CS769 Advanced NLP

Syntactic Parsing I: Constituency Grammar
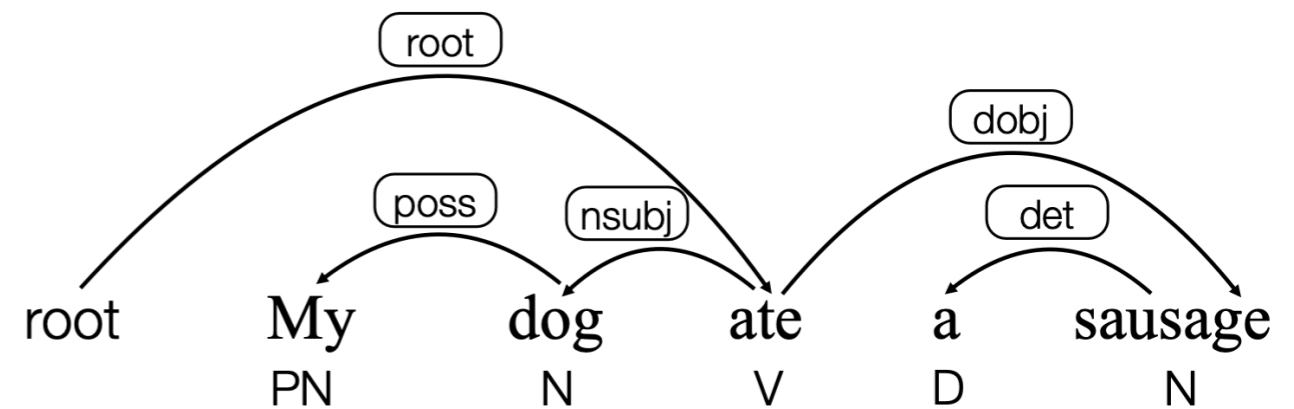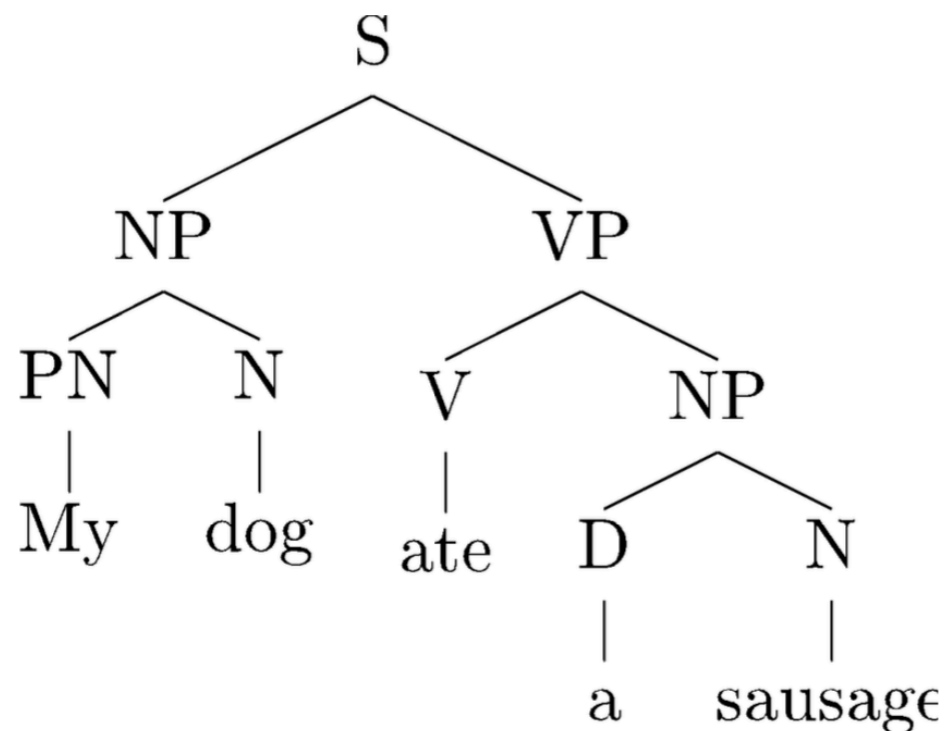
Junjie Hu

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Slides adapted from Bob, Hao, Dan
https://junjiehu.github.io/cs769-spring22/

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

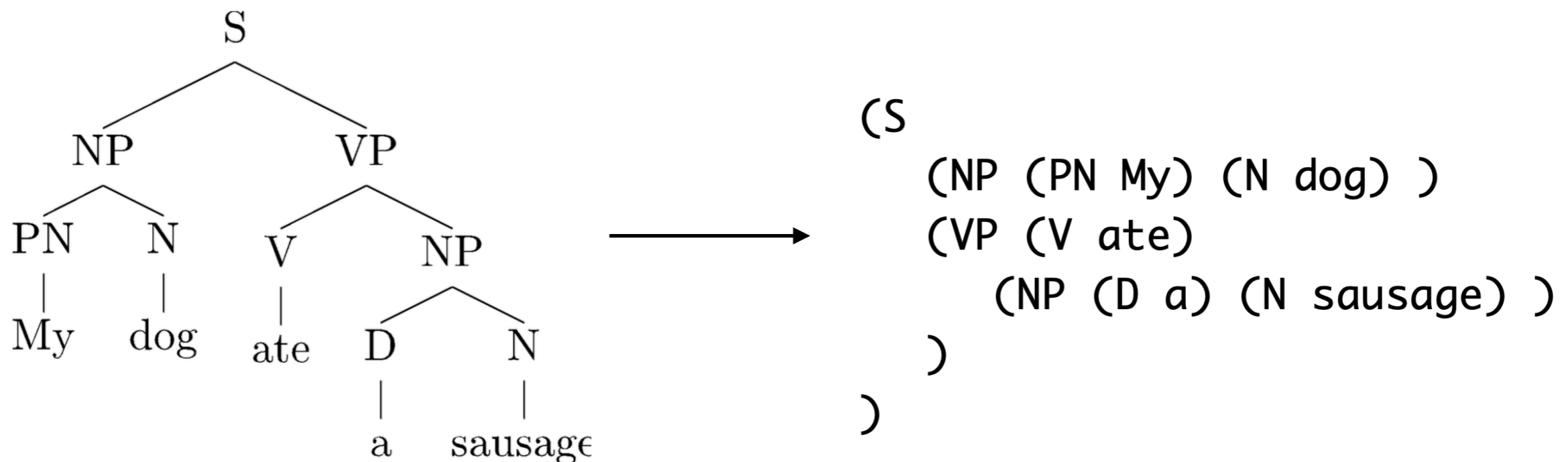**Dependency tree**:
Focus on relations between words

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
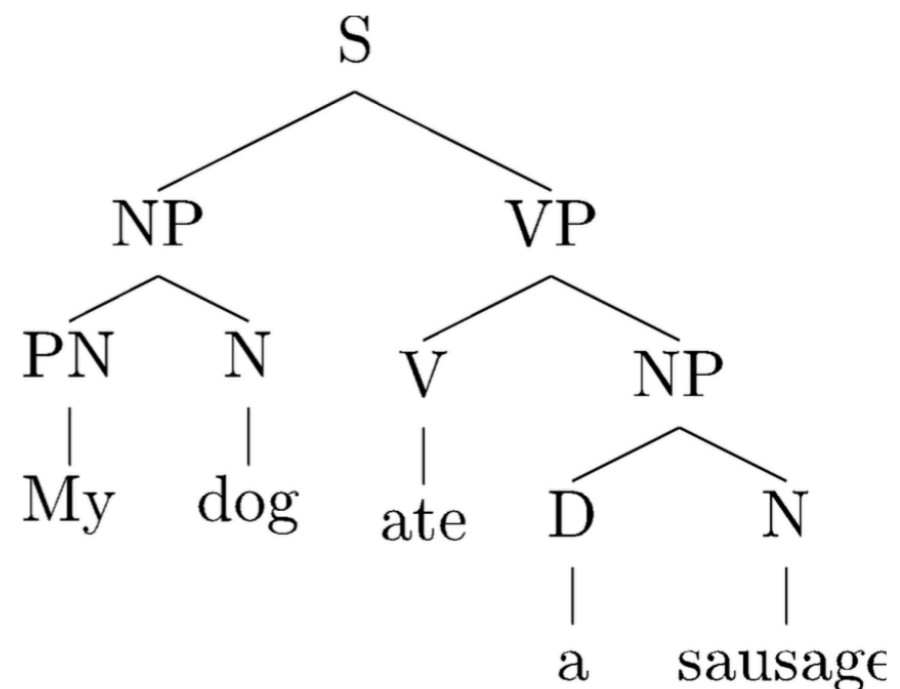
Bracketing notation

- 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
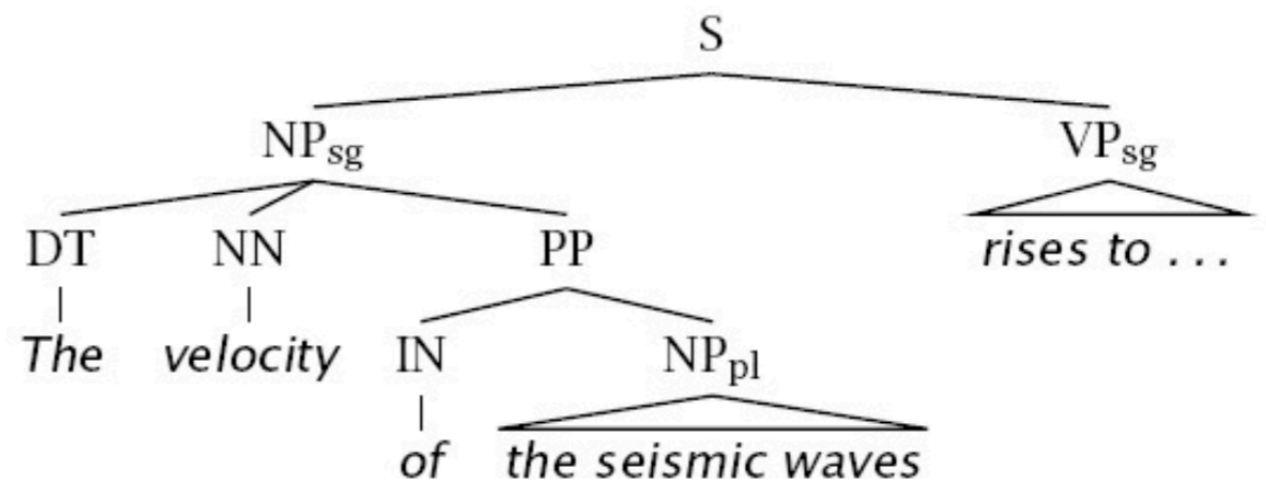
La velocité des ondes sismiques

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

root   My   dog   ate   a   sausage
   PN   N   V   D   N
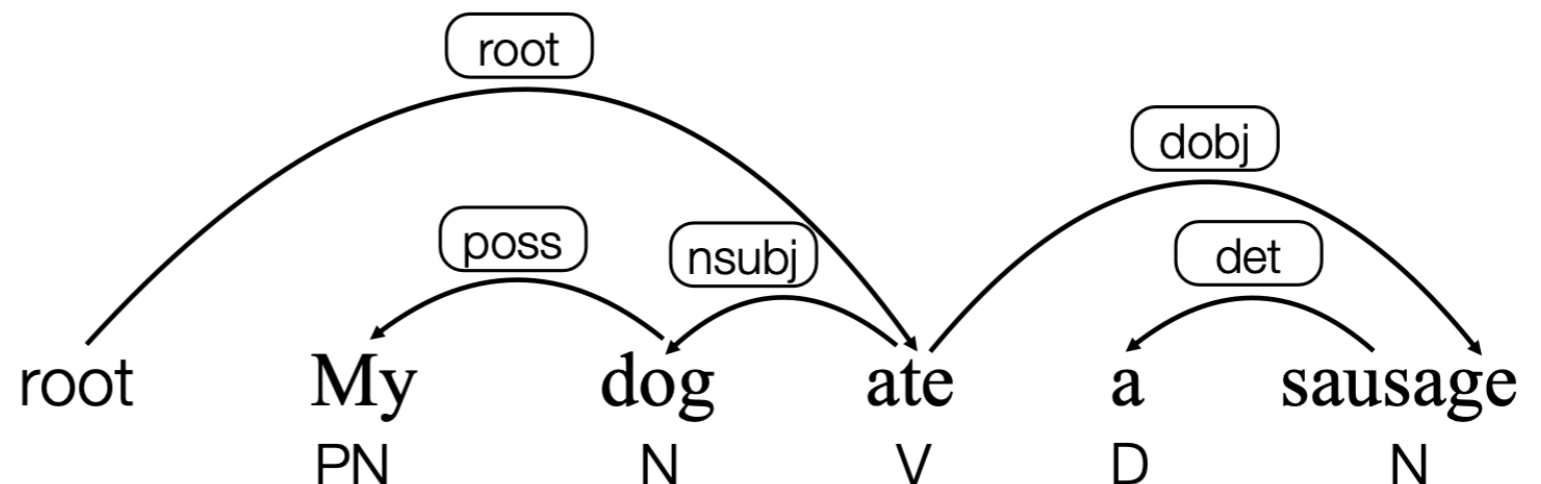
Dependency parsing

- 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

NP: dog   VP: ate

Lexicalized non-terminal
w/ head word
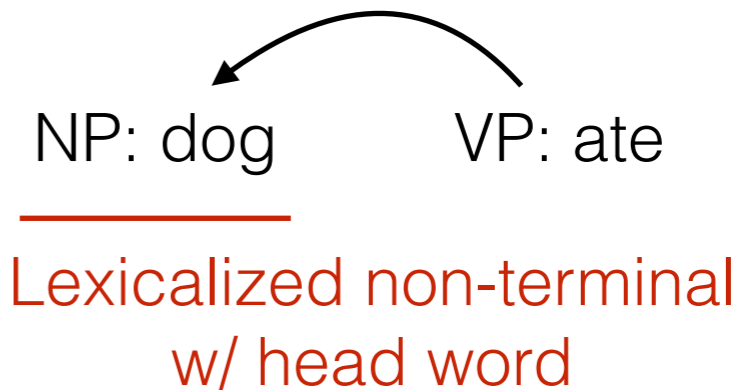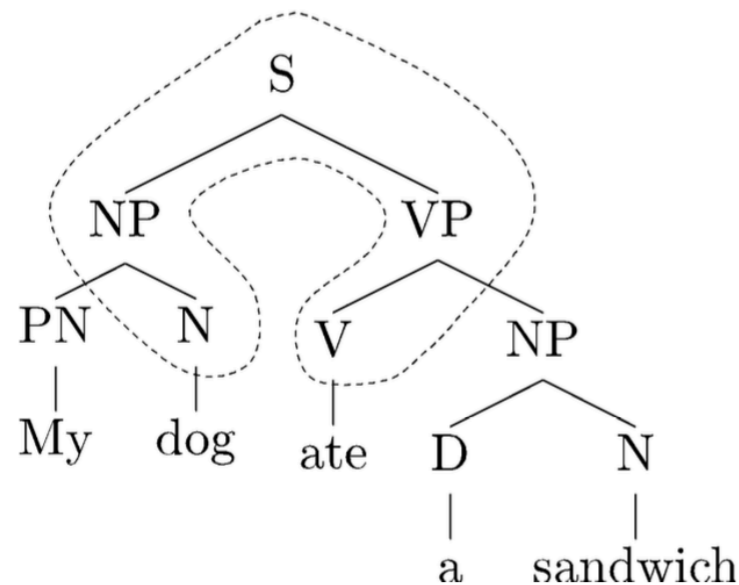
- Dependency trees can (potentially) → constituency trees
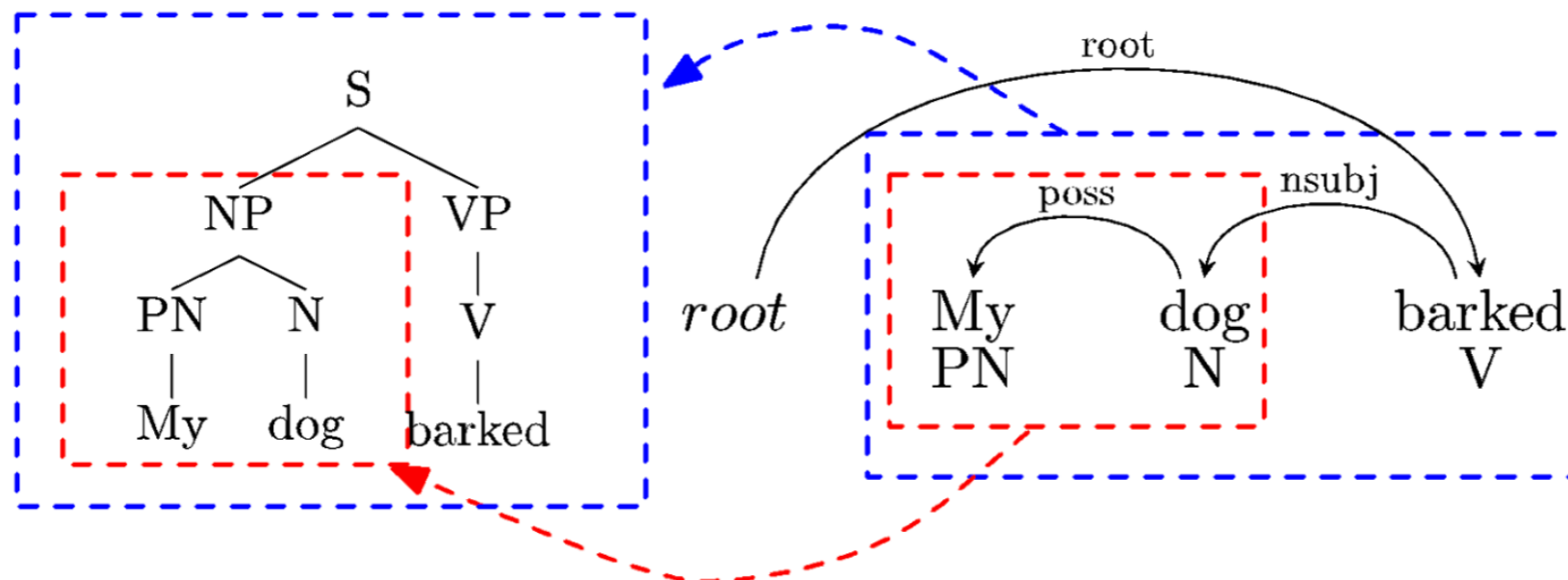
Context Free Grammar (CFG) & Probabilistic CFG

