CS769 Advanced NLP

Syntactic Parsing I: Constituency Grammar

Junjie Hu

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Slides adapted from Bob, Hao, Dan

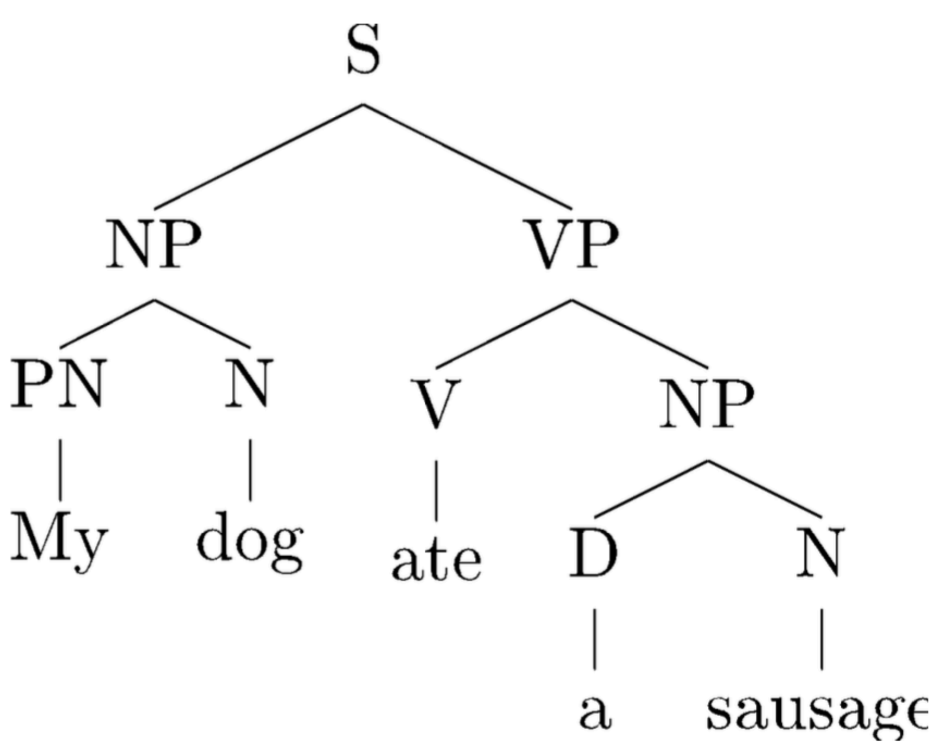
https://junjiehu.github.io/cs769-fall23/

Goals for Today

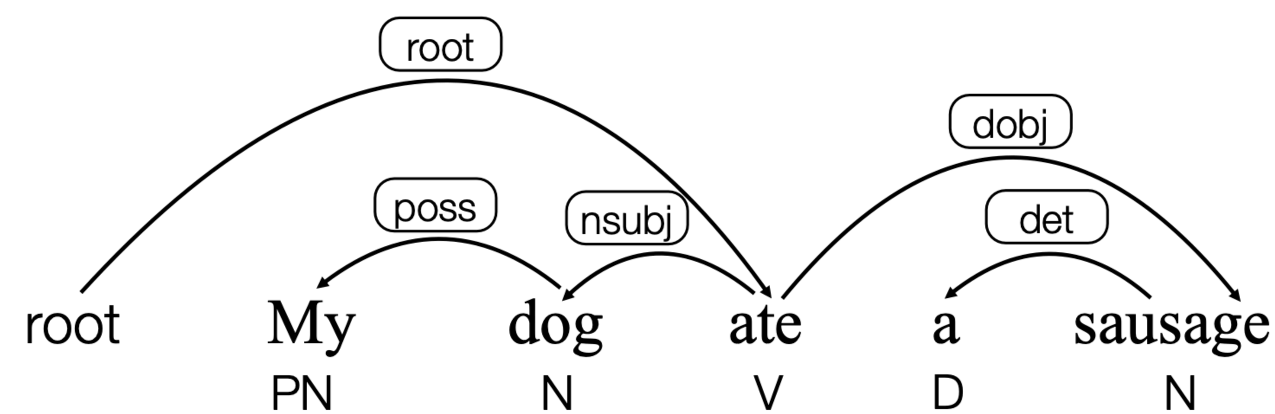
- Syntactic Parsing
- Probabilistic Context-Free Grammar (PCFG)
- **Supervised PCFG** (**Generative**)
- **CYK Decoding Algorithm**
- **Supervised Span-based Neural Models** (**Discriminative**)

Syntactic Parsing

- The process of predicting **syntactic representations**
- Two types of linguistic structures:



Constituency (aka phrase structure) tree:
Focus on the structure of the sentence

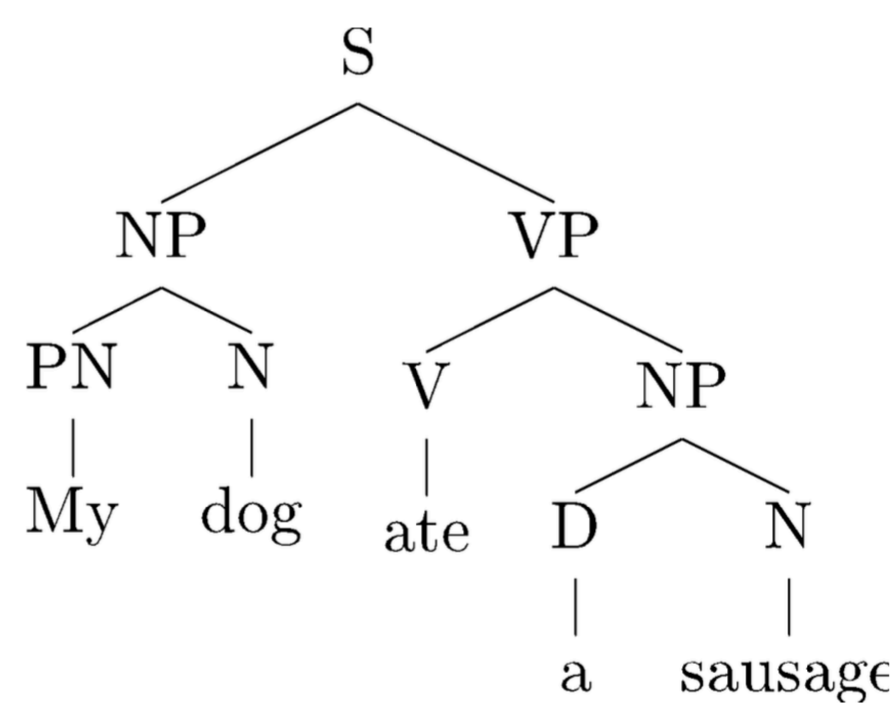


Dependency tree:
Focus on relations between words

Constituency Trees

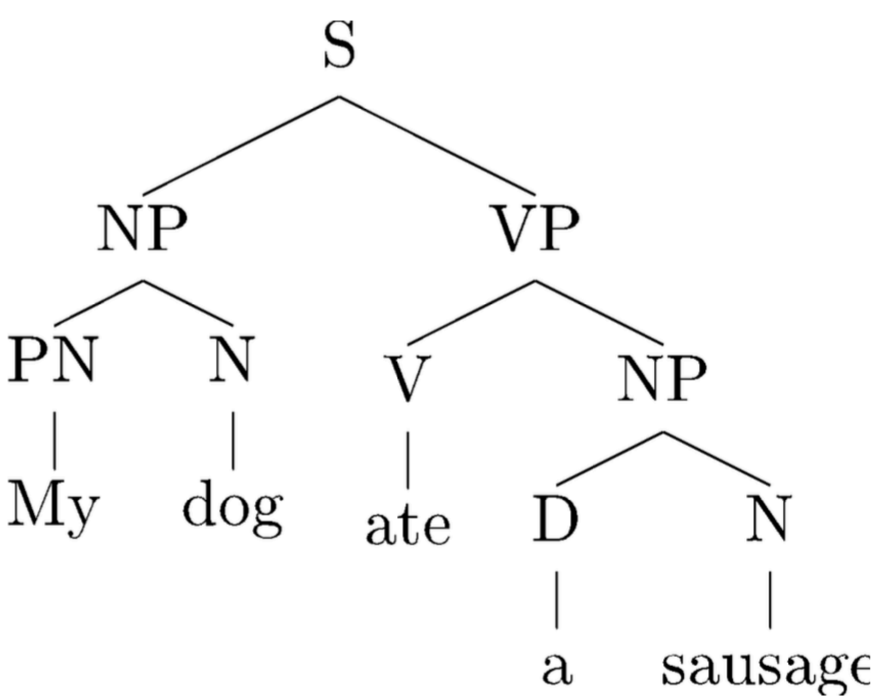
- Internal nodes (or non-terminals) correspond to phrases
  - S: a sentence
  - NP (noun phrase): My dog, a sandwich, …
  - VP (verb phrase): ate a sausage, …
  - PP (prepositional phrases): with a friend, in a car, …

- Nodes immediately above words are part-of-speech tags (or preterminals).
  - PN: pronoun
  - D: determiner
  - V: verb
  - N: noun
  - P: preposition



Bracketing notation

- Often convenient to represent a tree as **a bracketed sequence**:
- In principle, constituency tree can be an n-nary tree, however, it is easy to convert it to a binary tree (by adding a null non-terminal Ø). By convention, we often just represent the structure as a binary tree.

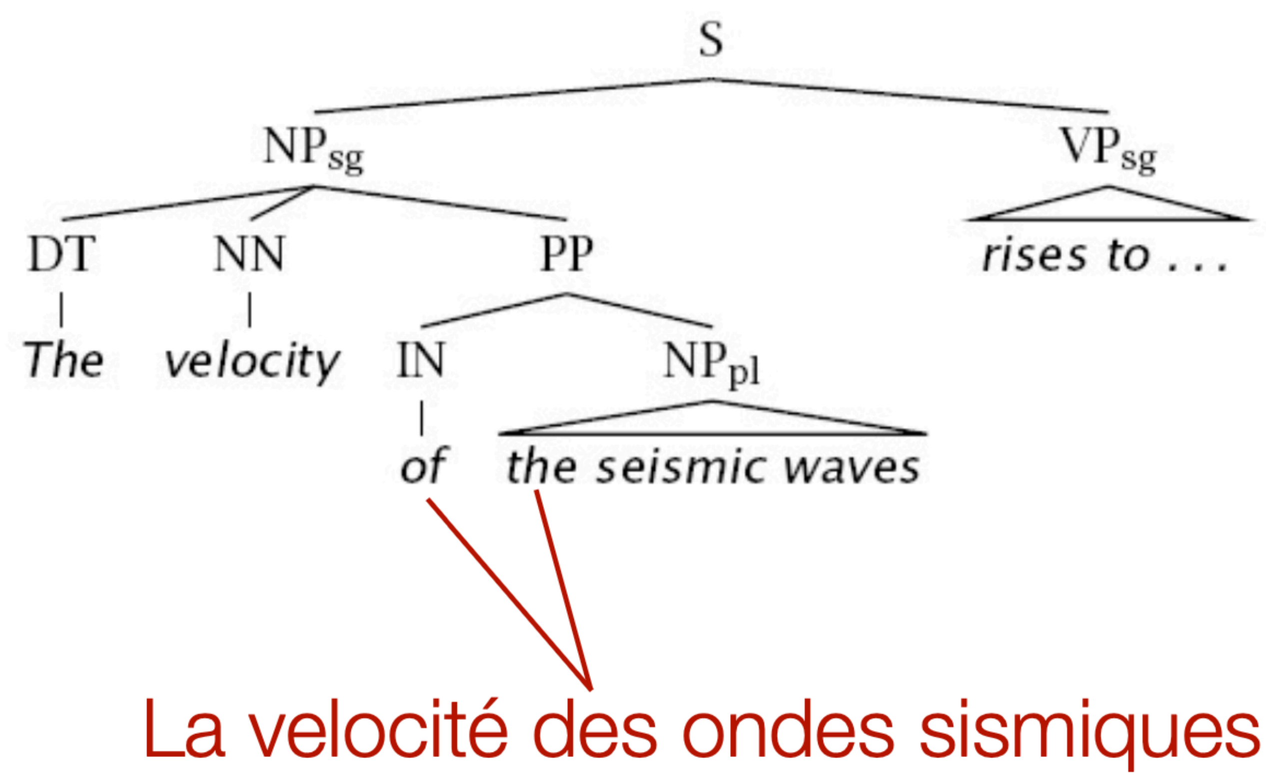



```
(S
   (NP (PN My) (N dog) )
   (VP (V ate)
      (NP (D a) (N sausage) )
   )
)
```

Constituency is not always clear

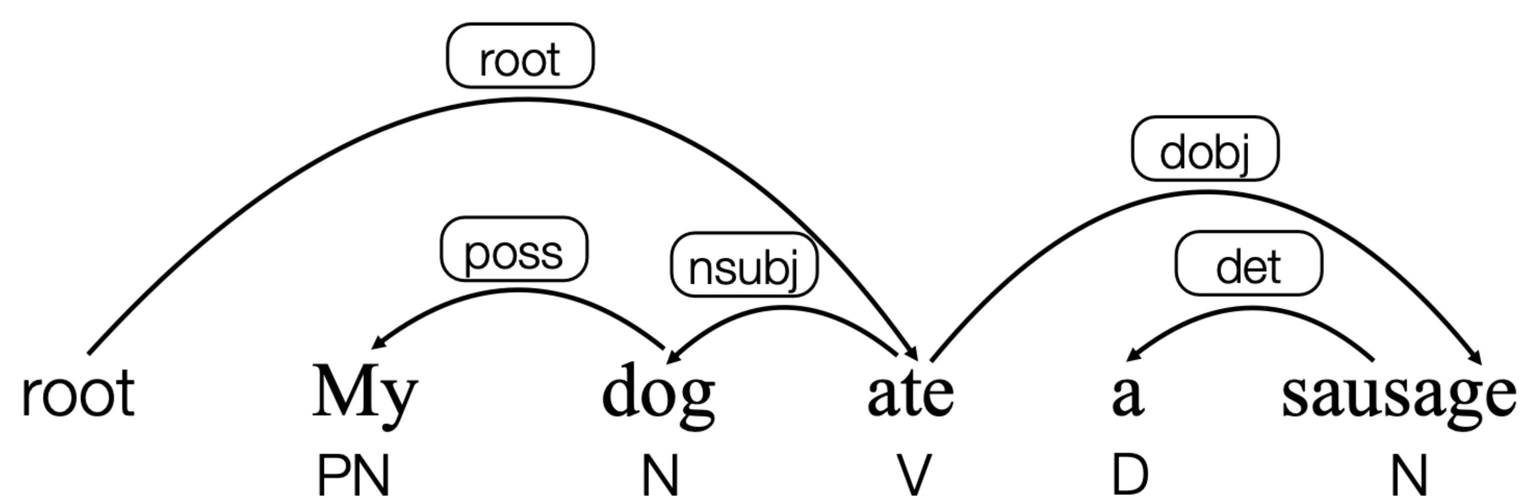
- Coordination:
  - Example: He went to and came from the store.

