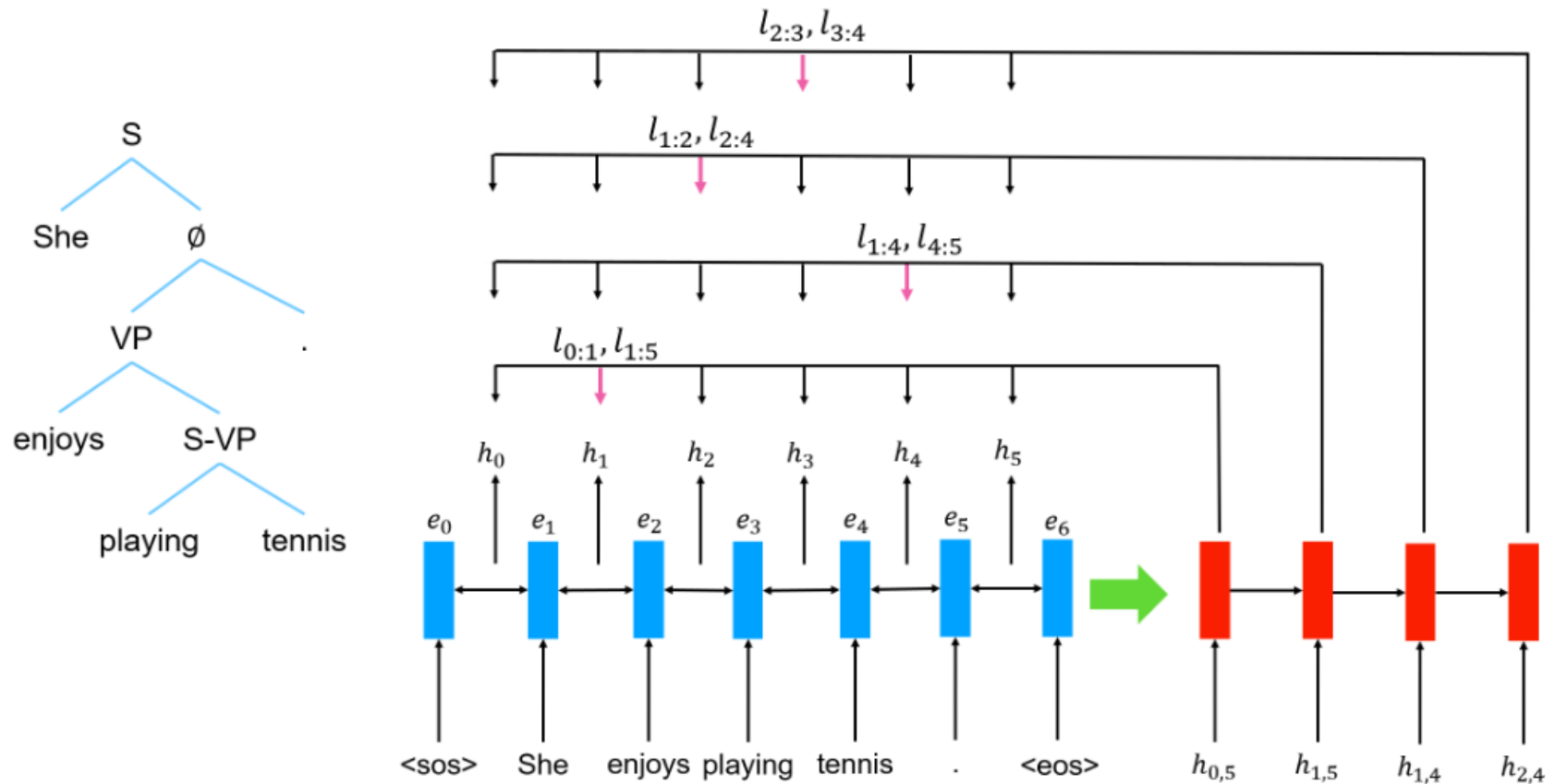
(relative) # of common ancestor shared with the next word

lowest common ancestor (LCA) to the next word

unary chain connecting the LCA and the current word

Top-down splitting

- ▶ Recursively split the input sentence into a binary tree





Summary



Constituency Parsing

- ▶ Concepts, evaluation
- ▶ Span-based Parsing
 - ▶ Tree score = sum of constituent scores
 - ▶ Parsing: CYK
- ▶ (Probabilistic) Context-Free Grammars
 - ▶ Tree score = product of rule probabilities
 - ▶ Parsing: CYK
 - ▶ Learning
 - ▶ Supervised: generative & discriminative methods
 - ▶ Unsupervised: EM with inside-outside algorithm
- ▶ Transition-based parsing
 - ▶ Tree score = product of action probabilities
 - ▶ Bottom-up parsing