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

S
NP   VP

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

```
        S
       / \
     NP   VP
      |
     PN
```

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

S
├── NP
│   └── PN
│   └── I
└── VP

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

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

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 $A \to s$, where A is a non-terminal, s is a string of symbols, $A \in N, s \in (\Sigma \cup N)*$

$$
\begin{aligned}
S &\to A, & A \in N \\
A &\to BC, & A \in N, B, C \in N \cup \Sigma \\
A &\to \alpha, & \alpha \in \Sigma
\end{aligned}
$$

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: $P(A \to s)$

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

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

PCFG (Example)

S → NP VP   1.0   (NP a girl) (VP ate a sandwich)

VP → V   0.2

VP → V NP   0.4   (V ate) (NP a sandwich)

VP → VP PP   0.4   (VP saw a girl) (PP with a telescope)

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

NP → NP PP   0.3   (NP a girl) (PP with a sandwich)

NP → D N   0.5   (D a) (N sandwich)

NP → PN   0.2

PP → P NP   1.0   (P with) (NP a sandwich)

N → *girl*   0.2

N → *telescope*   0.7

N → *sandwich*   0.1

PN → *I*   1.0

V → *saw*   0.5

V → *ate*   0.5

P → *with*   0.6

P → *in*   0.4

D → *a*   0.3

D → *the*   0.7

PCFG (Example)

S
|
/ \
NP   VP
        1.0

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
		V → *ate*	0.5
NP → NP PP	0.3	P → *with*	0.6
NP → D N	0.5	P → *in*	0.4
NP → PN	0.2	D → *a*	0.3
PP → P NP	1.0	D → *the*	0.7

$P(T) = 1.0 *$

PCFG (Example)

```
        S
       / \  1.0
     NP   VP
      |  0.2
     PN
```

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
		V → *ate*	0.5
NP → NP PP	0.3		
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
        / \    1.0
      NP   VP
      |  0.2
      PN
      |  1.0
      I
```

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
		V → *ate*	0.5
NP → NP PP	0.3		
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
       / \ 1.0
     NP    VP
     | 0.2   \  0.4
     PN    / \
     | 1.0 V   NP
     I
```

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
		V → *ate*		0.5
NP → NP PP	0.3			
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	N → *sandwich*		0.1
VP → V NP	0.4			
VP → VP PP	0.4	PN → *I*		1.0
		V → *saw*		0.5
		V → *ate*		0.5
NP → NP PP	0.3	P → *with*		0.6
NP → D N	0.5	P → *in*		0.4
NP → PN	0.2	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 *
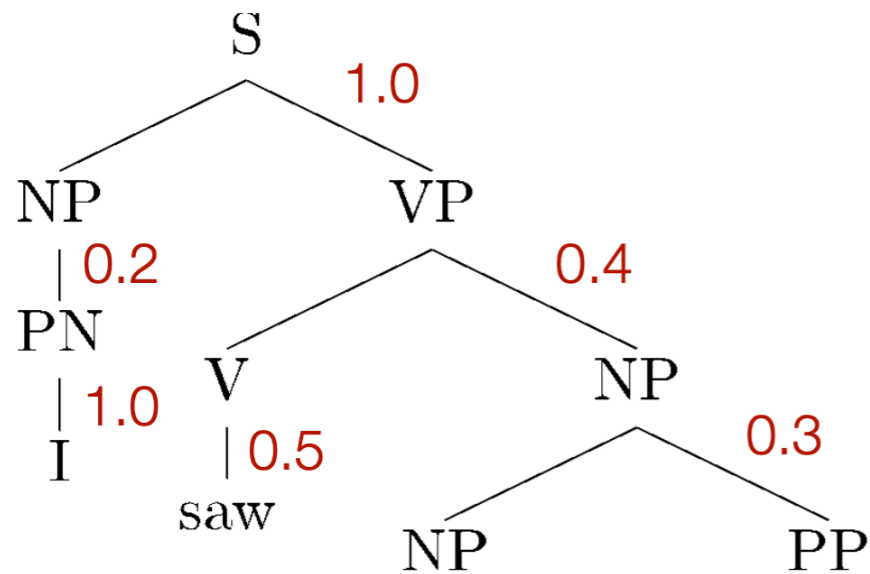
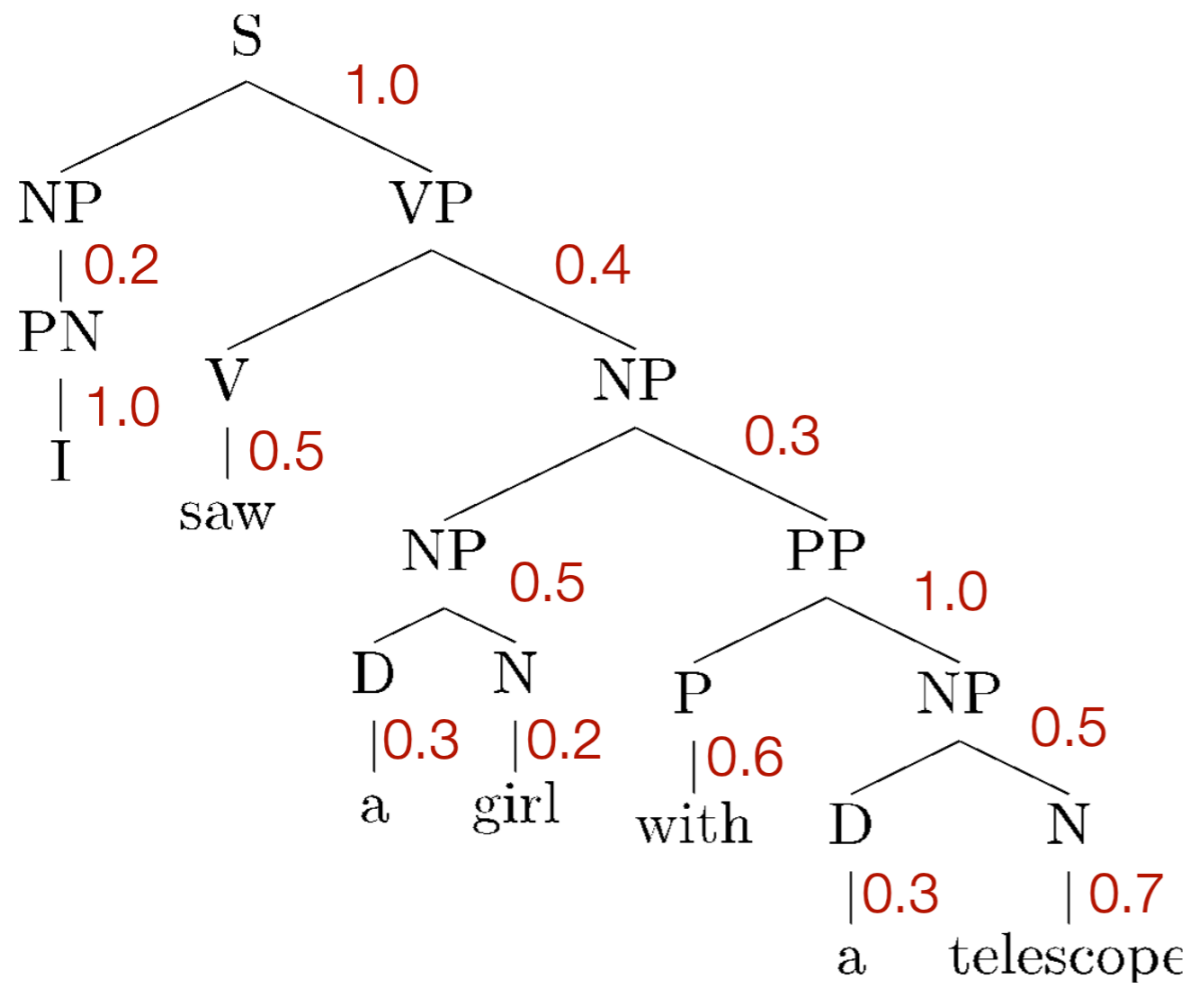# PCFG (Example)

S → NP VP   1.0   N → *girl*   0.2

N → *telescope*   0.7

VP → V   0.2

N → *sandwich*   0.1

VP → V NP   0.4

VP → VP PP   0.4   PN → *I*   1.0

V → *saw*   0.5

V → *ate*   0.5

NP → NP PP   0.3

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
VP → V	0.2
VP → V NP	0.4
VP → VP PP	0.4
NP → NP PP	0.3
NP → D N	0.5
NP → PN	0.2
PP → P NP	1.0

N → *girl*	0.2
N → *telescope*	0.7
N → *sandwich*	0.1
PN → *I*	1.0
V → *saw*	0.5
V → *ate*	0.5
P → *with*	0.6
P → *in*	0.4
D → *a*	0.3
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
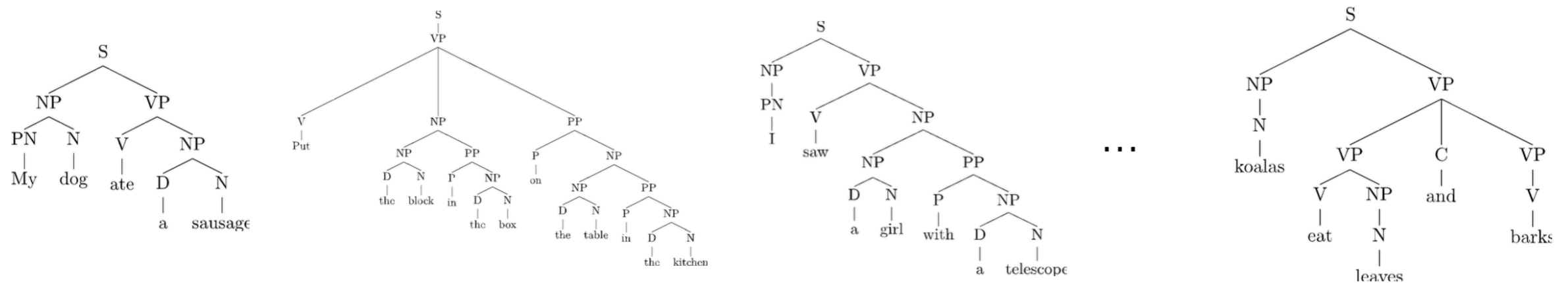
- PCFG: a generative model, maximizing the joint probability of a sentence given a tree.

  - If we constraint the search space to be all valid trees that can generate the sentence, this becomes:

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

PCFG Supervised Learning

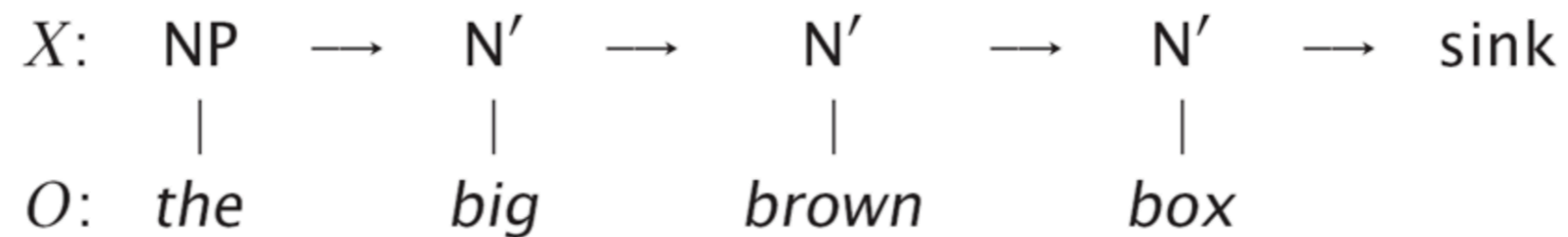- Estimate probability of each rule by maximum likelihood estimation:

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

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

\# times the rule was used in the data

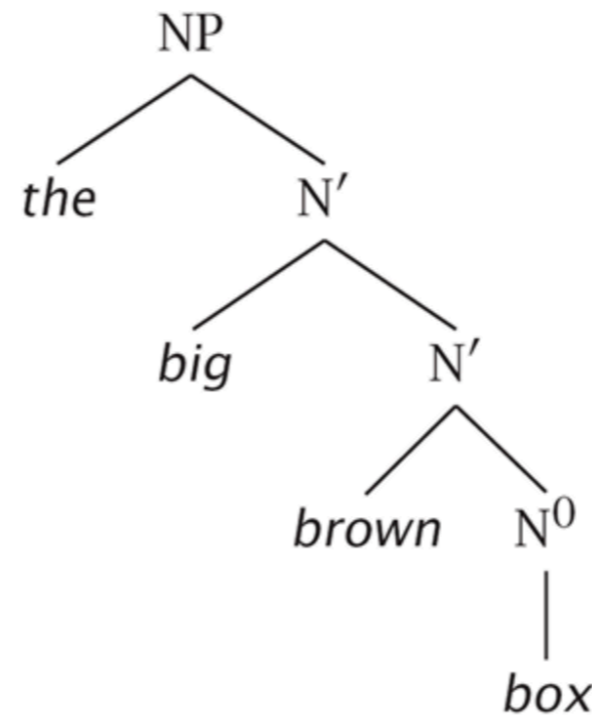\# times the nonterminal was used in the data

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

- If we don't have training data, use EM algorithm to estimate the probability
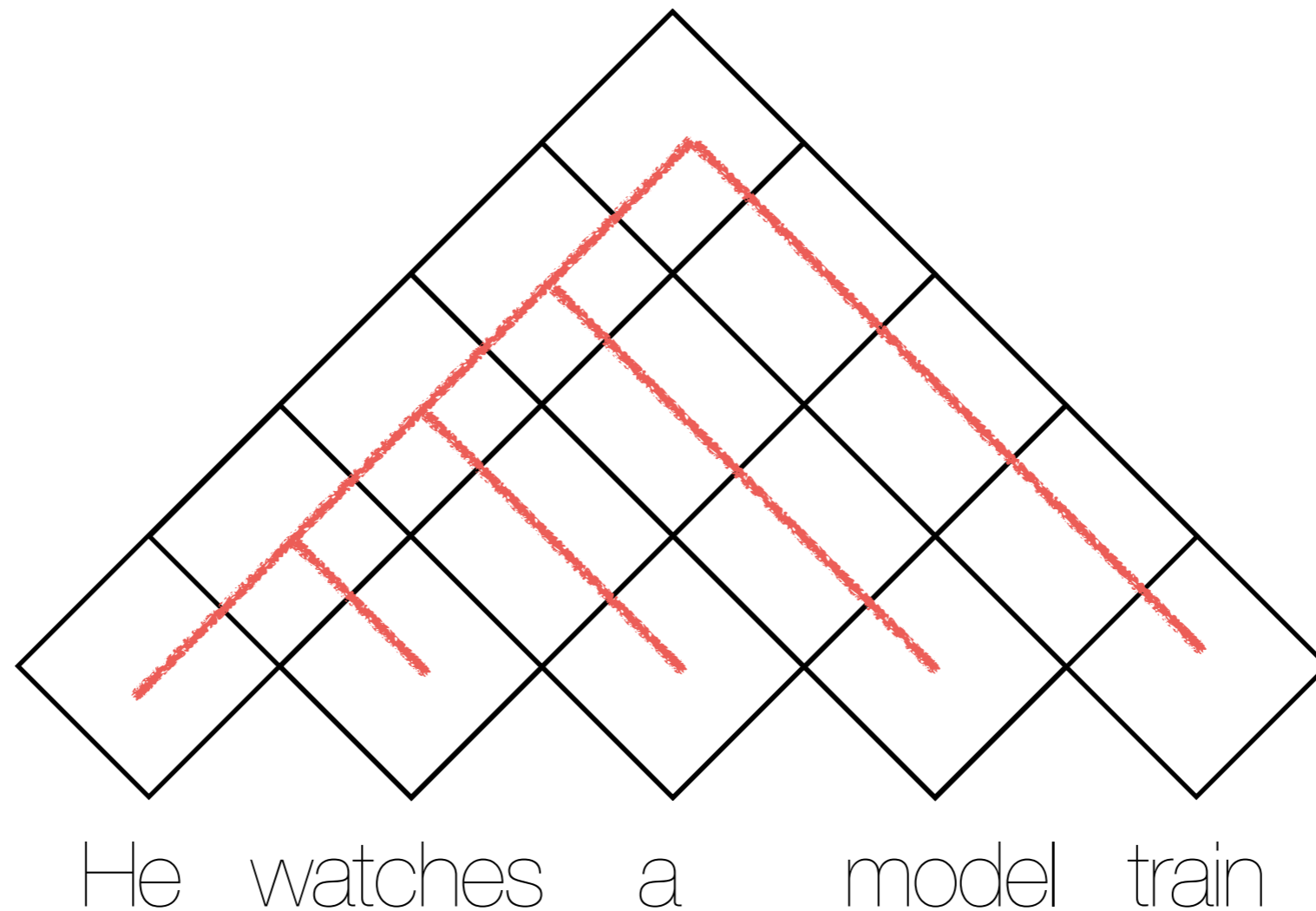
HMM vs PCFG