- Phonological reduction:
  - I will go → I'll go
  - I want to go → I wanna go
  - A le centre → au centre



Dependency Trees

- Nodes are words (along with part-of-speech tags)
- Directed arcs encode syntactic dependencies between words
- Labels are types of relations between words:
  - **root**: root of the tree, usually points to a verb
  - **poss**: possessive
  - **dobj**: direct object
  - **nsub**: (noun) subject
  - **det**: determiner

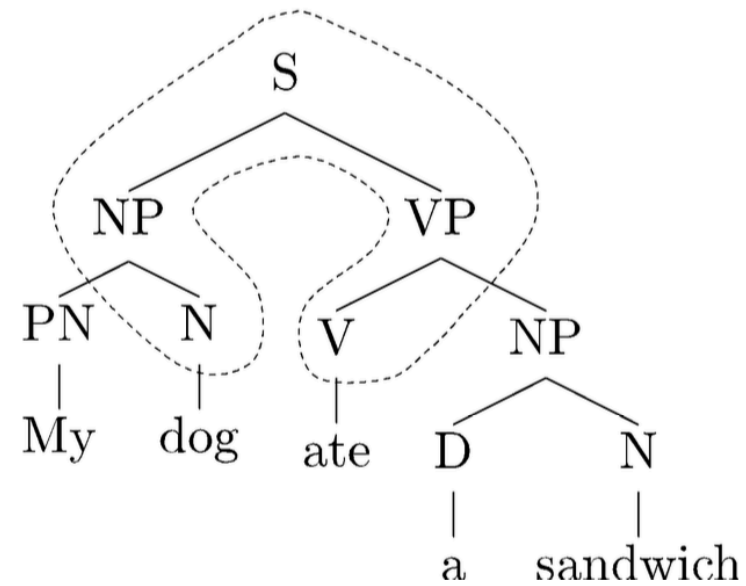


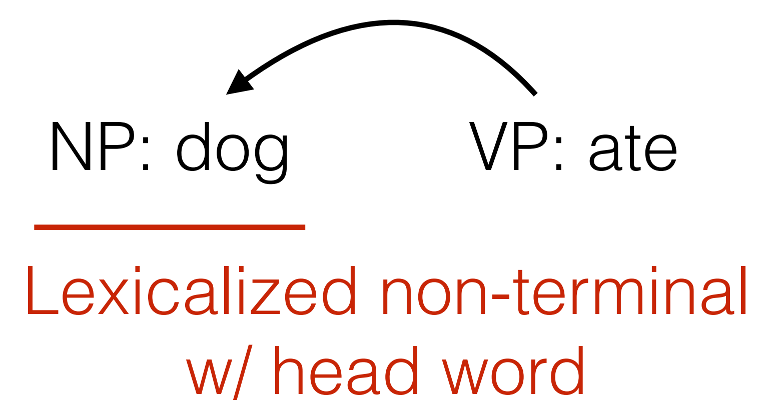
Dependency parsing

- Recover shallow semantics
- Shallow semantic information can be (approximately) derived from syntactic information
  - Subjects (nsubj) are often **agents**: *initiators / doers of an action*
  - Direct objects (dobj) are often **patients**: *affected entities*
- But not always true. Even for agents and patients, consider:
  - Mary is baking a cake in the oven
  - A cake is baking in the oven
- In general, it is not trivial even for the most shallow forms of semantics
  - e.g., prepositions: ***in*** can encode direction, position, temporal information, …

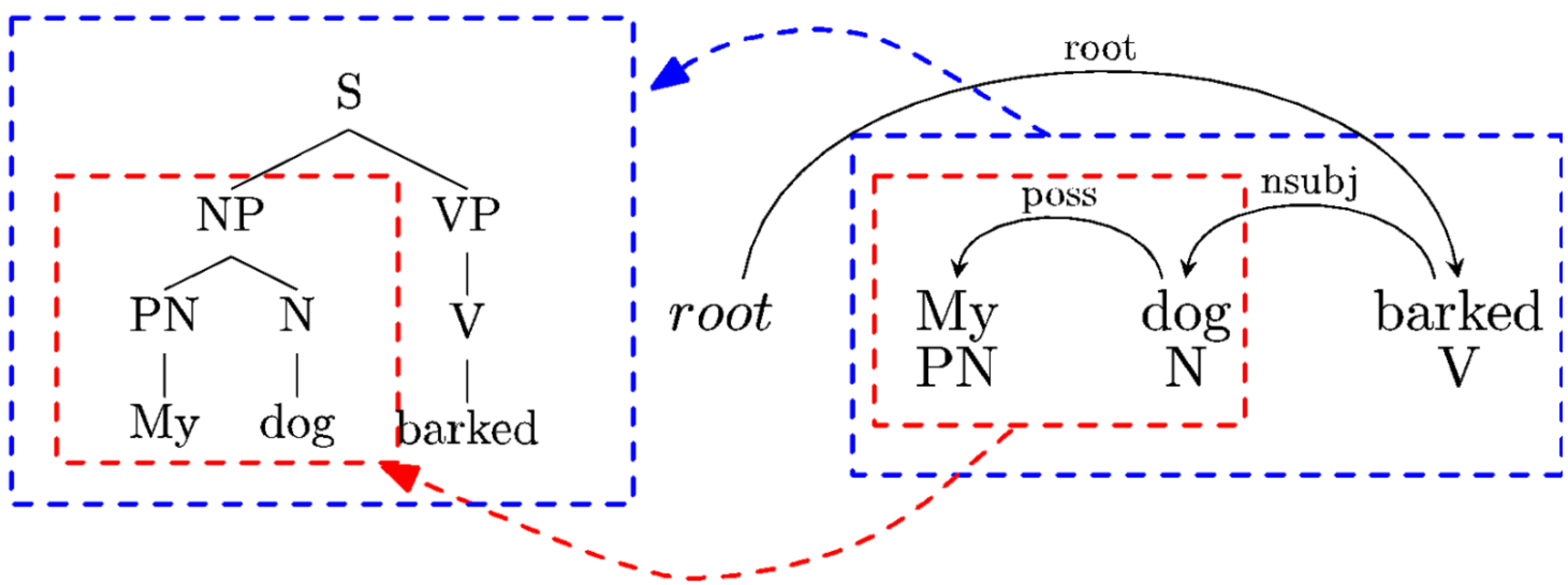
Constituency ↔ Dependency

- Constituency trees can (potentially) → dependency trees





- Dependency trees can (potentially) → constituency trees



Context Free Grammar (CFG)
& Probabilistic CFG

Context-free grammars (CFG)

- Context-free grammars (CFG): a formalism for parsing.

Grammar (CFG)

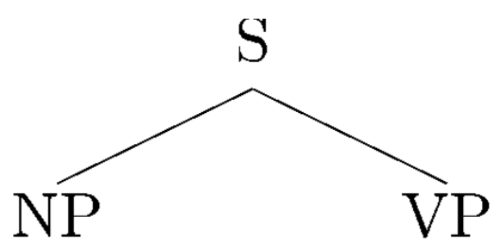
ROOT → S	NP → NP PP
S → NP VP	VP → VBP NP
NP → DT NN	VP → VBP NP PP
NP → NN NNS	PP → IN NP

Lexicon

NN → interest
NNS → raises
VBP → interest
VBP → raises
…

- Other (non-CF) grammar formalism: LFG, HPSG, TAG, CCG, …

CFG for Syntactic Parsing



Grammar (CFG)

$\boxed{S \rightarrow NP\ VP}$

VP → V
VP → V NP
VP → VP PP

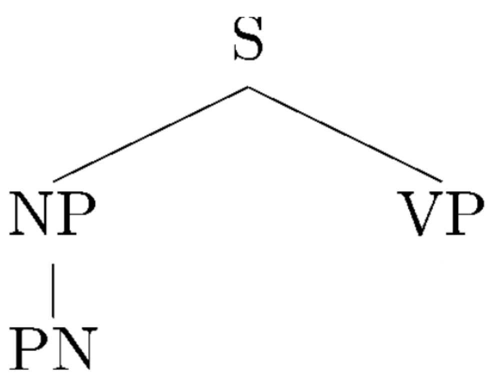
NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
VP → V NP
VP → VP PP

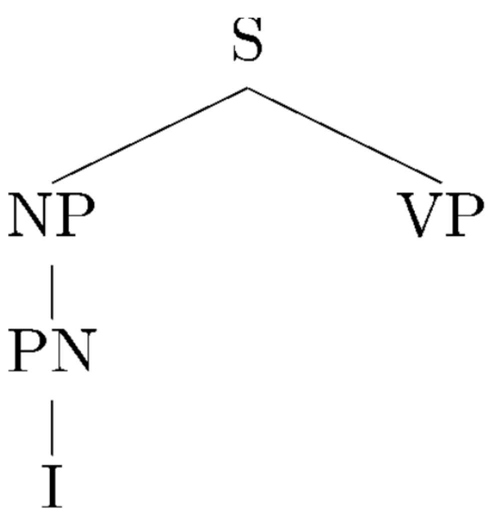
NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
$\boxed{\text{VP} \rightarrow \text{V NP}}$
VP → VP PP

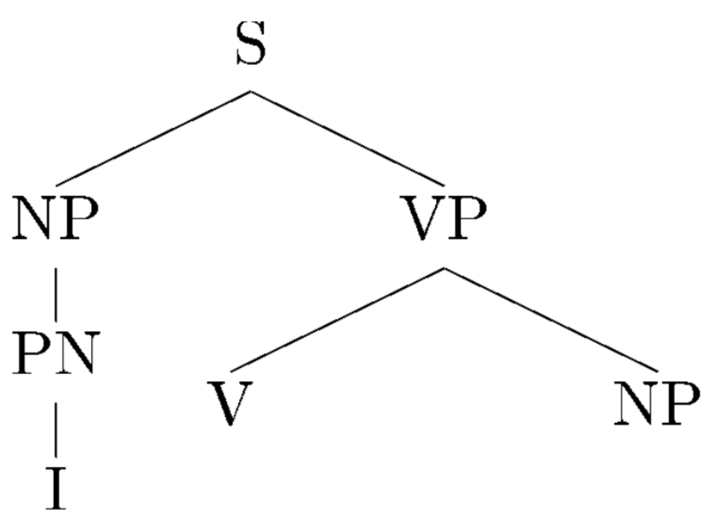
NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
$\boxed{\text{PN} \rightarrow \text{I}}$
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
VP → V NP
VP → VP PP

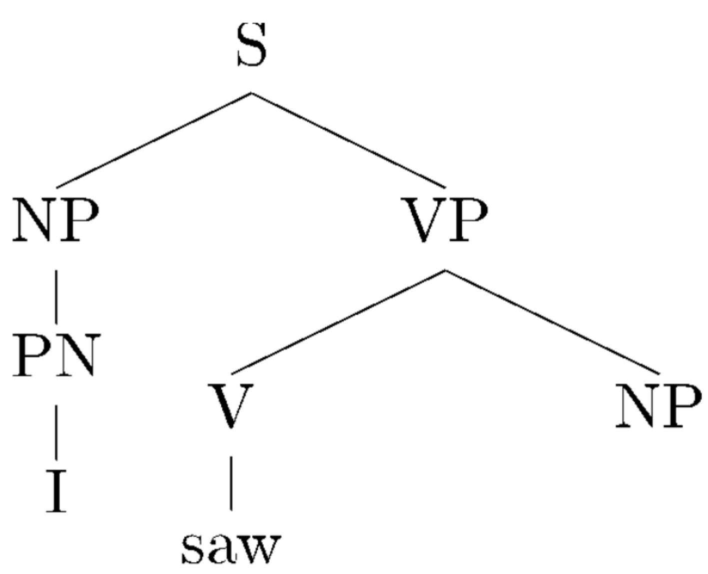
NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
VP → V NP
VP → VP PP

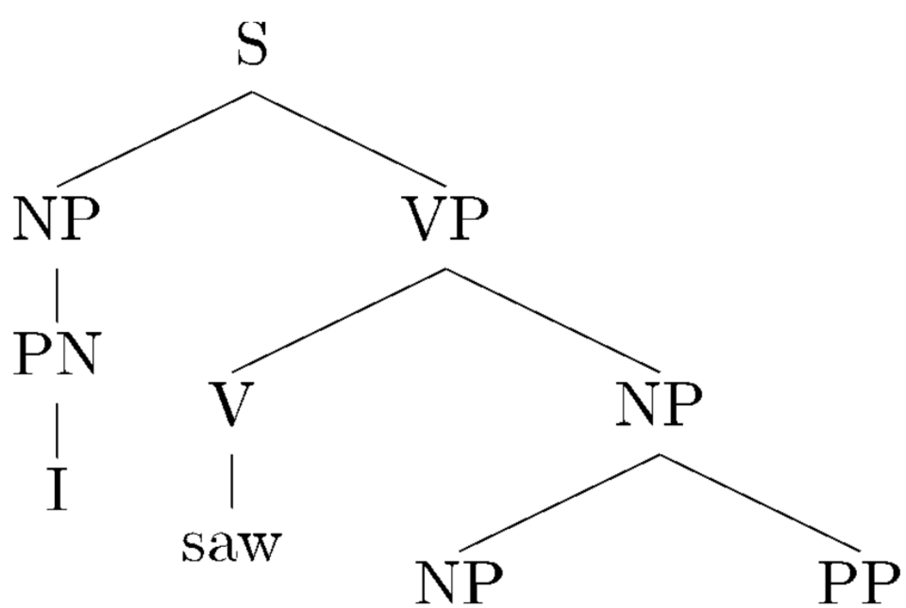
NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
VP → V NP
VP → VP PP

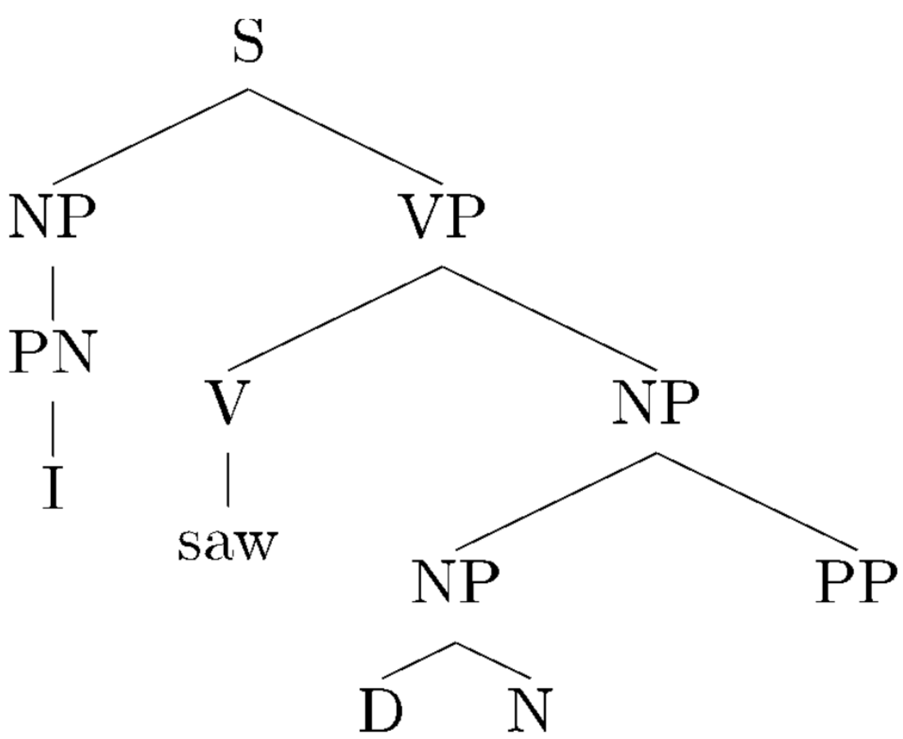
$\boxed{\text{NP} \rightarrow \text{NP PP}}$
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
VP → V NP
VP → VP PP

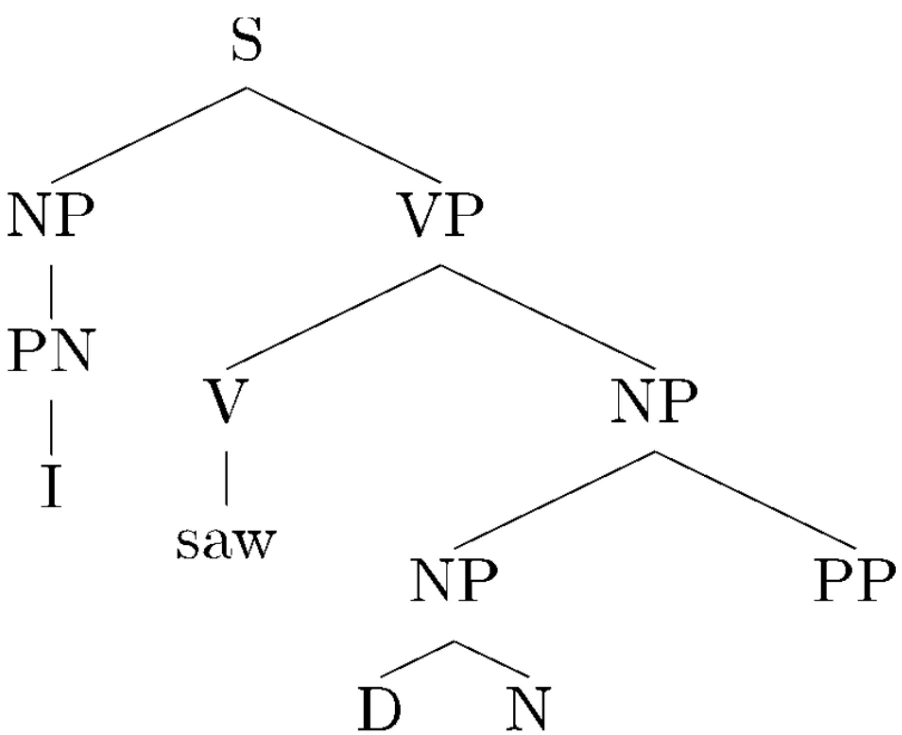
NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
VP → V NP
VP → VP PP