HMM: Linear Markov Chain

X: NP $\longrightarrow$ N′ $\longrightarrow$ N′ $\longrightarrow$ N′ $\longrightarrow$ sink

O: the   big   brown   box

PCFG: tree

NP
- the
- N′
  - big
  - N′
    - brown
    - N⁰
      - box

PCFG Decoding

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

He   watches   a   model   train

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?
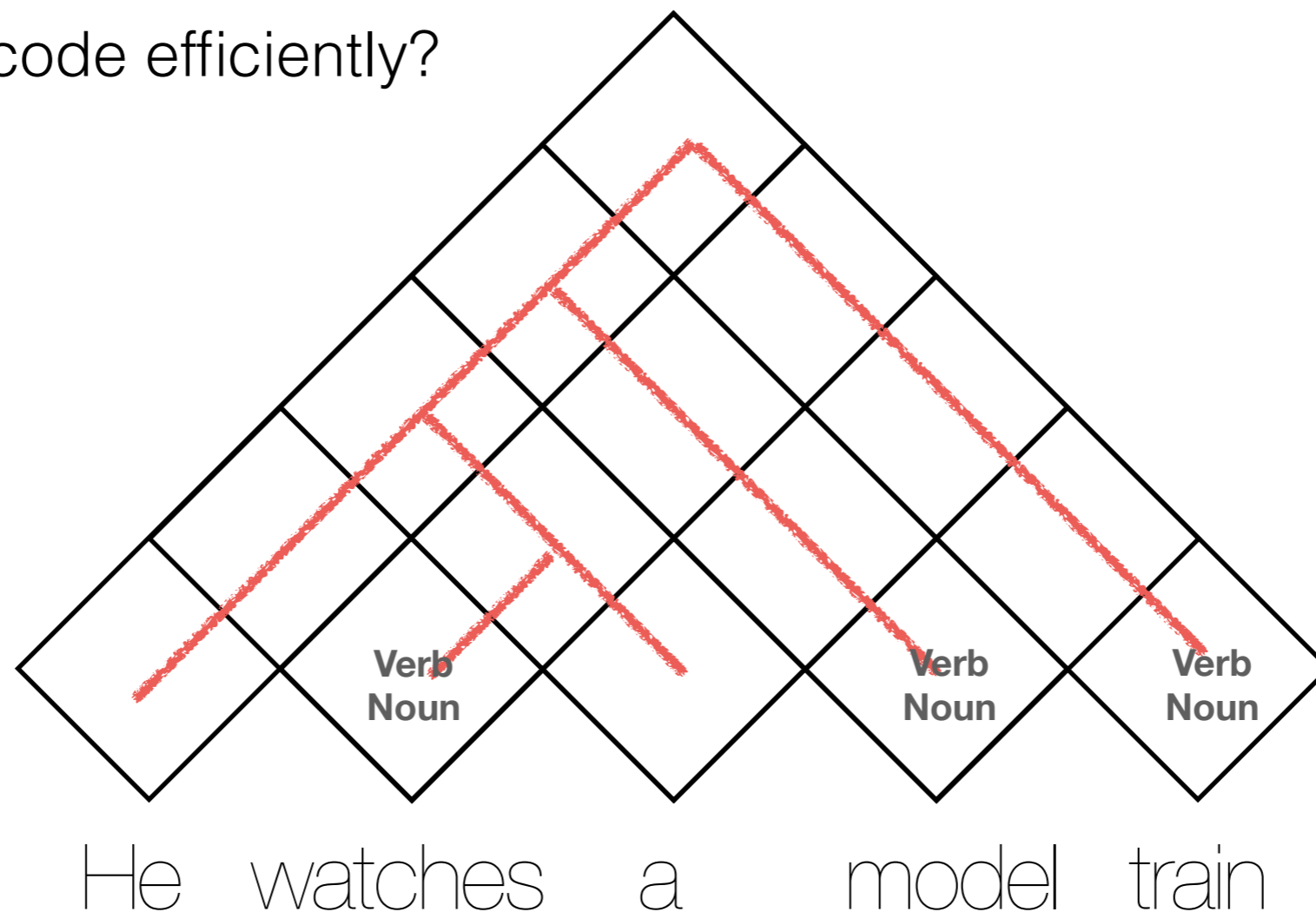
PCFG Decoding: CYK Algorithm
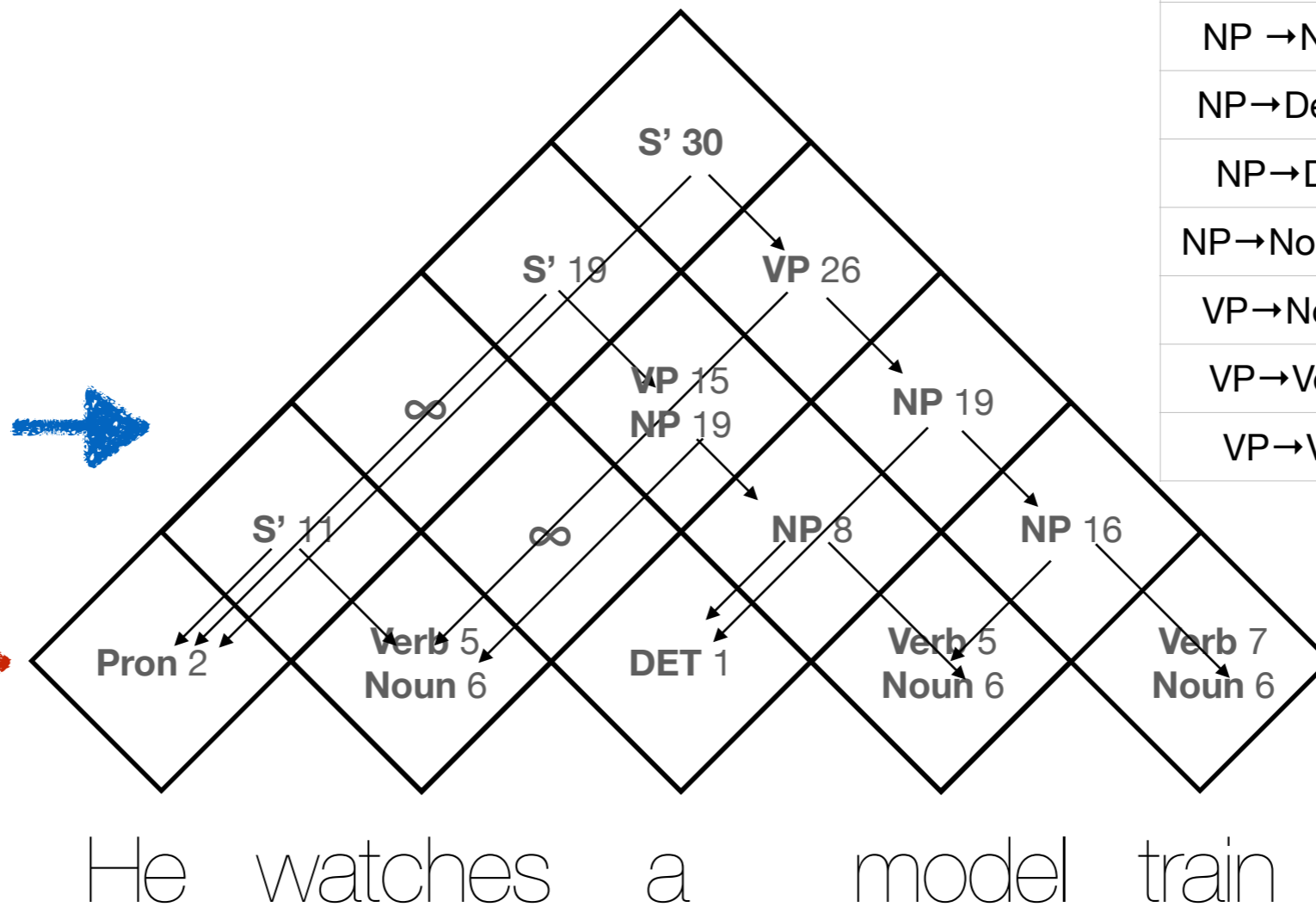
Bottom-up Dynamic Programming

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
$A \rightarrow BC$

Lexical rule
(Unary rule)

$A \rightarrow \alpha, \alpha \in \Sigma$

S' 30

S' 19   **VP 26**

∞   **VP 15**   **NP 19**
        NP 19

S' 11   ∞   **NP 8**   **NP 16**

Pron 2   **Verb 5**   **DET 1**   **Verb 5**   **Verb 7**
               Noun 6   **Noun 6**   **Noun 6**

He   watches   a   model   train

PCFG Decoding: CYK Algorithm

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

S' 30

S' 19   VP 26

∞   VP 15   NP 19
     NP 19

S' 11   ∞   NP 8   NP 16

Pron 2   Verb 5   DET 1   Verb 5   Verb 7
          Noun 6   Noun 6   Noun 6

He   watches   a   model   train
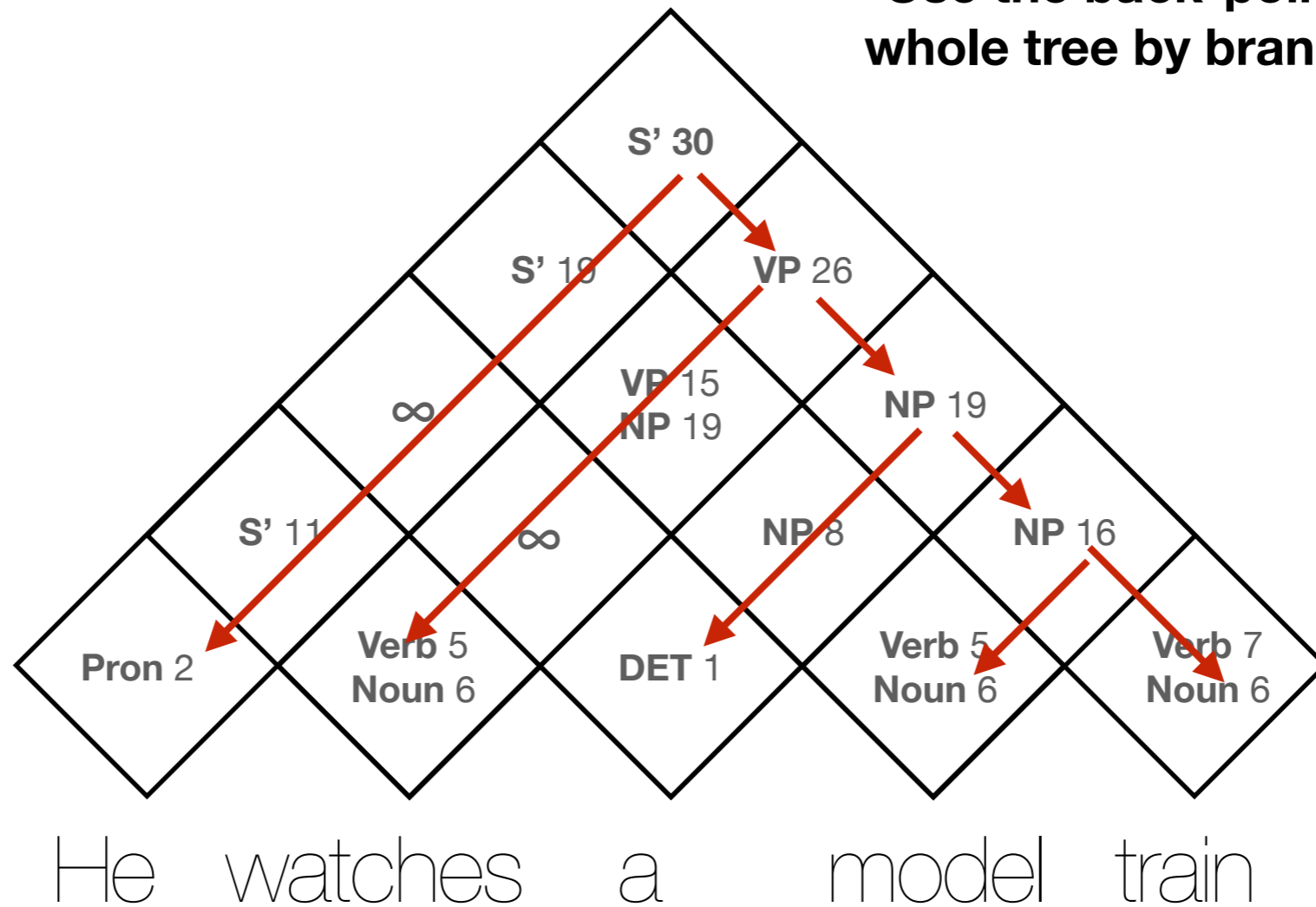
PCFG CYK Decoding

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

  - Penn Treebank
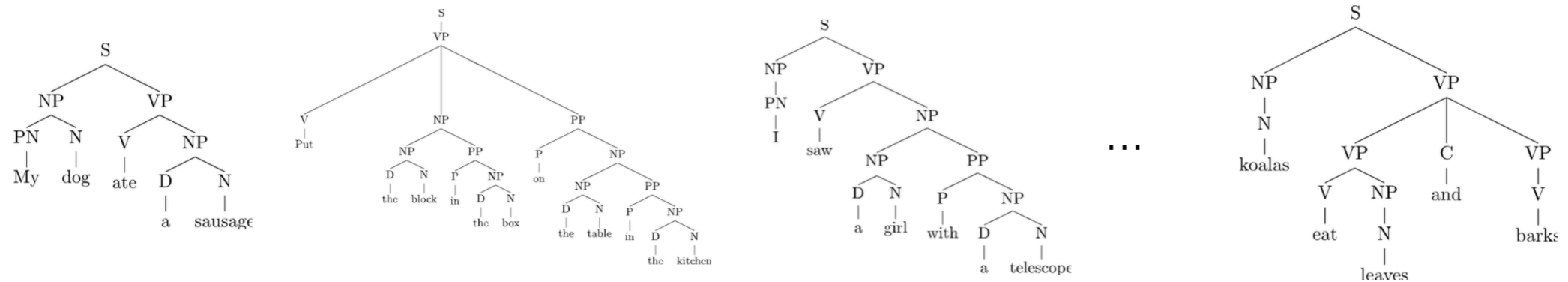
- Estimate probability of each rule by maximum likelihood estimation:

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

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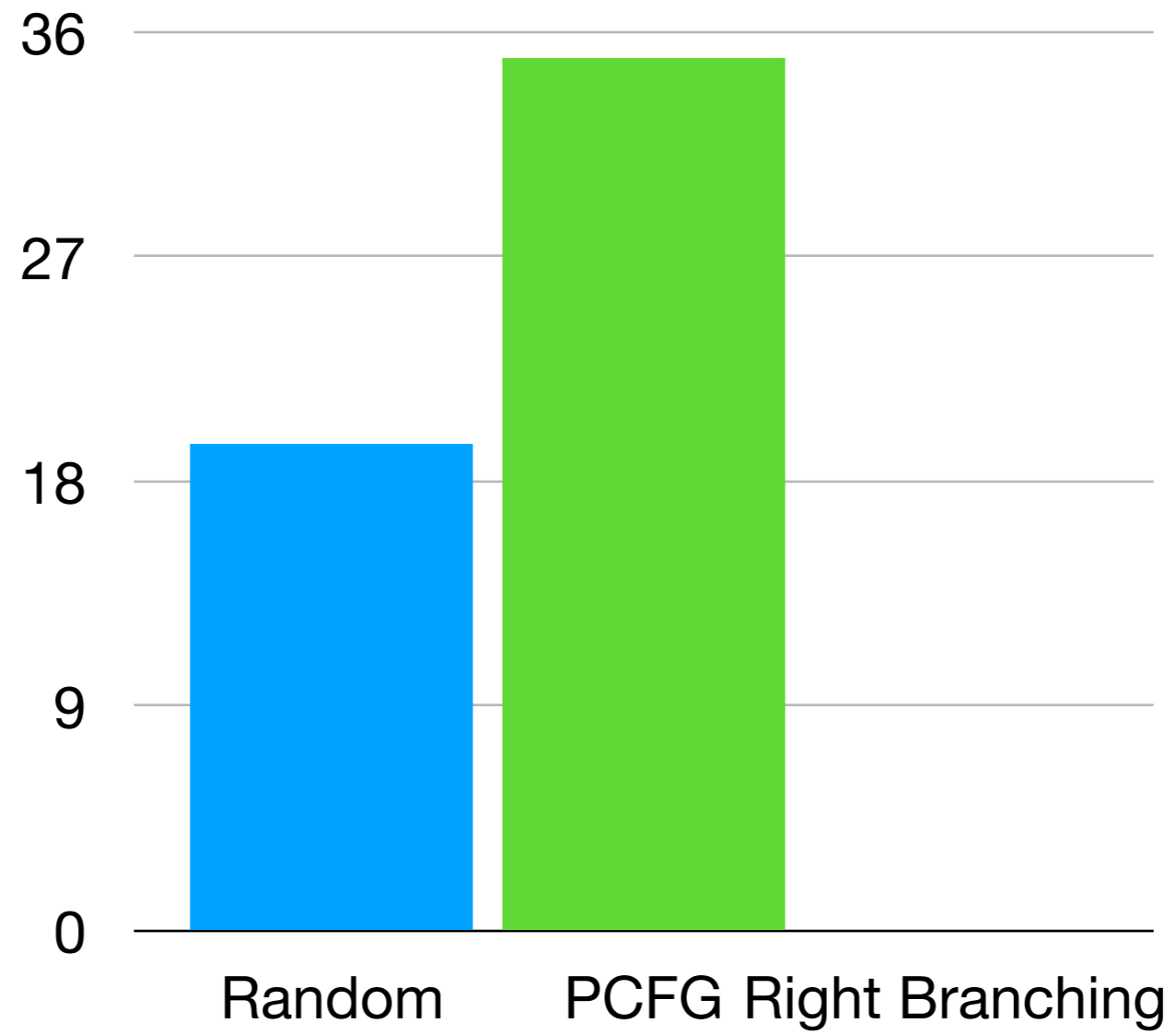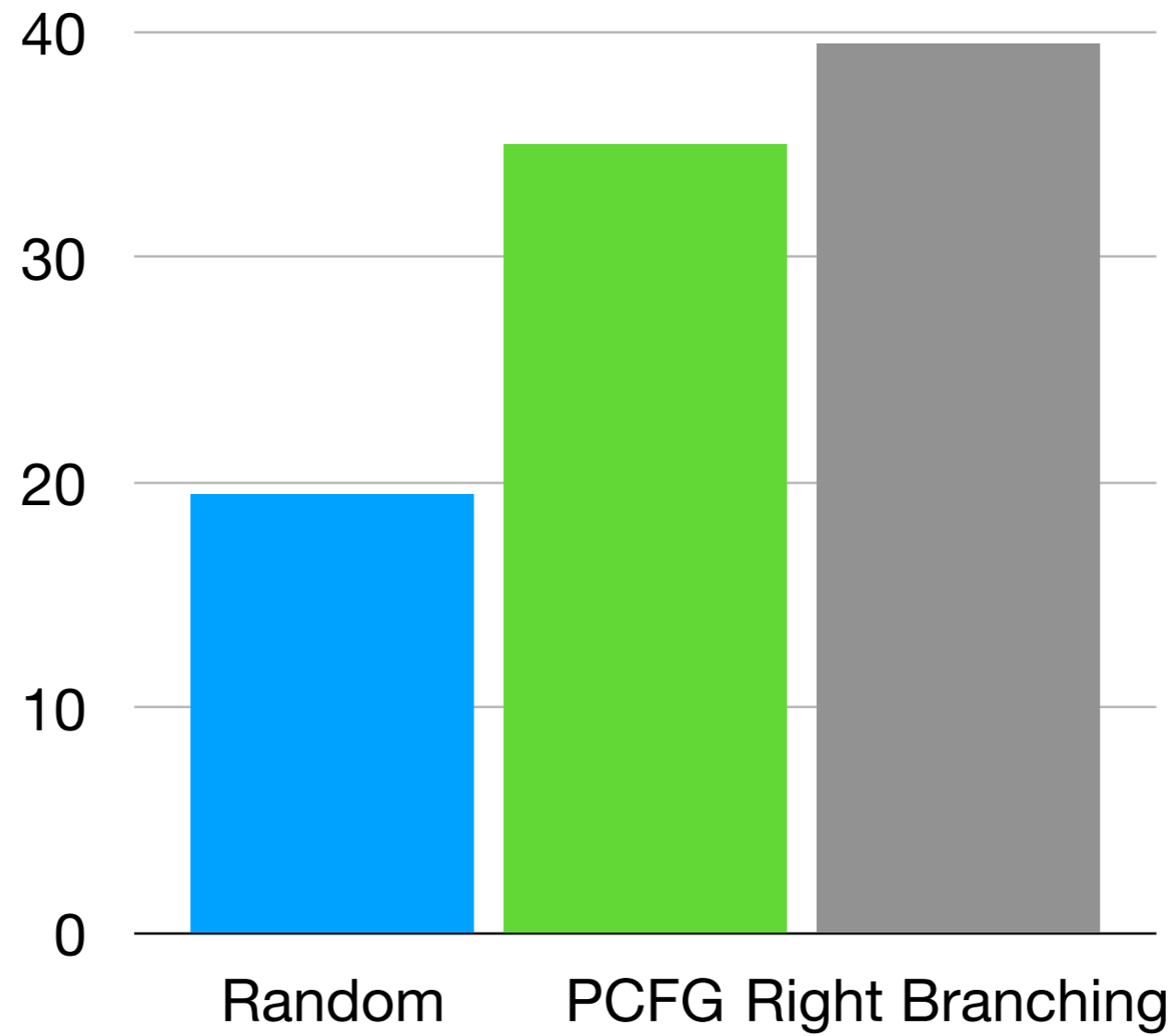
$$s_{\text{label}}(i, j, A) = \max_{k, B, C} P(A \to BC) \times s_{\text{label}}(i, k, B) \times s_{\text{label}}(k + 1, j, C)$$

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

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

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

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	$\oplus$	$\otimes$	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

Great News: It works (better than random)

Bad News: It is even worse than right branching. Why?

Supervised Parsing: Span-based Neural Models

Span-Based Parsing

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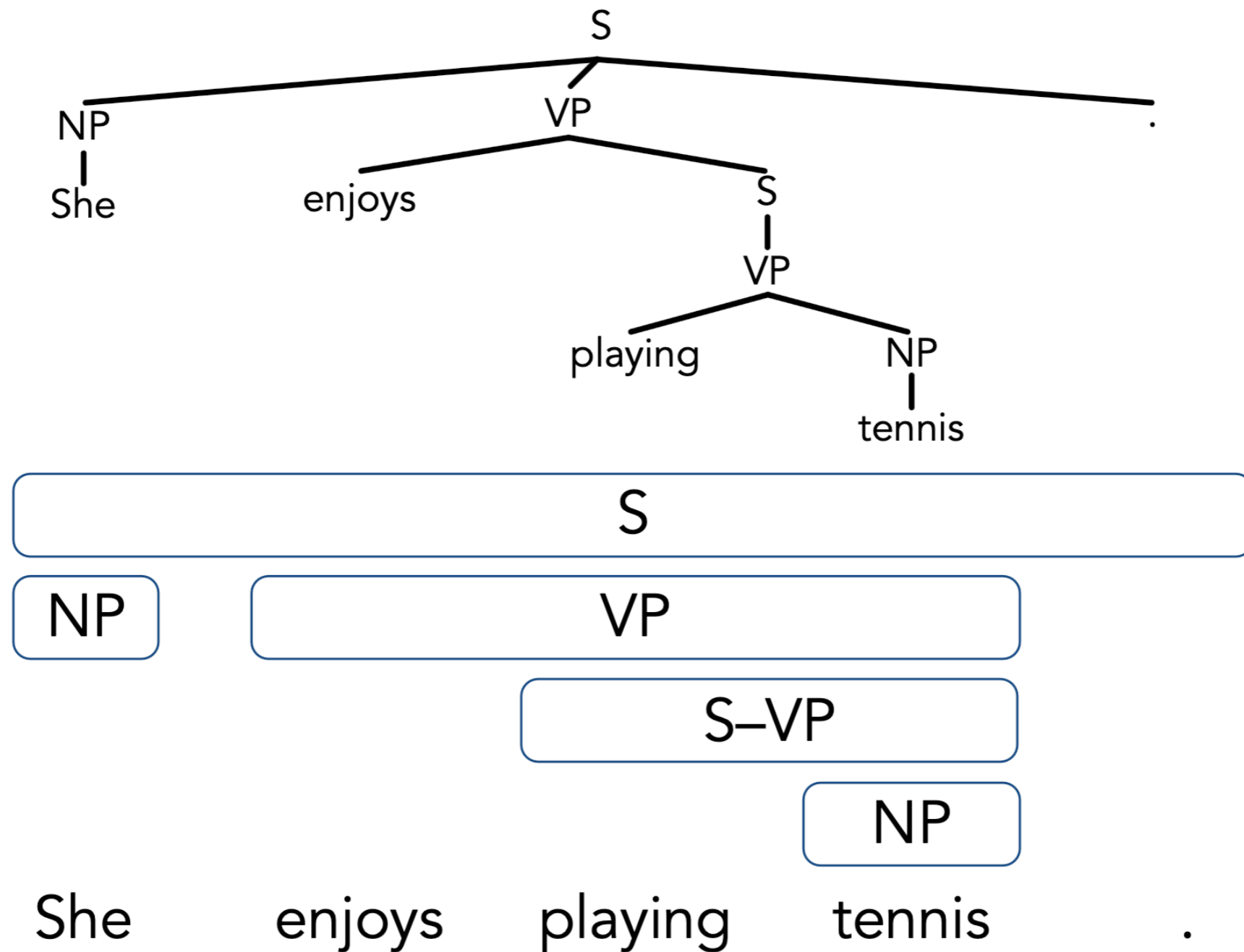
Span-Based Parsing

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

Scoring a span from the i-th word to j-th word being the label of ℓ
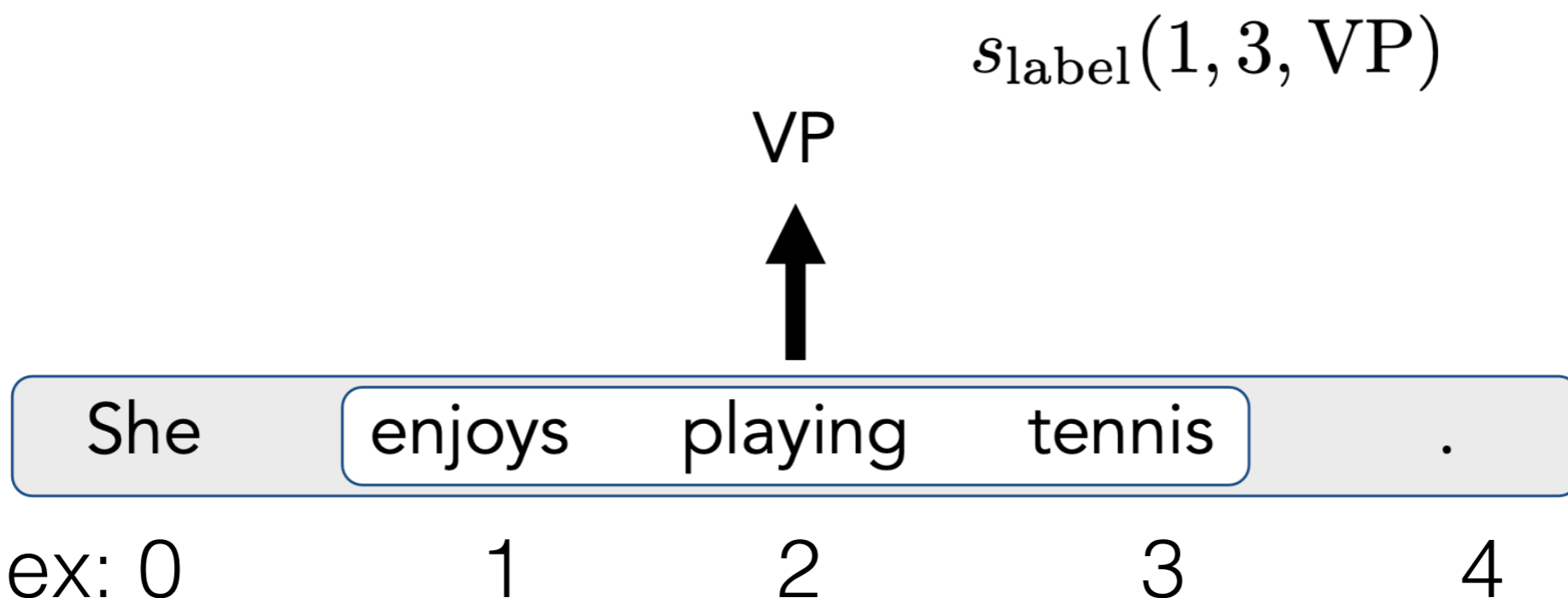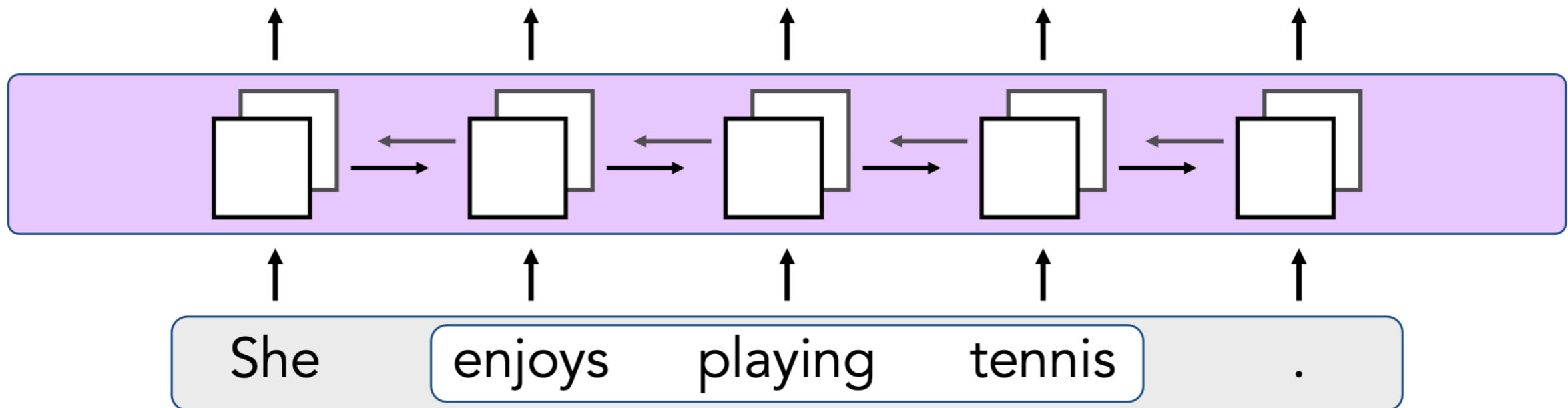
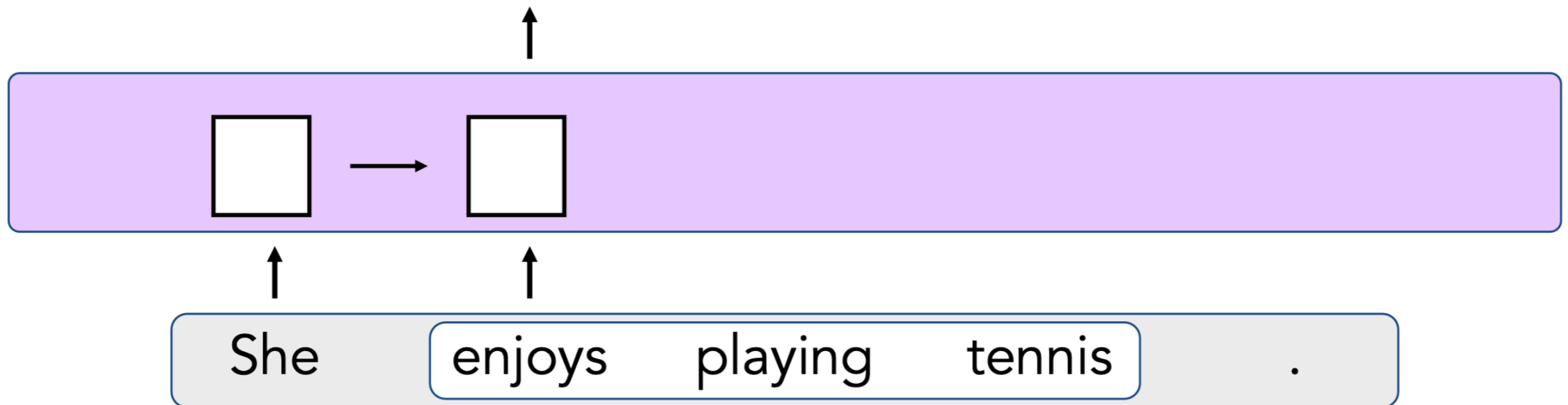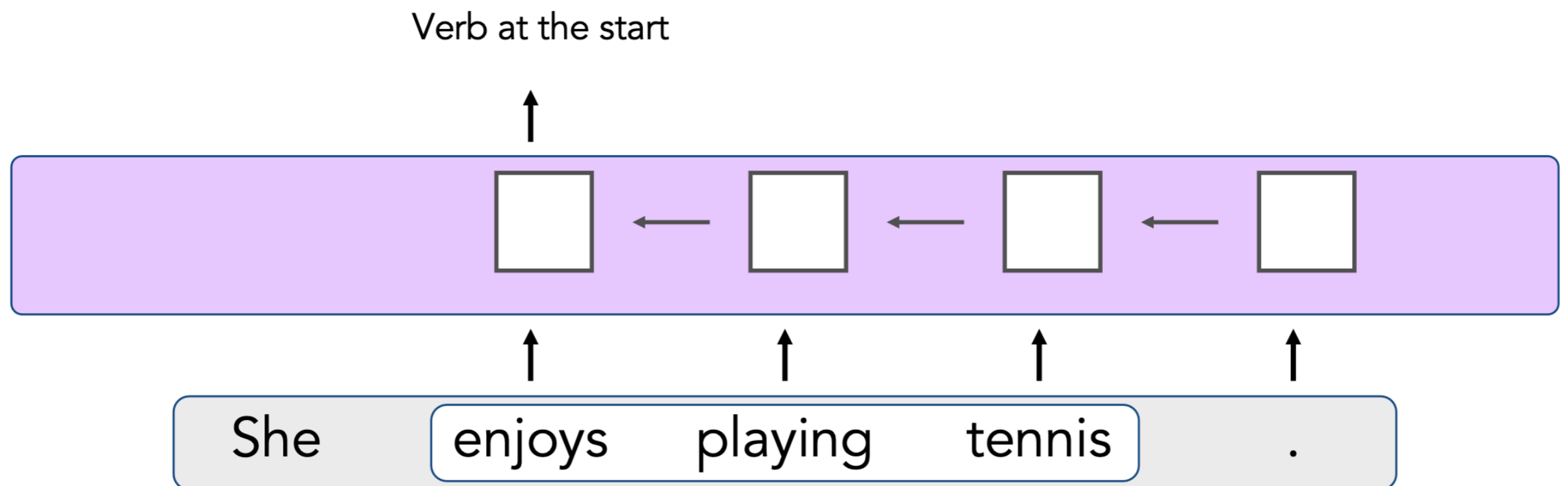$$s_{\text{label}}(1, 3, \text{VP})$$