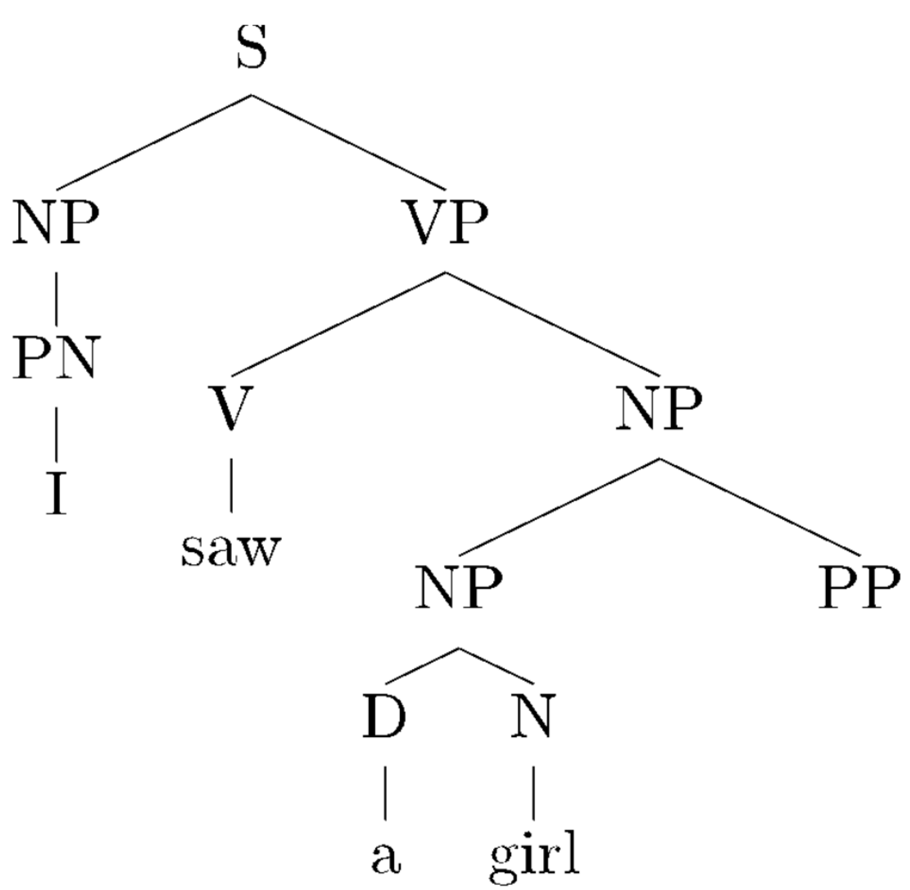
NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
VP → V NP
VP → VP PP

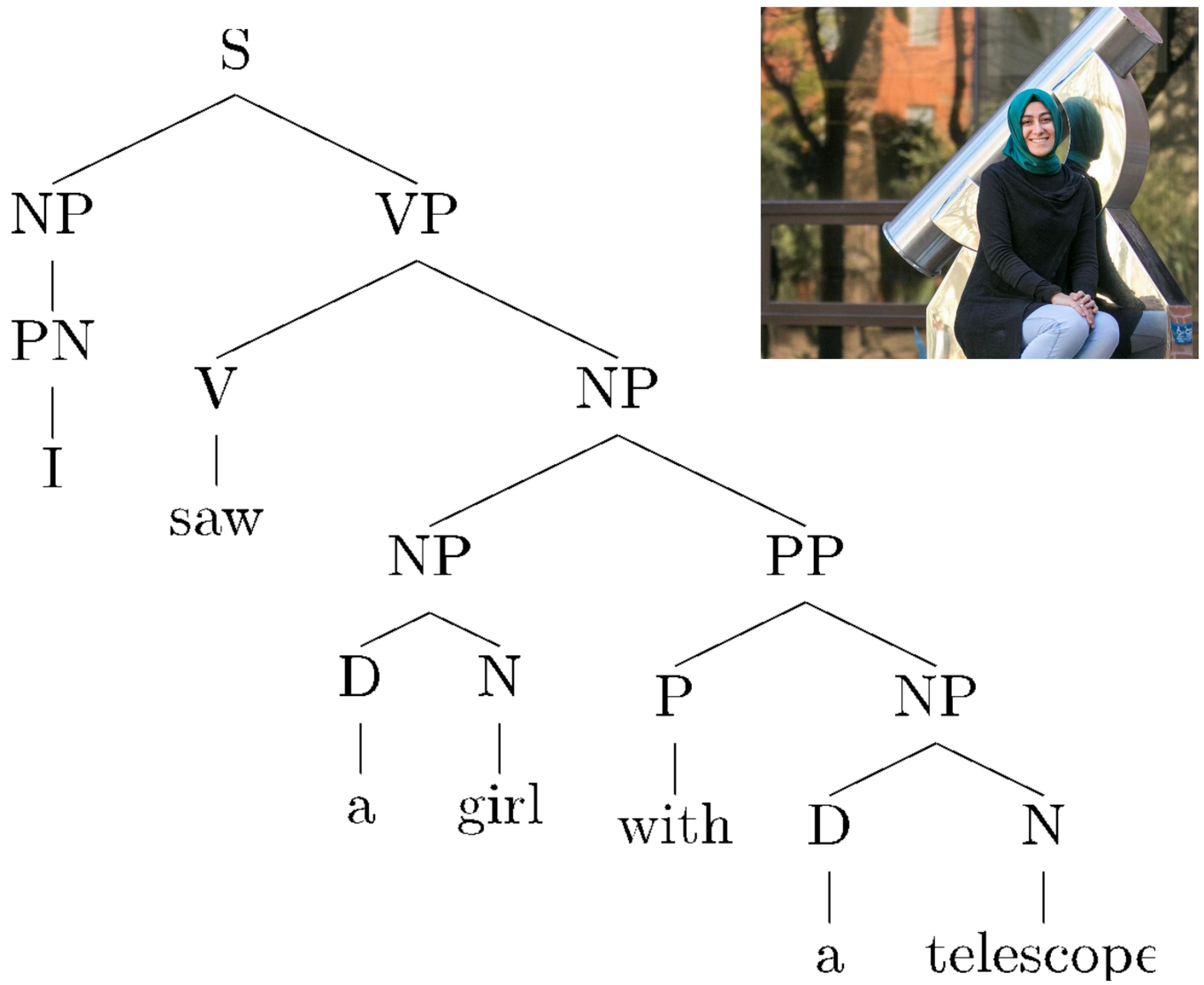
NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

CFG for Syntactic Parsing



Grammar (CFG)

S → NP VP

VP → V
VP → V NP
VP → VP PP

NP → NP PP
NP → D N
NP → PN

PP → P NP

Lexicon

N → girl
N → telescope
N → sandwich
PN → I
V → saw
V → ate
P → with
P → in
D → a
D → the

Probabilistic context-free grammars (PCFG)

- **CFG**: A 4-tupe (N, Σ, R, S):
  - N: a set of non-terminal symbols
  - Σ: a set of terminal symbols (disjoint from N)
  - S: a designated start symbol and a member of N
  - R: a set of rules, each of the form $\boxed{A \rightarrow s}$, where A is a non-terminal, s is a string of symbols, $A \in N, s \in (\Sigma \cup N)*$

$$\boxed{\begin{array}{r} S \rightarrow A, \\ A \rightarrow BC, \\ A \rightarrow \alpha, \end{array}} \quad \begin{array}{l} A \in N \\ A \in N, B, C \in N \cup \Sigma \\ \alpha \in \Sigma \end{array}$$

$\longleftarrow$ **Without loss of generality, only consider binary branching; Chomsky Normal Form**

- **PCFG** adds a top-down production probability per rule.
  - Model the probability of each rule: $P(A \rightarrow s)$

$$\forall A \rightarrow s \in R : 0 \leq P(A \rightarrow s) \leq 1$$

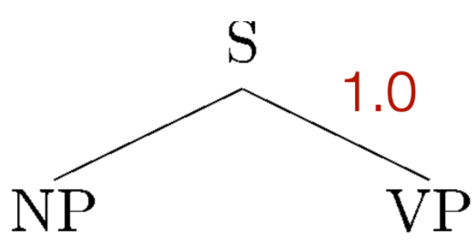
$$\forall A \in N : \sum_{s \text{ where } A \rightarrow s \in R} P(A \rightarrow s) = 1$$

PCFG (Example)

$S \rightarrow NP\ VP$	1.0	(NP a girl) (VP ate a sandwich)	$N \rightarrow$ *girl*	0.2
			$N \rightarrow$ *telescope*	0.7
$VP \rightarrow V$	0.2			
$VP \rightarrow V\ NP$	0.4	(V ate) (NP a sandwich)	$N \rightarrow$ *sandwich*	0.1
$VP \rightarrow VP\ PP$	0.4	(VP saw a girl) (PP with a telescope)	$PN \rightarrow$ *I*	1.0
			$V \rightarrow$ *saw*	0.5
$NP \rightarrow NP\ PP$	0.3	(NP a girl) (PP with a sandwich)	$V \rightarrow$ *ate*	0.5
$NP \rightarrow D\ N$	0.5	(D a) (N sandwich)	$P \rightarrow$ *with*	0.6
$NP \rightarrow PN$	0.2		$P \rightarrow$ *in*	0.4
			$D \rightarrow$ *a*	0.3
$PP \rightarrow P\ NP$	1.0	(P with) (NP a sandwich)	$D \rightarrow$ *the*	0.7

Now we can score a tree as a product of probabilities corresponding to the used rules!

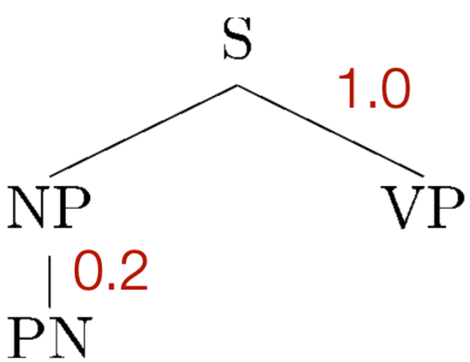
PCFG (Example)



S → NP VP	1.0	N → *girl*	0.2
		N → *telescope*	0.7
VP → V	0.2		
VP → V NP	0.4	N → *sandwich*	0.1
VP → VP PP	0.4	PN → *I*	1.0
		V → *saw*	0.5
NP → NP PP	0.3	V → *ate*	0.5
NP → D N	0.5	P → *with*	0.6
NP → PN	0.2	P → *in*	0.4
		D → *a*	0.3
PP → P NP	1.0	D → *the*	0.7

$P(T) = 1.0$ *

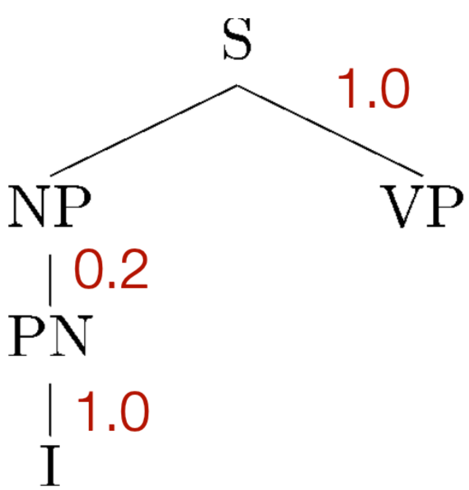
PCFG (Example)



S → NP VP	1.0	N → *girl*	0.2
		N → *telescope*	0.7
VP → V	0.2		
VP → V NP	0.4	N → *sandwich*	0.1
VP → VP PP	0.4	PN → *I*	1.0
		V → *saw*	0.5
NP → NP PP	0.3	V → *ate*	0.5
NP → D N	0.5	P → *with*	0.6
NP → PN	0.2	P → *in*	0.4
		D → *a*	0.3
PP → P NP	1.0	D → *the*	0.7

$P(T) = 1.0 * 0.2 *$

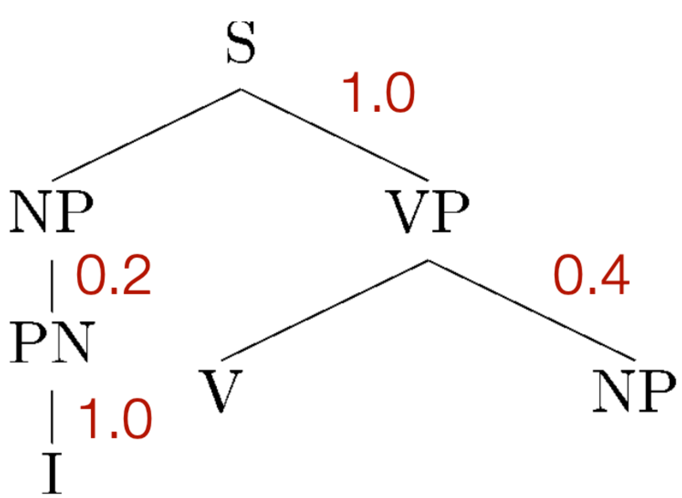
PCFG (Example)



S → NP VP	1.0	N → *girl*	0.2
		N → *telescope*	0.7
VP → V	0.2		
VP → V NP	0.4	N → *sandwich*	0.1
VP → VP PP	0.4	PN → *I*	1.0
		V → *saw*	0.5
NP → NP PP	0.3	V → *ate*	0.5
NP → D N	0.5	P → *with*	0.6
NP → PN	0.2	P → *in*	0.4
		D → *a*	0.3
PP → P NP	1.0	D → *the*	0.7

$P(T)$ = 1.0 * 0.2 * 1.0 *

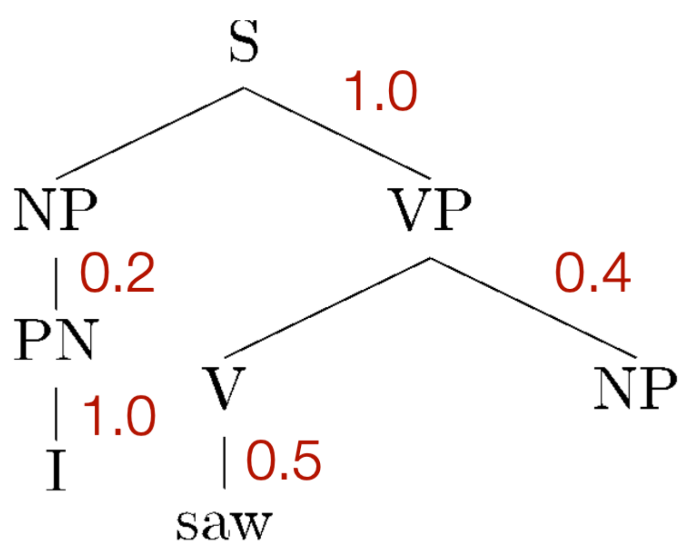
PCFG (Example)



S → NP VP	1.0	N → *girl*	0.2
VP → V	0.2	N → *telescope*	0.7
VP → V NP	0.4	N → *sandwich*	0.1
VP → VP PP	0.4	PN → *I*	1.0
		V → *saw*	0.5
NP → NP PP	0.3	V → *ate*	0.5
NP → D N	0.5	P → *with*	0.6
NP → PN	0.2	P → *in*	0.4
		D → *a*	0.3
PP → P NP	1.0	D → *the*	0.7

$P(T)$ = 1.0 * 0.2 * 1.0 * 0.4 *

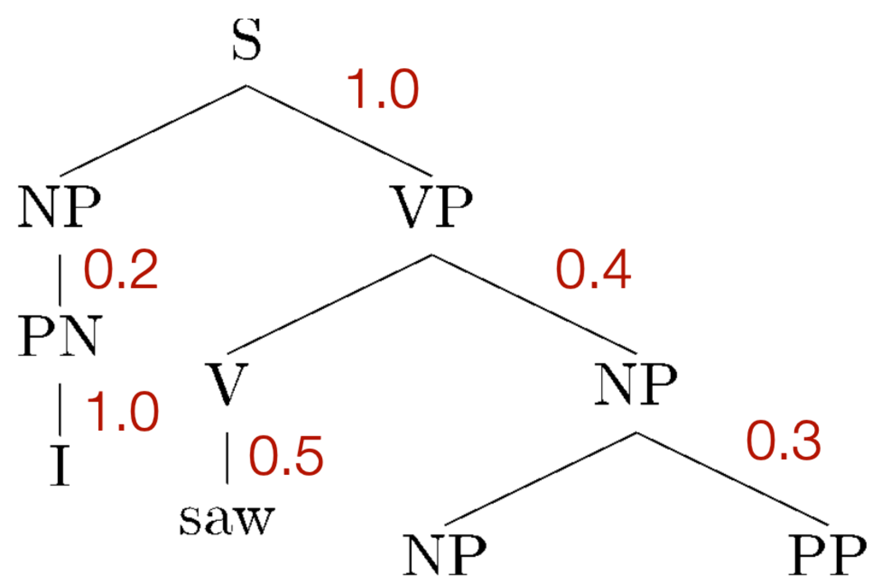
PCFG (Example)



S → NP VP	1.0	N → *girl*	0.2
		N → *telescope*	0.7
VP → V	0.2		
VP → V NP	0.4	N → *sandwich*	0.1
VP → VP PP	0.4	PN → *I*	1.0
		V → *saw*	0.5
NP → NP PP	0.3	V → *ate*	0.5
NP → D N	0.5	P → *with*	0.6
NP → PN	0.2	P → *in*	0.4
		D → *a*	0.3
PP → P NP	1.0	D → *the*	0.7