VP

She | enjoys playing tennis | .

Word index: 0   1   2   3   4

Span-Based Parsing

Span-Based Parsing

Pronoun to the left

She enjoys playing tennis .

Span-Based Parsing

Span-Based Parsing

Pronoun to the left
Verb at the start

She | enjoys | playing | tennis | .

Span-Based Parsing

Pronoun to the left
Verb at the start

Period to the right
Noun and verbs to the left

She enjoys playing tennis .

Span-Based Parsing

Pronoun to the left
Verb inside
Period to the right

She enjoys playing tennis .

Span-Based Parsing

Span-Based Parsing

VP

She   enjoys   playing   tennis   .

Span-Based Parsing

Span-Based Parsing

Ø

↑

| She | enjoys | playing | tennis | . |

VP

↑

| She | enjoys | playing | tennis | . |
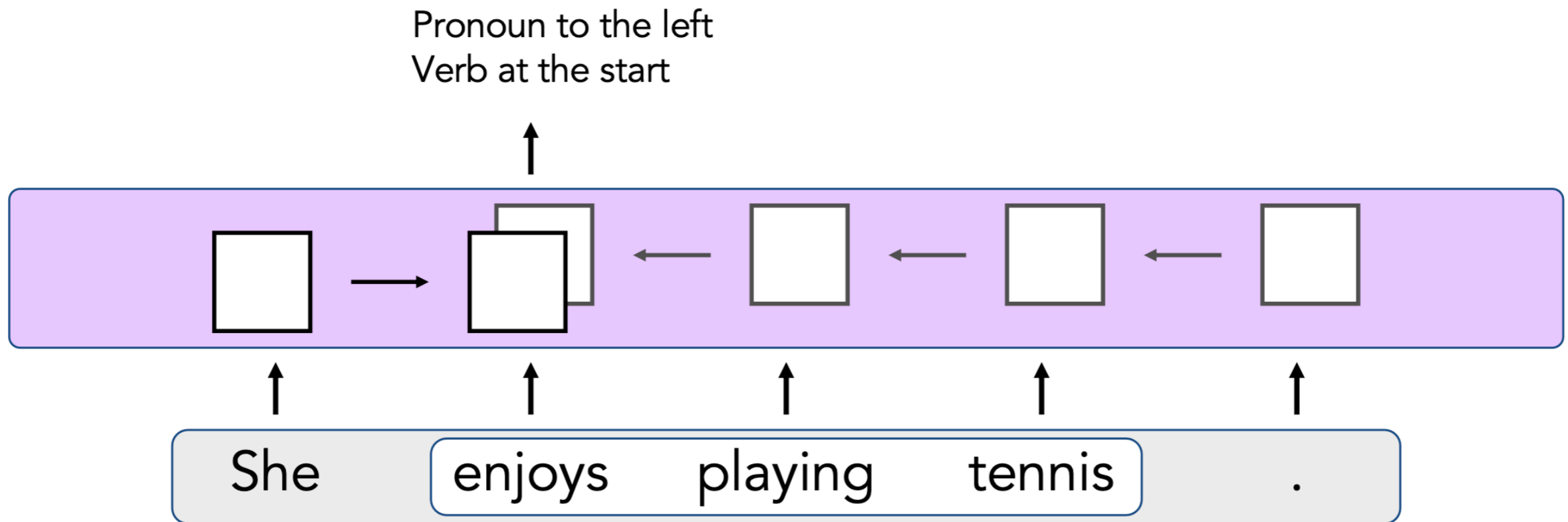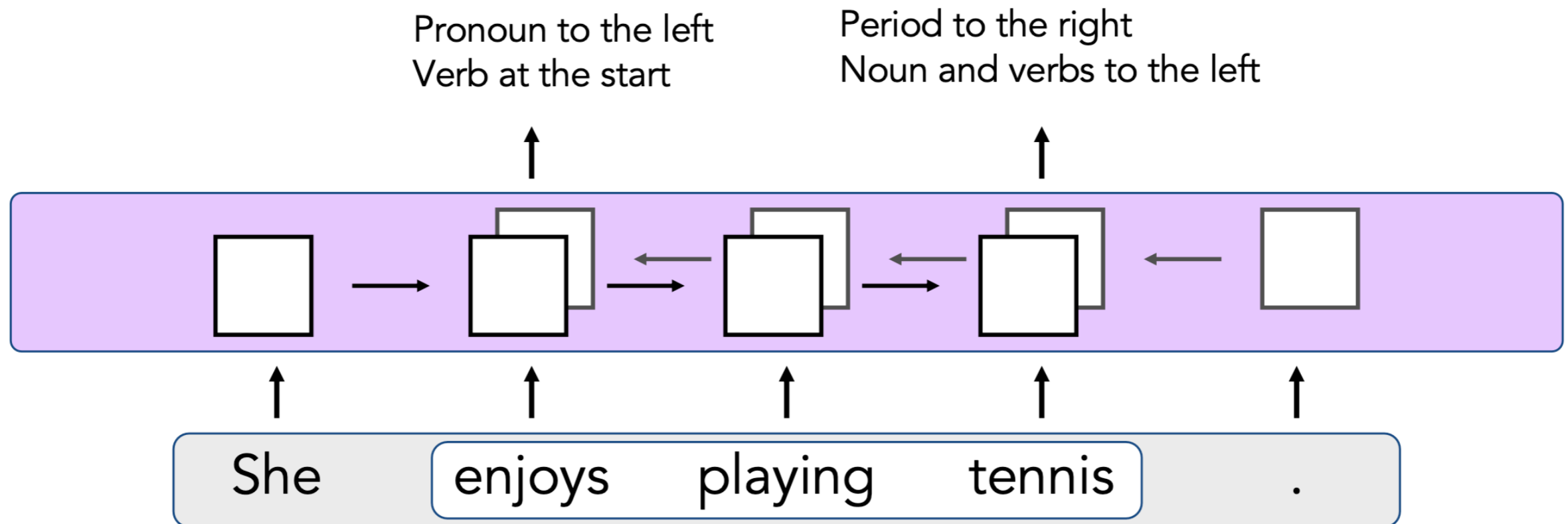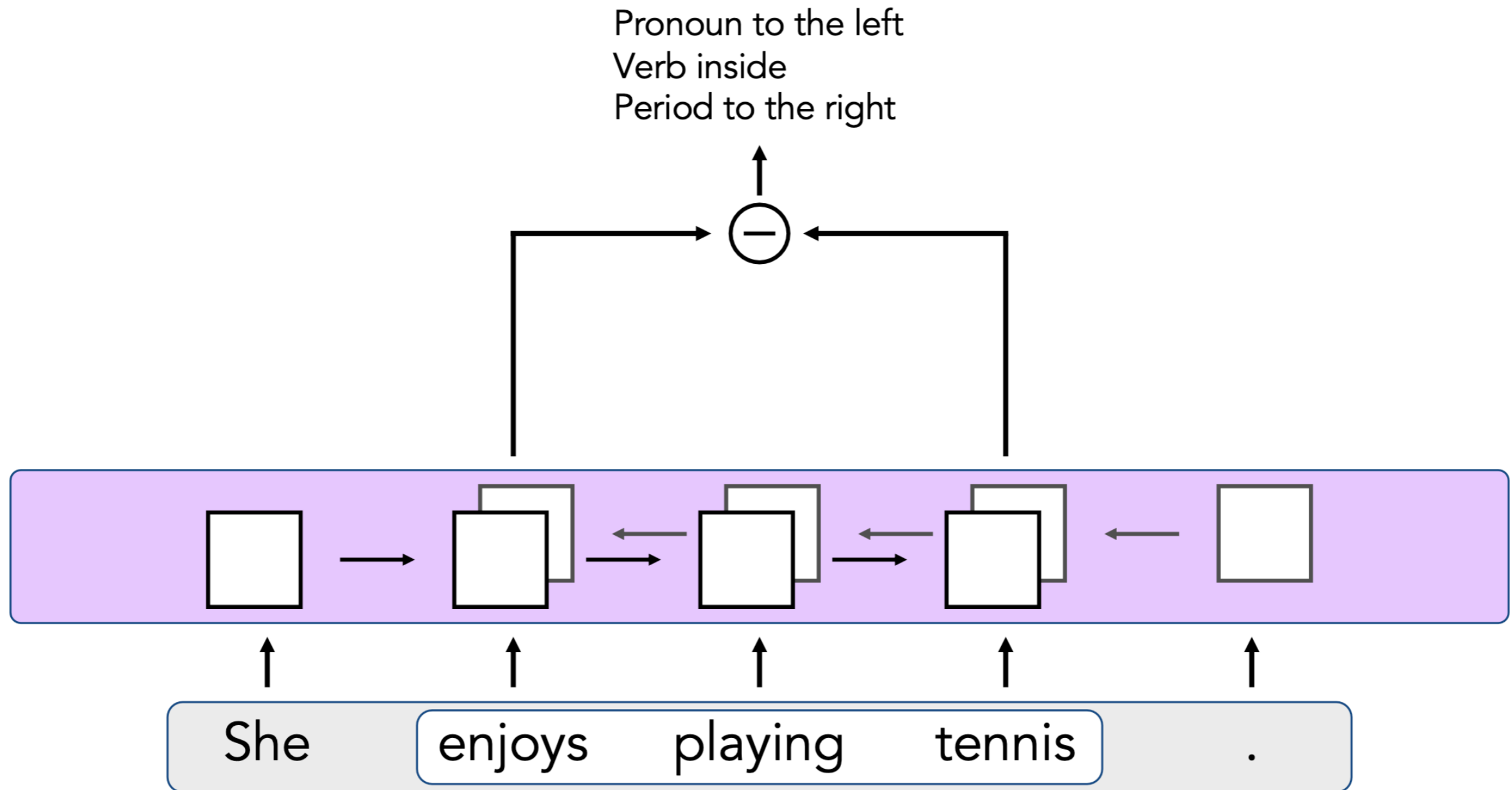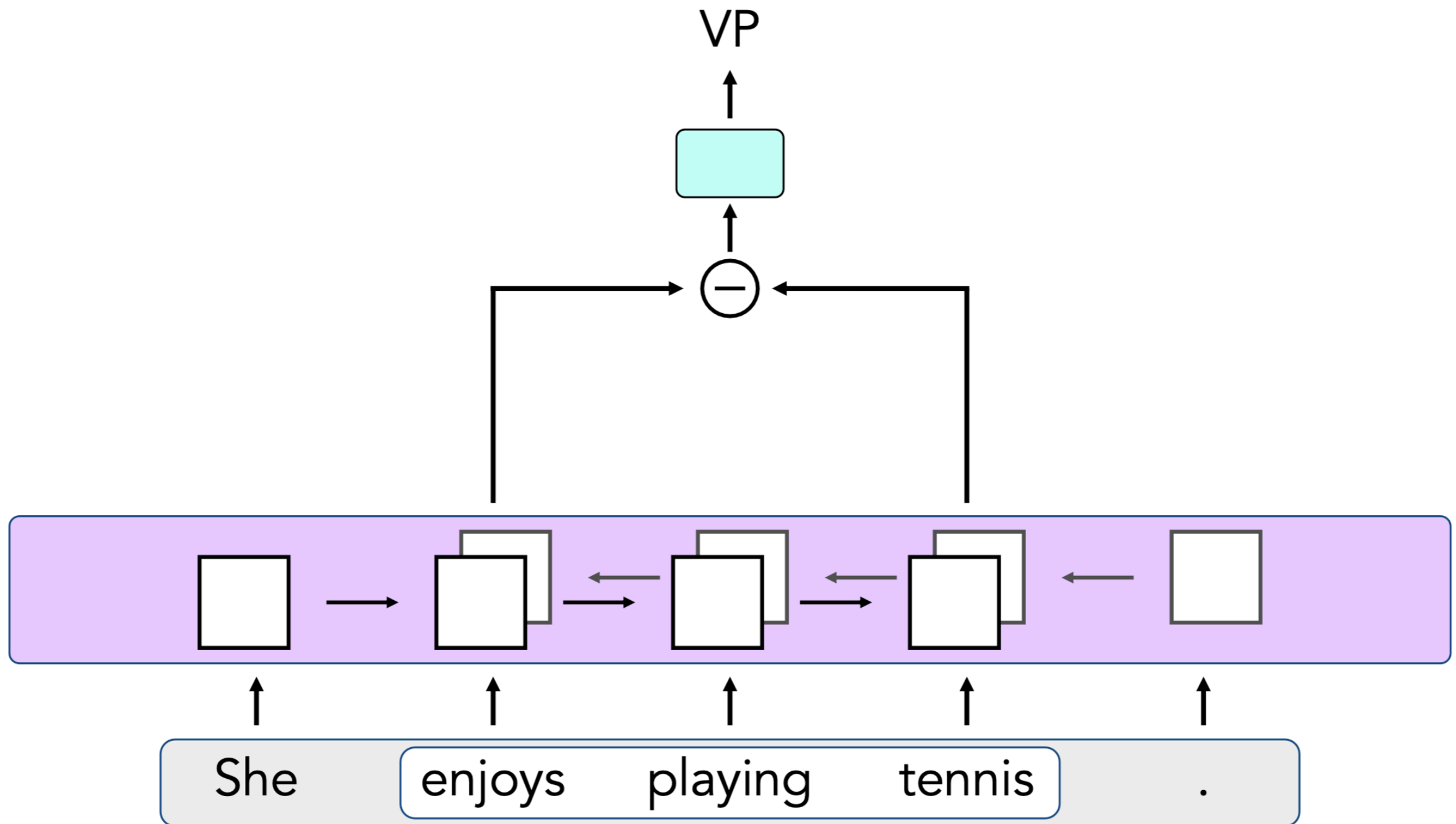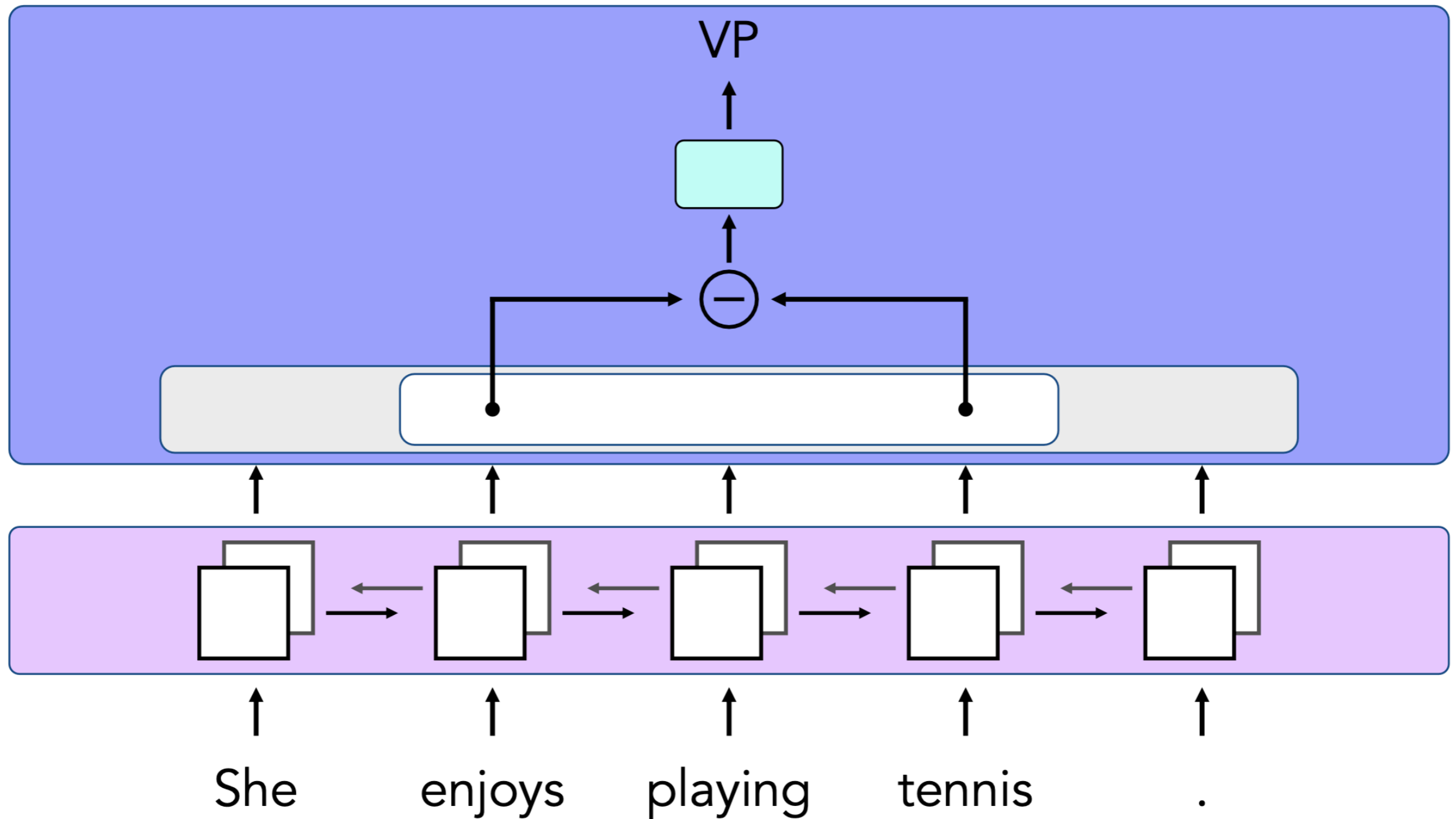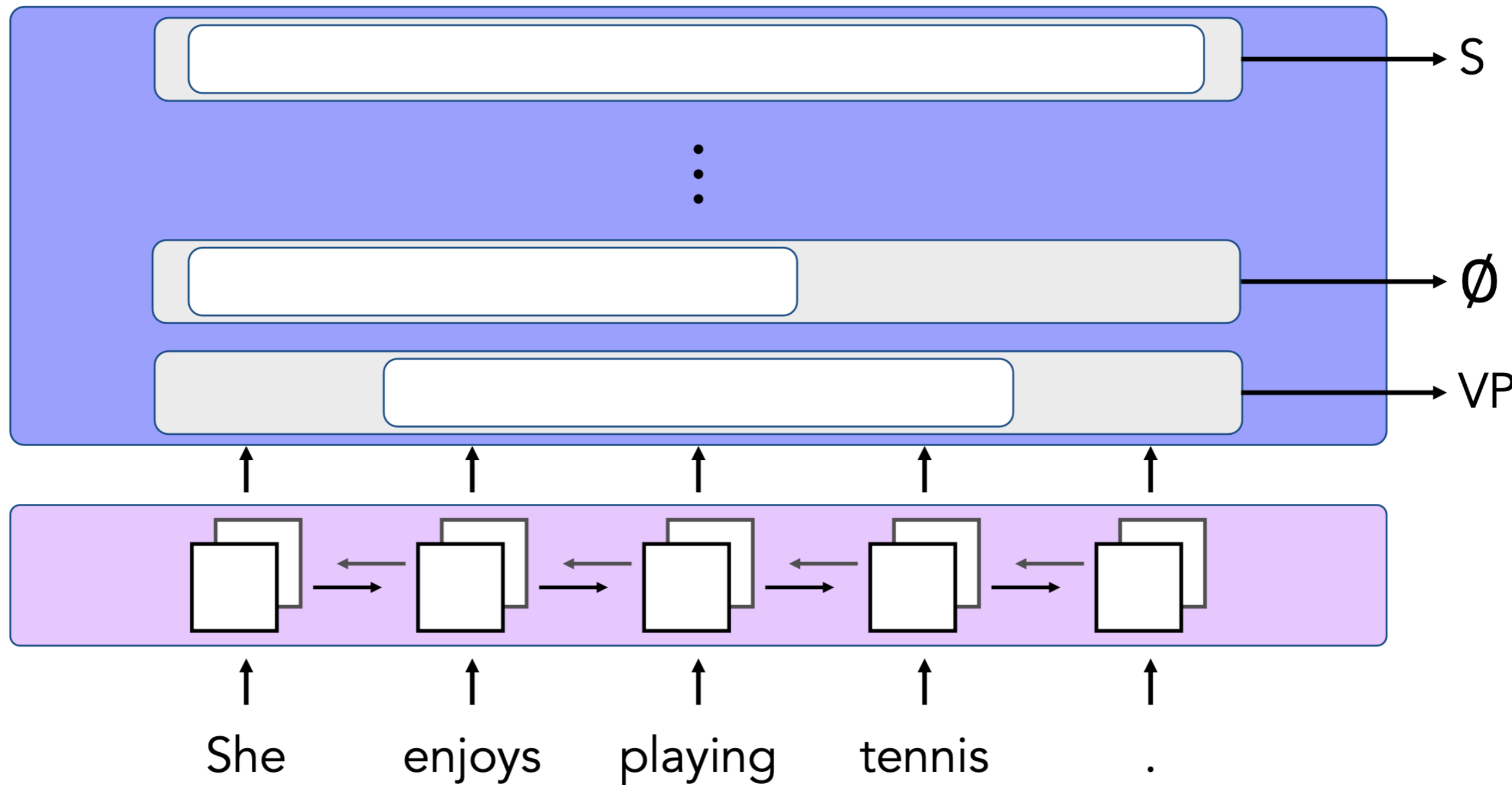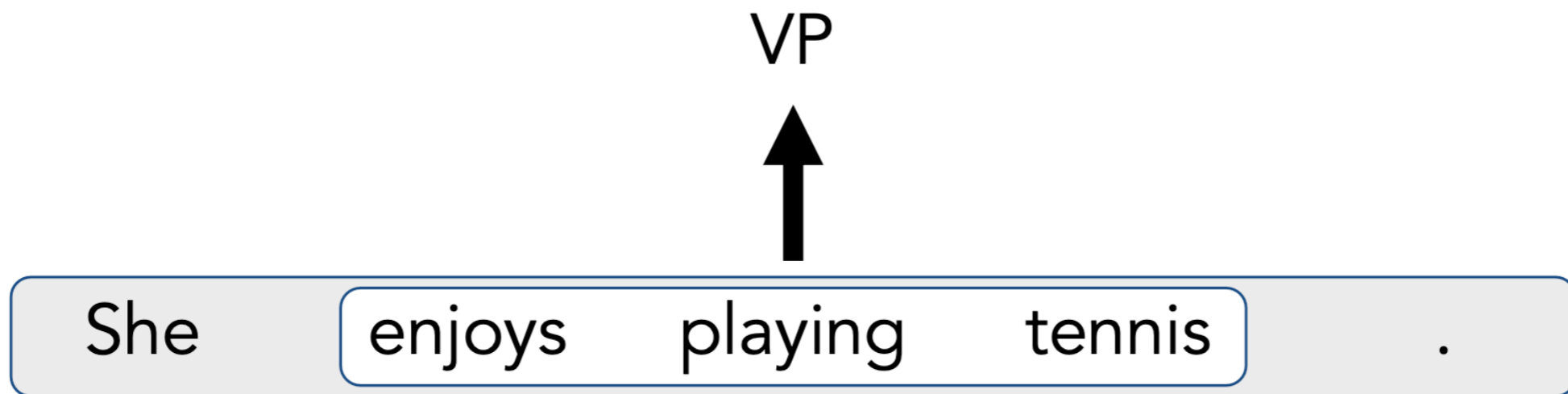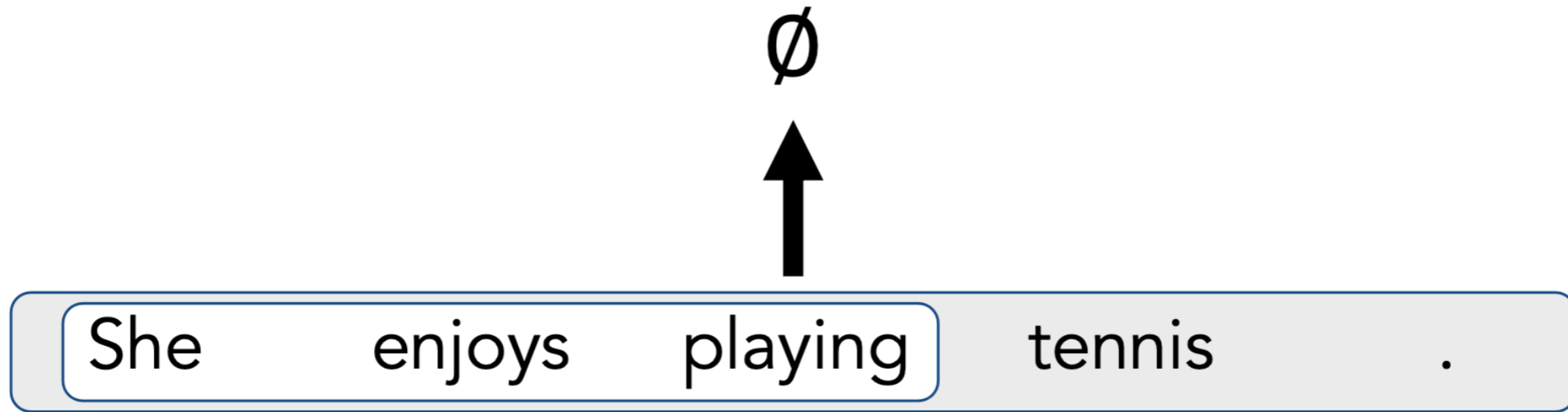
Training: Margin Loss