$P(T) = 1.0 * 0.2 * 1.0 * 0.4 * 0.5 *$

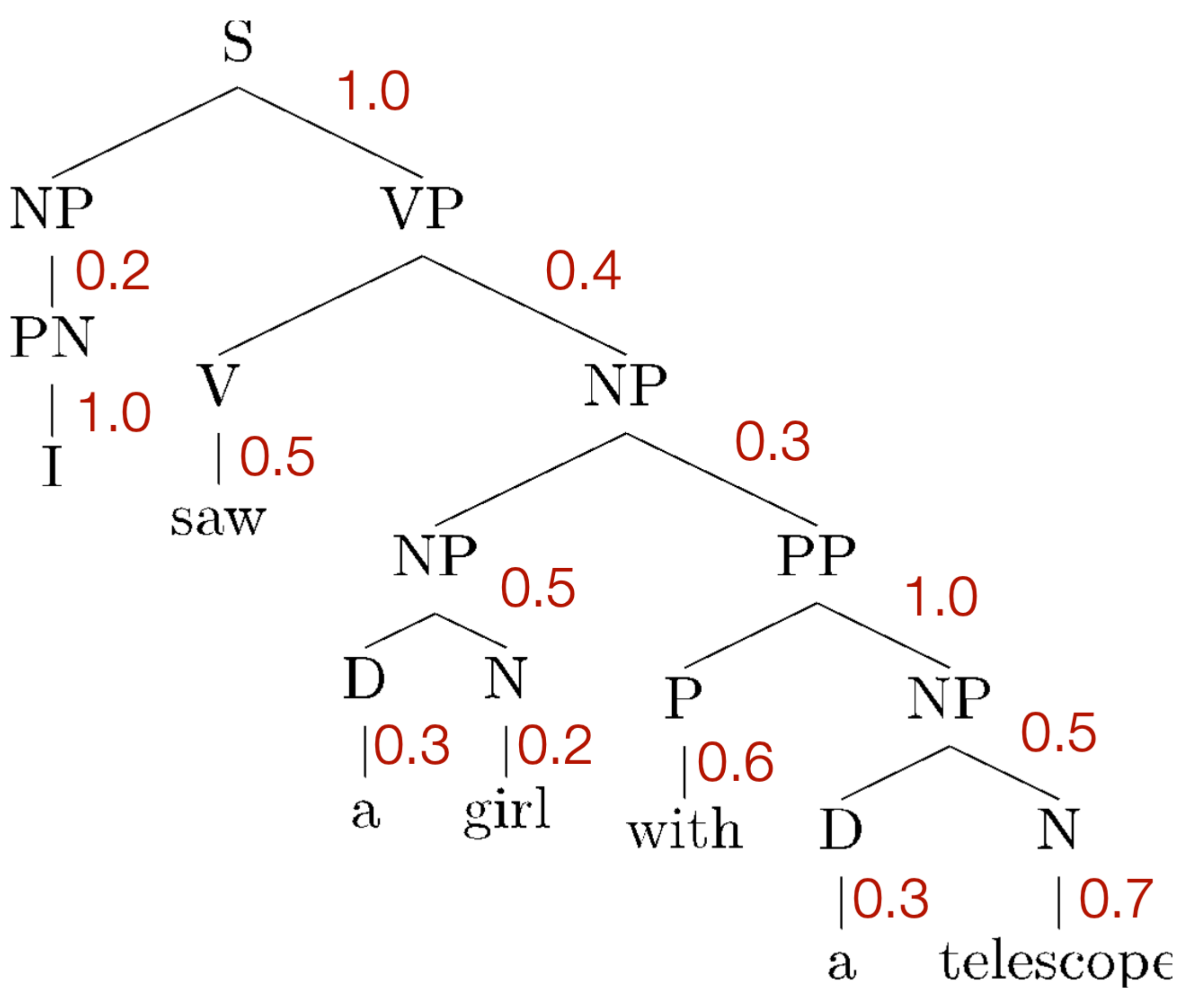
PCFG (Example)



S → NP VP	1.0	N → *girl*	0.2
VP → V	0.2	N → *telescope*	0.7
VP → V NP	0.4	N → *sandwich*	0.1
VP → VP PP	0.4	PN → *I*	1.0
		V → *saw*	0.5
NP → NP PP	0.3	V → *ate*	0.5
NP → D N	0.5	P → *with*	0.6
NP → PN	0.2	P → *in*	0.4
		D → *a*	0.3
PP → P NP	1.0	D → *the*	0.7

$P(T) = 1.0 * 0.2 * 1.0 * 0.4 * 0.5 * 0.3 *$

PCFG (Example)



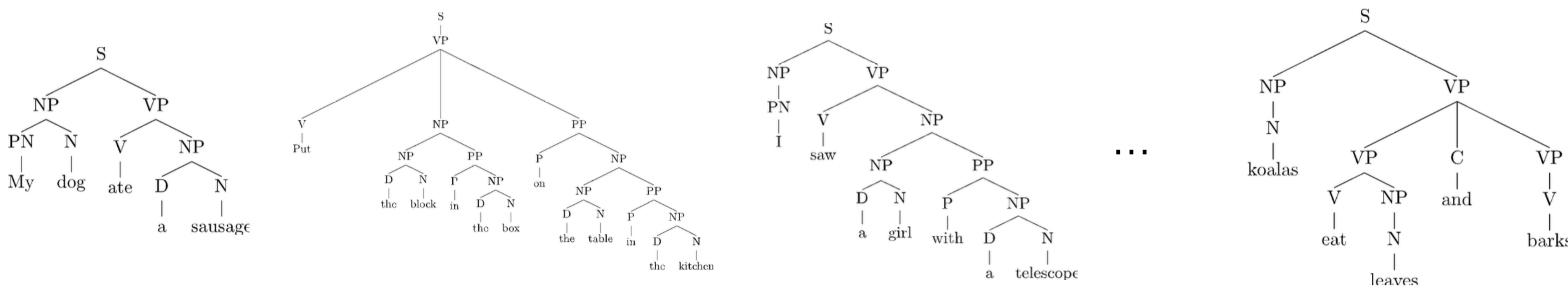
S → NP VP	1.0	N → *girl*	0.2
		N → *telescope*	0.7
VP → V	0.2		
VP → V NP	0.4	N → *sandwich*	0.1
VP → VP PP	0.4	PN → *I*	1.0
		V → *saw*	0.5
NP → NP PP	0.3	V → *ate*	0.5
NP → D N	0.5	P → *with*	0.6
NP → PN	0.2	P → *in*	0.4
		D → *a*	0.3
PP → P NP	1.0	D → *the*	0.7

$P(T)$ = 1.0 * 0.2 * 1.0 * 0.4 * 0.5 * 0.3 * 0.5 * 0.3 * 0.2 * 1.0 * 0.6 * 0.5 * 0.3 * 0.7 = 2.26e-5

PCFG Supervised Learning & Decoding

PCFG Supervised Learning

- A treebank: a collection of sentences annotated with constituency trees
  - Penn Treebank: (X, T) pairs



- PCFG: a generative model, maximizing the joint probability of a sentence given a tree.
  - If we constraint the search space to be all valid trees that can generate the sentence, this becomes:

$$\max P(X,T) = \max P(X|T)P(T) = \max_{T \in \mathrm{GEN}(X)} P(X|T)P(T)$$

PCFG Supervised Learning

- Estimate probability of each rule by maximum likelihood estimation:

$$P(T) = \sum_{A \to s \in R} P(A \to s), \quad T \in \mathrm{GEN}(X)$$

$$P(A \to s) = \frac{Count(A \to s)}{Count(A)}$$

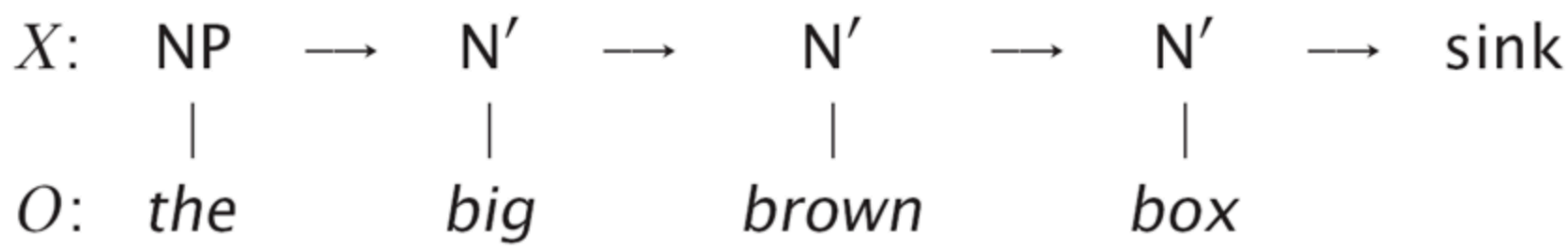
times the rule was used in the data

times the nonterminal was used in the data

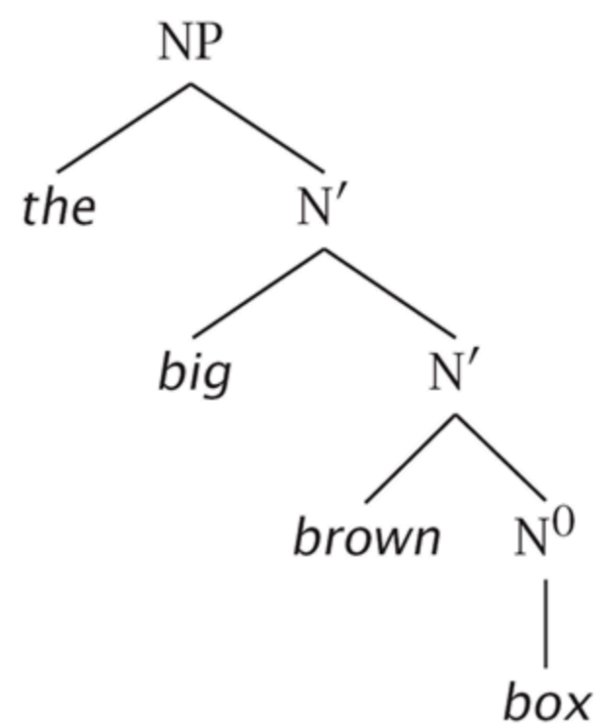
- Smoothing is helpful (esp. for rules that produce one word)
- If we don't have training data, use EM algorithm to estimate the probability

HMM vs PCFG

HMM: Linear Markov Chain

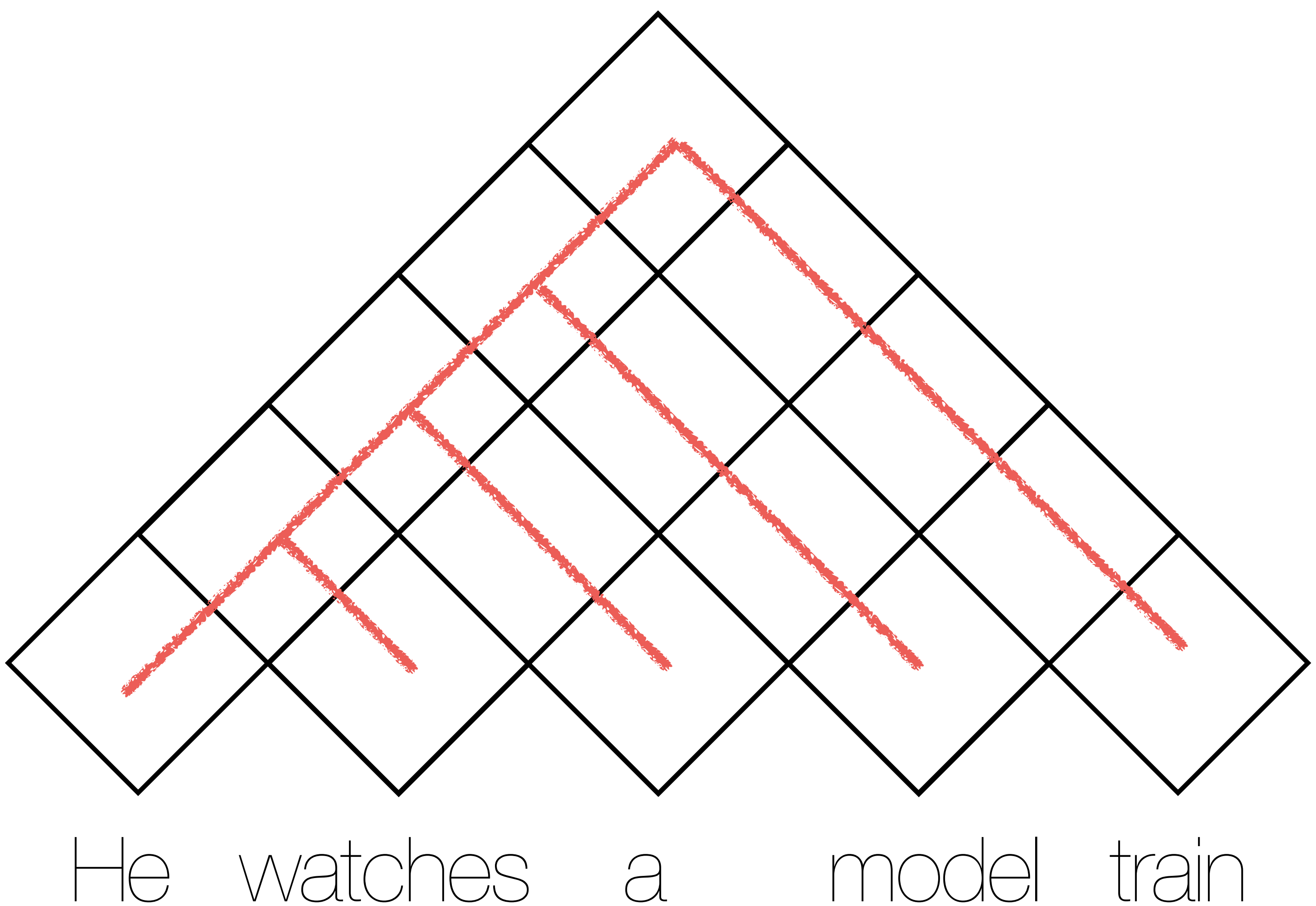


PCFG: tree



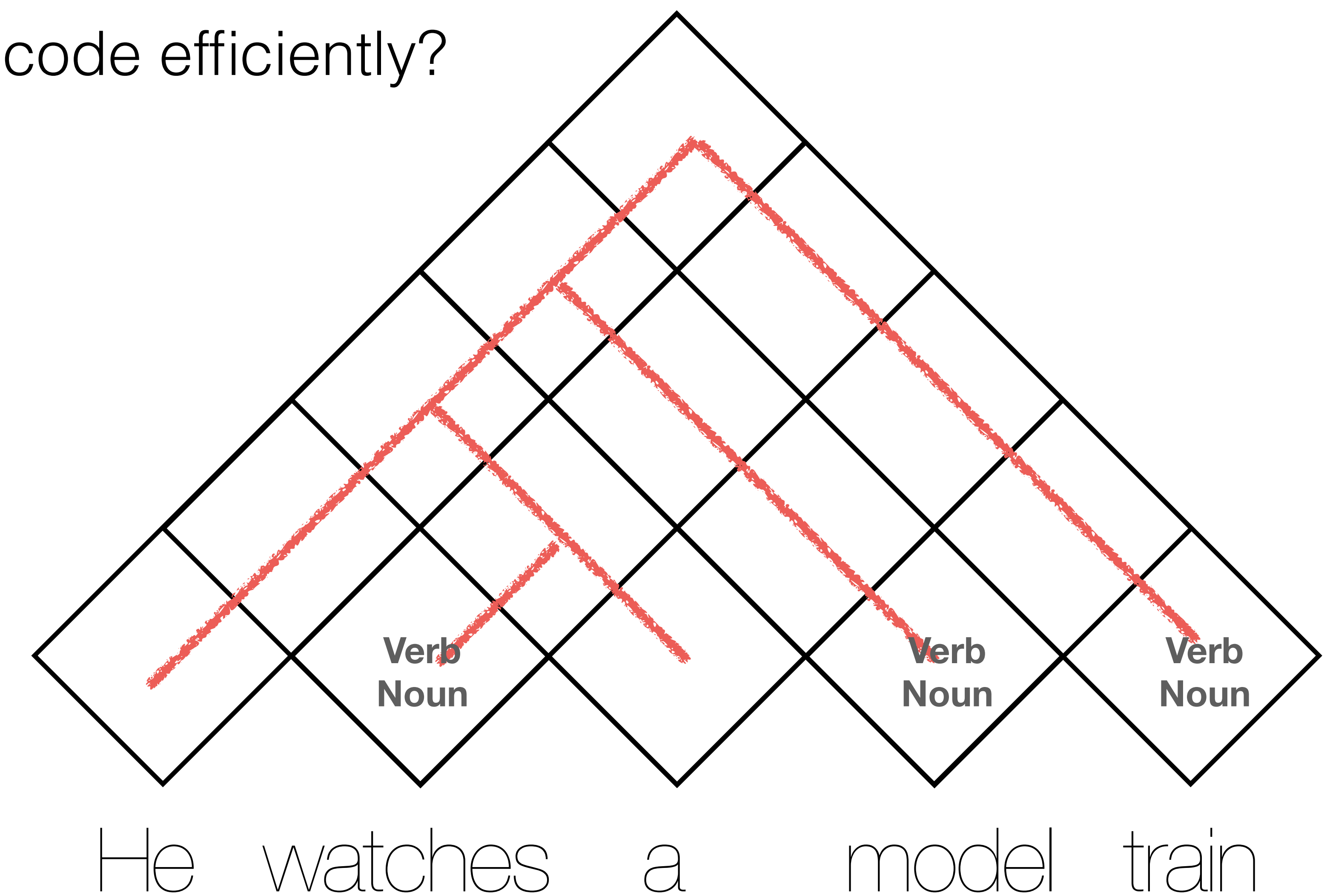
PCFG Decoding

- Brute force solution: enumerate all possible binary trees, score them, find the tree with maximum score



PCFG Decoding

- Brute force solution: enumerate all possible binary trees, score them, find the tree with maximum score
- For a sentence of n words, there are (n-1)! possible binary trees. Each word may have more than 1 possible POS tags
- How to decode efficiently?



PCFG Decoding: CYK Algorithm

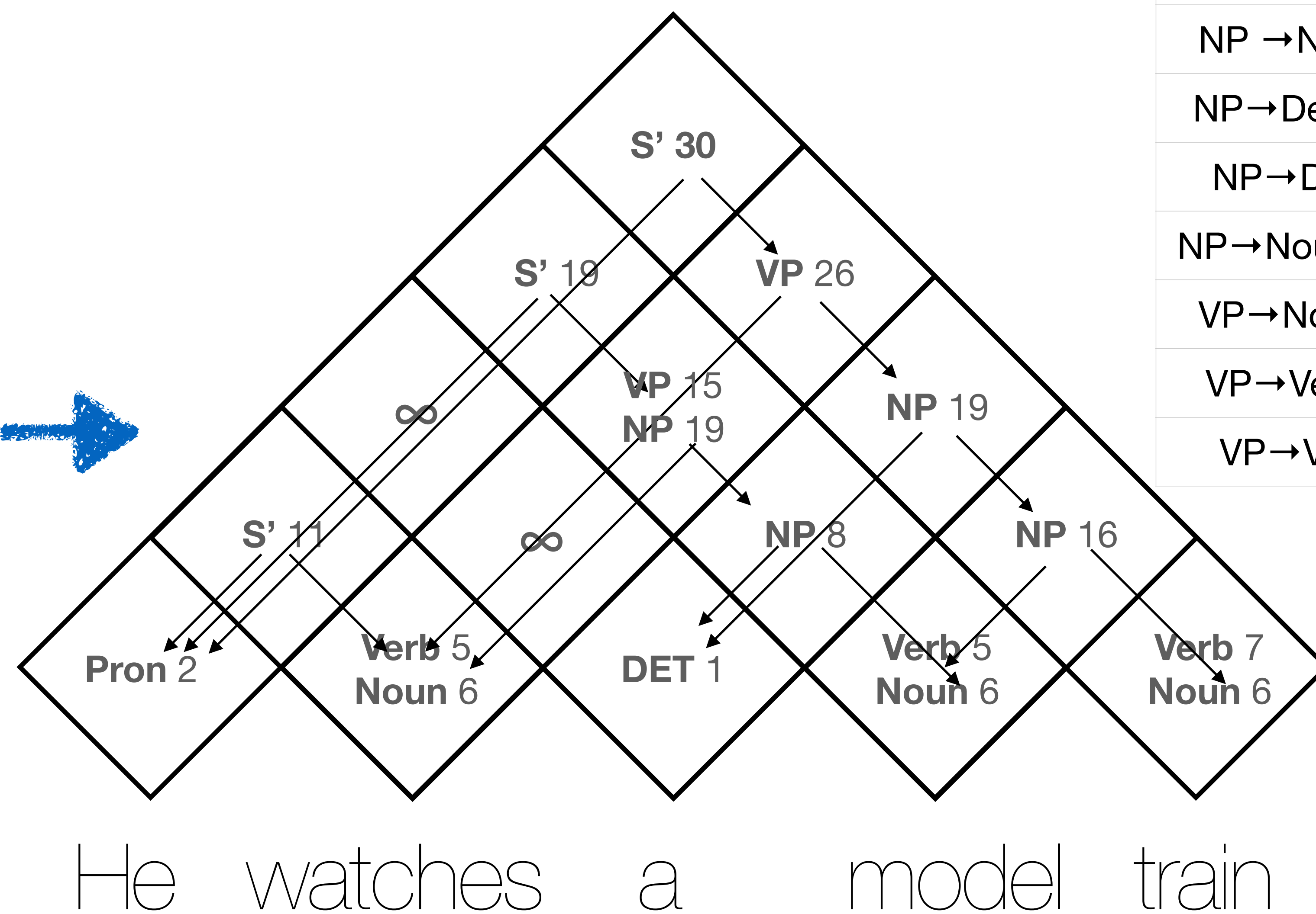
Bottom-up Dynamic Programming

Remember to store back-pointer!

Binary Rule	-log prob
S' →Pron Verb	4
S' →Pron VP	2
S' →NP VP	2
NP →NP Verb	5
NP→Det Noun	2
NP→Det NP	2
NP→Noun Noun	4
VP→Noun NP	5
VP→Verb NP	2
VP→VP NP	2

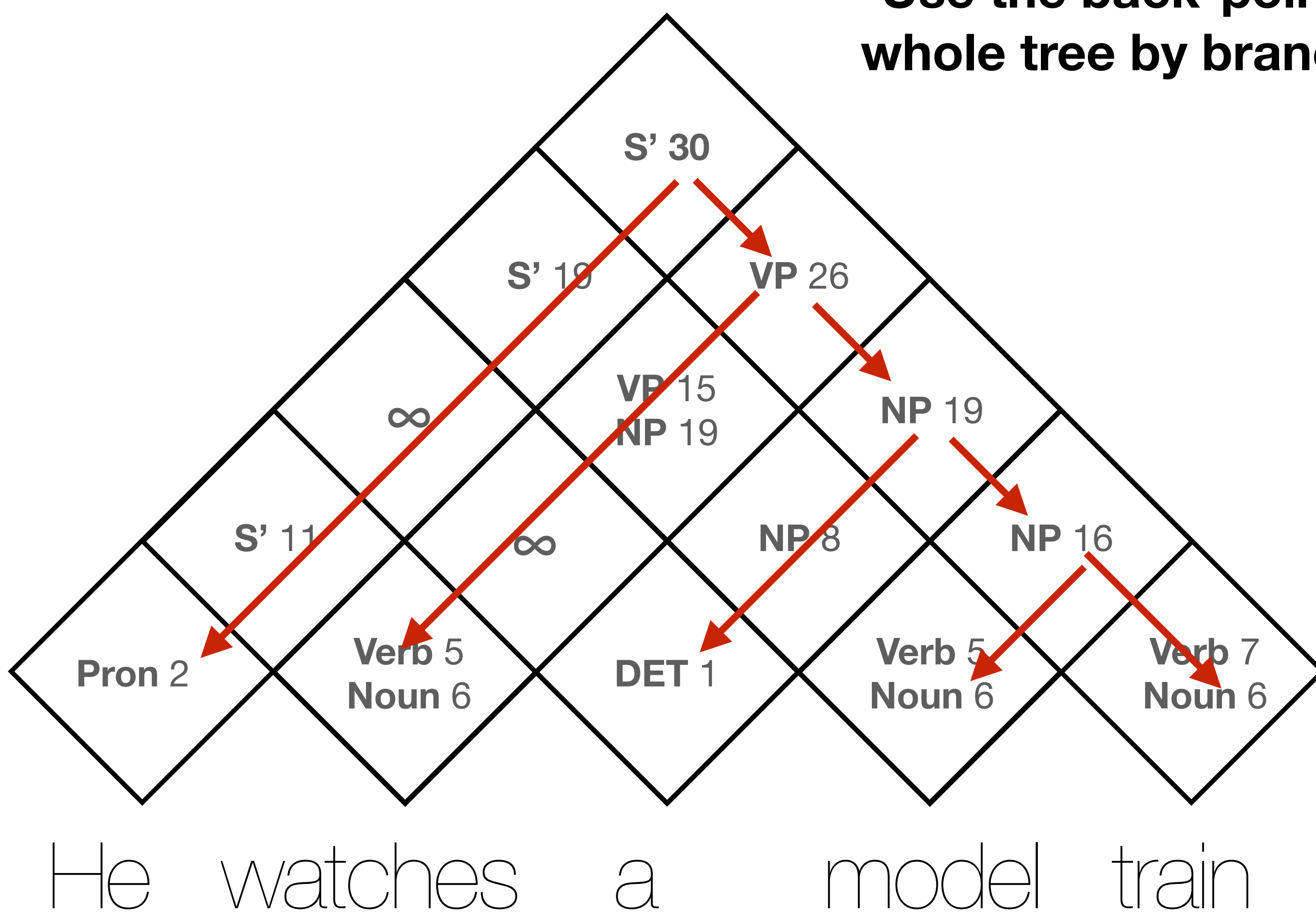
Binary rule
$A \to BC$

Lexical rule
(Unary rule)
$A \to \alpha, \alpha \in \Sigma$



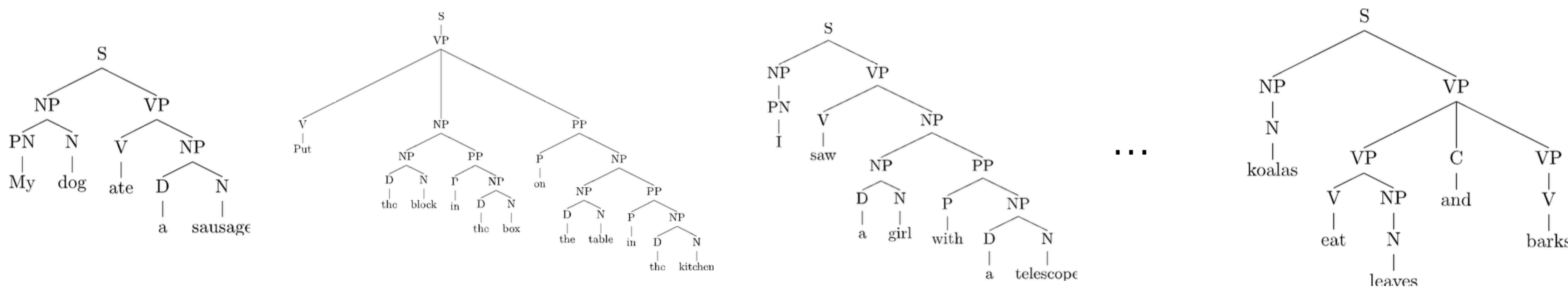
PCFG Decoding: CYK Algorithm

Use the back-pointer to build the whole tree by branching from root.



PCFG CYK Decoding

- A treebank: a collection of sentences annotated with constituency trees
    - Penn Treebank



- Estimate probability of each rule by maximum likelihood estimation:

$$P(A \to s) = \frac{Count(A \to s)}{Count(A)}$$

times the rule was used in the data

times the nonterminal was used in the data

- Smoothing is helpful (esp. for rules that produce one word)

PCFG Decoding: CYK Algorithm

- The score (or unnormalized probability) of a constituent with a non-terminal is often called inside probability
  - Computed by dynamic programming

$$s_{\text{label}}(i, j, A) = \max_{k,B,C} P(A \to BC) \times s_{\text{label}}(i, k, B) \times s_{\text{label}}(k+1, j, C)$$

  - The best optimal score of the whole sentence of length *n* is derived by

$$s_{\text{label}}(1, n, S)$$

Semiring Conversion

- The score (or unnormalized probability) of a constituent with a non-terminal is often called inside probability
  - Computed by dynamic programming
  - Numerically unstable

$$s_{\text{label}}(i, j, A) = \max_{k,B,C} P(A \to BC) \times s_{\text{label}}(i, k, B) \times s_{\text{label}}(k+1, j, C)$$

  - Define the minimum cost score, and rewrite the scores

$$s'_{\text{label}}(i, j, A) = -\log s_{\text{label}}(i, j, A)$$

$$s'_{\text{label}}(i, j, A) = \min_{k,B,C} (-\log P(A \to BC) + s'_{\text{label}}(i, k, B) + s'_{\text{label}}(k+1, j, C))$$

Semiring Parsing

"Add" $\oplus$ "Multiply" $\otimes$

$$s_{\text{label}}(i,j,A) = \boxed{\max_{k,B,C}} P(A \to BC) \boxed{\times} s_{\text{label}}(i,k,B) \boxed{\times} s_{\text{label}}(k+1,j,C)$$

$$s'_{\text{label}}(i,j,A) = \boxed{\min_{k,B,C}} (-\log P(A \to BC) \boxed{+} s'_{\text{label}}(i,k,B) \boxed{+} s'_{\text{label}}(k+1,j,C))$$

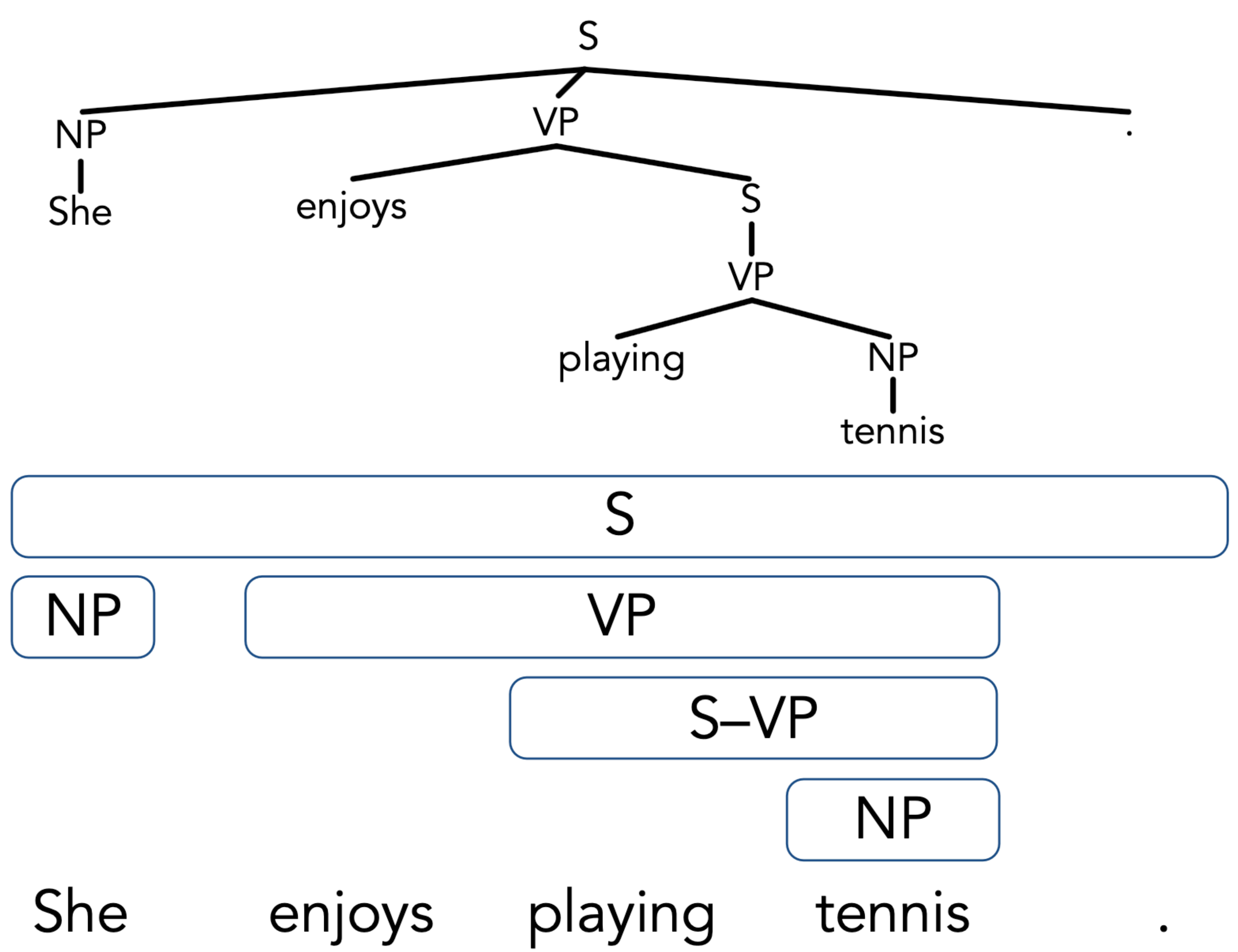
	weights	⊕	⊗	⓪	①
total prob	[0, 1]	+	x	0	1
max prob	[0, 1]	max	x	0	1
min -logp	[0, ∞]	min	+	∞	0
log prob	[-∞, 0]	logsumexp	+	-∞	0
recognizer	T/F	or	and	F	T