- Find the best tree using the current model

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

- Margin loss:

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

Decoding: CYK

- Same as counting-based PCFG

- Use the learned scores for possible spans in the following chart

She   enjoys   playing   tennis   .

0   1   2   3   4   5
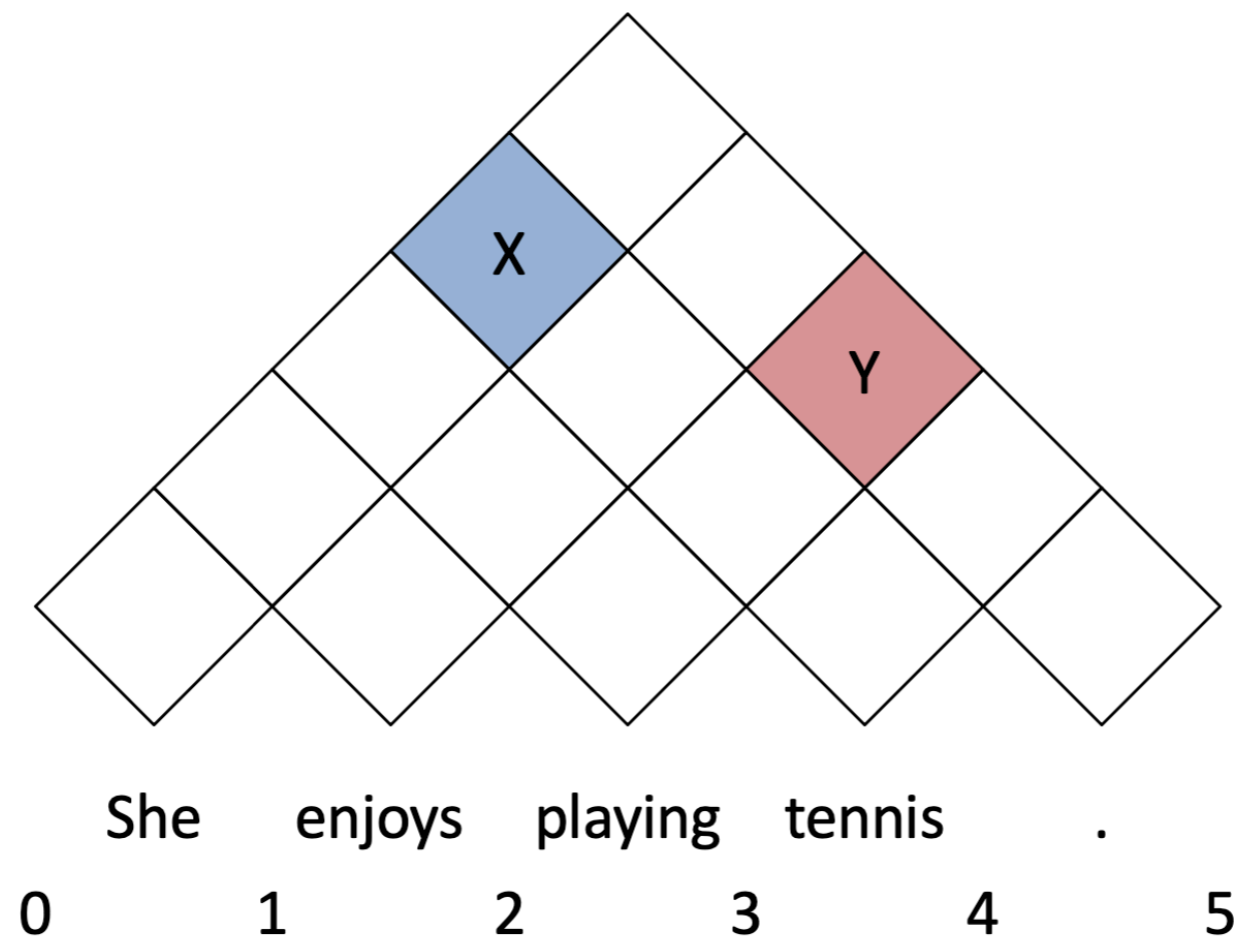
Improves over non-neural methods

Grammar-Based [Carreras et al, 08]: 91.0

LSTM-Based [Stern et al, 17]: 92.6

F1 (English, dev)
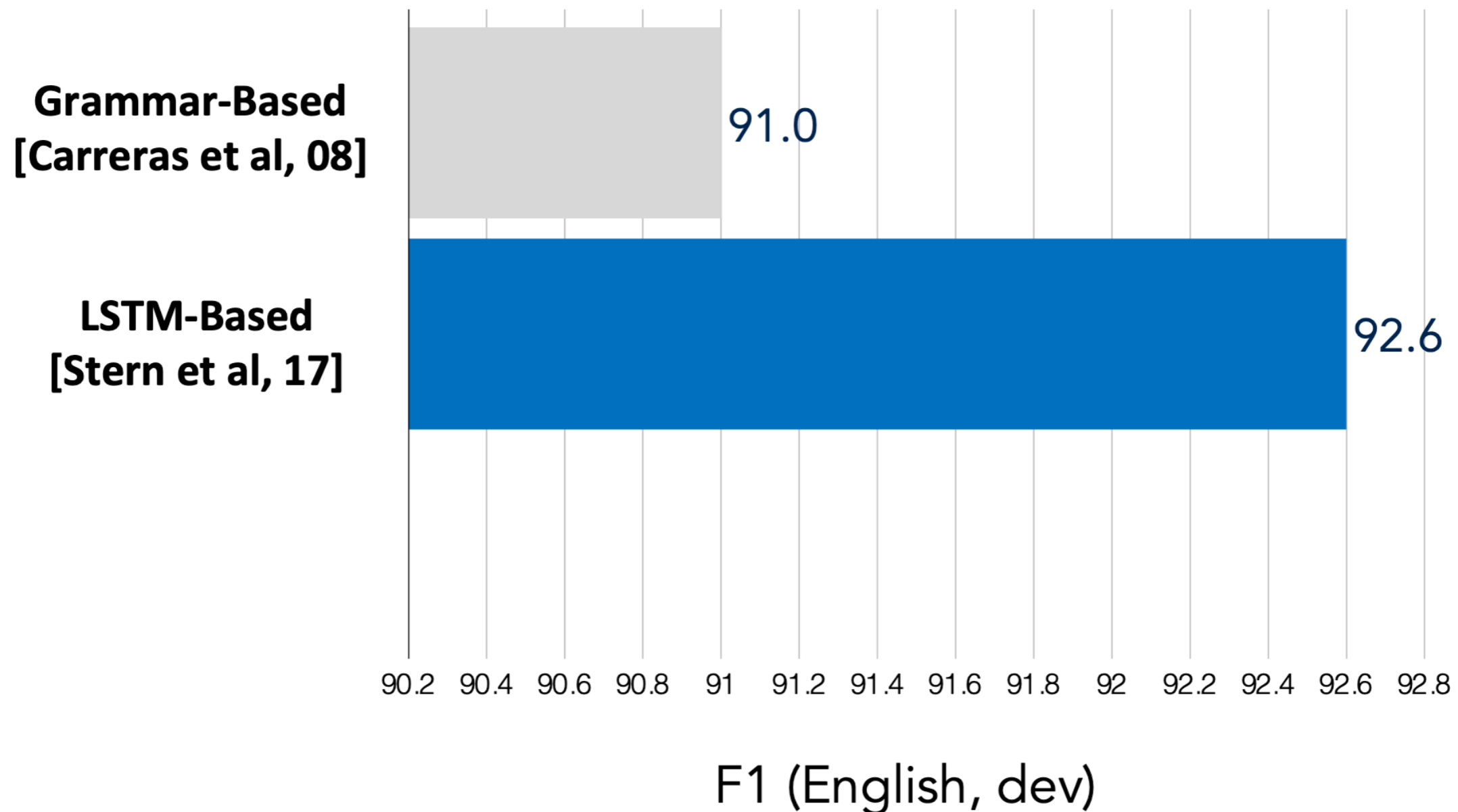
SoTA Self-attention-based Parser

- Use the Transformer encoder instead of Bi-LSTM

  - Split the word hidden vector from Transformer into two half vectors $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$

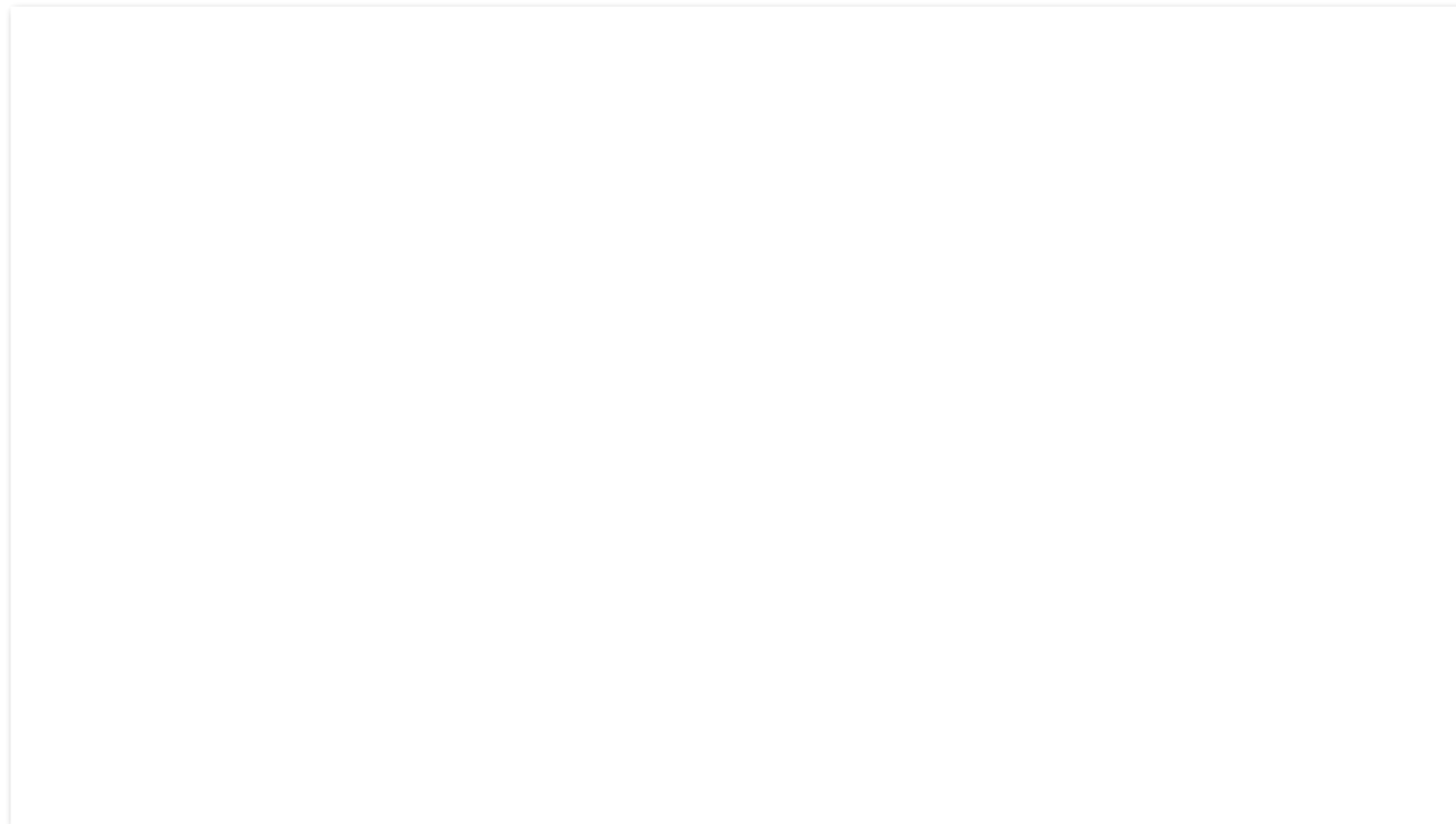 - Replace the forward and backward hidden vectors of Bi-LSTM by the new vectors

Kitaev et al. 2018. Constituency Parsing with a Self-Attentive Encoder
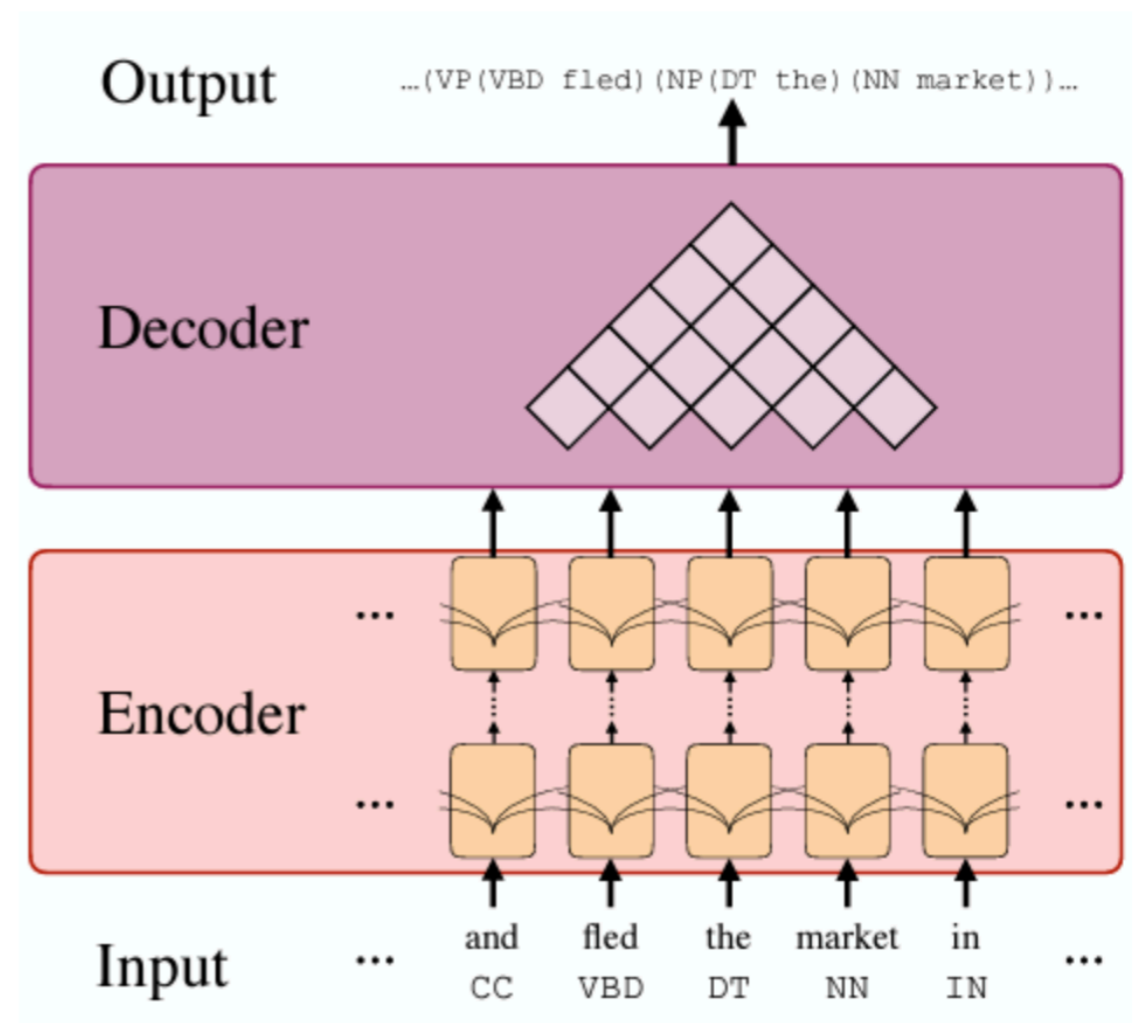
Historical Trends on Penn Treebank
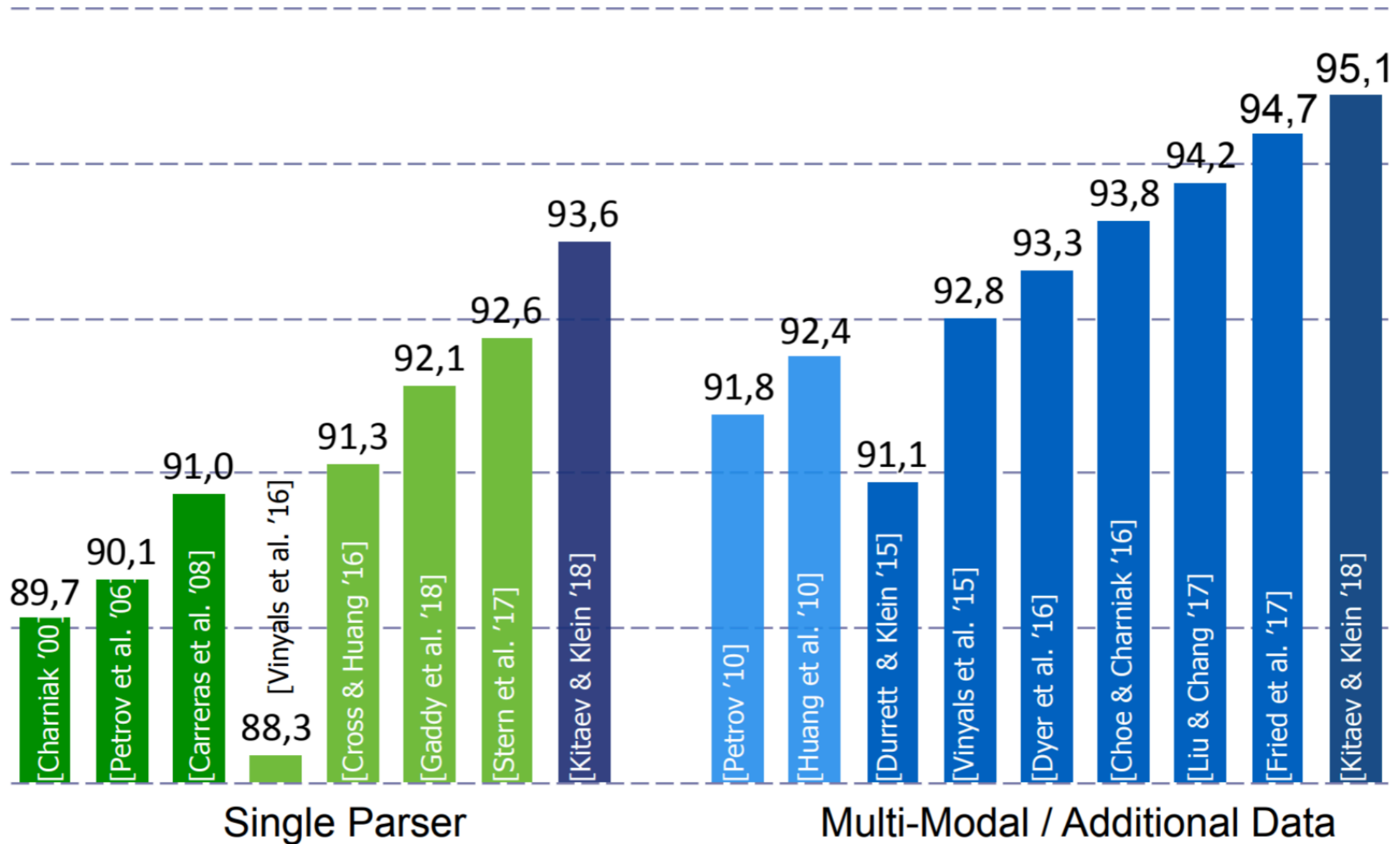
Unsupervised Parsing (Grammar Induction)
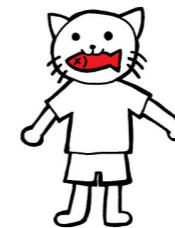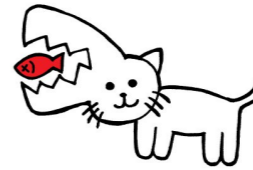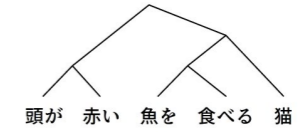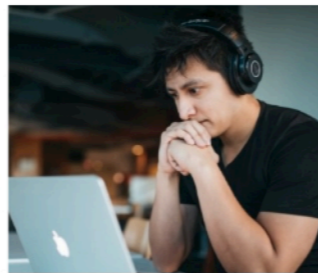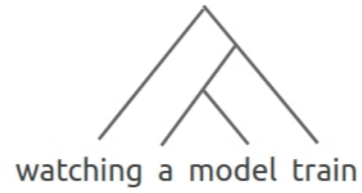
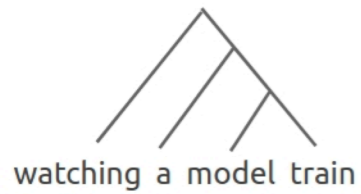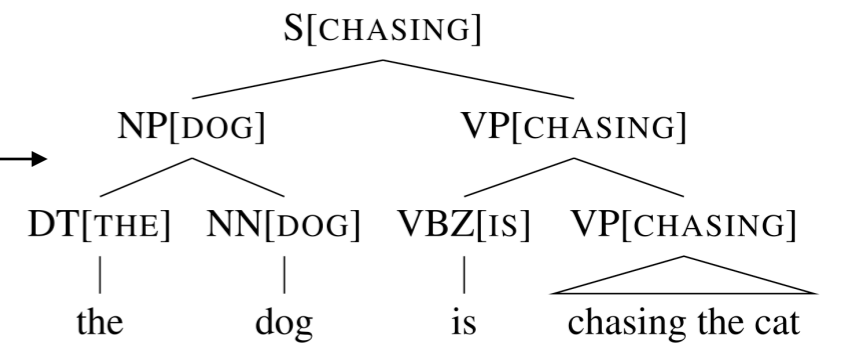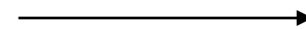# Grammar Induction (Unsupervised Parsing)
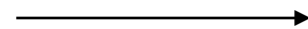
Learning a set of (probabilistic) grammar rules