Semiring is an algebraic structure in ring theory: https://en.wikipedia.org/wiki/Semiring

Supervised Parsing:
Span-based Neural Models

Span-Based Parsing

$$P(Y_{i:j} = c | X_{i:j}) = w_c \cdot F_c(X_{i:j})$$



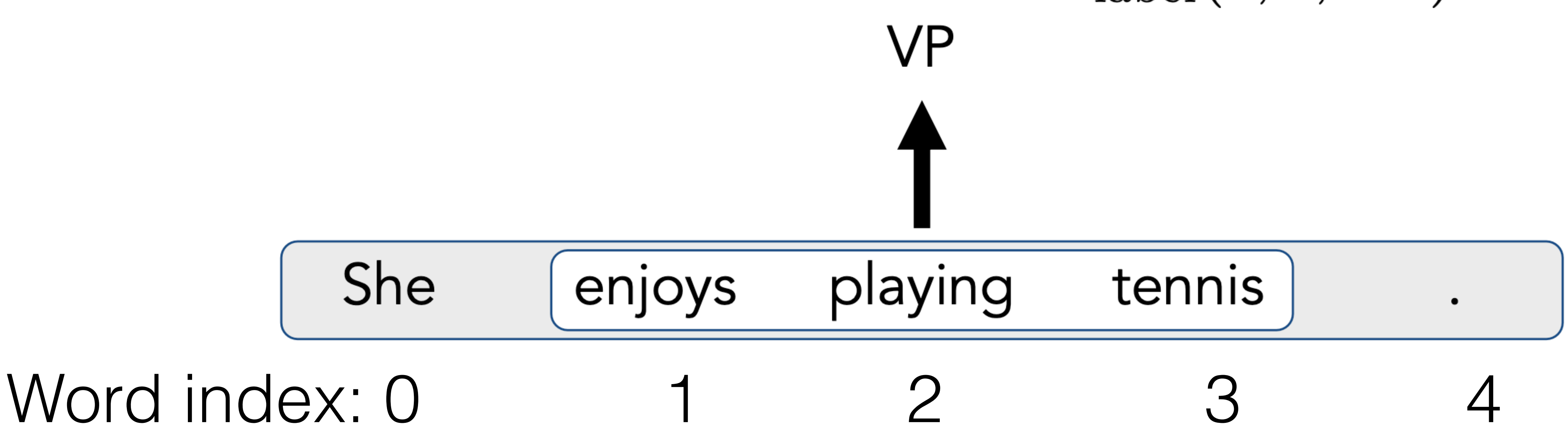
Stern et. al 2016. A Minimal Span-Based Neural Constituency Parser