Typical grammar induction methods
unsupervised constituency and dependency parsing

Explosion of ambiguity
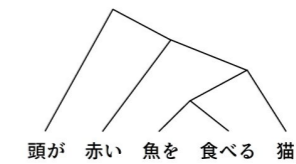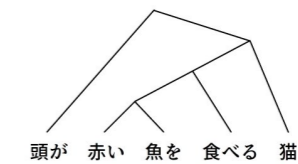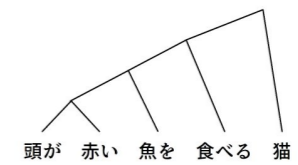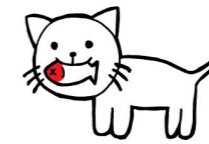
**Mystery: humans learn to parse
without learning to parse (from labels)**
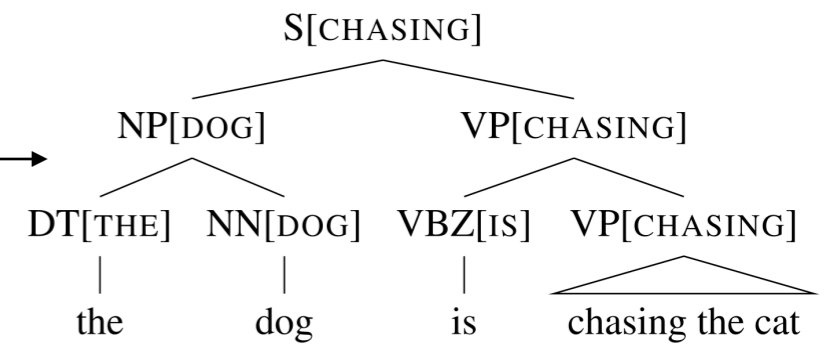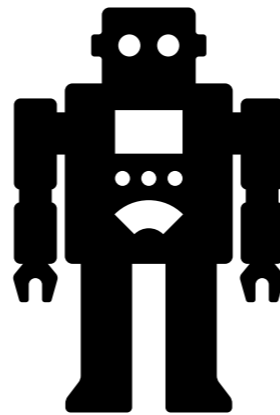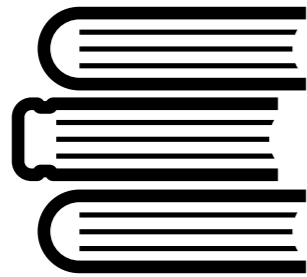
Grammar Induction

Neural PCFGs

- Neural parameterization for PCFGs

$$\pi_{T \to w} = \textsc{NeuralNet}(\mathbf{w}_T) = \frac{\exp(\mathbf{u}_w^\top f(\mathbf{w}_T))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_{w'}^\top f(\mathbf{w}_T))}$$

$$\pi_{T \to w} \propto \exp \Big(\underbrace{\mathbf{u}_w^\top}_{\text{output emb.}} \overbrace{f(\underbrace{\mathbf{w}_T}_{\text{input emb.}})}^{\text{shared neural net}} \Big)$$

- same training method

- Where's the magic?

Neural L-PCFGs

- You can further improve Neural PCFGs by adding head annotations

S[CHASING]
```
         S[CHASING]
        /          \
   NP[DOG]       VP[CHASING]
   /     \        /        \
DT[THE] NN[DOG] VBZ[IS]  VP[CHASING]
  |       |       |      /         \
 the     dog      is   chasing the cat
```

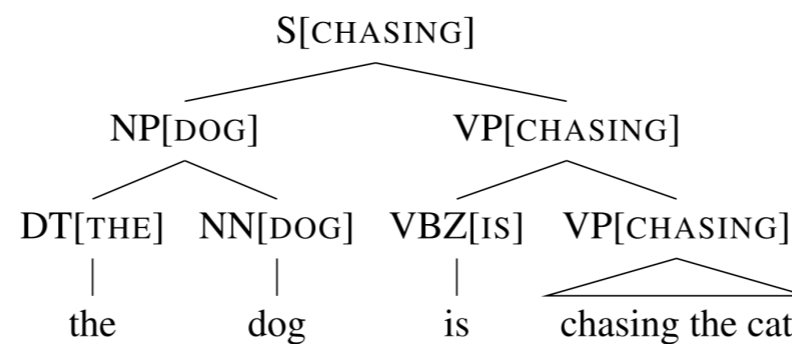$$S \to A, \qquad A \in \mathcal{N}$$
$$A \to BC, \quad A \in \mathcal{N}, B, C \in \mathcal{N} \cup \mathcal{P}$$
$$T \to \alpha, \qquad T \in \mathcal{P}$$

$$(1) \quad S \to A[\alpha], \qquad A \in \mathcal{N}$$
$$(2l) \quad A[\alpha] \to B[\alpha]C[\beta], \quad A \in \mathcal{N}, B, C \in \mathcal{N} \cup \mathcal{P}$$
$$(2r) \quad A[\alpha] \to B[\beta]C[\alpha], \quad A \in \mathcal{N}, B, C \in \mathcal{N} \cup \mathcal{P}$$
$$(3) \quad T[\alpha] \to \alpha, \qquad T \in \mathcal{P}$$

Results

Legend: Compound PCFG, DMV, Neural L-PCFGs

right branching

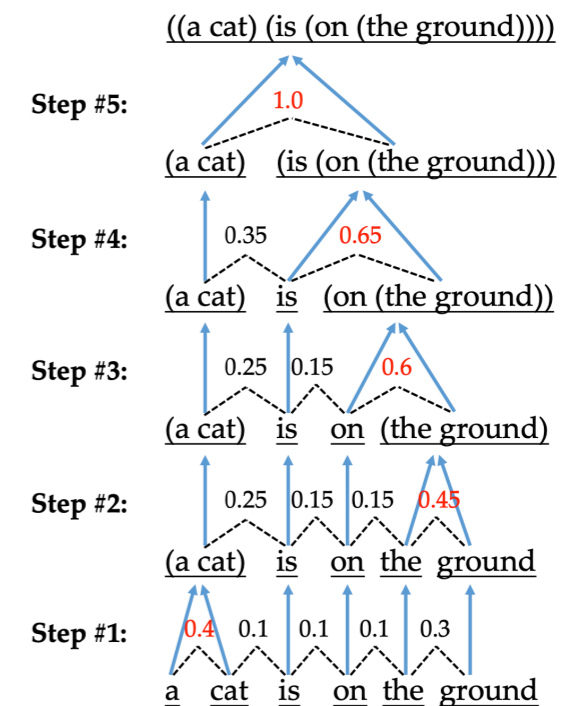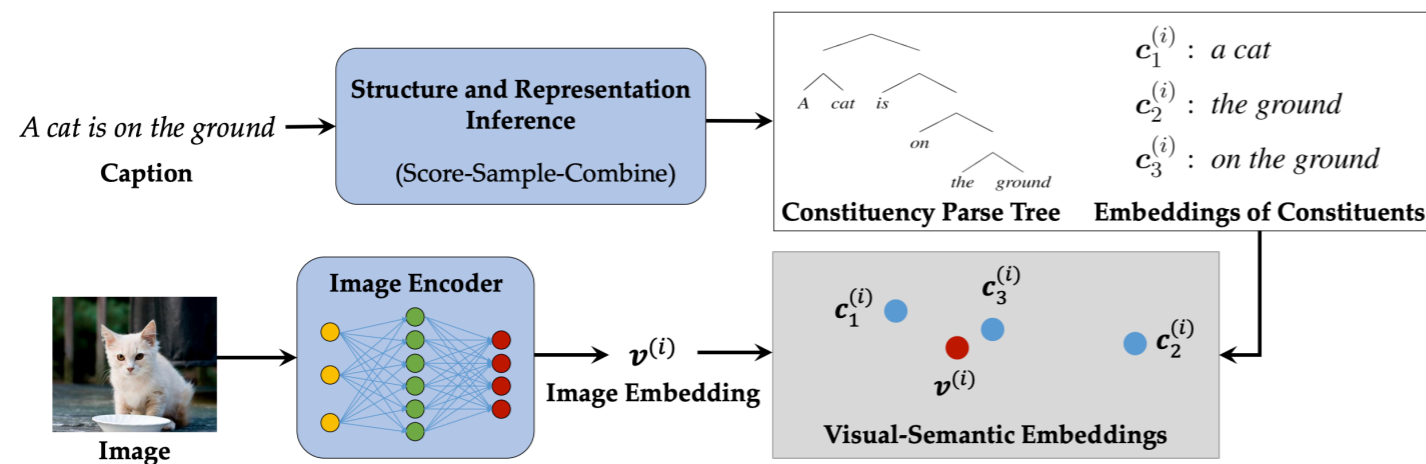Categories: DAS, UAS, F1

Y-axis: 0, 15, 30, 45, 60
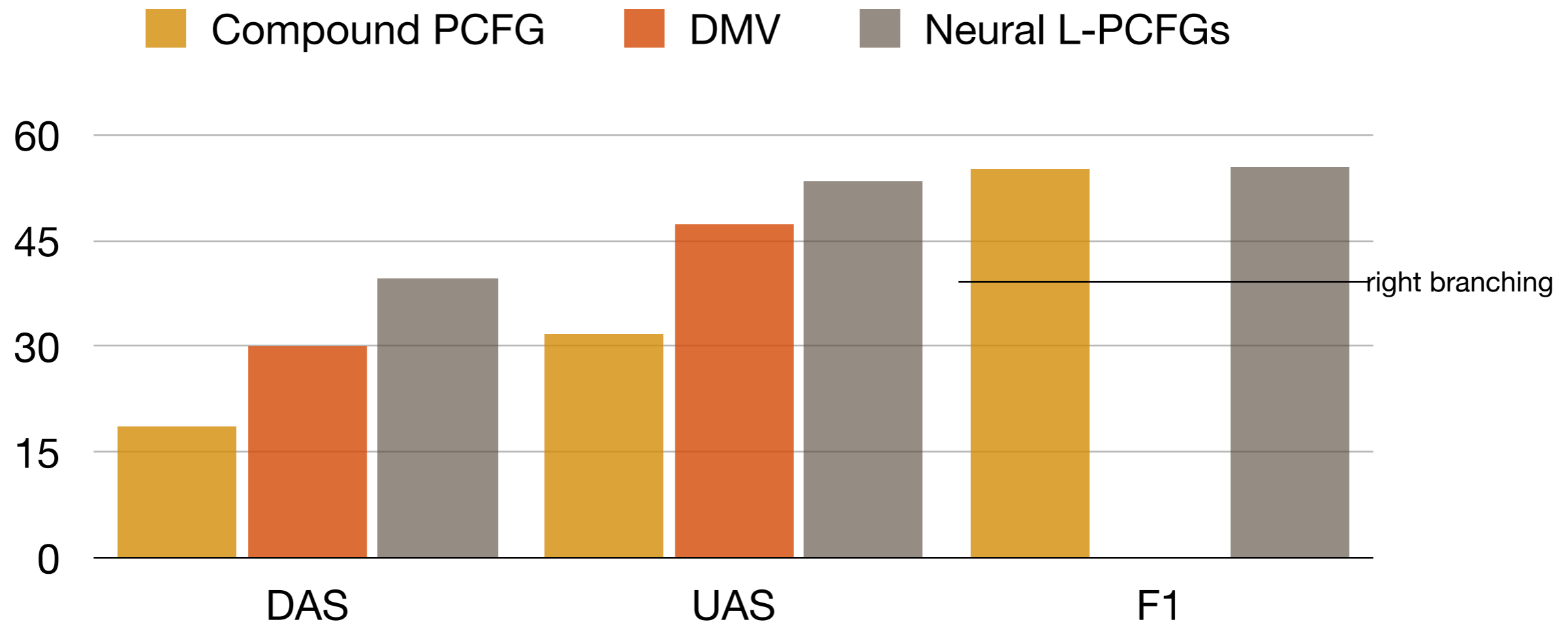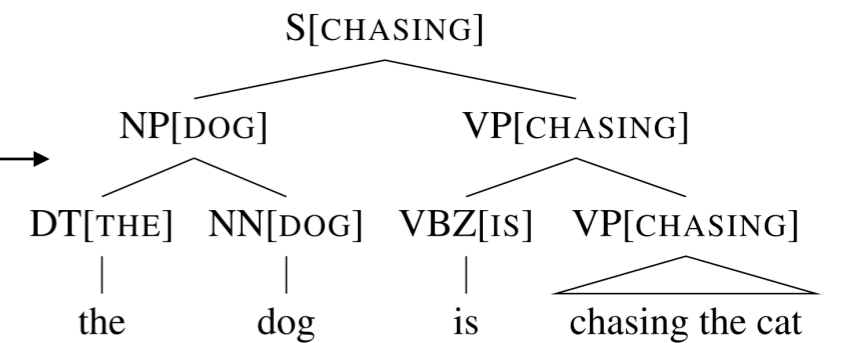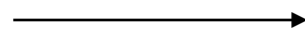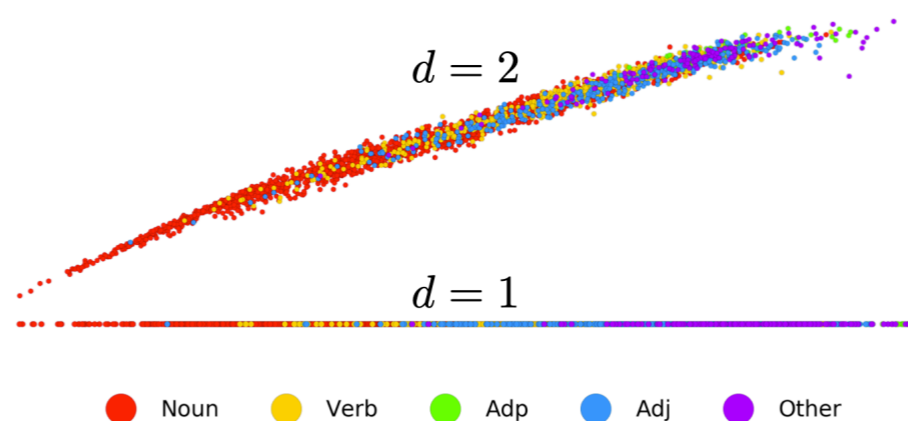
Key to the mystery: visual prior?

Visual Prior Grammar Induction

- Visual grounded neural syntax acquisition

Visual Prior Grammar Induction

- Visual grounded neural syntax acquisition

  - Similar results even if the dimension of embeddings get shrunk to 1 or 2.

  - embedding tags

  - concretene

Model	NP	VP	PP	ADJP	Avg. F_1
Shi2019	79.6	26.2	42.0	22.0	50.4 ± 0.3
Shi2019*	80.5	26.9	45.0	21.3	51.4 ± 1.1
$1, s_{WS}, c_{ME}$	77.2	17.0	53.4	18.2	49.7 ± 5.9
$2, s_{WS}, c_{ME}$	**80.8**	19.1	52.3	17.1	51.6 ± 0.6
+HI					
Shi2019	74.6	32.5	66.5	21.7	53.3 ± 0.2
Shi2019*	73.1	33.9	64.5	22.5	51.8 ± 0.3
$1, s_{WS}, c_{ME}$	74.0	35.2	62.0	24.2	51.8 ± 0.4
$2, s_{WS}, c_{ME}$	73.8	30.2	63.7	21.9	51.3 ± 0.4
+HI+FastText					
Shi2019	78.8	24.4	65.6	22.0	54.4 ± 0.3
Shi2019*	77.3	23.9	64.3	21.9	53.3 ± 0.1
$1, s_{WS}, c_{ME}$	76.6	21.9	68.7	20.6	53.5 ± 1.4
$2, s_{WS}, c_{ME}$	77.5	22.8	66.3	19.3	53.6 ± 0.2
+HI+FastText-IN					
Shi2019*	78.3	26.6	67.5	22.1	54.9 ± 0.1
$1, s_M, c_{MX}$	79.6	29.0	38.3	23.5	49.7 ± 0.2
$1, s_{MHI}, c_{MX}$	77.6	**45.0**	72.3	**24.3**	**57.5 $\pm$ 0.1**
$1, s_M, c_{ME}$	80.0	26.9	62.2	23.2	54.3 ± 0.2
$1, s_{MHI}, c_{ME}$	76.5	20.5	63.6	22.7	52.2 ± 0.3
$1, s_{WS}, c_{ME}$	77.7	26.3	**72.5**	22.0	55.5 ± 0.1
$2, s_{WS}, c_{ME}$	78.5	26.3	69.5	21.1	55.2 ± 0.1

Visual Prior Grammar Induction

- Recommend readings

  - *Visually Grounded Compound PCFGs.*

  - *Dependency Induction Through the Lens of Visual Perception*

Questions?