Span-Based Parsing

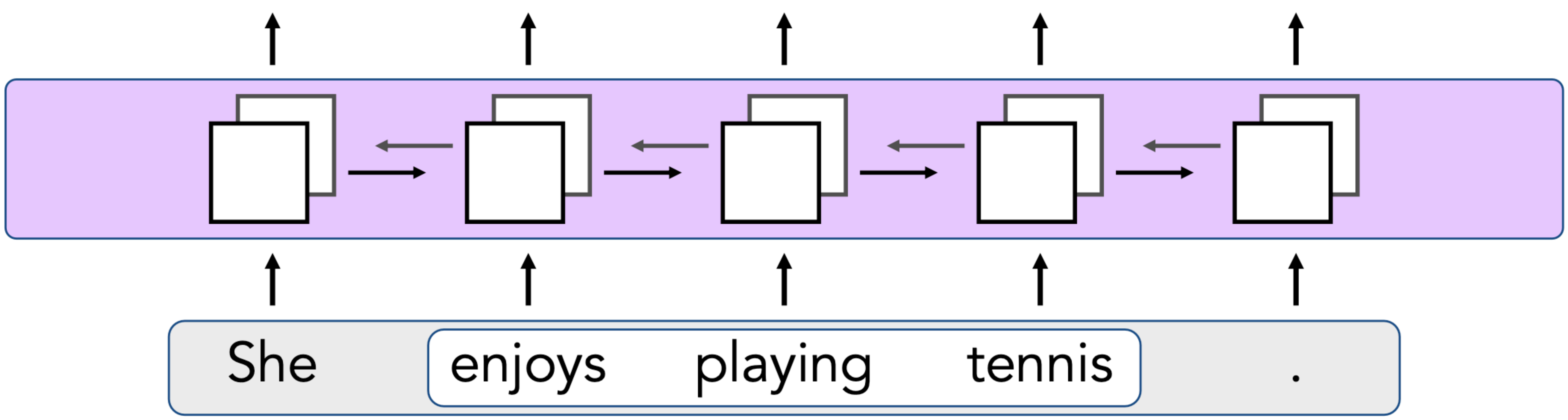
$$s_{\text{label}}(i, j, \ell)$$

Scoring a span from the i-th word to j-th word being the label of ℓ

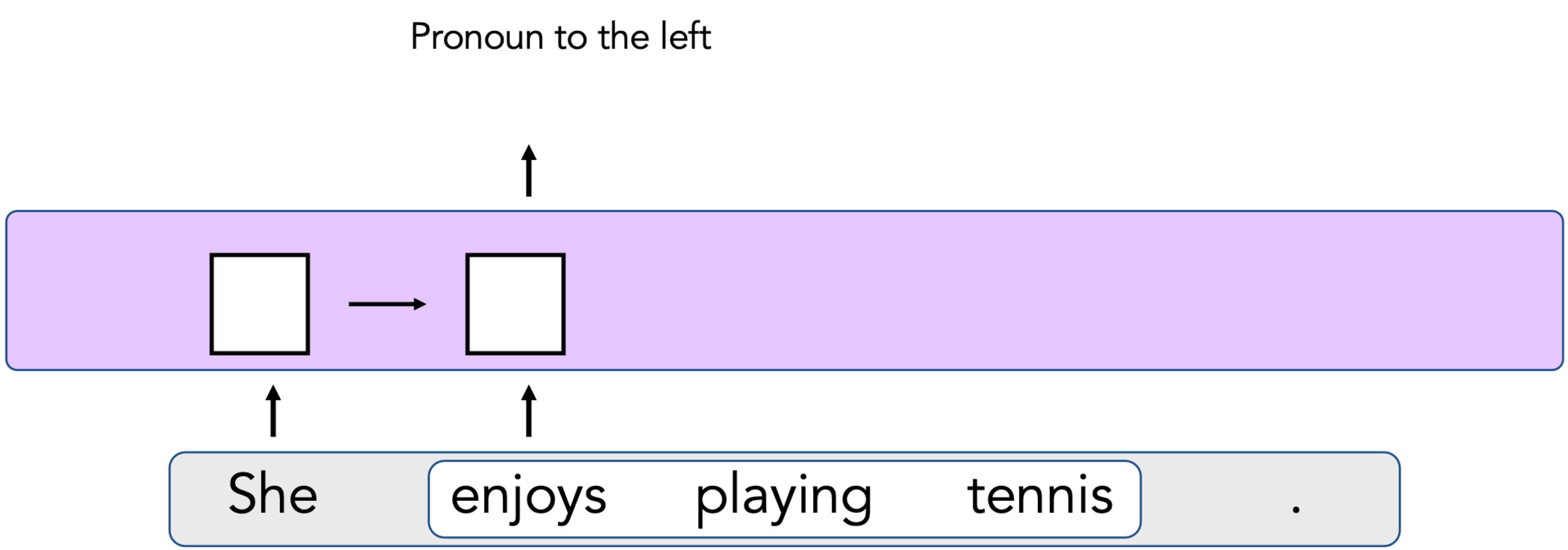
$$s_{\text{label}}(1, 3, \text{VP})$$



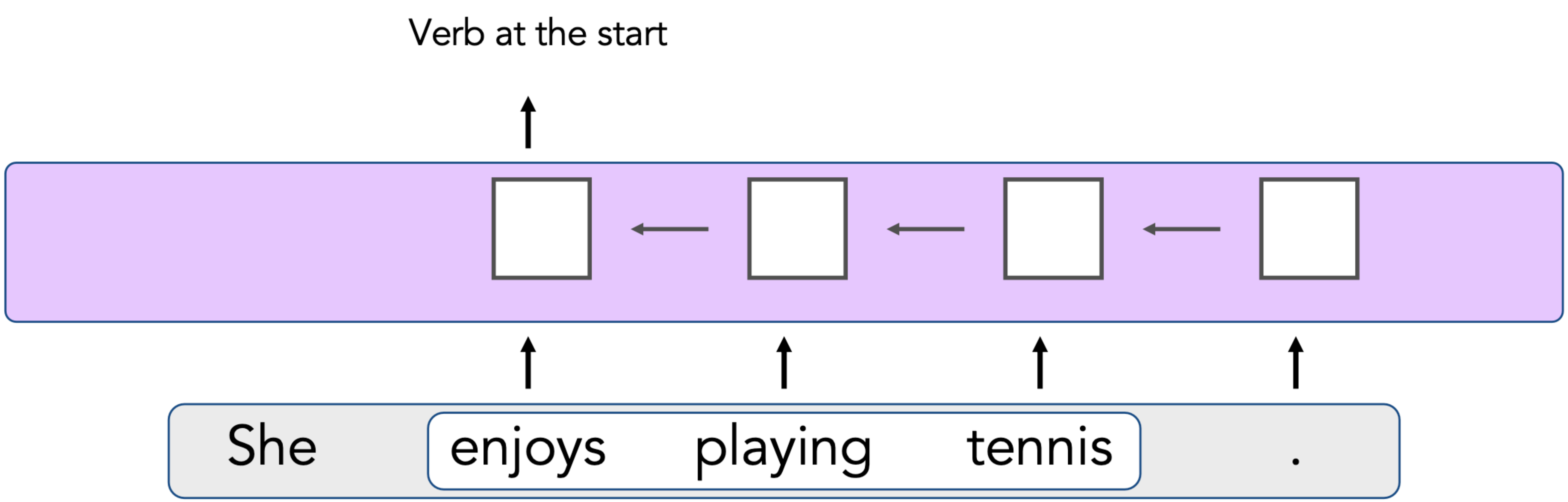
Span-Based Parsing



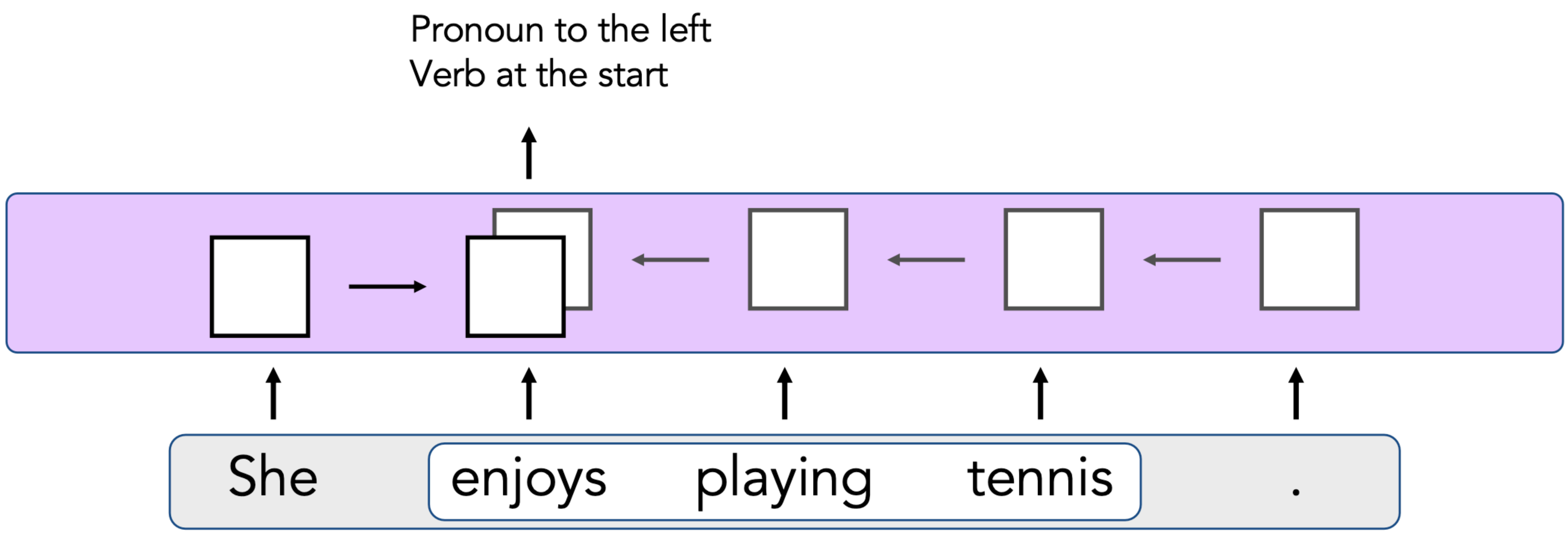
Span-Based Parsing



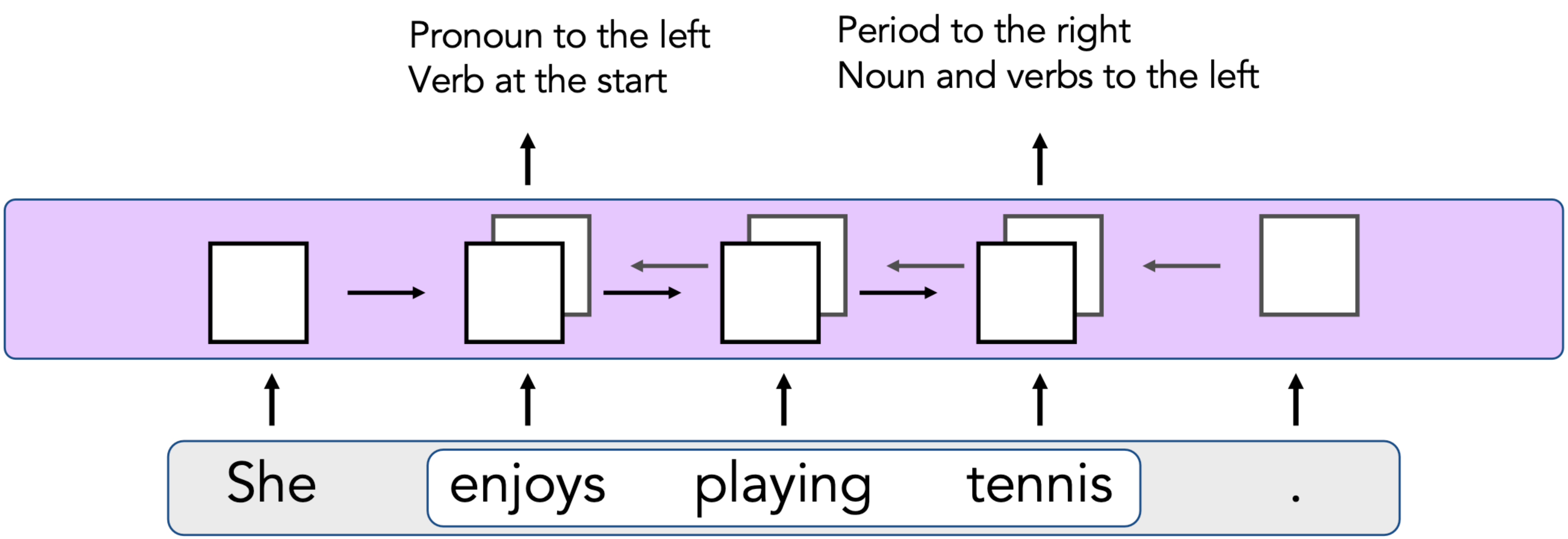
Span-Based Parsing



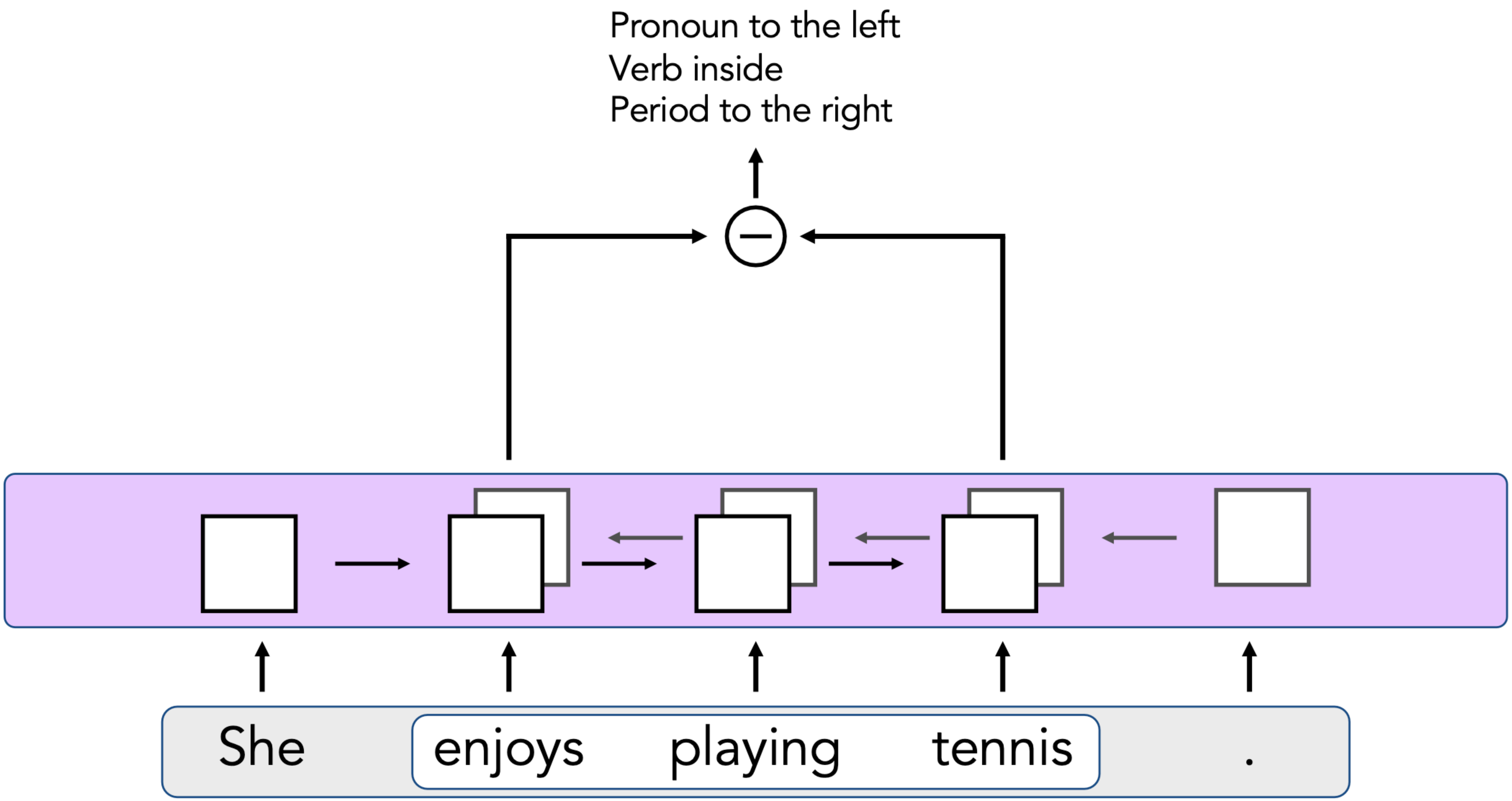
Span-Based Parsing



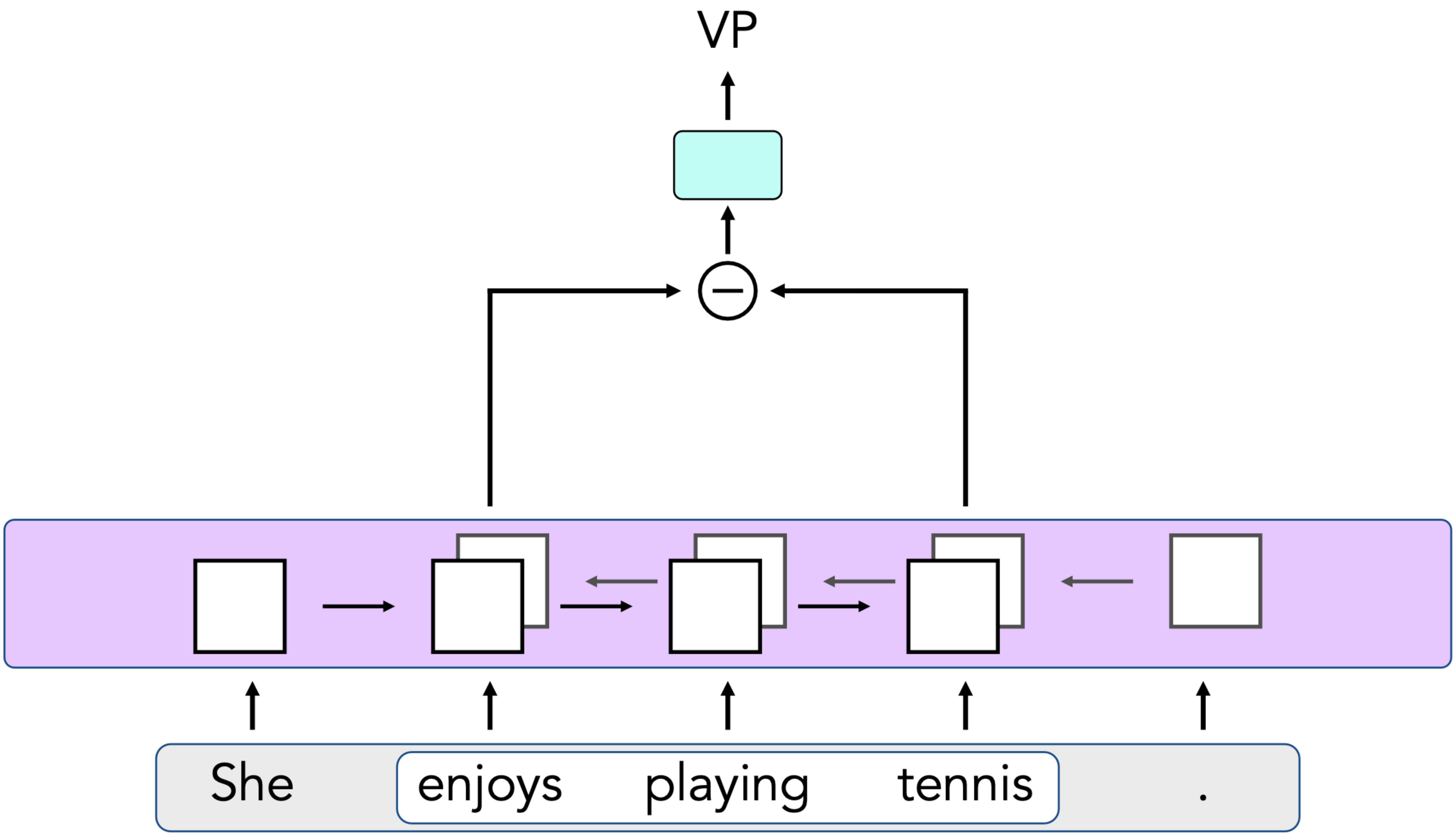
Span-Based Parsing



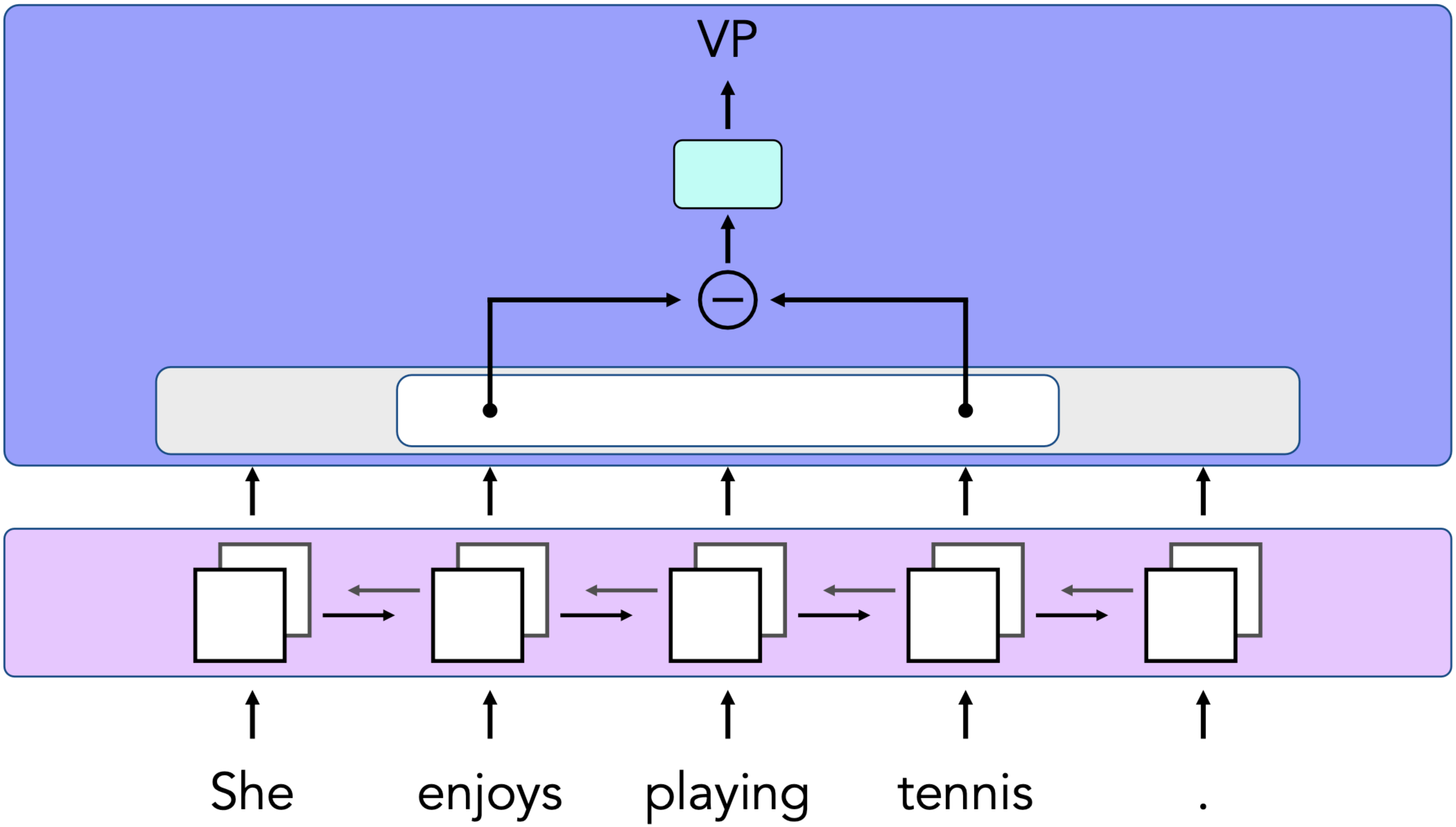
Span-Based Parsing



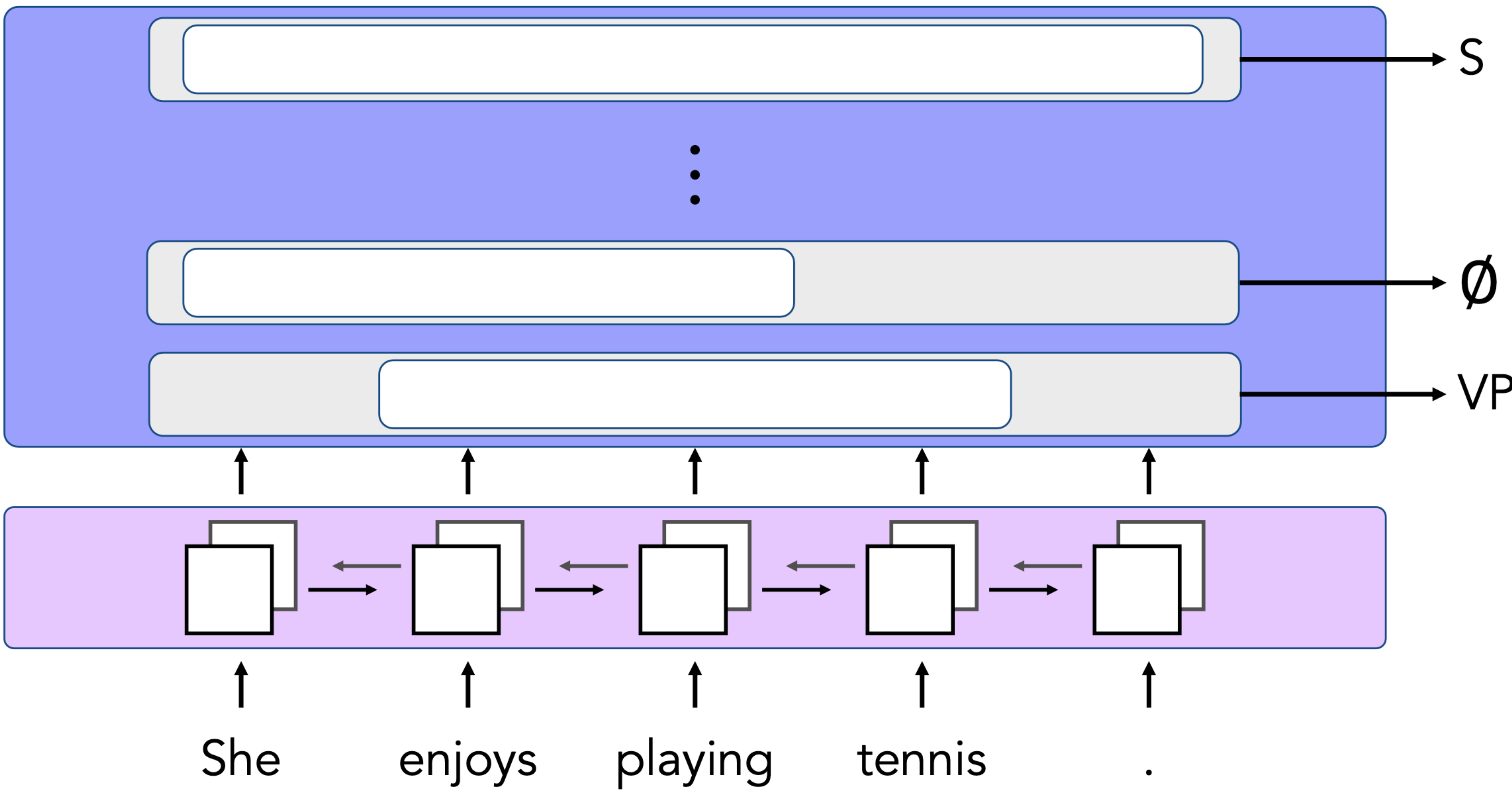
Span-Based Parsing



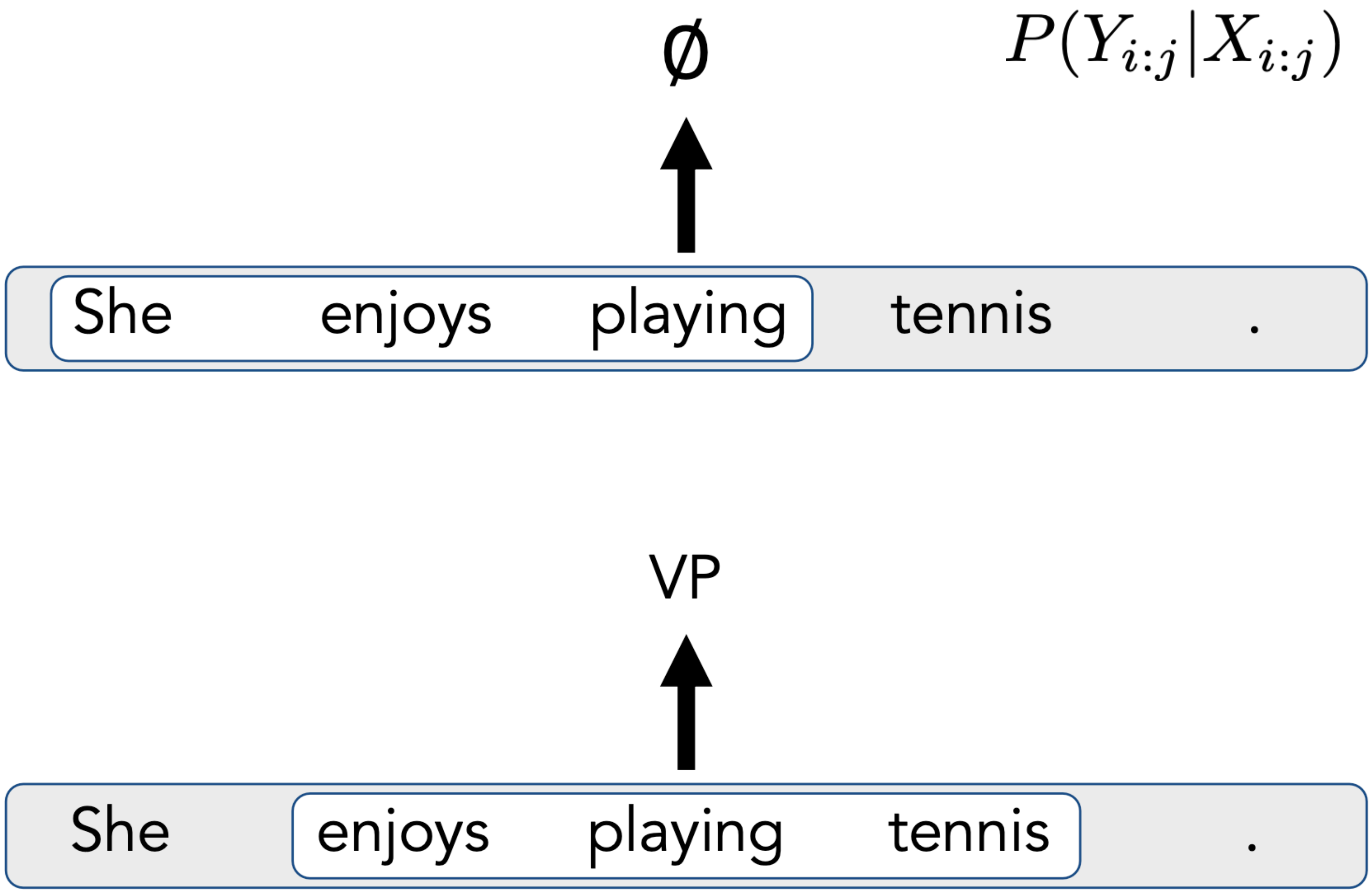
Span-Based Parsing



Span-Based Parsing



Span-Based Parsing



Training: Margin Loss

- Find the best tree using the current model

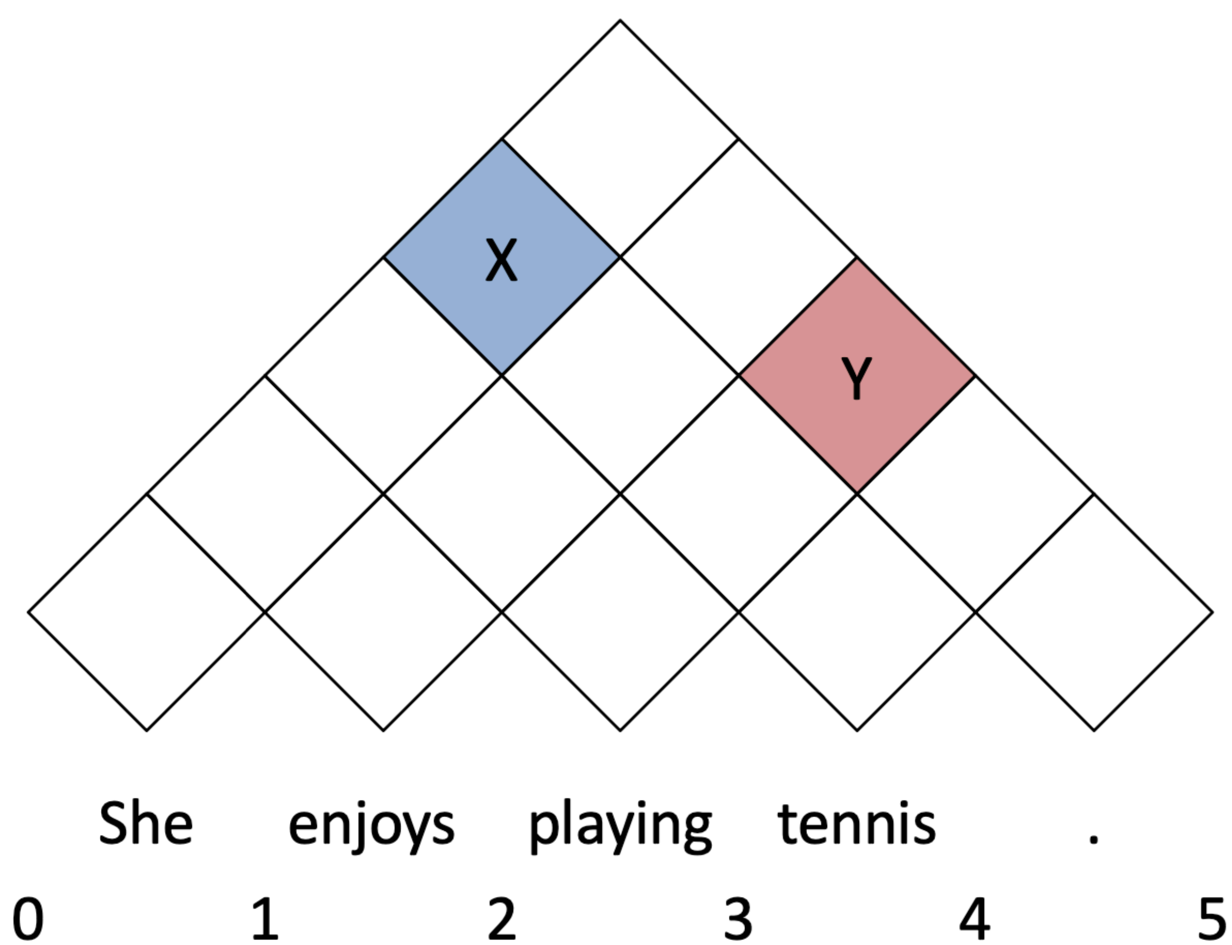
$$\widehat{T} = \underset{T}{\operatorname{argmax}} \left[s_{\text{tree}}(T) \right].$$

- Margin loss:

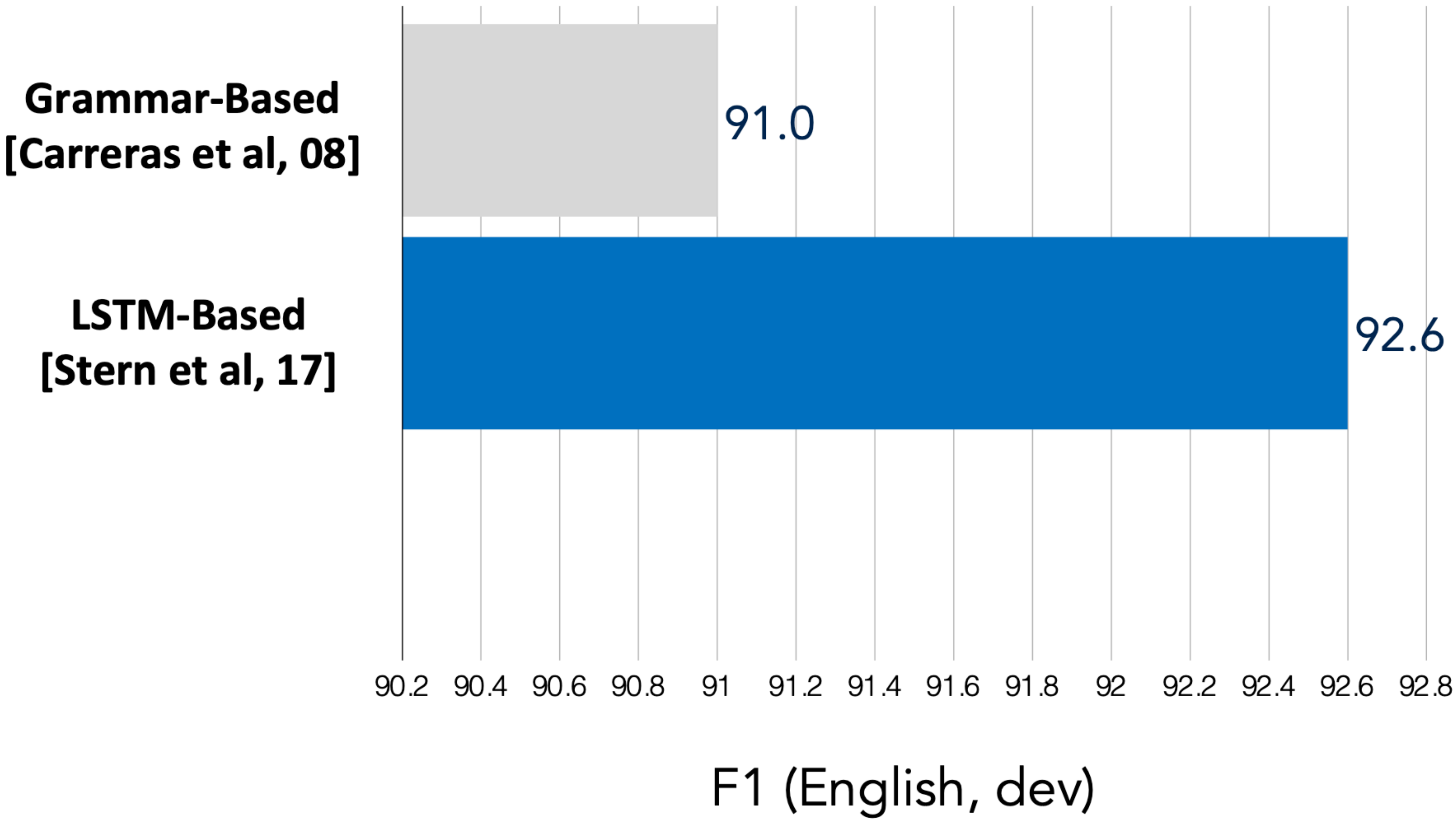
$$\max\left(0,\ 1 - s_{\text{tree}}(T^*) + s_{\text{tree}}(\widehat{T})\right)$$

Decoding: CYK

- Same as counting-based PCFG
- Use the learned scores for possible spans in the following chart

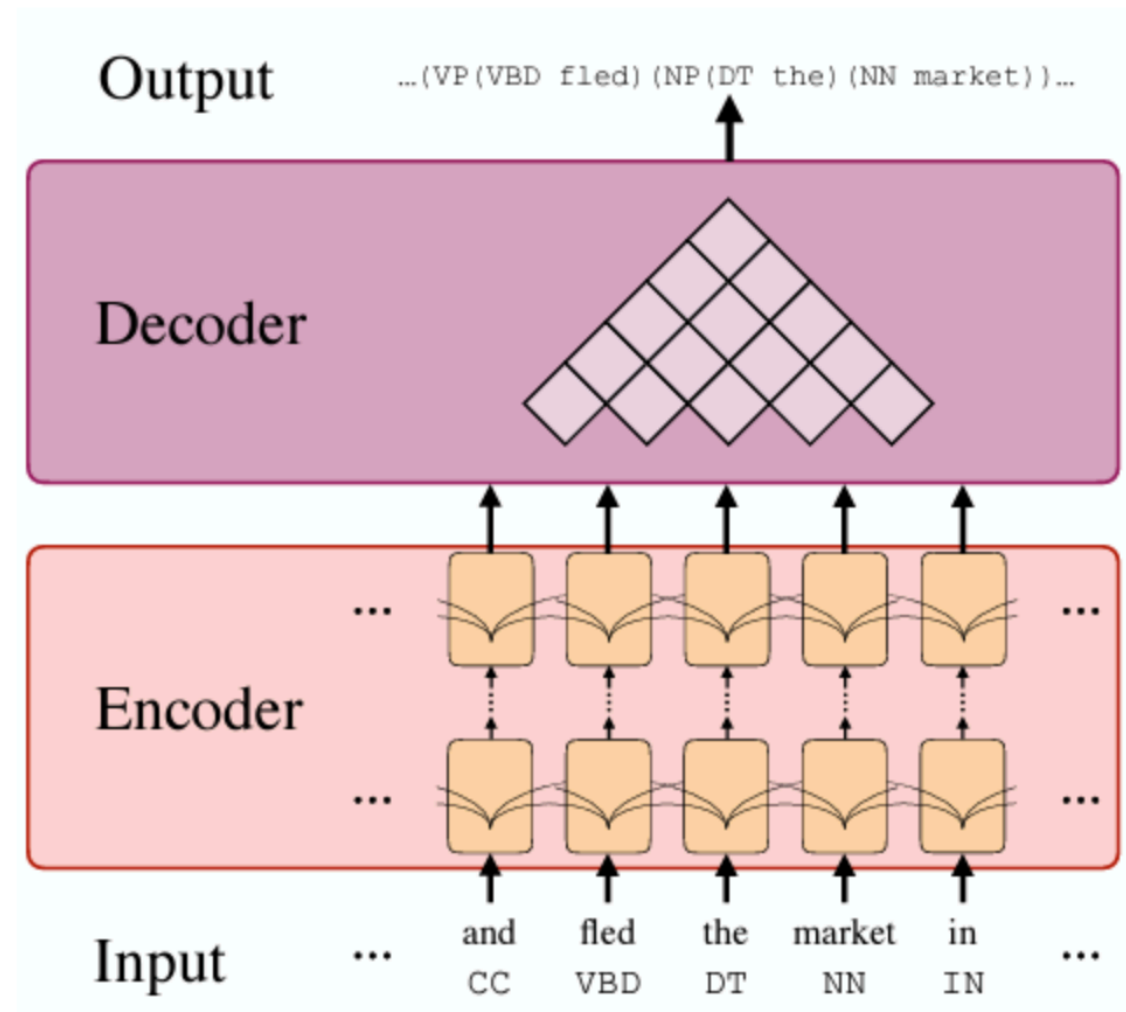


Improves over non-neural methods



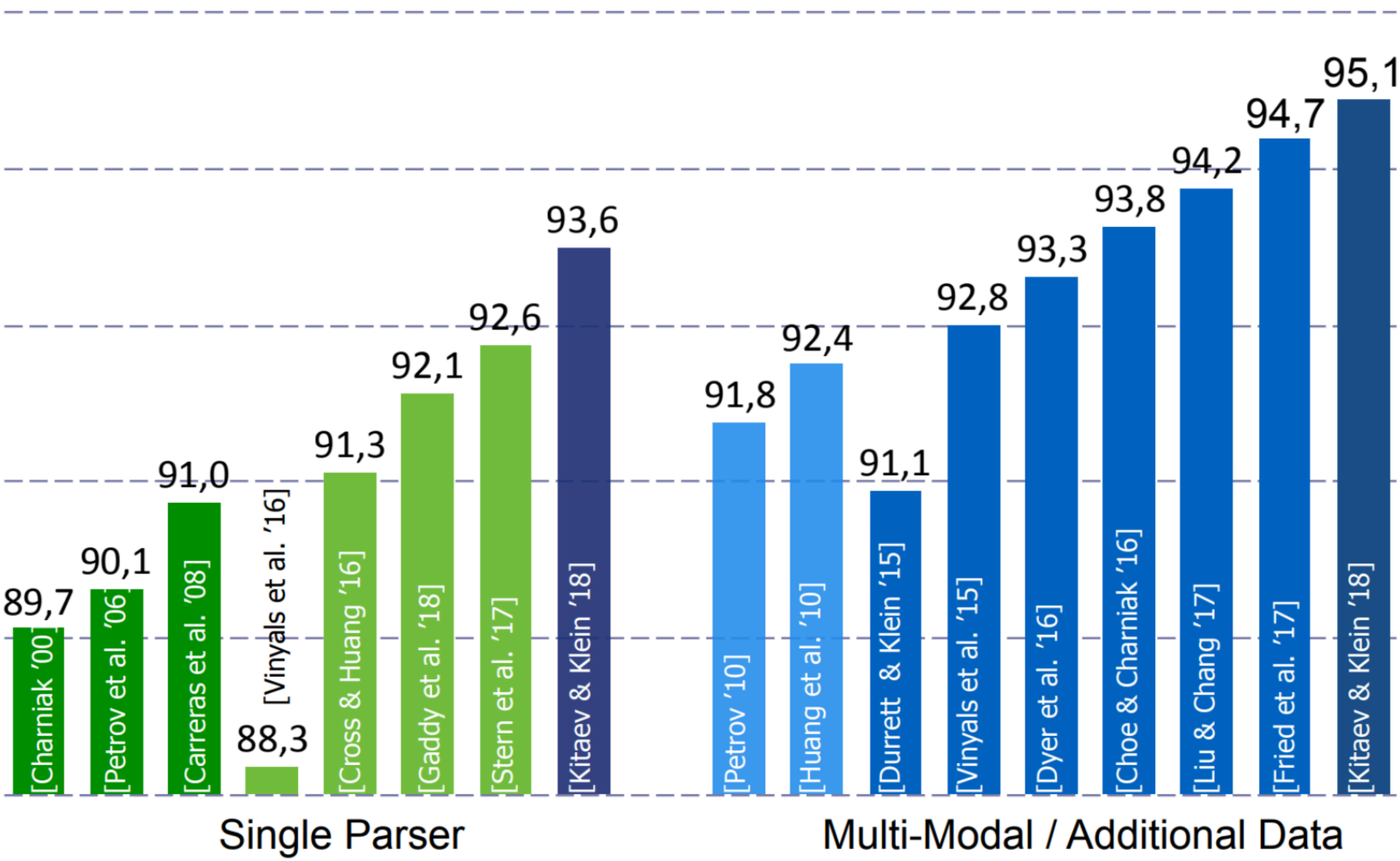
SoTA Self-attention-based Parser

- Use the Transformer encoder instead of Bi-LSTM
  - Split the word hidden vector from Transformer into two half vectors $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$
  - Replace the forward and backward hidden vectors of Bi-LSTM by the new vectors



Kitaev et al. 2018. Constituency Parsing with a Self-Attentive Encoder

Historical Trends on Penn Treebank



Questions?